合宙 Air780EP 模组使用手册

版本号：V1.0.7

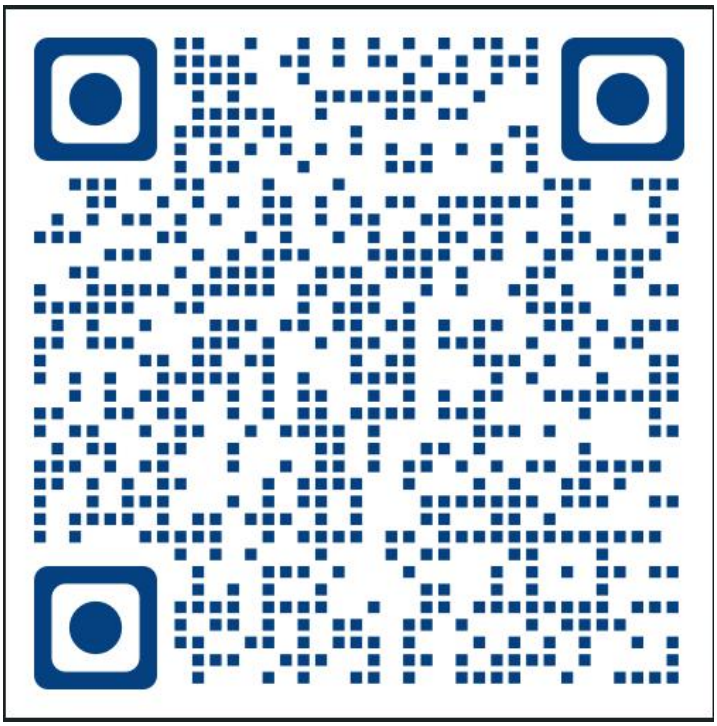
发布日期：2025.01.13

联系我们

如需任何帮助，请通过以下方式随时联系上海合宙通信科技有限公司：

1、 可以点击右方链接获取联系方式： 联系我们

2、 也可以扫描下方二维码获取联系方式：



声明

合宙通信提供该文档内容以支持客户的产品设计。客户必须按照文档中提供的规范、参数来设计产品。同时，您理解并同意，合宙通信提供的参考设计仅作为示例。您同意在设计您目标产品时使用您独立的分析、评估和判断。在使用本文档所指导的任何硬软件或服务之前，请仔细阅读本声明。您在此承认并同意，合宙通信可在未事先通知的情况下，自行决定随时增加、修改或重述本文档。

免责声明如下：

1)合宙通信不承担任何因未能遵守有关操作或设计规范而造成损害的责任。

2)合宙通信不承担因本文档中的任何因不准确、遗漏、或使用本文档中的信息而产生的任何责任。

3)合宙通信尽力确保开发中功能的完整性、准确性、及时性，但不排除上述功能错误或遗漏的可能。除非另有协议规定，否则合宙通信对开发中功能的使用不做任何暗示或法定的保证。在适用法律允许的最大范围内，合宙通信不对任何因使用开发中功能而遭受的损害承担责任，无论此类损害是否可以预见。

4)合宙通信对第三方网站及第三方资源的信息、内容、广告、商业报价、产品、服务和材料的可访问性、安全性、准确性、可用性、合法性和完整性不承担任何法律责任。

本文档版权属于上海合宙通信科技有限公司所有，保留一切权利。

修改记录

版本	日期	作者	修改描述
V1.0.0	2024-05-06	朱天华、沈园园	第一版
V1.0.1	2024-05-10	朱天华、沈园园	产品手册更新为最新版本
	2024-05-11	朱天华、沈园园	更换 C 语言内容
	2024-05-13	朱天华、沈园园	更改其它细节问题
V1.0.2	2024-05-16	朱天华、沈园园	添加本手册阅读说明
	2024-05-17	朱天华、沈园园	修改公司介绍
	2024-05-17	朱天华、沈园园	添加合宙功耗分析仪简要说明
	2024-05-17	朱天华、沈园园	更改其它细节问题
V1.0.3	2024-05-20	朱天华、沈园园	添加 CSDK API 接口
	2024-05-22	朱天华、沈园园	添加 Linux/Android 应用指南
V1.0.4	2024-05-27	朱天华、沈园园	添加云平台章节内容
	2024-05-31	朱天华、沈园园	补充部分 CSDK API 接口说明
V1.0.5	2024-06-28	沈园园	添加 LustOS 开发应用指南文档
V1.0.6	2024-07-03	沈园园	修改 io - io 操作(扩展)file:seek 示例错误
V1.0.7	2025-01-13	沈园园	去掉不支持功能

后续更新计划

章节	内容
第七章	开发板使用说明
第十一章	项目实战

目录

合宙 Air780EP 模组使用手册 1
联系我们 2
声明 3
修改记录 4
后续更新计划 5
一 本手册阅读说明 25
1.1 刚接触合宙新朋友 25
1.2 熟悉合宙的老朋友 25
1.2.1 AT 开发方式 26
1.2.2 LuatOS 开发方式 26
二 公司介绍 27
三 公司产品选型手册 27
四 产品规格书 27
4.1 功能特性 27
4.2 管脚排列图 29
4.3 电气参数 30
4.4 实网功耗测试报告 31
4.5 高低温射频传导测试报告 31
4.6 吞吐量测试报告 32
4.7 证书资料 33
4.7.1 CCC 证书 33
4.7.2 入网许可证书 35
4.7.3 SRRC 证书 36
4.7.4 ROHS 证书 38
4.8 软件开发方式 39
4.9 技术支持服务 39
4.10 模块生产测试记录 39
五 硬件设计指南 39
5.1 绪论 39
5.2 综述 40
5.2.1 模块主要性能 40
5.3 应用接口 42

5.3.1 管脚描述 42
5.3.2 工作模式 49
5.3.3 电源供电 51
5.3.4 开关机 53
5.3.5 串口 57
5.3.6 USB 接口 61
5.3.7 USB 下载模式 62
5.3.8 I2C 63
5.3.9 SIM 卡接口 64
5.3.10 LDO 输出 67
5.3.11 功能管脚 68
5.3.12 省电功能 70
5.3.13 模式切换汇总 72
5.4 射频接口 72
5.4.1 射频参考电路 73
5.4.2 RF 输出功率 73
5.4.3 RF 传导灵敏度 73
5.4.4 工作频率 74
5.4.5 推荐 RF 焊接方式 74
5.5 电器特性，可靠性，射频特性 75
5.5.1 绝对最大值 75
5.5.2 推荐工作条件 75
5.5.3 工作温度 76
5.5.4 功耗 76
5.5.5 静电防护 79
5.6 模块尺寸图 79
5.6.1 推荐 PCB 封装 79
5.7 存储和生产 81
5.7.1 存储 81
5.7.2 生产焊接 81
5.8 术语缩写 82
六 快速入门 84
6.1 概述 84
6.2 硬件环境 84
6.3 软件环境 84

6.3.1 USB 驱动 84
6.3.2 Luatools 下载调试工具 84
6.3.3 AT 软件开发资料 84
6.3.4 LuatOS 软件开发资料 84
6.4 开发板使用说明 84
6.4.1 硬件配置 84
6.4.2 管脚定义 85
6.4.3 管脚功能说明 85
6.4.4 顶视图 88
6.4.5 底视图 89
6.4.6 供电 89
6.4.7 开关机 90
天线 90
固件升级 91
6.5 软件示例 92
6.5.1 LuatOS 示例 92
七 开发板使用说明（后续补充） 96
八 工具使用指南 96
8.1 USB 驱动 96
驱动方式 96
虚拟串口 96
AT 固件中虚拟串口的作用 98
LuatOS 固件中虚拟串口的作用 98
8.2 Luatools 下载调试工具 99
Luatools 下载 99
Luatools 安装 99
Luatools 核心功能 101
Luatools 工具目录介绍 103
Luatools 烧录 Air780EP 固件 103
8.3 EPAT 底层日志抓取调试工具 104
8.3.1 EPAT 简介 104
8.3.2 安装 104
8.3.3 Online Mode – 联机捕获 Log 105
8.3.4 Offline Mode – 查看 Log 111
8.3.5 Message Filter 功能 112

8.3.6 SigLog 114
8.3.7 RamDump 功能 115
8.4 量产多路下载工具 116
工具下载 116
使用手册 116
USB 下载 117
串口下载 120
注意事项 121
九 软件开发指南（AT） 121
9.1 AT 开发简介 121
AT 命令语法介绍 121
AT 开发基本架构 123
AT 开发优缺点 123
9.2 AT 固件 123
9.3 AT 命令手册 123
9.4 AT 开发环境搭建 124
AT 命令端口 124
AT 命令发送 124
9.5 AT 开发应用指南 125
9.5.1 TCP(SSL)应用指南 125
9.5.2 HTTP(S)应用指南 139
9.5.3 MQTT 应用指南 152
9.5.4 FTP 应用指南 160
9.5.5 远程升级(FOTA)应用指南 166
9.5.6 网络状态与信号应用指南 174
9.5.7 基站/WIFI 定位应用指南 185
9.5.8 文件系统应用指南 188
9.5.9 阿里云应用指南 190
9.5.10 ONENET（中国移动物联网平台）应用指南 202
9.5.11 CTWing（中国电信物联网平台）应用指南 212
9.5.12 DMP（中国联通雁飞格物物联网平台）应用指南 223
9.5.13 短消息应用指南 233
9.5.14 休眠应用指南 237
十 软件开发指南（LuatOS） 240
10.1 Lua 语言学习 240
10.2 LuatOS 设计框架 241

10.3 LuatOS 软件包 241
内核固件 241
应用脚本 demo 241
10.4 LuatOS API 242
10.5 LuatOS 开发应用指南 242
10.5.1 FOTA 开发应用指南 242
10.5.2 SOCKET 开发应用指南 253
10.5.3 HTTP 开发应用指南 261
10.5.4 HTTPDNS 开发应用指南 264
10.5.5 MQTT 开发应用指南 266
10.5.6 FTP 开发应用指南 273
10.5.7 阿里云开发应用指南 278
10.5.8 onenet 开发应用指南 291
10.5.9 mobile 开发应用指南 301
10.5.10 meminfo 开发应用指南 306
10.5.11 lbsLoc 开发应用指南 309
10.5.12 os_date_time 开发应用指南 312
10.5.13 rtc 开发应用指南 315
10.5.14 sntp 开发应用指南 317
10.5.15 crypto 开发应用指南 320
10.5.16 rsa 开发应用指南 327
10.5.17 fskv 开发应用指南 330
10.5.18 fs 开发应用指南 334
10.5.19 miniz 开发应用指南 341
10.5.20 errDump 开发应用指南 343
10.5.21 zbuff 开发应用指南 346
10.5.22 hmeta 开发应用指南 351
10.5.23 json 开发应用指南 353
10.5.24 gpio 开发应用指南 357
10.5.25 adc 开发应用指南 364
10.5.26 netled 开发应用指南 367
10.5.27 uart 开发应用指南 370
10.5.28 pwm 开发应用指南 375
10.5.29 string 开发应用指南 378
10.5.30 sim_det 开发应用指南 382
10.5.31 usb_uart 开发应用指南 386

十一 项目实战（后续补充）.....391
十二 云平台服务.....391
12.1 差分包生成工具.....391
12.2 云编译系统.....392
12.2. 1 已支持的平台:.....392
12.2.2 提醒.....392
12.2.3 访问构建网站.....393
12.2.4 点击登录.....393
12.2.5 点击右上角构建.....393
12.2.6 勾选想要的组件.....395
12.2.7 点击右上角保存修改，之后点击准备就绪.....395
12.2.8 常见问题：.....396
12.3 iRTU 管理系统.....397
12.3.1iRTU 开发简介.....397
12.3.2 固件版本及源码获取.....408
12.3.3 iRTU 新手教程.....408
12.3.4 iRTU 使用手册.....411
12.3.5 iRTU 使用实例.....411
12.4 errDump 远程日志.....465
12.4.1 概述.....465
12.4.2 原理.....465
12.4.3 测试脚本.....465
12.4.4 测试效果.....467
12.5 FOTA 远程升级.....467
12.5.1 帮助中心.....467
12.5.2 FAOTA 应用指南.....468
12.6 MES 查询.....468
12.7 基站查询工具.....469
12.8 基站 WIFI 定位.....470
12.8.1 基站/WIFI 定位原理.....470
12.8.2 免费和收费服务区别.....470
12.8.3 各固件的配置方式.....470
12.8.4 使用方法.....471
12.8.5 常见问题.....472
12.9 坐标转换.....473

12.10 网络测试工具 474
十三 合宙产品选型手册 475
十四 合宙功耗分析仪 499
14.1 Air9000 产品手册 499
14.2 Air9000 上手指南 526
14.3 Air9000P 产品手册 553
14.4 Air9000P 上手指南 587
十五 实网环境功耗测试电流曲线 598
移动卡+500 微安模式+每隔 5 分钟发送 1KB TCP 数据 598
移动卡+200 微安模式+每隔 5 分钟发送 1KB TCP 数据 599
移动卡+2 微安模式+无业务数据交互 600
电信卡+500 微安模式+每隔 5 分钟发送 1KB TCP 数据 601
电信卡+200 微安模式+每隔 5 分钟发送 1KB TCP 数据 602
电信卡+2 微安模式+无业务数据交互 603
十六 Lua 语言学习教程 603
1 序言 603
2 变量 605
3 字符串 610
4 逻辑运算 615
5 分支判断 618
6 函数 622
7 Table 629
8 循环 640
9 详解 string 库 647
跨文件调用 658
十七 AT 命令手册 661
1 AT 命令概述 661
1.1 文档目的 661
1.2 惯例和术语缩写 661
1.3 AT 命令语法 662
1.4 AT 命令最大响应时间 664
2 基本命令 664
2.1 查询制造商名称：AT+CGMI 664
2.2 查询模块型号：AT+CGMM 665
2.3 查询模块版本信息：AT+CGMR 665
2.4 查询 IMEI 号：AT+CGSN 666

2.5 查询 SIM 卡 ICCID 号码：AT+CCID(/ICCID) 666
2.6 查询 IMSI：AT+CIMI 667
2.7 查询产品信息：ATI 668
2.8 查询模块 FIRMWARE 版本：AT+VER 668
2.9 查询平台硬件版本：AT^HVER 669
2.10 查询各种信息：AT*I 669
2.11 写 SN 号命令：AT+WISN 670
2.12 重启模块：AT+RESET 671
2.13 重启模块并恢复出厂设置：AT+RSTSET 671
2.14 查询开机原因值：AT*EXINFO? 671
2.15 关闭 EPAT 导出的 PCAP 包的大小限制：AT+CATCHPCAP 672
3 配置命令 673
3.1 选择 TE 字符集：AT+CSCS 673
3.2 保存用户当前的配置：AT&W 673
3.3 设置命令回显模式：ATE 674
3.4 恢复所有参数为出厂配置：AT&F 674
3.5 设置结果码抑制模式：ATQ 674
3.6 FOTA 空中升级：AT+UPGRADE 675
3.7 设置指令行终止符：ATS3 684
3.8 设置命令行编辑字符：ATS5 684
3.9 设置 CDC 功能模式：AT&C 685
3.10 设置 DTR 功能模式：AT&D 685
3.11 实时时钟：AT+CCLK 686
3.12 设置 USB 模式：AT+SETUSB 686
3.13 设备错误：AT+CMEE 687
3.14 错误码描述：+CME ERROR:<err> 688
4 设备控制命令 694
4.1 模块功能模式：AT+CFUN 694
4.2 关机：AT+CPOWD 695
4.3 输入 PIN 码：AT+CPIN 695
4.4 设备锁定：AT+CLCK 696
4.5 修改密码：AT+CPWD 697
4.6 网络灯闪烁的时间间隔：AT+SLEDS 698
4.7 设置 TE-TA 波特率：AT+IPR 699
4.8 设置 TE-TA 帧格式：AT+ICF 700

4.9 设置指令行换行字符：ATS4 701
4.10 TE-TA 本地流量控制：AT+IFC 702
4.11 开启和关闭 SIM 卡在位硬件检测：AT+CSDT 703
4.12 检测某个插槽的 SIM 卡是否在位：AT*SIMDETEC 704
4.13 获取 Sim 类型：AT*EUICC 705
4.14 通过 UART 口设置睡眠唤醒：AT+CSCLK 705
4.15 设置在数传模式下，等待多长时间进入休眠状态：AT*RTIME 708
4.16 超低功耗指令：AT+POWERMODE 709
4.17 设置睡眠等待时间：AT+WAKETIM 711
4.18 设置 RI 指示功能：AT+CFGRI 712
4.19 保存 RI 设置功能：AT+CFGRISAVE 713
4.20 设置唤醒字符串指令：AT^WAKEUPHEX 713
4.21 读取 ADC：AT+CADC 714
4.22 读取 VBAT 电压：AT+CBC 715
4.23 打开/关闭网络灯：AT+CNETLIGHT 715
4.24 SIM 实体卡，虚拟卡切换：AT+ECSIMCFC 716
4.25 SIM 卡切换：AT+SIMCROSS 717
4.26 SIM 卡自动切换开关：AT*SIMAUTO 718
4.27 RNDIS/ECM 功能开关：AT+RNDISCALL 719
4.28 重新打开 USB：AT+SYSNV=1,"ureboot",1 719
4.29 GPIO 口控制：AT+CGPIO 720
4.30 IO 电平控制：AT+IOSET 720
5 网络服务相关命令 721
5.1 查询信号质量：AT+CSQ 721
5.2 查询信号质量（扩展）：AT+CESQ 721
5.3 打开 CSQ 主动上报：AT*CSQ 723
5.4 查询卡的 MCC 以及 MNC：AT+MNCC 724
5.5 网络注册信息：AT+CREG 724
5.6 E-UTRAN EPS 网络注册状态：AT+CEREG 726
5.7 设置网络模式：AT^SYSCONFIG 728
5.8 查询 cid 相关的上下文定义：AT+CGCONTRDP 729
5.9 运营商查询和选择：AT+COPS 731
5.10 自动时区更新：AT+CTZU 733
5.11 打开 NITZ 自动上报：AT+CTZR 733
5.12 (URC)NITZ 自动上报：+NITZ:<time>,<ds> 734

5.13 设置 NITZ 自动上报时区：AT+NITZDISSET 734
5.14 查询当前工作频段：AT*BANDIND 735
5.15 查询接入机制（Access Technology）：AT^CACAP 738
5.16 查询当前的系统信息：AT^SYSINFO 739
5.17 设置基站定位（LBS）服务器的域名和端口：AT+GSMLOCFG 740
5.18 读取基站定位（LBS）信息和时间：AT+CIPGSMLOC 741
5.19 读取 WIFI 定位信息和时间：AT+WIFILOC 743
5.20 获取 WIFI 信息：AT+WIFISCAN 745
5.21 显示 WIFISCAN 扫描数量的设置指令：AT+WIFISCANCOUNT 745
5.22 WIFISCAN 扫描否显示 SSID 和时间提前量的设置指令：AT+WIFISCANCONF 747
5.23 流量查询命令：AT^DATAINFO 748
5.24 (URC)系统模式： ^MODE 749
5.25 小区信息查询：AT+CCED 749
5.26 在工程模式下查询 GSM/UMTS/LTE 信息：AT+EEMGINFO 752
5.27 LTE 工程模式下服务小区信息：+EEMLTESVC 753
5.28 LTE 工程模式下同频信息：+EEMLTEINTRA 754
5.29 LTE 工程模式下频间信息：+EEMLTEINTER 754
5.30 (URC)网络服务类型指示 755
6 音频相关命令 755
6.1 TTS（Text To Speech）功能：AT+CTTS 755
6.2 设置 TTS 播放模式：AT+CTTSPARAM 756
6.3 语音文件播放：AT+CAUDPLAY 757
6.4 本地音频播放音量控制：AT+CRSL 759
7 短消息命令 759
7.1 PDU 短信编码格式介绍 759
7.2 选择短消息服务:AT+CSMS 761
7.3 短消息优先存储区选择：AT+CPMS 762
7.4 短消息中心地址：AT+CSCA 763
7.5 短消息格式：AT+CMGF 764
7.6 设置短信 TEXT 模式参数：AT+CSMP 765
7.7 控制 TEXT 模式下短信头信息显示：AT+CSDH 766
7.8 新消息指示：AT+CNMI 768
7.9 新短消息确认：AT+CNMA 771
7.10 发送短信：AT+CMGS 772
7.11 把消息写入存储器：AT+CMGW 774

7.12 从存储器发送短信：AT+CMSS 776
7.13 短信链路控制命令：AT+CMMS 778
7.14 读短信：AT+CMGR 778
7.15 列举短消息：AT+CMGL 781
7.16 删除短消息：AT+CMGD 783
7.17 短信业务失败结果码：CMS ERROR：<err> 784
8 分组域相关命令 788
8.1 GPRS 网络注册状态：AT+CGREG 788
8.2 GPRS 附着分离：AT+CGATT 790
8.3 PDP 上下文定义：AT+CGDCONT 791
8.4 PDP 上下文鉴权参数：AT+CGAUTH 793
8.5 显示 PDP 地址：AT+CGPADDR 793
8.6 PDP 上下文激活：AT+CGACT 794
8.7 控制非请求 GPRS 事件上报：AT+CGEREP 795
8.8 (URC)分组域事件 URC 上报：+CGEV 796
8.9 通过 cid 查询本地 IP 地址：AT*GETIP 797
9 NTP 相关命令 797
9.1 设置 GPRS 承载场景 ID：AT+CNTPCID 798
9.2 同步网络时间：AT+CNTP 798
11 文件系统读写命令 800
11.1 创建一个文件：AT+FSCREATE 800
11.2 读文件：AT+FSREAD 800
11.3 读文件的时候是否返回头：AT+FSREADHEAD 801
11.4 写文件：AT+FSWRITE 801
11.5 获取盘符：AT+FSDRIVE 802
11.6 显示文件目录列表：AT+FSLS 802
11.7 获取可用空间大小：AT+FSMEM 803
11.8 创建目录：AT+FSMKDIR 803
11.9 删除目录：AT+FSRMDIR 804
11.10 删除文件：AT+FSDEL 804
11.11 获取文件大小：AT+FSFLSIZE 804
11.12 使用方法举例 805
12 嵌入式 TCPIP 命令 807
12.1 启动多 IP 连接：AT+CIPMUX 807
12.2 启动任务并设置接入点 APN、用户名、密码：AT+CSTT 807

12.3 专网卡设置 APN、用户名、密码和鉴权方式：AT+CPNETAPN 808
12.4 激活移动场景(或发起 GPRS 或 CSD 无线连接)：AT+CIICR 809
12.5 查询本地 IP 地址：AT+CIFSR 809
12.6 设置 TCP 使用 SSL 功能：AT+CIPSSL 810
12.7 配置 TCP SSL 参数：AT+SSLCFG 810
12.8 配置本地 TCP 端口：AT+CLPORT 815
12.9 建立 TCP 连接或注册 UDP 端口号：AT+CIPSTART 816
12.10 选择 TCPIP 应用模式：AT+CIPMODE 818
12.11 选择非透传数据发送模式：AT+CIPQSEND 818
12.12 设置接收的数据末尾是否自动添加回车换行：AT+CIPRXF 819
12.13 配置透明传输模式：AT+CIPCCFG 820
12.14 发送数据：AT+CIPSEND 821
12.15 设置自动发送数据前的定时时间：AT+CIPATS 823
12.16 设置发送数据时是否显示'>'和发送情况提示：AT+CIPSPRT 824
12.17 查询当前连接状态：AT+CIPSTATUS 825
12.18 查询已连接数据传输状态：AT+CIPACK 826
12.19 设置为 CSD 或 GPRS 连接模式：AT+CIPCSGP 827
12.20 配置域名服务器 DNS：AT+CDNSCFG 827
12.21 域名解析：AT+CDNSGIP 829
12.22 设置单链接接收数据时是否显示发送方的 IP 地址和端口号：AT+CIPSRIP .. 830
12.23 设置单链接接收数据是否显示 IP 头：AT+CIPHEAD 831
12.24 设置单链接接收数据是否在 IP 头显示传输协议：AT+CIPSHOWTP 831
12.25 多链接时接收数据：+RECEIVE 832
12.26 保存 TCPIP 应用上下文：AT+CIPSCONT 832
12.27 手动获得网络数据：AT+CIPRXGET 833
12.28 关闭 TCP 或 UDP 连接：AT+CIPCLOSE 837
12.29 关闭移动场景：AT+CIPSHUT 837
12.30 修改 RNDIS 网卡网关 IP 地址：AT+ROUTEIP 838
12.31 Ping 回声请求命令:AT+CIPPING 838
12.32 设置 TCP 保活（keep-alive）参数:AT+CIPTKA 840
12.33 心跳包参数设置:AT^HEARTCONFIG 841
12.34 设置心跳包内容:AT^HEARTBEAT 842
12.35 设置 HEX 编码格式心跳包内容:AT^HEARTBEATHEX 842
12.36 查询心跳包发送情况:AT^HEARTINQUIRE 844
12.37 从数据模式或 PPP 在线模式切换至命令模式：+++ 845

12.38 从命令模式切换至数据模式：ATO 845
12.39 TCP/UDP 错误码 846
12.40 状态机 847
12.41 模块上电初始化以及 TCPIP 流程 849
12.42 使用方法举例 851
13 IP 应用相关命令 863
13.1 IP 应用设置：AT+SAPBR 863
14 HTTP 相关命令 864
14.1 初始化 HTTP 服务：AT+HTTPINIT 864
14.2 启用 SSL：AT+HTTPSSL 865
14.3 设置 HTTP 参数值：AT+HTTPPARA 865
14.4 写数据：AT+HTTPDATA 867
14.5 HTTP 方式激活：AT+HTTPACTION 868
14.6 HTTP 方式激活(扩展)：AT+HTTPEXACTION 870
14.7 查询 HTTP 服务响应：AT+HTTPREAD 870
14.8 获取 HTTP 服务响应数据：AT+HTTPGET 871
14.9 获取 HTTP 服务响应数据(扩展)：AT+HTTPEXGET 871
14.10 推送 HTTP 数据(扩展)：AT+HTTPEXPOST 872
14.11 下载文件并保存到文件系统中：AT+HTTPGETTOFS 872
14.12 查询 HTTP 头信息：AT+HTTPHEAD 873
14.13 保存 HTTP 应用上下文：AT+HTTPSCONT 873
14.14 终止 HTTP 任务：AT+HTTPTERM 874
14.15 HTTP 错误码：ERROR：<err code> 874
14.16 使用方法举例 875
15 FTP 相关命令 881
15.1 设置 FTP 控制端口：AT+FTPPORT 881
15.2 设置 FTP 主动或被动模式：AT+FTPMODE 882
15.3 设置 FTP 数据传输类型：AT+FTPTYPE 882
15.4 设置 FTP 输入类型：AT+FTPPUTOPT 883
15.5 设置 FTP 承载标识：AT+FTPCID 883
15.6 设置 FTP 下载续传：AT+FTPREST 883
15.7 设置 FTP 服务器地址：AT+FTPSERV 884
15.8 设置 FTP 用户名称：AT+FTPUN 884
15.9 设置 FTP 密码：AT+FTPPW 885
15.10 设置 FTP 下载文件名称：AT+FTPGETNAME 885

15.11 设置 FTP 下载文件路径：AT+FTPGETPATH 886
15.12 设置 FTP 上传文件名称：AT+FTPPUTNAME 886
15.13 设置 FTP 上传文件路径：AT+FTPPUTPATH 886
15.14 远程服务器上创建文件目录：AT+FTPMKD 887
15.15 远程服务器上删除文件目录：AT+FTPRMD 887
15.16 下载文件：AT+FTPGET 888
15.17 上传文件：AT+FTPPUT 889
15.18 下载文件(扩展)：AT+FTPEXTGET 890
15.19 上传文件(扩展)：AT+FTPEXTPUT 891
15.20 下载文件并保存到文件系统中：AT+FTPGETTOFS 892
15.21 从文件系统上传文件到服务器：AT+FTPPUTFRMFS 892
15.22 从文件系统加载到 RAM 中并用 FTPPUT 上传：AT+FTPFILEPUT 893
15.23 获取远程服务器上文件目录：AT+FTPLIST 894
15.24 获取远程服务器上文件大小：AT+FTPSIZE 895
15.25 获取 FTP 状态：AT+FTPSTATE 895
15.26 保存 FTP 应用上下文：AT+FTPSCONT 895
15.27 删除服务器上指定的文件：AT+FTPDELE 896
15.28 退出当前 FTP 会话：AT+FTPQUIT 896
15.29 使用方法举例 897
16 MQTT 相关命令 907
16.1 设置 MQTT 相关参数：AT+MCONFIG 907
16.2 建立 TCP 连接：AT+MIPSTART 908
16.3 客户端向服务器请求会话连接：AT+MCONNECT 909
16.4 发布消息：AT+MPUB 910
16.5 发布定长消息：AT+MPUBEX 911
16.6 订阅主题：AT+MSUB 913
16.7 取消订阅主题：AT+MUNSUB 913
16.8 打印收到的所有的订阅消息：AT+MQTTMSGGET 914
16.9 设置订阅消息的打印模式：AT+MQTTMSGSET 915
16.10 MQTT 消息编码格式切换：AT+MQTTMODE 915
16.11 关闭 TCP 连接：AT+MIPCLOSE 916
16.12 关闭 MQTT 连接：AT+MDISCONNECT 916
16.13 查询 MQTT 连接状态：AT+MQTTSTATU 916
16.14 使用方法举例 917
18 GPS 相关的命令 922

18.1GPS 开关：AT+CGNSPWR 922
18.2 读取 GNSS 信息：AT+CGNSINF 922
18.3 打开 GNSS URC 上报：AT+CGNSURC 923
18.4 将读取到的 GNSS 数据发送到 AT 口：AT+CGNSTST 924
18.5 给 GNSS 发送控制命令：AT+CGNSCMD 924
18.6 设置辅助定位：AT+CGNSAID 925
18.7 删除 EPO 文件：AT+CGNSDEL 925
18.8 定义 NMEA 解析 ：AT+CGNSSEQ 926
18.9 使用方法举例 926
十八 LuatOS API 927
adc - 数模转换 927
antbot - 蚂蚁链 932
audio - 多媒体-音频 938
bit64 - 32 位系统上对 64 位数据的基本算术运算和逻辑运算 949
camera - 摄像头 956
cc - 通话功能 963
codec - 多媒体-编解码 968
coremark - 跑分 973
crypto - 加解密和 hash 函数 974
eink - 墨水屏操作库 993
errDump - 错误上报 1009
fastlz - FastLZ 压缩 1013
fatfs - 读写 fatfs 格式 1015
fonts - 字体库 1019
fota - 底层固件升级 1021
fs - 文件系统额外操作 1027
fskv - kv 数据库,掉电不丢数据 1029
ftp - ftp 客户端 1037
gmssl - 国密算法 1041
gpio - GPIO 操作 1047
gtfont - 高通字库芯片 1056
hmeta - 硬件元数据 1057
http - http 客户端 1059
httpsrv - http 服务端 1062
i2c - I2C 操作 1064

i2s - 数字音频 1075
iconv - iconv 操作 1082
io - io 操作(扩展) 1085
ioqueue - io 序列操作 1094
iotauth - IoT 鉴权库，用于生成各种云平台的参数 1105
ir - 红外遥控 1110
json - json 生成和解析库 1111
keyboard - 键盘矩阵 1113
lcd - lcd 驱动模块 1115
libgnss - NMEA 数据处理 1138
log - 日志库 1155
lora - lora 驱动模块 1160
lora2 - lora2 驱动模块(支持多挂) 1166
lvgl - LVGL 图像库 1173
max30102 - 心率模块(MAX30102) 1202
mcu - 封装 mcu 一些特殊操作 1204
miniz - 简易 zlib 压缩 1214
mlx90640 - 红外测温(MLX90640) 1217
mobile - 蜂窝网络 1224
mqtt - mqtt 客户端 1248
nes - nes 模拟器 1261
os - os 操作 1262
otp - OTP 操作库 1267
pack - 打包和解包格式串 1270
pm - 电源管理 1272
protobuf - ProtoBuffs 编解码 1284
pwm - PWM 模块 1288
repl - “读取-求值-输出” 循环 1290
rsa - RSA 加密解密 1293
rtc - 实时时钟 1297
rtos - RTOS 底层操作库 1301
sensor - 传感器操作库 1309
sfd - SPI FLASH 操作库 1320
sfud - SPI FLASH sfud 软件包 1324
sms - 短信 1331

socket - 网络接口 1335
spi - spi 操作库 1353
sqlite3 - sqlite3 数据库操作 1365
statem - SM 状态机 1368
string - 字符串操作函数 1369
sys - sys 库 1377
sysplus - sys 库的强力补充 1385
u8g2 - u8g2 图形处理库 1389
uart - 串口操作库 1411
w5500 - w5500 以太网驱动 1421
wdt - watchdog 操作库 1424
websocket - websocket 客户端 1426
wlan - wifi 操作 1433
ws2812 - 幻彩灯带 RGB 控制器(WS2812 系列) 1446
xxtea - xxtea 加密解密 1450
ymodem - ymodem 协议 1451
zbuff - c 内存数据操作库 1455
十九 高低温射频传导测试报告 1474
测试总览 1474
测试设备 1475
测试环境 1476
测试项目 1478
测试数据 1478
最大输出功率@频段 B1 1478
射频灵敏度@射频 B1 1479
最大输出功率@频段 B3 1480
射频灵敏度@射频 B3 1482
最大输出功率@频段 B5 1482
射频灵敏度@射频 B5 1483
最大输出功率@频段 B8 1484
射频灵敏度@射频 B8 1484
最大输出功率@频段 B34 1485
射频灵敏度@射频 B34 1485
最大输出功率@频段 B38 1487
射频灵敏度@射频 B38 1487

最大输出功率@频段 B39 1488
射频灵敏度@射频 B39 1488
最大输出功率@频段 B40 1489
射频灵敏度@射频 B40 1490
最大输出功率@频段 B41 1490
射频灵敏度@射频 B41 1491
总览(高温 85 度) 1492
总览(低温-40 度) 1493
二十 合宙创业故事：我的物联网大学（已完结） 1494
1 迷茫的前三年 1494
壹 业务的困境 1494
贰 团队信任危机 1495
叁 关于公司的对话 1497
2 Luat 的出世 1498
壹 启动火种 1498
贰 通过 Luat 认识市场 1500
叁 神奇又辉煌的 AT 指令集 1501
肆 OpenCPU 需求的兴起 1504
伍 成长的烦恼 1505
3 在问题中突围，还是灭亡？ 1507
壹 4G 的突围，成为主流模块品牌 1507
贰 过于分散的客户群，是健康的吗？ 1510
叁 扩展流量卡业务 1511
肆 解决问题的核心是预防问题 1512
伍 逼上梁山，自研自动化运营系统 1513
陆 问题重新定义 1515
4 沉默不会爆发，沉默一定会灭亡！ 1516
5 个人英雄主义和组织效率的矛盾 1519
6 2019 年的市场形势 1521
7 跟一个女采购见光死的故事 1523
8 怎么样才能让团队更专业？ 1525
9 不能远程升级的 4G 模块都是耍流氓 1527
10 Cat.1 模块价格究竟可以多低？ 1532
11 男人的郁闷怎么解决？ 1536
12 王强的逆袭翻身 1540

13 生意还是事业？ 1545
14 物联网生意经 1549
二十一 合宙创业故事：合宙的第一个 10 年 1550
1 客户破解了我的孤独症 1550
2 每个人的世界都完全不同 1553
3 不再说愿景使命价值观 1558
4 高质量的复盘是成功之父 1561
5 逐字稿是很笨很笨的工具 1564
6 成功有秘诀吗？还真有 1567
7 不可能的行动变成可能，也有方法 1573
8 基本功也有降维打击吗？ 1578
9 建立客户信任还有公式？ 1582
10 亲密度关系的分类和拆解 1585
11 经营良性可持续的长期关系 1588

一 本手册阅读说明

本章基于使用者对合宙产品的熟悉程度以及软件开发方式等因素，给出本手册阅读顺序说明。以下阅读顺序只是建议，使用者可以根据自己的习惯和需求选择自己想要阅读的章节。

1.1 刚接触合宙新朋友

对合宙公司和合宙产品不熟悉的新朋友，建议先看一下第 2 章节和第 3 章节，建议阅读顺序如下：

1. 二 公司介绍
2. 十三 合宙产品选型手册
3. 十四 合宙功耗分析仪
4. 十二 云平台服务
5. 四 产品规格书
6. 十五 实网环境功耗测试电流曲线
7. 十九 高低温射频传导测试报告
8. 五 硬件设计指南
9. 八 工具使用指南
10. 六 快速入门
11. 七 开发板使用说明（后续补充）
12. 十一 项目实战（后续补充）
13. 二十 合宙创业故事：我的物联网大学（已完结）
14. 二十一 合宙创业故事：合宙的第一个 10 年

1.2 熟悉合宙的老朋友

对于合宙和合宙产品都比较熟悉的老朋友，根据开发方式不同，建议阅读顺序如下：

1.2.1 AT 开发方式

1. 四 产品规格书
2. 五 硬件设计指南
3. 六 快速入门
4. 七 开发板使用说明（后续补充）
5. 十七 AT 命令手册
6. 九 软件开发指南（AT）
7. 十一 项目实战（后续补充）

1.2.2 LuatOS 开发方式

1. 四 产品规格书
2. 五 硬件设计指南
3. 十六 Lua 语言学习教程
4. 十八 LuatOS API
5. 十 软件开发指南（LuatOS）
6. 六 快速入门
7. 七 开发板使用说明（后续补充）
8. 十一 项目实战（后续补充）

二 公司介绍

上海合宙通信科技有限公司，成立于 2014 年 5 月，至今已经超过 10 年，是一家专注于解决低功耗联网解决方案的高科技公司。

上海合宙通过对整体系统持续降低功耗， 对嵌入式操作 LuatOS 开发便利性持续迭代，为企业客户提供完善的整体物联网解决方案，让客户能够在市场中迅速发挥出各自的价值。

为了持续降低功耗， 上海合宙推出了功耗分析仪 Air9000 和 Air9000P，让客户硬件功耗调试更加便利，让客户的产品能够卖的更贵。

合宙的总部在上海， 在深圳，合肥都设立有分公司，在四川眉山有 1 万平方的自动化工厂， 期待在合宙发展的第二个 10 年，能够帮客户发挥出更多的市场价值。

三 公司产品选型手册

关于合宙产品选型手册的介绍，请参考本文档 第十三章 合宙产品选型手册。

四 产品规格书

4.1 功能特性

- 主芯片
  - 移芯 EC718P，支持 LTE 3GPP Rel.14 技术；TSMC 22nm 工艺制程；
  - 尺寸 6.1mm*6.1mm；
  - CPU：Cortex M3 @ 306MHz*2；
- 频段
  - LTE-FDD：B1、B3、B5、B8；最大上行速率 5Mbps，最大下行速率 10Mbps；
  - LTE-TDD：B34、B38、B39、B40、B41；上下行配置 1，最大上行速率 4Mbps，最大下行速率 6Mbps；上下行配置 2，最大上行速率 2Mbps，最大下行速率 8Mbps；
  - 4G 国内全网通；
- 模组基本信息
  - 封装：LGA 封装，109 个管脚；
  - 尺寸：17.7mm*15.8mm*2.3mm；

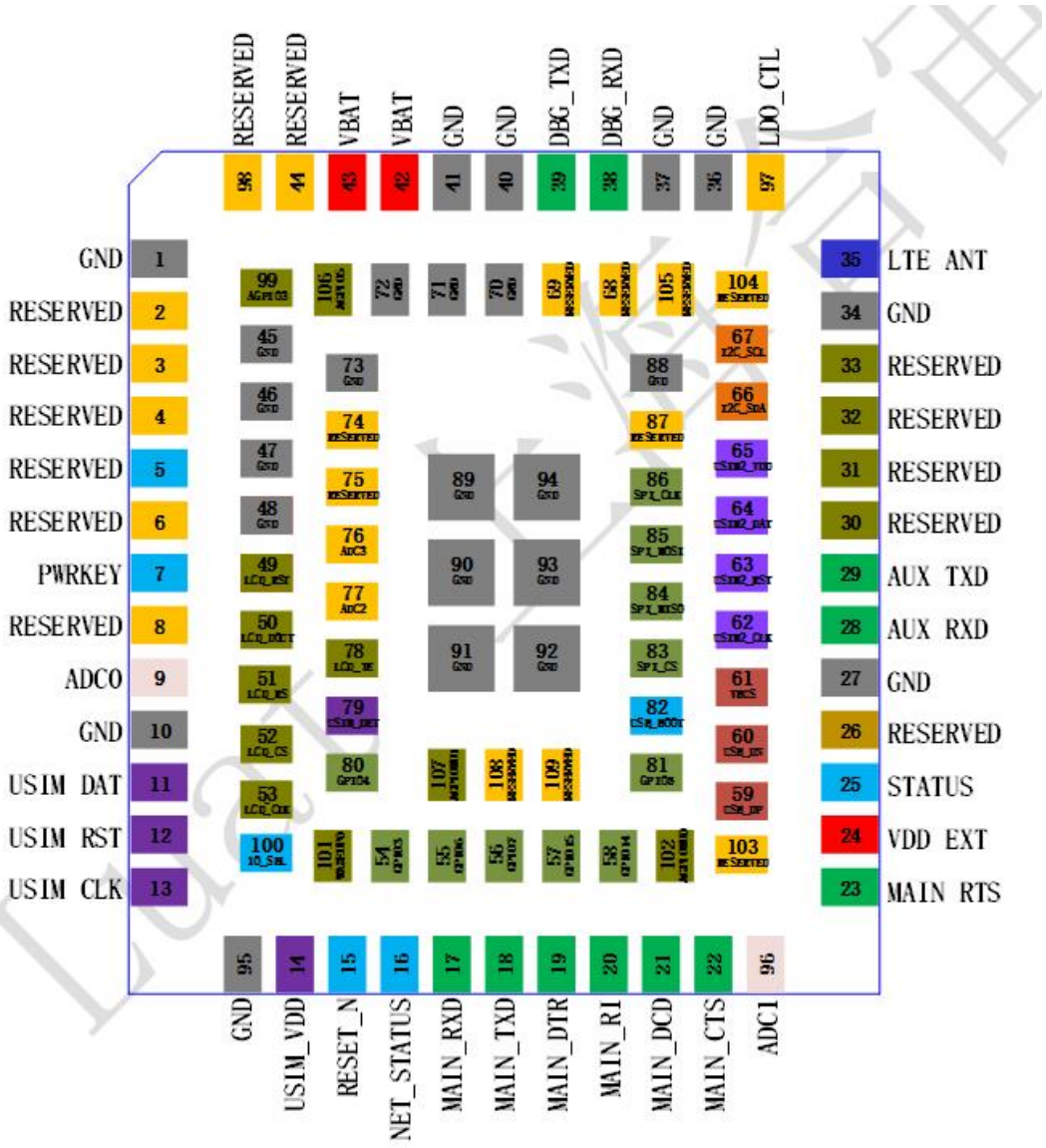
  - 重量：约 1.16 克；
- Memory
  - FLASH：4MB；
  - RAM：2MB SRAM + 2MB pSRAM；
- SIM 卡：支持双卡单待；
- USB：支持 1 路 USB 2.0；
- SPI：支持 2 路，最高时钟 25.6MHz；
- I2C：支持 2 路；
- PWM：可用 4 路，最高频率 13MHz，
- UART：支持 4 路 UART；
  - UART0 默认为日志输出调试口；在需要低功耗的项目中，如果低功耗场景下出现 bug 需要调试，必须使用 UART0 抓日志分析解决问题，所以在低功耗的项目中，UART0 必须做日志调试口使用，否则问题无法分析解决；在不需要低功耗的项目中，如果另外 3 路 UART 满足不了项目需求，可以考虑将 UART0 做普通 UART 使用，此时出现问题可以使用 USB 接口调试；
  - 在上电开机过程中，UART0 会输出一段日志信息，这段日志信息是固化在 bootloader 中的，没有办法去掉；UART0 接外设时，需要评估下对外设通信的影响；
  - UART1 波特率配置为 9600 时可做低功耗串口；当模组休眠后，UART1 连接的外设，可以随时发送数据给模组，唤醒模组并且不丢失数据；
- ADC：支持 4 路 12bit ADC；除此之外，内部还支持温度以及 VBAT 检测；
- IO 电平：
  - 硬件上可以通过第 100 引脚配置 1.8V 或者 3.3V；v
  - 软件上可以配置范围 1650 MV 到 2000MV，2650MV 到 3400MV，步进 50MV；
- 网络协议：IPv4、IPv6、TCP、UDP、SSL、MQTT(S)、HTTP(S)、FTP(S)、NTP、NITZ；
- 远程升级：支持差分升级，部分固件支持全量升级；合宙提供免费的升级后台，也可以自建升级后台；
- 支持 WIFI SCAN；
- GPIO：共 39 个（部分 GPIO 和 UART、SPI、WAKEUP 等其功能引脚复用）
  - 9 个 AGPIO，休眠状态下可保持输出高或者低；
  - 其余普通 GPIO 在休眠后掉电，不可使用，均处于高阻状态；
  - 可以复用为 WAKEUP 功能的 GPIO，支持同时配置为双边沿中断或者同时配置为高低电平中断；
  - 不可复用为 WAKEUP 功能的 GPIO，仅支持单边沿中断或者单电平中断；
- 超低功耗：支持 500 微安、200 微安、2 微安三种超低功耗休眠模式；
- 休眠唤醒：休眠后可以通过 WAKEUP 引脚、UART1、USB、POWERKEY、RTC、CHARGE

PAD 唤醒；

- 运营商业务
  - 短信：支持中国移动和中国联通的短信功能，不支持中国电信的短信功能；
  - VoLTE：不支持 VoLTE 通话功能；
- 公共云平台：支持阿里云、天翼云、华为云、腾讯云、ONENET 等；
- 蓝牙：不支持；
- 算法&数据格式：
  - MD5/SHA1/SHA256/HMAC-MD5/HMAC-SHA1/HMAC-SHA256/XXTEA/AES/RSA/CRC8/CRC16/CRC32/Base64；
  - 支持 JSON、Protobuffer；

4.2 管脚排列图

模组采用 LGA 封装，109 个 SMT 焊盘管脚；



4.3 电气参数

电压电流耐受值

参数	最小值	最大值	单位
VBAT	-0.3	4.7	V
VBUS	-0.3	5.5	V
电源供电峰值电流	0	1.5	A
电源供电平均电流（TDMA 一帧时间）	0	0.7	A
数字管脚处电压	-0.3	3.6	V
模拟管脚处电压（ADC）	-0.3	3.6	V

工作条件

参数	最小值	典型值	最大值	单位
VBAT	3.3	3.8	4.3	V
VBUS	3.3	5.0	5.25	V

工作温度

温度	最低值	典型值	最高值	单位
正常工作温度	-35	25	75	℃
受限工作温度	-40	/	85	℃
存储温度	-45	/	90	℃

4.4 实网功耗测试报告

以下测试数据由 合宙功耗分析仪 Air9000 ，在实网环境下，测试生成。

Air9000 适用于各类电池供电产品的动态功耗测试，电流范围 0 到 5 安，最小分辨率 0.5 微安；电压范围 0 到 24 伏，最小分辨率 1 毫伏。更详细的介绍，请参考本文档 第十六章 Air9000-合宙功耗分析仪 。

功耗测试的详细电流波形图，请参考本文档 第十七章 实网环境功耗测试电流曲线 。

测试条件：Air780EP系列型号的模组　移动网络 B34 RSRP=-94到-91 CESQ=49到51 (中等信号强度) 供电3.8V								
休眠模式	平均功耗mA (5分钟发送一次心跳)	心跳包发送		休眠平均功耗		500mAH电池待机 (天)		
		功耗mA	耗电电量uAH	功耗mA	耗电电量uAH(5分钟)	5分钟上报间隔	1小时上报间隔	24小时上报间隔
500微安模式	0.73	31	12.7	0.574	47.8	28.53	35.53	36.28
200微安模式	0.553	35	12.9	0.353	29.5	37.67	56.78	58.76
2微安模式	0.0028 (无心跳)	\	\	0.0028	\	104.4	1085	>10年

测试条件：Air780EP系列型号的模组　电信网络 B3 RSRP=-88 CESQ=52 (中等信号强度) 供电3.8V								
休眠模式	平均功耗mA (5分钟发送一次心跳)	心跳包发送		休眠平均功耗		500mAH电池待机 (天)		
		功耗mA	耗电电量uAH	功耗mA	耗电电量uAH(5分钟)	5分钟上报间隔	1小时上报间隔	24小时上报间隔
500微安模式	0.656	76	26.28	0.337	28.1	31.75	57.31	61.58
200微安模式	0.426	36	13.15	0.268	22.4	48.9	73.89	77.34
2微安模式	0.0028 (无心跳)	\	\	0.0028	\	104.4	1085	>10年

4.5 高低温射频传导测试报告

测试条件： 带宽 10M Hz；低温-40℃；高温 85℃；			
	测试项目	**测试标准**	**测试结果**
1	UE Maximum Output Power	＞19.3dBm~＜24.7dBm	Pass
2	Minimum Output Power	＜-39dBm	Pass
3	Frequency Error	≤ 0.1 PPM	Pass
4	Error Vector Magnitude	＜12.50%dBm	Pass

	(EVM)		
5	Adjacent Channel Leakage power Ratio	＞32.2dBc	Pass
6	Reference sensitivity level	@10M＜-96.3dBm	Pass

4.6 吞吐量测试报告

测试条件：Air780EP系列型号的模组　CSQ=25到29 (信号较好)　供电3.8V			
运营商物联网卡	频段	上行速率 (Kbits/s)	下行速率 (Kbits/s)
中国电信	band 3	4060	3338
中国联通	band 3	3535	3337
中国移动	band 8	4730	3290
	band 34	945	4289

4.7 证书资料

4.7.1 CCC 证书

证书编号：2022011606510638

发证日期：2023 年 06 月 01 日
有效期至：2027 年 11 月 28 日

认证委托人名称及注册地址	上海合宙通信科技有限公司 上海市黄浦区制造局路 787 号二幢 555A 室
产品生产者名称及注册地址	上海合宙通信科技有限公司 上海市黄浦区制造局路 787 号二幢 555A 室
生产企业名称及生产地址	四川合佳科技有限公司 四川省眉山市洪雅县将军镇八面山路四川洪雅经济开发区科创园一号楼
产品名称和系列、规格、型号	LTE 模块（具有 TD-LTE、FDD-LTE 功能） Air780E：3.3-4.3VDC（由主板专用接口供电）
产品标准和技术要求	YD/T2583.14-2013;GB 4943.1-2022

上述产品符合 CNCA-C16-01:2014 认证规则的要求，特发此证。
证书有效期内本证书的有效性依据发证机构的定期监督获得保持。
本证书为变更证书，证书首次颁发日期：2022 年 11 月 29 日
经中国合格评定国家认可委员会认可 CNAS C001-P

可通过扫描下方二维码或登录国家认监委网站（www.cnca.gov.cn）查验证书信息

签发：谢肇煦

中国质量认证中心

http://www.cqc.com.cn　　中国·北京·南四环西路188号9区 100070　　电话：+86 10 83886666

CERTIFICATE FOR CHINA COMPULSORY PRODUCT CERTIFICATION

CERTIFICATE NO.: 2022011606510638

Valid from: Jun.01,2023
Valid until: Nov.28,2027

NAME AND REGISTERED ADDRESS OF THE APPLICANT	Shanghai Hezhou Communication Technology Co., Ltd. Room 555A, Building 2, No. 787, Manufacturing Bureau Road, Huangpu District, Shanghai
NAME AND REGISTERED ADDRESS OF THE MANUFACTURER	Shanghai Hezhou Communication Technology Co., Ltd. Room 555A, Building 2, No. 787, Manufacturing Bureau Road, Huangpu District, Shanghai
NAME AND LOCATION OF THE FACTORY	Sichuan Hejia Technology Co., Ltd. Building 1, Kechuang Park, Hongya Economic Development Zone, Bamianshan Road, General Town, Hongya County, Meishan City,Sichuan Province
PRODUCT MODEL(S)	LTE Module Air780E：3.3-4.3VDC（Powered by motherboard dedicated interface）
STANDARDS AND TECHNICAL REQUIREMENTS	YD/T2583.14-2013;GB 4943.1-2022

This is to certify that the above mentioned product(s) complies with the requirements of implementation rules for compulsory certification(REFNO. CNCA-C16-01:2014)

The validity of the certificate is subject to positive result of the regular follow up inspection by issuing certification body until the expiry date.

Date of original issued: Nov.29,2022

Accredited by China National Accreditation Service for Conformity Assessment CNAS C001-P

The certificate information is available through the QR code below or CNCA's website: www.cnca.gov.cn

SIGNATURE: 谢肇煦

CHINA QUALITY CERTIFICATION CENTRE

http://www.cqc.com.cn Section 9, No.188, Nansihuan Xilu, Beijing 100070 P.R.China Tel: +86 10 83886666

4.7.2 入网许可证书

中华人民共和国工业和信息化部

电信设备进网试用批文

批文号：00-B975-238032

根据《中华人民共和国电信条例》及国家有关规定，经工业和信息化部审查，同意下述电信设备接入公用电信网试用，特发此批文。

申请单位：上海合宙通信科技有限公司

生产企业：上海合宙通信科技有限公司

设备名称：TD-LTE无线数据终端

设备型号：Air780E

设备产地：四川省眉山市

试用范围：不限

试用规模：不限

备　　注：1）该设备支持TD-LTE/LTE FDD制式；2）该设备为LTE单天线终端设备；3）本证书仅涵盖与接入公用电信网相关的通信标准和要求。

发证机关（盖章）

签发日期：2023 年 12月 18 日

有效期至：2025 年 12月 18 日

4.7.3 SRRC 证书

无线电发射设备
Radio Transmission Equipment
型号核准证
Type Approval Certificate

上海合宙通信科技有限公司：

根据《中华人民共和国无线电管理
In accordance with the provisions on the Radio
条例》，经审查，下列无线电发射设备
Regulations of the People's Republic of China , the following
符合中华人民共和国无线电管理规定和
radio transmission equipment , after examination , conforms
技术标准，其核准代码为：CMIIT ID:2023CP1169(M)
to the provisions with its CMIIT ID:

有效期：五年
Validity

中华人民共和国工业和信息化部
（发证机关）
Sealed by Issuing authority
2023 年 01 月 19 日
Year Month Date

编号：
Number 2023-1169

设备名称：TD-LTE/LTE FDD模块
Equipment Name

设备型号：Air780E
Equipment Type

主要功能：
Main Function 数据传输

调制方式：QPSK/16QAM
Modulation Mode

主要技术参数及其指标值：
Main Technical Parameters

频率范围：1880-1915MHz 2300-2370MHz 2575-2635MHz 2010-2025MHz/1920-1965MHz 1710-1785MHz 889-915MHz 824-835MHz
Frequency Range

频率容限：±0.1ppm+15Hz
Frequency Tolerance

发射功率：23dBm±2.7dB
Transmitting Power

占用带宽：≤20MHz
≤20/15/10/5/3/1.4MHz
Occupied Bandwidth

杂散发射限值：≤-30dBm
Spurious Emission Limits

（核发单位章）
Sealed by issuing authority
2023 年 01 月 19 日
Year Month Date

4.7.4 ROHS 证书

Report No.: LCSA022423036R

中国认可
国际互认
检测
TESTING
CNAS L4595

Verification Report

Applicant	:	**Shanghai Hezhou Communication Technology Co., Ltd.**
Address	:	**Room 555A, Building 2, No. 787, Manufacturing Bureau Road, Huangpu District, Shanghai**

Report on the submitted samples said to be:

Sample Name(s)	:	**LTE Module**
Trade Mark	:	**N/A**
Part No.	:	**Air780E**
Sample Received Date	:	**February 24, 2023**
Testing Period	:	**February 24, 2023 ~ March 03, 2023**
Date of Report	:	**March 03, 2023**
Testing Location	:	**901, No.40 Building, Xialang Industrial Zone, Heshuikou Community, Matian Street, Guangming District, Shenzhen, Guangdong, China**
Results	:	**Please refer to next page(s).**

TEST REQUEST	CONCLUSION
As specified by client, based on the performed tests on submitted sample, the result of Lead(Pb), Cadmium(Cd), Mercury(Hg), Hexavalent Chromium(Cr(VI)), PBBs, PBDEs, Dibutyl Phthalate(DBP), Butylbenzyl Phthalate(BBP), Di-2-ethylhexyl Phthalate(DEHP) and Diisobutyl phthalate(DIBP) content comply with the limits set by RoHS Directive 2011/65/EU with amendment (EU) 2015/863.	PASS

Signed for and on behalf of LCS

LCS TESTING LABORATORY APPROVED

Lily dan

Shenzhen LCS Compliance Testing Laboratory Ltd.
Add: 901, No.40 Building, Xialang Industrial Zone, Heshuikou Community, Matian Street, Guangming District, Shenzhen, Guangdong, China
Tel: +(86) 0755-82591330 | E-mail: webmaster@lcs-cert.com | Web: www.lcs-cert.com
Scan code to check authenticity

4.8 软件开发方式

- AT：MCU 做主控，MCU 发 AT 命令给模组的开发方式，模组仅提供标准的 AT 固件，所有的业务控制逻辑在 MCU 端实现；
- LuatOS：无需 MCU，模组支持软件二次开发，业务逻辑直接在模组上实现，开发语言为 Lua 脚本语言；
  - 用户可用 ROM 空间：脚本：256KB；文件系统：128KB；FSKV 分区：64KB；FOTA 分区：420KB；
  - 用户可用 RAM 空间：内核系统(SRAM)：250KB；Lua 虚拟机(SRAM)：256KB；pSRAM：2MB；
  - 此处列举的用户可用 ROM 和 RAM 空间为发布的系统固件默认配置数值或者上限数值；如果默认发布的系统固件功能满足不了需求，可以在云编译系统中自定义编译系统固件，一般来说，自定义编译的系统固件功能越多，给用户可用的剩余空间就越小；

4.9 技术支持服务

- 企业客户：合宙通信为每一位企业客户创建企业微信 VIP 小群提供专业的技术支持服务，如果还没有创建企业微信 VIP 小群，可以联系合宙的销售人员创建，如果没有销售人员的联系方式，请参考本文档中 联系我们 章节；
- 个人客户：合宙通信为每一位个人客户提供以下几种支持渠道
  - 论坛：点击右方链接 客户支持服务论坛，进入论坛发帖提问；
  - QQ 群：当前推荐 降功耗找合宙 10 群: 967368887；

4.10 模块生产测试记录

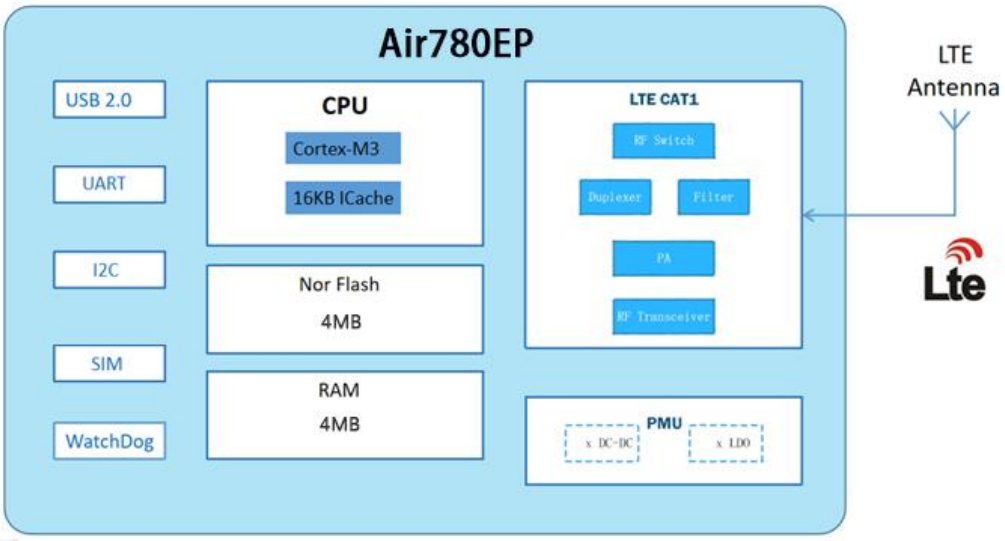
点击右方链接获取模块生成测试记录：模块生产测试记录

五 硬件设计指南

5.1 绪论

Air780EP 是一款基于移芯 EC718P 平台设计的 LTE Cat 1 无线通信模组。支持 FDD-LTE/TDD-LTE 的 4G 远距离无线 传输技术。另外，模组提供了 USB/UART/I2C 等通用接口满足 IoT 行业的各种应用诉求。

下图为 Air780EP 模块功能框图：



图表 1：功能框图

5.2 综述

5.2.1 模块主要性能

表格 1：模块主要性能列表

型号	Air780EP
CPU	♦ CortexM3@306MHz*2 ♦ 16KB ICache
Flash	♦ Nor Flash 4MB
RAM	♦ RAM 4MB
支持频段	♦ LTE-FDD：B1/B3/B5/B8 ♦ LTE-TDD：B34/B38/B39/B40/B41
发射功率	♦ LTE-FDD：Class3(23dBm+-2dB) ♦ LTE-TDD：Class3(23dBm+1/-3dB)

供电	♦ VBAT 3.3V ~ 4.3V，典型值3.8V
IO 电平	♦ 1.8V/3.3V 可配置
LTE 特性	♦ 最大支持non-CA CAT1 ♦ 支持1.4~20MHz射频带宽 ♦ LTE-FDD：最大上行速率 5Mbps，最大下行速率 10Mbps ♦ LTE-TDD：上下行配置1 最大上行速率 4Mbps，最大下行速率 6Mbps ♦ LTE-TDD：上下行配置2 最大上行速率 2Mbps，最大下行速率 8Mbps
网络协议特性	♦ 已支持TCP/UDP/HTTP/NITZ/NTP/HTTPS/MQTT
USIM 卡接口	♦ 支持USIM/SIM卡：1.8V和3V
USB 接口	♦ 支持 USB 2.0 High speed（只支持从模式），数据传输速率最大到480Mbps ♦ 用于AT指令、数据传输、软件调试、软件升级 ♦ USB 虚拟串口驱动：支持Windows 7/8.1/10，等操作系统下的 USB 驱动
串口	MAIN_UART： ♦ 通用串口，可用于AT命令和数据传输 ♦ 支持硬件流控（RTS/CTS） AUX_UART： ♦ 通用串口 DBG_UART： ♦ 用于输出调试信息
I2C	♦ 1路I2C接口
WatchDog	♦ 软件看门狗
天线接口	♦ 一个LTE天线接口
温度范围	♦ 正常工作温度：-35° C～+70° C ♦ 极限工作温度：-40° C～+85° C
RoHS	♦ 所有器件完全符合RoHS标准
物理特性	♦ 尺寸：17.7mm*15.8mm*2.3mm ♦ 重量：约1.16g

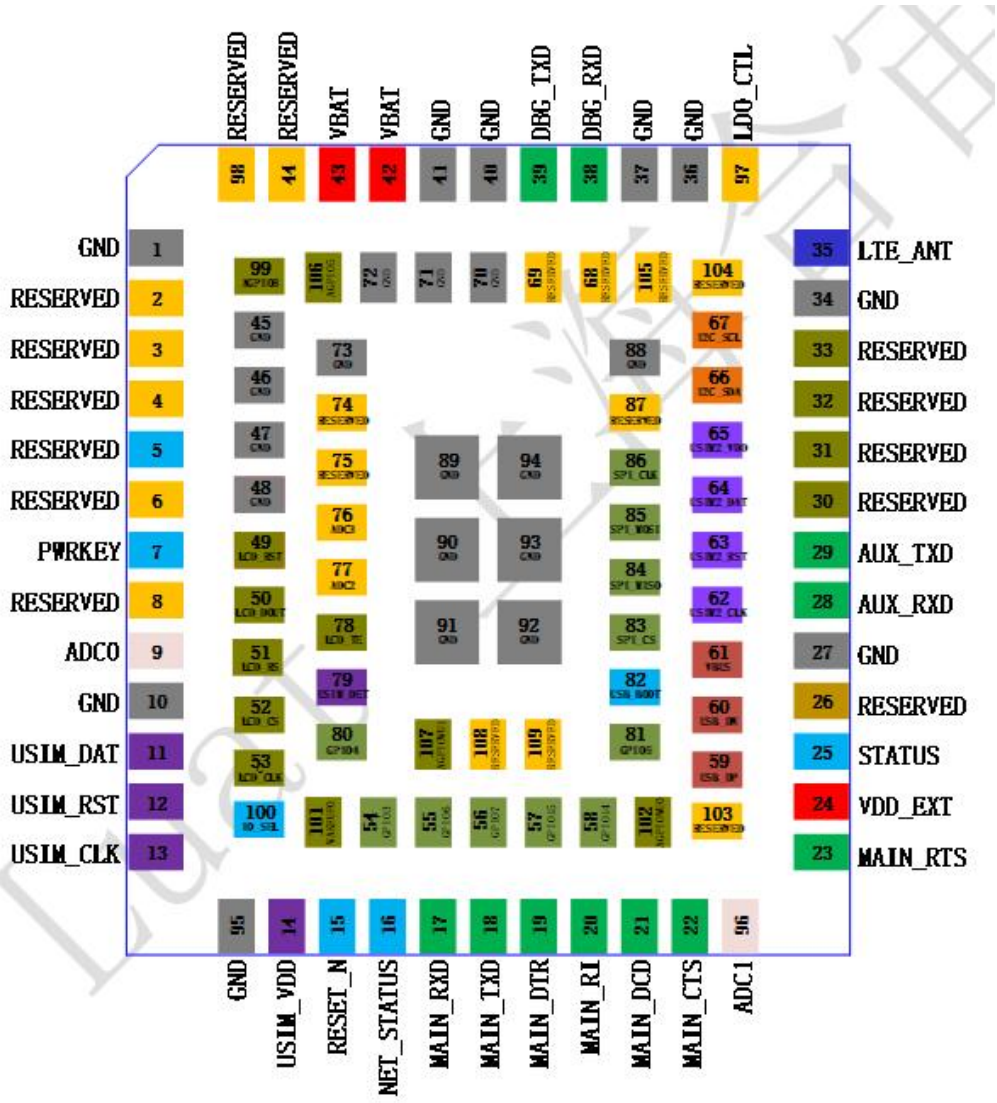
封装	♦ LGA，109个管脚，实际可用管脚详见管脚图
备注	♦ 4G 全网通

5.3 应用接口

模块采用 LGA 封装，109 个 SMT 焊盘管脚，以下章节将详细阐述 Air780EP 各接口的功能

5.3.1 管脚描述

5.3.1.1 基本描述



图表 2：Air780EP 管脚排列图（正视图）

表格 3：管脚描述

电源

管脚名	管脚号	IO	描述	电气特性	备注
VBAT	43,42	PI	模块主电源	VBAT=3.3V~4.3	供电能力满足瞬时电流大于 1.5A。
VDD_EXT	24	DO	通过 IO_SEL 硬件管脚配置，输出 3.3V 或 1.8V ,缺省 1,8V.	Imax=200mA	休眠模式下 VDD_EXT 处于掉电状态，且会持续输出高脉冲，禁止用于唤醒中断管脚

			上拉。		
GND	1,10,27,34,36,37,40,41,45,46,47,48,88,70,71,72,73,89,90,91,92,93,94,95		参考地		

控制管脚

管脚名	管脚号	IO	描述	电气特性	备注
PWKEY	7	DI	开关机控制管脚	VBAT=3.3V~4.3	电压低于 3.3V 时射频指标会恶化
RESET_N	15	DI	模块复位输入，低脉冲有效，复位后模块处于硬件关机状态		不用则悬空,无需上拉
USB_BOOT	82	DI	下载模式控制关机，开机过程中拉高到 VDD_EXT 进入下载模式		
IO_SEL	100	DI	IO 电平设置管脚，设置 VDD_EXT 管脚电压		悬空（缺省）：VDD_EXT 设置为 1.8V 拉低：VDD_EXT 设置为 3.3V

串口

管脚名	管脚号	IO	描述	电气特性	备注
MAIN_TXD	18	DO	主串口数据发送	DC 电平:VDD_EXT	不用则悬空
MAIN_RXD	17	DI	主串口数据接收	DC 电平:VDD_EXT	不用则悬空
MAIN_DTR	19	DI	主串口数据终端就绪，地有效，将模块从休眠状态唤醒	DC 电平:LDO_AON	不用则悬空
MAIN_RI	20	DO	主串口数据振铃信号	DC 电平:LDO_AON	不用则悬空
MAIN_CTS	22	DO	DTE 清除发送信号	DC 电平:VDD_EXT	不用则悬空,不能与 PIN5 同时使用
MAIN_RTS	23	DI	DTE 请求发送信号	DC 电	不用则悬空

				平:VDD_EXT	
MAIN_DCD	21	DO	主串口输出载波检测信号	DC 电平:VDD_EXT	不用则悬空，不能与 PIN49 同时使用
AUX_TXD	29	DO	扩展串口数据发送	DC 电平:VDD_EXT	不用则悬空
AUX_RXD	28	DI	扩展串口数据接收	DC 电平:VDD_EXT	不用则悬空
DBG_TXD	39	DO	调试串口输出	DC 电平:VDD_EXT	只能用于调试信息的监控
DBG_RXD	38	DI	调试串口输入	DC 电平:VDD_EXT	只能用于调试信息的监控

USB 接口

管脚名	管脚号	IO	描述	电气特性	备注
VBUS	61	DI	USB 插入唤醒	建议不超过 5.5V	内部电阻分压
USB_DP	59	IO	USB 数据差分信号		90 欧姆差分主控控制
USB_DN	60	IO	USB 数据差分信号		90 欧姆差分主控控制

USIM 接口

管脚名	管脚号	IO	描述	电气特性	备注
USIM_RST	12	DO	USIM 卡接口复位信号	1.8V/3.3V	
USIM_CLK	13	DO	USIM 卡接口时钟信号	1.8V/3.3V	
USIM_DAT	11	IO	USIM 卡接数据信号	1.8V/3.3V	
USIM_VDD	14	PO	USIM 卡接电源信号	1.8V/3.3V	
USIM_DET	79	DI	USIM 卡接插入检测型号	1.8V/3.3V	此功能缺省下关闭
USIM2_CLK	62	DO	USIM2 卡接口时钟信号	1.8V/3.3V	使用 USIM2 接口不能同时使用 PIN99,PIN81
USIM2_RST	63	DO	USIM2 卡接口复位信号	1.8V/3.3V	使用 USIM2 接口不能同时使用 PIN99,PIN80
USIM2_DAT	64	IO	USIM2 卡接电源信号	1.8V/3.3V	使用 USIM2 接口不能同时使用 PIN99,PIN55
USIM2_VDD	65	PO	USIM2 卡接电源信号	1.8V/3.3V	使用 USIM2 接口不能同时使用 PIN99

模数转换 ADC 接口

管脚名	管脚号	IO	描述	电气特性	备注
ADC0	9	AI	模数转换 ADC 通道 0	分辨率 12bits，量程范围 0-3.4V	若超量程需要外部电阻分压
ADC1	96	AI	模数转换 ADC 通道 1	分辨率 12bits，量程范围 0-3.4V	若超量程需要外部电阻分压
ADC2	77	AI	模数转换 ADC 通道 2	分辨率 12bits，量程范围 0-3.4V	若超量程需要外部电阻分压
ADC3	76	AI	模数转换 ADC 通道 3	分辨率 12bits，量程范围 0-3.4V	若超量程需要外部电阻分压

I2C 接口 (OPEN CPU)

管脚名	管脚号	IO	描述	电气特性	备注
I2C_SDA	66	OD	I2C 接口数据信号	DC 电平:VDD_EXT	不用则悬空
I2C_SCL	67	OD	I2C 接口时钟信号	DC 电平:VDD_EXT	不用则悬空

天线接口

管脚名	管脚号	IO	描述	电气特性	备注
LTE_ANT	35		4G LTE 射频天线接口		建议预留 π 形天线匹配，走线需 50 欧姆阻抗匹配

LED 指示灯接口

管脚名	管脚号	IO	描述	电气特性	备注
NET_STATUS	16	DO	网络状态指示灯	DC 电平:LDO_AON	不用则悬空
STATUS	25	DO	开机状态指示灯	DC 电平:LDO_AON	不用则悬空

SPI 接口(OPEN CPU)

管脚名	管脚号	IO	描述	电气特性	备注
SPI_CLK	86	IO	SPI 时钟信号	DC 电平:VDD_EXT	Opencpu 二次开发用
SPI_MOSI	85	IO	SPI 数据输出	DC 电	Opencpu 二次开发用

				平:VDD_EXT	
SPI_MISO	84	IO	SPI 数据输入	DC 电平:VDD_EXT	Opencpu 二次开发用,
SPI_CS	83	IO	SPI 片选信号	DC 电平:VDD_EXT	Opencpu 二次开发用,

普通 GPIO (休眠掉电)(OPEN CPU)

管脚名	管脚号	IO	描述	电气特性	备注
GPIO4	80	IO	通用 GPIO	DC 电平:VDD_EXT	Opencpu 二次开发用，不能与 PIN63 同时使用
GPIO3	54	IO	通用 GPIO	DC 电平:VDD_EXT	Opencpu 二次开发用
GPIO6	55	IO	通用 GPIO	DC 电平:VDD_EXT	Opencpu 二次开发用，不能与 PIN64 同时使用
GPIO7	56	IO	通用 GPIO	DC 电平:VDD_EXT	Opencpu 二次开发用,
GPIO15	57	IO	通用 GPIO	DC 电平:VDD_EXT	Opencpu 二次开发用
GPIO14	58	IO	通用 GPIO	DC 电平:VDD_EXT	Opencpu 二次开发用
GPIO5	81	IO	通用 GPIO	DC 电平:VDD_EXT	Opencpu 二次开发用，不能与 PIN62 同时使用

AGPIO (休眠保持)(OPEN CPU)

管脚名	管脚号	IO	描述	电气特性	备注
AGPIO3	99	IO	外通用 GPIO	DC 电平:LDO_AON	Opencpu 二次开发用,不能与 USIM2 同时使用
AGPIO5	106	IO	外通用 GPIO	DC 电平:LDO_AON	Opencpu 二次开发用
WAKEUP1	105	IO	外通用 GPIO	DC 电平:LDO_AON	Opencpu 二次开发用,不能与 PIN61 同时使用
AGPIOWU0	102	IO	外通用 GPIO	DC 电平:LDO_AON	Opencpu 二次开发用
AGPIOWU1	107	IO	外通用 GPIO	DC 电平:LDO_AON	Opencpu 二次开发用

保留管脚

管脚名	管脚号	IO	描述	电气特性	备注
RESERVED	其余管脚	DO	保留		保持悬空

*注:

1. AT 指令集暂不支持 GPIO 及特除功能进行操作,具体标注为上表中带“*”的管脚。

表格 4：IO 参数定义

类型	描述
IO	Input/Output
DI	Digital Input
DO	Digital Output
PI	Power Input
PO	Power Output
AI	Analog Input
AO	Analog Output
OD	Open Drain Output

5.3.1.2 GPIO 复用详细描述

I/O	Air780EP/Air780EPV PIN Name		Pad Name	paddr [7:2]	Powerup default	Alt Func0	Alt Func1	Alt Func2	Alt Func3	Alt Func4	Alt Func5	Alt Func6	Open CPU Ref	Notes
SWD（休眠掉电）	LDO_CTL	97	SWCLK0	11	I&PU	SWCLKA				GPIO16			GPIO16	
	IO_SEL	100	SWDIO0	12	I&PU	SWDIOA				GPIO17			GPIO17	
	I2C_SCL	67	SWCLK1	13	I&PU	SWCLKC		I2C0_SCL	I2C1_SCL	GPIO18	PWM0	KPC_R4	GPIO18	
	I2C_SDA	66	SWDIO1	14	I&PU	SWDIOC		I2C0_SDA	I2C1_SDA	GPIO19	PWM1	KPC_C4	GPIO19	
GPIO（休眠掉电）	USB_BOOT	82	GPIO0	15	I&PD	GPIO0						KPC_R4	GPIO0	boot
	UART1_CTS/AU_OUT	22\5	GPIO1	16	NI&NP	GPIO1		UART1_DCDn	UART1_RTSn	PWM1n	PWM0	KPC_R3	GPIO1	
	UART1_RTS	23	GPIO2	17	NI&NP	GPIO2		UART1_DTRn	UART1_CTSn	ONEW	PWM1	KPC_R2	GPIO2	
	CAM_MCLK	54	GPIO3	18	NI&NP	GPIO3	USP1_MCLK	USP1_WRX		ONEW	PWM2	KPC_C4	USP1_MCLK	CSPI_MCLK
	CAM_SPI_CLK	80	GPIO4	19	NI&NP	GPIO4	USP1_BCLK	I2C1_SDA	UART1_RTSn	USIM1_URSTn		KPC_R1	USP1_BCLK	CSPI_BCLK
	CAM_PWDN	81	GPIO5	20	NI&NP	GPIO5	USP1_LRCK	I2C1_SCL	UART1_CTSn	USIM1_UCLK		KPC_R0	GPIO5	CAM-PD
	CAM_SPI_D0	55	GPIO6	21	NI&NP	GPIO6	USP1_DIN	UART2_RXD	UART1_RTSn	USIM1_UIO		KPC_C3	USP1_DIN	CSPI_RX0
	CAM_SPI_D1	56	GPIO7	22	NI&NP	GPIO7	USP1_DOUT	UART2_TXD	UART1_CTSn	ONEW		KPC_C2	USP1_DOUT	CSPI_RX1
	SPI_CS	83	GPIO8	23	NI&NP	GPIO8	SPI0_SSn0	I2C1_SDA	UART2_RTSn	UART0_RTSn			SPI0_SSn0	Flash
	SPI_MOSI	85	GPIO9	24	NI&NP	GPIO9	SPI0_MOSI	I2C1_SCL	UART2_CTSn	UART0_CTSn			SPI0_MOSI	Flash
	SPI_MISO	84	GPIO10	25	NI&NP	GPIO10	SPI0_MISO		UART2_RXD				SPI0_MISO	Flash
	SPI_SCLK	86	GPIO11	26	NI&NP	GPIO11	SPI0_SCLK	SPI1_SSn1	UART2_TXD				SPI0_SCLK	Flash
	AUX_RXD	28	GPIO12	27	NI&NP	GPIO12	SPI1_SSn0	UART1_RTSn	UART2_RXD	USIM1_UIO	UART3_RTSn	KPC_C1	KPC_C1	KPC-键盘阵列
	AUX_TXD	29	GPIO13	28	NI&NP	GPIO13	SPI1_MOSI	UART1_CTSn	UART2_TXD	USIM1_URSTn	UART3_CTSn	KPC_C0	KPC_C0	KPC-键盘阵列
	CAM_I2C_SDA	58	GPIO14	29	NI&NP	GPIO14	SPI1_MISO	I2C0_SDA	UART3_RXD	USIM1_UCLK	PWM0	KPC_C3	I2C0_SDA	Camera
	CAM_I2C_SCL	57	GPIO15	30	NI&NP	GPIO15	SPI1_SCLK	I2C0_SCL	UART3_TXD	USP2_MCLK	PWM1	KPC_C2	I2C0_SCL	Camera
	DBG_RXD	38	GPIO16	31	NI&NP	GPIO16	UART0_RXD	I2C0_SDA					UART0	Log
	DBG_TXD	39	GPIO17	32	NI&NP	GPIO17	UART0_TXD	I2C0_SCL					UART0	Log
	MAIN_RXD	17	GPIO18	33	NI&NP	GPIO18	UART1_RXD						UART1_RXD	AT / SE
	MAIN_TXD	18	GPIO19	34	NI&NP	GPIO19	UART1_TXD						UART1_TXD	AT / SE
Audio（休眠掉电）	I2S_BCK	30	GPIO29	35	NI&NP	GPIO29	USP0_BCLK				PWM0		GPIO29	Codec / I2S
	I2S_LRCK	31	GPIO30	36	NI&NP	GPIO30	USP0_LRCK				PWM1		GPIO30	Codec / I2S
	I2S_DIN	32	GPIO31	37	NI&NP	GPIO31	USP0_DIN			USP1_MCLK	PWM2		GPIO31	Codec / I2S
	I2S_DOUT	33	GPIO32	38	NI&NP	GPIO32	USP0_DOUT						GPIO32	Codec / I2S
	I2S_MCLK	26	GPIO33	39	NI&NP	GPIO33	USP0_MCLK	USP0_WRX			PWM4		GPIO33	Codec / I2S
LSPI（休眠掉电）	LCD_CLK	53	GPIO34	40	NI&NP	GPIO34	USP2_BCLK	I2C0_SDA	UART3_RXD				USP2_BCLK	LSPI_DCX (CLK)
	LCD_CS	52	GPIO35	41	NI&NP	GPIO35	USP2_LRCK	I2C0_SCL	UART3_TXD				USP2_LRCK	LSPI_CSX
	LCD_RST/UART1_DCD	49\21	GPIO36	42	NI&NP	GPIO36	USP2_DIN	I2C1_SCL	UART0_RTSn				USP2_DIN	
	LCD_DOUT	50	GPIO37	43	NI&NP	GPIO37	USP2_DOUT	I2C1_SDA	UART0_CTSn				USP2_DOUT	LSP_SDA
	LCD_RS	51	GPIO38	44	NI&NP	GPIO38	USP2_MCLK	USP2_WRX					USP2_MCLK	LSP_WRX
AON GPIO（休眠保持）	AGPIOWU0	102	AGPIOWU0	45	NI&NP	GPIO20			PWM4n	FEM7		KPC_C2	KPC_C2	KPC-键盘阵列
	AGPIOWU1	107	AGPIOWU1	46	NI&NP	GPIO21				FEM6	PWM4	KPC_C3	KPC_C3	KPC-键盘阵列
	MAIN_DTR	19	AGPIOWU2	47	NI&NP	GPIO22			PWM4n	FEM5		KPC_C4	GPIO22	
	AGPIO3	99	AGPIO3	48	NI&NP	GPIO23			PWM1n	FEM4	PWM0	KPC_R4	KPC_R4	KPC-键盘阵列
	MAIN_RI	20	AGPIO4	49	NI&NP	GPIO24			PWM0n	FEM3	PWM1	KPC_R3	KPC_R3	KPC-键盘阵列
	AGPIO5	106	AGPIO5	50	NI&NP	GPIO25				FEM2	PWM2	KPC_R2	KPC_R2	KPC-键盘阵列
	STATUS	25	AGPIO6	51	NI&NP	GPIO26			PWM2n	FEM1		KPC_R1	KPC_R1	KPC-键盘阵列
	NET_STATUS	16	AGPIO7	52	NI&NP	GPIO27				FEM0	PWM4	KPC_R0	KPC_R0	KPC-键盘阵列
	LCD_TE	78	AGPIO8	53	NI&NP	GPIO28			PWM4n	ONEW			GPIO28	
AON（休眠保持）	WAKEUP0	101	WAKEUP0			WAKEUP0							WAKEUP0	
	VBUS	61	WAKEUP1			WAKEUP1							WAKEUP1	USB_DET
	USIM_DET	79	WAKEUP2			WAKEUP2							WAKEUP2	USIM_DET
	PWRKEY	7	PWRKEY			PWRKEY							PWRKEY	PWRKEY

* I&PU: input, pull-up ; I&PD: input, pull-down ; NI&NP: notinput, nopull

注意事项：

1 AONGPIO管脚休眠模式下可保持，保持高或低。

2 WAKEUP管脚固定电平1.8V，由于内部分压，内部上拉电平测量在1.1V左右

3 WAKEUP管脚内部上下拉非常弱，驱动能力<30uA.

4 系统休眠后外部只能通过WAKEUP管脚或者LPUART串口唤醒，AONGPIO虽然在休眠下不掉电，但是无法触发中断。

5 普通GPIO在休眠后均会处于高阻状态。

6 DBG_TX、DBG_RX默认功能为系统底层日志口，进行模块硬件设计时，在剩余功能引脚充足的前提下，避免使用DBG_TX和DBG_RX。
如果将此引脚复用为其他功能，则无法从DBG_TX和DBG_RX抓取系统日志。
在某些场景下，如果模块出现异常，无法抓到问题日志，只能通过硬件改版，引出DBG_TX、DBG_RX，抓取日志再进行分析。
包括但不限于以下两种场景：
1、低功耗场景：
在低功耗场景下，USB无法使用，只能通过DBG_TX、DBG_RX来抓取日志。
2、非低功耗场景：
模块接入USB时，工作正常，未接入USB时，工作异常的情况，只能通过DBG_TX、DBG_RX来抓取日志。

7 所有GPIO和wakeuppad都支持双边沿中断：
可以复用为wakeup的io，休眠以及唤醒状态下都能使用；
其余io唤醒状态下可用，休眠状态下不能使用；
wakeup io可以唤醒休眠，其余GPIO都不可以。

5.3.2 工作模式

下表简要的叙述了接下来几章提到的各种工作模式。

表格 5：工作模式

模式	描述	
正常工作	ACTIVE	连接正常工作。有数据或者语音或者短信交互。此模式下，模块功耗取决于环境信号的强弱，动态DTX控制以及射频工作频率。

	IDLE	MCU 内核时钟关闭，系统中断随时可以唤醒模块。模块注册上网络，没有数据，语音和短信交互。进入和退出IDLE模式均由系统自动管理
休眠模式	SLEEP1	休眠模式下。外设均会被关闭，大部分IO处于掉电状态，仅有AGPIO能够保持电平，功耗极大降低。通过AT+CSCLK=1或者AT+CSCLK=2进入此模式
深度休眠模式	SLEEP2	在休眠模式基础上，关闭SRAM，仅保持64KBSRAM（ASMB）区域存储必要信息。功耗进一步降低，在此模式下DeepSleep Timer仍然然可以运行。通过WAKUP管脚可以唤醒，但是软件需要重新初始化。AT版本不支持此休眠模式
超深度休眠模式	HIBERNATE	在休眠模式基础上，进一步关闭64KBSRAM（ASMB）区域，功耗最低。在此模式下DeepSleep Timer仍然然可以运行。通过WAKUP管脚可以唤醒，但是软件需要重新初始化。AT版本不支持此休眠模式
关机模式	OFF	此模式下PMU停止给基带和射频供电，软件停止工作，串口不通，但VBAT管脚依然通电

注意：

- 当模块进入休眠模式或深度休眠模式后，VDD_EXT 电源会掉电，相应电压域的GPIO以及串口（MAIN_UART 除外）均会处于掉电关闭状态，掉电 IO 口均无法响应中断，无法唤醒模块退出休眠模 式
- 模块进入休眠状态后只能通过以下管脚中断唤醒退出休眠模式。

管脚名	序号	功能	描述
PWRKEY	7	开机关机	通拉低开机管脚触发中断
MAIN_TXD/RXD	17,18	主串口	通过给串口发数据唤醒模块
MAIN_DTR	19	模块唤醒管脚	拉低触发中断唤醒
USIM_DET	79	SIM 卡插拔检测	拉低触发中断唤醒
VBUS	61	USB 插入唤醒	USB插入，或拉高触发
WAKEUP0	101	外部中断唤醒	拉低触发中断唤醒
AGPIOWU0	102	外部中断唤醒	拉低触发中断唤醒
AGPIOWU1	107	外部中断唤醒	拉低触发中断唤醒

5.3.3 电源供电

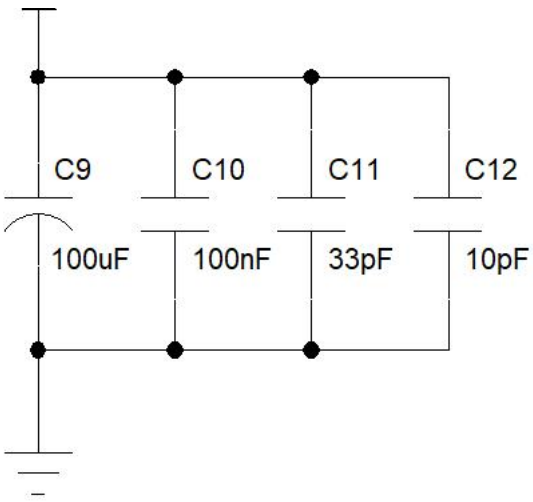
管脚名	管脚号	描述
VBAT	42,43	模块基带电源，供电范围3.3V~4.3V

5.3.3.1 模块电源工作特性

在模块应用设计中，电源设计是很重要的一部分。由于 LTE 射频工作时最大峰值电流高达 1.5A，在最大发 射功率时会有约 700mA 的持续工作电流，电源必须能够提供足够的电流，不然有可能会引起供电电压的跌落甚 至模块直接掉电重启。

5.3.3.2 减小电压跌落

模块电源 VBAT 电压输入范围为 3.3V~4.3V，但是模块在射频发射时通常会在 VBAT 电源上产生电源电压跌落 现象，这是由于电源或者走线路径上的阻抗导致，一般难以避免。因此在设计上要特别注意模块的电源设计， 在 VBAT 输入端，建议并联一个低 ESR(ESR=0.7Ω)的 100uF 的钽电容，以及 100nF、33pF、10pF 滤波电容，VBAT 输 入端参考电路如图 4 所示。并且建议 VBAT 的 PCB 走线尽量短且足够宽，减小 VBAT 走线的等效阻抗，确保在最大 发射功率时大电流下不会产生太大的电压跌落。建议 VBAT 走线宽度不少于 1mm，并且走线越长，线宽越宽。



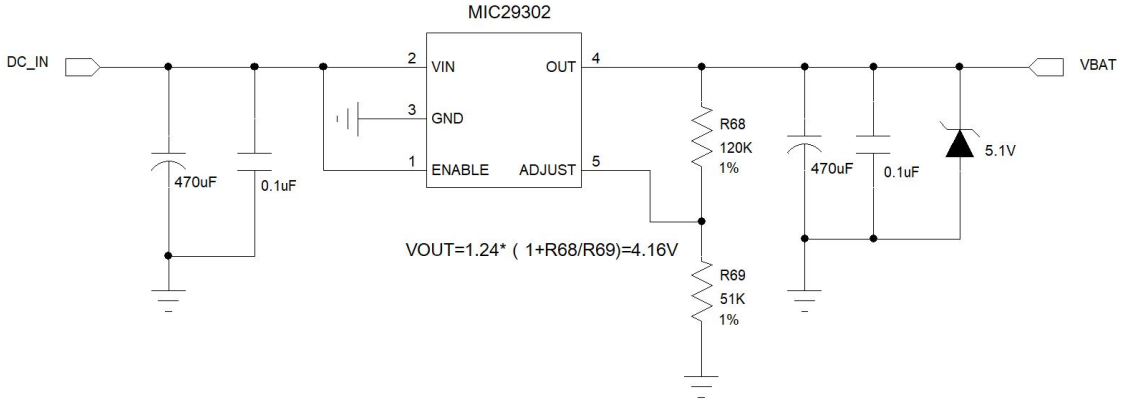
图表 3：VBAT 输入参考电路

5.3.3.3 供电参考电路

电源设计对模块的供电至关重要，必须选择能够提供至少 1A 电流能力的电源。若输入电压跟模块的供电 电压的压差小于 2V，建议选择 LDO 作为供电电源。若输入输出之间存在的压差大于 2V，则推荐使用开关电源转 换器以提高电源转换效率。

LDO 供电：

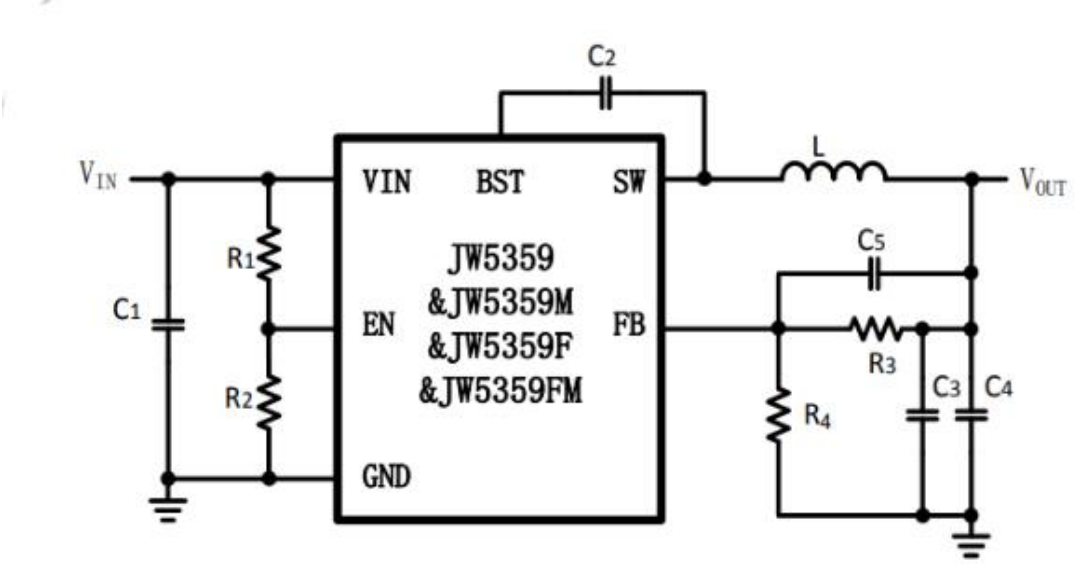
下图是 5V 供电的参考设计，采用了 Micrel 公司的 LDO，型号为 MIC29302WU 。它的输出电压是 4.16V，负载 电流峰值到 3A 。为确保输出电源的稳定，建议在输出端预留一个稳压管，并且靠近模块 VBAT 管脚摆放。建议 选择反向击穿电压为 5.1V，耗散功率为 1W 以上的稳压管。



图表 4：供电输入参考设计

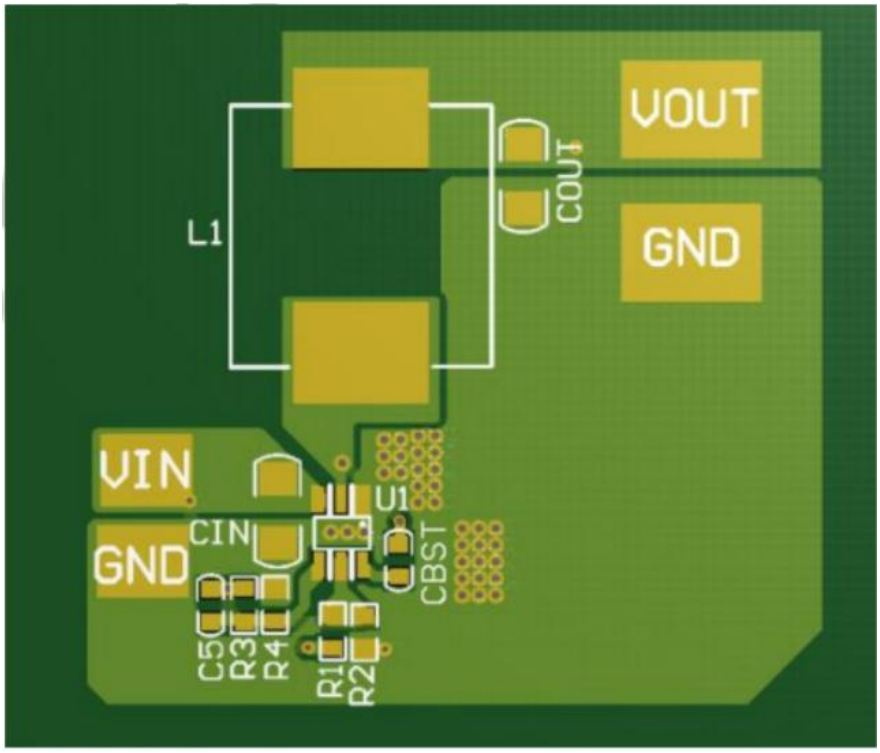
DC-DC 供电：

下图是 DCDC 开关电源的参考设计，采用的是杰华特公司的 JW5359M 开关电源芯片它的最大输出电流在 2A，同时输入电压范围 4.7V~18V 注意 C1 的选型要根据输入电压来选择耐压值。如果输入电压大于 12V 需要在输入端 C1 边上再加一颗 470uF 的电解电容。



图表 6：DCDC 供电输入参考设计

PCB 布局必须按下图供应商推荐布局做



5.3.4 开关机

5.3.4.1 开机

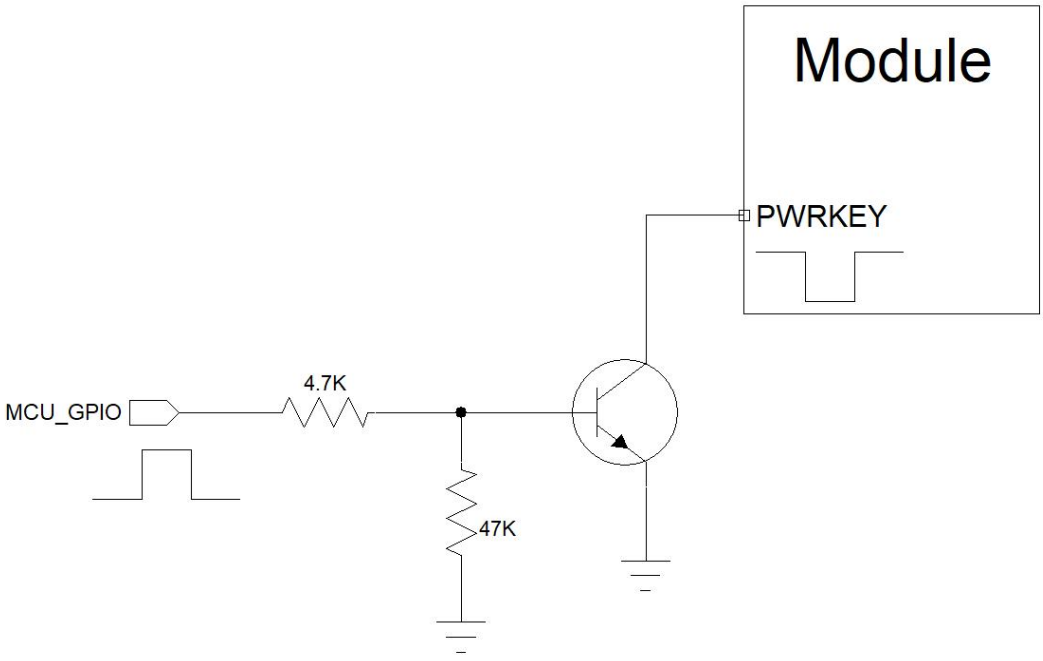
管脚名	类型	序号	描述
PWRKEY	DI	7	模块开机/关机控制脚， 内部上拉到VBAT

在 VBAT 供电后，可以通过如下两种方式来触发 Air780EP 开机：

1. 按键开机： PWRKEY 管脚通过轻触按键连接到地，按键按下 1 秒以上实现开机。
2. 上电开机：将 PWRKEY 管脚直接短接到地，VBAT 上电后就可以实现开机。

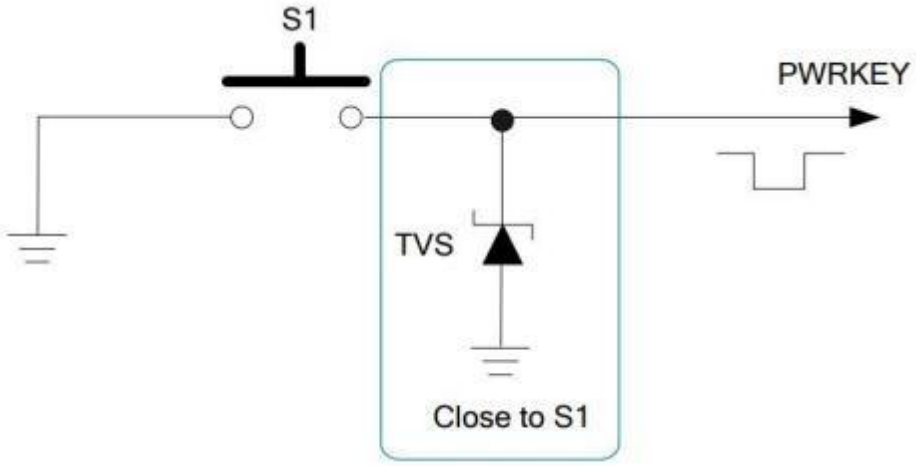
5.3.4.1.1 PWRKEY 管脚开机

VBAT 上电后，可以通过 PWRKEY 管脚启动模块，把 PWRKEY 管脚拉低 1 秒以上之后模块会进入开机流程，软 件会检测 VBAT 管脚电压，若 VBAT 管脚电压大于软件设置的开机电压（3.3V），会继续开机动作直至系统开机完成；否则，会停止执行开机动作，系统会关机，开机成功后 PWRKEY 管脚可以释放。可以通过检测 VDD_EXT 管脚的电平来判别模块是否开机。推荐使用开集驱动电路来控制 PWRKEY 管脚。下图为参考电路：



图表 5：开集驱动参考开机电路

另一种控制 PWRKEY 管脚的方法是直接使用一个按钮开关。按钮附近需放置一个 TVS 管用以 ESD 保护。下图为参考电路：



图表 6：按键开机参考电路

5.3.4.1.2 上电开机

将模块的 PWRKEY 直接接地可以实现上电自动开机功能。需要注意，在上电开机模式下，将无法关机。

对于用电池供电的应用场景不建议用 **PWRKEY** 接地的上电自动开机方式。

5.3.4.2 关机

以下的方式可以关闭模块：

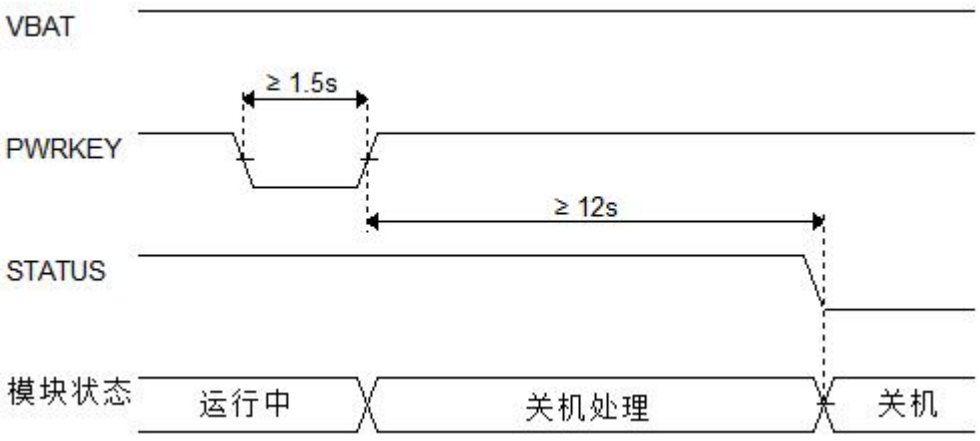
- 正常关机：使用 PWRKEY 管脚关机
- 正常关机：通过 AT 指令 AT+CPOWD 关机

5.3.4.2.1 PWRKEY 管脚关机

PWRKEY 管脚拉低 1.5s 以上时间，模块会执行关机动作。

关机过程中，模块需要注销网络，注销时间与当前网络状态有关，经测定用时约 2s~12s，因此建议延长 12s 后再进行断电或重启，以确保在完全断电之前让软件保存好重要数据。

时序图如下：



5.3.4.2.2 低电压自动关机

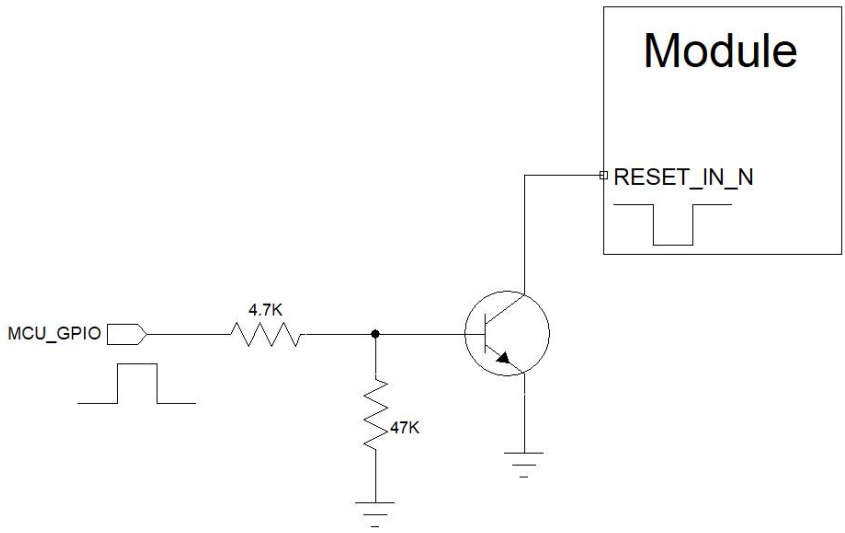
模块在运行状态时当VBAT 管脚电压低于模块工作的最低工作电压时（默认设置3.3V），软件会执行关机 动作关闭模块，以防低电压状态下运行出现各种异常。

5.3.4.3 复位

管脚名	类型	序号	电压域	描述
RESET_N	DI	15	-	模块复位输入，低有效；无需外部上拉

RESET_N 引脚可用于使模块复位。拉低RESET_N 引脚100ms 以上可使模块复位。RESET_N 信号对干 扰比较敏感， 因此建议在模块接口板上的走线应尽量的短，且需包地处理。

参考电路：



注意：

1. 复位功能建议仅在AT+CPOWD 和PWRKEY 关机失败后使用。

2. RESET_N 复位管脚拉低释放后，模块会处于硬件关机状态，如果想要重启功能，需要在RESET_N 复位后重新拉低POWERKEY 管脚进行开机动作。

5.3.5 串口

模块提供了 4 个通用异步收发器：主串口 MAIN_UART 、AUX_UART 、DBG_UART 、UART3。

5.3.5.1 MAIN_UART

表格 6：MAIN_UART 管脚定义

管脚名	类型	序号	电压域	描述
MAIN _TXD	DO	18	VDD_EXT	MAIN_UART 发送数据
MAIN _RXD	DI	17	VDD_EXT	MAIN_UART 接收数据
MAIN _CTS	DO	22	VDD_EXT	流控管脚，MAIN_UART 请求发送数据
MAIN _RTS	DI	23	VDD_EXT	流控管脚，MAIN_UART 清除发送

对于AT 开发方式，3.5.1. MAIN_UART 用来进行AT 指令通讯。MAIN_UART 支持固定波特率，不支持自适 应波特率

在默认情况下，模块的硬件流控是关闭的。当客户需要硬件流控时，管脚 **RTS,CTS** 必须连接到客户端，**AT** 命令**“AT+IFC=2,2”**可以用来打开硬件流控。**AT** 命令**“AT+IFC=0,0”**可以用来关闭流控。具体请参考《AirM2M 无线 模块AT 命令手册》。

MAIN_UART 在休眠状态下保持的功能，能够唤醒模块

MAIN_UART 的特点如下：

- 包括数据线 TXD 和 RXD ，硬件流控控制线 RTS 和 CTS。
- 8 个数据位，无奇偶校验，一个停止位。
- 硬件流控默认关闭。
- 用以 AT 命令传送，数传等。
- 支持波特率如下： 600,1200,2400,4800, 14400,9600, 19200,38400,57600, 115200,230400,460800,921600bps

注意：

MAIN_UART 在开机过程中短时会输出固定调试信息

5.3.5.2 AUX_UART

表格 7：AUX_UART 管脚定义

管脚名	类型	序号	电压域	描述
AUX_TXD	DO	29	VDD_EXT	AUX_UART 发送数据
AUX_RXD	DI	28	VDD_EXT	AUX_UART 接收数据

AUX_UART 为辅助串口，不支持 AT 指令交互，用于某些外设通信，如对接 GNSS 等。
AUX_UART 休眠后会关闭，无法通过给 AUX_UART 发送数据进行唤醒。

5.3.5.3 DBG_UART

管脚名	类型	序号	电压域	描述
DBG_TXD	DO	39	VDD_EXT	调试串口，输出 AP log，
DBG_RXD	DI	38	VDD_EXT	调试串口，接收调试指令

DBG_UART 用来软件调试时输出AP trace，建议预留测试点。

DBG_UART 在开机过程中短时会输出固定调试信息

5.3.5.4 UART3

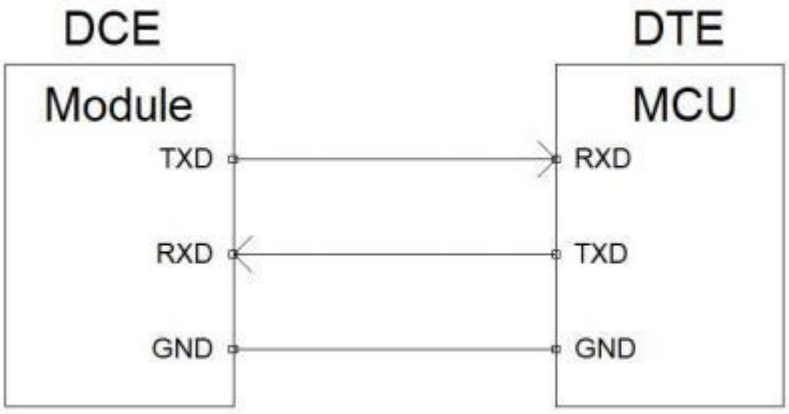
管脚名	类型	序号	电压域	描述
UART3_TXD	DO	52	VDD_EXT	与 LCD_CS 复用
UART3_RXD	DI	53	VDD_EXT	与 LCD_CLK 复用

如果使用不到 LCD 功能，可以再复用一组 UART3

5.3.5.5 串口连接方式

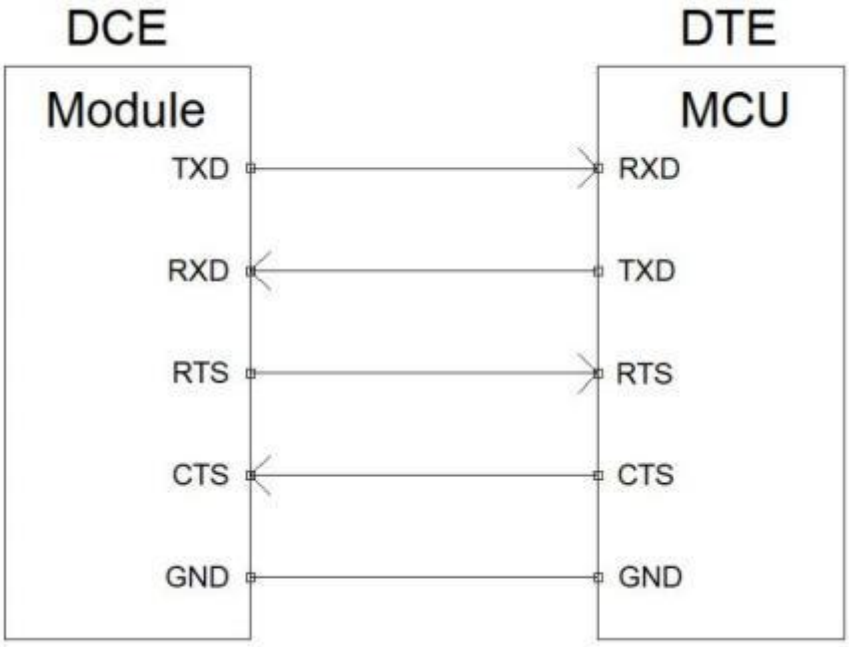
串口的连接方式较为灵活，如下是三种常用的连接方式。

三线制的串口请参考如下的连接方式：



图表 7：串口三线制连接方式示意图

带流控的串口连接请参考如下电路连接，此连接方式可提高大数据量传输的可靠性，防止数据丢失。

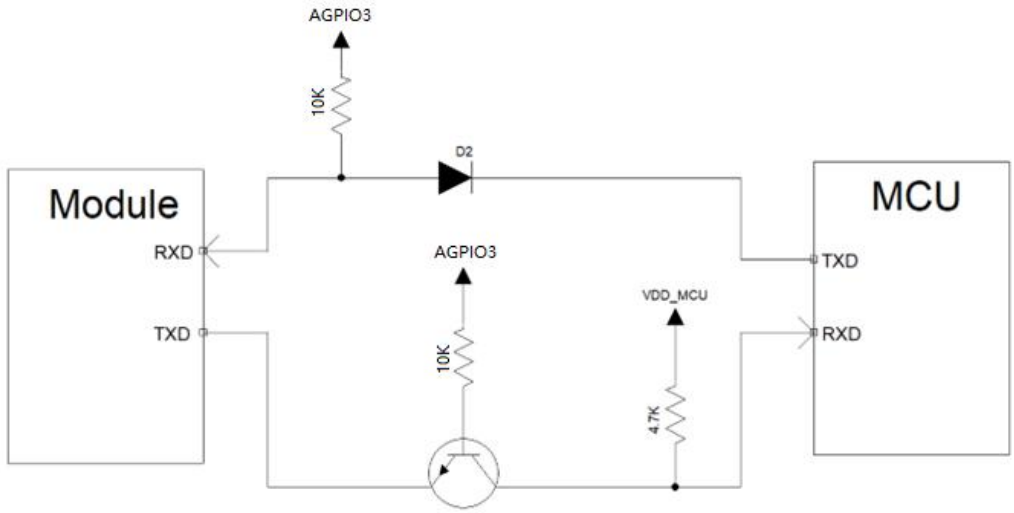


图表 8：带流控的串口连接方式示意图

5.3.5.6 串口电压转换

Air780EP 模块的串口电平为1.8V 或3.3V，通过PIN100 IO_SEL 配置IO 口电平，能够满足大部分外设， 主控的串口直接需求，但是如果要和5V 或者以上的MCU 或其他串口外设通信，那就必须要加电平转换 电路：

电平转换参考电路如下：



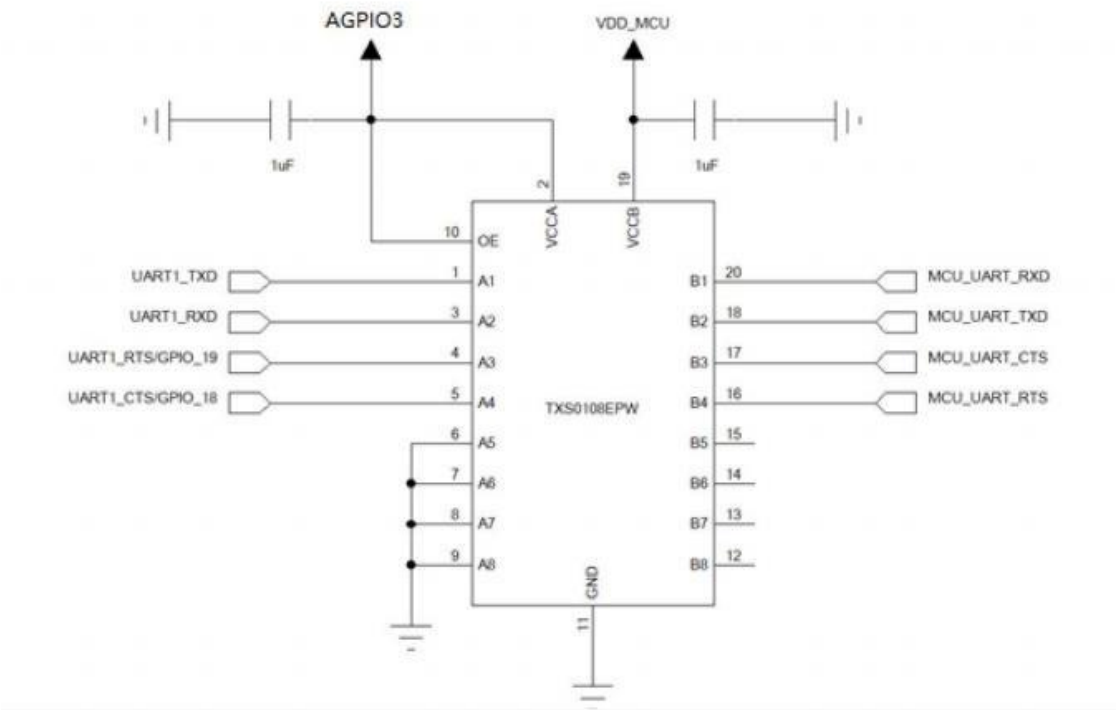
注意

- 此电平转换电路不适用波特率高于 460800 bps 的应用。
- 由于休眠状态下 VDD_EXT 会掉电，因此如果在需要休眠的应用场景用 VDD_EXT 做串口电平转换上拉的话， 会导致模块无法休眠的情况。因此在需要进入休眠的场景下，建议用AGPIO3 进行上拉。
- D2 必须选用低导通压降的肖特基二极管。

肖特基二极管以及 NPN 三极管的推荐型号如下：

物料名称	型号	厂商	描述
肖特基二极管	RB521S-30	江苏长电	Schottky Diode;30V;200mA;SOD523;1.6*0.8*0.6mm
	PSB521S-30	上海智晶	Schottky Diode;30V;200mA;SOD523;1.6*0.8*0.6mm
	LRB521S-30T1G	LRC	Schottky Diode;30V;200mA;SOD523;1.6*0.8*0.6mm
	PSBD521S-30	Prisemi	Schottky Diode;30V;200mA;SOD523;1.6*0.8*0.6mm
NPN 三极管	MMBT3904	江苏长电	Transistor;NPN;40V;200mA;SOT23;1.1mm;ROHS
	MMBT3904	上海智晶	Transistor;NPN;40V;200mA;SOT23;1.1mm;ROHS
	LMBT3904LT1G	LRC	Transistor;NPN;40V;200mA;SOT23;1.1mm;ROHS

对于波特率高于460800bps 的应用，可以通过外加电平转换芯片来实现电压转换，参考电路如下：



此电路采用的是电平转换芯片是TI 的TXS0108E，8 位双向电压电平转换器，适用于漏极开路和推挽应用， 最大支持速率：

推挽：110Mbps

开漏：1.2Mbps

5.3.6 USB 接口

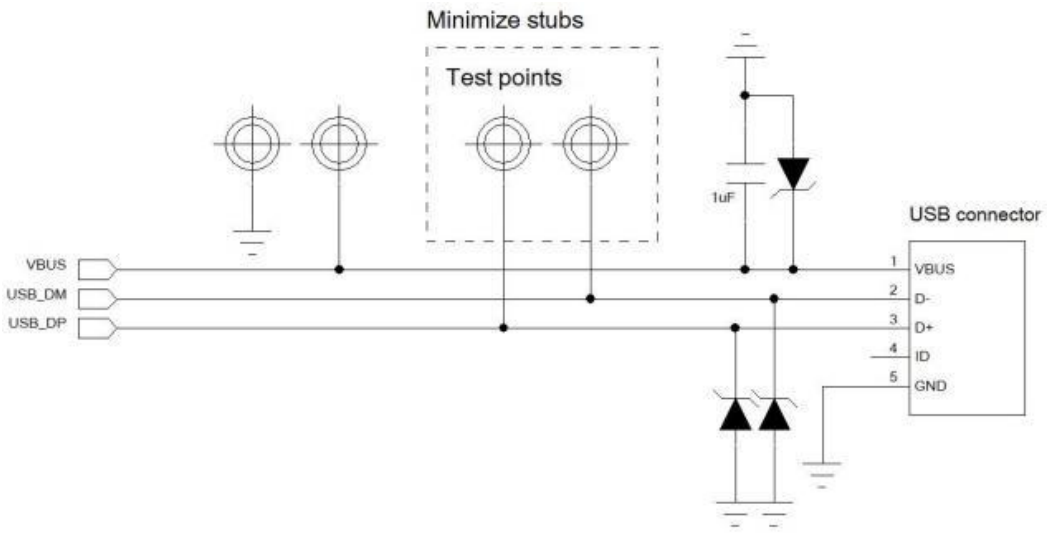
Air780EP 的 USB 符合 USB2.0 规范，支持高速（480Mbps）、全速（12Mbps）模式和低速（1.2Mbps） 模式。USB 接口可用于AT 命令传送，数据传输，软件调试和软件升级。

表格 8：USB 管脚定义

管脚名	类型	序号	描述
USB_DP	IO	59	USB 差分信号正，走线需控制 90 欧姆差分阻抗
USB_DM	IO	60	USB 差分信号负，走线需控制 90 欧姆差分阻抗

VBUS	DI	61	USB 插入唤醒，模块内部电阻分压。（非必须）

USB 接口参考设计电路如下：



图表9：USB 接口参考设计

注意事项如下：

1. USB 走线需要严格按照差分线控制，做到平行和等长；
2. USB 走线的阻抗需要控制到差分90 欧姆；
3. 需要尽可能的减少 USB 走线的stubs，减少信号反射；USB 信号的测试点最好直接放在走线上以 减少stub；
4. 尽可能的减少 USB 走线的过孔数量；
5. 在靠近 USB 连接器或者测试点的地方添加TVS 保护管，由于 USB 的速率较高，需要注意TVS 管 的选型，保证选用的TVS 保护管的寄生电容小于 1pF
6. VBUS 作为 USB 插入唤醒作用，并不直接参与 USB 插入检测，非必须，在不需要 USB 插入唤醒的 场景也可以不接

5.3.7 USB 下载模式

管脚名	类型	序号	电压域	描述
USB_BOOT	DI	82	**LDO_AON**	在开机之前上拉到 VDD_EXT，模块会强行进入 USB 下 载模式，USB_BOOT 须留测试点，方便后续升级软件

Air780EP 模块进入 USB 下载模式：

1. 在开机之前，把 USB_BOOT 上拉到VDD_EXT

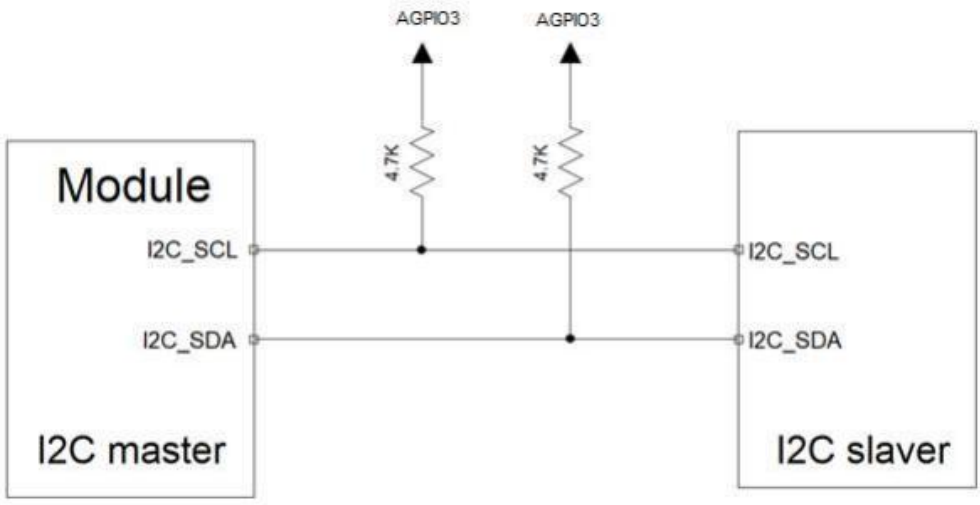
5.3.8 I2C

管脚名	类型	序号	电压域	描述
I2C_SCL *	IO	67	VDD_EXT	I2C 时钟信号，用作 I2C 时需外加上拉
I2C_SDA *	IO	66	VDD_EXT	I2C 数据信号，用作 I2C 时需外加上拉

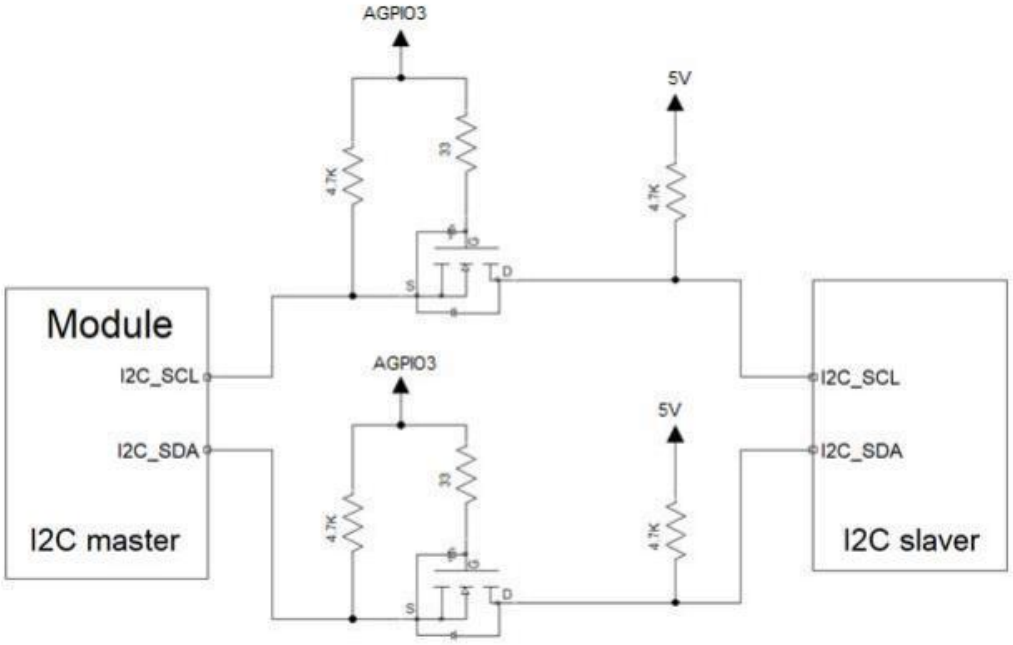
Air780EP 可支持两路I2C 接口：

- 兼容 Philips I2C 标准协议
- 支持 Fast mode （400Kbps）和 Slow mode（100Kbps）
- 只支持 master 模式，不支持 slaver 模式
- 可通过软件来配置内部的上拉电阻，1.8K 或者 20K
- 理论上最多可支持 127 个从设备

I2C 的参考电路如下：



Air780EP 的I2C 接口电压是1.8V/3.3V 可配置，通过PIN100 IO_SEL 配置IO 口电平，能够满足大部分 外设的直接需求，但是如果要和5V 或者以上电平的外设通信，那就必须要加电平转换电路：



上图推荐用 AGPIO3 上拉，在不需要模块进入休眠的场景或者允许休眠掉电的场景也可以用 VDD_EXT 上拉

电平转换用的 NMOS 管必须选用结电容小于 50pF 的型号，推荐型号如下：

物料名称	型号	厂商	描述
NMOS	BSS138	江苏长电	N 沟道,50V,0.22A,SOT-23,ROHS
	BSS138	UMW(友台半导体)	N 沟道,50V,0.3A,SOT-23,ROHS

5.3.9 SIM 卡接口

Air780EP 支持 2 路 SIM 卡接口，支持 ETSI 和 IMT-2000 卡规范，支持 1.8V 和 3.0V USIM 卡。以满足双 SIM 卡切换的需求。

5.3.9.1 SIM 接口

下表介绍了 SIM 接口的管脚定义。

表格 9：SIM 卡接口管脚定义

接口	管脚名	序号	描述
SIM1	USIM_VDD	14	SIM 卡供电电源，最大供电电流 10mA。 模块可以自动识别 1.8V 或者 3V(U)SIM 卡。
	USIM_RST	12	SIM 卡复位信号
	USIM_DAT	11	SIM 卡数据信号
	USIM_CLK	13	SIM 卡时钟信号
	USIM_DET	79	SIM 卡插拔检测
SIM2	USIM2_VDD	65	SIM2 卡供电电源，最大供电电流 10mA。 模块可以自动识别 1.8V 或者 3V(U)SIM 卡。
	USIM2_RST	63	SIM2 卡复位信号
	USIM2_DAT	64	SIM2 卡数据信号
	USIM2_CLK	62	SIM2 卡时钟信号

5.3.9.2 双 SIM 卡切换说明

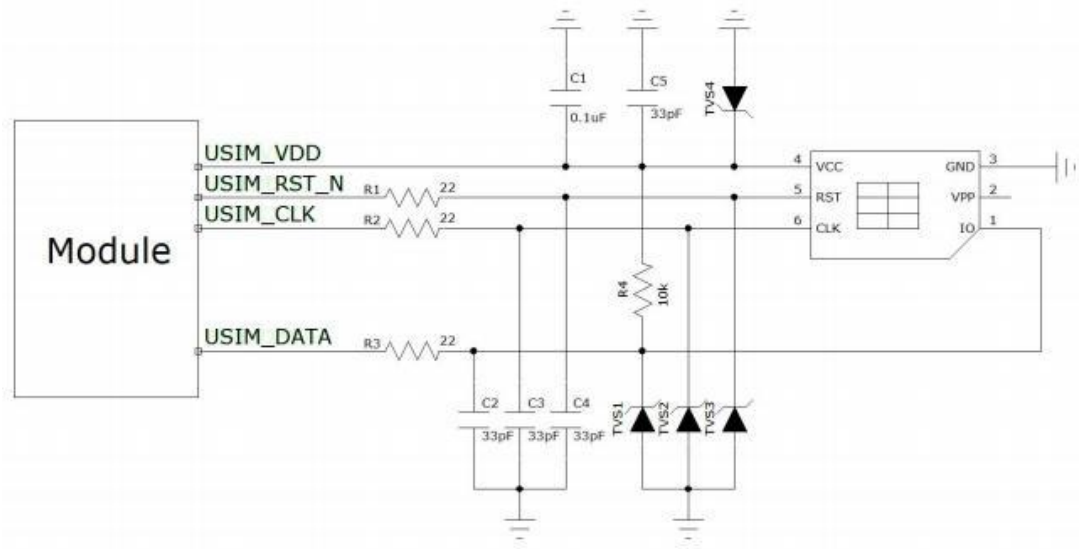
Air780EP 支持双卡单待，同一时间只能使用其中一个SIM 通道。可以通过相应的AT 指令进行SIM 卡 通道切换：或者选择自动切换功能，系统会根据外部网络信号强度进行自动切换（自动切换功能通常用
于同时使用不同运营商SIM 卡的场景）。具体指令请参考《4G 模块AT 命令手册》

注意：

- 模块开机会默认检测SIM1 通道，在SIM1 通道检测到SIM 卡不在位的情况下才会去检测SIM2 通 道。
- USIM_DET 信号为SIM 卡插拔检测管脚，上下边沿电平触发中断，触发系统进行SIM1 通道的卡 在位检测。而SIM2 通道不支持SIM 卡插拔检测。
- 对于内置贴片 SIM 卡的双卡应用场景，如网络摄像头（IPC）场景，建议将贴片 SIM 卡置于SIM2 通道，外置插拔SIM 卡座置于SIM1 通道，以实现优先使用外置插拔SIM 卡的效果。

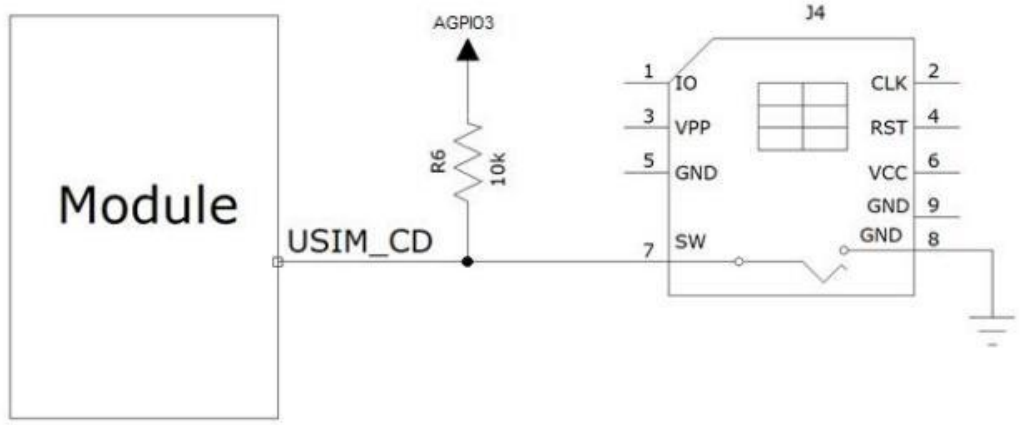
5.3.9.3 SIM 接口参考电路

下图是 SIM 接口的参考电路，使用 6pin 的SIM 卡座。



图表 10：使用 6pin SIM 卡座参考电路图(SIM)

如果需要用到 sim 卡在位检测，推荐电路如下。



图表 11：使用带检测 PIN SIM 卡座参考电路图

在 SIM 卡接口的电路设计中，为了确保 SIM 卡的良好的功能性能和不被损坏，在电路设计中建议遵循以下设计 原则：

1. SIM 卡座与模块距离摆件不能太远，越近越好，尽量保证 SIM 卡信号线布线不超过 20cm。
2. SIM 卡信号线布线远离 RF 线和 VBAT 电源线。
3. 为了防止可能存在的 USIM_CLK 信号对 USIM_DATA 信号的串扰，两者布线不要太靠近，在两条走线之间增 加地屏蔽。且对 USIM_RST_N 信号也需要地保护。
4. 为了保证良好的 ESD 保护，建议加 TVS 管，并靠近 SIM 卡座摆放。选择的 ESD 器件

寄生电容不大于 50pF 。在 模块和 SIM 卡之间也可以串联 22 欧姆的电阻用以抑制杂散 EMI ， 增强 ESD 防护。SIM 卡的外围电路必须尽量 靠近 SIM 卡座。

5. 在需要模块进入休眠的场景 SIM_DET 禁止用 VDD_EXT 上拉，否则会造成无法休眠的问题。建议用 AGPIO3 上 拉，或者用 SIM_DET IO 内部上拉的方式

5.3.10 LDO 输出

管脚名	类型	序号	描述
VDD_EXT	PO	24	LDO 输出，3.3V,1.8V 可调，配合 PIN 100 外接下拉电阻做硬件配置,不可作为唤醒 IO 关机和串口上拉。

注意：

- VDD_EXT 作为大部分 IO 的内部供电，在模块进入休眠后会掉电关闭，以降低功耗。
- VDD_EXT 在模块休眠期间，会随着模块底层频繁唤醒做网络交互以维持网络注册状态。客观上造成 VDD_EXT 在模块休眠期间输出周期不等的高脉冲，如果用 VDD_EXT 做某些 IO 如唤醒 IO 管脚，或者 MAIN_UART 的上拉，就会造成频繁触发 IO 中断导致模块无法进入休眠。

- 在需要休眠的场景可以用AGPIO3 进行上拉

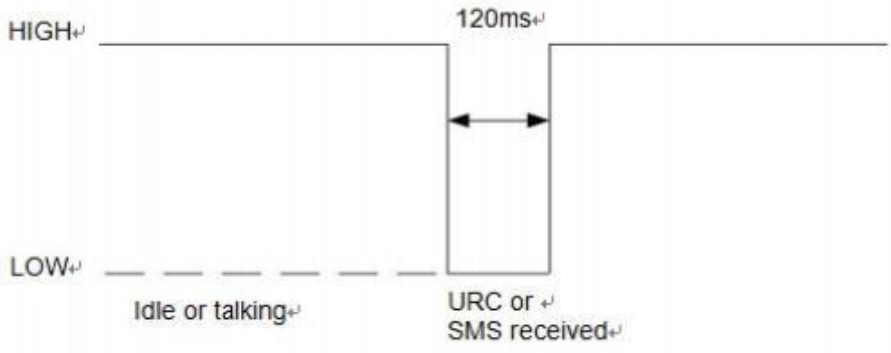
5.3.11 功能管脚

5.3.11.1 MAIN_RI

管脚名	类型	序号	电压域	作用
MAIN_RI	DO	20	**LDO_AON**	振铃信号，唤醒输出管脚，用于唤醒 AP

表格 10：MAIN_RI 信号动作

状态	MAIN_RI 应答
待机	高电平
数据传输	变为低电平，之后： 1.数据连接建立时变为高电平 2.收到短信时变为高电平
短信	当收到新的短信，MAIN_RI 变为低电平，持续 120ms ，再变为高电平
URC	某些 URC 信息可以触发 MAIN_RI 拉低 120ms



图表 15：收到 URC 信息或者短信时 MAIN_RI 时序

5.3.11.2 MAIN_DTR

管脚名	类型	序号	电压域	作用
MAIN_DTR	DI	19	**LDO_AON**	模块休眠唤醒管脚，拉高允许模块进入休眠模式；在休 眠模式下，拉

				低可唤醒模块

模块支持两种睡眠模式：

睡眠模式 1：发送 AT+CSCLK=1 ，通过 MAIN_DTR 管脚电平控制模块是否进入睡眠

睡眠模式 2：发送 AT+CSCLK=2 ，模块在串口空闲一段时间后自动进入睡眠

具体参阅 3.20.2 睡眠模式

5.3.11.3 状态指示灯

Air780EP 用一个管脚来指示开机状态，用两个管脚信号来指示网络的状态。如下两表分别描述了管脚定义 和不同网络状态下的逻辑电平变化：

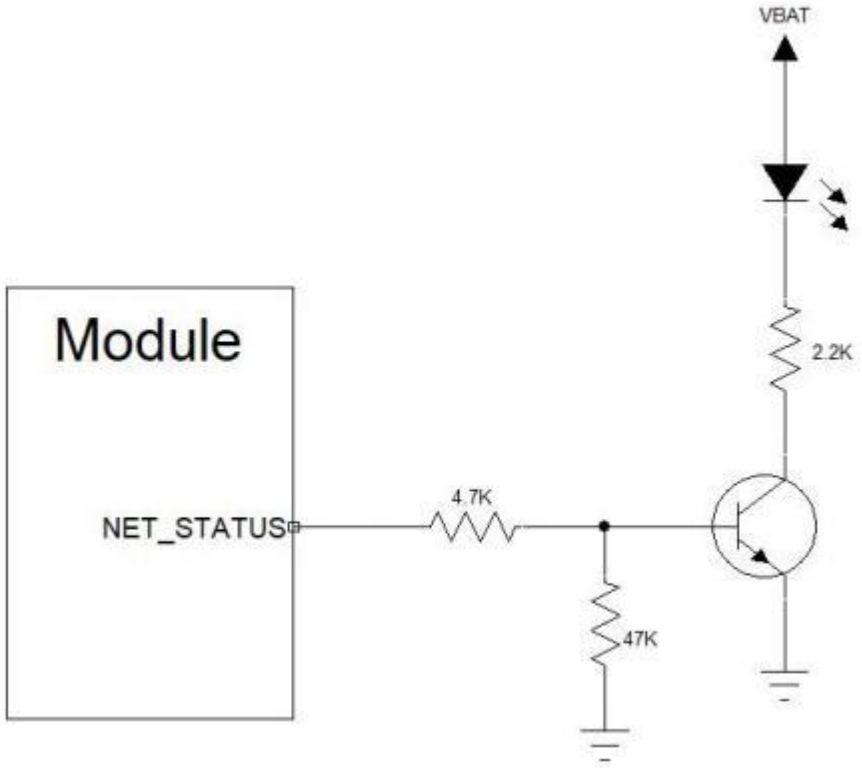
表格 11：网络指示管脚定义

管脚名	类型	序号	电压域	作用
NET_STATUS	DO	16	**LDO_AON**	指示模块的网络运行状态

表格 12：指示网络管脚的工作状态

状态	管脚工作状态	网络状态
NET_STATUS	亮 0.2 秒，灭 1.8 秒	搜网状态
	亮 1.8 秒，灭 0.2 秒	待机
	亮 0.125 秒，灭 0.125 秒	数据传输状态 注意：该状态提示仅限于 PPP 拨号成功或者 AT 指令主动激活 PDP 成 功，RNDIS 联网成功

指示灯参考电路如下图所示：



图表 16：指示灯参考电路

5.3.12 省电功能

根据系统需求，有两种方式可以使模块进入到低功耗的状态。对于 AT 版本使用“AT+CFUN”命令可以使模块 进入最少功能状态。

5.3.12.1 最少功能模式/飞行模式

最少功能模式可以将模块功能减少到最小程度，此模式可以通过发送“AT+CFUN=<fun>”命令来设置。<fun> 参数可以选择 0，1，4。

- 0：最少功能（关闭 RF 和 SIM 卡）；
- 1：全功能（默认）；
- 4：关闭 RF 发送和接收功能；

如果使用“AT+CFUN=0”将模块设置为最少功能模式，射频部分和 SIM 卡部分的功能将会关闭。而串口依然 有效，但是与射频部分以及 SIM 卡部分相关的 AT 命令则不可用。

如果使用“AT+CFUN=4”设置模块，RF 部分功能将会关闭，而串口依然有效。所有与 RF 部分相关的 AT 命令不 可用。

模块通过“AT+CFUN=0”或者“AT+CFUN=4”设置以后，可以通过“AT+CFUN=1”命令设置返回到全功能状态。

5.3.12.2 睡眠模式（慢时钟模式）

5.3.12.2.1 串口应用

串口应用下支持两种睡眠模式：

- 睡眠模式 1：通过 MAIN_DTR 管脚电平控制模块是否进入睡眠
- 睡眠模式 2：模块在串口空闲一段时间后自动进入睡眠

5.3.12.2.1.1 睡眠模式 1

开启条件：

发送 AT 指令 AT+CSCLK=1

模块进入睡眠：

控制 MAIN_DTR 脚拉高，模块会进入睡眠模式 1

模块退出睡眠：

拉低 MAIN_DTR 脚 50ms 以上，模块会退出睡眠模式可以接受 AT 指令

模块在睡眠模式 **1** 时的软件功能：

不响应 AT 指令，但是收到数据/短信/来电会有 URC 上报

HOST 睡眠时，模块收到数据/短信/来电如何唤醒 **HOST**：

MAIN_RI 信号

5.3.12.2.1.2 睡眠模式 2

开启条件：

发送 AT 指令 AT+CSLCK=2

模块进入睡眠：

串口空闲超过 AT+WAKETIM 配置的时间（默认 5s），模块自动进入睡眠模式 2

模块退出睡眠：

串口连续发送 AT 直到模块回应时即退出睡眠模式 2

模块在睡眠模式 **2** 时的软件功能：

不响应 AT 指令，但是收到数据/短信/来电会有 URC 上报

HOST 睡眠时，模块收到数据/短信/来电如何唤醒 **HOST**：

MAIN_RI 信号

5.3.12.2.2 USB 应用

开启条件：

USB HOST 必须支持 USB suspend/resume

模块进入睡眠：

HOST 发起 USB suspend

模块退出睡眠：

HOST 发起 USB resume

HOST 睡眠时，模块收到数据**/**短信**/**来电如何唤醒 **HOST**：
MAIN_RI 信号

5.3.13 模式切换汇总

表格 13：模式切换汇总

当前模式	下一模式		
	关机	正常模式	睡眠模式
关机	/	使用 PWRKEY 开机	/
正常模式	使用 PWRKEY 管脚，或 VBAT 电压低于关机电压	/	软件调用睡眠接口，AT 版本不做动作 30s 自动休眠
睡眠模式	使用 PWRKEY 管脚，或 VBAT 电压低于关机电压	GPIO 管脚中断、定时器、接收短信或网络数据	/

具体的功耗数据请查询功耗章节。

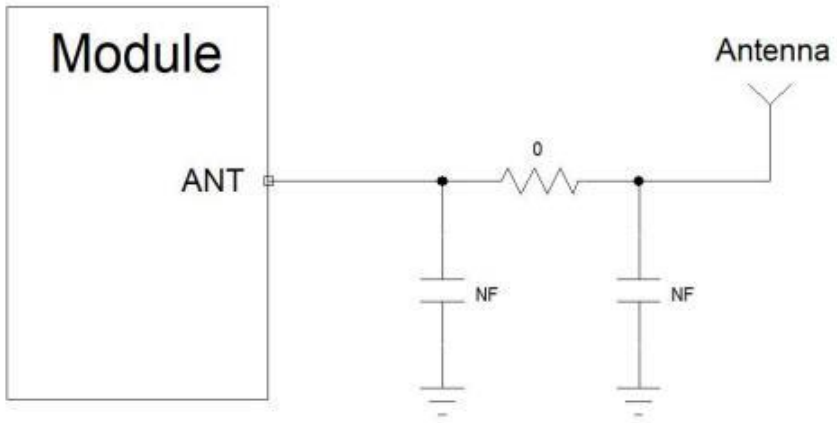
5.4 射频接口

天线接口管脚定义如下：

表格 14：RF_ANT 管脚定义

管脚名	序号	描述
LTE_ANT	35	LTE 天线接口

5.4.1 射频参考电路



图表 17：射频参考电路

注意：

- 连接到模块RF天线焊盘的RF走线必须使用微带线或者其他类型的 RF走线，阻抗必须控制在50欧姆左右。
- 在靠近天线的地方预留Π型匹配电路，两颗电容默认不贴片，电阻默认贴0欧姆，待天线厂调试好天线以后再贴上实际调试的匹配电路；
- Luat模块阻抗线及天线设计建议：
  https://doc.openluat.com/article/2430

5.4.2 RF 输出功率

表格 15：RF 传导功率

频段	最大	最小
LTE FDD B1/B3/B5/B8	23dBm +-2dB	<-44dBm
LTE TDD B34/38/B39/B40/B41	23dBm +-2dB	<-42dBm

5.4.3 RF 传导灵敏度

表格 16：RF 传导灵敏度

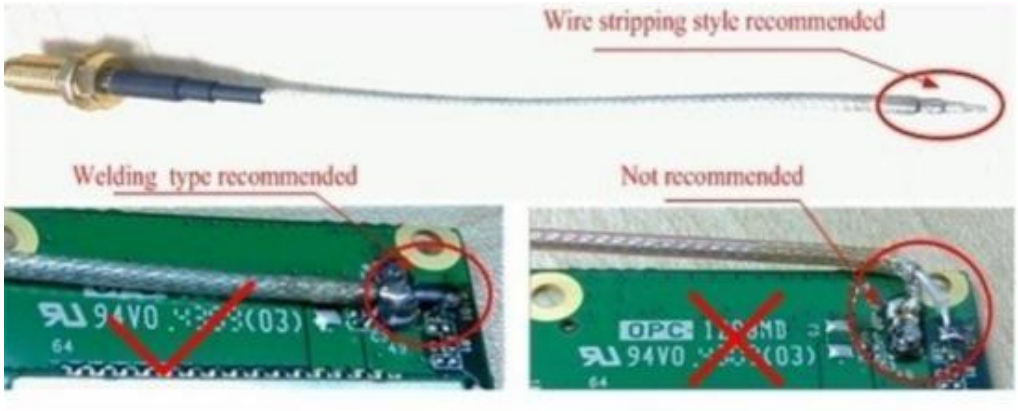
频段	接收灵敏度
LTE FDD B1(10M)	< -99dBm
LTE FDD B3(10M)	< -99dBm
LTE FDD B5(10M)	< -99dBm
LTE FDD B8(10M)	< -99dBm
LTE TDD B34(10M)	< -100dBm
LTE TDD B38(10M)	< -99dBm
LTE TDD B39(10M)	< -100dBm
LTE TDD B40(10M)	< -99dBm
LTE TDD B41(10M)	< -99dBm

5.4.4 工作频率

3GPP 频段	发送	接收	单位
LTE-FDD B1	1920~1980	2110~2170	MHz
LTE-FDD B3	1710~1785	1805~1880	MHz
LTE-FDD B5	824~849	869~894	MHz
LTE-FDD B8	880~915	925~960	MHz
LTE-TDD B34	2010~2025	2010~2025	MHz
LTE-TDD B38	2570~2620	2570~2620	MHz
LTE-TDD B39	1880~1920	1880~1920	MHz
LTE-TDD B40	2300~2400	2300~2400	MHz
LTE-TDD B41	2555~2655	2555~2655	MHz

5.4.5 推荐 RF 焊接方式

如果连接外置天线的射频连接器是通过焊接方式与模块相连的，请务必注意连接线的剥线方式及焊接方法，尤其是地要焊接充分，请按照下图中正确的焊接方式进行操作，以避免因焊接不良引起线损增大。



图表 18：射频焊接方式建议

5.5 电器特性，可靠性，射频特性

5.5.1 绝对最大值

下表所示是模块数字、模拟管脚的电源供电电压电流最大耐受值。

表格 17：绝对最大值

参数	最小	最大	单位
V_{BAT}	-0.3	4.7	V
VBUS	-0.3	5.5	V
电源供电峰值电流	0	1.5	A
电源供电平均电流（TDMA一帧时间）	0	0.7	A
数字管脚处电压	-0.3	3.6	V
模拟管脚处电压(ADC)	-0.3	3.6	V

5.5.2 推荐工作条件

表格 18：推荐工作条件

参数	最小	典型	最大	单位
V_{BAT}	3.3	3.8	4.3	V

VBUS	3.3	5.0	5.25	V

5.5.3 工作温度

表格 19：工作温度

温度	最低	典型	最高	单位
正常工作温度	-35	25	75	℃
受限工作温度	-40		85	℃
存储温度	-45		90	℃

5.5.4 功耗

5.5.4.1 模块工作电流

测试仪器：综测仪 R&S CMW500 ， 合宙功耗分析仪 Air9000P

测试条件：VBAT=3.8V ， 环境温度 25℃， 插入白卡，连接综测仪 CMW500

参数		测试条件	最小	典型	最大	单位
I_{VBAT}	漏电流	第一次上电				uA
		开机后关机（RTC 正常工作）				uA
	休眠待机电流	LTE-FDD @PF=32				mA
		LTE-FDD @PF=64				mA
		LTE-FDD @PF=128				mA
		LTE-FDD @PF=256				mA
		LTE-TDD @PF=32				mA
		LTE-TDD @PF=64				mA
		LTE-TDD @PF=128				mA
		LTE-TDD @PF=256				mA
	空闲模式电流	LTE-FDD @PF=64				mA
		LTE-TDD @PF=64				mA
	飞 行 模 式	平均电流(1.8V IO电平)				uA

	AT+CFUN=4 ， AT+CSCLK=3	底电流(1.8V IO电平)				uA
		平均电流(3.3V IO电平)				uA
		底电流(3.3V IO电平)				uA
	LTE-FDD B1 CH300 BW=10M	TX power = 23dbm				mA
	LTE-FDD B3 CH1575 BW=10M	TX power = 23dbm				mA
	LTE-FDD B5 CH2525 BW=10M	TX power = 23dbm				mA
	LTE-FDD B8 CH3625 BW=10M	TX power = 23dbm				mA
	LTE-TDD B34 CH36275 BW=10M	TX power = 23dbm				mA
	LTE-TDD B38 CH38000 BW=10M	TX power = 23dbm				mA
	LTE-TDD B39 CH38450 BW=10M	TX power = 23dbm				mA
	LTE-TDD B40 CH39150 BW=10M	TX power = 23dbm				mA
	LTE-TDD B41 CH40620 BW=10M	TX power = 23dbm				mA

5.5.4.2 实网模拟长连接功耗

模块联网功耗数据

模块低功耗模式下联网连接服务器定时心跳测试，模拟实际应用下的定时上报场景下功耗，从而能够估算出电池的使用时间。超低功耗方案参考：http://www.airpsm.cn

测试仪器：合宙 Air9000P 测试条件：移动网络 B34 RSRP=48（中等强度）供电 4V；TCP 连接 XX 分钟自动心跳包								
	平均功耗 mA（5分心跳）	心跳包发送		休眠平均功耗		500mAH 电池待机（天）		
		功耗 mA	耗电电量 uAH	功耗 mA	耗电电量uAH（5分钟间隔）	5分钟上报间隔	1小时上报间隔	24小时上报间隔
AT+POWERMODE=”PRO”	0.79	37	16.4	0.599	47.9	26	35.2	36.2
AT+POWERMODE=”SRD”	0.58	36	16.2	0.384	32.1	36	52	54
AT+POWERMODE=”PSM”	0.0028（无心跳）	\	\	\	\	104.4	1085	>10年

各阶段耗流(中等信号强度下实网测试测试)

阶段	平均电流	持续时间	总耗能
开机注册成功	19ma	4s	24uAH
发送数据（20字节）	36ma	1.2s	16.2uAH
发送数据 （20字节）	23.1ma	2.8s	18.3uAH

注意：

由于是实网测试，网络信号强度，注册频段，服务器响应时间都会对测试的值有较大影响，因此，此数参考。

5.5.5 静电防护

在模块应用中，由于人体静电，微电子间带电摩擦等产生的静电，通过各种途径放电给模块，可能会对模 块造成一定的损坏，所以 ESD 保护必须要重视，不管是在生产组装、测试，研发等过程，尤其在产品设计中，
都应采取防 ESD 保护措施。如电路设计在接口处或易受 ESD 点增加 ESD 保护，生产中带防 ESD 手套等。

下表为模块重点 PIN 脚的 ESD 耐受电压情况。

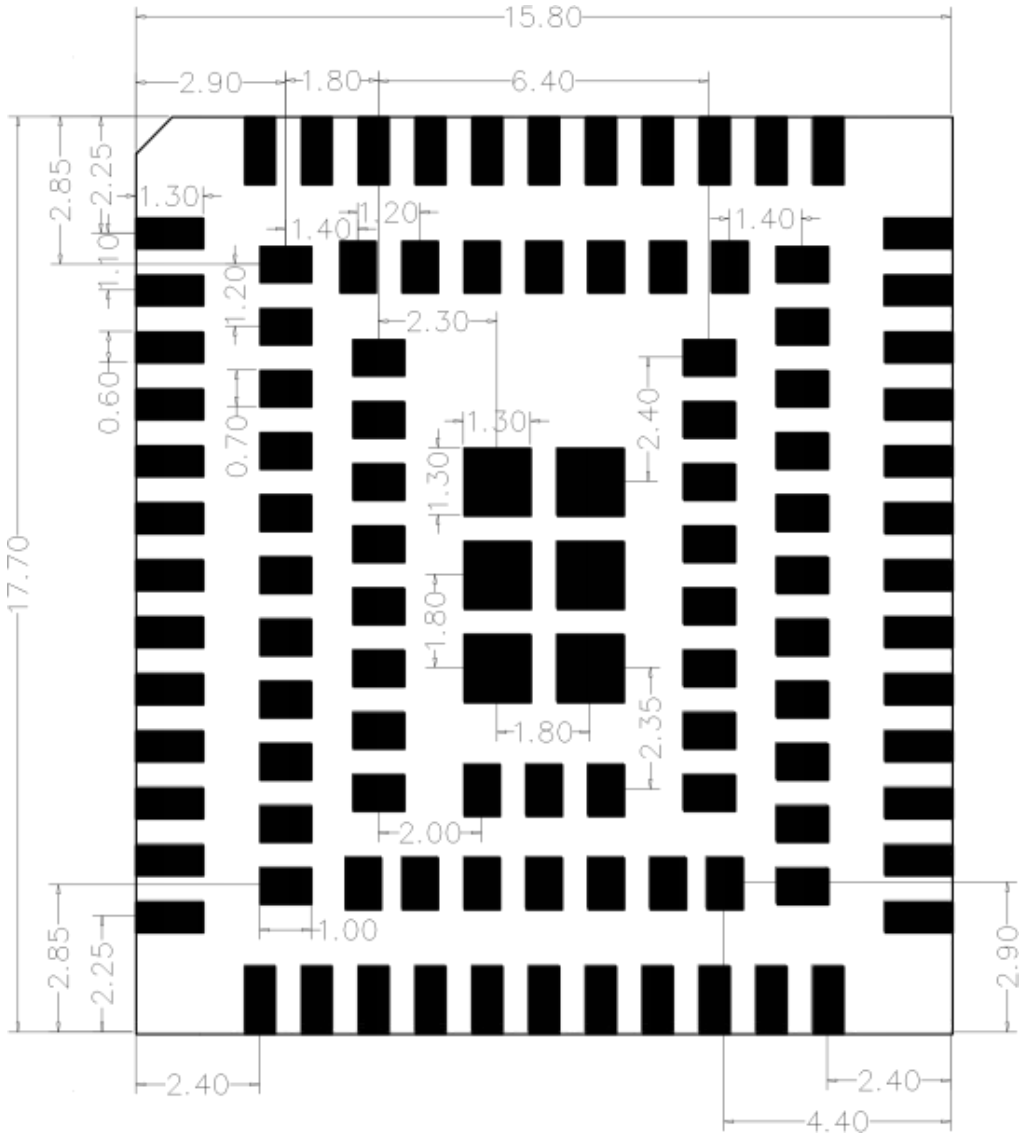
表格 20：ESD 性能参数（温度：25℃，湿度：45%）

管脚名	接触放电	空气放电
VBAT,GND	±5KV	±10KV
LTE_ANT	±5KV	±10KV
Others	±0.5KV	±1KV

5.6 模块尺寸图

该章节描述模块的机械尺寸以及客户使用该模块设计的推荐封装尺寸。

5.6.1 推荐 PCB 封装



图表 19：正视图，Air780EP PCB 封装（单位：毫米）

注意：

1. **PCB** 板上模块和其他元器件之间的间距建议至少 **3mm**；
2. 请访问： Air780EP 文档中心来获取模块的原理图 PCB 封装库。

5.7 存储和生产

5.7.1 存储

Air780EP 以真空密封袋的形式出货。模块的存储需遵循如下条件：

环境温度低于 40 摄氏度，空气湿度小于 90%情况下，模块可在真空密封袋中存放 12 个月。 当真空密封袋打开后，若满足以下条件，模块可直接进行回流焊或其它高温流程：

- 模块环境温度低于 30 摄氏度，空气湿度小于 60% ，工厂在 72 小时以内完成贴片。
- 空气湿度小于 10%

若模块处于如下条件，需要在贴片前进行烘烤：

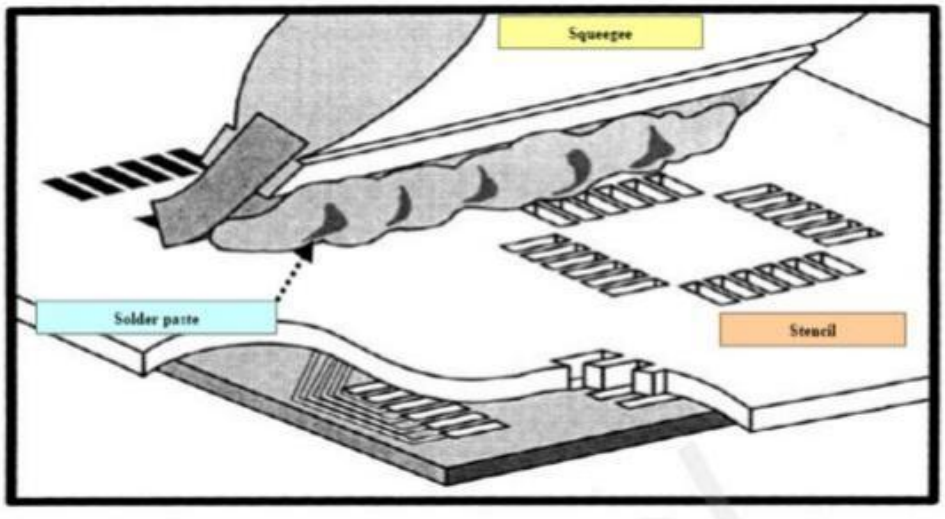
- 当环境温度为 23 摄氏度（允许上下 5 摄氏度的波动）时，湿度指示卡显示湿度大于 10%
- 当真空密封袋打开后，模块环境温度低于 30 摄氏度，空气湿度小于 60% ，但工厂未能在 72 小时以内完成贴 片
- 当真空密封袋打开后，模块存储空气湿度大于 10%

如果模块需要烘烤，请在 125 摄氏度下（允许上下 5 摄氏度的波动）烘烤 48 小时。

注意：模块的包装无法承受如此高温，在模块烘烤之前，请移除模块包装。如果只需要短时间的烘烤，请 参考 **IPC/JEDECJ-STD-033** 规范。

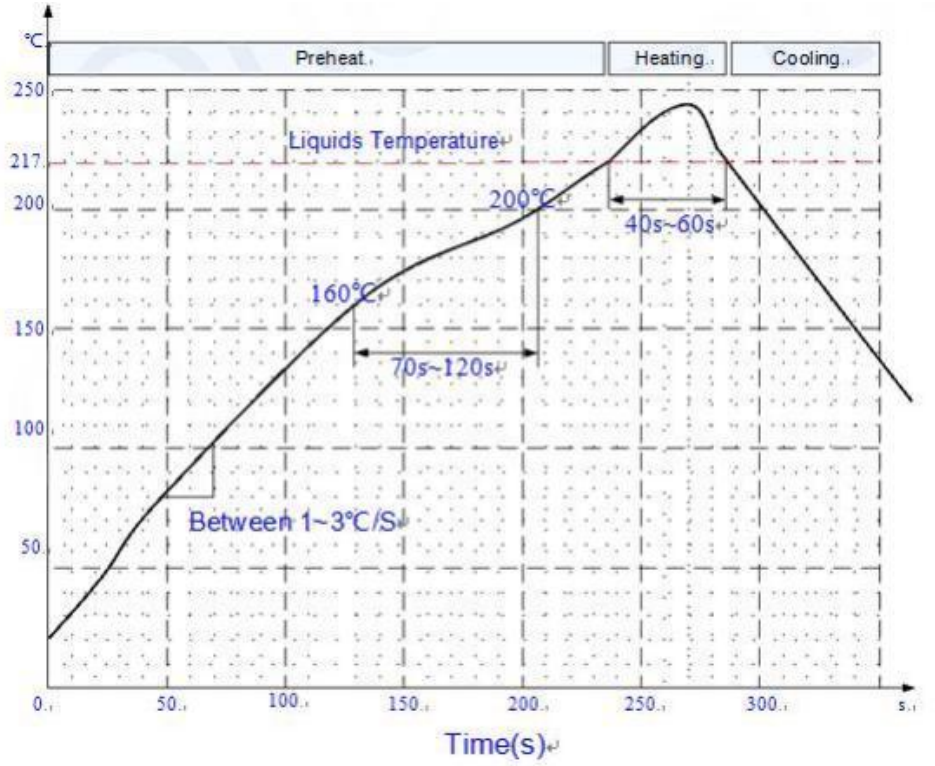
5.7.2 生产焊接

用印刷刮板在网板上印刷锡膏，使锡膏通过网板开口漏印到 PCB 上，印刷刮板力度需调整合适，为保证模 块印膏质量，Air780EP 模块焊盘部分对应的钢网厚度应为 0.2mm。



图表 20：印膏图

为避免模块反复受热损伤，建议客户 PCB 板第一面完成回流焊后再贴模块。推荐的炉温曲线图如下图所示：



图表 21：炉温曲线

5.8 术语缩写

表格 21：术语缩写

术语	英文全称	中文全称
ADC	Analog to Digital Converter	模数转换器
bps	Bits Per Second	比特/秒
CTS	Clear to Send	清除发送
DFOTA	Differential Firmware Over-the-Air	无线差分固件升级
DTR	Data Terminal Ready	数据终端就绪
ESD	Electro Static discharge	静电放电
ESR	Equivalent Series Resistance	等效串联电阻
EVB	Evaluation Board	评估板
FDD	Frequency Division Duplex	频分双工
FTP	File Transfer Protocol	文件传输协议

FTPS	FTP-over-SSL	对常用的文件传输协议（FTP）添加传输层安全（TLS）和安全套接层（SSL）加密协议支持的扩展协议
GPIO	General Purpose Input Output	通用输入输出管脚
GPS	Global Positioning System	全球定位系统
HTTP	Hypertext Transfer Protocol	超文本传输协议
HTTPS	Hypertext Transfer Protocol over Secure Socket Layer	超文本传输安全协议
LCC	Leadless Chip Carriers	不带引脚的正方形封装
LGA	Land Grid Array	栅格阵列封装
LTE	Long Term Evolution	长期演进
MQTT	Message Queuing Telemetry Transport	消息队列遥测传输
MSL	Moisture Sensitivity Levels	湿度敏感等级
NITZ	Network Identity and Time Zone	网络标识和时区
NTP	Network Time Protocol	网络时间协议
PA	Power Amplifier	功率放大器
PCB	Printed Circuit Board	印制电路板
PCM	Pulse Code Modulation	脉冲编码调制
PDU	Protocol Data Unit	协议数据单元
PMIC	Power Management IC	电源管理集成电路
PPP	Point-to-Point Protocol	点到点协议
RF	Radio Frequency	射频
RTS	Require To Send	请求发送
SMS	Short Message Service	短信
SSL	Secure Sockets Layer	安全套接层
TCP	Transmission Control Protocol	传输控制协议
TDD	Time Division Duplexing	时分双工
UART	Universal Asynchronous Receiver & Transmitter	通用异步收发机
UDP	User Datagram Protocol	用户数据报协议
UMTS	Universal Mobile Telecommunications System	通用移动通信系统
USB	Universal Serial Bus	通用串行总线
(U)SIM	(Universal) Subscriber Identity Module	（通用）用户身份识别模块
VSWR	Voltage Standing Wave Ratio	电压驻波比

六 快速入门

本章节基于 Core-Air780EP 开发板，来演示一个软件功能 demo，该 demo 实现的功能为：模组连接一个 tcp 服务器，实现数据收发功能。

6.1 概述

Core-Air780EP 开发板是合宙通信推出的基于 Air780EP 模组所开发的，包含电源，SIM 卡，USB，天线，音频等必要功能的最小硬件系统。以方便用户在设计前期对 Air780EP 模块进行性能评估，功能调试，软件开发等用途。

6.2 硬件环境

1、Core-Air780EP 开发板 1 块，SIM 卡；

2、Type-C USB 线 1 根，直接使用手机上的 Type-C 线就可以；

3、电脑一台（win10 及以上操作系统）；

6.3 软件环境

6.3.1 USB 驱动

参考本文档：8.1 USB 驱动 ；

6.3.2 Luatools 下载调试工具

参考本文档：8.2 Luatools 下载调试工具 ；

6.3.3 AT 软件开发资料

参考本文档：九 软件开发指南（AT） ；

6.3.4 LuatOS 软件开发资料

参考本文档：十 软件开发指南（LuatOS） ；

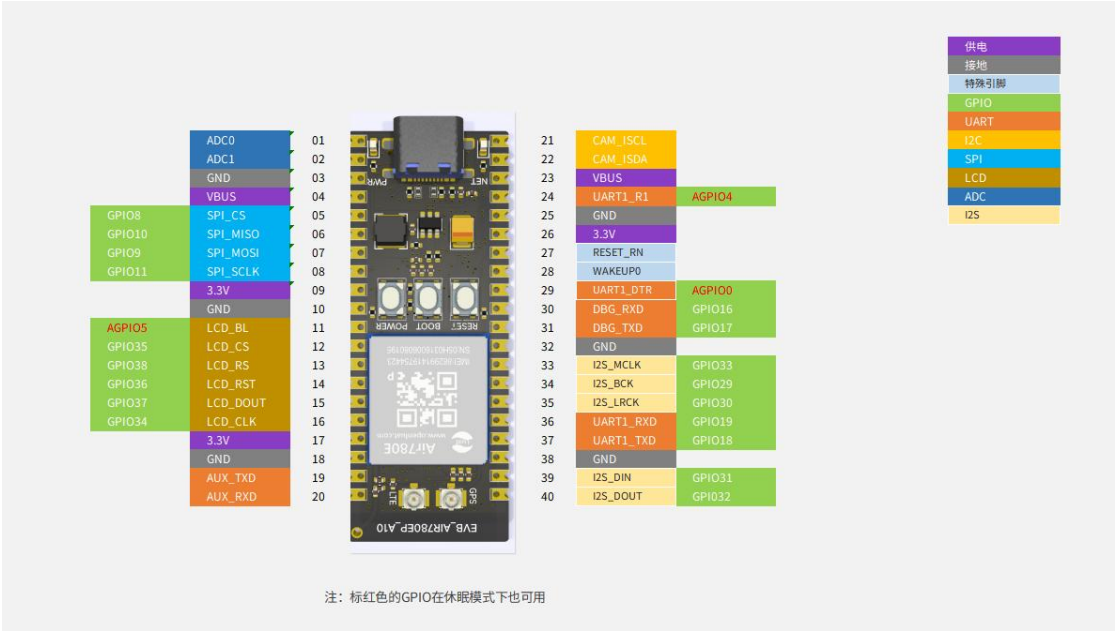
6.4 开发板使用说明

6.4.1 硬件配置

- 一代 IPEX 天线连接器（选配）
- 4G 弹簧天线
- 一个下载/调试串口，两个通用串口

- IO 口默认电平 3.3V （1.8V 可调）
- 支持 USB 5V 直接供电
- 自弹式 Micro SIM 卡座
- 1 个电源 LED 指示灯，1 个网络指示灯
- 标准 2.54mm 邮票孔，兼容排针
- 3 个按键，开机键，下载模式键，复位键
- 2 路 ADC 输入
- 1 路 I2C 接口

6.4.2 管脚定义



其它管脚复用功能参考本文档：GPIO 复用详细描述 ；

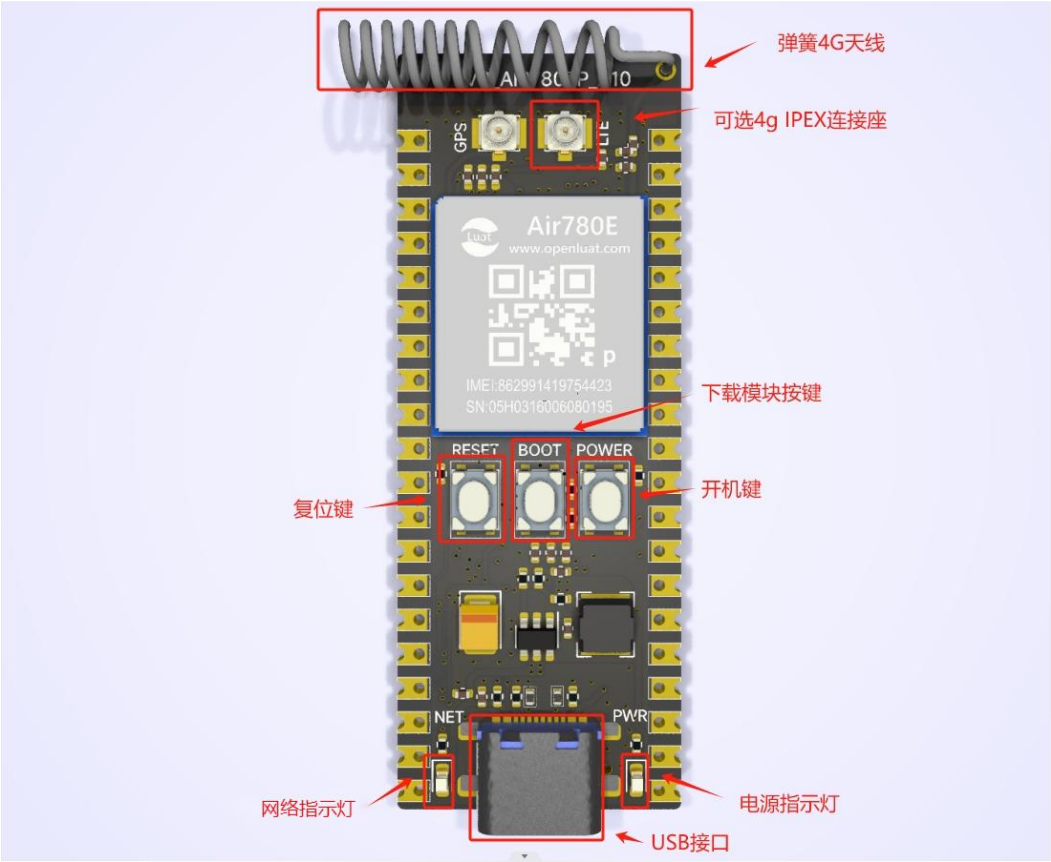
6.4.3 管脚功能说明

编号	名称	描述	引脚	备注
1	A0	模数转换输入通道 0	ADC0	
2	A1	模数转换输入通道 1	ADC1	
3	GND	参考地	GND	
4	VBUS	5V 输入	5V	与 USB 5V 管脚连通
5	G8	SPI0 片选信号	SPI0_CS/GPO8	

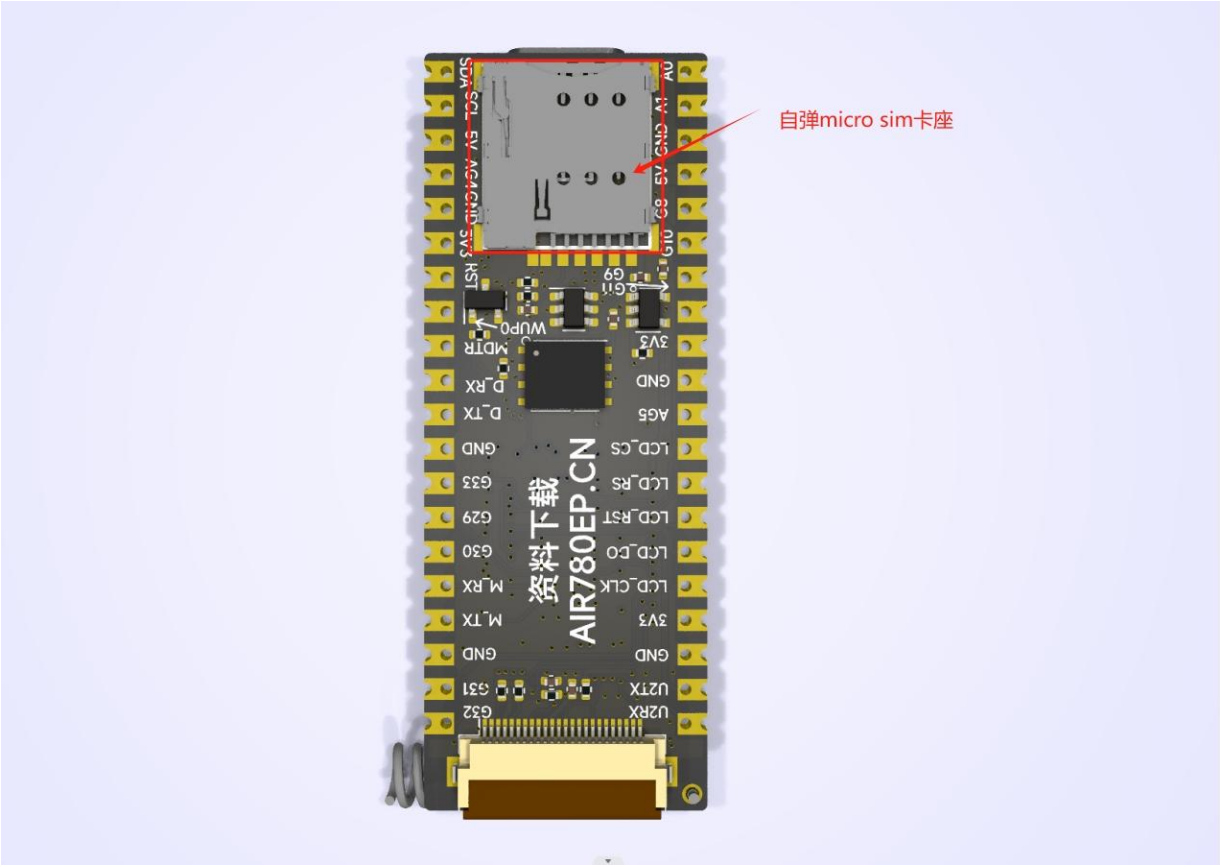
编号	名称	描述	引脚	备注
6	G10	SPI0 数据输入	SPI0_MISO/GPIO10	
7	G9	SPI0 数据输出	SPI0_MOSI/GPIO9	
8	G11	SPI0 时钟信号	SPI0_SCLK/GPIO11	
9	3V3	3.3V 输出	3.3V	板载外部 LDO，200mA
10	GND	参考地	GND	
11	AG5	LCD 背光	AGPIO5	休眠状态下可用
12	LCD_CS	LCD 片选信号	LCD_CS/GPIO35	
13	LCD_RS	LCD 命令/数据选择信号	LCD_RS/GPIO38	
14	LCD_RST	LCD 复位信号	LCD_RST/GPIO36	
15	LCD_DC	LCD 数据信号	LCD_DOUT/GPIO37	
16	LCD_CLK	LCD 时钟信号	LCD_CLK/GPIO34	休眠状态下可用
17	3V3	3.3V 输出	3V3	
18	GND	参考地	GND	
19	U2TX	辅助串口数据发送	AUX_TXD/U2TX	
20	U2RX	辅助串口数据接收	AUX_RXD/U2RX	
21	SCL	I2C 接口时钟信号	CAM_I2C_SCL	
22	SDA	I2C 接口数据信号	CAM_I2C_SDA	
23	VBUS	5V 输入	5V	
24	AG4	振铃信号	MAIN_RI/AGPIO4	

编号	名称	描述	引脚	备注
25	GND	参考地	GND	
26	3V3	3.3V 输出	3V3	
27	RST	模块复位信号	RESET_N	
28	WUPO	WAKEUP	WAKEUPO	
29	MDTR	模块休眠控制管脚	MAIN_DTR/AGPIO2	
30	D_RX	调试串口数据输入	U0RX/DBGRX/GPIO16	
31	D_TX	调试串口数据输出	U0TX/DBGTX/GPIO17	
32	GND	参考地	GND	
33	G33	GPIO	GPIO33	
34	G29	GPIO	GPIO29	
35	G30	GPIO	GPIO30	
36	M_RX	主串口数据输入	MAIN_TXD/GPIO19	
37	M_TX	主串口数据输出	MAIN_RXD/GPIO18	
38	GND	参考地	GND	
39	G31	GPIO	GPIO31	
40	G32	GPIO	GPIO32	

6.4.4 顶视图


弹簧4G天线
可选4g IPEX连接座
下载模块按键
开机键
复位键
网络指示灯
电源指示灯
USB接口
GPS
LTE
Air780E
www.openluat.com
IMEI:862991419754423
SN:05H0316006080195
RESET
BOOT
POWER
NET
PWR

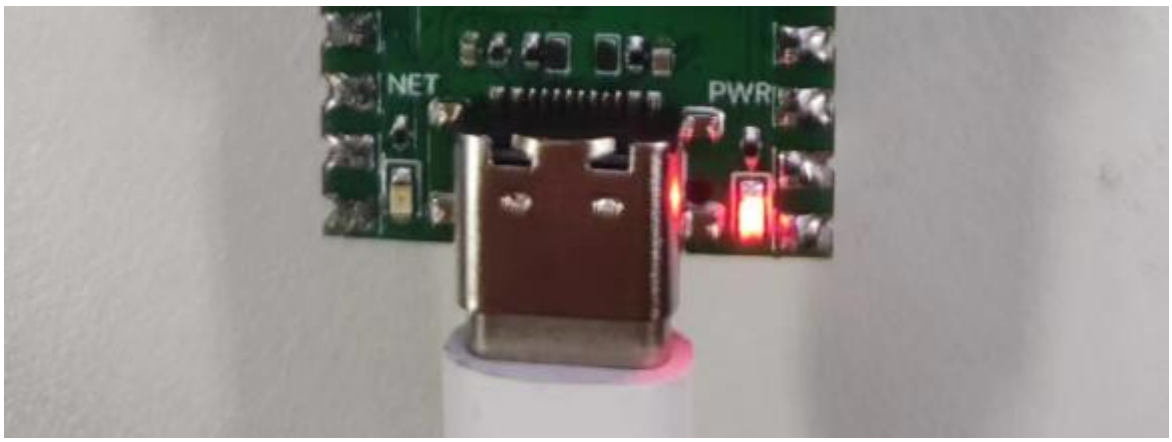
6.4.5 底视图



6.4.6 供电

6.4.6.1 通过 USB 接口直接供电

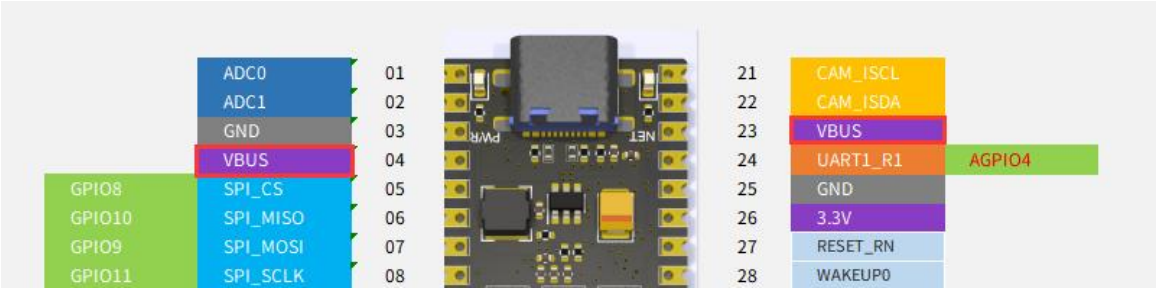
给 Core-Air780EP 开发板最直接的供电方式是用 type-C 的 USB 数据线连接电脑的 USB 接口，通过电脑的 USB 接口直接给开发板提供 5V 供电。开发板内部集成 DCDC 开关电源，将 5V 输入转换为 4V 给 Air780EP 模块供电。当开发板有外部供电时，电源指示灯会亮起，表示开发板已上电。



推荐使用 PC 的 USB3.0 接口给开发板供电。

6.4.6.2 通过开发板 5V 管脚供电

Core-Air780EP 开发板同时在两排邮票孔管脚处引出 5V 电源输入管脚，方便用户可以通过 5V 管脚和 GND 管脚通过外部电源给开发板供电，以方便进行功耗电流测试等。5V 管脚与开发板上 USB 接口处的 5V 管脚相通，因此在外部 USB 插入的情况下，禁止用 5V 供电，以防对 PC 设备造成损坏。



5V 供电管脚供电电压不能超过 5.5V，否则会有开发板烧毁的风险

6.4.7 开关机

6.4.7.1 按键开关机

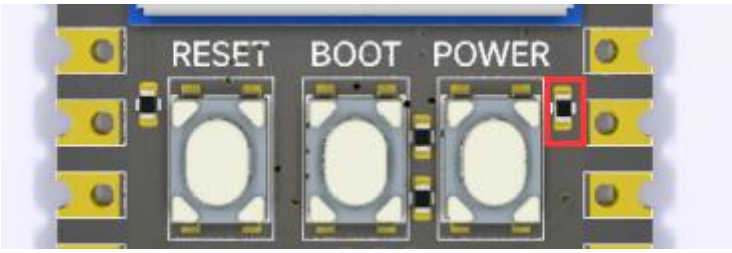
Core-Air780EP 在满足供电条件之后，长按开机键（1.5S）以上就可以触发开机。

开机后，如果再长按开机键（1.5S）以上触发关机流程。

6.4.7.2 上电自动开机

想让开发板上电同时自动开机，这种方式也能实现，但是需要调整开发板上的电阻了。

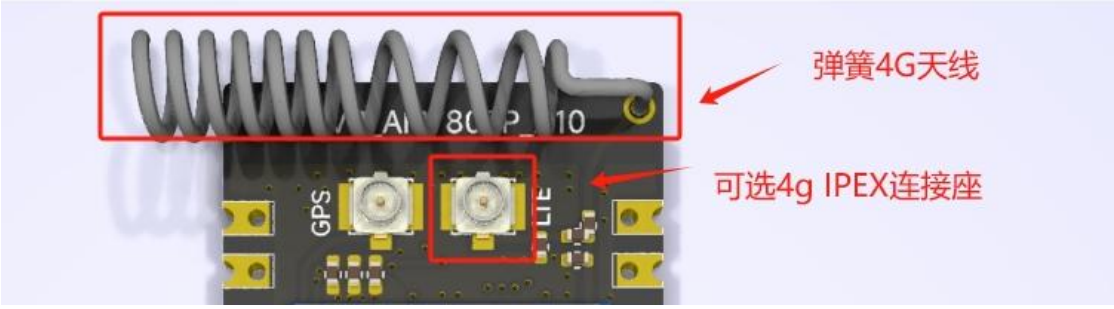
把下图标准的电阻短接，会将 Air780EP 的 POWKEY 信号拉低，也就实现开发板上电开机。



改成上电开机后，开机按键不再有任何作用。上电开机的操作不推荐，仅仅限于有硬件基础开发者调试用。

天线

为了方便用户使用，CORE-AIR780EP 开发默认自带弹簧螺旋 4G 天线，无需用户额外适配购买天线。



自带弹簧螺旋 4G 天线由于条件所限，射频性能不是最优，仅仅用于方便软件功能调试而使用，不代表 Air780EP 模块的真实射频性能。如果有射频传导测试的评估需求，请联系合宙官方人员，提供射频测试专用评估板。

针对部分用户有对 Air780EP 模块硬件射频评估测试需求，Core-Air780EP 开发板还专门设计了一个 1 代 IPEX 射频座，可以用于直接测试射频传导性能，或者外接专业的 4G 天线，做其他应用测试用。

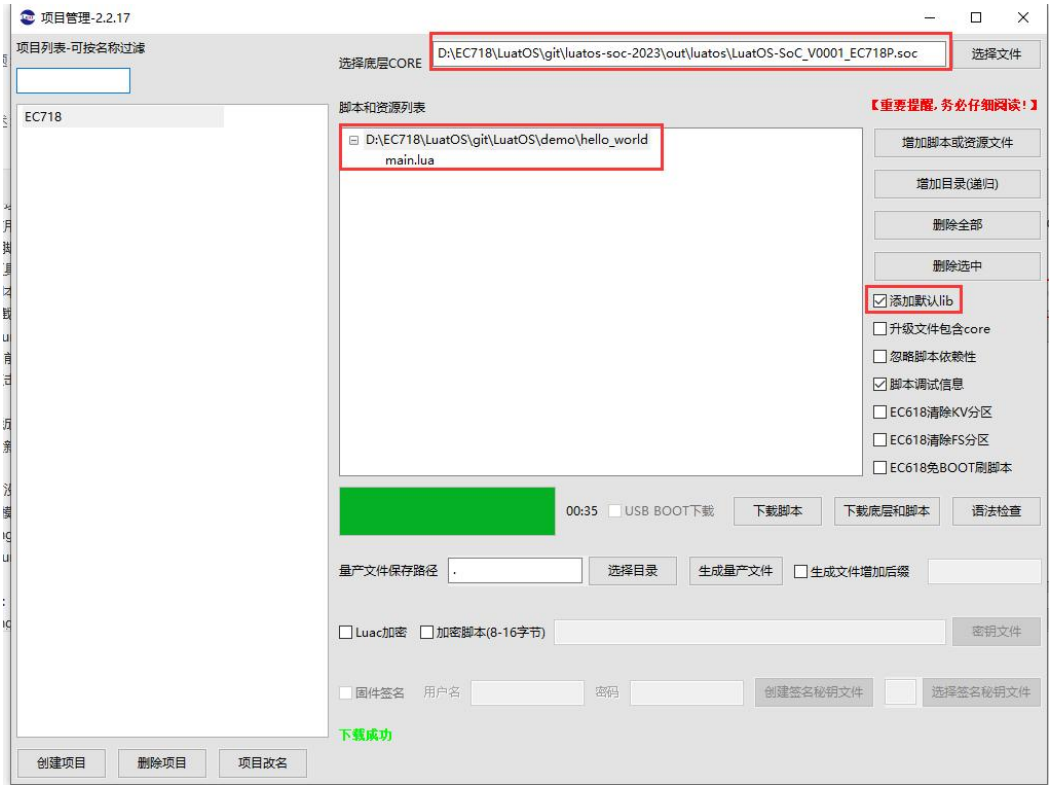
固件升级

Core-Air780EP 开发板固件升级可以直接通过 USB 口进行，使用 Luatools 工具进行固件升级更新。

按照以下步骤操作：

1、首先 USB 连接 PC，保持上电但不开机状态。

2、PC 上打开 LuaTool，选择好要更新的固件和脚本。

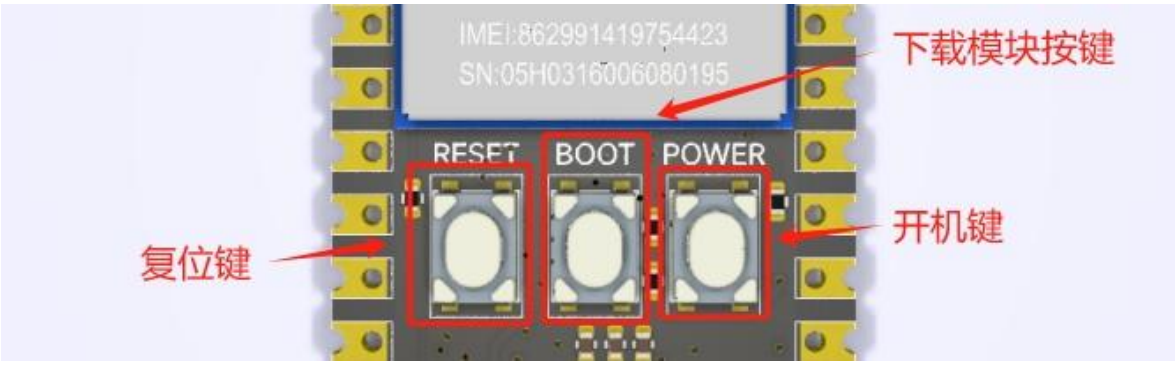


以上为 LuatOS 下载方式，CSDK 和 AT 只需要下载对应的固件就可以。

3、按住下载模式按键（boot 键）不放，同时再长按开机键开机，这时开发板会进入下载模式，luatool 下载进度条会开始跑，这时可以松开 boot 按键。直到工具提示下

载完成。

如果，未能成功进入下载模式，而是进入正常开模式，这时可以按住 boot 键，再短按复位按键，让开发板重启，重新进入下载模式。



4、如何判断有没有进入下载模式:可以通过 PC 端的设备管理器中虚拟出来的 USB 断开数量来判断：

- 正常开机模式：

- 下载模式：

6.5 软件示例

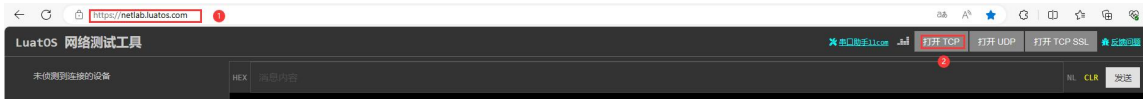
6.5.1 LuatOS 示例

本节通过 socket 的 demo 的说明和演示，达到用户能快速入门开发的目的。

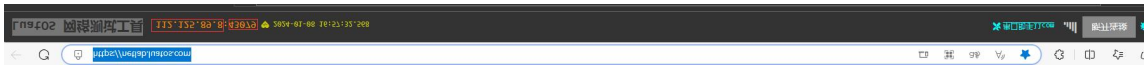
socket 的 demo 功能主要是连接一个 TCP 服务器，实现通过串口实现数据的收发功能。操作演示流程如下：

1、通过前面软件环境的描述先搭建软件开发环境（主要是：下载 Luatools 下载调试工具，下载对应的固件和脚本）。

2、点击右侧打开 LuatOS 网络测试工具界面，然后打开 TCP。如下所示：



这里可以看到对应的 IP：112.125.89.8，PORT：43079



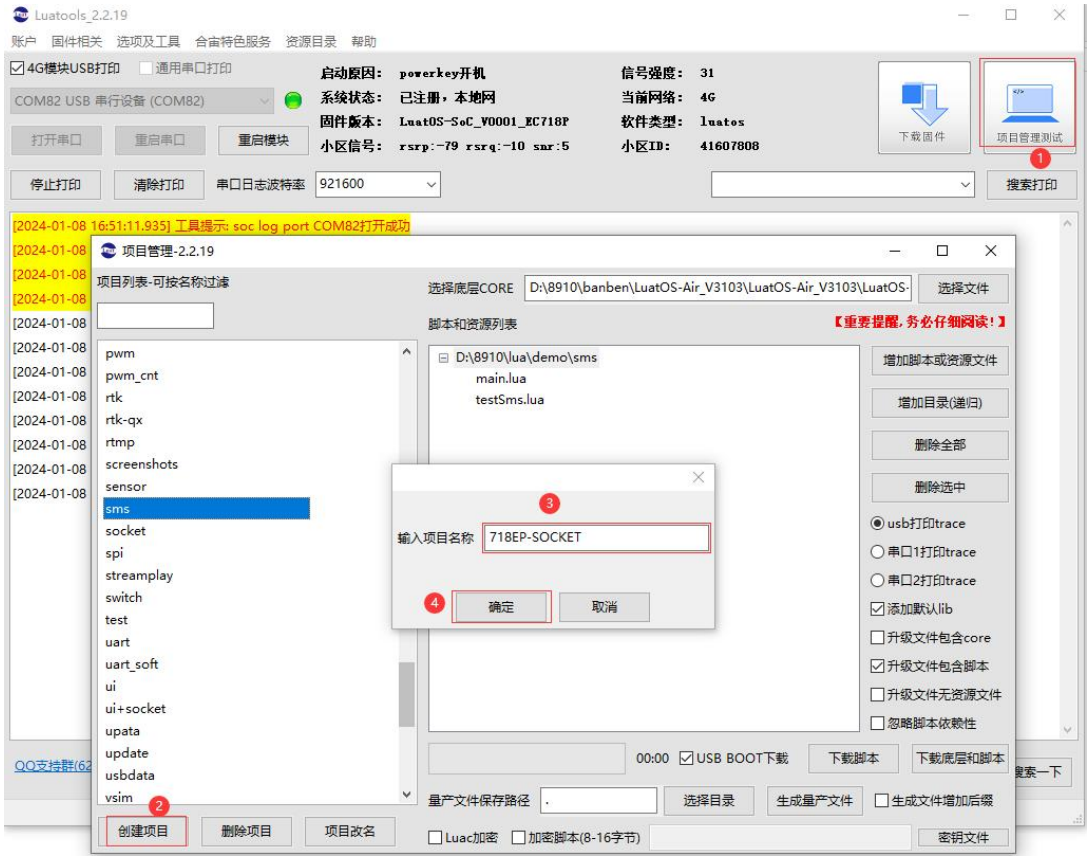
3、在 demo（demo 的 git 路径：git\LuatOS\demo\socket\EC618）最后添加调试接口，填入对应的串口 ID，服务器 IP 和端口，如下所示：

```
dtuDemo(uart.VUART_0,"112.125.89.8","43079")
```

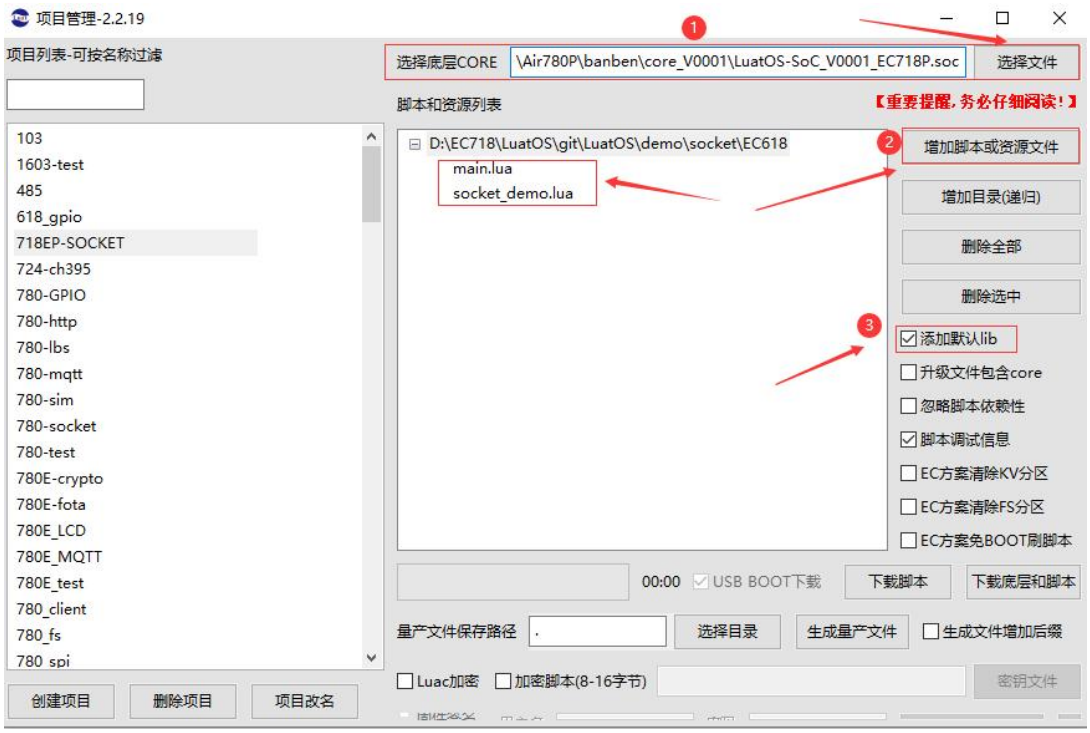
注释掉 main.lua 中 SerDemo(15000)脚本，不然会报错。注释后的代码如下所示：

```
require "socket_demo"

-- require "server_demo"

--SerDemo(15000)
```

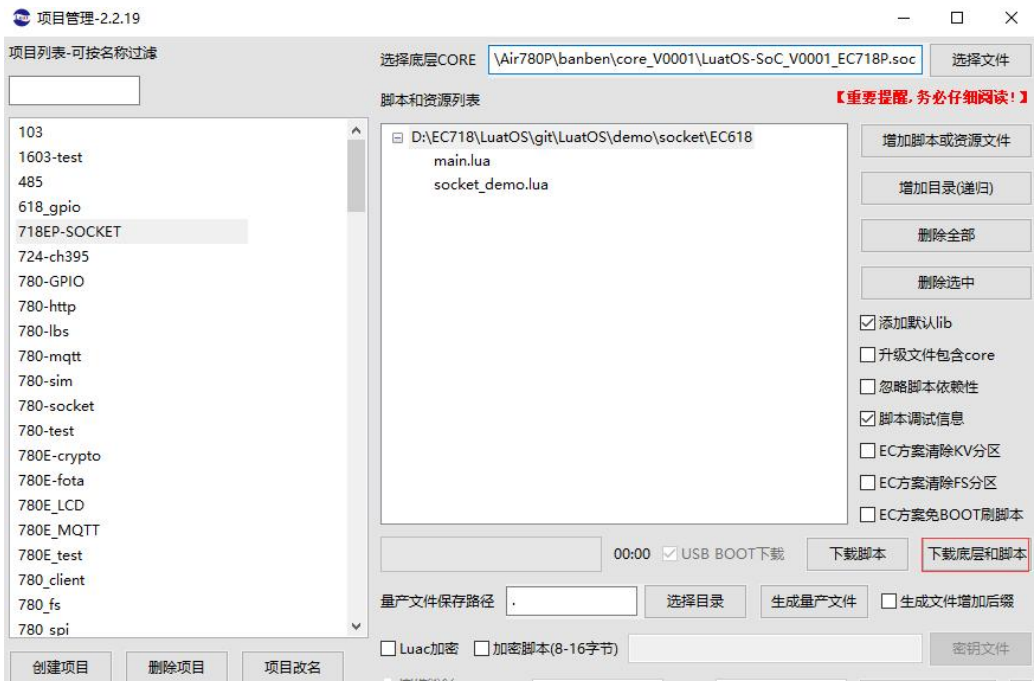
4、用 Luatools 创建一个新项目，这个创建的是 780EP-SOCKET，如下所示：



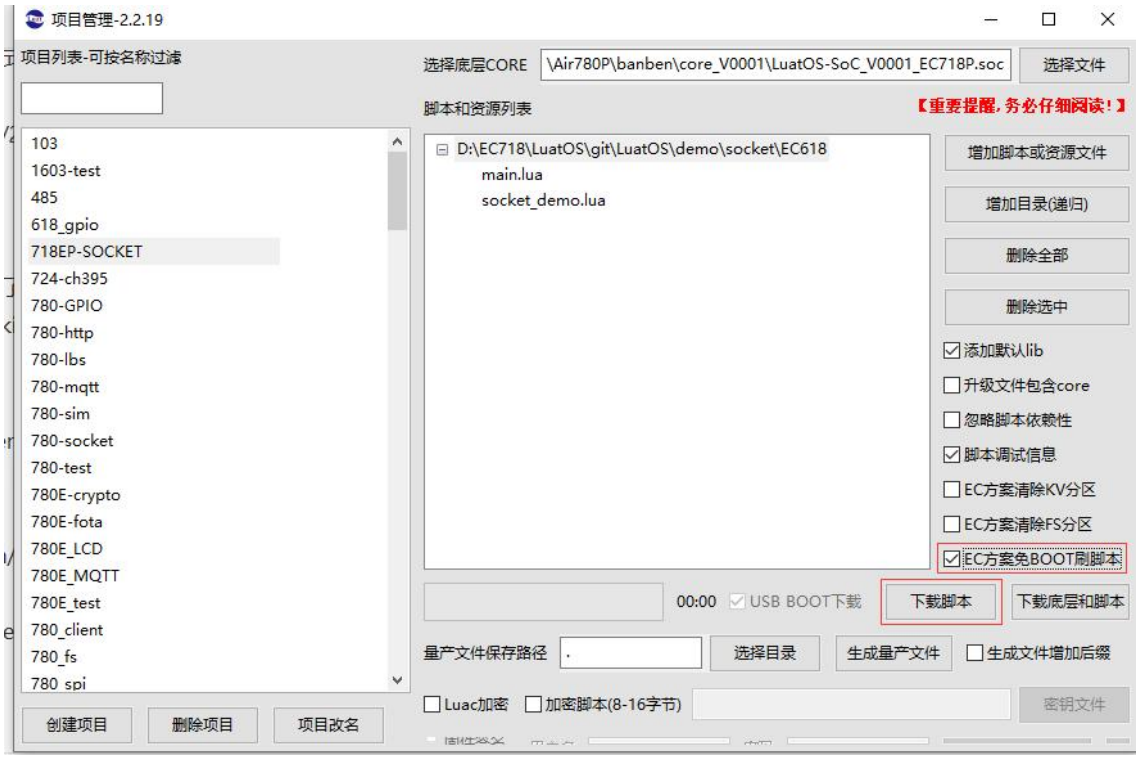
5、选择对应的固件和脚本，如下所示：



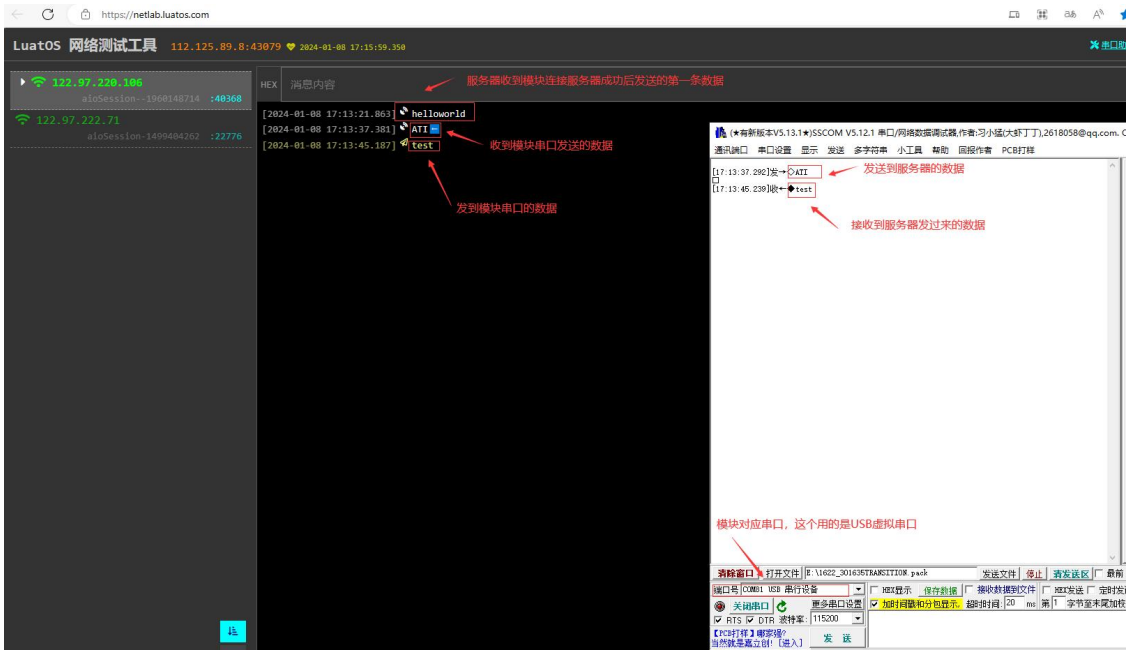
6、下载固件和脚本，第一次下载需要选择下载底层和脚本，后面调试过程中，底层固件不变的情况下可选选择只下载脚本，如下所示：



调试过程中可以勾选免 BOOT 刷脚本方式下载，如下图所示：



7、模块和服务交换数据说明，如下所示：



8、Luatoos 工具对应 LOG，如下所示：

这里主要是演示功能，具体代码可以通过下载 DEMO 来分析。

七 开发板使用说明（后续补充）

八 工具使用指南

8.1 USB 驱动

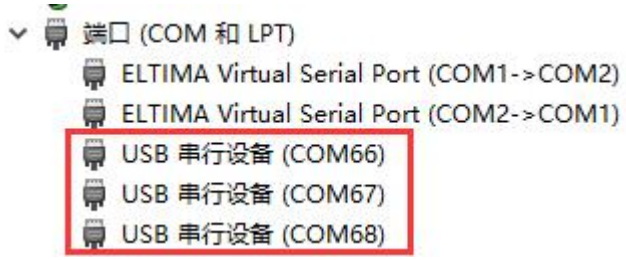
驱动方式

Air780EP 采用的驱动方式为微软官方推荐 USB 串行驱动程序，默认的 WINDOWS 系统已经支持此驱动，所以在 WINDOWS 系统上可以免安装驱动，就能方便的开发项目(注意：仅支持 WIN10 以及 WIN10 以上的系统)。

虚拟串口

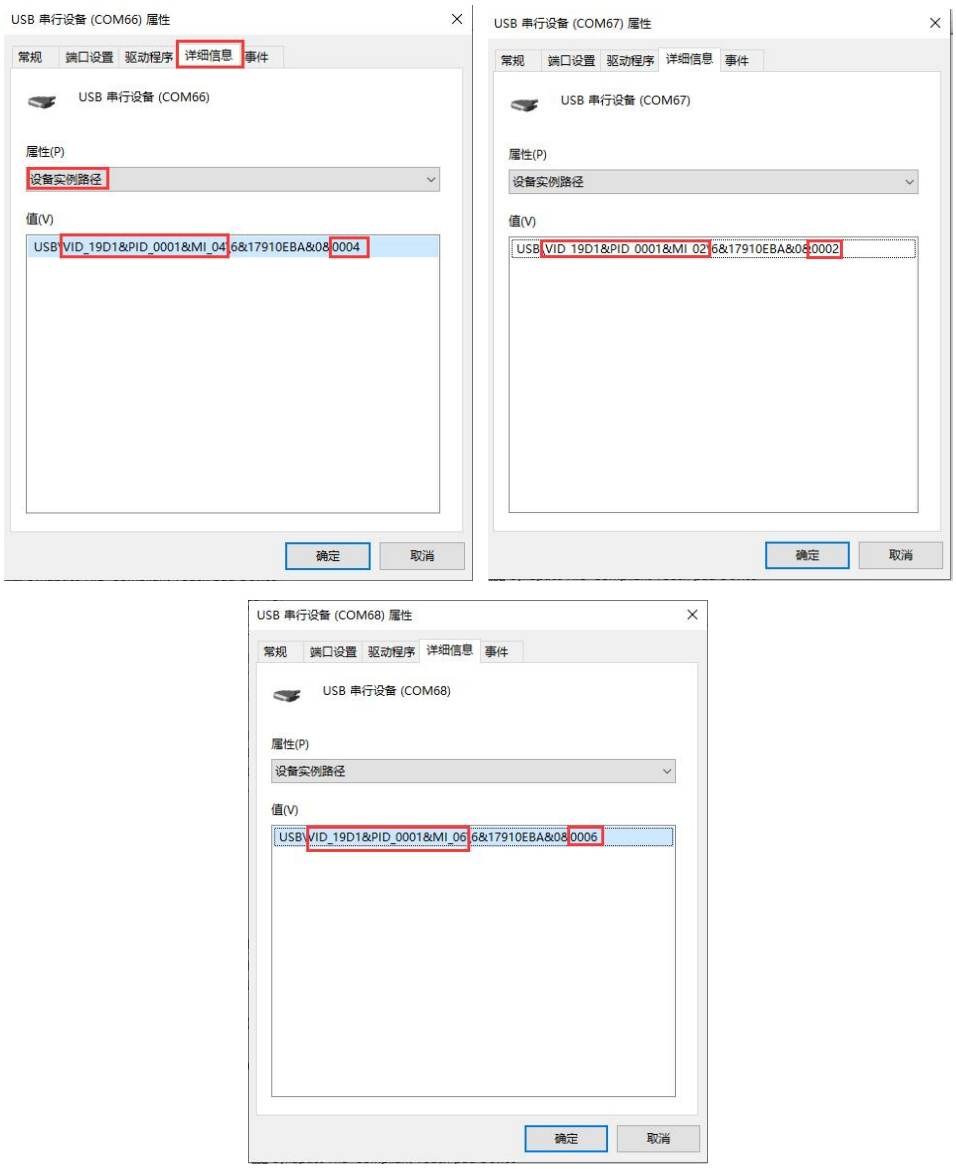
USB 驱动无需安装，Air780EP 通过 usb 口连接到电脑后(Air780EP 需要 powerkey 接地 1.2S，VBAT 供电 4.2-3.3V，USB 口和电脑相连，USB 线为数据传输线并非充电线，区别在于线里有没有 DM 与 DP 。一个很方便的验证方法为，C 口插入手机，电脑是否弹出大容量存储设备，如果弹出则证明该线为数据线，反之则

该线只能充电即只有 VCC 和 GND)，如果正常开机，会虚拟出如下三个端口（不同电脑的端口号不一样，以实际测试为准）：



在不同类型的固件（AT、LuatOS、CSDK）中，这三个虚拟串口的作用不完全相同，接下来针对 AT、LuatOS、CSDK 三种固件，分别介绍下这三个虚拟串口的作用。

首先先对这三个虚拟串口做个区分，一眼看上去，这三个虚拟串口名字完全一样，只是端口号不同，需要注意的是，不能根据端口号来区分每个虚拟串口的作用，需要通过串口属性来区分，如下图所示，在端口上点击鼠标右键，选择属性菜单，属性对话框中，选择 详细信息->设备示例路径，可以看到虚拟串口的设备实例路径的值，在这个值中，仅关注最后的一个 ID 即可，分别是 0002、0004、0006，通过这三个值可以区分虚拟串口的功能，接下来会借助于这个 ID 对虚拟串口的功能进行描述。



AT 固件中虚拟串口的作用

- 0002 号虚拟串口：AT 命令通信串口；同时，Luatools 默认使用此串口实时打印出来交互的 AT 命令；
- 0004 号虚拟串口：抓取底层日志的串口，可以通过 Luatools 抓取，也可以通过 EPAT 工具抓取；
- 0006 号虚拟串口：AT 命令通信串口；

LuatOS 固件中虚拟串口的作用

- 0002 号虚拟串口：Luatools 抓取应用日志的串口；
- 0004 号虚拟串口：抓取底层日志的串口，可以通过 Luatools 抓取，也可以

通过 EPAT 工具抓取；

- 0006 号虚拟串口：用户应用串口，用户根据自己的项目需要，可以在代码中控制这个虚拟串口收发自定义的应用数据；

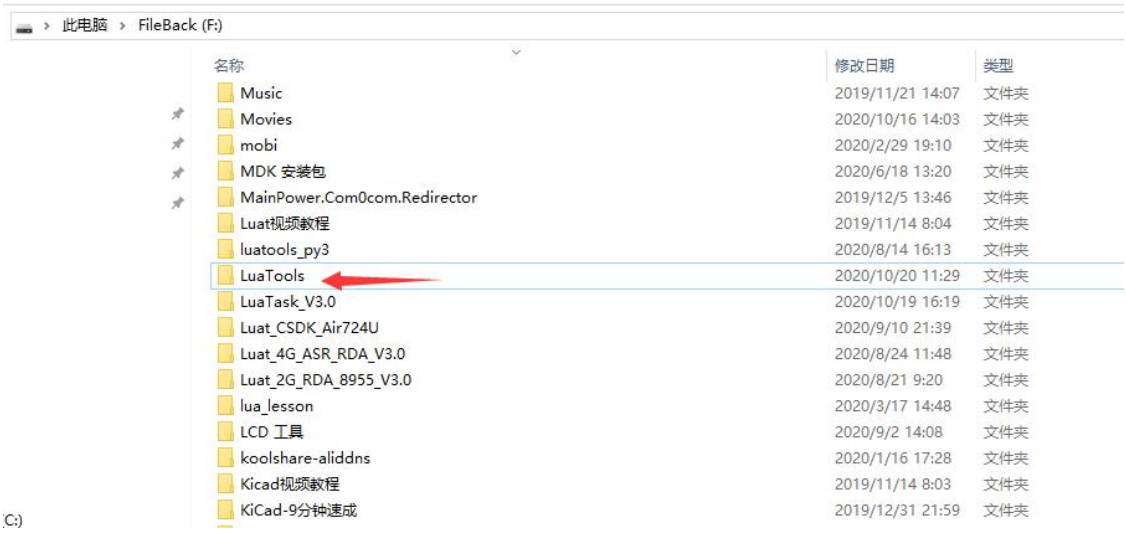
8.2 Luatools 下载调试工具

Luatools 下载

点击右方链接下载 Luatools 工具：Luatools 工具下载路径 ；

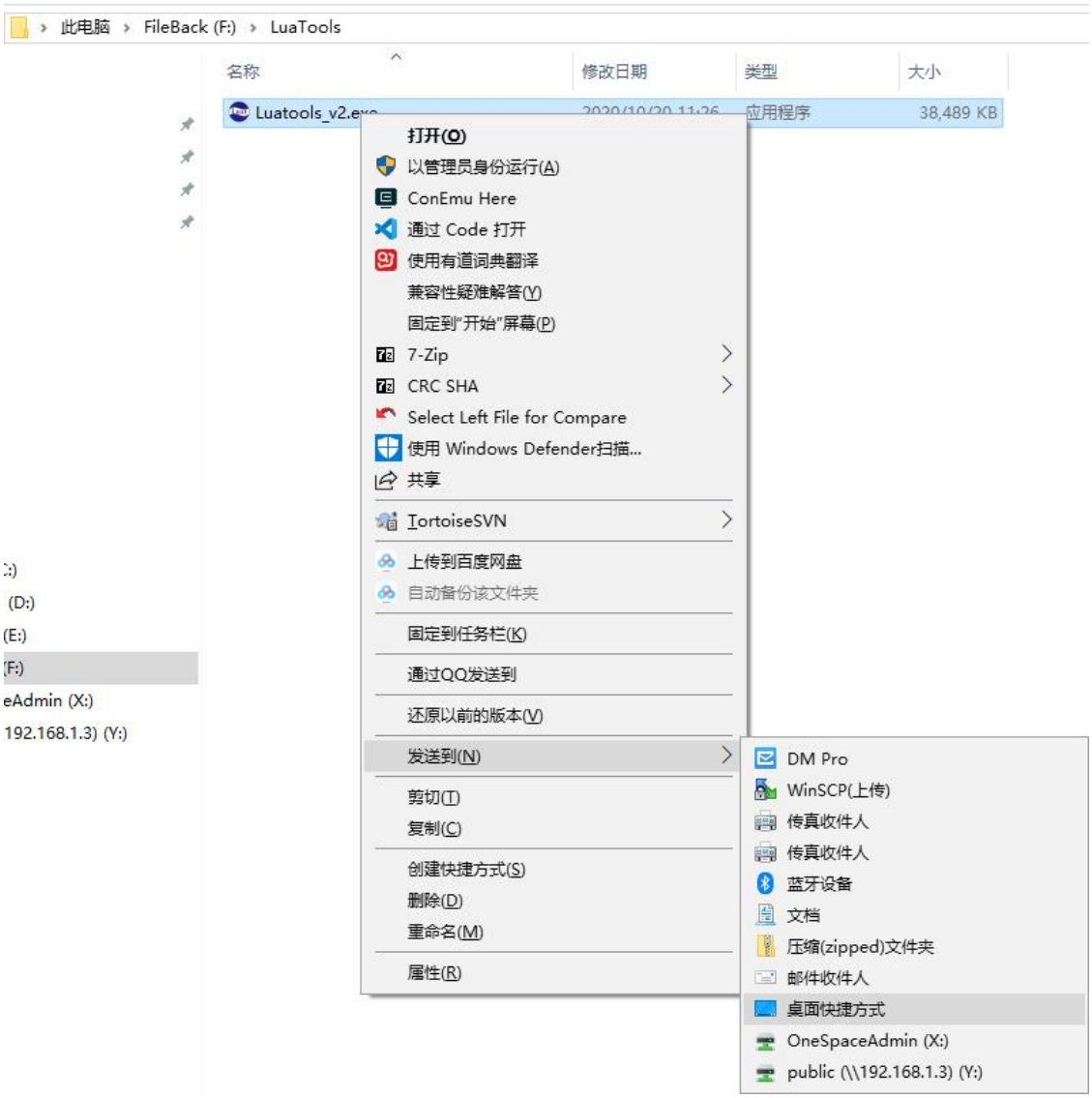
Luatools 安装

1. Luatools 工具下载后的文件名为：Luatools_v3.exe
2. 在电脑中新建文件夹并重命名为 Luatools

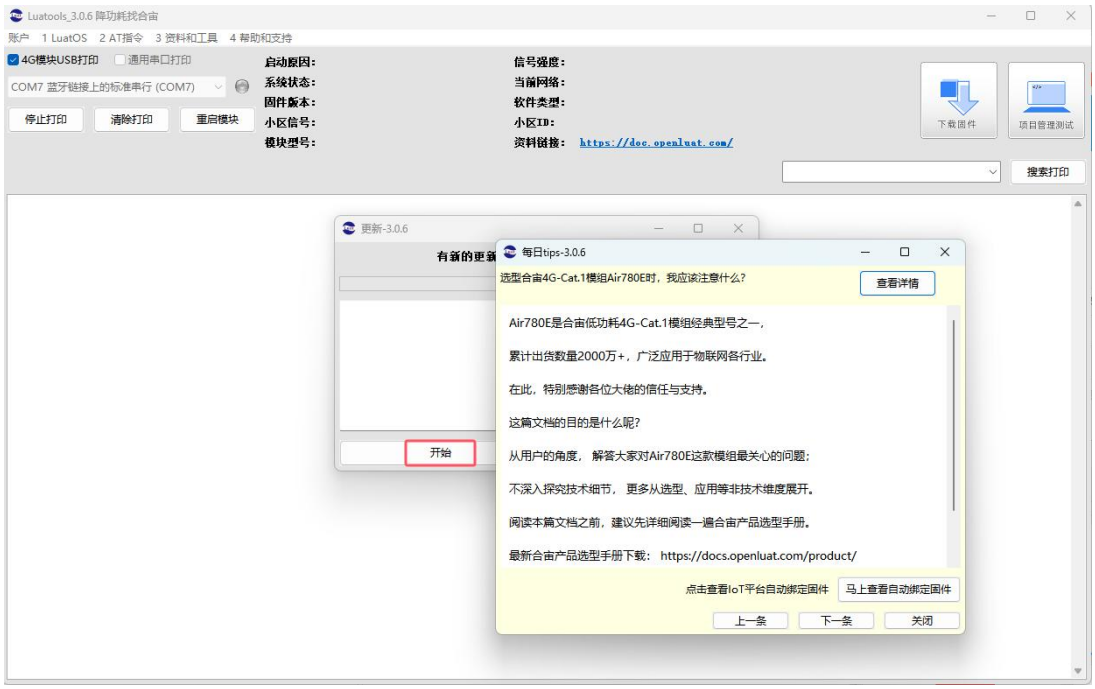


3. 将下载的 Luatools_v3.exe 拷贝或移动到 新建的 Luatools 文件夹内：

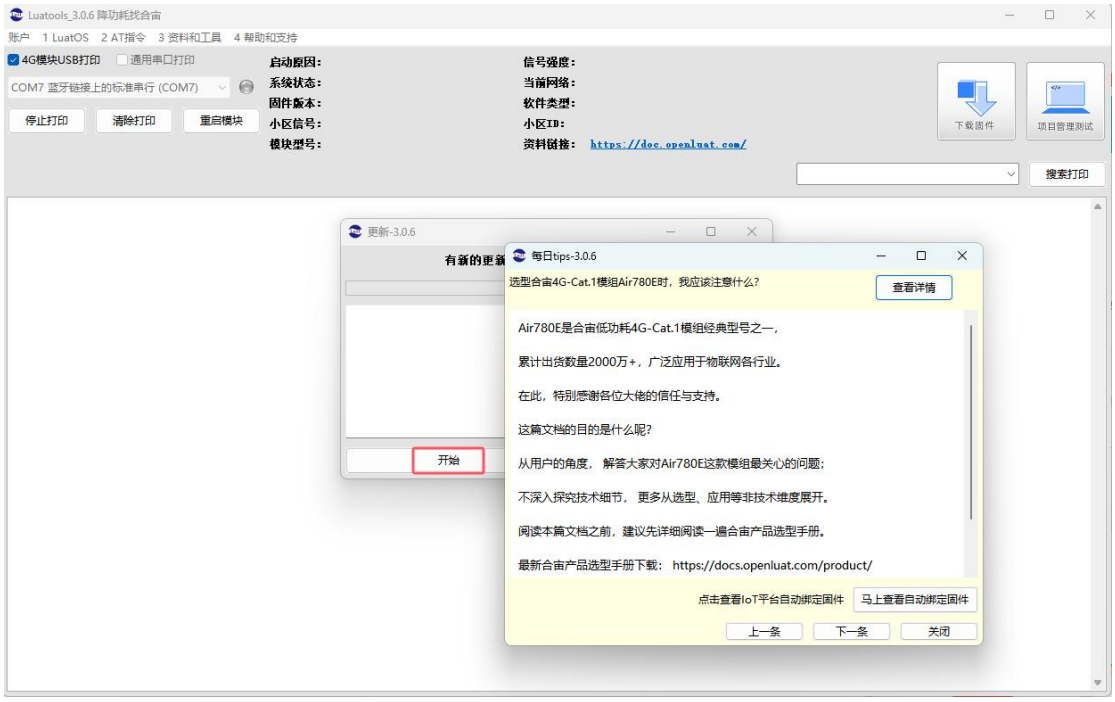
4. 点 Luatools_v3.exe 右键发送快捷方式到桌面即可



5. 双击 Luatools_v3.exe 启动程序并更新，如果有杀软拦截，请务必通过，请务必通过，请务必通过，重要的事情说三遍，特别的软件开发请勿用 xxx 安全全家桶。

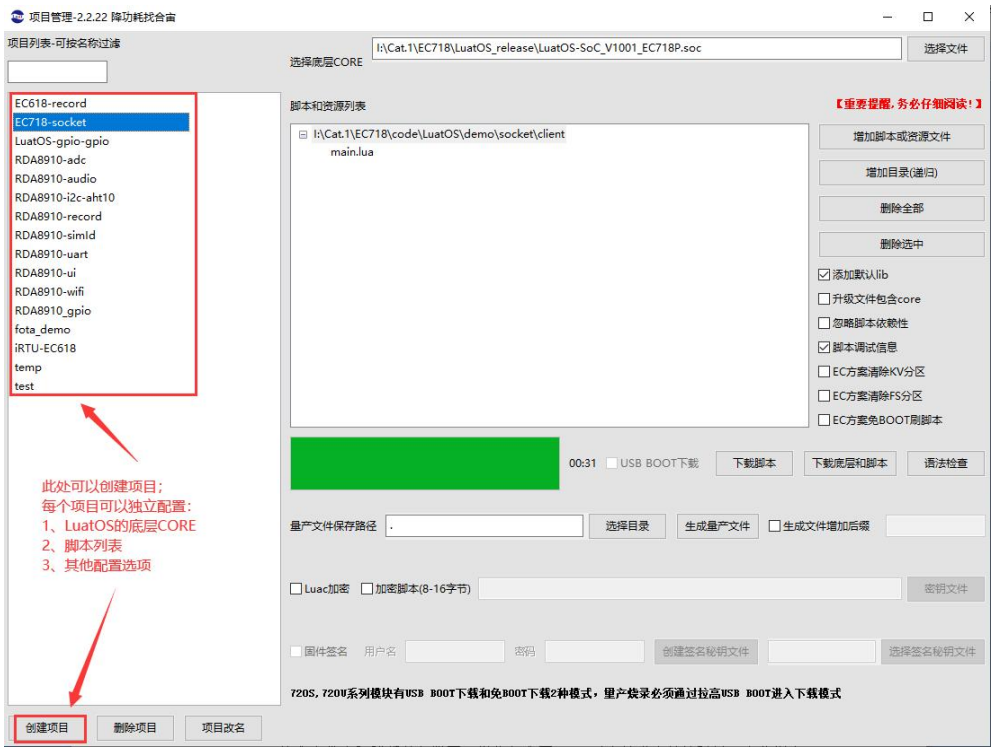


6. 更新完成后，Luatools 文件夹下会生成几个文件夹：

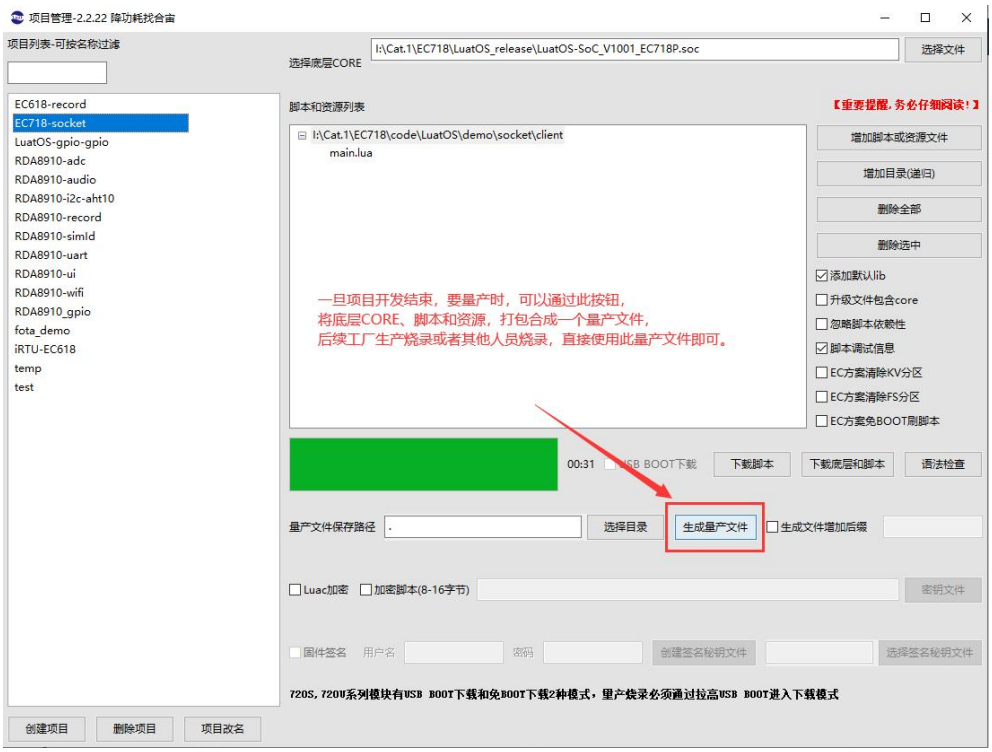


Luatools 核心功能

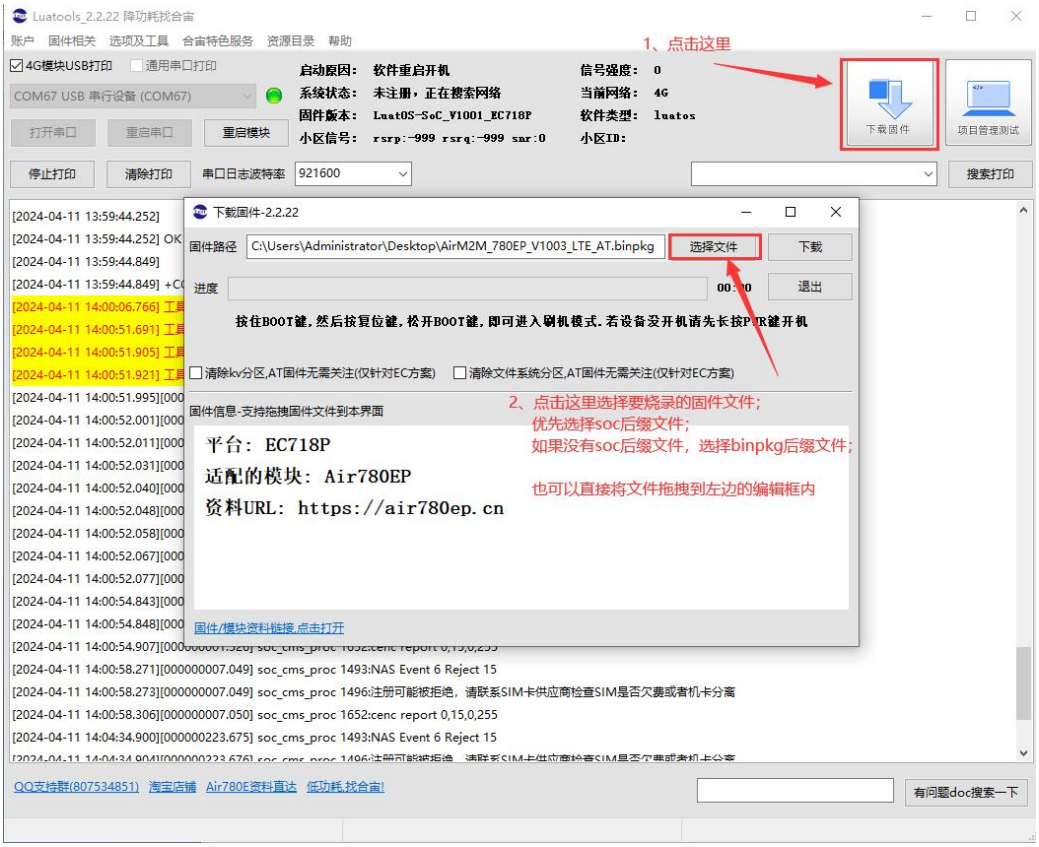
1. LuatOS 二次开发脚本项目管理



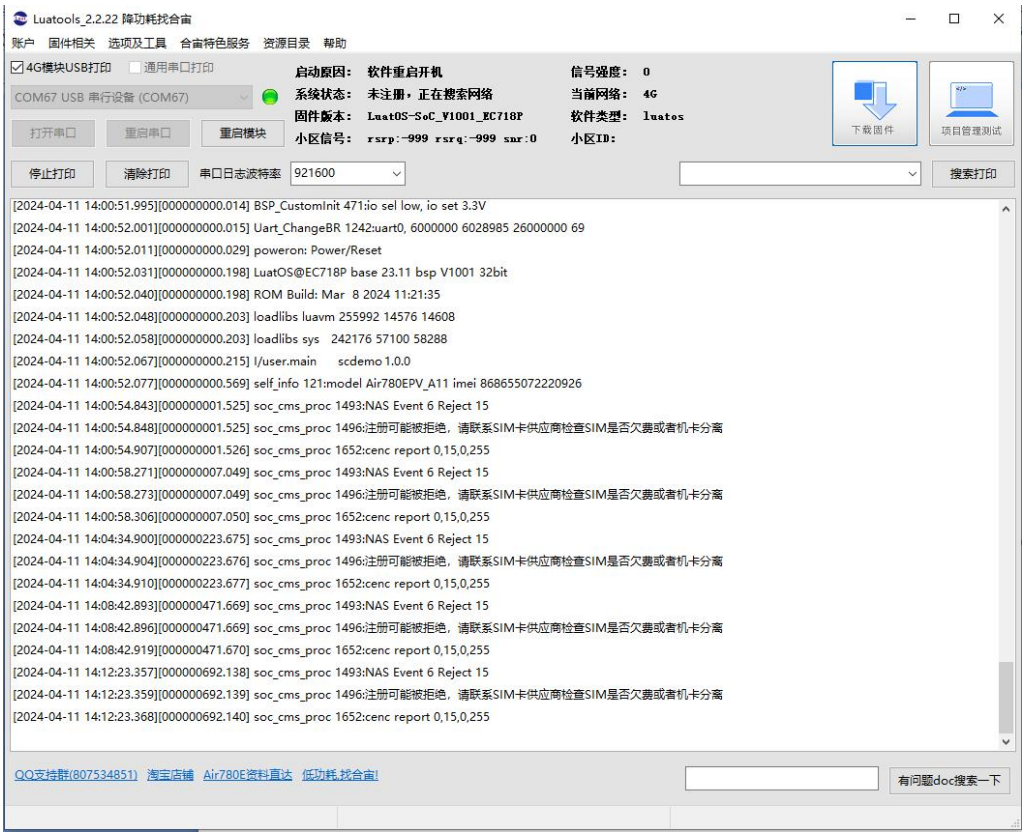
2. LuatOS 二次开发脚本项目生成量产固件



3. 烧录 AT、LuatOS 固件



4. 抓取模组运行应用日志



5. 抓取模组运行底层日志，自动保存在 Luatools 的 log\4gdiag 目录下

Luatools 工具目录介绍

1. _temp：临时文件。
2. config：对 Luatools_V2 进行的一些配置会存放到这个文件夹。
3. log ： 这个文件夹和我们关系紧密，当你向合宙 FAE 寻求技术支持的时候，需要提交这个目录下的文件。log 目录下的 txt 文件是应用日志文件。log/4gdiag 下的文件是底层日志文件。
4. resource ： 合宙模组量产的 AT 固件，LuaatOS 的底层 core 文件、脚本 lib 和脚本 demo。
5. project ： 使用项目管理后会自动生成这个目录，用来管理下载项目的：

Luatools 烧录 Air780EP 固件

参考本文档章节：固件升级 ；

8.3 EPAT 底层日志抓取调试工具

点击右方链接下载 EPAT 工具：芯片原厂 EPAT log 工具 EPAT_V1.3.262.573 ；

8.3.1 EPAT 简介

EPAT：EigenComm Platform Analysis Tools。用于抓取和分析 EigenComm UE Log，对 UE 进行调试和分析。

EPAT offline mode：用于打开 UE log 并显示 log，在 offline mode 下可以启动多个 EPAT 进程,同时打开多 UE log,便于 log 之间比较。

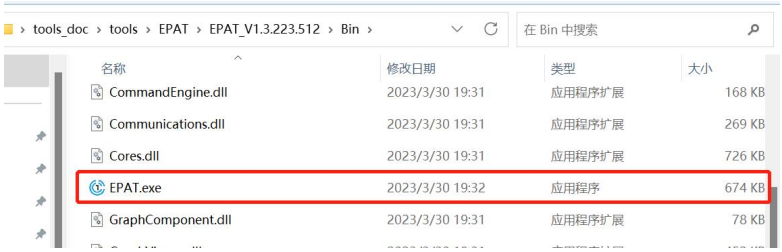
EPAT online mode：用于捕获 UE log 及 UE 在线状态， 在 Online mode 下仅能启动一个 EPAT.

8.3.2 安装

EPAT 启动

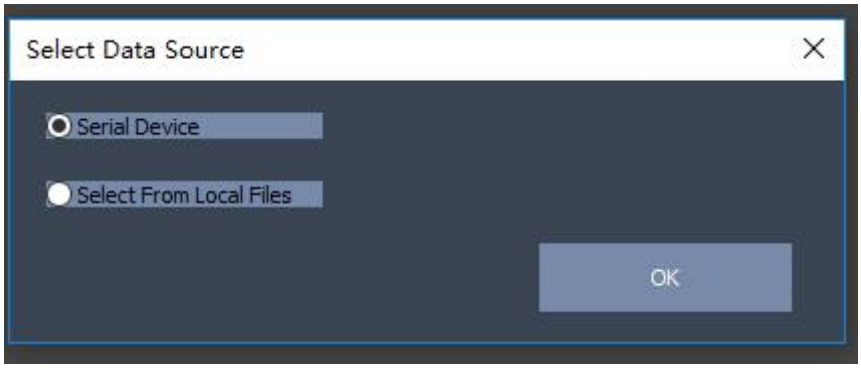
EPAT 绿色安装，解压后即可使用。

在 EPAT\bin 目录下，运行 EPAT.exe。



启动模式选择

启动后将弹出模式选择框，根据需求选择适合你的 mode。



如果联机捕获 UE Log 请选择 Serial Device。

如果查看保存的 zip log，请选择 Select From Local Files。

8.3.3 Online Mode – 联机捕获 Log

连接 UE

EC718 上， USB 和 UART0 都可以抓 log，USB 和 UART0 抓 log 的区别如下：

USB：开机需要在 system 里初始化，初始化完成以前无法抓 log。抓 log 速率高，一般不会丢 log。另外 USB 抓 log 对波特率没有要求；

UART0：初始化比 USB 早很多，可以有效的抓系统开机阶段的 log。抓 log 速率低，业务量大的时候可能会丢 log。另外 UART0 抓 log 对波特率有要求，需要配置合适的波特率。

下面是查询和配置 UART 抓 log 时的波特率的 AT 指令。

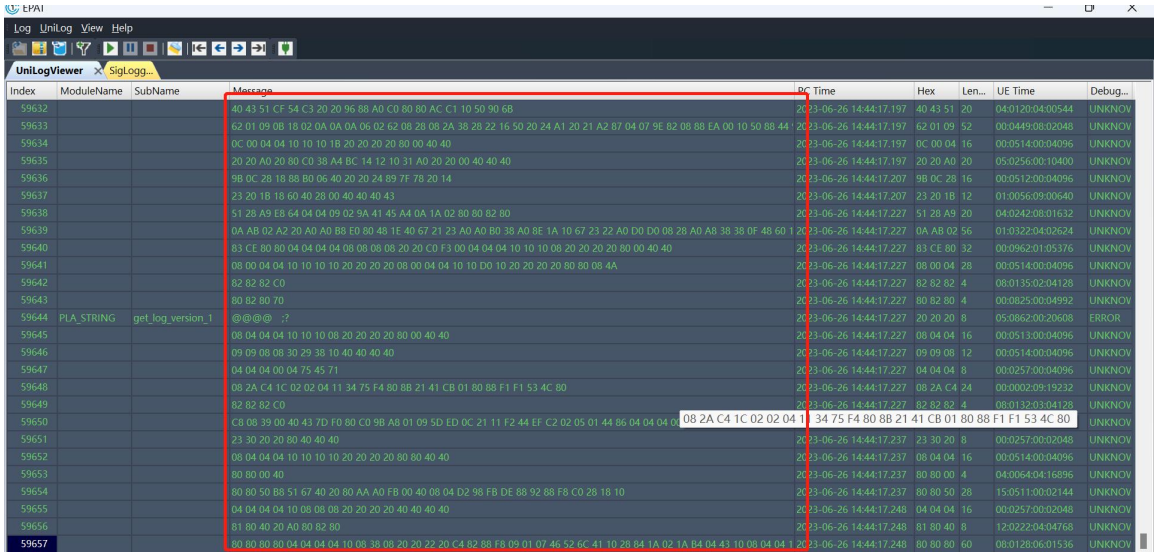
```
AT+ECPCFG?
+ECPCFG: "faultAction":4
+ECPCFG: "uartDumpPort":1
+ECPCFG: "startWDT":1
+ECPCFG: "logCtrl":2
+ECPCFG: "logLevel":0
+ECPCFG: "logOwnerId&logOwnerLevel":0x00000000
+ECPCFG: "logBaudrate":3000000
+ECPCFG: "slpWaitTime":0
+ECPCFG: "logPortSel":2
+ECPCFG: "usbCtrl":0
+ECPCFG: "usbSwTrace":0
```

查询 UART 口抓 log 时的波特率为 3000000

```
AT+ECPCFG="logBaudrate",3000000
OK
```

配置 UART 口抓 log 时的波特率为 3000000。

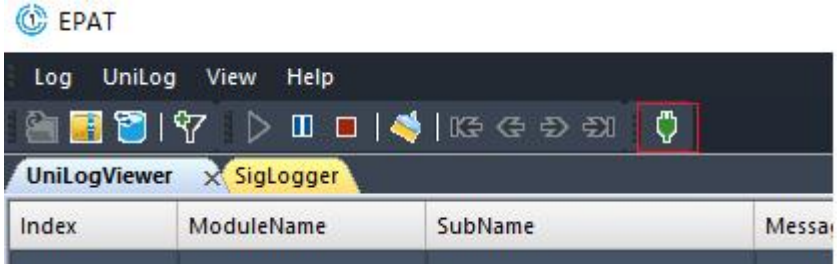
注意：波特率不匹配时，可能导致抓不到有效的 log 信息。如下图，就是配置了不合适的波特率，EPAT 抓不到有效的 log 信息。



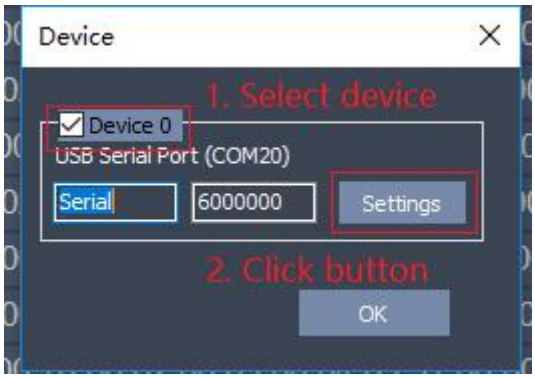
综上：建议客户在开发阶段抓 log 时，同时配置上 USB 和 UART，当同时配置了 USB 和 UART 时，EPAT 会判断 USB 是否能正常抓 log，如果可以，则会优先使用 USB 抓 log，当 USB 抓不到 log 时，会自动切换到 UART 去抓 log。

设备对话框

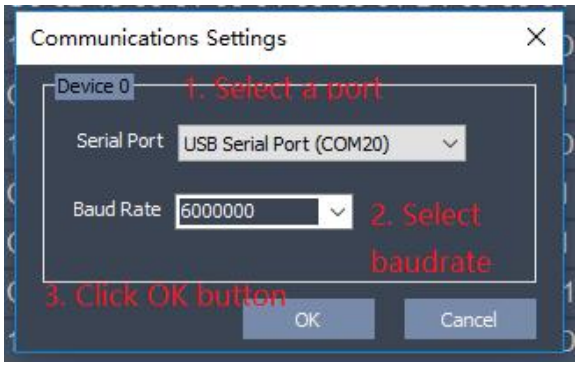
在主界面的工具栏选择 Device Communication 按钮。



将会弹出 Device 设置对话框。



设置连接参数



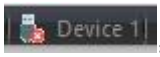
同时配置 USB 和 UART 抓 log

勾选上 Device 0 和 Device 1，设置好端口号和波特率即可。

查看连接状态

在 EPAT 状态栏可以查看连接状态。

：表示连接成功



：表示未连接

：表示连接失败

Update DB

查看状态

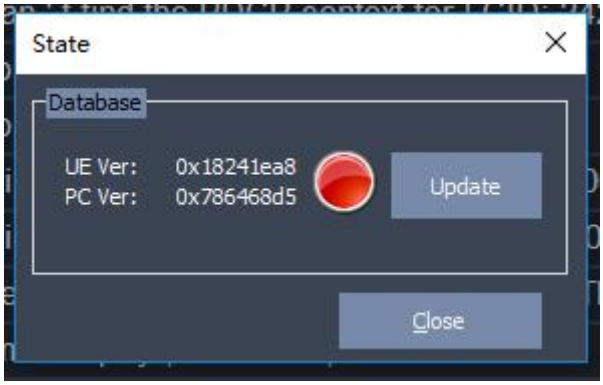
工具栏的 Database State 按钮有三种状态来表示数据库的匹配状态。

：UE DB 读取失败

：DB 不匹配

：DB 匹配

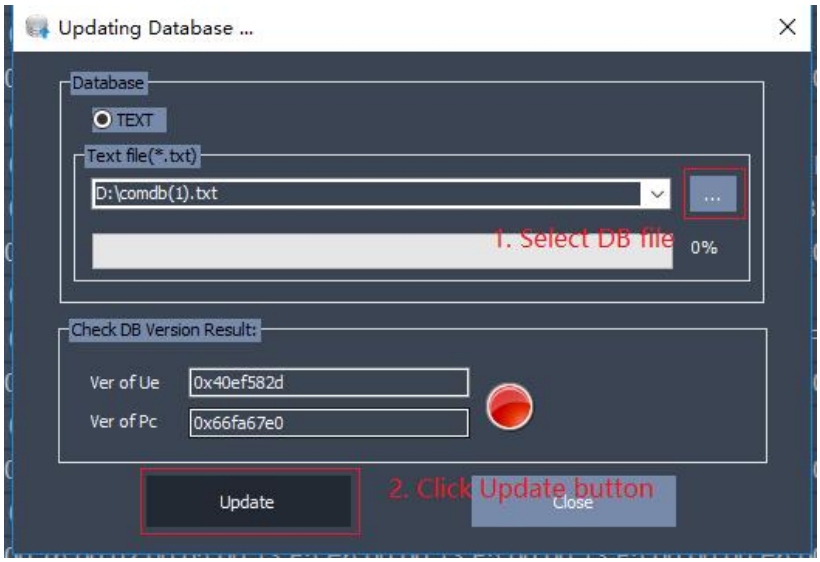
点击上面的 Database State 按钮，弹出详细数据库信息界面



如果手动更新数据库，点击 Update 按钮。

更新数据库

在更新数据库界面，手动更新你的数据库文件。**在抓 LOG 之前一定需要匹配 LOG 数据库，Database State 按钮必须是绿色。**

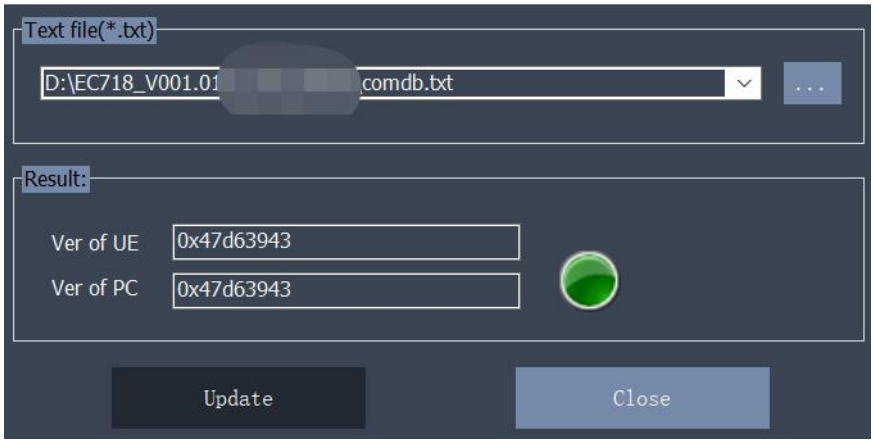


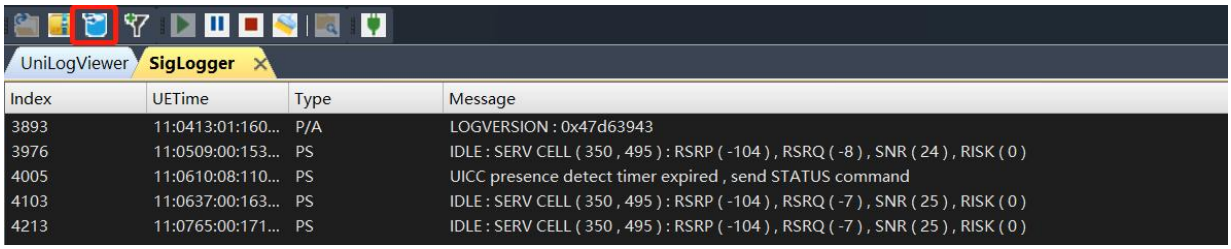
更新后，上图 Check DB Version Result 区域显示结果：

绿色：成功且匹配

红色：不匹配

灰色：读取 UE DB 失败。





Start/Pause/Stop/Clear 功能

Pause

表示 Viewer 不刷新 log 显示，但继续保存 Log. 用于不确定当前的 log 是否已经发现 UE 的故障，仅临时查看一下，log 继续写情况下。

Stop

表示 Viewer 不刷新 log 显示，并停止保存 log. 用于确认当前的 log 已经想要保存、后面接受的 log 不需要的保存的情况下。

Clear

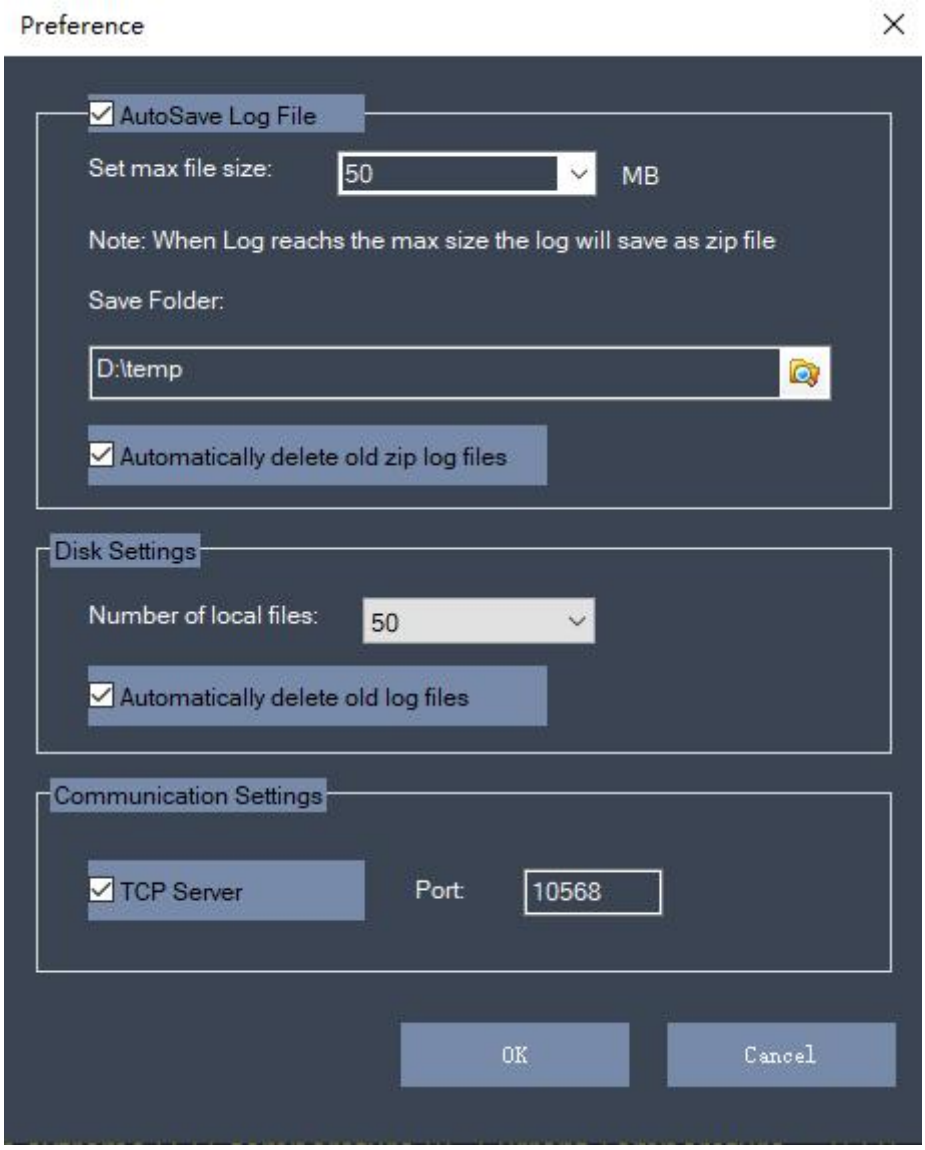
清空 Viewer ,并新开 log 文件继续写 log.

导出/保存 Log 文件

选择工具栏的 Save 按钮或者 CTRL+S 快捷键，弹出保存 Log 对话框。保存 log 文件到本地磁盘。

偏好设置

点击主菜单上的 Log->Preference，在弹出的对话框中可以设置自动保存和相关功能。



自动保存

在选中 AutoSave Log File 后，Set max file size 中设置文件大小，当 log 文件达到此大小时自动保存到磁盘。在 Save Folder 输入框中指定保存文件的位置。

如果选项 Automatically delete old zip log files 被选中，自动保存的 zip log 文件数量如果超过预设数量的上限，旧的文件会被自动删除。此数量的默认值是 50.

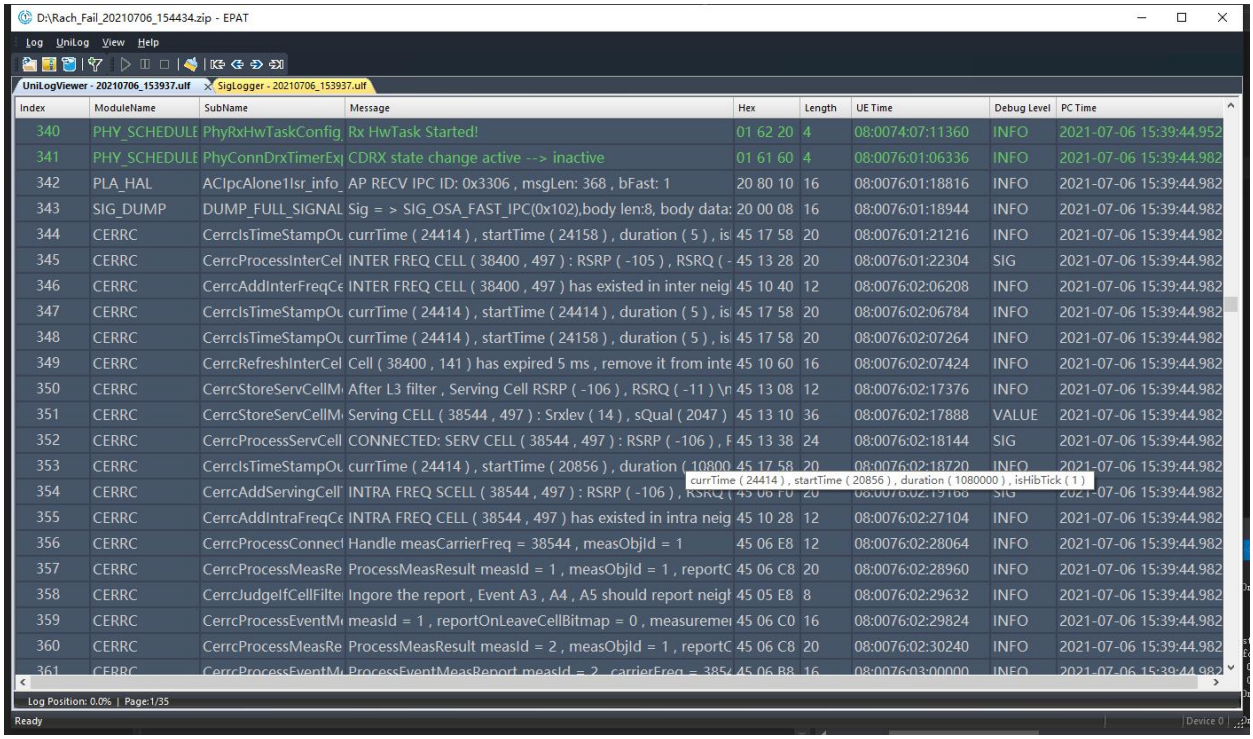
本地文件的设置

选中 Automatically delete old log files 后，本地记录的 log 文件如果超过指定的数量会自动删除旧的文件。此数量由 Number of local files 确定。

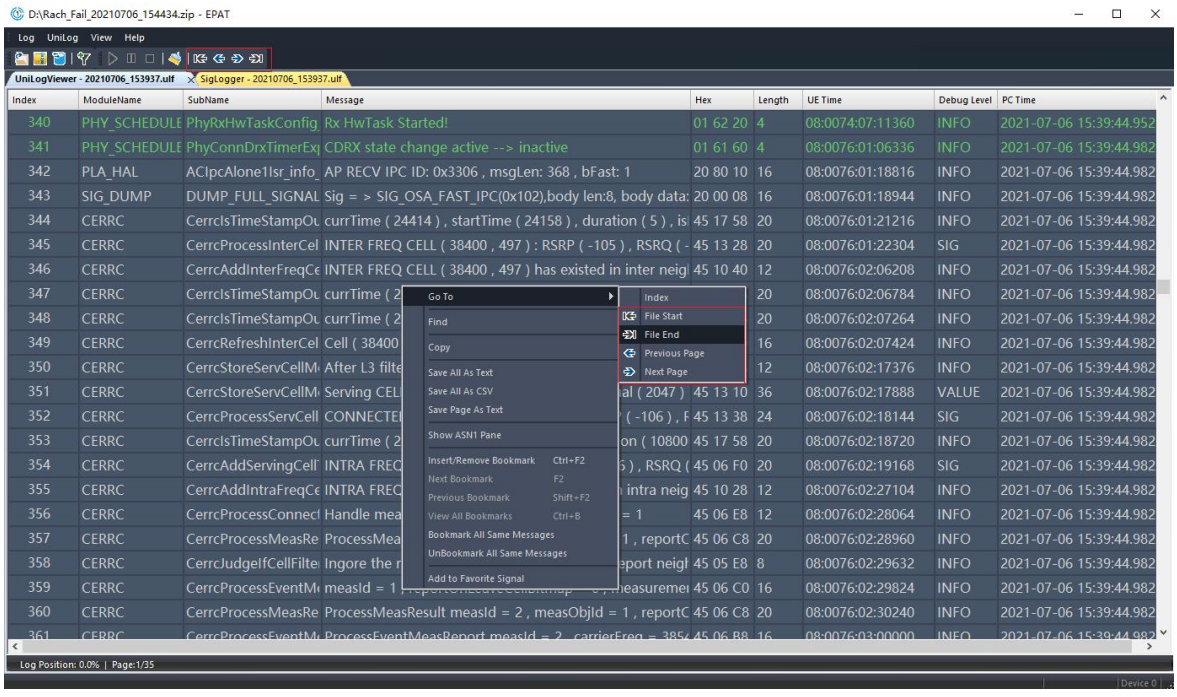
8.3.4 Offline Mode – 查看 Log

载入 Log 文件

点击工具栏的 Open 按钮或者快捷键组合 CTRL+O，选择目标 log 文件。即可打开此文件。



翻页功能

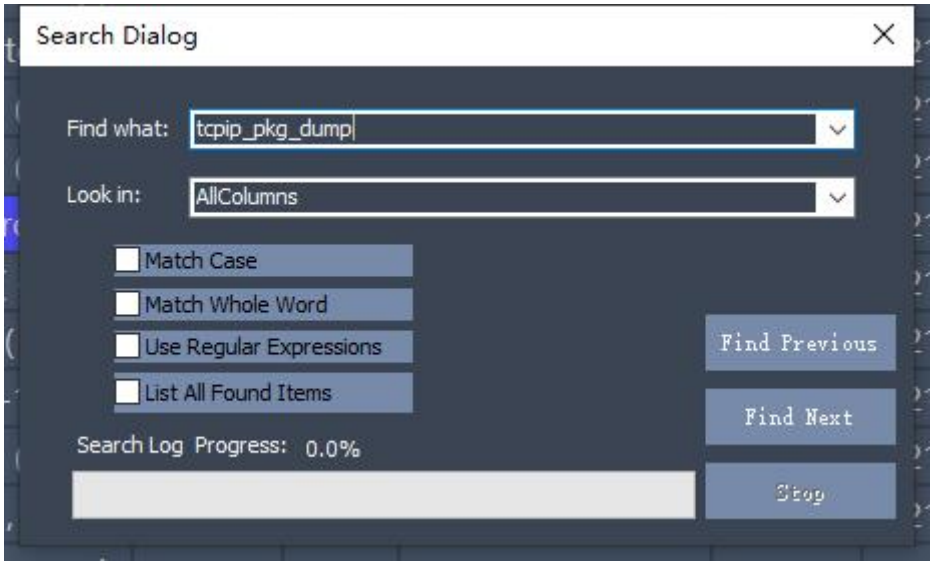


选择工具栏或者在信息窗口右键菜单，可以使用翻页功能。

分别有前一页、后一页、文件开始、文件末尾四种翻页方式。

查找

信息窗口右键菜单中选择 Find 或者快捷键组合 CTRL+F。弹出查找对话框。



Log 保存

信息窗口右键菜单中可以选择保存 Log 功能。

目前支持的方式有：

1、 保存当前 Log 到 Text 文件。

2、 保存当前 Log 到 CSV 文件。

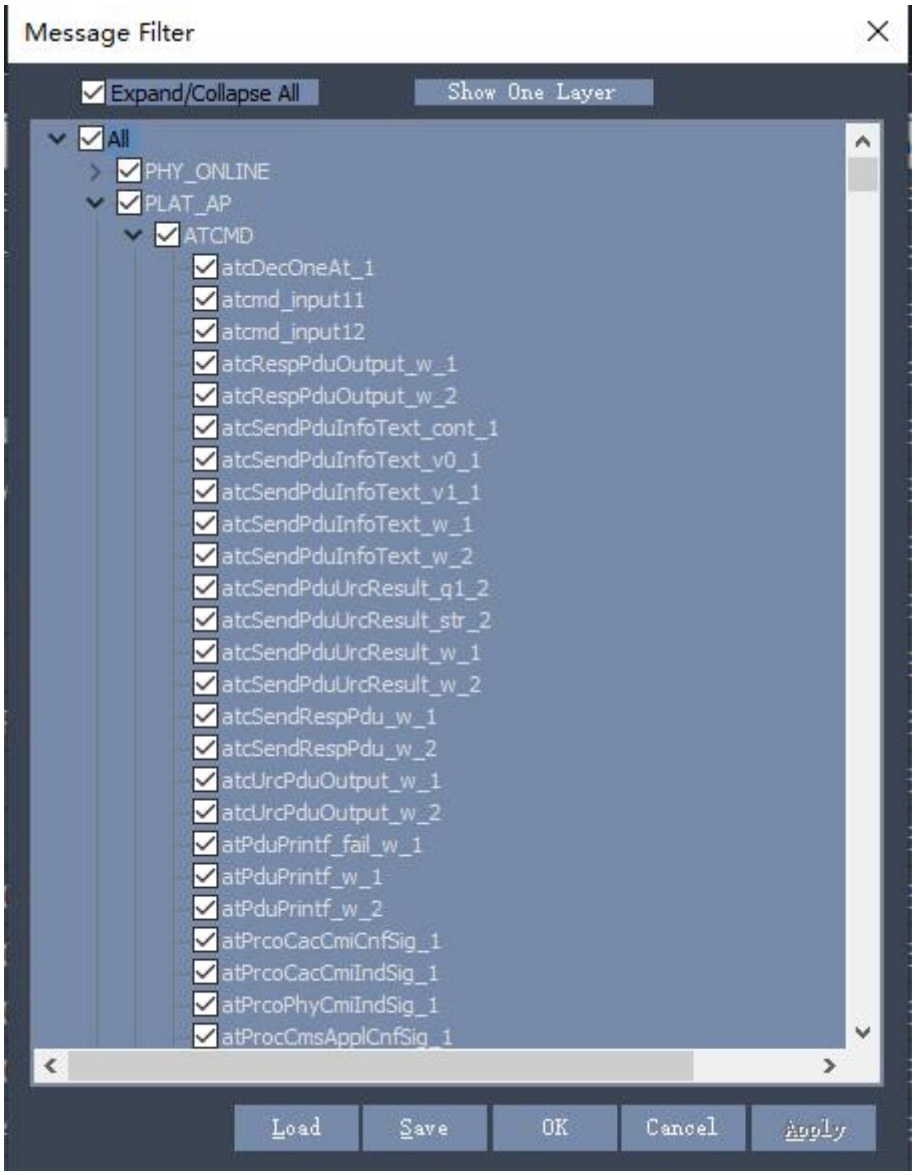
3、 保存当前页 Log 到 Text 文件。

同步跳转

在 UnilogViewer 或者 SigLogger 界面中双击任意一条 log，将会自动跳转到其他界面上对应的位置。

8.3.5 Message Filter 功能

选择工具栏的 filter 按钮，将弹出 Message Filter 对话框。



当前列表中显示当前系统所有的 signal，通过单个 signal 名字前面的 checkbox 可以看出此 signal 是显示或者不显示状态，可以通过在界面上的操作选择你所期望显示的 message。

Load

载入事先保存在本地磁盘的 filter 配置文件。

Save

在界面上设置完成后，将当前设置保存到本地磁盘。可以通过上面的 Load 功能加载这个配置文件。

Ok

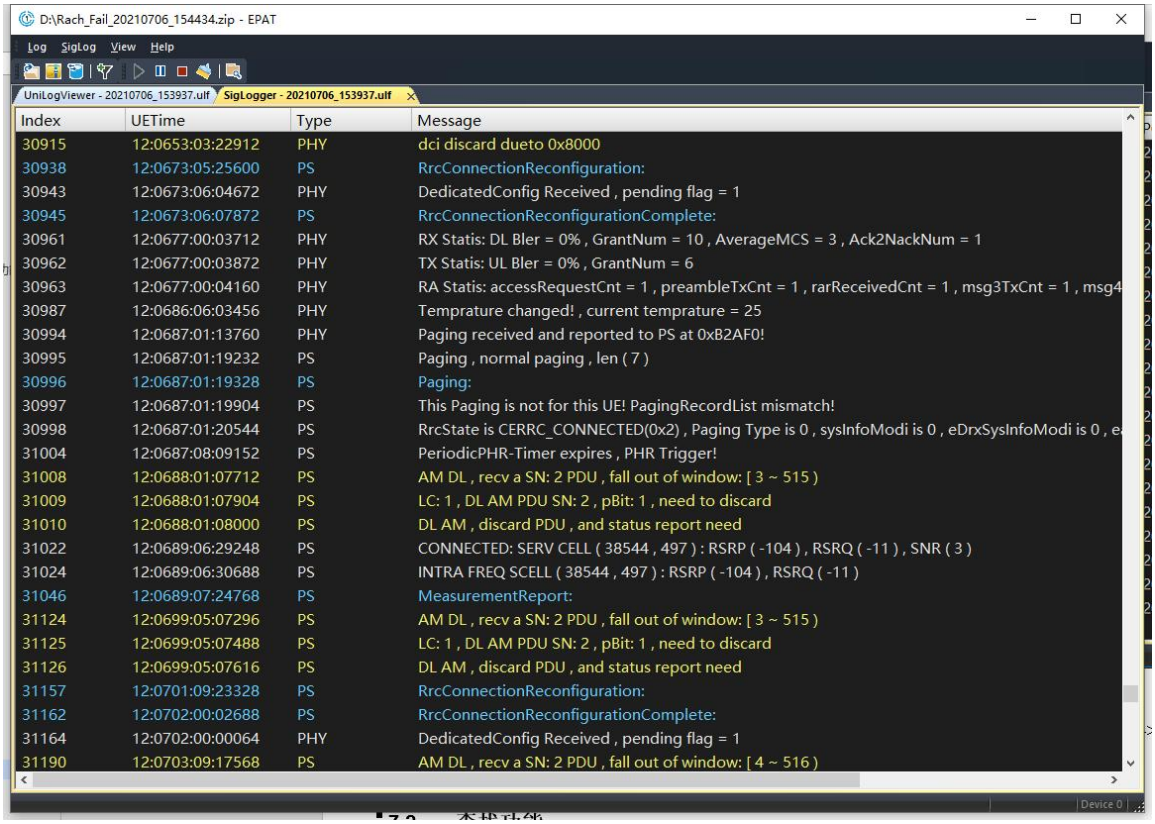
确定应用当前配置到整个 EPAT 并使之立即生效，并关闭当前对话框。

Apply

应用当前配置到整个 EPAT 并使之立即生效。

8.3.6 SigLog

查看 SigLog

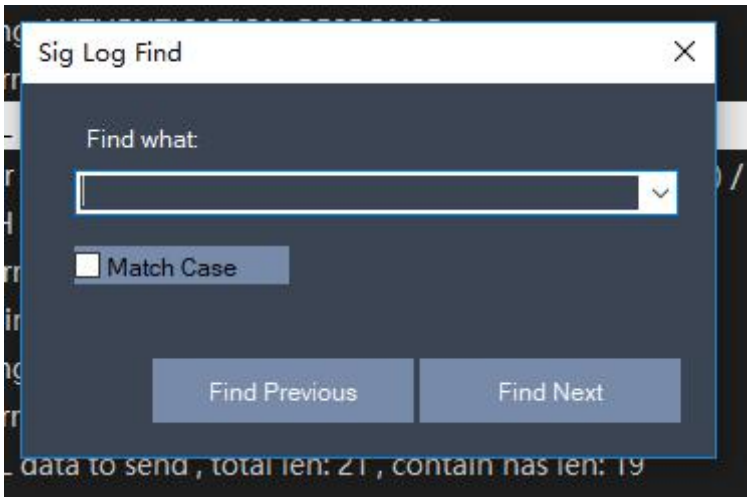


主界面选择 SigLogger Tab 页可以查看 SigLog，如果此界面不可见，可以通过主菜单的 Modules->SigLogger 来打开。

如果 log 显示不同的颜色，表示 log 拥有不同的 Log Level。

查找功能

通过菜单 SigLog->Find 或者快捷组合键 CTRL+F 弹出如下的查找对话框，可以查找当前页面的内容。



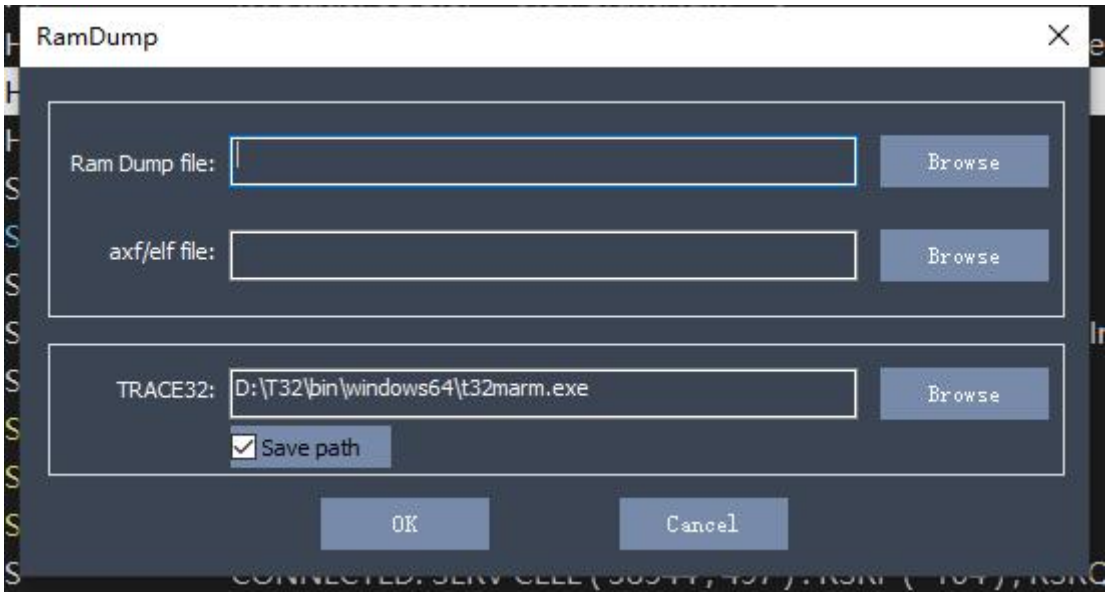
Export Pcap

通过菜单 SigLog->Export As pcap file，可以将当前 log 文件中相关的数据保存为 pcap 格式文件。

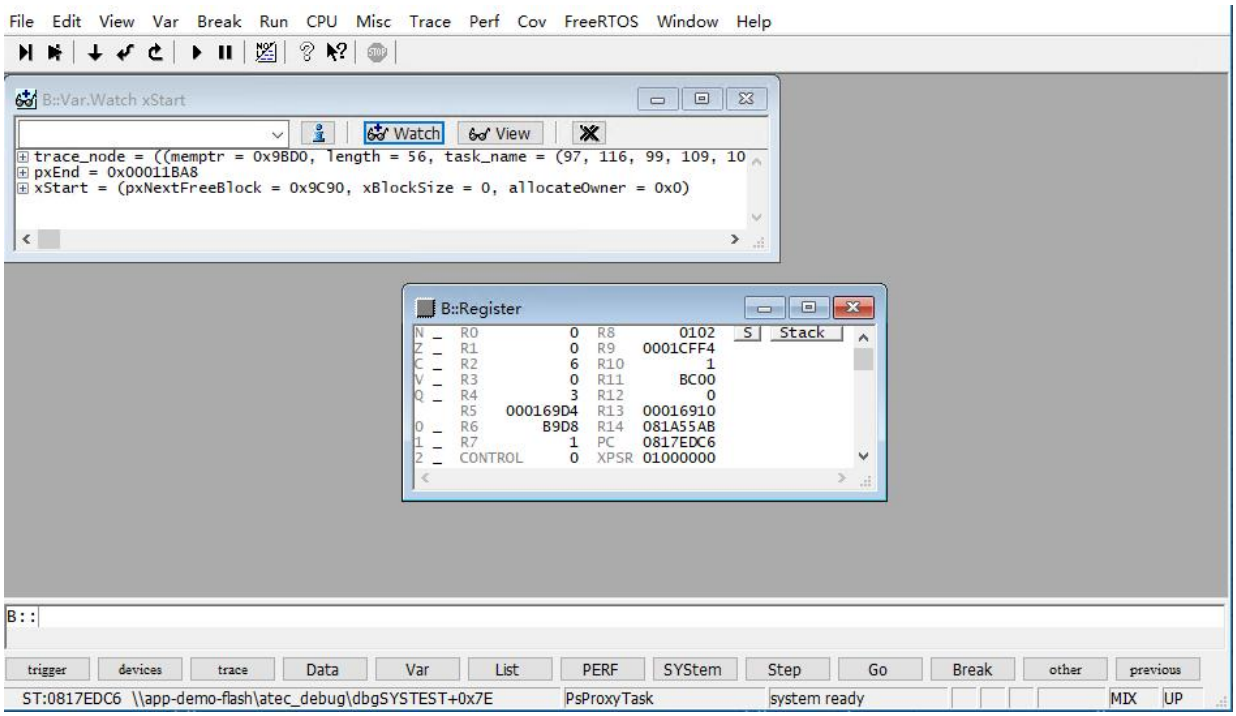
8.3.7 RamDump 功能

触发

当我们的设备发生 Assert 或者 HardFault 错误时，本软件自动检测到此状态并打开如下对话框。



1、 Ram Dump File：默认情况下设备发生错误时生成的 Dump 文件在当前目录下，如果不正确可以通过 Browe 按钮来选择正确的 Dump 文件。
2、 Axf/elf File:设备中 firmware 对应的 axf 或者 elf 文件。
3、 TRACE32：通过 Browse 按钮查找本地电脑安装的 TRACE32 主程序。
4、 以上的信息都正确选择后，点击 OK 按钮将自动运行 TRACE32 程序。
运行 TRACE32 的结果大致如下：



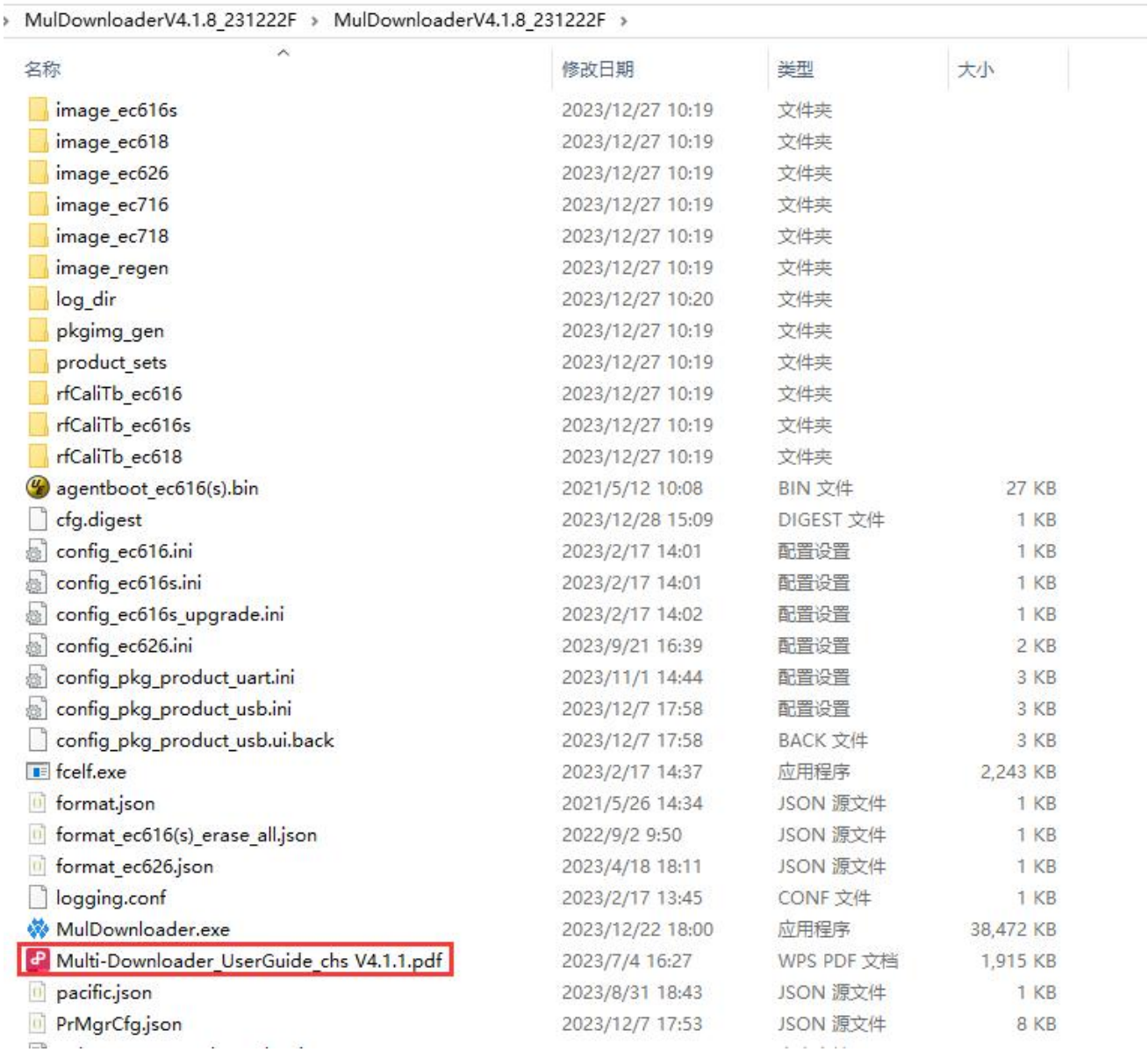
8.4 量产多路下载工具

工具下载

点击右方链接下载工具：MulDownloaderV4.1.9_240511F.7z ；

使用手册

下载上面的量产工具，解压后的 Multi-Downloader_UserGuide_chs V4.1.1.pdf 及为使用说明，如下图所示。

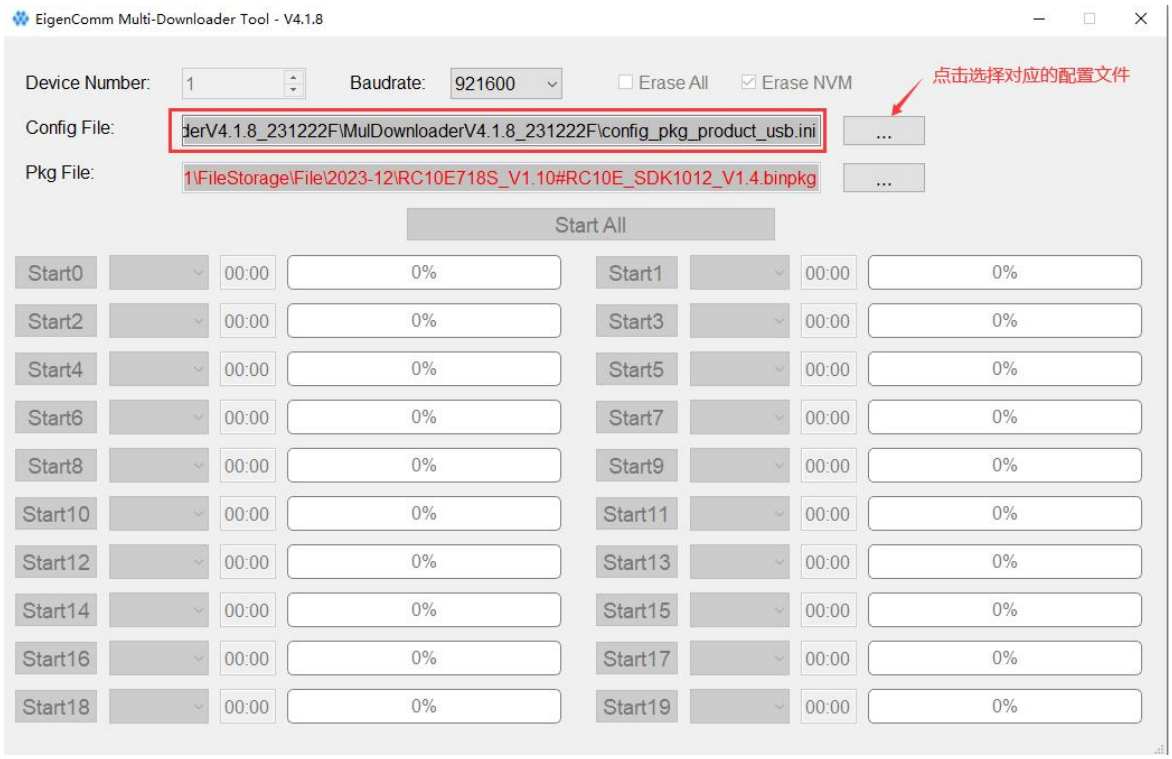


USB 下载

注意：USB 下载需要在 BOOT 模式下进行。

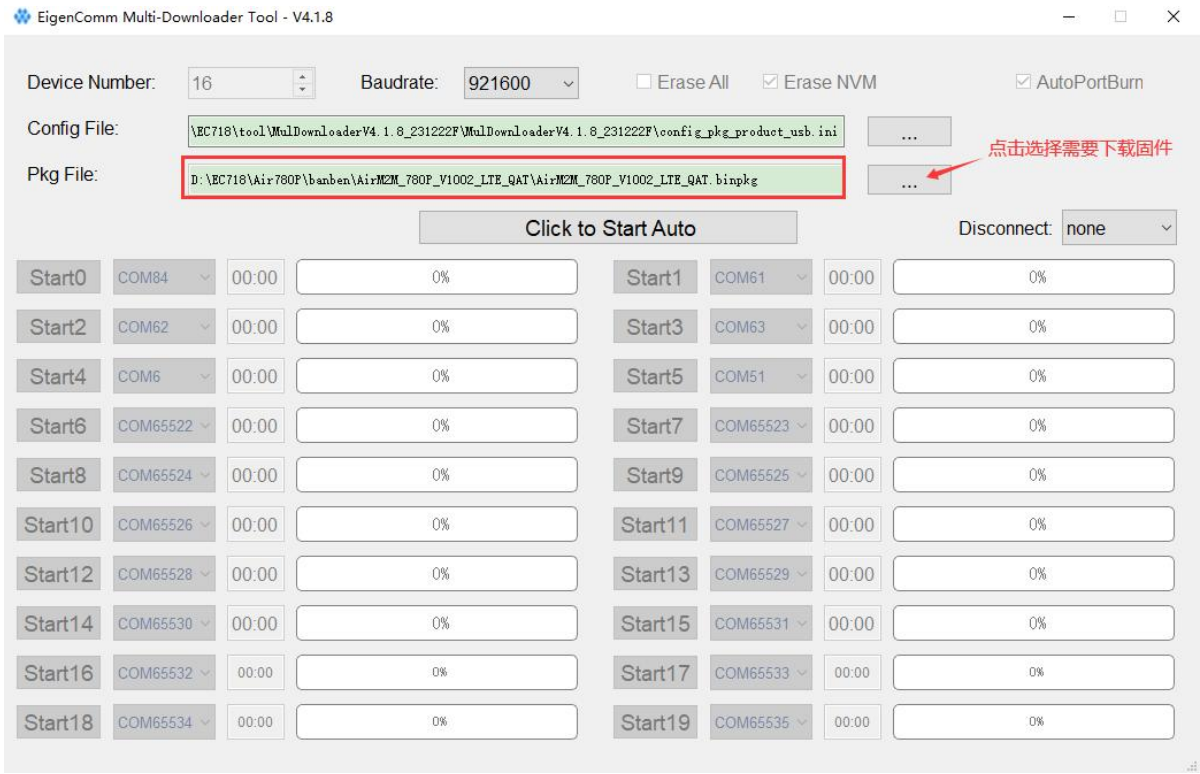
选择配置文件

点击 MulDownloader.exe 运行下载工具，选择对应配置文件。这里 USB 下载方式选择的配置文件为 config_pkg_product_usb.ini 如下图所示：



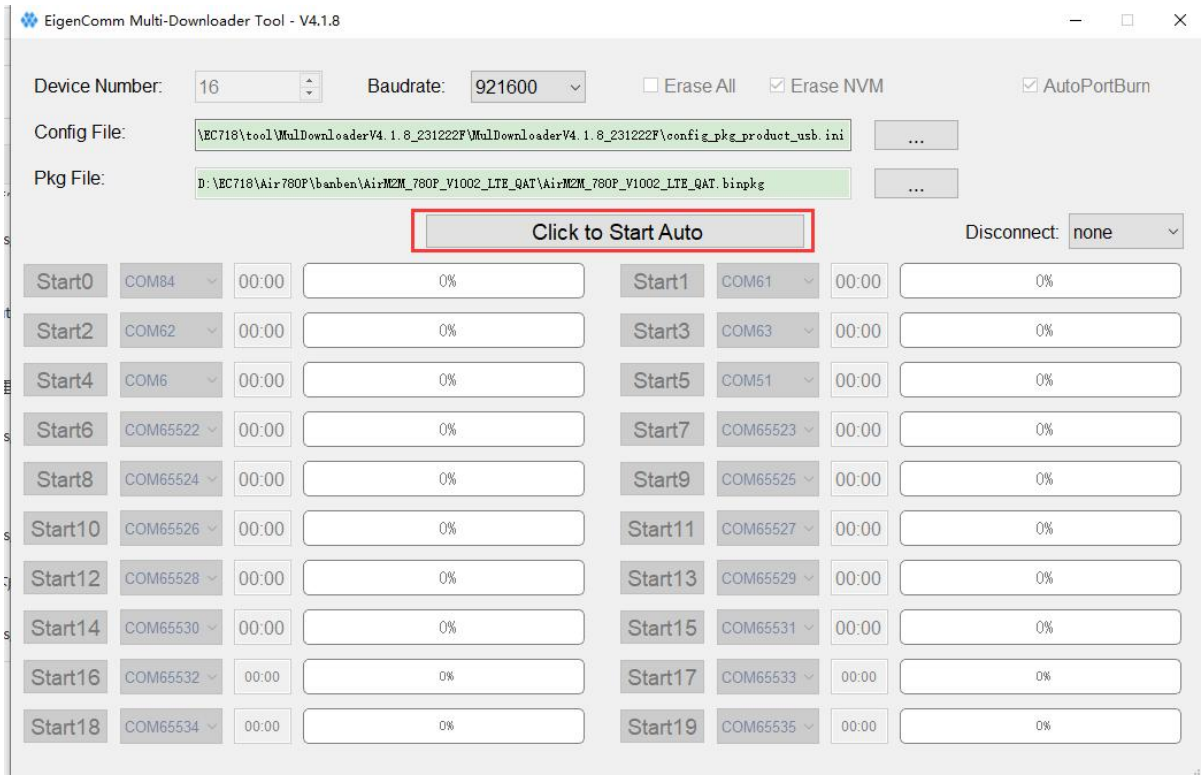
选择需要下载的固件

点击 Pkg File 后面的设置按钮，选择 Pkg File 文件及模块需要下载的固件，这里选择的固件是 AirM2M_780P_V1002_LTE_QAT.binpkg。如下图所示：

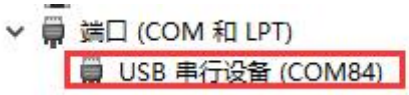


下载

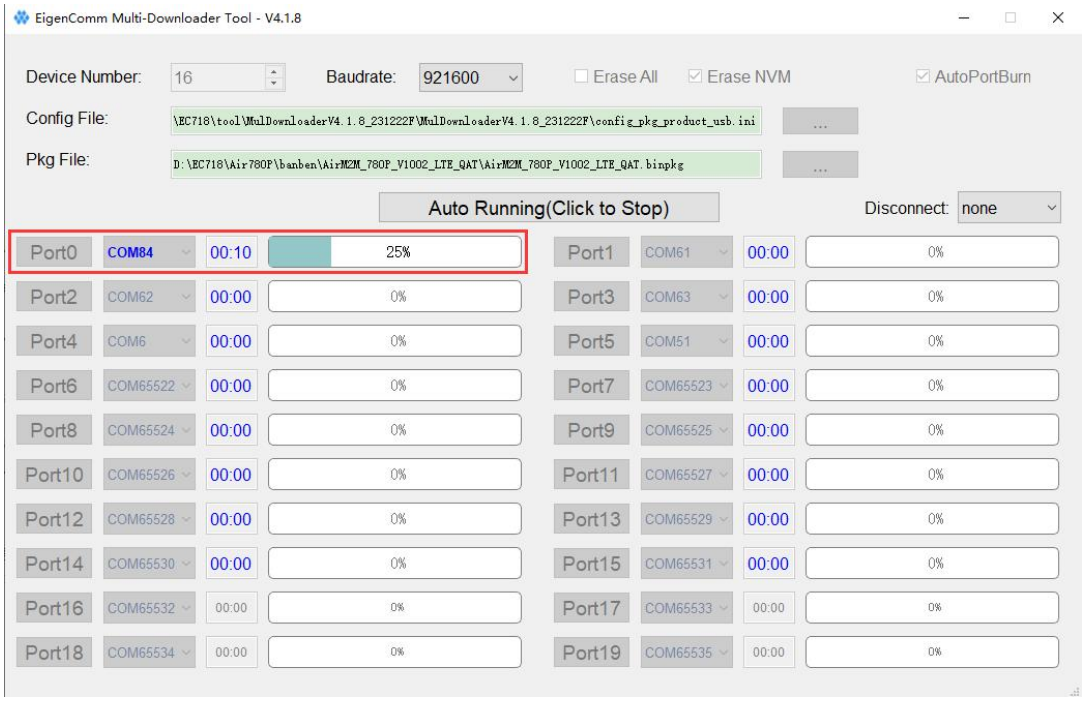
先按住 BOOT 键，点击 Click to Start Auto，在按开机键（开机状态下按复位键）进行下载，如下所示。



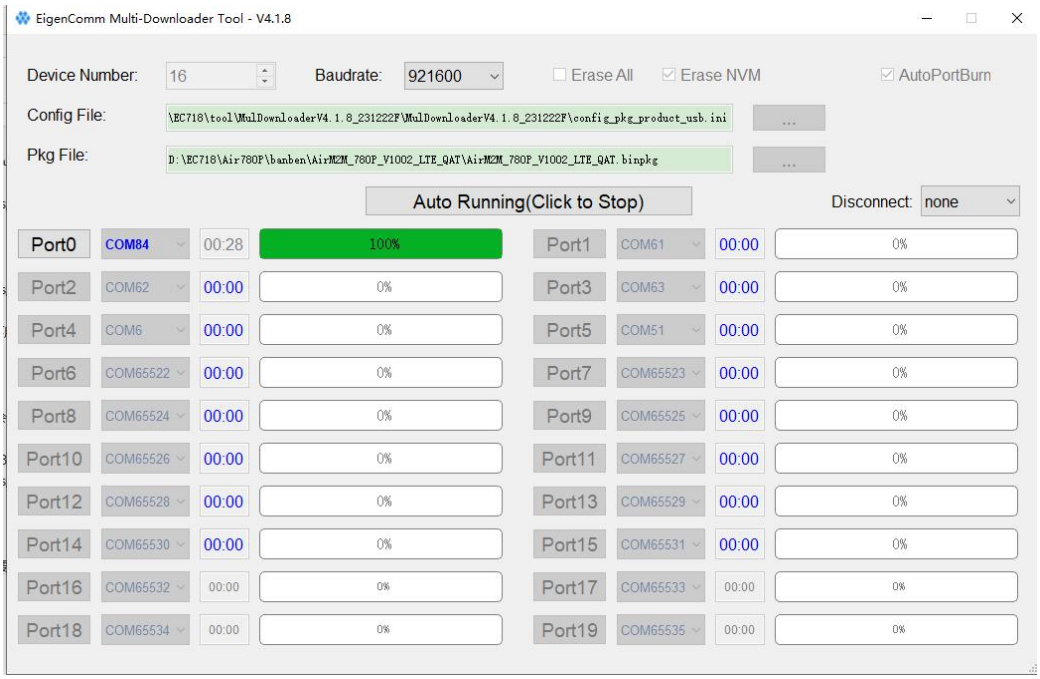
工具会自动识别下载口进行下载，这里的下载端口为 84。



下载过程会显示下载进度，如下所示。

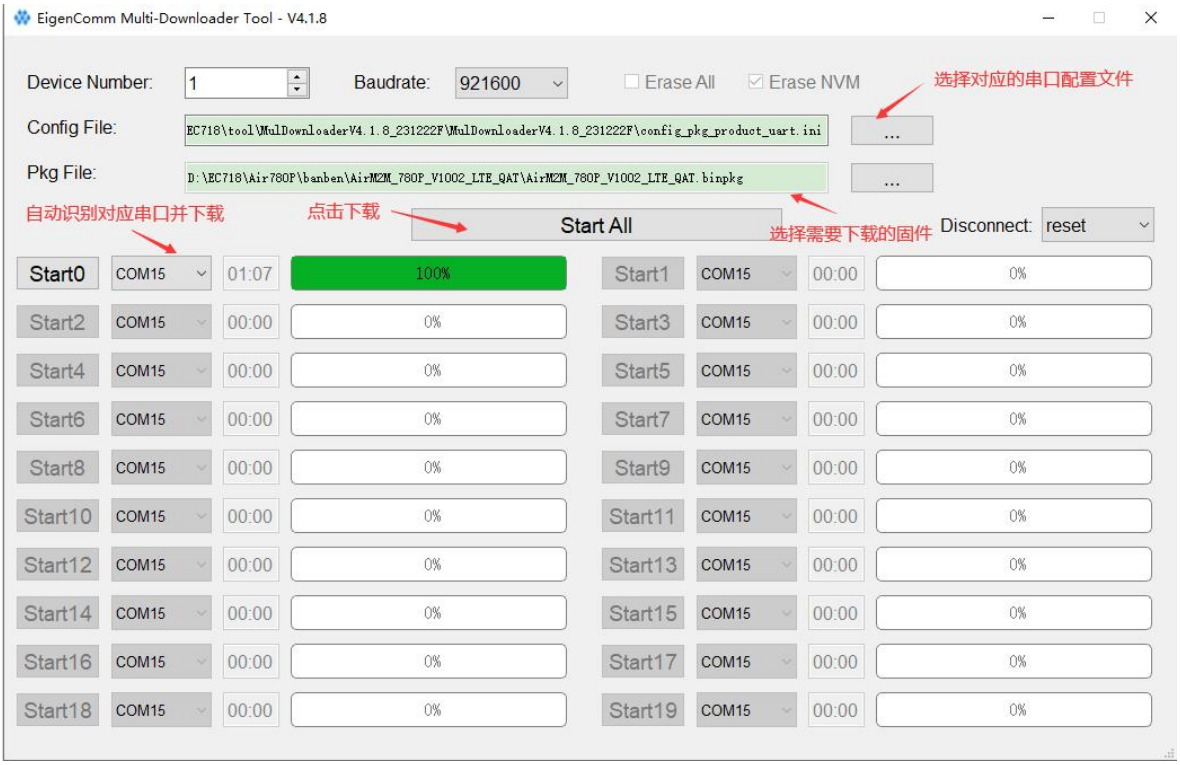


下载完成会显示现在进度为 100%，如下所示。

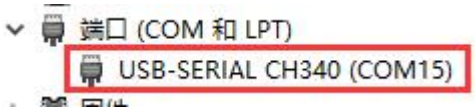


串口下载

可以用主串口来下载对应版本，这里需要选择文件为：config_pkg_product_uart.ini。选择需要下载的固件这里选择的固件是 AirM2M_780P_V1002_LTE_QAT.binpkg。点击 Start ALL 后，重新上电开机就会开始下载。



这里的下载端口为 15。



注意事项

文件路径中不能有中文目录，否则会下载失败

默认是如下配置，不需要去修改。

九 软件开发指南（AT）

9.1 AT 开发简介

AT 命令采用标准串口进行数据收发，将以前复杂的设备通讯方式转换成简单的串口编程，大大简化了产品的硬件设计和软件开发成本，这使得几乎所有的网络模组如 GPRS、3G/4G 、NB-IoT、蓝牙、WiFi 、GPS 等模组都采用 AT 命令。

随着物联网时代的发展，AT 命令被广泛的用于单片机（MCU）+ 网络模块 这样的物联网开发模式。但这种开发模式单片机是必不可少的部分，因而会使产品的成本相应的高些。

AT 固件是标准固件，所以能支持的驱动也比较少，只有模块状态灯、摄像头扫码、SD 卡等外设驱动功能。要想要支持更多的外设建议用 LuatOS 二次开发固件进行二次开发。

AT 命令语法介绍

本手册中所有命令行必须以"AT"或"at"作为开头，以回车（<CR>）作为结尾。响应通常紧随命令之后，它的样式是“<回车><新行><响应内容><回车><新行>”(<CR><LF><响应内容><CR><LF>) 。整本手册里，只有<响应内容>被自始至终介绍，而<回车><新行>被有意省略了。

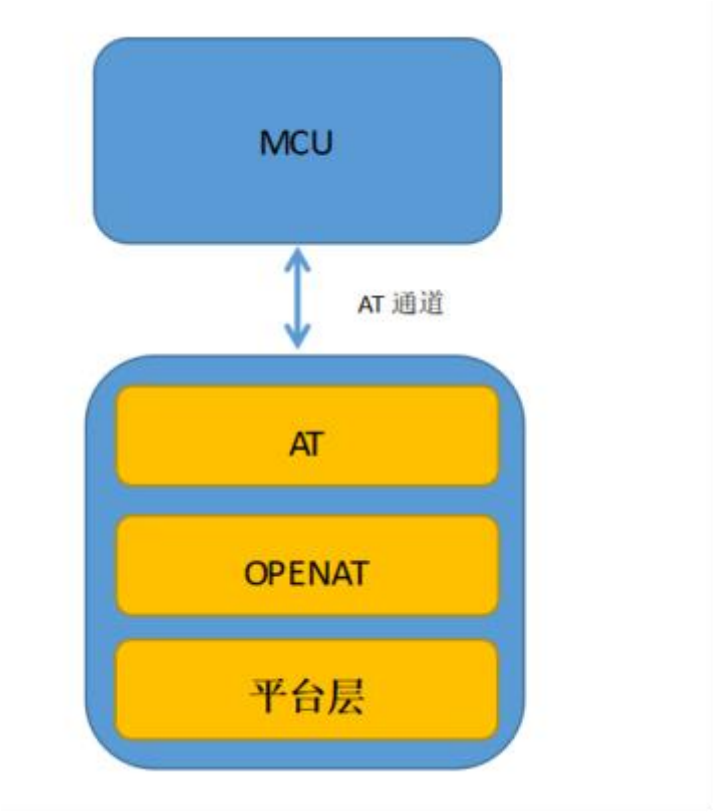
合宙无线模块提供的 AT 命令包含符合 GSM07.05、GSM07.07 和 ITU-T Recommendation V.25ter 的命令。

所有 AT 命令从语法上可以分为三类："基础类"，"S 参数类"以及"扩展类"，分别介绍如下：

- 基础类 AT 命令
- 这类 AT 命令具有"AT<x><n>"或者"AT&<x><n>"格式，其中"<x>"是命令；"<n>"可以是一个或者多个参数。例如："ATE<n>"，该命令用于为 DCE 开关回显功能，即 DCE 会依据"<n>"值决定是否把接收的字符回显给 DTE。"<n>"是选配参数，若没有赋值，模块将使用默认值。
- 参数类 AT 命令
- 这类 AT 命令格式为"ATS<n>=<m>"，其中"<n>"是 S 寄存器的索引；"<m>"是赋予的值。"<m>" 是选配参数，若没有赋值，模块将使用默认值。
- 扩展类 AT 命令
- 一般来讲，扩展类命令依据功能可以分为以下几种类型：

命令类型	语法	说明
测试命令	AT+<X>=?	该命令用于查询设置命令或内部程序设置的参数以及其取值范围
查询命令	AT+<X>?	该命令用于返回参数的当前值
设置命令	AT+<X>=<…>	该命令用于设置用户自定义的参数值
执行命令	AT+<X>	该命令用于读取受 GSM 模块内部程序控制的不可变参数
扩展类命令语法：		
在命令行 TA 返回结果中<>内为必配参数，[ ]内为选配参数；		
在每条命令中，选配参数和必配参数必须按照规定的顺序排列，各参数间必须用逗号隔开；		
字符串参数一般放在双引号中；		
实际使用中，<>，[ ]不必输入。		

AT 开发基本架构



AT 开发优缺点

AT 开发的优势：AT 指令作为单片机和通信模块的桥梁，很好的兼顾了单片机开发的特点，又满足了产品联网的需求。AT 指令是一种通用的使用方式，在无线模块领域也流行了 20 多年。

AT 开发的缺点：成本相应的高些，能支持的驱动也比较少。

9.2 AT 固件

尚未对外正式发布，请等待；

9.3 AT 命令手册

参考本文档 第 20 章 AT 命令手册

9.4 AT 开发环境搭建

AT 命令端口

Air780EP 开发板连接到电脑后，有两种类型的端口可以支持发送 AT 命令：

1、 USB 虚拟串口，详情参考： AT 固件中虚拟串口的作用 中的 0006 号和 0002 号虚拟串口；

2、 开发板上 MAIN UART 在电脑端对应的串口；

AT 命令发送

1. 下载 llcom 串口工具，做如下设置。

2. 选择对应 AT 命令端口，然后打开串口。

3. 输入对应 AT 命令点击发送。

9.5 AT 开发应用指南

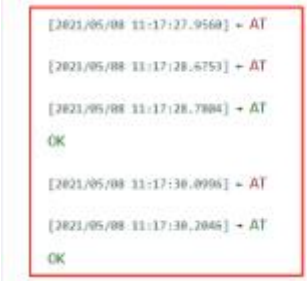
9.5.1 TCP(SSL)应用指南

TCP 示例

开机

通过拉低 powerkey2 秒进行开机，开机以后通过串口循环发送 AT 直到收到 OK，如果 90 秒没有收到 OK 请拉低 RESET_IN_N 引脚 150ms 以上。

开机发送 AT 训练波特率



查询 SIM 卡状态

发送 **AT+CPIN?** 查询 SIM 卡状态，直到收到+CPIN: READY，模块主动上报 SIM REMOVED 和查询返回+CME ERROR: 10 是模块没有检测到 SIM 卡，如果 10s 内没有收到建议重启模块，如果重启问题依旧需要检查卡是否有插好，或者排查板子的硬件连接。

```
LLCOM - 能跑Lua脚本的串口调试工具 - 1.0.5.8

[2021/05/08 11:21:27.4199] ← AT+CPIN?

[2021/05/08 11:21:27.5259] → AT+CPIN?

+CPIN: READY

OK
```

查询模块信号强度

一般信号强度 18 以上才能稳定通讯；11-17 能连接，会有断网的可能，10 以下基本连接不上，在不能联网的情况下，可以先参考一下信号强度如何。

```
LLCOM - 能跑Lua脚本的串口调试工具 - 1.0.5.9

[2021/06/02 14:21:34.3062] → AT+CSQ

+CSQ: 30,99

OK

[2021/06/02 14:21:39.3047] → AT+CESQ

+CESQ: 99,99,255,255,27,56

OK
```

查询网络注册情况

AT+CGATT? 查询是否注册网络收到+CGATT: 1

值是 1 即为注册成功，正常情况下注册时间不会超过两分钟，如果超过两分钟没有注册可以进入飞行模式五秒后退出再查询，或者直接重启模块。

值是 0 则没有注册成功

激活网络

AT+CSTT 配置网络，非私有 APN 以外 Cat1 的固件支持根据卡自动配置 APN，直接输入 AT+CSTT 即可，模块会按照自动获取的 APN 设置 CSTT 的 APN。

AT+CIICR 激活网络，在 IP START 的状态使用 AT+CIICR 激活网络

AT+CIFSR 查询是否获取 IP，如果成功就可以开始配置 TCP 连接了，如果不成功使用 AT+CIPSHUT 关闭移动网络，从 AT+CSTT 重新进行。一定要查询一次再进行服务器连接。

```
LLCOM - 能跑Lua脚本的串口调试工具 - 1.0.5.8

[2021/05/08 11:26:02.0621] ← AT+CSTT

[2021/05/08 11:26:02.1682] → AT+CSTT

OK

[2021/05/08 11:26:46.0582] ← AT+CIICR

[2021/05/08 11:26:46.1652] → AT+CIICR

OK

[2021/05/08 11:26:48.6641] ← AT+CIFSR

[2021/05/08 11:26:48.7711] → AT+CIFSR

10.14.71.143
```

连接 TCP

此处使用的是非 SSL 连接

TCP/UDP 测试网页 TCP/UDP 测试网页

使用 AT+CIPSTART 来建立连接

发送数据

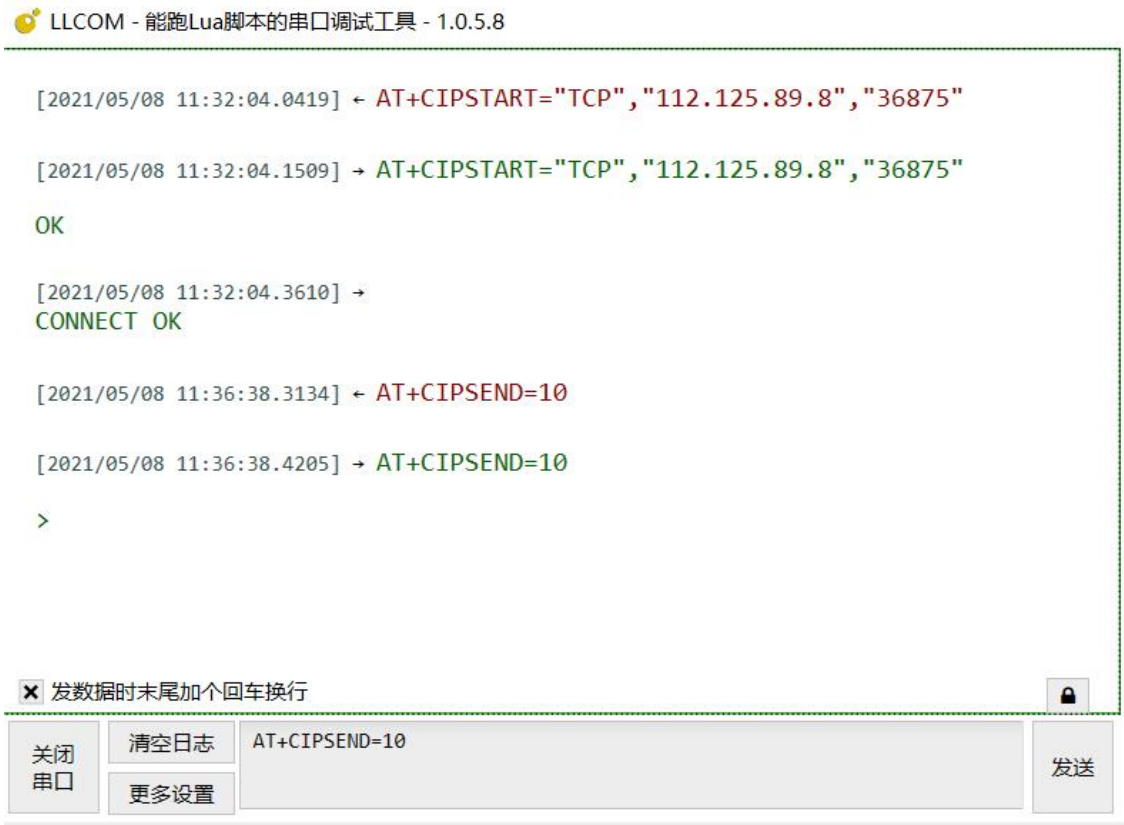
AT+CIPSEND 可以发送定长或不定长数据，收到>以后就可以发送数据（最大 1460 字节）。

如果是定长数据，输入数据达到长度会向服务器自动发送数据。

如果是不定长数据，输入数据后发送十六进制 1A 结束发送，发送完 1A 以后设备会向服务器发送 1A 之前的所有数据。

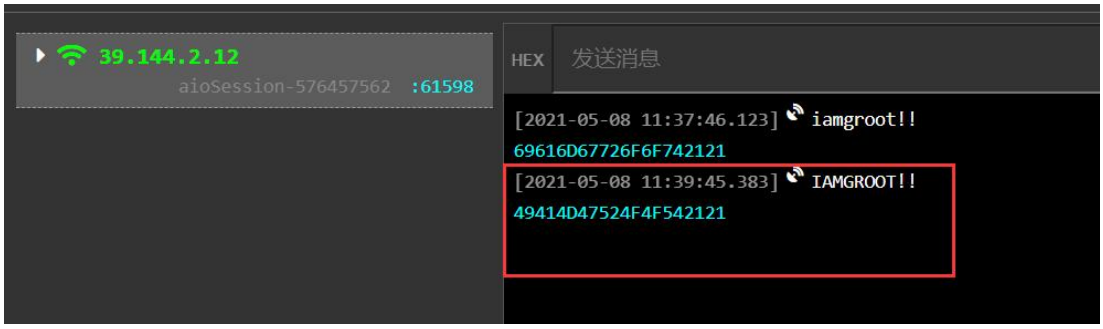
定长数据，假设这里发送 10 个字节数据

AT+CIPSEND=10



输入、发送数据

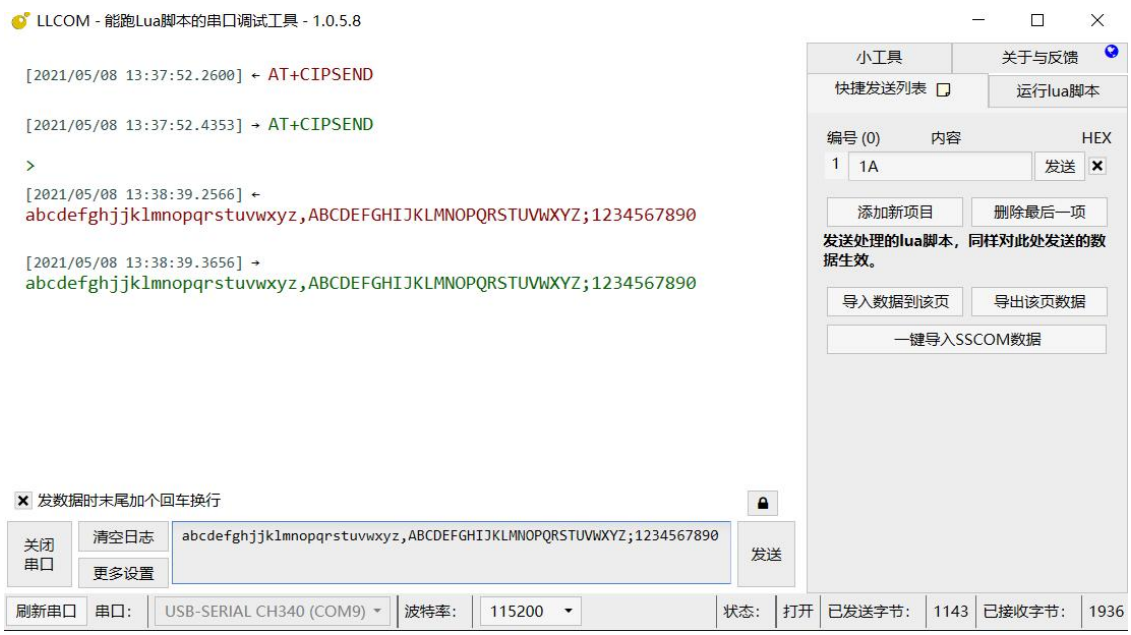
可以看到调试网页已经接收到了数据



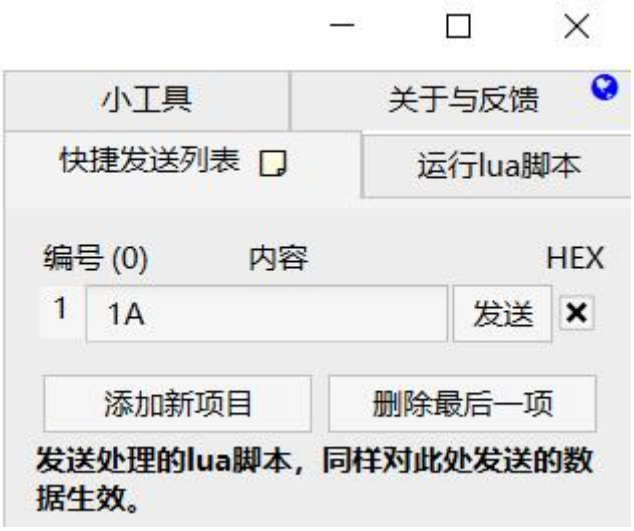
不定长数据

直接发送 AT+CIPSEND，会返回一个>符号，之后可以输入数据，数据输入完毕后，发送十六进制的 1A，结束发送，模块会将输入的数据发送到服务器

发送 AT+CIPSEND 后，可以看到返回了一个>，后续输入的内容没有返回值

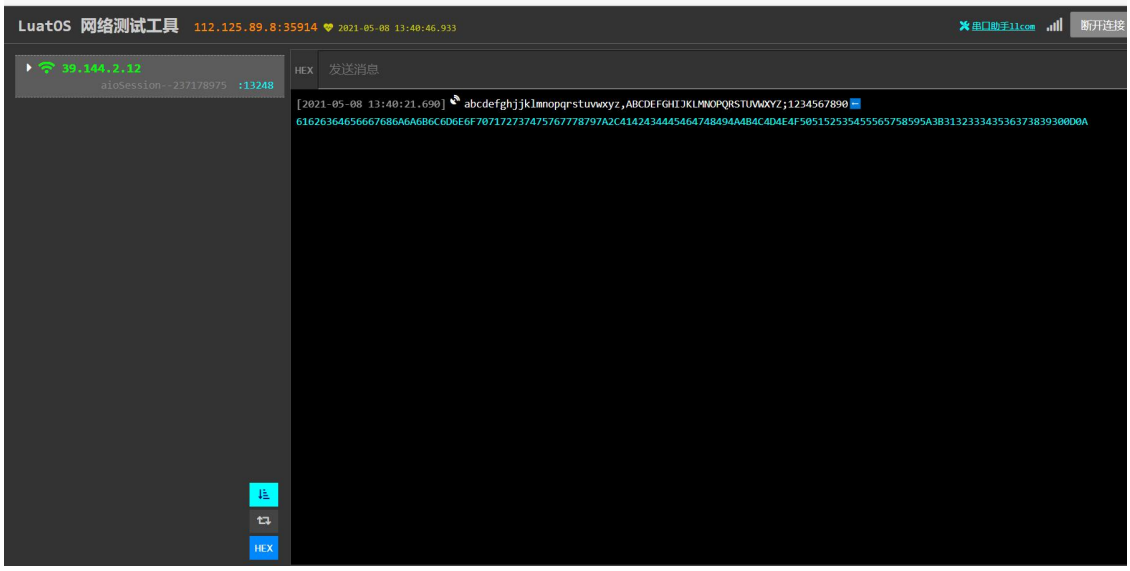


此时再发送十六进制的 1A 来结束发送



返回了 SEND OK

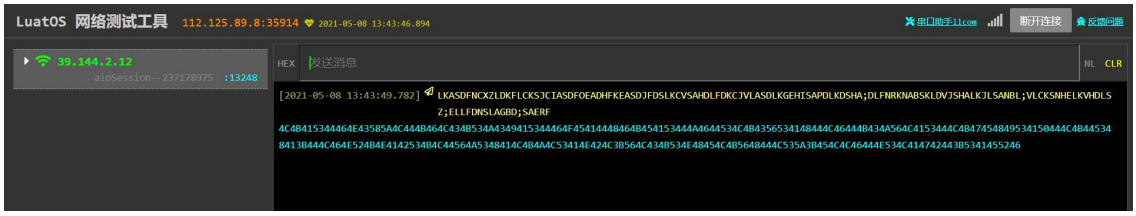
再看调试服务器网页，已经收到了刚才发送的数据



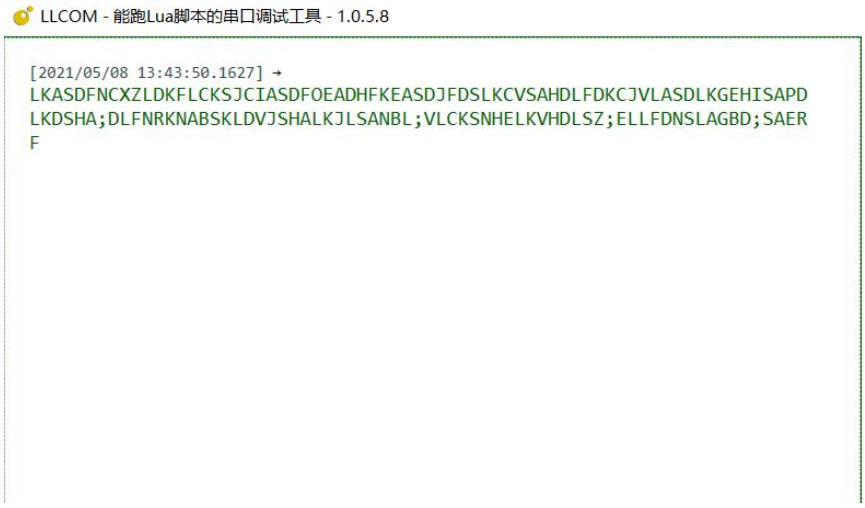
接收数据

当连接服务器成功以后，模块默认配置是服务器下发的数据会通过串口自动上报，直接就可以接收数据。

从调试网页发送数据到模块



可以看到模块收到了数据



其他说明

最新的 AT 固件支持开机直接发送 AT+CIPSTART 联网，不过不推荐使用，推荐根据状态机一步一步实现

TCP 使用 SSL 示例

开机

通过拉低 powerkey2 秒进行开机，开机以后通过串口循环发送 AT 直到收到 OK，如果 90 秒没有收到 OK 请拉低 RESET_IN_N 引脚 150ms 以上。或使用其他方法见模块硬件设计手册张开关机章节。

查询卡状态

AT+CPIN?查询卡状态，直到收到+CPIN: READY，如果 10s 内没有收到建议重启模块

查询网络注册情况

AT+CGATT?查询是否注册网络收到+CGATT: 1 值是 1 即为注册成功，正常情况下注册时间不会超过两分钟，如果超过两分钟没有注册可以进入飞行模式五秒后退出再查询，或者直接重启模块。

配置连接

AT+CIPMUX=0 设置为单链接模式

AT+CIPQSEND=1 设置为快发

注意：这两条一定要在启用网络之前配置，不然会失败。

激活网络

AT+CSTT 配置网络，非私有 APN 以外 Cat1 的固件支持根据卡自动配置 APN，直接输入 AT+CSTT 即可，模块会按照自动获取的 APN 设置 CSTT 的 APN。

AT+CIICR 激活网络，在 IP START 的状态使用 AT+CIICR 激活网络，激活以后通过 **AT+CIFSR** 查询是否获取 IP，如果成功就可以开始配置 TCP 连接了，如果不成功使用 AT+CIPSHUT 关闭移动网络，从 AT+CSTT 重新进行。

配置连接

AT+CIPSSL=1 开启 SSL 功能开关为开

AT+FSCREATE="server.crt" 创建 CA 证书文件

AT+FSCREATE="client.crt"创建客户端证书文件

AT+FSCREATE="client.key"创建客户端密钥文件

AT+FSWRITE="server.crt",0,2080,15 文件长度 2080 只是举例，要根据实际填写。下同。

AT+FSWRITE="client.crt", 0,128,10 输入客户端证书文件

AT+FSWRITE="client.key",0,188,10 输入客户端密钥文件

AT+SSLCFG="cacert",0, "server.crt"设置服务器 CA 证书 SSL 上下文 id，在单链接的情况下缺省为 0

AT+SSLCFG="clientcert",0, "client.crt"设置客户端证书

AT+SSLCFG="clientkey", 0,"client.key"设置客户端 KEY

AT+SSLCFG="seclevel",0,2 设置安全等级

AT+SSLCFG="ciphersuite",0,0X0035 设置加密套件

AT+SSLCFG="clientrandom",0,101B12C3141516171F19202122232425262728293031323334353637 设置随机数

连接服务器

AT+CIPSTART=TCP,tcplab.openluat.com,57513 连接服务器，请使用自己服务器测试

AT+CIPSEND=10 发送数据（确定长度）

AT+CIPSHUT 关闭连接

AT+CIPSSL=0 关闭 SSL

指令报错的异常处理及示例说明

^MODE: 17,17

+E_UTRAN Service

+CGEV: ME PDN ACT 1

+NITZ: 22/10/26,07:37:55+32,0

//以上是模块开机后主动上报的数据

[16:25:26.408]发→◇AT

[16:25:26.410]收←◆AT

OK

[16:25:29.710]发→◇ATI

□

[16:25:29.715]收←◆ATI

AirM2M_780E_V1021_LTE_AT

OK

[16:25:30.205]发→◇AT+CPIN?

□

[16:25:31.205]收←◆AT+CPIN?

+CME ERROR: 10 //未检测到 SIM 卡，重新安装 sim 卡后，需要给模块复位重启才能识别到卡

[16:25:33.202]发→◇AT+CPIN?

□

[16:25:33.205]收←◆AT+CPIN?

+CPIN: READY

OK

[16:25:35.646]发→◇AT+CSQ

□

[16:25:35.650]收←◆AT+CSQ

+CSQ: 23,0

OK

[16:25:38.032]发→◇AT+CGATT?

□

[16:25:38.034]收←◆AT+CGATT?

+CGATT: 1

OK

[16:25:39.490]发→◇AT+CEREG?

□

[16:25:39.492]收←◆AT+CEREG?

+CEREG: 0,1

OK

//以上为主动查询模块的联网情况， 模块已经附着网络成功

[16:25:55.169]发→◇AT+CIPMODE=0

□

[16:25:55.176]收←◆AT+CIPMODE=0

OK //设置 TCPIP 应用模式为非透明传输模式

[16:25:55.908]发→◇AT+CIPMUX=0

□

[16:25:55.911]收←◆AT+CIPMUX=0

OK //设置为单链接模式(默认是单路链接)

[16:25:56.610]发→◇AT+CIPQSEND=1

□

[16:25:56.612]收←◆AT+CIPQSEND=1

OK //设置非透传数据为快发送模式

[16:26:00.198]发→◇AT+CSTT

□

[16:26:00.201]收←◆AT+CSTT

OK //启动任务并设置接入点 APN、用户名、密码(cat1 模块内部有写 APN，可以不需要设置，默认为空，专网卡、定向卡除外)

[16:26:04.469]发→◇AT+CSTT

□

[16:26:04.472]收←◆AT+CSTT

+CME ERROR: 3 //提示操作错误，相同的指令连续输入 2 次，会报这个错误，可以通过发 AT+CIPSHUT 关闭移动场景后，在输入此命令

[16:26:04.479]发→◇AT+CIICRR

□

[16:26:04.621]收←◆AT+CIICRR

ERROR //激活移动场景 （出现这个指令错误，一般都是模块内部没有写这条指令或者指令输入错误）

[17:27:53.099]发→◇AT+CIICR

□

[17:27:53.100]收←◆AT+CIICR

+PDP: DEACT

+CME ERROR: 65535 //查询本地 IP 地址 （在模块附着网络失败后会出现此错误，可以通过 AT+CGATT? 来查询，可以走断网异常流程来处理

[16:26:07.030]发→◇AT+CIFSR

□

[16:26:07.034]收←◆AT+CIFSR

10.17.70.180

[16:26:08.509]发→◇AT+CIPSTART=1,“TCP”,”112.125.89.8“,37076

[16:26:08.516]收←◆AT+CIPSTART=1,“TCP”,”112.125.89.8“,37076

+CME ERROR: 3 //在单路链接的情况下，使用多路链接的指令会出现此错误

[16:26:30.509]发→◇AT+CIPSTART=“TCP”,”112.125.89.8“,37076

□

[16:26:30.516]收←◆AT+CIPSTART=“TCP”,”112.125.89.8“,37076

OK

STATE: IP STATUS

CONNECT FAIL //在联网正常的情况下，一直无法链接服务器，可以检查一下指令的符号是否有输入错误，使用中文的符号会导致异常

[16:41:31.516]发→◇AT+CIPSTART=“TCP”,”112.125.89.8“,37076

□

[16:26:31.980]收←◆AT+CIPSTART=“TCP”,”112.125.89.8“,37076

OK

[18:00:01.142]收←◆

CONNECT OK //链接成功

[16:26:38.667]发→◇AT+CIPSEND

□

[16:26:38.675]收←◆AT+CIPSEND

>

//出现此符号后才能发送数据，没有出现时发送数据会异常

[16:26:43.648]发→◇666888

□

[16:26:43.651]收←◆666888

[16:26:46.292]发→◇□ //HEX 格式的大写 1A

[16:26:46.306]收←◆

DATA ACCEPT:8

[16:26:48.597]发→◇AT+CIPSTATUS

□

[16:26:48.604]收←◆AT+CIPSTATUS

OK

STATE: CONNECT OK //查询链路状态

[18:07:05.517]发→◇AT+CIPSHUT

□

[18:07:05.521]收←◆AT+CIPSHUT

[18:07:07.521]收←◆

SHUT OK //关闭移动场景

常见问题

(1) **连接服务器失败**

1. 服务器必须是公网地址
2. 使用 PC 上的 TCP UDP 测试工具客户端、或者 mqtt.fx，连接服务器确认一下是否可以连接成功，排除 服务器故障
3. 检查下模块信号、网络注册、网络附着、PDP 激活状态
4. 检查下 SIM 卡是否欠费【4G 模块有一种欠费表现：无法注册 4G 网络，可以注册 2G 网络】

(2)**最多同时支持多少个连接**

AT 版本最多 8 个连接

(3) 关于"TCP 单/多链接时，刚开始发送数据成功，服务器能收到,**间隔十几分钟，模块端 AT+CIPSTATUS 查询链接存在，AT+CIPSEND 数据发送成功，但服务器却没收到数据**"问题

用心跳包(HEARTCONFIG)来解决

详细解释和 AT 示例，可参考链接
http://doc.openluat.com/shareArticle/9gI328ebaVFSr

(4)**780E 模块 tcp 联网成功 每次开机都需要重新下发 AT 指令连接吗？**

是的，模块不保存已输入 AT 指令

(5)**模块上报 PDP: DEACT ， 这个要怎么处理**

4G 模块会产生一个+PDP: DEACT 的 URC 提示，出现此提示说明已经激活的 PDP 承载被网络端去激活了，此时仍然需要向 4G 模块发送 AT+CIPSHUT 命令主动关闭 PDP，然后再重新激活，再重新连接 TCP 服务器。

(6)**重试多次 PDP 一直激活失败**

如果重试多次，PDP 一直激活失败，则尝试使用如下手段恢复：

1、使用 RESET 引脚复位模块

2、极端情况下，直接给模块断电，再上电，POWER KEY 引脚拉低开机

(7) **如何发送心跳包**

cat1 模块如何发送心跳包 cat1 模块如何发送心跳包

(8) **由数据透传状态转向命令状态，发+++切换不了**

发送时不要勾选回车换行

(9) **TCP 调试网址**

https://netlab.luatos.com/

(10) **当我连续发送几百个字节的数据的时候，在里面会夹杂 0x0a0x0d 这种，如何避免这个呢**

通过 AT+CIPRXF 设置接收的数据末尾是否自动添加回车换行（网络连接前设置）

12.12 设置接收的数据末尾是否自动添加回车换行：AT+CIPRXF

语法规则：

命令类型	语法	返回
设置命令	AT+CIPRXF=<n>	OK
查询命令	AT+CIPRXF?	+CIPRXF: <n> OK
测试命令	AT+CIPRXF=?	+CIPRXF: (<n>的取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<n>	模式	0	每笔收到的数据末尾自动添加一个回车换行（\r\n）
		1	每笔收到的数据末尾不添加回车换行（\r\n）

(11) **支持什么加密证书**

支持证书配置，支持单向认证和双向认证

支持如下六种加密套件：

0X0035 TLS_RSA_WITH_AES_256_CBC_SHA

0X002F TLS_RSA_WITH_AES_128_CBC_SHA

0X0005 TLS_RSA_WITH_RC4_128_SHA

0X0004 TLS_RSA_WITH_RC4_128_MD5

0X000A TLS_RSA_WITH_3DES_EDE_CBC_SHA

0X003D TLS_RSA_WITH_AES_256_CBC_SHA256

- (12)**数据发送接收缓存问题**
- 使用 CIPSTART 建立的 socket 连接，接收数据没有缓存机制，收到数据后，立即通过 AT 口输出，每包最多 1460 字节；例如服务器下发一个 70KB 的文件，模块会连续收到多个包，通过 AT 口连续多次输出，发送每包最多也为 1460 字节

9.5.2 HTTP(S)应用指南

应用概述

4G 模块支持 HTTP 和 HTTPS 协议， HTTP 应用的基本流程如下：

1、激活 PDP（参考：http://oldask.openluat.com/article/937）

2、初始化 HTTP 服务

3、设置 HTTP 会话参数

4、如果要支持 SSL，配置 SSL 参数

5、如果使用 POST 命令，输入 POST 数据

6、发起 HTTP 请求

7、收到 HTTP 应答，读取应答数据

8、终止 HTTP 服务

第 1 步出现异常后：参考 http://oldask.openluat.com/article/937 进行异常处理

第 2 步到第 5 步，只要输入格式正确，基本不会出问题；如果出错，可以跳过，直接处理第 6 步的异常

第 6 步和第 7 步出现异常后：终止 HTTP 服务，有选择性的去激活 PDP；然后再有选择性的激活 PDP，从第 2 步开始重新执行

示例

HTTP GET 示例

先来看个正常情况下的 AT 命令序列，如果不理解 AT 命令含义，请自行参考 AT 手册

```
AT+CPIN?

+CPIN: READY    //查询 sim 卡是否正常

OK

AT+CGATT?

+CGATT: 1        //查询是否附着上数据网络，如果返回+CGATT: 0 表示未附着上

OK

AT+SAPBR=3,1,"CONTYPE","GPRS"

OK
```

```
AT+SAPBR=3,1,"APN",""     //设置 APN，此处""表示使用从网络端自动获取到的 APN

OK

AT+SAPBR=1,1              //发起激活 PDP 的请求

OK

AT+SAPBR=2,1              //注意：此命令仅仅查询 PDP 地址，可以不执行

+SAPBR: 1,1,"10.159.1.145"     //请求到的 PDP 地址

OK

AT+HTTPINIT

OK

AT+HTTPPARA="CID",1

OK

AT+HTTPPARA="URL","airtest.openluat.com"

OK

AT+HTTPACTION=0

OK

+HTTPACTION: 0,200,285

AT+HTTPREAD

+HTTPREAD: 285

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    Hello
</body>
</html>
OK
```

在这个过程中，注意事项如下：

1、发送 AT+HTTPACTION 命令后，收到 OK 仅仅表示 4G 模块开始处理这条命令，并不表示请求发送成功，收到了应答；只有收到+HTTPACTION: n,statusCode,len 才表示请求结束，statusCode 表示应答成功，其余都表示应答失败

2、HTTP 应答数据的缓冲区大小为 4KB，如果返回的 http body 数据超过了这个大小，则需要断点下载，可以使用 AT+HTTPPARA 命令，通过设置 BREAK 和 BREAKEND 参数来实现

TCP 被动断开示例

发送 AT+HTTPACTION 命令，和服务器建立了 http 连接后，如果连接异常被动断开，会输出+HTTPACTION: <Method>,<StatusCode>,<DataLen>提示

PDP 被动去激活示例

先来看下 PDP 被动去激活时的 AT 命令处理序列，如果不理解 AT 命令含义，请自行参考 AT 手册。

```
......            //此处省略了 PDP 激活的过程，参考基本流程章节

AT+HTTPINIT

OK

AT+HTTPPARA="CID",1

OK

AT+HTTPPARA="URL","www.baidu.com"

OK

AT+HTTPACTION=0

OK

+SAPBR 1: DEACT        //此处 PDP 被动去激活

+HTTPACTION: 0,601,0

AT+HTTPTERM

OK                     //此处无论返回 OK、ERROR 还是 CME ERROR，都直接跳过，不用做正确性判断

AT+SAPBR=0,1

+CME ERROR: 3          //此处无论返回 OK、ERROR 还是 CME ERROR，都直接跳过，不用做正确性判断
```

```
......                    //此处省略了 PDP 激活的过程，参考基本流程章节

AT+HTTPINIT

OK

AT+HTTPPARA="CID",1

OK

AT+HTTPPARA="URL","www.baidu.com"

OK

AT+HTTPACTION=0

OK

+HTTPACTION: 0,200,154293
```

参考基本流程章节流程图，右下角的"收到+SAPBR 1: DEACT 错误提示，表示 PDP 被动去激活"为异常处理的触发点

HTTPS 示例

带 SSL 证书双向验证功能的 HTTPS 过程：

```
 AT+FSCREATE="ca.crt" 创建服务器端 CA 证书文件

 OK

AT+FSCREATE="client.crt" 创建客户端证书文件

 OK

 AT+FSCREATE="client.key" 创建客户端密钥文件

 OK

 AT+FSWRITE="ca.crt",0,2080,15 文件长度 2080 字节只是举例，要根据实际填写。下同。

> 这里输入 CA 证书文件
```

```
OK

AT+FSWRITE="client.crt",0,128,10

> 这里输入客户端证书文件

OK

AT+FSWRITE="client.key",0,188,10

> 这里输入客户端密钥文件

OK

AT+SAPBR=3,1,"CONTYPE","GPRS"

OK

AT+SAPBR=3,1,"APN","" 设置 PDP 承载之 APN 参数 模块注册网络后会从网络自动获取<apn>并激活一个 PDP 上下文，用于 RNDIS 上网使用（此<apn>可以通 过 AT+CGDCONT?来查询），所以输入 AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按 照自动获取的<apn>来设置 APN

OK

AT+SAPBR=1,1

OK

AT+SAPBR=2,1

+SAPBR: 1,1,010.169.179.213 OK

AT+SSLCFG="cacert",153,"ca.crt" 设置服务器 CA 证书 SSL 上下文 id，在 TCP 单链接的情况下缺省为 0； 在 HTTPS 链接下为 153，下同

OK

AT+SSLCFG="clientcert",153,"client.crt" 设置客户端证书

OK

AT+SSLCFG="clientkey",153,"client.key" 设置客户端 KEY
```

```
OK

AT+SSLCFG="seclevel",153,2 设置安全等级

OK

AT+SSLCFG="ciphersuite",153,0X0035 设置加密套件

OK

AT+SSLCFG="clientrandom",153,01B12C31
41516171F19202122232425262728293031 323334353637D 设置随机数

OK

AT+HTTPINIT HTTP 协议栈初始化

OK

AT+HTTPPARA="CID",1 设置 HTTP 会话参数：CID

OK

AT+HTTPPARA="URL","https://**.***.***" 设置 HTTP 会话参数：URL 请写具体的网
址，而不要照抄

OK

AT+HTTPACTION=0 GET 开始

OK

+HTTPACTION:0,200,1348 +HTTPACTION:0,200,1348 +HTTPACTION:0,200,1348 … …
出现这些 URC 上报表明 GET 数据成功，等待 READ

AT+HTTPREAD 读取从 HTTP 服务器 GET 的数据

+HTTPREAD:1592 …… OK …..表示 HTTP 数据

AT+HTTPTERM 结束 HTTP 服务

OK
```

…… OK ……表示 HTTP 数据

AT+HTTPTERM 结束 HTTP 服务

OK

常见问题

1、HTTP 支持多连接吗

目前 HTTP 仅支持单连接，不支持多连接

2、HTTPS 如何使用

本文主要描述了基本流程和异常处理，对于 HTTPS 使用方法没有做过多描述，这一部分，请自行参考 AT 手册 HTTP 章节下《使用方法举例》中的"带 SSL 证书验证功能的 HTTPS 流程"使用方法；支持的 SSL 参数，请自行参考 AT+SSLCFG 命令说明

如果 SSL 的参数配置不变，则每次开机运行过程中，仅设置一次即可

3、重试多次 PDP，HTTP 应用一直连接失败

如果重试多次 PDP 激活，PDP 一直激活失败，或者 HTTP 一直请求应答失败，则尝试使用如下手段恢复：

1、使用 RESET 引脚复位模块

2、极端情况下，直接给模块断电，再上电，POWER KEY 引脚拉低开机

4、HTTP 如何下载大文件（断点续传）

在实际的应用场景中，可能需要下载一个非常大的文件，例如几百 K 字节、几 M 字节，但是 4G 模块中 HTTP 可用的内存缓冲区为 300KB 左右（780E 模块 4KB 左右），当文件大小超过这个缓冲区时，就要使用断点续传功能来分段下载处理了。下面以“下载一个 11975260 字节的文件”为例，来说明如何使用断点续传功能（注意：本示例仅仅演示了正常流程的 HTTP AT 命令，完整流程以及异常处理流程请参考本文应用流程部分）

```
AT+HTTPINIT

OK

AT+HTTPPARA="URL","http://openluat-erp.oss-cn-hangzhou.aliyuncs.com/erp_site_file/product_file/sw_file_20200108162920_Luat_V0028_ASR1802.zip"

OK

AT+HTTPACTION=2

OK

+HTTPACTION: 2,200,0

AT+HTTPHEAD

+HTTPHEAD: 454

server: aliyunoss
```

```
date: thu, 16 jan 2020 06:50:58 gmt

content-type: application/zip

content-length: 11975260                                    //此处的 11975260 表示文件总大小

connection: keep-alive

x-oss-request-id: 5e2007d108f4be32353a92ae

accept-ranges: bytes

etag: "a5b9cc75c0f26413bbaf00a0fa952bb2"

last-modified: wed, 08 jan 2020 08:29:29 gmt

x-oss-object-type: normal

x-oss-hash-crc64ecma: 16925484473913319613

x-oss-storage-class: standard

content-md5: pbnmdcdyzbo7rwcg+pursg==

x-oss-server-time: 111

OK

//如下指令，表示下载文件的第一个 3KB 数据

AT+HTTPPARA="BREAK",0

OK

AT+HTTPPARA="BREAKEND",3071

OK

AT+HTTPACTION=0

OK

+HTTPACTION: 0,206,3072
```

```
AT+HTTPREAD

+HTTPREAD: 3072

......                              //此处输出 3072 字节数据

OK

//表示下载文件的第一个 3KB 数据完成

//如下指令，表示下载文件的第二个 3KB 数据

AT+HTTPPARA="BREAK",3072

OK

AT+HTTPPARA="BREAKEND",6143

OK

AT+HTTPACTION=0

OK

+HTTPACTION: 0,206,3072

AT+HTTPREAD

+HTTPREAD: 3072

......                              //此处输出 3072 字节数据

OK

//表示下载文件的第二个 3KB 数据完成


//此处参考上文指令，一直循环读取文件的下一个 300KB 数据，直到读取结束


......
```

```
AT+HTTPTERM

OK
```

5、HTTP 下载的大文件如何可靠的发送给 MCU（AT 流控）

在上面第四个问题指令里加入下面一段 AT 指令中，如下一段 AT 指令

```
AT+HTTPREAD

+HTTPREAD: 3072

......                                  //此处输出 3072 字节数据

OK
```

...处的 3072 字节数据，是模块通过 UART AT 口发送给 MCU，在实际传输过程中，由于串口芯片驱动、MCU 端的处理能力、波特率的选择都存在不确定性，可能会导致 MCU 端接收到的数据，实际上没有 3072 字节，这就要求 UART AT 口打开流控功能。模块支持硬流控和软流控两种：

1. 硬流控，参考如下步骤操作：

（1） 模块和 的 UART 口，CTS、RTS 要交叉相连

（2） MCU 端固件要支持并且打开硬流控功能

（3） MCU 端要发送 AT+IFC=2,2 命令到模块端，打开模块端的硬流控功能

2. 软流控，参考如下步骤操作：

（1） MCU 端固件要支持并且打开软流控功能

（2） MCU 端要发送 AT+IFC=1,1 命令到模块端，打开模块端的硬流控功能

6、为什么我只发了 10 字节消息，100 次却消耗了那么多流量？

因为还有 HTTP 自带的请求头。如何统计流量

7、为什么频繁请求会失败?

支持的 http 连接总数有限数量为 tcp 连接数量 8 个，建议一个 http 连接返回请求结果之后，再去请求下一个连接；不要使用循环定时器方式不断的发起新的 http 请求。

8、如何 POST 文件?

主要是使用 AT+HTTPDATA 命令录入文件数据后，发送 AT+HTTPACTION=1 利用 post 上传

9、数据发送接收缓存问题

AT 版本：有缓存机制，内存中有一个的缓冲区（780E 系列模块标准 AT 为 4KB，last 版本为 128k），发送和接受使用的是同一块缓冲区，发送和收到数据后，插入此缓冲区，然后通过 AT 口输出 urc，提示收到的数据长度；缓冲区满之后，再收到新数据，会丢弃新收到的数据，并通过 AT 口输出 urc 提示出错；需要读取数据时，发送 AT+HTTPREAD 命令读取，可分段读取，也可全部读取

注意：缓冲区位于内存中，断电或者重启后，缓存表中的数据会被清空；虽然缓冲区可以缓存很多数据，但是建议收到数据时，通过 AT+HTTPREAD 及时读取出来，以防缓冲区满出错

HTTP 下载大文件内存不足问题

可以使用上面的断点续传

或者新的 AT 指令 AT+HTTPEXACTION / AT+HTTPEXGET

示例:

```
AT+CPIN?

+CPIN: READY    //查询 sim 卡是否正常

OK

AT+CGATT?

+CGATT: 1        //查询是否附着上数据网络，如果返回+CGATT: 0 表示未附着上

OK

AT+SAPBR=3,1,"CONTYPE","GPRS"

OK

AT+SAPBR=3,1,"APN",""    //设置 APN，此处""表示使用从网络端自动获取到的 APN

OK

AT+SAPBR=1,1             //发起激活 PDP 的请求

OK

AT+SAPBR=2,1             //注意：此命令仅仅查询 PDP 地址，可以不执行

+SAPBR: 1,1,"10.43.103.155"    //请求到的 PDP 地址

AT+HTTPINIT

OK

AT+HTTPPARA="CID",1

OK
```

```
AT+HTTPPARA="URL","airtest.openluat.com"

OK

AT+HTTPEXACTION=0

OK

+HTTPEXGET  //收到这个返回以后 发送下面这条指令

AT+HTTPEXGET  //重复发送这条指令直到文件下载完

+HTTPEXGET: 3880  //本次输出 3880 字节数据

(此处省略了打印信息)

OK

+HTTPEXGET

AT+HTTPEXGET

+HTTPEXGET:

(此处省略了打印信息)

OK

+HTTPEXACTION: 0,200  //下载完毕结束会话
```

9.5.3 MQTT 应用指南

应用概述

```
    4G 模块支持 MQTT 和 MQTT SSl 协议，  MQTT 应用的基本流程如下：

    1、如果要支持 SSL，配置 SSL 参数

    2、通过 TCP 连接到 MQTT 服务器

    3、发送 MQTT CONNECT 到服务器，打开会话连接
```

4、订阅或者发布消息

示例

发布消息示例

先来看个正常情况下的 AT 命令序列，如果不理解 AT 命令含义，请自行参考 AT 手册

```
AT+CPIN?

+CPIN: READY    //查询 sim 卡是否正常

OK

AT+CGATT?

+CGATT: 1        //查询是否附着上数据网络，如果返回+CGATT: 0 表示未附着上

OK

AT+MCONFIG="866289037465624","user","password"

OK

AT+MIPSTART="lbsmqtt.airm2m.com","1884"

OK

CONNECT OK

AT+MCONNECT=1,60

OK

CONNACK OK

AT+MSUB="mqtt/sub_topic",0

OK

SUBACK

AT+MPUB="mqtt/pub_topic",0,0,"publish 0 payload"

OK
```

```
AT+MPUB="mqtt/pub_topic",1,0,"publish 1 payload"

OK

PUBACK

AT+MPUB="mqtt/pub_topic",2,0,"publish 2 payload"

OK

PUBREC

PUBCOMP
```

在这个过程中，注意事项如下：

1、发送 MIPSTART 命令后，收到 OK 仅仅表示 4G 模块开始处理这条命令，并不表示 TCP 已经连接成功；只有收到 CONNECT OK 才是成功的应答（其余错误应答，参考 AT 手册）

2、发送 MCONNECT 命令后，收到 OK 仅仅表示 4G 模块开始处理这条命令，并不表示 MQTT 已经连接成功；只有收到 CONNACK OK 才是成功的应答（其余错误应答，参考 AT 手册）

3、发送 MSUB 命令后，收到 OK 仅仅表示 4G 模块开始处理这条命令，并不表示 MQTT 已经订阅成功；只有收到 SUBACK 才是成功的应答（其余错误应答，参考 AT 手册）

4、发送 MPUB 命令后，如果 QOS 不是 0，则收到 OK 仅仅表示 4G 模块开始处理这条命令，并不表示 MQTT 已经发布成功；只有收到 PUBACK（QOS 为 1 时）或者 PUBCOMP（QOS 为 2 时）才是成功的应答（其余错误应答，参考 AT 手册）

接收消息示例

先来看下接收到订阅消息时的 AT 命令处理序列，如果不理解 AT 命令含义，请自行参考 AT 手册

```
AT+MCONFIG="866289037465624","user","password"

OK

AT+MIPSTART="lbsmqtt.airm2m.com","1884"

OK

CONNECT OK

AT+MCONNECT=1,60
```

```
OK

CONNACK OK

AT+MSUB="mqtt/sub_topic",0

OK

SUBACK

AT+MPUB="mqtt/sub_topic",0,0,"123456"

//在 mqtt/sub_topic 上收到订阅消息，payload 为 123456

//默认是直接输出收到的订阅消息，不做缓存

+MSUB: "mqtt/sub_topic",6 byte,123456

//设置订阅消息的处理模式，设置为缓存模式

//接下来收到订阅消息时，会提示订阅消息的缓存位置

AT+MQTTMSGSET=1

OK

//在 mqtt/sub_topic 上收到一条订阅消息，缓存到位置 0

+MSUB: 0

//主动读取缓存的所有订阅消息

AT+MQTTMSGGET

+MSUB:mqtt/sub_topic,6 byte,123456

OK

//在 mqtt/sub_topic 上收到一条订阅消息，缓存到位置 0

+MSUB: 0

//在 mqtt/sub_topic 上收到一条订阅消息，缓存到位置 1

+MSUB: 1
```

```
//主动读取缓存的所有订阅消息

AT+MQTTMSGGET

+MSUB:mqtt/sub_topic,6 byte,123456

+MSUB:mqtt/sub_topic,6 byte,123456

OK
```

TCP 被动断开示例

先来看下连接被动断开时的 AT 命令处理序列，如果不理解 AT 命令含义，请自行参考 AT 手册

```
AT+MCONFIG="866289037465624","user","password"

OK

AT+MIPSTART="lbsmqtt.airm2m.com","1884"

OK

CONNECT OK

AT+MCONNECT=1,60

OK

CONNACK OK

CLOSED                //此处 TCP 连接被动断开

AT+MIPCLOSE

+CME ERROR: 767      //此处无论返回 OK、ERROR 还是 CME ERROR，都直接跳过，不用做正确性判断

AT+MCONFIG="866289037465624","user","password"

OK

AT+MIPSTART="lbsmqtt.airm2m.com","1884"

OK
```

```
CONNECT OK

AT+MCONNECT=1,60

OK

CONNACK OK
```

PDP 被动去激活示例

先来看下 PDP 被动去激活时的 AT 命令处理序列，如果不理解 AT 命令含义，请自行参考 AT 手册

```
AT+MCONFIG="866289037465624","user","password"

OK

AT+MIPSTART="lbsmqtt.airm2m.com","1884"

OK

CONNECT OK

AT+MCONNECT=1,60

OK

CONNACK OK

+CGEV: NW PDN DEACT 1       //此处 PDP 被动去激活

AT+MIPCLOSE

OK                    //此处无论返回 OK、ERROR 还是 CME ERROR，都直接跳过，不用做正确性判断

AT+MCONFIG="866289037465624","user","password"

OK

AT+MIPSTART="lbsmqtt.airm2m.com","1884"

OK

CONNECT OK
```

```
AT+MCONNECT=1,60

OK

CONNACK OK
```

常见问题

1、MQTT 支持多连接吗

目前 MQTT 仅支持单连接，不支持多连接

2、MQTT 的遗嘱如何使用

通过 AT+MCONFIG 命令可以设置遗嘱的 qos、retain 标志、topic、payload；在如下几种（包含但是又不仅限于如下情况）情况下，服务器会主动发布遗嘱消息到订阅的客户端：

1、模块和服务器通信异常（例如模块突然关机、模块进入了一个没有网络信号的环境等）超过 1.5 倍（一般是 1.5 倍，但不排除服务器可以修改这个时间）的 keep alive 时间（可以通过 AT+MCONNECT 设置 keep alive 时间）

2、模块主动执行 AT+MDISCONNECT 或者 AT+MIPCLOSE

曾经有用户碰到过如下问题：

模块设置了遗嘱 topic 为/will_topic，遗嘱 payload 为 offline，keep alive 时间为 5 分钟，模块关机后，等待 2 分钟，在等待的 2 分钟内，其他订阅了/will_topic 的客户端没有收到遗嘱消息（这是正常的，因为还没有到达 1.5 倍的 keep alive 时间），2 分钟后，模块开机 mqtt 重连成功后，其他客户端却收到了遗嘱消息，这是什么原因呢？这部分机制是服务器端设计的，和模块无关，个人认为：服务器端仍然在维持 2 分钟前的那个 mqtt 连接，现在模块开机，用同样的 client id 重连后，会断开之前维护的相同 client id 的 mqtt 连接，断开时，就发布了一次遗嘱消息。那这种问题，如何解决呢？模块可以在 AT+MCONNECT 返回 CONNACK OK 之后，通过 AT+MPUB 一个 topic 为/will_topic，payload 为 online 的消息，这样订阅了遗嘱 topic 的其他客户端就会收到这个 online 消息

3、MQTT SSL 如何使用

本文主要描述了基本流程和异常处理，对于 MQTT SSL 使用方法没有做过多描述，这一部分，请自行参考 AT 手册 MQTT 章节下《使用方法举例》中的"SSL 带证书验证流程"使用方法；支持的 SSL 参数，请自行参考 AT+SSLCFG 命令说明

如果 SSL 的参数配置不变，则每次开机运行过程中，仅设置一次即可

4、重试多次 PDP，MQTT 应用一直连接失败

如果重试多次 PDP 激活，PDP 一直激活失败，或者 MQTT 一直连接失败，则尝试使用如下手段恢复：

1、使用 RESET 引脚复位模块

2、极端情况下，直接给模块断电，再上电，POWER KEY 引脚拉低开机

5、MPUB 命令中，payload 包含"如何发送？

消息中内嵌的双引号请用\22 表达；控制字符 \r(0x0D)请用\0D 表达；控制字符\n（0x0A）请用\0A 表达 如果是 MCU 发消息，请用\\22，\\0D，\\0A 来 表达，即\需要转义成\\

6、连接服务器失败

1. 检查下模块信号、网络注册、网络附着、PDP 激活状态
2. 检查下 SIM 卡是否欠费【4G 模块有一种欠费表现：无法注册 4G 网络，可以注册 2G 网络】
3. 使用 mqtt.fx，连接服务器确认一下是否可以连接成功，排除服务器故障
4. 部分国外的开源项目提供免费的 MQTT 代理服务器，因为网络的原因，国内存在严重的延迟或者丢包现象，导致程序运行出现问题，此现象在 AT 开发时影响严重
5. 确认是不是域名解析失败导致，可以通过 AT+CDNSGIP=<domain name>确认一下域名是否能正常解析（**注意：该命令只有在执行完 at+cstt、at+ciicr、at+cifsr 后才能正常工作**），不能正常解析，可以通过 AT+CDNSCFG=ip1,ip2 设置域名解析服务器。
6. 在 MIPSTART /SSLMIPSTART 返 回 CONNECT OK 后 才 能 发 MCONNECT 命令，而且要立即发，否则会被服务器踢掉。

7、数据接收缓存问题

1. 不缓存：通过 AT+MQTTMSGSET=0 设置；收到订阅的 publish 报文后，立即通过 AT 口输出主题、payload 长度、payload 内容，每个报文中支持的 payload 内容最长 4100 字节。

2. 内存缓存：通过 AT+MQTTMSGSET=1 设置；内存中有一个缓存表，一条消息最多 4kb，最多支持缓存 4 条 publish 报文；收到订阅的 publish 报文后，插入缓冲表中的空闲位置，然后通过 AT 口输出存储位置；缓存表满之后，新收到的 publish 报文会覆盖最旧的 publish 报文。

注意：缓存表位于内存中，断电或者重启后，缓存表中的数据会被清空；建议收到数据时，通过 AT+MQTTMSGGET 及时读取出来，以防缓冲区满覆盖丢失数据

8、数据发送问题

AT+MPUB 最大 publish 2384 字节,AT+MPUBEX 长度取值范围 1-4100

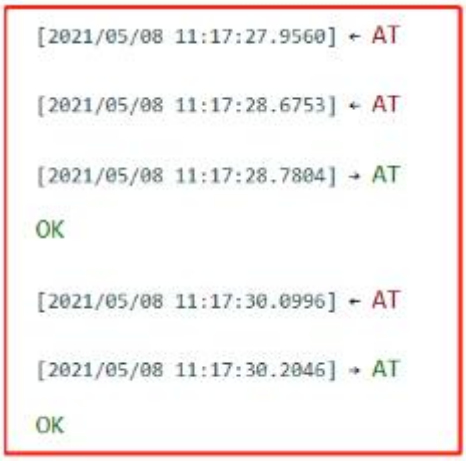
9.5.4 FTP 应用指南

连接网络

开机

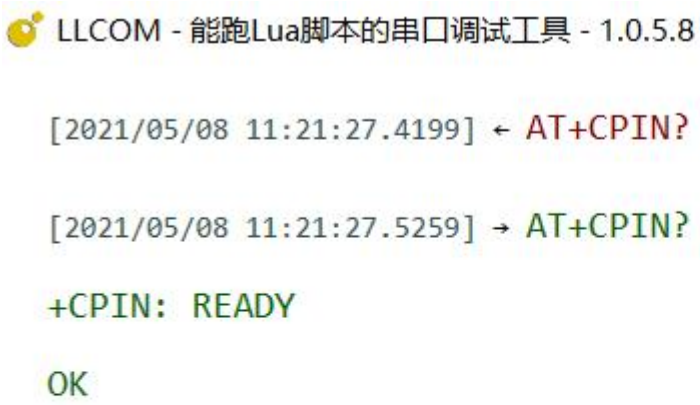
通过拉低 powerkey2 秒进行开机，开机以后通过串口循环发送 AT 直到收到 OK，如果 90 秒没有收到 OK 请拉低 RESET_IN_N 引脚 150ms 以上。

开机发送 AT 训练波特率



查询 SIM 卡状态

发送 **AT+CPIN?** 查询 SIM 卡状态，直到收到+CPIN: READY，如果 10s 内没有收到建议重启模块



查询网络注册情况

AT+CGATT? 查询是否注册网络收到+CGATT: 1

值是 1 即为注册成功，正常情况下注册时间不会超过两分钟，如果超过两分钟没有注册可以进入飞行模式五秒后退出再查询，或者直接重启模块。

值是 0 则没有注册成功

FTP 操作

2.1 FTP GET

[15:57:00.582]发→◇AT+SAPBR=3,1,“CONTYPE”,“GPRS”

[15:57:00.585]收←◆AT+SAPBR=3,1,“CONTYPE”,“GPRS”

OK

[15:57:01.517]发→◇AT+SAPBR=3,1,“APN”,“”

[15:57:01.520]收←◆AT+SAPBR=3,1,“APN”,“”

OK

[15:57:02.366]发→◇AT+SAPBR=1,1

[15:57:02.371]收←◆AT+SAPBR=1,1

OK

[15:57:03.302]发→◇AT+SAPBR=2,1

[15:57:03.304]收←◆AT+SAPBR=2,1

+SAPBR: 1,1,“10.162.173.20”

OK

//以上为 PDP 激活过程

[15:57:06.326]发→◇AT+FTPCID=1

[15:57:06.331]收←◆AT+FTPCID=1

OK

[15:57:24.862]发→◇AT+FTPSERV="XX.X.87.100"//设置服务器 ip 地址，也可以使用域名，测试以实际服务器为准

[15:57:24.868]收←◆AT+FTPSERV="XX.X.87.100"

OK

[15:57:26.798]发→◇AT+FTPUN="user"

[15:57:26.803]收←◆AT+FTPUN="user"

OK

[15:57:28.950]发→◇AT+FTPPW="123456"

[15:57:28.955]收←◆AT+FTPPW="123456"

OK

//以上为 FTP 服务器基本参数设置

[15:57:29.450]发→◇AT+FTPGETPATH="/"

[15:57:29.457]收←◆AT+FTPGETPATH="/"//设置下载文件路径

OK

[15:57:31.062]发→◇AT+FTPGETNAME="1223.txt"//设置服务器端目标文件的名称

[15:57:31.068]收←◆AT+FTPGETNAME="1223.txt"

OK

[15:57:39.557]发→◇AT+FTPGET=1 //开始下载会话

[15:57:39.563]收←◆AT+FTPGET=1

OK

[15:57:40.516]收←◆

+FTPGET: 1,1 //必须等到这条上报，才能发下一条指令

[15:57:43.550]发→◇AT+FTPGET=2,1460//一次读入最多 1460 字节

[15:57:43.556]收←◆AT+FTPGET=2,1460

+FTPGET: 2,1460

When doctors think all of their patients are heterosexual, those who identify .....(此处省略若干数据)..... LGBTQ patients sho//这是读入的 1460 字节数据

OK

[15:57:48.286]发→◇AT+FTPGET=2,1460//如果目标文件大于 1460 字节，需要在第一次读完数据后，再次发送这条指令，读取下一部分数据，直至结束

[15:57:48.294]收←◆AT+FTPGET=2,1460

+FTPGET: 2,214

uld get the same quality of care other patients get. Increasing LGBTQ student enrollment in medical schools and training in LGBTQ health issues can help reach those goals, advocates say.

Research shows that patie//文件较小，本次读入 214 字节，已全部读完

OK

[15:57:59.871]收←◆

+FTPGET: 1,0 //提示会话结束

[15:58:35.726]发→◇AT+FTPQUIT

[15:58:35.733]收←◆AT+FTPQUIT

+CME ERROR: 3

//已上报+FTPGET: 1,0 表示会话已结束，所以，AT+FTPQUIT退出当前FTP会话 返回error3，属于正常。也 可 以 在 提 示+FTPGET: 1,0 前，主动用 AT+FTPQUIT 退出当前 FTP 会话 。

2.2 FTP PUT(举例：给同一文件 PUT 2 次数据，每次 1360 字节（1358 字节+\r\n)

.....此处省略基本查询和 PDP 激活过程.....

[18:16:00.412]发→◇AT+FTPCID=1

[18:16:00.415]收←◆AT+FTPCID=1

OK

[18:16:03.612]发→◇AT+FTPSERV="XX.X.87.100"

[18:16:03.615]收←◆AT+FTPSERV="XX.X.87.100"

OK

[18:16:04.605]发→◇AT+FTPUN="user"

[18:16:04.613]收←◆AT+FTPUN="user"

OK

[18:16:05.541]发→◇AT+FTPPW="123456"

[18:16:05.546]收←◆AT+FTPPW="123456"

OK

//以上为 FTP 服务器基本参数设置

[18:16:26.604]发→◇AT+FTPPUTNAME="HZ.txt"//设置上传文件的名称

[18:16:26.609]收←◆AT+FTPPUTNAME="HZ.txt"

OK

[18:16:29.198]发→◇AT+FTPPUTPATH="/11/"//设置上传文件的路径

[18:16:29.204]收←◆AT+FTPPUTPATH="/11/"

OK

[18:16:31.229]发→◇AT+FTPPUT=1

[18:16:31.235]收←◆AT+FTPPUT=1

OK

[18:16:31.877]收←◆

+FTPPUT: 1,1,1360//必须等到这个上报

[18:16:34.117]发→◇AT+FTPPUT=2,1360//单次上传的最大长度为 1360 字节

[18:16:34.122]收←◆AT+FTPPUT=2,1360

+FTPPUT: 2,1360

[18:16:37.017]发
→◇1111111111111111111111111111111111111111111111111111111111111111.....（此处省略若干字
节）.....1111111111111111111111111111111111111111111111111111111111111166

[18:16:37.032]收←◆

OK

+FTPPUT: 1,1,1360//第一次成功 PUT 1360 字节数据

[18:16:46.669]发→◇AT+FTPPUT=2,1360

[18:16:46.676]收←◆AT+FTPPUT=2,1360

+FTPPUT: 2,1360

[18:16:49.678]发
→◇1111111111111111111111111111111111111111111111111111111111111111.....（此处省略若干字节）.....1111111111111188

[18:16:49.695]收←◆

OK

+FTPPUT: 1,1,1360//给同一文件，第二次成功 PUT 1360 字节数据

[18:17:08.590]发→◇AT+FTPQUIT

[18:17:08.593]收←◆AT+FTPQUIT//主动结束 FTP 会话

OK

[18:17:08.639]收←◆

+FTPPUT: 1,86

2.3FTP 下载 断点续传流程

2.3.1 确认服务器支持断点续传

2.3.2 流程如下

.....此处已省略模块开机基本查询和模块 PDP 激活过程.....

[18:56:53.795]发→◇AT+FTPCID=1

[18:56:53.801]收←◆AT+FTPCID=1

xxO

[18:56:56.442]发→◇AT+FTPSERV="xx.x.87.100"

[18:56:56.447]收←◆AT+FTPSERV="xx.x.87.100"

OK

[18:56:58.090]发→◇AT+FTPUN="user"

[18:56:58.092]收←◆AT+FTPUN="user"

OK

[18:56:59.794]发→◇AT+FTPPW="123456"

[18:56:59.800]收←◆AT+FTPPW="123456"

OK

[18:57:01.450]发→◇AT+FTPGETPATH="/"

[18:57:01.457]收←◆AT+FTPGETPATH="/"

OK

[18:57:09.402]发→◇AT+FTPGETNAME="1223.txt"

[18:57:09.409]收←◆AT+FTPGETNAME="1223.txt"

OK

[18:57:11.362]发→◇AT+FTPGET=1

[18:57:11.367]收←◆AT+FTPGET=1

OK

[18:57:12.340]收←◆

+FTPGET: 1,1

[18:57:17.906]发→◇AT+FTPGET=2,100

[18:57:17.912]收←◆AT+FTPGET=2,100

+FTPGET: 2,100

When doctors think all of their patients are heterosexual, those who identify in other ways can feel//成功下载 100 字节

OK

[18:57:26.557]收←◆

+FTPGET: 1,1

[18:57:27.866]发→◇AT+FTPQUIT//关闭会话

[18:57:27.872]收←◆AT+FTPQUIT

OK

[18:57:27.991]收←◆

+FTPGET: 1,86

[18:57:33.745]发→◇AT+FTPREST=101//设置下载节点为 101 字节，（此断点只要小于文档数据长度，可任意设置），表示模块此次会从 101 字节处开始下载

[18:57:33.751]收←◆AT+FTPREST=101

OK

[18:57:55.929]发→◇AT+FTPGET=1

[18:57:55.936]收←◆AT+FTPGET=1

OK

[18:57:56.577]收←◆

+FTPGET: 1,1

[18:58:21.071]收←◆

+FTPGET: 1,1

[18:58:24.089]发→◇AT+FTPGET=2,500//从 101 字节处开始下载，本次下载 500 字节

[18:58:24.093]收←◆AT+FTPGET=2,500

+FTPGET: 2,500

ignored. They also are less likely to seek medical care. This can lead to health problems including high rates of depression, suicidal behavior, alcohol and drug use and ineffective health screenings. That information comes from LGBTQ advocates in the United States. But changes are coming to the medical field, these advocates say. Feroe displays a button resembling a Harvard School of Medicine coat of arms lion, in rainbow colors that symbolize LGBTQ pride, left, and a button featuring pronou

OK

[18:58:31.069]收←◆

+FTPGET: 1,1

[18:58:45.233]发→◇AT+FTPQUIT

[18:58:45.236]收←◆AT+FTPQUIT//关闭会话

OK

[18:58:45.714]收←◆

+FTPGET: 1,86

其他说明

3.备注

3.1 模块 ftp 默认是被动模式，不支持主动模式，无法设置主动模式，公网下要想模块正常连接服务器，服务器肯定不能设置为主动模式，否则哪家模块都不能用。

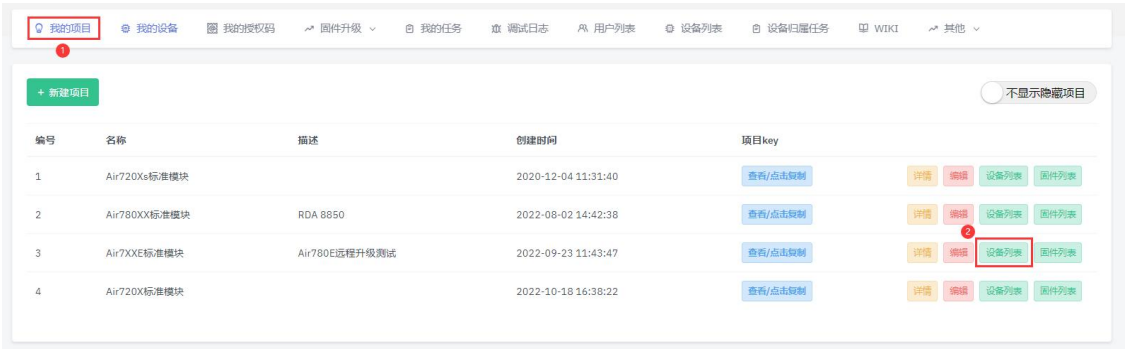
9.5.5 远程升级(FOTA)应用指南

AT 版本的远程升级主要是对 AT 固件的版本升级，升级方式可以通过合宙官方 iot 平台升级，同样也可以使用自己的服务器来搭建升级服务。

合宙 IOT 平台配置

1.1 合宙 IOT 平台配置，点此登录，查询模块是否在自己账号下。

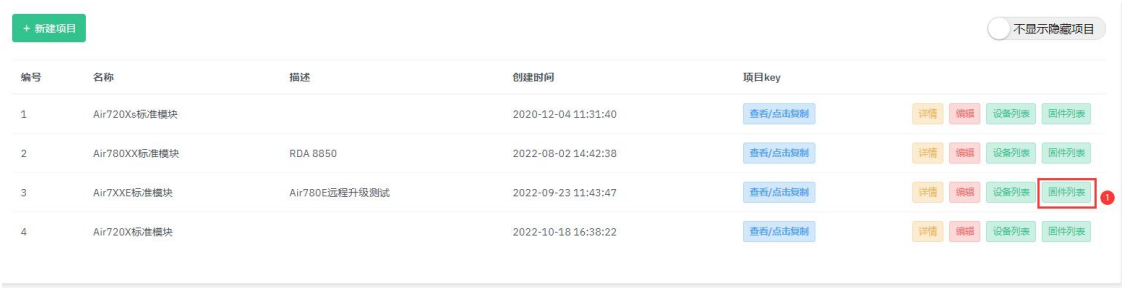
客户向合宙采购 4G 模块时，如果采购人员没有告知合宙这批模块放在 iot.openluat.com 上的哪个产品下，则合宙会以**采购人的手机号为账号，默认密码 888888**，创建一个“**Air7XXE 标准模块**”的项目，此次采购的所有模块都会放在这个项目下。



1.2 配置流程

下载最新固件，固件下载到本地，解压后，将 dfota.bin 文件上传 dfota。

新建固件，上传固件，找到对应项目，点击固件列表。



点击创建固件，上传对应文件。

上传文件后会自动生成固件名称，不需要修改。

如果配置时，是否全项目升级选的“是”，无需配置 imei，按照上述内容配置即可。

如果配置时，是否全项目升级选的“否”，还需添加模块 imei，如下图所示：

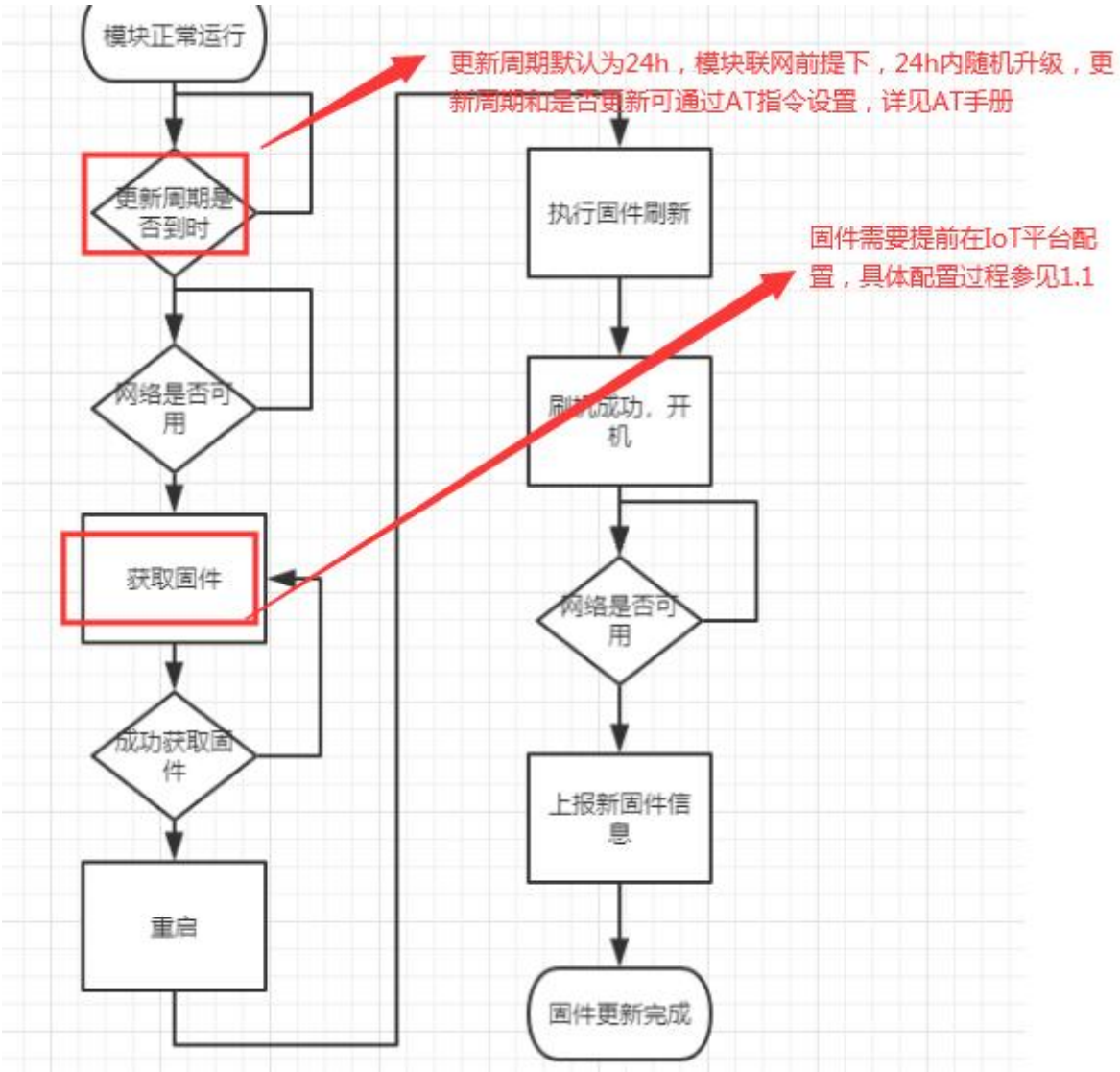
合宙 IOT 平台 AT 指令触发

手动触发升级

如下是模块从 V1020 升级到 V1021 的流程。

自动升级（默认 24 小时以内自动触发）

自动升级是模块定期向后台发送升级请求，来获取新固件进行更新的，大致流程如下:



可以通过 AT+UPGRADE="PERIOD",60 设置自动升级时间，这里设置为 60S 来作为演示：

注意：

A.客户可以通过 IOT 平台查看升级成功数量，失败日志等信息

B.升级过程中若模块断电、重启导致升级失败，可重新升级

C.模块手动触发升级时，若频繁从同一个版本请求（目前默认一小时内 6 次，24 小时内 10 次），IOT 平台设备状态会自动更改为禁止升级，需要在平台设备列表手动解除

D.升级时间根据差分包的大小不同，升级时间会不同。跨越的版本越大，差分包越大，升级所需的时间越长。

自定义服务器升级指南

- 使用自己的服务器来升级模块和使用 iot 平台升级最重要的区别是下载固件的来源不同。

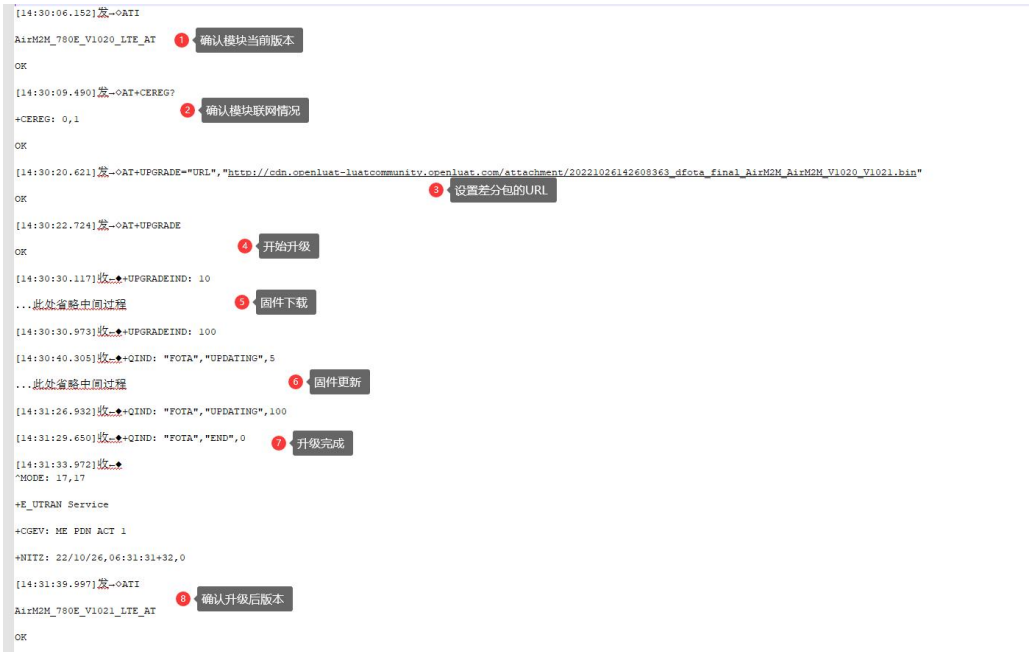
- 其实 FOTA 的过程是新老版本生成差分包再进行升级的过程，那么这个差分的过程在哪里呢？使用合宙 iot 平台升级，差分的过程在 iot 服务器后台，那使用自己的服务器该怎么办呢？答案当然是自己来完成差分的过程，放在服务器等待模块来下载。
- 手动生成差分包的过程使用 doc 平台工具中的差分工具 差分工具

下载生成的差分文件：

- 本次我们从 V1020 升级到 V1021 需要准备各自的 bin 远程升级文件，上传完成后等待差分完成，下载得到的差分文件，这里将名字改为 dfota_final_AirM2M_AirM2M_V1021_V1020.bin。

- 把生成的固件放到服务器上，然后通过 AT+UPGRADE="URL",""指令设置对应的 URL 链接。

升级流程如下，这里选择了 USB 的另外一个端口，可以上报模块的升级过程：



恭喜你成功从 V1021 版本通过自己服务器搭建 FOTA 服务的方式升级到了更新的 V1021 版本！

常见问题

1. at 版本自动远程升级，默认 24 小时检测一轮，在一轮检测时，如果链接服务器失败，在应用层有重试机制吗？如果有的话，多久再重连一次？
2. 有重连机制，间隔一分钟，最多尝试 5 次。

2.我可以从标准 AT 升级为其他 AT 版本吗？

不可以，FOTA 升级只能向上升级**相同版本**的 AT 固件，升级其他 AT 版本会报错（如：AirM2M_780E_V1020_LTE_AT 无法升级到 AirM2M_780E_V1021_LTE_LSAT）

1. 错误码
2. 如果升级失败，可以通过 AT+UPGRADE?查看错误原因
3.

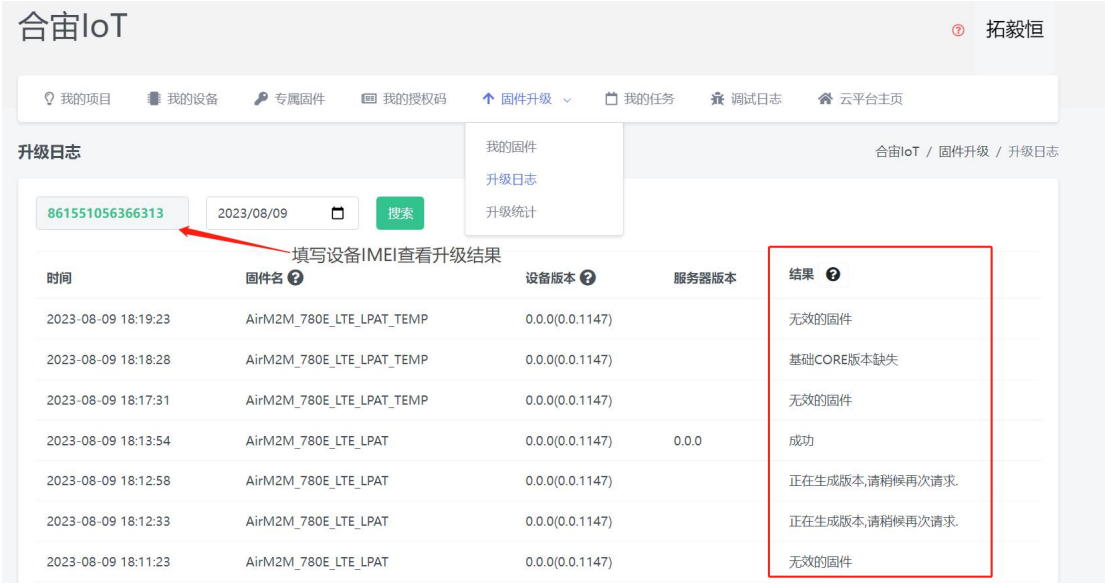
查询命令	AT+UPGRADE?	+UPGRADE: <state> OK	<state>=0 时的返回
		+UPGRADE: <state>,<percent> OK	<state>=1 时的返回
		+UPGRADE: <state>, [<error code>, <http respone>, <fota error cause>]	<state>=2 时的返回

<state>		0	未执行 FOTA
		1	正在从服务器下载固件
		2	固件下载结果
<percent>	升级百分比	1~100	当模块处于<state>=1 的状态时，下载固件的百分比
<error code>		-1000	无效的 URL
		-1001	网络错误
		-1002	服务器链接错误
		-1003	无效固件
		-1004	http 应答错误
		-1005	存储固件错误
		-1006	其他错误

<fota error cause>	FOTA 错误原因	3	无效的设备。找不到设备
		17	无权限
		25	无效的项目
		26	无效的固件

		27	已经是最新版本。通过了前面的检查，但版本号已是最新，或没打开升级开关、没指定 imei 等

也可以通过 **IOT 平台查询升级日志**来确认错误原因



9.5.6 网络状态与信号应用指南

概念解析

先简单介绍几个概念：

位置区：一般是若干个相邻基站区的覆盖区域的总和，位置区识别码叫做 LAC（4G 网络下叫做 TAC，若无特殊说明，本文以下章节，LAC 等价于 TAC）；

基站区：一座基站的覆盖范围，一般包含 3 个小区，即一座基站区等于 3 个小区的覆盖总和；

小区：最小的区域单位，小区识别码叫做 CI；

一个位置区一般有多个基站，一个基站一般包含 3 个或者多个小区，一个小区对应一个小区识别码 CI，小区识别码和人的名字一样有重复，加上位置区识别码 LAC 之后就确定了唯一性。

频段：这个概念太过抽象，直接以中国的三大运营商为例来阐述：

中国的三大运营商，建设了全球最多的 4G 基站。同时，工信部也分配了不同的频段给运营商作为 4G 覆盖使用。三大运营商的 4G 频段，都有相对的高、中、低频段。中国电信就有 B1、B3、B5，中国联通有 B1、B3、B8，中国移动则是 TD-LTE/FDD-LTE 混合组网，有最多的频段，包括 B3、B8、B34、B38、B39、B40、B41。下图是 4G 频段表以及上下行频率范围（可能不准确，仅供参考）：

Table 5.5-1 E-UTRA frequency bands

E-UTRA Operating Band	Uplink (UL) operating band BS receive UE transmit F_{UL_low} – F_{UL_high}	Downlink (DL) operating band BS transmit UE receive F_{DL_low} – F_{DL_high}	Duplex Mode
1	1920 MHz – 1980 MHz	2110 MHz – 2170 MHz	FDD
2	1850 MHz – 1910 MHz	1930 MHz – 1990 MHz	FDD
3	1710 MHz – 1785 MHz	1805 MHz – 1880 MHz	FDD
4	1710 MHz – 1755 MHz	2110 MHz – 2155 MHz	FDD
5	824 MHz – 849 MHz	869 MHz – 894MHz	FDD
6[1]	830 MHz – 840 MHz	875 MHz – 885 MHz	FDD
7	2500 MHz – 2570 MHz	2620 MHz – 2690 MHz	FDD
8	880 MHz – 915 MHz	925 MHz – 960 MHz	FDD
9	1749.9 MHz – 1784.9 MHz	1844.9 MHz – 1879.9 MHz	FDD
10	1710 MHz – 1770 MHz	2110 MHz – 2170 MHz	FDD
11	1427.9 MHz – 1447.9 MHz	1475.9 MHz – 1495.9 MHz	FDD
12	699 MHz – 716 MHz	729 MHz – 746 MHz	FDD
13	777 MHz – 787 MHz	746 MHz – 756 MHz	FDD
14	788 MHz – 798 MHz	758 MHz – 768 MHz	FDD
15	Reserved	Reserved	FDD
16	Reserved	Reserved	FDD
17	704 MHz – 716 MHz	734 MHz – 746 MHz	FDD
18	815 MHz – 830 MHz	860 MHz – 875 MHz	FDD
19	830 MHz – 845 MHz	875 MHz – 890 MHz	FDD
20	832 MHz – 862 MHz	791 MHz – 821 MHz	
21	1447.9 MHz – 1462.9 MHz	1495.9 MHz – 1510.9 MHz	FDD
22	3410 MHz – 3490 MHz	3510 MHz – 3590 MHz	FDD
23	2000 MHz – 2020 MHz	2180 MHz – 2200 MHz	FDD
24	1626.5 MHz – 1660.5 MHz	1525 MHz – 1559 MHz	FDD
25	1850 MHz – 1915 MHz	1930 MHz – 1995 MHz	FDD
26	814 MHz – 849 MHz	859 MHz – 894 MHz	FDD
27	807 MHz – 824 MHz	852 MHz – 869 MHz	FDD
28	703 MHz – 748 MHz	758 MHz – 803 MHz	FDD
…			
33	1900 MHz – 1920 MHz	1900 MHz – 1920 MHz	TDD
34	2010 MHz – 2025 MHz	2010 MHz – 2025 MHz	TDD
35	1850 MHz – 1910 MHz	1850 MHz – 1910 MHz	TDD
36	1930 MHz – 1990 MHz	1930 MHz – 1990 MHz	TDD
37	1910 MHz – 1930 MHz	1910 MHz – 1930 MHz	TDD
38	2570 MHz – 2620 MHz	2570 MHz – 2620 MHz	TDD
39	1880 MHz – 1920 MHz	1880 MHz – 1920 MHz	TDD
40	2300 MHz – 2400 MHz	2300 MHz – 2400 MHz	TDD
41	2496 MHz – 2690 MHz	2496 MHz – 2690 MHz	TDD
42	3400 MHz – 3600 MHz	3400 MHz – 3600 MHz	TDD
43	3600 MHz – 3800 MHz	3600 MHz – 3800 MHz	TDD
44	703 MHz – 803 MHz	703 MHz – 803 MHz	TDD

Note 1: Band 6 is not applicable.

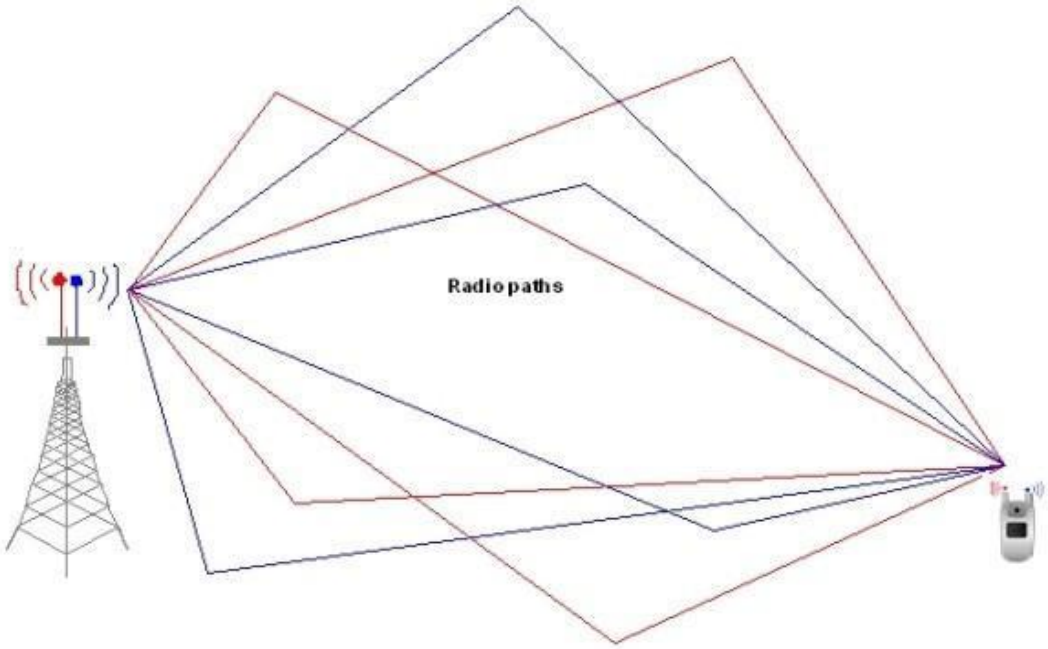
运营商在建设自己的 4G 基站时，根据工信部分配的频段，可以决定某个基站是支持分配的全部频段还是部分频段；例如中国移动在某个地区建设自己的基站，可以配置这个基站支

持 B3、B8、B34、B38、B39、B40、B41 全部频段，也可以仅支持 B3、B8 频段，完全取决于运营商。

Air780E 系列模块，支持 B1、B3、B5、B8、B34、B38、B39、B40、B41 频段，是支持中国 4G 网络全频段的纯 4G 模块（不支持 2G 和 3G 网络）。

信号强度：运营商的基站建设好之后，一旦开始工作，基站就对外发射信号。此时，终端设备（模块产品、手机产品等）一开机，就开始搜索信号；信号强度就表示：终端设备接收到的基站发射出的信号强度。

基站发射的信号，通常都是经过折射、反射、衍射、散射等方式才被终端设备接收到，接收到的信号强度和终端设备的天线灵敏度、基站的距离、终端设备所处的环境等有关系，如下图所示：



网络注册流程

网络注册，是指从开机搜网到注册到网络的过程，或者从退出飞行模式到注册到网络的过程，此过程（仅供参考，省去 N 多细节，欲知详情，请自行搜索 LTE 注网流程 了解）可以概括为如下几步：

1、 终端设备开机或者退出飞行模式后，读取 PLMN

- 公共陆地移动网（Public Land Mobile Network，简称：PLMN），由政府或它所批准的经营者，为公众提供陆地移动通信业务目的而建立和经营的网络
- PLMN=MCC+MNC，以中国大陆的运营商为例，有中国移动、中国电信、中国联通三大运营商，MCC 都是 460，中国移动的 MNC 有 00、02、07（后续可能还会扩充）
- 从 SIM 卡中以及模块固件的配置文件中读取 PLMN 列表

2、终端设备根据自己支持的频段，扫描基站信号，找到支持的 PLMN

3、根据小区选择标准（S 标准），选择一个最优的小区

4、驻留在最优小区

Air780E 系列模块，整个网络注册流程，自动完成，不需要人为参与。

AT 版本网络注册说明

Air780E 系列模块，仅支持 4G 网络，所以本章节仅描述和 4G 网络有关的命令 AT+CEREG。

这个命令最常使用的两种功能是：**设置命令 AT+CEREG=<n>和查询命令 AT+CEREG?**

语法规则如下图所示：

语法规则：

命令类型	语法	返回
设置命令	AT+CEREG=<n>	OK
查询命令	AT+CEREG?	+CEREG: <n>,<stat>[,[<tac>],[<ci>],[<AcT>[,<cause_type>,<reject_cause>]]] OK
测试命令	AT+CEREG=?	+CEREG: (list of supported <n>s) OK
URC上报	+CEREG: <stat>	<n>=1，当网络注册状态发生变化时
	+CEREG: <stat>[,[<tac>],[<ci>],[<AcT>]]	<n>=2，当网络注册状态或驻留网小区发生变化时
	+CEREG: <stat>[,[<tac>],[<ci>],[<AcT>][,<cause_type>,<reject_cause>]]	<n>=3，当网络注册状态或驻留网小区发生变化时

参数定义:

参数	定义	取值	对取值的说明
<n>	URC 上报状态	0	关闭主动上报
		1	允许主动上报+CEREG: <stat>
		2	允许主动上报+CREG:<stat>[,<lac>,<ci>]
		3	允许主动上报 +CEREG: <stat>[,[<tac>],[<ci>],[<AcT>][,<cause_type>,<reject_cause>]]
<stat>	当前网络注册状态（只有1和5表示正常状态，其余都不是正常状态）	0	未注册；ME 当前没有搜索要注册业务的新运营商
		1	已注册，本地网
		2	未注册，但 ME 正在搜索要注册业务的新运营商
		3	注册被拒绝
		4	未知(超出 E-UTRAN网覆盖范围)
		5	注册漫游网
		6	注册归属地"SMS only"业务
		7	注册漫游地"SMS only"业务
		8	仅附着紧急承载业务(See NOTE 2)
		9	注册归属地"CSFB not preferred"业务
		10	注册漫游地"CSFB not preferred"业务
		11	仅紧急业务可用
<tac>	跟踪区识别码	-	字符串型，16进制数
<ci>	小区识别码	-	字符串型，16进制数
<act>	已注册网络的接入技术	0	GSM
		1	GSM Compact
		2	UTRAN
		3	GSM w/EGPRS (NOTE 3)
		4	UTRAN w/HSDPA (NOTE 4)
		5	UTRAN w/HSUPA (NOTE 4)
		6	UTRAN w/HSDPA and HSUPA (NOTE 4)
		7	E-UTRAN
		8	UTRAN HSPA+
<cause_type>	注册失败类型	0	显示<reject_cause>包括一个 EMM 原因值（请参考 3GPP TS 24.301 Annex A）
		1	显示<reject_cause>值由厂家定义
<reject_cause>	注册失败原因		此值的类型由<cause_type>定义

AT 版本可以通过<stat>的值来判断网络注册状态，1 和 5 表示注册上网络，其余都表示未注册上网络

信号强度参考指标

信号强度有两种不同的参考指标：

rssi：接收信号强度指示；是一种传统的信号强度参数，过去一直用于 2G 网络的信号强度指示；很多用户仍然习惯使用这个参数来表示 4G 网络的信号强度；在合宙 4G Cat.1 模块中，rssi 的取值范围为 0 到 31，99；**AT+CSQ** 命令可以查询 rssi

rsrp：参考信号接收功率；是 4G 网络特有的测量参数，相对于 rssi 来说，可以更精确的测量 4G 网络的信号强度；rsrp 等级的取值范围为 0 到 97，255；**AT+CESQ** 命令可以查询 rsrp

rsrp 等级和 rsrp 功率的对应关系表如下：

rsrp 等级	rsrp 功率
0	rsrp < -140 dBm
1	-140 dBm <= rsrp < -139 dBm
2	-139 dBm <= rsrp < -138 dBm
	...
95	-46 dBm <= rsrp < -45 dBm
96	-45 dBm<= rsrp < -44 dBm
97	-44 dBm <= rsrp
255	未知或不可测

这个表格是 3GPP 协议的标准规范，rsrp 等级和 rsrp 功率是简单的线性关系；

实际测试，在实网中，rsrp 功率只要大于-71dBm，对应的 rsrp 等级大于等于 70，信号就已经很好，实网环境很难出现大于 70 的情况

rsrp 等级和 rssi 等级的转换关系，算法并不是简单的线性关系，实际网络下，rssi 的计算方式和多个网络参数有关，计算比较复杂，此处不再列举算式。下表离散采样了几个值来简单描述二者的对应关系，仅供参考

rsrp 等级	rssi 等级
>69	31
66	30
63	28
58	26

rsrp 等级	rssi 等级
54	24
50	22
44	19
40	17
35	15
30	12
26	10
19	9
15	7
9	7

信号强度仅仅表示当前驻留小区的网络覆盖程度，一般来说，信号强度和网络通信稳定率是正比关系。信号强度差，网络数据通信相对就不稳定；但不能绝对的认为信号强度好，网络数据通信就一定稳定，和网络是否拥堵等其他因素也有关系；经验值如下：

- rsrp 等级小于等于 15 或者 rssi 等级小于等于 7，可以认为网络很差，不足以支撑网络通信
- rsrp 等级小于等于 26 或者 rssi 等级小于等于 10，可以认为网络不太稳定，不足以支撑正常的网络通信，会概率性掉线、丢包
- rsrp 等级大于 26 或者 rssi 等级大于 10，可以认为网络较好

AT 版本信号强度说明

Air780EP 系列模块，仅支持 4G 网络，所以本章节仅 4G 网络信号强度。

AT+CESQ 可以查询 **rsrp**

语法规则如下图所示：

语法规则：

命令类型	语法	返回
执行命令	AT+CESQ	+CESQ: <rxlev>,<rxqual >,<rscp>,<ecno>,<rsrq>,<rsrp> OK

参数定义:

参数	定义	取值	对取值的说明
<rsrq>	reference signal received quality (请参考 3GPP TS 36.133 subclause 9. 1.7) ；整数型	0	rsrq < -19.5 dB
		1	-19.5 dB ≤ rsrq < -19 dB
		2	-19 dB ≤ rsrq < -18.5 dB
		...	...
		32	-4 dB ≤ rsrq < -3.5 dB
		33	-3.5 dB ≤ rsrq < -3 dB
		34	-3 dB ≤ rsrq
		255	未知或不可测
<rsrp>	reference signal received power (请参考 3GPP TS 36.133 subclause 9. 1.4) ；整数型	0	rsrp < -140 dBm
		1	-140 dBm ≤ rsrp < -139 dBm
		2	-139 dBm ≤ rsrp < -138 dBm
		...	...
		95	-46 dBm ≤ rsrp < -45 dBm
		96	-45 dBm ≤ rsrp < -44 dBm
		97	-44 dBm ≤ rsrp
		255	未知或不可测

AT+CSQ 可以查询 **rssi**

语法规则如下图所示：

语法规则：

命令类型	语法	返回
执行命令	AT+CSQ	+CSQ: <rssi>,<ber> OK
测试命令	AT+CSQ=?	+CSQ: (list of supported <rssi>s),(list of supported<ber>s) OK

参数定义：

参数	定义	取值	对应的 rsrp 功率（dbm）
<rssi>	接收信号强度指示 (received signal strength indication)	0	参考本文3.1章节中的两个表格理解
		1	
		.....	
		31	
		99	未知或不可测
<ber>	信道误码率(bit error rate，只有通话建立后，才能获知该值)	0~7	GSM 05.08 section 8.2.4所示的RXQUAL值
		99	未知或不可测

注意：AT+CESQ 查询出的 rsrp 比 AT+CSQ 查询出的 rssi 更能准确的反映信号强度，在条件允许的情况下，建议使用 AT+CESQ

常见问题

4.1 没有 sim 卡，可以查询到信号强度吗？

可以；即使没有 sim 卡，模块也能搜索到基站发射出的信号，能搜索到就能感知到信号；只是无法驻留到小区

4.2 如何判断主板信号接收性能？

参考：如何判主板的 4G/2G 信号接收性能的好坏

4.3 可以在中国大陆外的哪些国家和地区使用？

按照如下步骤确认：

1、参考 Air780EP 的规格书可知：Air780E 支持 B1/B3/B5/B8/B34/B38/B39/B40/B41 频段

2、参考世界各国蜂窝移动通信网络频段分布【此数据可能不准确，仅供参考，针对具体的国家和地区，需要客户去实际确认当前国家的运营商和频段信息】获取要查询的国家和地区频段，例如以中国台湾地区为例，中国台湾所有运营商的频段如下图所示：

更新时间：2017/11/5

频段（MHz）	Chunghwa Telecom		Taiwan Mobile		FarEasTone	
	uplink	downlink	uplink	downlink	uplink	downlink
700 b67						
850 b5						
900 b8	895~905	940~950				
1800 b3	1725~1735 1755~1775	1820~1830 1850~1870	1710~1725	1805~1820	1735~1755	1830~1850
2100 b1	1960~1975	2150~2165	1940~1960	2130~2150	1920~1935	2110~2125
2600 b7	2520~2540 2560~2570	2640~2660 2680~2690			2540~2560	2660~2680
2600 b38					2595~2620	

频段（MHz）	Gt (APT)		T Star	
	uplink	downlink	uplink	downlink
700 b67	703~713 723~728	758~768 778~783		
850 b5				
900 b8	905~915	950~960	885~895	930~940
1800 b3				
2100 b1			1935~1940	2125~2130
2600 b38	2570~2595		2500~2520	2620~2640

由上图可知：中国台湾的运营商以及支持的频段如下：

1、Chunghwa Telecom：B1/B3/B7/B8；但是 Air780E 不支持 B7 频段，所以 Air780E 在台湾不能使用 Chunghwa Telecom 运营商

2、Taiwan Mobile：B1/B3；Air780E 支持这两种频段，所以 Air780E 在台湾可以使用 Taiwan Mobile 运营商

3、FarEasTone：B1/B3/B7/B38；但是 Air780E 不支持 B7 频段，所以 Air780E 在台湾不能使用 FarEasTone 运营商

4、Gt(APT)：B8/B38/B67；但是 Air780E 不支持 B67 频段，所以 Air780E 在台湾不能使用 Gt(APT) 运营商

5、T Star：B1/B8/B38；Air780E 支持这两种频段，所以 Air780E 在台湾可以使用 T Star 运营商

综上所述，Air780E 可以在中国台湾地区使用，但是仅支持 Taiwan Mobile 和 T Star 两种运营商

注意：此数据仅供参考，因不同国家地区的运营商和支持的频段信息会不断发生变化，所以在模块选型时，一定要在当地确认实时信息

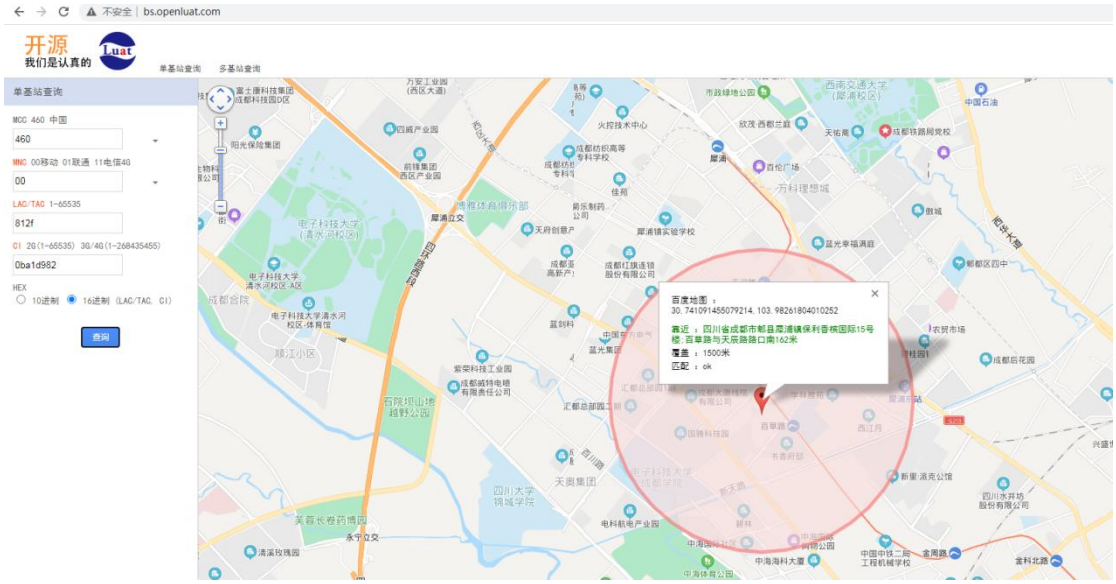
4.4 如何在网站上查询小区对应的位置信息？

例如：通过 AT+CEREG? 查询到的小区信息如下：

```
+CEREG: 2,1,"812f","0ba1d982",7
```

这里的"812f"和"0ba1d982"都是 16 进制格式

合宙官方提供了一个在线查询位置的位置服务：在线根据小区查询位置



注意：任何一个在线查询网站，查询结果仅供参考；无法保证 100%可以成功，因为小区 id 不断在新增，查询网站的后台数据库很难做到实时更新小区 id 数据库

4.5 为什么模块在盒子里外接天线盒子的封闭状态会影响信号值？

天线距离盒子太近盒子的密封状态会影响模块信号值，板子在密闭环境需要外接长一些的天线，这样信号值会高一些。

4.6 信号强度和网络数据业务的稳定性关系

信号强度的查询方法：

AT 版本：通过 AT+CSQ 查询 rssi 值，AT+CESQ 命令可以查询 rsrp 值，取值范围参考 Luat4G 模块 AT 命令手册

LuatOS-Air 版本：通过 net.getRssi()返回信号强度值

信号强度仅仅表示当前驻留小区的网络覆盖程度，一般来说，信号强度和网络通信稳定率是正比关系。信号强度差，网络数据通信相对就不稳定；但不能绝对的认为信号强度好，网络数据通信就一定稳定，和网络是否拥堵等其他因素也有关系；经验值参考本文：

三、信号强度 章节

4.7 联通基站全部拆除换成移动基站，先前的联通流量卡还能上网吗

运营商决定，以实测为准

4.8 同种模块，有的卡能够快速联网，有的需要好几分钟

如：14408/14409 号段为 IMSI 单设 46008 开头识别码，如果厂家使用原有的软件方案并未将 MNC46008 写入软件中，可能会产生兼容问题，不能连接网关服务器数据，无法正常联网使用网络。

解决方案如下：

1.联系生产厂家协助软件版本做出调整增加 imsi 识别

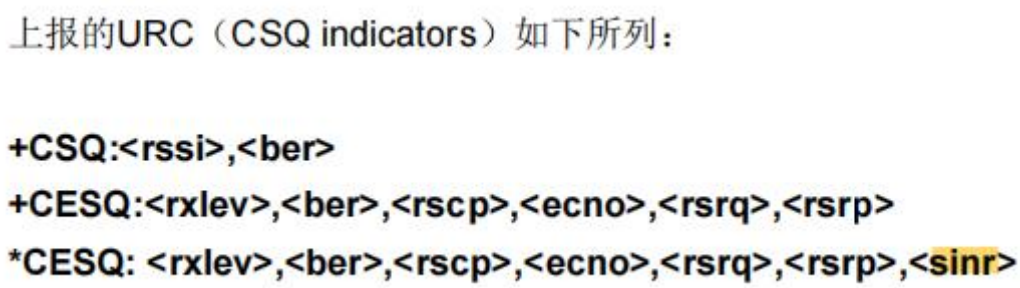
2.软件开发植入 MCC460 MNC08

3.14408、14409 号段的卡设置 APN：CMIOT

4.9 请问查询 SINR 和 TA 的指令是什么

SINR 获取方法如下，TA 获取不到

打开 CSQ 主动上报：AT*CSQ




```
AT*CSQ=1

OK

+CSQ: 17,99

+CESQ: 99,99,255,255,18,34

*CESQ: 99,99,255,255,18,34,6
```

语法规则：

命令类型	语法	返回
设置命令	AT*CSQ=<n>	OK
查询命令	AT*CSQ?	*CSQ:<n> OK
测试命令	AT*CSQ=?	*CSQ = (list of supported <n>s) OK

参数定义:

参数	定义	取值	对取值的说明
<n>		0	CSQ Indicator is disabled
		1	CSQ Indicator is enabled
<rssi>,<ber>	please refer to AT+CSQ		
<rxlev>,<rscp>,<ecno>,<rsrq>,<rsrp>	please refer to AT+CESQ		
<sinr>	signal to interference plus noise ratio		

4.10 移动电信联通的 SIM 卡都可以使用，如果限制只能使用一家运营商的卡的话，这个支持么？

通过频段去限制运营商，是存在问题的，同一个频段，因为带宽原因，可能不只一个运营商支持，比如 band40，有 2 个运营商都支持，目前没有直接限制运营商的办法

11 学生卡可以与 SIM 卡进行强绑定，也就是一张学生卡只能使用一个固定号码的手机号，这个可以支持么？

模块没法支持 ，现在 sim 卡有机卡绑定的规则 ，能满足这个要求

9.5.7 基站/WIFI 定位应用指南

应用概述

手机在插入 sim 卡开机以后，需要搜索周围的基站信息（就是找信号），通常在有信号的地方，手机能搜索到的基站不止一个，就像你的笔记本在小区里能搜到不止一个开着的路由器一样。然后手机会自动从这些基站中选择信号最好的（假如是基站 A）连接注册。其余的基站就不用了吗？不不，手机仍然搜索着它们，一旦你离开基站 A 一段距离，基站 A 的信号不如基站 B 了，手机会自动切换成基站 B。这也就是为什么同样是待机一天，你在火车上比在家里耗电要多的原因，手机需要不停的搜索、连接基站。

当手机的网络覆盖到一定程度的时候，基站定位也就应运而生了。

其原理如下：

众所周知，距离基站越远，信号越差，所以用信号的强度可以大致估计距离基站的远近。好了，由于基站都是中国移动中国联通等运营商建立的，所以他们可以确定每个基站准确的位置，当手机同时搜索到至少三个基站的信号时（现在的网络覆盖这是很轻松的一件事情），手机除了连接信号最好的一个外，同时根据基站信号的强弱，大致估计出距离基站的远近。由于基站位置是确定的，所以确定位置是很容易的，只需要以基站为圆心，距离为半径多次画圆即可，需要三个基站的信号可以准确的定位。

相对于三个基站的信号来说，如果只搜索到一个基站或者两个基站，也可以定位，只不过，相对来说不准确。

WIFI 定位的原理和基站定位原理类似。

4G 模块支持 wifi 定位/基站定位，应用的基本流程如下：

```
1、激活 PDP（参考：http://oldask.openluat.com/article/937）

2.发起定位请求



第 1 步出现异常后：参考 http://oldask.openluat.com/article/937 进行异常处理
```

示例

下面是 AT 命令序列，如果不理解 AT 命令含义，请自行参考 AT 手册。

基站定位

```
AT/训练波特率

OK

AT+SAPBR=1,1/激活 PDP

OK

AT+SAPBR=2,1/查询是否激活成功，成功能查询到 IP

+SAPBR: 1,1,"10.150.29.242"

OK

AT+CIPGSMLOC=1,1/查询基站定位

+CIPGSMLOC: 0,31.174749,121.600788,2021/05/11,11:29:41

OK
```

wifi 定位

```
AT/训练波特率

OK
```

```
AT+SAPBR=1,1/激活 PDP

OK

AT+SAPBR=2,1/查询是否激活成功，成功能查询到 IP

+SAPBR: 1,1,"10.150.29.242"

OK

AT+WIFILOC=1,1

+WIFILOC: 0,31.175017,121.600989,2021/05/11,13:54:21

OK
```

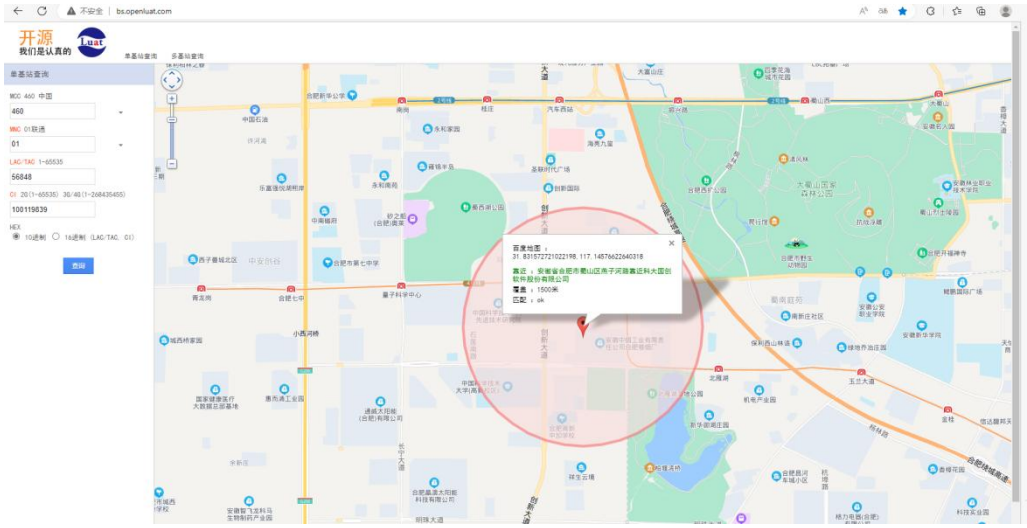
常见问题

1、如何在网站上查询小区对应的位置信息？

例如：通过 AT+CCED=0,1 查询到的小区信息如下：

+CCED:LTE current cell: 460,01,460060086257105,0,3,5,1650,100119839,62,24,56848,34,86

合宙官方提供了一个在线查询位置的位置服务：在线根据小区查询位置



2、wifi 定位失败的可能原因有哪些？

- 如果没有搜索到 wifi 热点，用手机对比确认下是否可以搜索到附件 wifi，如果手机可以搜到，确认下模块的板子上是否有接 wifi 天线
- 如果可以搜索到 wifi 热点，但是 wifi 热点很少，用手机对比确认下搜索到附件 wifi 数量，如果数量差别太大，确认下模块的板子上是否有接 wifi 天线或者所接的天线是否支持 2.4G 频段

3、基站定位获取的经纬度是什么格式的？

- 基站定位获取的经纬度为 WGS-84 格式。
- 各种坐标系说明以及转换方法参考：http://old.openluat.com/GPS-Offset.html

4、Wifi 定位获取的经纬度是什么坐标系的？

- Wifi 定位获取的经纬度是 WGS-84 地心坐标系（ GPS 全球定位系统建立的坐标系统）。

5、wifi 定位精度

- 根据设备获取的 WiFi 的信息进行定位，WIFI 定位精度一般不受使用环境影响，主要和单一 WIFI 辐射半径、WIFI 覆盖密度有关。一般来说，WIFI 精度在 3 米－200 米左右。

6、基站定位精度

- 搜索到的小区越多，定位的精度越高；一般来说，城市中心定位精度比郊区和农村定位精度高，城市中心的定位精度在几十米到几百米不等，郊区和农村的定位精度更低，可能会有几千米甚至更多的误差。
- 基站定位的误差都比较大，如果需要准确定位，请使用支持 GPS 的模块。

7、WIFI 定位支持国外定位吗？

- 不支持，只支持国内定位。

9.5.8 文件系统应用指南

应用概述

模块具有 4Mb 的 flash 空间，除了模块自身程序占用以外还剩余了一部分空间可供用户使用。

文件操作

- 创建一个一级目录 test，OK 说明目录创建成功

```
AT+FSMKDIR="/testdir"

OK
```

- 创建二级目录 test/testdir/testdir， OK 说明二级目录创建成功

```
AT+FSMKDIR="testdir/testdir"

OK
```

- 一级目录下创建文件/testdir/test.txt，OK 说明文件创建成功

```
AT+FSCREATE="/testdir/test.txt"

OK
```

- 显示一级目录列表

```
AT+FSLS="/testdir"

testdir\

test.txt

OK
```

- 文件内写入内容

```
AT+FSWRITE="/testdir/test.txt", 0, 10, 10
```

当出现">"后，输入写入的数据：1234567890。OK 表示写入成功

```
>1234567890

OK
```

- 读取文件内容

```
AT+FSREAD="/testdir/test.txt",0,10,10

1234567890

OK
```

- 获取文件系统可用空间大小

```
AT+FSMEM

+FSMEM: 204800 bytes

OK
```

- 获取文件大小

```
AT+FSFLSIZE="/testdir/test.txt"

+FSFLSIZE: 10

OK
```

- 删除文件

```
AT+FSDEL="/testdir/test.txt"

OK
```

- 删除目录

```
AT+FSRMDIR="/testdir/testdir"

OK
```

常见问题

文件读写有长度限制吗？

有限制，文件读写长度都不能超过 10240 字节， 如果超出此值， 则读写都应分次进行。

9.5.9 阿里云应用指南

应用概述

使用 AT 方式连接阿里云分为一机一密和一型一密，其中一机一密又包含 HTTP 认证二次连接和 MQTT 直连两种方式。

步骤

1. 训练波特率
2. 激活 PDP 参考
3. 连接阿里云

材料准备

准备设备证书及产品证书。

1.

```
--设备证书

{

  "ProductKey": "a1qsHBbKtmc",

  "DeviceName": "866714044533668",
```

```
  "DeviceSecret": "c25262e140bb58d4ec3de5028a2cc836"
}--产品证书
{
    "ProductSecret": "nNmPacH5aL8EViD4",
    "ProductKey": "a1qsHBbKtmc"
}
```

阿里云操作

产品操作

打开阿里云找到物联网平台，开通业务后进入控制台。

点开设备管理的产品页面，点击新建产品。根据需求和图示说明创建产品。

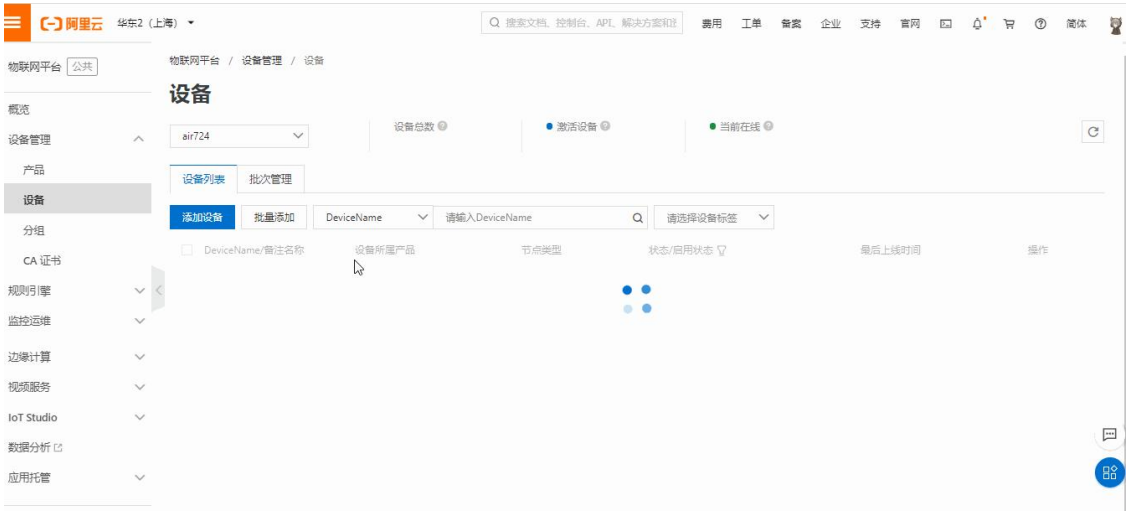
具体详细介绍见阿里云页面

设备操作

创建产品完成后就可以进入设备页面添加设备，在对应产品页面进入设备管理，按照提示添加设备

（在做正式产品时建议使用 imei 为 devicename，方便后期维护）

阿里云设备创建



连接阿里云

一机一密 AT HTTP 鉴权连接

首先需要使用 HmacMD5 计算哈希值用于签名，使用网址：计算工具

```
明文：
clientId866714044533668deviceName866714044533668productKeya1qsHBbKtmc

秘钥： c25262e140bb58d4ec3de5028a2cc836

结果： 655d19e9c43e27d8c66419a6724eb56a
```

拼接 DOWNLOAD 参数

productKey=a1qsHBbKtmc&sign=655d19e9c43e27d8c66419a6724eb56a&clientId=866714044533668&deviceName=866714044533668

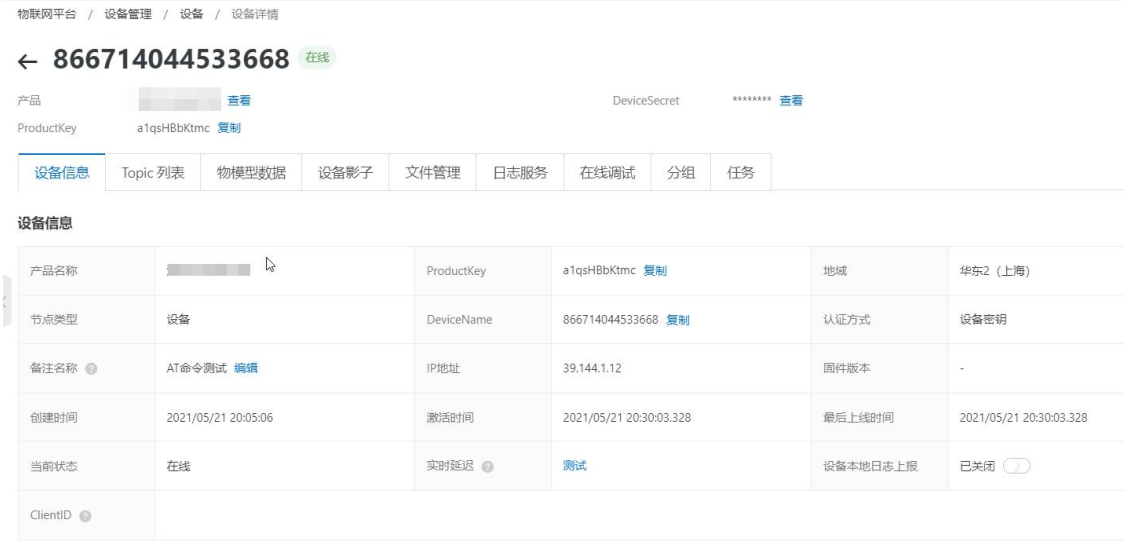
具体交互流程如下所示

```
[2021-05-21 20:24:55.620]# SEND ASCII>

AT+HTTPINIT

[2021-05-21 20:24:55.674]# RECV ASCII>

AT+HTTPINIT

OK

[2021-05-21 20:25:04.547]# SEND ASCII>
```

```
AT+HTTPPARA="URL",https://iot-auth.cn-shanghai.aliyuncs.com/auth/devicen
ame

[2021-05-21 20:25:04.601]# RECV ASCII>

AT+HTTPPARA="URL",https://iot-auth.cn-shanghai.aliyuncs.com/auth/devicen
ame

OK

[2021-05-21 20:25:11.724]# SEND ASCII>

AT+HTTPPARA="USER_DEFINED","Content-Type:
application/x-www-form-urlencoded"

[2021-05-21 20:25:11.776]# RECV ASCII>

AT+HTTPPARA="USER_DEFINED","Content-Type:
application/x-www-form-urlencoded"

OK

[2021-05-21 20:25:22.028]# SEND ASCII>

AT+HTTPDATA=120,20000

[2021-05-21 20:25:22.078]# RECV ASCII>

AT+HTTPDATA=120,20000

DOWNLOAD

[2021-05-21 20:25:29.643]# SEND ASCII>

productKey=a1qsHBbKtmc&sign=655d19e9c43e27d8c66419a6724eb56a&clientId=86
6714044533668&deviceName=866714044533668

[2021-05-21 20:25:42.079]# RECV ASCII>

OK

[2021-05-21 20:25:52.388]# SEND ASCII>

AT+HTTPACTION=1
```

```
[2021-05-21 20:25:52.450]# RECV ASCII>

AT+HTTPACTION=1

OK

[2021-05-21 20:25:53.034]# RECV ASCII>

+HTTPACTION: 1,200,124

[2021-05-21 20:25:59.556]# SEND ASCII>

AT+HTTPREAD

[2021-05-21 20:25:59.624]# RECV ASCII>

AT+HTTPREAD

+HTTPREAD: 124

{"code":200,"data":{"iotId":"8rCqP6OVgi90P5a479IC000000","iotToken":"^1^1621599952882^218a0a267fc6c36"},"message":"success"}

OK--收到服务器返回数据其中iotId为MQTT Client的用户名，iotToken的值为MQTT Clientd的密码

[2021-05-21 20:29:48.876]# SEND ASCII>

AT+MCONFIG="866714044533668","8rCqP6OVgi90P5a479IC000000","^1^1621599952882^218a0a267fc6c36"

[2021-05-21 20:29:48.932]# RECV ASCII>

AT+MCONFIG="866714044533668","8rCqP6OVgi90P5a479IC000000","^1^1621599952882^218a0a267fc6c36"

OK

[2021-05-21 20:29:57.900]# SEND ASCII>

AT+SSLMIPSTART="a1qsHBbKtmc.iot-as-mqtt.cn-shanghai.aliyuncs.com",1883
```

```
[2021-05-21 20:29:57.950]# RECV ASCII>

AT+SSLMIPSTART="a1qsHBbKtmc.iot-as-mqtt.cn-shanghai.aliyuncs.com",1883

OK

[2021-05-21 20:29:58.474]# RECV ASCII>

CONNECT OK

[2021-05-21 20:30:03.332]# SEND ASCII>

AT+MCONNECT=1,120

[2021-05-21 20:30:03.383]# RECV ASCII>

AT+MCONNECT=1,120

OK

[2021-05-21 20:30:03.493]# RECV ASCII>

CONNACK OK
```



一机一密 AT 直连 MQTT

当设备属于公共实例，使用一型一密预注册认证方式时，动态注册参数如下：

```
mqttClientId:

clientId+"|securemode=3,signmethod=hmacsha1,timestamp=132323232|"
```

```
mqttUsername: deviceName+"&"+productKey

mqttPassword: sign_hmac(deviceSecret,content)
```

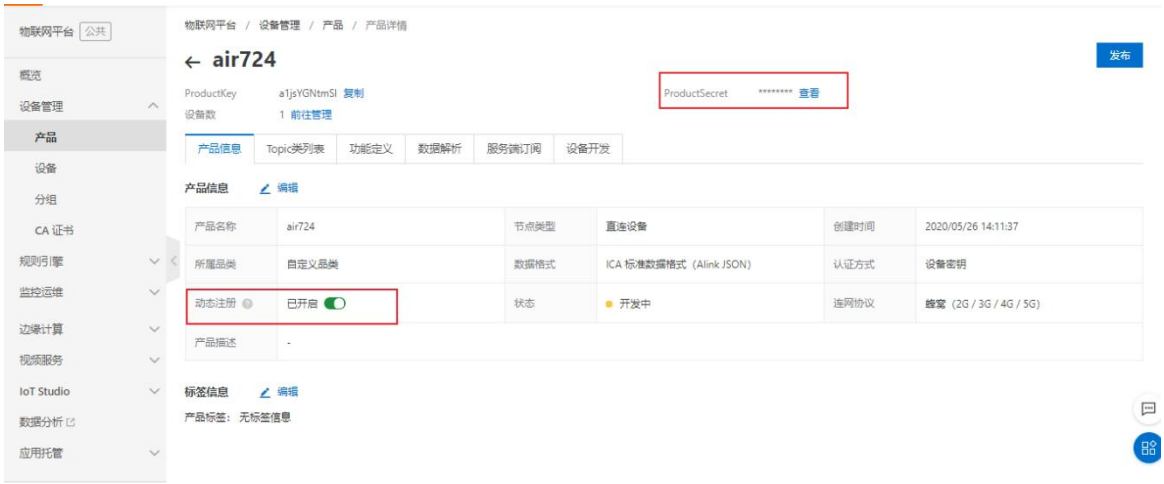
参数	说明
clientId	客户端 ID：长度在 64 个字符内，建议使用 IMEI 或 SN 码方便区分不同客户端
securemode	安全模式 2(TLS)直连模式 3)(TCP 直连模式)
signmethod	表示签名算法类型。支持 hmacmd5,hmacsha1,hmacsha256。默认为 hmacmd5
timestamp	表示当前时间毫秒值，可以不传递
mqttPassword	前文计算的签名值
具体交互流程如下所示	

```
[2021-05-21 21:26:59.670]# SEND ASCII>

AT+MCONFIG="866714044533668|securemode=3,signmethod=hmacmd5|","866714044533668&a1qsHBbKtmc","655d19e9c43e27d8c66419a6724eb56a"

[2021-05-21 21:26:59.725]# RECV ASCII>

AT+MCONFIG="866714044533668|securemode=3,signmethod=hmacmd5|","866714044533668&a1qsHBbKtmc","655d19e9c43e27d8c66419a6724eb56a"

OK

[2021-05-21 21:27:00.958]# SEND ASCII>

AT+SSLMIPSTART="a1qsHBbKtmc.iot-as-mqtt.cn-shanghai.aliyuncs.com",1883

[2021-05-21 21:27:01.007]# RECV ASCII>

AT+SSLMIPSTART="a1qsHBbKtmc.iot-as-mqtt.cn-shanghai.aliyuncs.com",1883

OK

[2021-05-21 21:27:01.567]# RECV ASCII>

CONNECT OK
```

```
[2021-05-21 21:27:02.333]# SEND ASCII>

AT+MCONNECT=1,120

[2021-05-21 21:27:02.385]# RECV ASCII>

AT+MCONNECT=1,120

OK

[2021-05-21 21:27:02.527]# RECV ASCII>

CONNACK OK
```

一型一密 AT 方式连接

一型一密与一机一密的主要区别在于第一次连接的时候需要根据协议请求秘钥。首先需要在阿里云打开动态注册开关如下图所示。



用到的参数如下所示。

```
{

    "ProductSecret": "nNmPacH5aL8EViD4",

    "ProductKey": "a1qsHBbKtmc"

    "DeviceName" "866714044533668"

}--random 为随机数，长度不定

productKey=a1qsHBbKtmc&deviceName=866714044533668&random=123456&sign=073
074d8b722eacb1f6d2054d99e9d64&signMethod=HmacMD5

e10adc3949ba59abbe56e057f20f883e
```

```
明文：deviceName866714044533668productKeya1qsHBbKtmcrandom123456

秘钥：nNmPacH5aL8EViD4

HmacMD5 计算结果：073074d8b722eacb1f6d2054d99e9d64

d8d847e2558c6382fc35d2c91f626501

a1qsHBbKtmc

nNmPacH5aL8EViD4

ProductKey = b1KCi45LcCP

ProductSecret = VWll9fiYWKiwraBk

DeviceName = 862991419835241

productKey=a1qsHBbKtmc&deviceName=866714044533668&random=123456&sign=073
074d8b722eacb1f6d2054d99e9d64&signMethod=HmacMD5

deviceName866714044533668productKeya1qsHBbKtmcrandom123456

nNmPacH5aL8EViD4073074d8b722eacb1f6d2054d99e9d64
```

当设备属于公共实例，使用一型一密预注册认证方式时，动态注册参数如下：

```
mqttClientId:
clientId+"|securemode=2,authType=xxxx,random=xxxx,signmethod=xxxx|"

mqttUserName: deviceName+"&"+productKey

mqttPassword: sign_hmac(productSecret,content)
```

AT 交互流程如下

```
[21:37:43.379]发→◇AT+HTTPINIT

□

[21:37:43.383]收←◆AT+HTTPINIT



OK
```

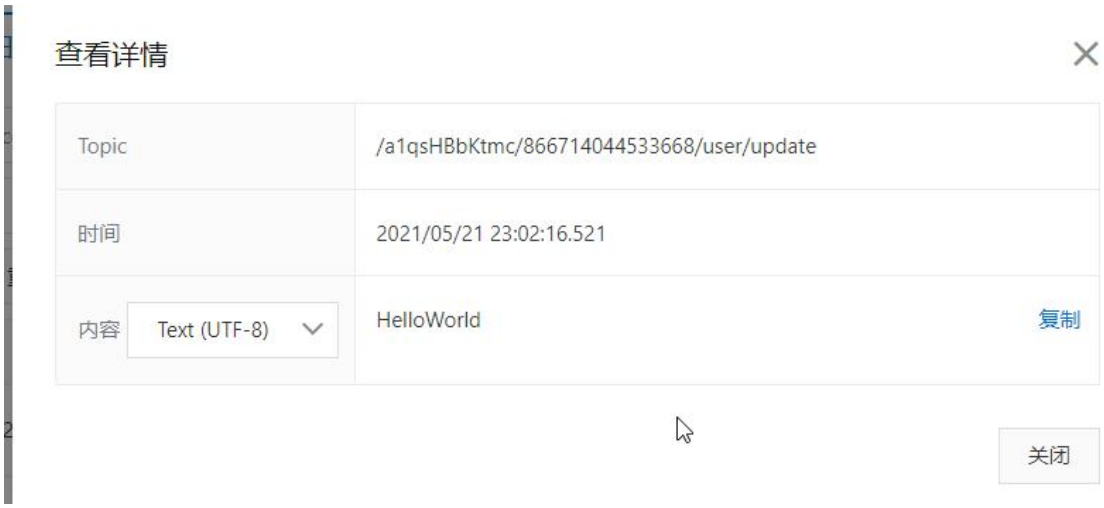
```
[21:37:48.704]发
→◇AT+HTTPPARA="URL","https://iot-auth.cn-shanghai.aliyuncs.com/auth/register/device"

□

[21:37:48.713]收
←◆AT+HTTPPARA="URL","https://iot-auth.cn-shanghai.aliyuncs.com/auth/register/device"

OK

[21:37:55.164]发→◇AT+HTTPPARA="USER_DEFINED","Content-Type: application/x-www-form-urlencoded"

□

[21:37:55.170]收←◆AT+HTTPPARA="USER_DEFINED","Content-Type: application/x-www-form-urlencoded"

OK

[21:38:00.207]发→◇AT+HTTPDATA=120,20000

□

[21:38:00.215]收←◆AT+HTTPDATA=120,20000

DOWNLOAD

[21:38:07.980]发
→◇productKey=a1qsHBbKtmc&deviceName=867435054361274&random=123456&sign=f25999d2e8ff8f5854b95d0ea9e0cd0c&signMethod=HmacMD5

□

[21:38:07.989]收←◆

OK

[21:38:17.459]发→◇AT+HTTPACTION=1
```

```
□

[21:38:17.467]收←◆AT+HTTPACTION=1

OK

[21:38:18.931]收←◆

+HTTPACTION: 1,200,149

[21:38:23.636]发→◇AT+HTTPREAD

□

[21:38:23.645]收←◆AT+HTTPREAD

+HTTPREAD: 149

{"code":200,"data":{"deviceName":"867435054361274","deviceSecret":"82cdb125154973a7c9fac52be5b757c2","productKey":"a1qsHBbKtmc"},"message":"success"}

OK
```

发布与订阅消息

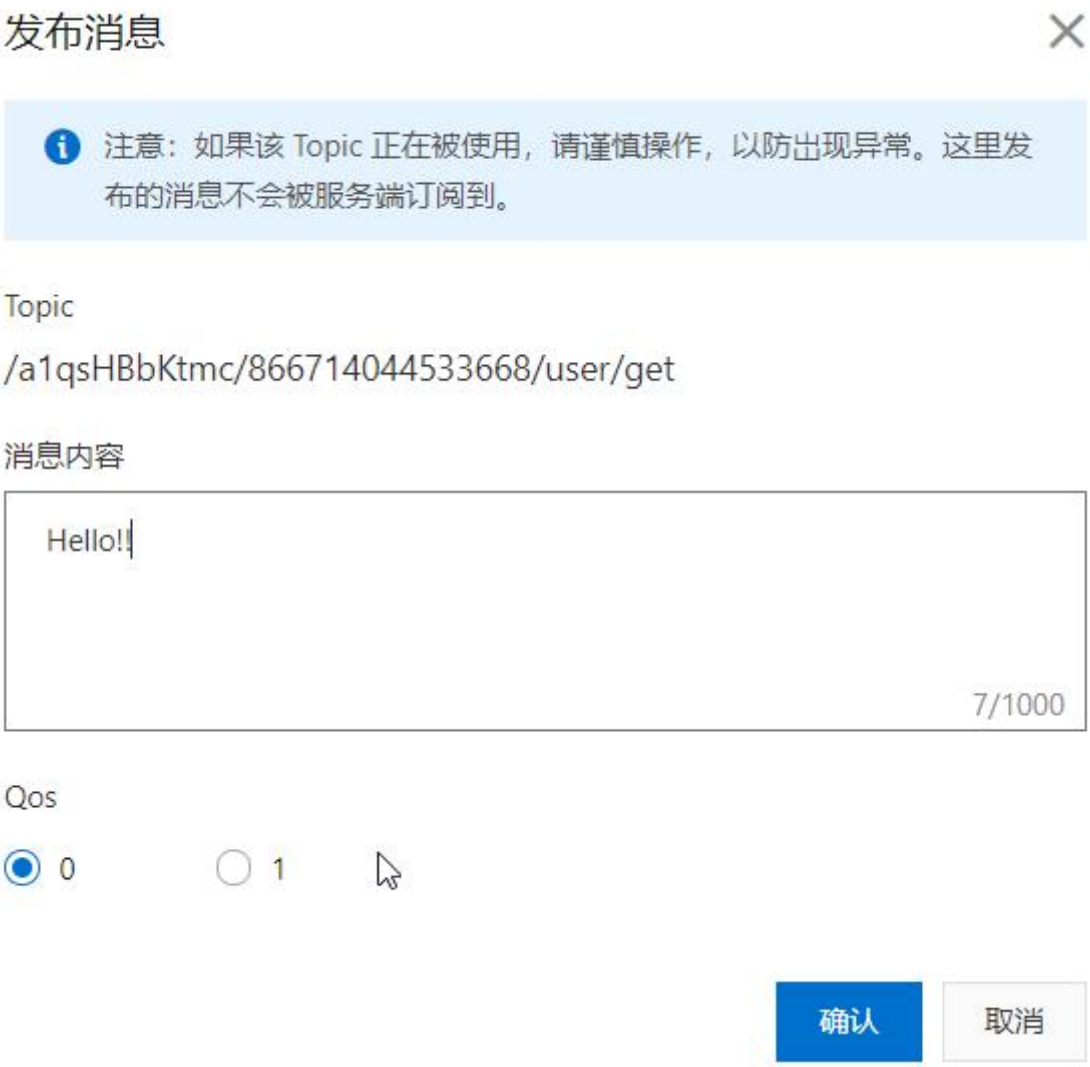
发布

```
[2021-05-21 23:02:04.961]# RECV ASCII>

CONNACK OK

[2021-05-21 23:02:16.503]# SEND ASCII>

AT+MPUB="/a1qsHBbKtmc/866714044533668/user/update",0,0,"HelloWorld"

[2021-05-21 23:02:16.554]# RECV ASCII>

AT+MPUB="/a1qsHBbKtmc/866714044533668/user/update",0,0,"HelloWorld"

OK
```



订阅

```
[2021-05-21 23:04:59.487]# SEND ASCII>

AT+MSUB="/a1qsHBbKtmc/866714044533668/user/get",0

[2021-05-21 23:04:59.603]# RECV ASCII>

AT+MSUB="/a1qsHBbKtmc/866714044533668/user/get",0

OK

[2021-05-21 23:04:59.725]# RECV ASCII>

SUBACK

[2021-05-21 23:05:16.759]# SEND ASCII>

AT+MQTTMSGSET=0

[2021-05-21 23:05:16.808]# RECV ASCII>

AT+MQTTMSGSET=0

OK

[2021-05-21 23:05:41.965]# RECV ASCII>

+MSUB: "/a1qsHBbKtmc/866714044533668/user/get",7 byte,Hello!!
```



9.5.10 ONENET（中国移动物联网平台）应用指南

应用概述

使用 AT 方式通过 MQTT 协议连接 onenet studio。

整个连接 onenet studio 的过程分为四个步骤：

1. 训练波特率；
2. 激活 PDP 参考；
3. 连接 onenet；

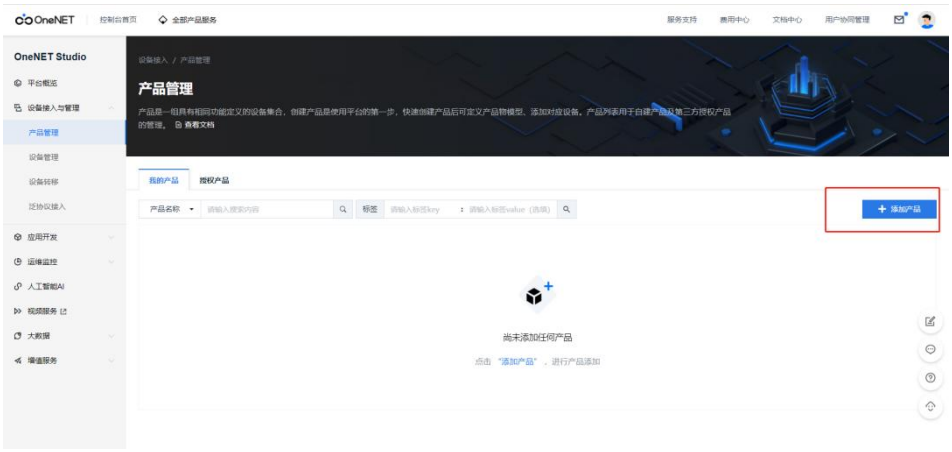
4.发布与订阅消息；

材料准备

1、 Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。

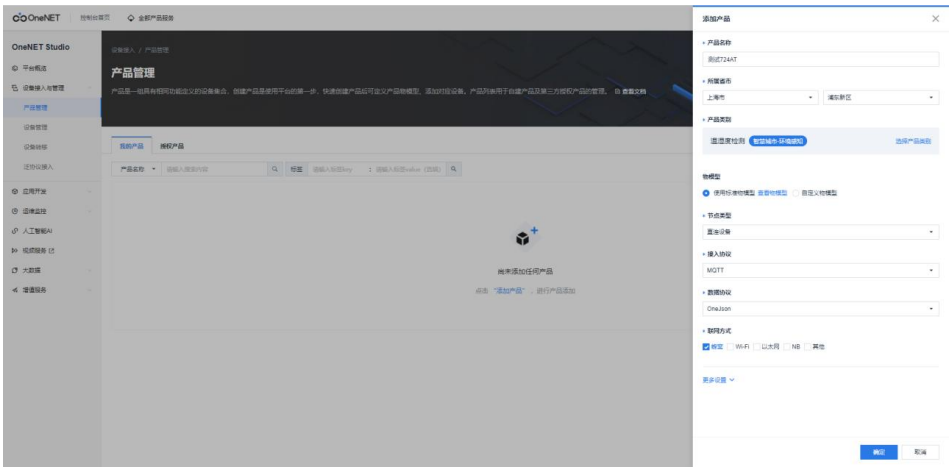
2、 PC 电脑，串口工具。

3、 在 onenet 上创建产品



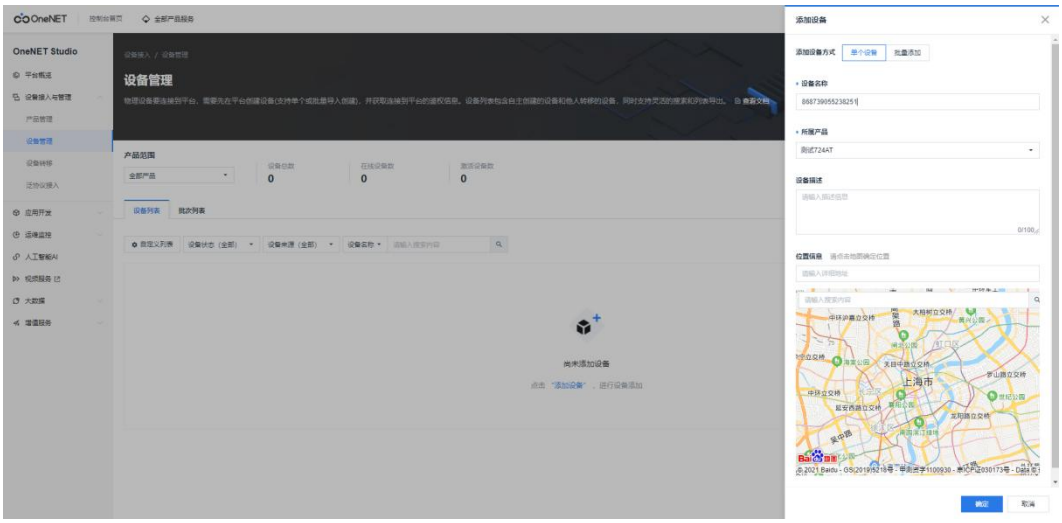
4、 产品名称、所属省市、产品类别根据自己需求填写，物模型没有要求可以选择标准物模型

5、 节点类型选择直连设备、接入协议选择 MQTT、数据协议固定为 OneJson、联网方式选择蜂窝



6、创建设备

填入设备名，选择产品，这里我使用的是模块的 imei 来做设备名称



7、查看 onenet 接入协议

MQTT 设备连接_开发者文档_OneNET (10086.cn))

1. 服务地址

设备接入支持标准MQTTV3.1.1版本，支持TLS加密，接入服务地址如下：
各省二级平台MQTT接入服务地址详见：省平台接入地址详情页

连接协议	证书	地址
MQTT	—	mqtts.heclouds.com:1883
MQTTS	证书下载	mqttstls.heclouds.com:8883

2. 产品与设备创建

按照创建产品与创建设备介绍，创建MQTT产品，并创建设备

其中：

产品名称用户下具有**唯一性**

设备名称产品内具有**唯一性**，推荐采用设备sn、mac地址、IMEI等信息命名设备。

3. 设备安全认证

产品、设备创建时，平台为每类产品、每个设备均分配了唯一的 key，设备登录时需要使用通过key计算出的访问token 来进行访问安全认证，见接入安全认证

设备可通过MQTT connnect报文进行登录，connect报文中三要素填写方法如下：

参数	是否必须	参数说明
clientId	是	设备名称
username	是	平台分配的产品ID
password	是	填写经过 key 计算的 token

可以看到，登入 onenet 时，mqtt 的 clientID、username 和 password 都有指定

clientId 即为刚才创建的设备名称

username 为创建的产品 ID

password 是需要经过加密计算得来的鉴权 token

计算方法详见如下文章，这篇文章测试时直接使用 onenet 提供的测试工具来计算鉴权密码，鉴权计算测试工具下面连接中下载

接入安全认证_开发者文档_OneNET (10086.cn)

连接 onenet

训练波特率

- 开机循环向串口 1 发送 AT，训练波特率，直到模块回复 ok

```
[2021/08/11 19:08:36.7975] ← AT
```

```
[2021/08/11 19:08:37.3333] ← AT

[2021/08/11 19:08:37.4643] → AT

OK

[2021/08/11 19:08:38.0495] ← AT

[2021/08/11 19:08:38.1822] → AT

OK

[2021/08/11 19:08:39.7062] ← AT

[2021/08/11 19:08:39.8116] → AT

OK
```

查询卡、网络注册状态、激活 PDP

具体交互流程如下所示

```
查询 SIM 卡状态

[2021/08/11 19:11:00.5642] ← AT+CPIN?

[2021/08/11 19:11:00.6723] → AT+CPIN?

+CPIN: READY

OK

查询信号质量

[2021/08/11 19:11:09.8339] ← AT+CSQ

[2021/08/11 19:11:09.9396] → AT+CSQ

+CSQ: 31,99

OK

查询网络附着状态

[2021/08/11 19:11:13.4545] ← AT+CGATT?
```

```
[2021/08/11 19:11:13.5635] → AT+CGATT?

+CGATT: 1

OK

激活 PDP

[2021/08/11 19:12:12.8205] ← AT+CSTT

[2021/08/11 19:12:12.9255] → AT+CSTT

OK

[2021/08/11 19:12:16.2912] ← AT+CIICR

[2021/08/11 19:12:16.3950] → AT+CIICR

OK

[2021/08/11 19:12:18.9946] ← AT+CIFSR

[2021/08/11 19:12:19.1043] → AT+CIFSR10.81.141.189
```

MQTT 直连 onenet

设备注册参数如下：

```
mqttClientId: devicename

mqttUsername: productID

mqttPassword: token
```

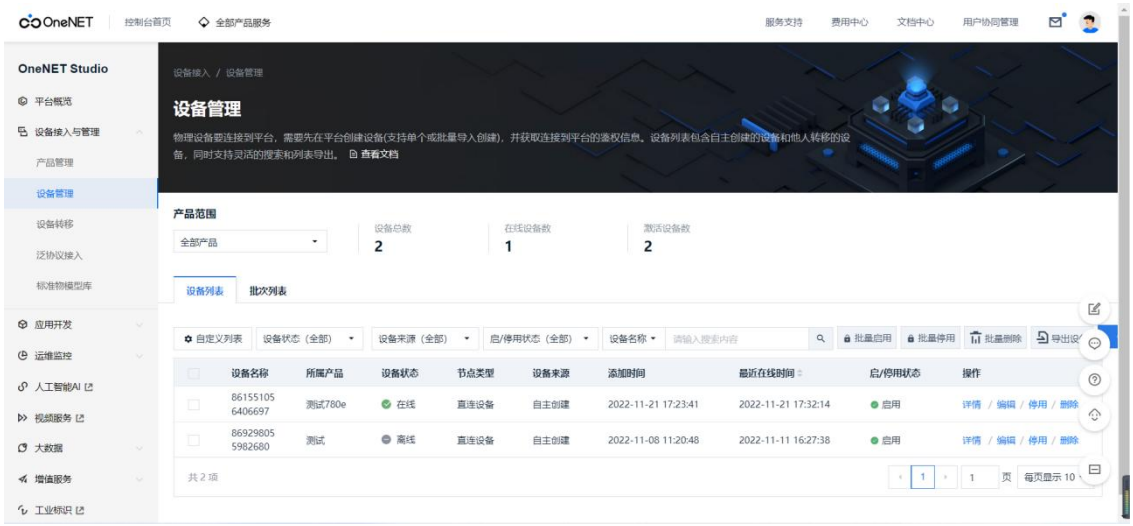
参数	说明
devicename	设备名称
productID	平台分配的产品 id
token	经过鉴权算法得到的 token
具体交互流程如下所示	

下图为工具计算出的 token

```
[2021/08/12 11:11:06.3625] ←
AT+MCONFIG="868739055238251","q23GT8XVOu","version=2018-10-31&res=products%2Fq23GT8XVOu%2Fdevices%2F868739055238251&et=1628739000&method=md5&sign=VmqFxCV3SeVlJITQTgV22A%3D%3D"

[2021/08/12 11:11:06.4655] →
AT+MCONFIG="868739055238251","q23GT8XVOu","version=2018-10-31&res=products%2Fq23GT8XVOu%2Fdevices%2F868739055238251&et=1628739000&method=md5&sign=VmqFxCV3SeVlJITQTgV22A%3D%3D"

OK

[2021/08/12 11:11:30.8763] ← AT+MIPSTART="studio-mqtt.heclouds.com",1883

[2021/08/12 11:11:30.9803] → AT+MIPSTART="studio-mqtt.heclouds.com",1883

OK

[2021/08/12 11:11:31.2635] →

CONNECT OK

[2021/08/12 11:11:39.0602] ← AT+MCONNECT=1,120

[2021/08/12 11:11:39.1625] → AT+MCONNECT=1,120

OK
```

```
[2021/08/12 11:11:39.3007] →

CONNACK OK
```

可以从平台上看到设备已经在线了



发布与订阅消息

onenet studio 有固定的通信 TOPIC，可以到官网查看，这里演示下设备属性上报与响应的主题

通信主题_开发者文档_OneNET (10086.cn)

Topic列表

物联网平台预定义物模型通信Topic，各物模型功能Topic消息的数据格式，见OneJSON数据格式。 Topic类以正斜线（/）进行分层，区分每个类目。其中，有两个类目为既定类目：${pid}表示产品的产品id；${device-name}表示设备名称；${identifier}表示服务标识符

功能	类别	行为	描述	Topic类	操作权限
物模型通信Topic	属性	设备属性上报	请求	$sys/{pid}/{device-name}/thing/property/post	发布
			响应	$sys/{pid}/{device-name}/thing/property/post/reply	订阅
		设备属性设置（同步）	请求	$sys/{pid}/{device-name}/thing/property/set	订阅
			响应	$sys/{pid}/{device-name}/thing/property/set_reply	发布
		设备获取属性期望值	请求	$sys/{pid}/{device-name}/thing/property/desired/get	发布
			响应	$sys/{pid}/{device-name}/thing/property/desired/get/reply	订阅
		清除属性期望值	请求	$sys/{pid}/{device-name}/thing/property/desired/delete	发布
			响应	$sys/{pid}/{device-name}/thing/property/desired/delete/reply	订阅
		设备属性获取	请求	$sys/{pid}/{device-name}/thing/property/get	订阅
			响应	$sys/{pid}/{device-name}/thing/property/get_reply	发布
	事件	设备事件上报	请求	$sys/{pid}/{device-name}/thing/event/post	发布
			响应	$sys/{pid}/{device-name}/thing/event/post/reply	订阅
	服务	设备服务调用	请求	$sys/{pid}/{device-name}/thing/service/{identifier}/invoke	订阅
			响应	$sys/{pid}/{device-name}/thing/service/{identifier}/invoke_reply	发布

订阅

```
[2021/08/12 11:19:48.2274] ←
AT+MSUB="$sys/q23GT8XVOu/868739055238251/thing/property/post/reply",0

[2021/08/12 11:19:48.3469] →
AT+MSUB="$sys/q23GT8XVOu/868739055238251/thing/property/post/reply",0

OK

[2021/08/12 11:19:48.4718] →

SUBACK
```

发布

向设备上报属性的主题发布消息，消息携带设备的属性，不同的产品具有不同的属性，可以在设备物模型处查看

例如：在文章开始创建的产品具有如下图所示的属性

温度、湿度和地理位置

向上报属性的主题发布一条 payload 格式错误的消息

```
[2021/08/12 11:22:39.8677] ←
AT+MPUB="$sys/q23GT8XVOu/868739055238251/thing/property/post",0,0,"{}"

[2021/08/12 11:22:39.9918] →
AT+MPUB="$sys/q23GT8XVOu/868739055238251/thing/property/post",0,0,"{}"

OK
```

平台返回缺少参数

```
[2021/08/12 11:22:40.2127] →
```

```
+MSUB: "$sys/q23GT8XVOu/868739055238251/thing/property/post/reply",47
byte,{"id":null,"code":2403,"msg":"required msg id"}
```

如下是正常流程

设备上报温度数据：22°C

```
[2021/08/12 11:32:05.2056] ←
AT+MPUB="$sys/q23GT8XVOu/868739055238251/thing/property/post",0,0,"{\22i
d\22:\22123\22,\22params\22:{\22CurrentTemperature\22:{\22value\22:22.0}}
}"

[2021/08/12 11:32:05.3090] →
AT+MPUB="$sys/q23GT8XVOu/868739055238251/thing/property/post",0,0,"{\22i
d\22:\22123\22,\22params\22:{\22CurrentTemperature\22:{\22value\22:22.0}}
}"

OK
```

平台返回成功

```
[2021/08/12 11:32:05.5342] →

+MSUB: "$sys/q23GT8XVOu/868739055238251/thing/property/post/reply",39
byte,{"id":"123","code":200,"msg":"success"}
```

可以看到平台已经有数据更新

9.5.11 CTWing（中国电信物联网平台）应用指南

应用概述

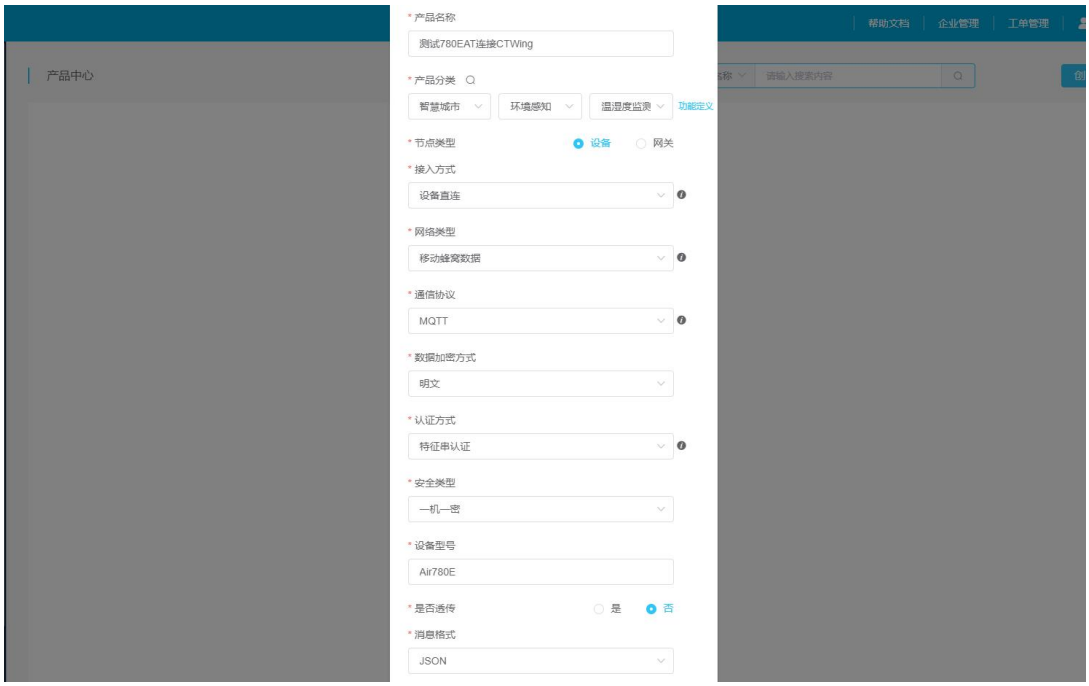
测试使用 AT 方式控制模块通过 MQTT 协议连接 CTWing 物联网平台

材料准备

1、 Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。

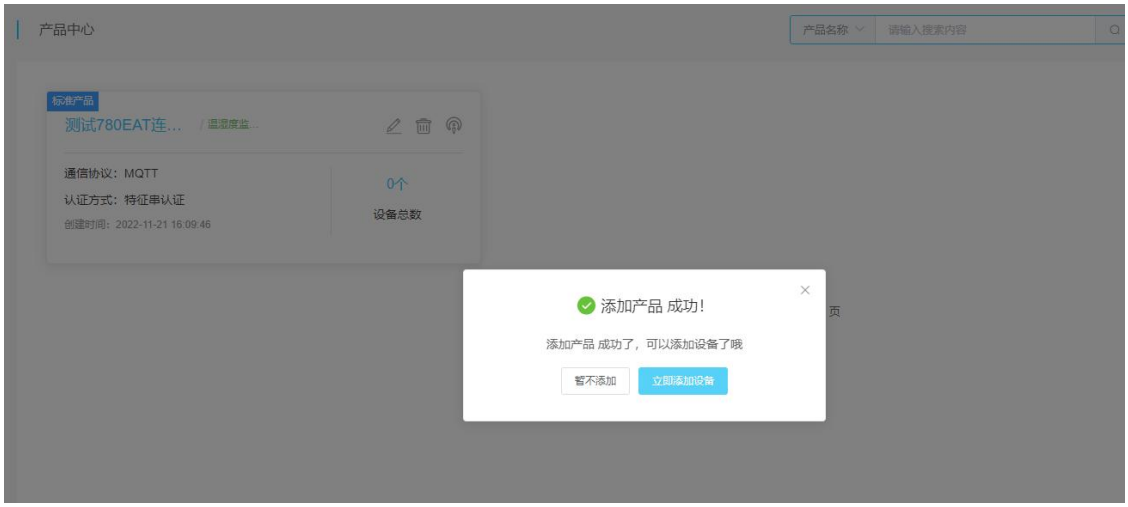
2、PC 电脑，串口工具

3、在 CTWing 上创建产品



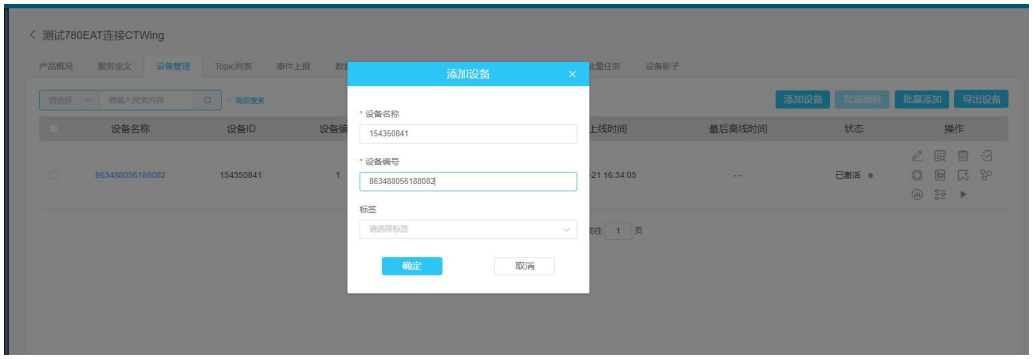
产品名称、产品分类、设备型号根据自己需求填写

节点类型选择设备、接入方式选择设备直连、网络类型选择移动蜂窝数据、通信协议选择 MQTT、数据加密方式选择明文、认证方式选择特征串认证、安全类型选择一机一密、是否透传选择否、消息格式固定为 JSON



4、创建设备

填入设备名，设备编号，这里是以模组 imei 为设备名



创建成功

查看 CTWing 接入协议

业务数据开发 · 开发文档_文档中心-中国电信通用使能平台（ctwing.cn)

MQTT 登录时的参数都有指定，三个参数分别在如下几个位置找到

ClientId 为设备 ID

Username 为平台用户名

Password 为设备特征串

ClientId：在设备列表内设备 ID 一栏

Username：为 AEP 控制台登录账户的名字

Pssword：特征串在设备详情页内获取

连接 CTWing

- 开机循环向串口 1 发送 AT，训练波特率，直到模块回复 ok

```
[16:28:59.979]发→◇AT

[16:28:59.981]收←◆AT

OK

[16:29:00.463]发→◇AT

[16:29:00.465]收←◆AT

OK

[16:29:01.805]发→◇AT

[16:29:01.806]收←◆AT

OK
```

- 查询卡、网络注册状态、激活 PDP

  具体交互流程如下所示

```
查询 SIM 卡状态

[16:29:12.582]发→◇AT+CPIN?

[16:29:12.588]收←◆AT+CPIN?
```

```
+CPIN: READY

OK
```

查询信号质量

```
[16:29:18.363]发→◇AT+CSQ

[16:29:18.367]收←◆AT+CSQ

+CSQ: 23,0

OK
```

查询网络附着状态

```
[16:29:30.256]发→◇AT+CGATT?

[16:29:30.262]收←◆AT+CGATT?

+CGATT: 1

OK
```

激活 PDP

```
[16:29:36.314]发→◇AT+CSTT

[16:29:36.319]收←◆AT+CSTT

OK

[16:29:42.572]发→◇AT+CIICR

[16:29:42.578]收←◆AT+CIICR

OK

[16:29:46.432]发→◇AT+CIFSR

[16:29:46.436]收←◆AT+CIFSR10.221.203.159
```

MQTT 直连 CTWing

接口介绍

通信协议	地址	端口	说明
MQTT	mqtt.ctwing.cn	1883	TCP非加密接口
MQTTS	mqtt.ctwing.cn	8883	TLS加密接口

```
[16:33:18.384]发
→◇AT+MCONFIG="154350841","18338431242","_9eiuFm7fpCV9Kz50bVhmyWmBYugSoSXMqyp1j4TnM0"

[16:33:18.393]收
←◆AT+MCONFIG="154350841","18338431242","_9eiuFm7fpCV9Kz50bVhmyWmBYugSoSXMqyp1j4TnM0"

OK

[16:33:30.010]发→◇AT+MIPSTART="mqtt.ctwing.cn","1883"

[16:33:30.014]收←◆AT+MIPSTART="mqtt.ctwing.cn","1883"

OK

[16:33:30.316]收←◆

CONNECT OK

[16:33:37.663]发→◇AT+MCONNECT=1,120

[16:33:37.668]收←◆AT+MCONNECT=1,120

OK

[16:33:37.837]收←◆

CONNACK OK
```

可以从平台上看到设备已经在线了

发布与订阅消息

CTWing 有固定的通信 TOPIC，在产品详情页 TOPIC 列表内，这里演示下设备属性上报与平台指令下发的主题

属性定义在产品详情页的服务定义一栏内获取

订阅

订阅设置湿度过高告警使能的 topic，平台向 set_high_humidity_alarm_enable 主题下发指令

```
[16:37:00.001]发→◇AT+MSUB="set_high_humidity_alarm_enable",0
```

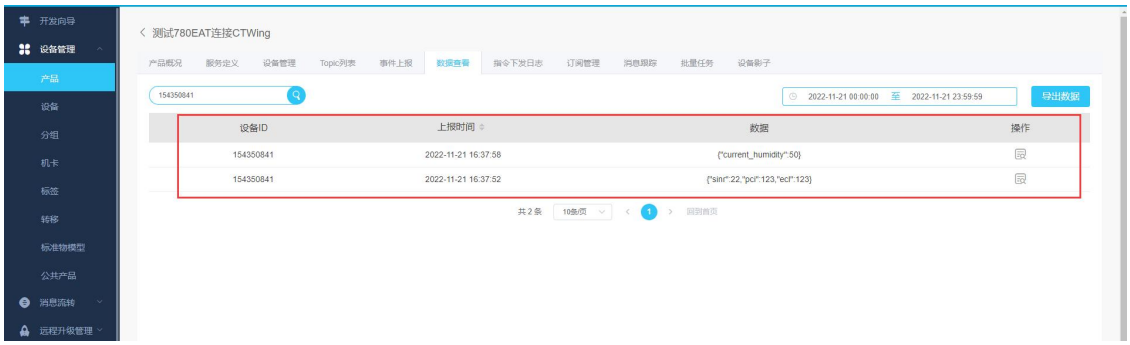
```
[16:37:00.009]收←◆AT+MSUB="set_high_humidity_alarm_enable",0

OK

[16:37:00.145]收←◆

SUBACK

[16:36:49.279]收←◆

+MSUB: "set_high_humidity_alarm_enable",55
byte,{"taskId":1,"payload":{"high_humidity_alarm_enable":0}}
```

发布

向设备上报属性的主题发布消息，消息携带设备的属性，不同的产品具有不同的属性，可以在设备物模型处查看

```
[16:37:24.991]发
→◇AT+MPUB="signal_report",0,0,"{\22sinr\22:22,\22pci\22:123,\22ecl\22:1
23}"

[16:37:24.996]收
←◆AT+MPUB="signal_report",0,0,"{\22sinr\22:22,\22pci\22:123,\22ecl\22:1
23}"

OK

[16:37:31.440]发
→◇AT+MPUB="data_report",0,0,"{\22current_humidity\22:50}"

[16:37:31.448]收
←◆AT+MPUB="data_report",0,0,"{\22current_humidity\22:50}"

OK
```

在设备列表的设备详情页、数据查看一栏内查看上报的数据

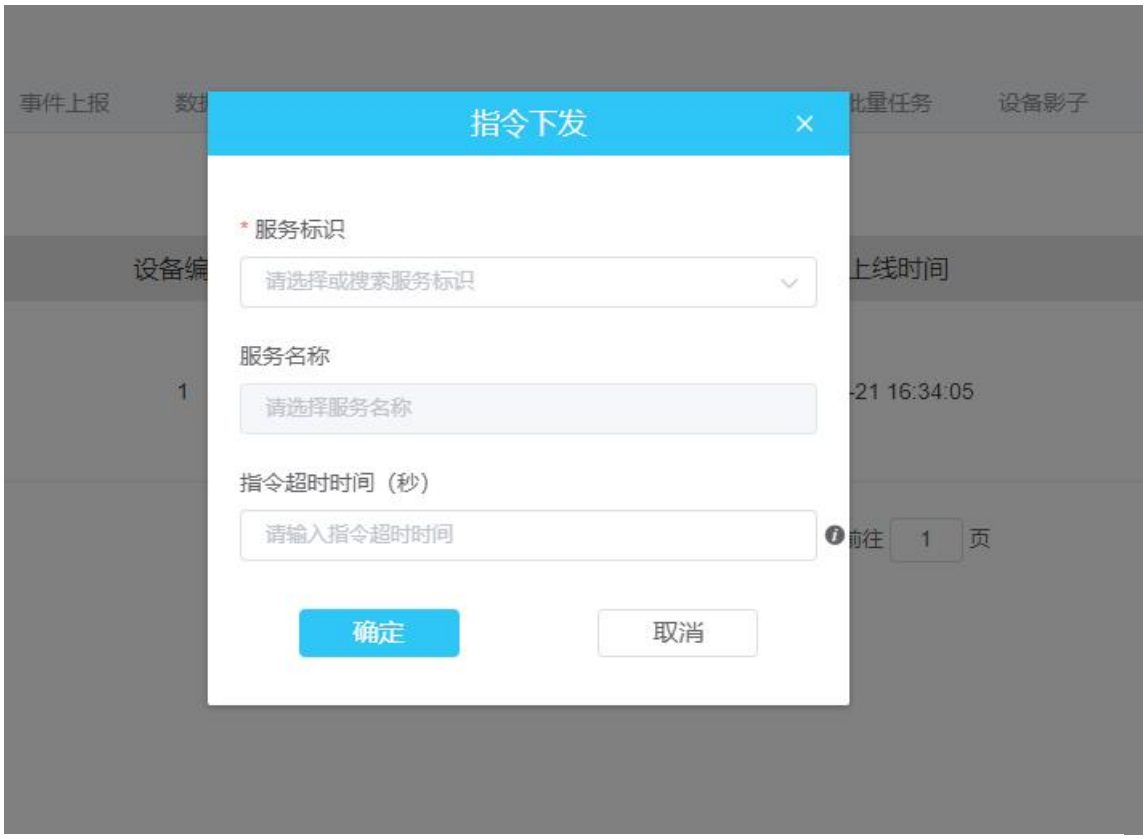


平台指令下发

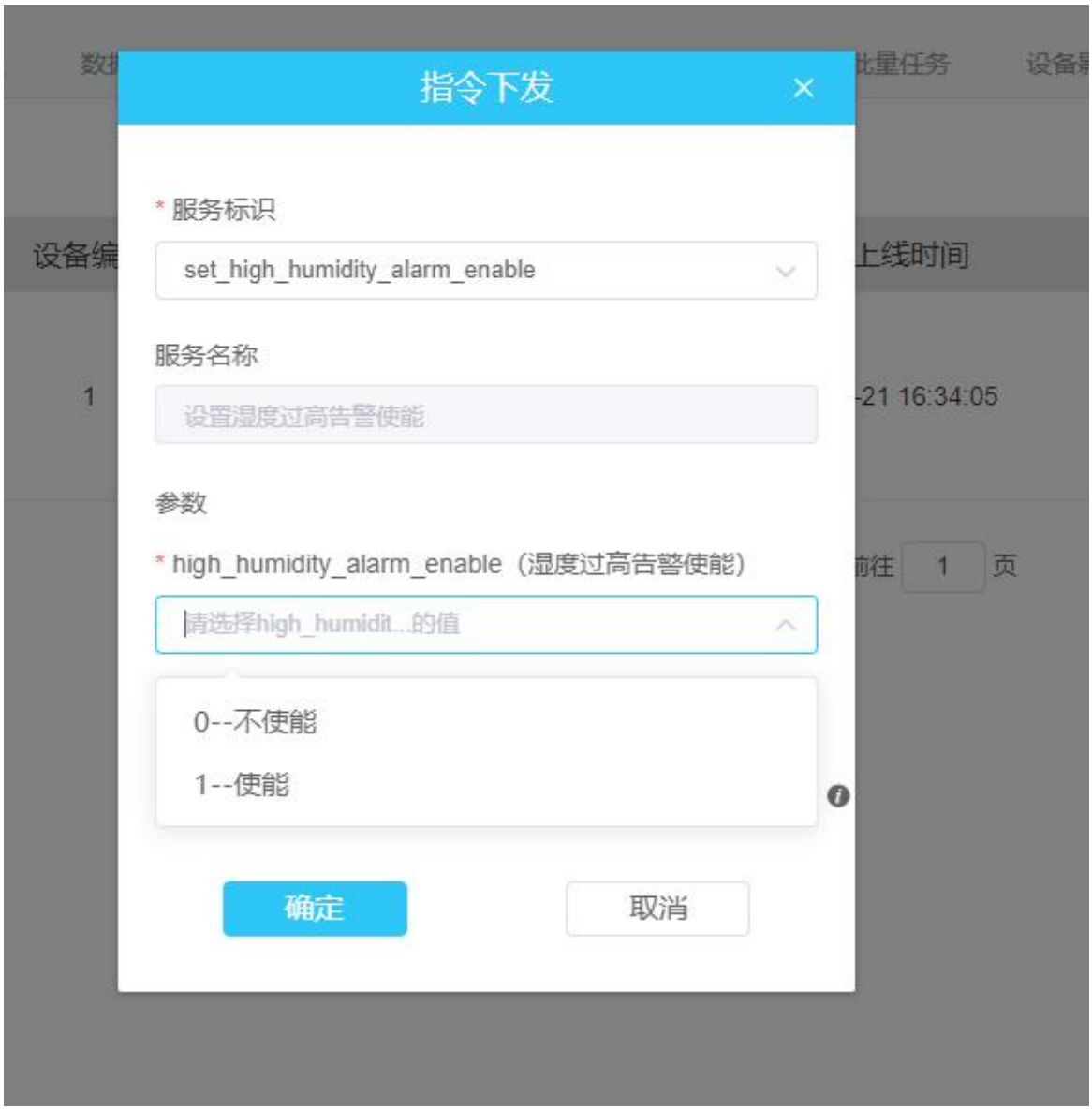
点击产品一栏，选择设备管理

在操作一栏点击指令下发

点击指令下发以后，会出现一个指令下发的界面



例如：我们选择又给设置湿度过高告警使能



点击确定后，即可从串口工具看到平台指令下发给模组的数据

```
[10:19:17.147]→

+MSUB: "set_high_humidity_alarm_enable",56
byte,{"taskId":15,"payload":{"high_humidity_alarm_enable":0}}
```

常见问题

1. 一型一密的方式该怎么连接

一型一密连接参数使用的密码（特征串）是同一个，只需要改变设备 ID 建立连接。

```
[09:47:33.895]发→◇AT+CIFSR
□
[09:47:33.898]收←◆AT+CIFSR

10.78.3.90

[09:48:05.026]发→◇AT+MCONFIG="166804515_200002","Lucario","Wv4bSKeRKyzclM0bs_svcBp5ZL8fVH9dAFIvg5lg_cI"
□
[09:48:05.037]收←◆AT+MCONFIG="166804515_200002","Lucario","Wv4bSKeRKyzclM0bs_svcBp5ZL8fVH9dAFIvg5lg_cI"

OK

[09:48:16.212]发→◇AT+MIPSTART="mqtt.ctwing.cn","1883"
□
[09:48:16.216]收←◆AT+MIPSTART="mqtt.ctwing.cn","1883"

OK

[09:48:16.426]收←◆
CONNECT OK

[09:48:17.423]发→◇AT+MCONNECT=1,120
□
[09:48:17.434]收←◆AT+MCONNECT=1,120

OK

[09:48:17.528]收←◆
CONNACK OK

[09:49:03.128]发→◇AT+MDISCONNECT
□
[09:49:03.131]收←◆AT+MDISCONNECT

OK

[09:49:18.272]发→◇AT+MCONFIG="166804515_200001","Lucario","Wv4bSKeRKyzclM0bs_svcBp5ZL8fVH9dAFIvg5lg_cI"
□
[09:49:18.276]收←◆AT+MCONFIG="166804515_200001","Lucario","Wv4bSKeRKyzclM0bs_svcBp5ZL8fVH9dAFIvg5lg_cI"   同一个特征串
                                只改个设备ID就可以了
OK

[09:49:19.603]发→◇AT+MIPSTART="mqtt.ctwing.cn","1883"
□
```

9.5.12 DMP（中国联通雁飞格物物联网平台）应用指南

应用概述

测试使用 AT 方式控制模块通过 MQTT 协议连接联通雁飞格物物联网平台

材料准备

1、 Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。

2、PC 电脑，串口工具

3、在雁飞格物平台上创建产品

产品名称根据自己需求填写

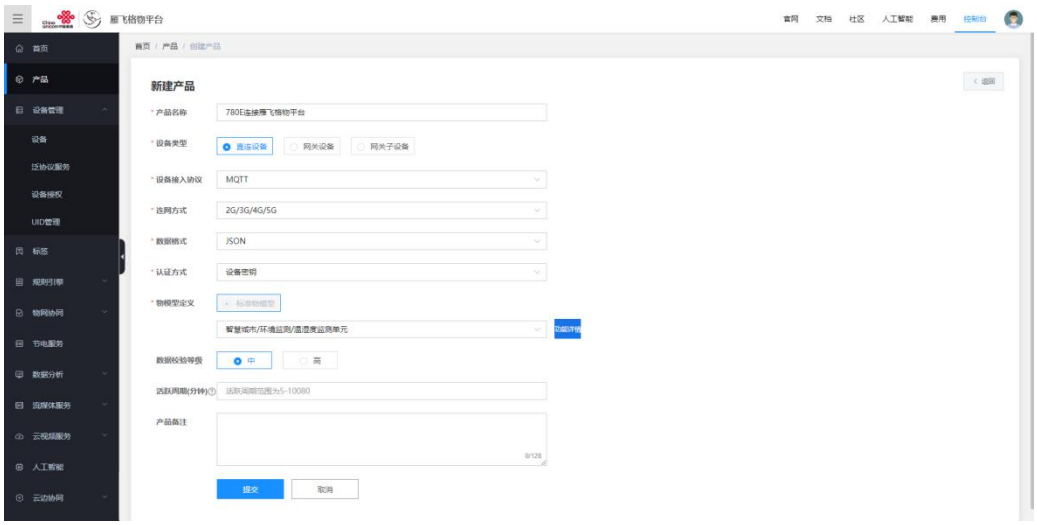
设备类型：直连设备

设备接入协议：MQTT

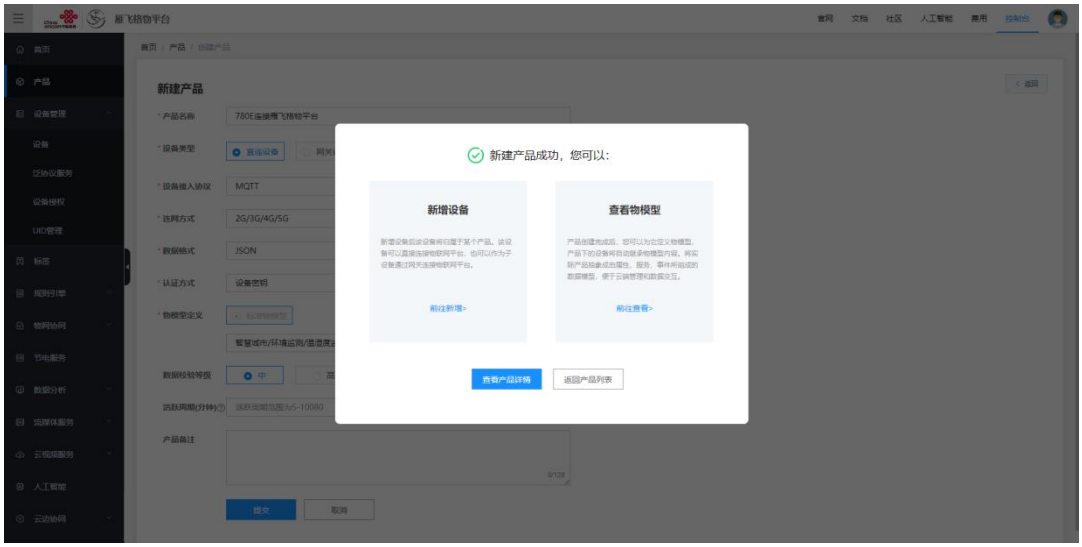
联网方式：2G/3G/4G/5G

认证方式：设备密钥

物模型定义根据自己需求填写、这里由于演示需要选择温湿度监测单元



点击提交即可



4、创建设备

点到设备一栏，新建设备

产品一栏选择上一步创建的产品，设备名称、deviceKey、IMEI 根据自身需要填写

注意：！！！如果 deviceKey 为空，则联通物联网平台会自动生成

点击提交后，设备即可创建成功

查看雁飞格物平台接入协议

文档展示 - 雁飞格物平台 (cuiot.cn)

MQTT 协议规定连接参数须有 ClientID,Username 和 Password 三个字段，雁飞格物平台规定的三元组格式详细信息可在接入文档中查看

ClientId：{deviceId}|{productKey}|{signMethod}|{authType}|{operator}

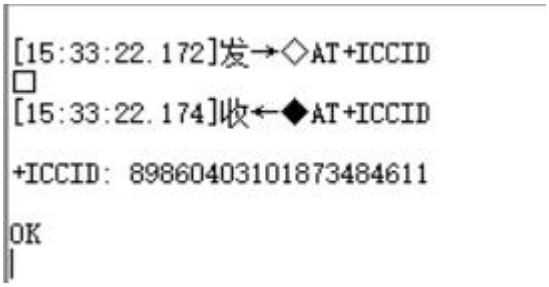
Username：{deviceKey}|{productKey}

Password：将 deviceId， deviceKey，productKey 参数值依次进行拼接再进行 hmac_sha256 或 sm3 得到组成 ClientId

operator：由协议可见，当联网方式为蜂窝网时，operator 字段表示运营商标识符，这里测试用的是移动卡，operator 字段则为 2

deviceId：deviceId 需要和 operator 搭配使用，operator 为 1、2、3、4 任意一个时，deviceId 为流量卡的 iccid

我们发 AT+ICCID 查一下当前设备流量卡的 iccid，作为 deviceId 来使用



authType：一机一密填 0 即可

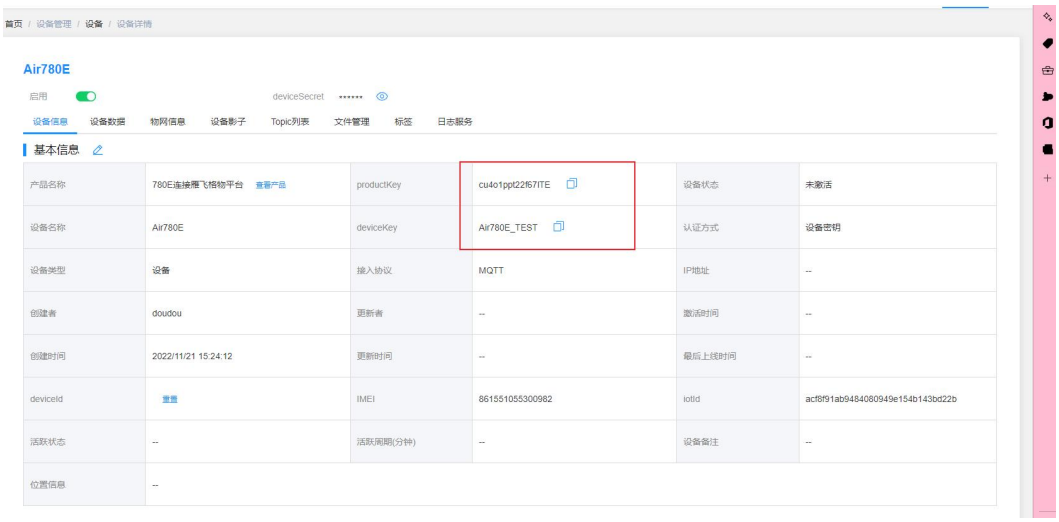
signMethod：此次示例使用 hmac_sha256，写 0 即表示接入验证用的是 hmac_sha256

productKey：在产品列表即可找到

到此，clientId 已经拼接好了，拼接后的 clientId 如下

89860403101873484611|cu4o1ppt22f67ITE|0|0|2

组成 userName



在设备详细信息内找到 deviceKey 和 productKey，拼接在一起即可

userName 组成如下

username：Air780E_TEST|cu4o1ppt22f67ITE

组成 password

如文档所说，将 deviceId，deviceKey 和 productKey 拼接再进行 hmac_sha256 加密即可得到 password

加密密钥为设备的 deviceSecre，在设备详情页中可以找到

加密之前 deviceId，deviceKey、productKey 拼接的字符串为：
89860403101873484611Air780E_TESTcu4o1ppt22f67ITE

密钥为 307083897582E2465702D0351A20BFC1

加密后得到的 password 为
678b5ef1138ad614a2e42852d80cb3db94ec4a6828af6c612888411ceacbc9c1

连接雁飞格物平台

- 开机向串口 1 发送 AT，模块回复 ok

```
[2022/11/21 15:56:02.7040] ← AT

[2022/11/21 15:56:02.8337] → AT

OK
```

- 查询卡、网络注册状态、激活 PDP

  具体交互流程如下所示

```
查询 SIM 卡状态

[2022/11/21 15:56:46.3035] ← AT+CPIN?

[2022/11/21 15:56:46.4275] → AT+CPIN?

+CPIN: READY

OK

查询信号质量

[2022/11/21 15:56:53.8953] ← AT+CSQ

[2022/11/21 15:56:54.0032] → AT+CSQ

+CSQ: 23,0

OK

查询网络附着状态

[2022/11/21 15:57:01.7109] ← AT+CGATT?

[2022/11/21 15:57:01.8166] → AT+CGATT?

+CGATT: 1

OK
```

```
激活 PDP

[2022/11/21 15:57:05.3596] ← AT+CSTT

[2022/11/21 15:57:05.4658] → AT+CSTT

OK

[2022/11/21 15:57:07.5059] ← AT+CIICR

[2022/11/21 15:57:07.6111] → AT+CIICR

OK

[2022/11/21 15:57:13.3857] ← AT+CIFSR

[2022/11/21 15:57:13.4899] → AT+CIFSR10.43.254.64
```

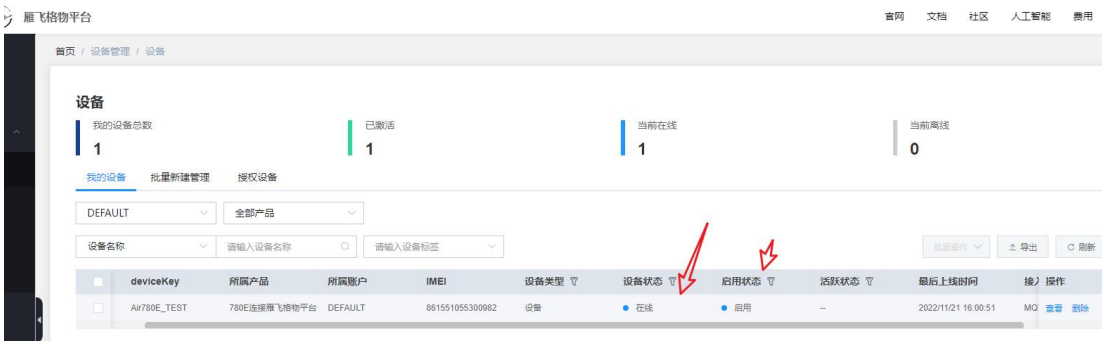
MQTT 直连雁飞格物平台

字段	说明
接入地址	平台MQTT接入域名为dmp-mqtt.cuiot.cn，端口为1883，设备若支持域名接入，请尽量使用域名作为接入地址。使用TLS加密接入域名为dmp-mqtts.cuiot.cn，端口号为8883。

```
[2022/11/21 15:59:47.4123] ←
AT+MCONFIG="898604031018734846l1|cu4o1ppt22f67lTE|0|0|2","Air780E_TEST|cu4o1ppt22f67lTE","678b5ef1138ad614a2e42852d80cb3db94ec4a6828af6c612888411ceacbc9c1"

[2022/11/21 15:59:47.5186] →
AT+MCONFIG="898604031018734846l1|cu4o1ppt22f67lTE|0|0|2","Air780E_TEST|cu4o1ppt22f67lTE","678b5ef1138ad614a2e42852d80cb3db94ec4a6828af6c612888411ceacbc9c1"

OK

[2022/11/21 16:00:48.7303] ← AT+MIPSTART="dmp-mqtt.cuiot.cn",1883

[2022/11/21 16:00:48.8542] → AT+MIPSTART="dmp-mqtt.cuiot.cn",1883

OK

[2022/11/21 16:00:49.1099] →
```

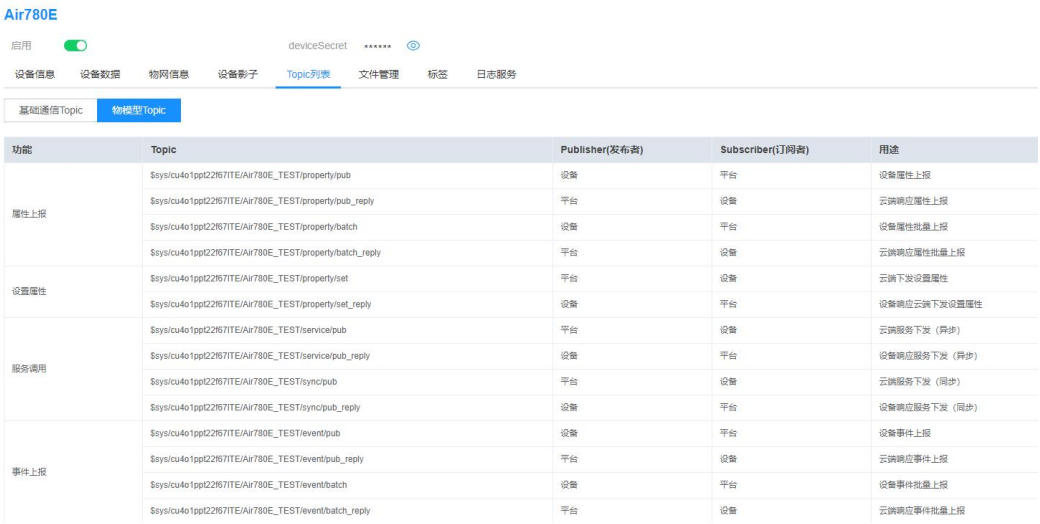
```
CONNECT OK

[2022/11/21 16:00:50.4232] ← AT+MCONNECT=1,60

[2022/11/21 16:00:50.5276] → AT+MCONNECT=1,60

OK

[2022/11/21 16:00:50.6951] →

CONNACK OK
```

返回 CONNACK OK 表示连接成功，可以从平台上看到设备已经在线了

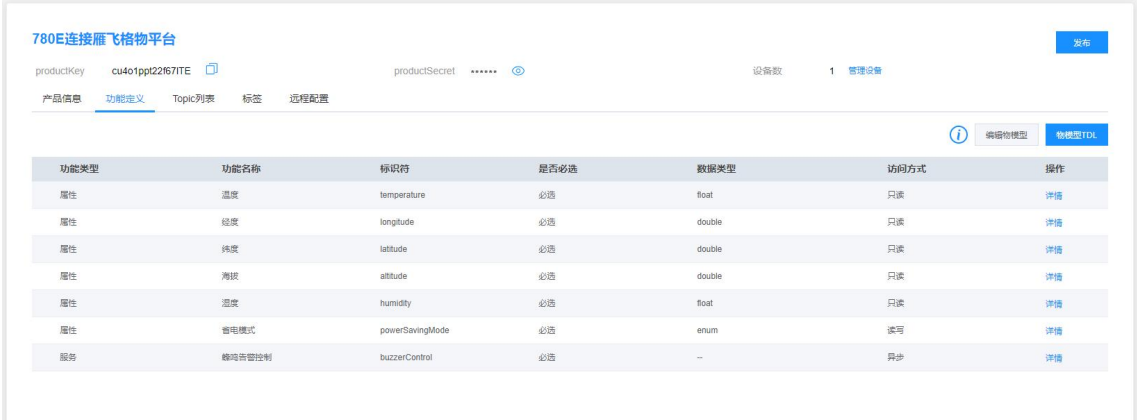


发布与订阅消息

雁飞格物平台有固定的通信 TOPIC，在设备详情页 TOPIC 列表内，这里演示设备上传物模型数据和订阅上传物模型响应功能



属性定义在产品详情页的功能定义一栏内获取



订阅

订阅云端响应属性上报主题

```
[2022/11/21 16:06:36.7815] ←

AT+MSUB="$sys/cu4o1ppt22f67lTE/Air780E_TEST/property/pub_reply",0


[2022/11/21 16:06:36.8878] →

AT+MSUB="$sys/cu4o1ppt22f67lTE/Air780E_TEST/property/pub_reply",0


OK


[2022/11/21 16:06:37.0950] →


SUBACK
```

发布

向设备上报属性的主题发布消息，消息携带设备的属性，不同的产品具有不同的属性，可以在设备物模型处查看

具体 payload 上传协议需要查看雁飞格物平台物模型文档 文档展示 - 雁飞格物平台 (cuiot.cn)

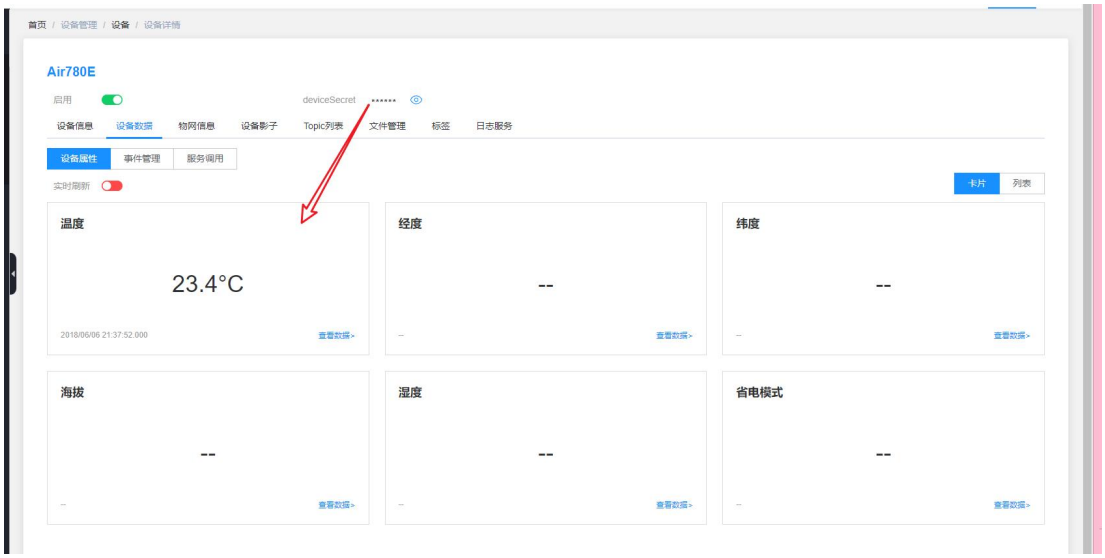
此处上报一条温度属性

```
[2022/11/21 16:12:24.6508] ←

AT+MPUB="$sys/cu4o1ppt22f67lTE/Air780E_TEST/property/pub",0,0,"{\22messa
```

```
geId\22:\22123\22,\22params\22:{\22key\22:\22temperature\22,\22value\22:
23.4,\22ts\22:\221528292272000\22}}"

[2022/11/21 16:12:24.7696] →
AT+MPUB="$sys/cu4o1ppt22f67lTE/Air780E_TEST/property/pub",0,0,"{\22messa
geId\22:\22123\22,\22params\22:{\22key\22:\22temperature\22,\22value\22:
23.4,\22ts\22:\221528292272000\22}}"

OK

[2022/11/21 16:12:25.0063] →

+MSUB: "$sys/cu4o1ppt22f67lTE/Air780E_TEST/property/pub_reply",55
byte,{"code":"000000","message":"Success","messageId":"123"}
```

平台返回 success，表示属性上报成功

在设备列表的设备详情页、设备数据一栏内查看上报的数据



9.5.13 短消息应用指南

应用概述

4G 模块发短信基本流程如下：

1、设置为 TEXT 模式

2、查询当前的 TEXT 模式短信参数

3、查询字符集

4、发短信给测试手机号

5、输入具体信息内容发送

材料准备

1、 Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。

2、PC 电脑，串口工具

应用示例

基本流程

先来看个正常情况下的 AT 命令序列，如果不理解 AT 命令含义，请自行参考 AT 手册。

```
[10:40:55.700]发→◇ATI//查询软件版本

□

[10:40:55.700]收←◆ATI

AirM2M_780E_V301836_LTE_AT
```

```
OK

[10:40:58.330]发→◇AT+CPIN?//查询 sim 卡状态

□

[10:40:58.330]收←◆AT+CPIN?

+CPIN: READY

OK

[10:40:59.128]发→◇AT+CSQ//查询信号质量

□

[10:40:59.133]收←◆AT+CSQ

+CSQ: 23,99

OK

[10:41:00.011]发→◇AT+CGATT?//查询附着状态

□

[10:41:00.015]收←◆AT+CGATT?

+CGATT: 1

OK

[10:42:10.046]发→◇AT+CMGF=1//设置为 TEXT 模式

□

[10:42:10.049]收←◆AT+CMGF=1

OK

[10:42:18.802]发→◇AT+CSMP?//查询当前的 TEXT 模式短信参数

□

[10:42:18.808]收←◆AT+CSMP?
```

```
+CSMP: 17,167,0,0

OK

[10:42:24.314]发→◇AT+CSCS?//查询字符集

□

[10:42:24.320]收←◆AT+CSCS?

+CSCS: "PCCP936"

OK

[10:43:10.883]发→◇AT+CMGS="1862*****86"//要发送的手机号。

□

[10:43:10.889]收←◆AT+CMGS="1862*****86"

>

[10:43:34.037]发→◇hello,world123! //输入要发送的具体内容

□

[10:43:34.043]收←◆hello,world123!

[10:43:47.623]发→◇□

[10:43:47.623]收←◆

[10:43:48.149]收←◆

+CMGS: 1

OK//ok 代表发送成功，手机上可以看到收到的信息
```

常见问题

1、PDU 格式发送短信问题

AT 版本发送短信时先查看自己的 SIM 卡是否设置中心地址，发送短信时发送 PDU 包这一步不要勾选换行符和 16 进制

2、是否支持彩信功能

不支持

3、如果要发送中文信息：AT+CSMP=17,167,0,8

中文信息要是 16 进制后面加上 1A 点发送就行！

4、目前不支持电信卡

9.5.14 休眠应用指南

模块的几种工作模式

Air780E 支持多种省电模式，通过进入不同的省电模式达到降低功耗的目的。

1. 全功能模式（默认）：一般的工作模式，CPU，时钟全速运行。
2. 睡眠模式：CPU 休眠，射频部分工作，周期进行寻呼，仅维持网络连接，能随时接收网络端的电话，短信等。功耗较低，主要的省电模式。可以通过 GPIO 中断，内部计时器中断，网络消息，来电，短信等唤醒。
3. 飞行模式：CPU 休眠,射频与 SIM 卡关闭，无法注册网络，无法接收网络端的任何信息。功耗最低。只能通过发 AT 指令或者软件主动调用接口退出。
4. 注意：模块是否有进入睡眠模式，从外部电气信号或调试日志中无法体现，只能通过检测模块的消耗电流来判断。进入休眠后模块的电流会在 0.6mA 的底电流的基础上跳动。

模式切换汇总

当前模式	下一模式		
	关机	正常模式	睡眠模式
关机		使用 PWRKEY 开机	
正常模式	使用 PWRKEY 管脚，或 VBAT 电压低于关机电压		软件调用睡眠接口，AT 版本不做动作 30s 自动休眠
睡眠模式	使用 PWRKEY 或 VBAT 电压低于关机电压	GPIO 管脚中断、定时器、接收短信或网络数据	

串口应用下支持两种睡眠模式：

- 睡眠模式 1：通过 MAIN_DTR 管脚电平控制模块是否进入睡眠。

- 睡眠模式 2：模块在串口空闲一段时间后自动进入睡眠。
- 两种模式功耗相同，串口不工作，无法进行收发数据。
- 注意： 通过串口使模块进入睡眠的先决条件是：不接 USB 口

材料准备

1、 Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。

2、PC 电脑，串口工具

应用示例

睡眠模式 1

- 开启条件：
- 发送 AT 指令 AT+CSCLK=1
- 模块进入睡眠：
- MAIN_DTR 为高时，允许系统进入休眠，当系统进入空闲状态时，就会自动进入睡眠模式。
- 模块退出睡眠：

1. 拉低 MAIN_DTR 脚 50ms 以上，模块会退出睡眠模式可以接受 AT 指令。
2. 外部 GPIO 中断，计时器中断，短信，来电，网络端下发数据。

- 模块在睡眠模式 1 时的软件功能：
- 不响应 AT 指令，但是收到数据/短信/来电串口会唤醒并有 URC 上报

```
AT

OK

AT+CSCLK=1

OK

AT+CSCLK?

+CSCLK: 1

OK
```

睡眠模式 2

- 开启条件：
- 发送 AT 指令 AT+CSLCK=2
- 模块进入睡眠：
- 系统空闲时，经过一定时间后（默认 5s），模块自动进入睡眠模式 2。
- 通过 AT+WAKETIM 配置休眠时间。例如 AT+WAKETIM=8，系统空闲后 8S 进入睡眠模式，取值（0~100）。
- 模块退出睡眠：
- 串口连续发送 AT 直到模块回应时即退出睡眠模式 2
- 模块在睡眠模式 2 时的软件功能：
- 不响应 AT 指令，但是收到数据/短信/来电会有 URC 上报

```
AT

OK

AT+CSCLK=2

OK

AT+CSCLK?

+CSCLK: 2
```

```
OK
```

常见问题

模块在收到短信、网络数据等一系列 urc 上报时，MCU 可以发送 AT 指令给模块吗？

可以，接收到网络数据时模块会被唤醒

模块休眠时能通过 AT 指令唤醒吗？

休眠模式一通过拉低 MAIN_DTR 管脚退出休眠模式，发送命令 AT+CSCLK=2 禁止模块休眠

休眠模式二可以通过连续发送 AT 唤醒模块，当模块未接收到网络数据和 AT 命令时，模块会自动休眠

飞行模式和超低功耗模式哪个更省电？

飞行模式的功耗更低，但退出飞行模式重新联网需要时间，飞行模式下，没法维持模块与服务器连接，根据实际业务情况选。

如果开关飞行频率过大，功耗也会增加。

为什么模块无法进入休眠

1.查看模块是否插入 USB，USB 连接的状态下模块保持唤醒，无法休眠

2.检查硬件电路

十 软件开发指南（LuatOS）

Air780EP 系列模组，是合宙通信基于移芯 EC718P 平台设计研发的新款 4G Cat.1 模组。Air780EP 模组，做为此系列模组中的主打二次开发的明星产品，在软件开发方面，除了支持经典的标准 AT 和 USB Driver 应用开放方式外，还支持 CSDK 和 LuatOS 二次开发。

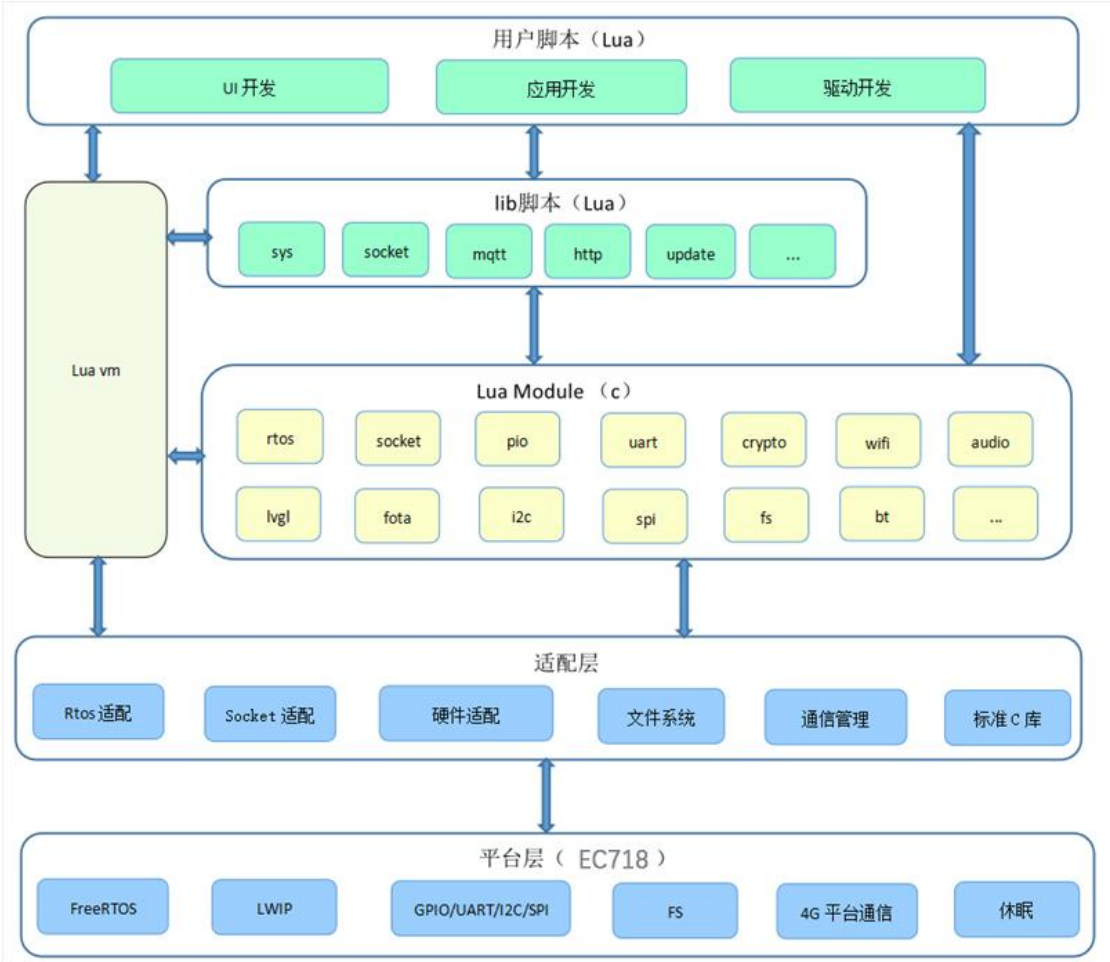
本章节对 Air780EP 模组的 LuatOS 软件二次开发方案做详细介绍。

10.1 Lua 语言学习

LuatOS 中软件开发使用的编程语言为 Lua 脚本语言；

Lua 脚本语言的学习请参考：第十八章 Lua 语言学习教程。

10.2 LuatOS 设计框架



10.3 LuatOS 软件包

LuatOS 软件由两部分组成：内核固件和应用脚本。

内核固件

有以下两种方式可以获取内核固件

1、直接下载合宙编译发布的正式版内核固件：点击此处打开下载页面 后，搜索 LuatOS-SoC@EC718P 下载需要的版本；

2、直接下载源码，自行编译最新内测版的内核固件：参考此处。

应用脚本 demo

有以下两种方式可以获取应用脚本 demo

1、直接下载合宙编译发布的正式版内核固件：点击此处打开下载页面 后，搜索 LuatOS-SoC@EC718P 下载需要的版本；版本中包含所有应用脚本 demo；

2、点击此处下载 LuatOS 主仓库，应用脚本存放在仓库的 demo 目录下。

10.4 LuatOS API

参考本文档：第二十一章 LuatOS API 。

10.5 LuatOS 开发应用指南

10.5.1 FOTA 开发应用指南

一. 概述

update 是物联网中比较常用的功能，本文介绍如何用 Air780EP 开发板，如何做 update 升级。

LuatOS-Air 开发模式下，固件分为两部分：core 和 script

远程升级时：core 为差分升级；script 为全量覆盖升级

远程升级时：可以仅升级 script；可以仅升级 core；也可以同时升级 core+script

支持合宙 iot 平台升级和自建第三方服务器升级

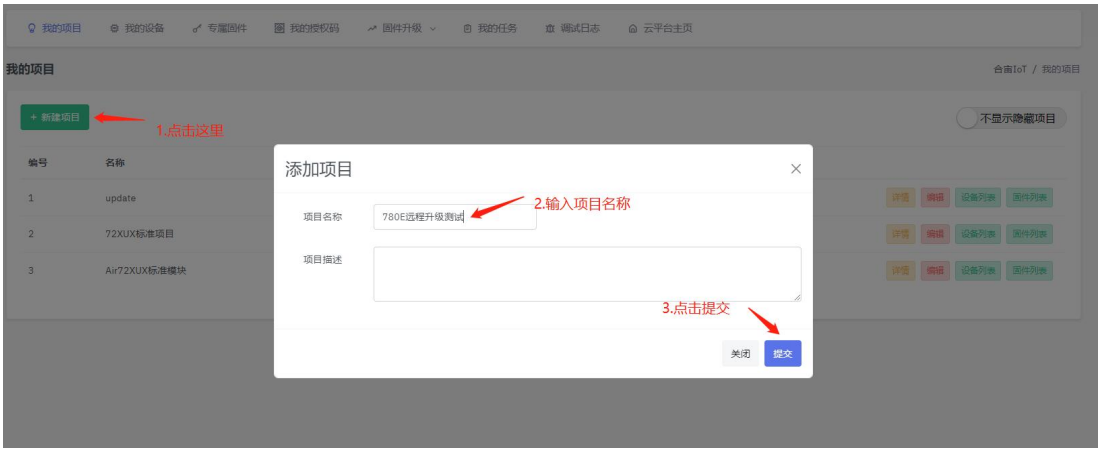
二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑，以及登录官方 IOT 后台 https://iot.openluat.com/
3. 固件和 DEMO 下载地址 LuatOS 软件包

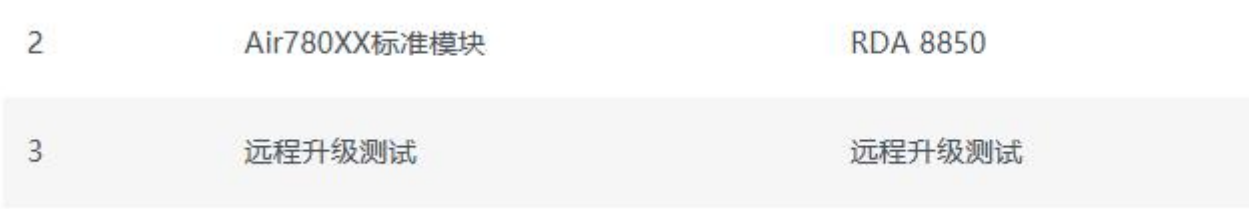
三. 合宙 iot 平台升级 core+script 过程

1. iot 平台创建产品

1. 登录合宙 iot 平台
2. 如果没有账户，注册一个账户
3. 我的项目->新建产品



4. 复制 PRODUCT_KEY



5.

6.

2. 模块端脚本修改，生成升级包

core	旧版本 core	新版本 core
例子	LuatOS-SoC_V1103_EC618.soc	LuatOS-SoC_V1103_EC618.soc
要求	1、大于等于旧版本 core 版本号	2、新旧版本 core 支持的功能相同

script main.lua	旧版本 （模块端本地烧录的）	新版本 （生成的量产文件，远程升级包）	要求
PROJECT（项目名称）	fotademo	fotademo	新旧版本保持一致

script main.lua	旧版本 （模块端本地烧录的）	新版本 （生成的量产文件，远程升级包）	要求
VERSION（软件版本号）	1.0.1	1.0.2	大于等于旧版本 VERSION
PRODUCT_KEY（项目密钥）	3TNamWSWhGQKS7pXWyIjoBZWC4xFRMCU	3TNamWSWhGQKS7pXWyIjoBZWC4xFRMCU	和 iot 平台创建的产品 Product Key 保持一致

3. 按照新版本的需求，修改 main.lua

如下图所示，PROJECT 和 PRODUCT_KEY 保持不变，PRODUCT_KEY 需和服务器保持一致，VERSION 修改为 1.0.2。

注意事项：

1、固件版本 V1103 及以上，配合 libfota.lua

2、main.lua 内填写版本号要符合 x.y.z 形式，其中 x/y/z 均为数值，不限于个位数

3、若 fota 使用合宙 iot 平台，版本号中的 y 会被忽略，推荐写 0，务必留意

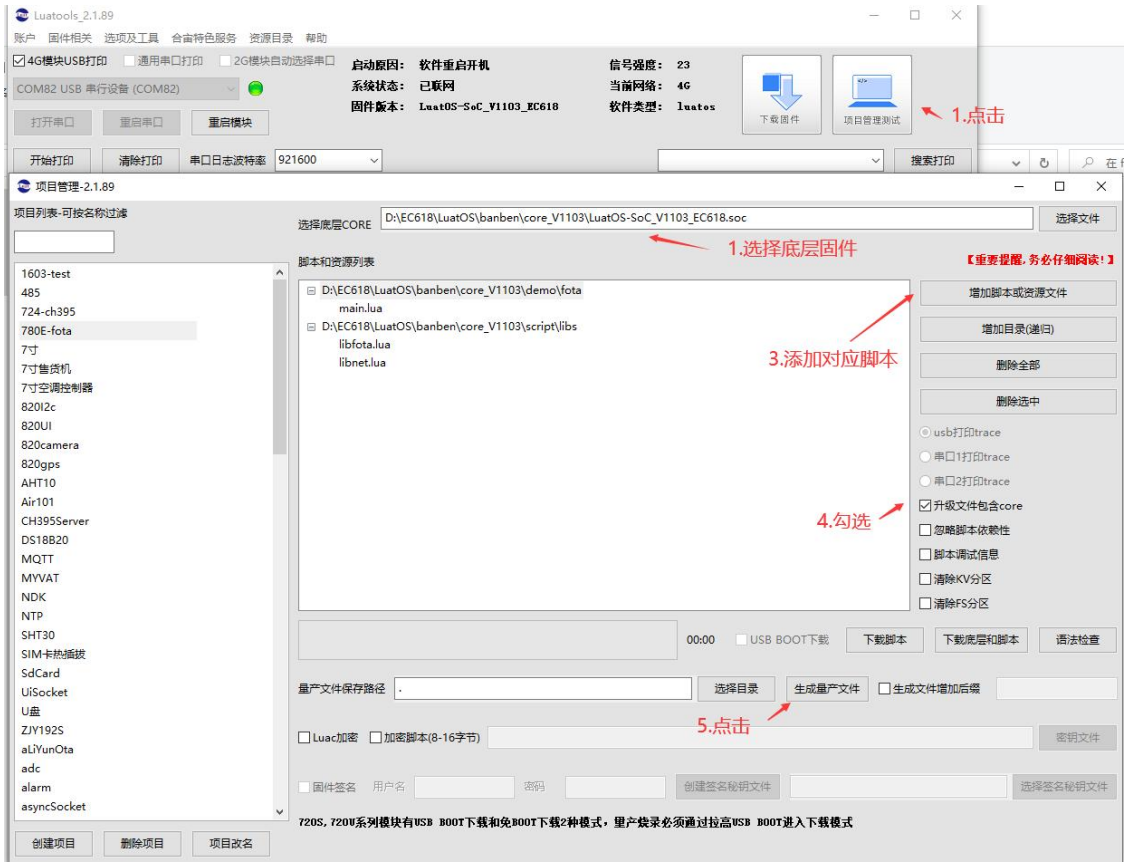
```
-- LuaTools需要PROJECT和VERSION这两个信息
PROJECT = "fotademo"
VERSION = "1.0.2"

--[[
本demo 适用于 Air780E/Air780EG/Air600E
1. 需要 V1103及以上的固件
2. 需要 LuaTools 2.1.89 及以上的升级文件生成
]]

-- 使用合宙iot平台时需要这个参数
PRODUCT_KEY = "3TNamWSWhGQKS7pXWyIjoBZWC4xFRMCU" -- 到 iot.openluat.com 创建项目，获取正确的项目id
```

4. 使用 Luatools 生成 bin 文件差分包

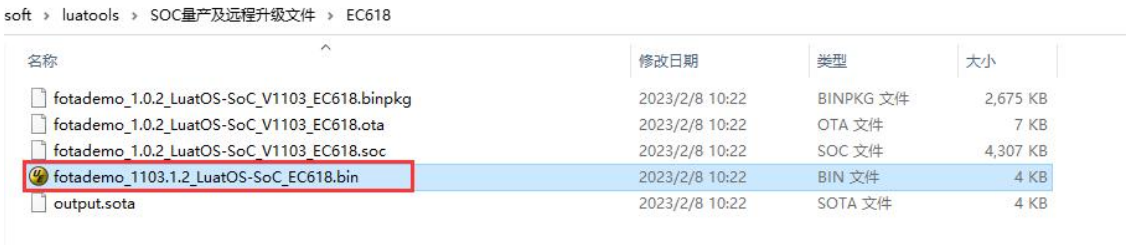
注意：只有 Luatools 版本 2.1.89 以上支持生成 bin 文件差分包，低版本仅支持生成 sota 文件

只升级脚本，差分包制作流程如下：

固件和 DEMO 下载地址 LuatOS 软件包，这里下载的是 LuatOS-SoC@EC618 V110 项目下的 core_V1103.zip 软件包。下载对应软件包后按如下步骤生成对应的差分包。



在 Luatools 目录下的 SOC 量产及远程升级文件下，找到 EC618 文件夹，打开后缀名为.bin 的文件 fotademo_1103.1.2_LuatOS-SoC_EC618.bin，这个就是文件差分包。



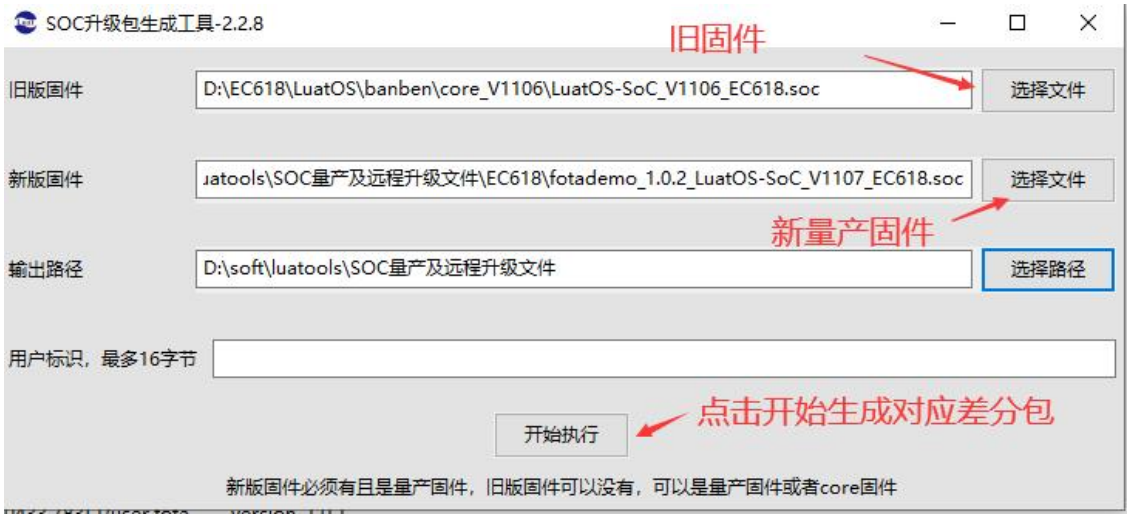
需要升级底层 CORE，差分包制作流程如下：

1）生成新版本的量产固件，如下所示：

2）根据新旧固件生成对应差分文件，如下所示：

a、点击选项及工具选项，选择升级包制作工具

b、选择对应文件，点击开始执行生成对应差分文件



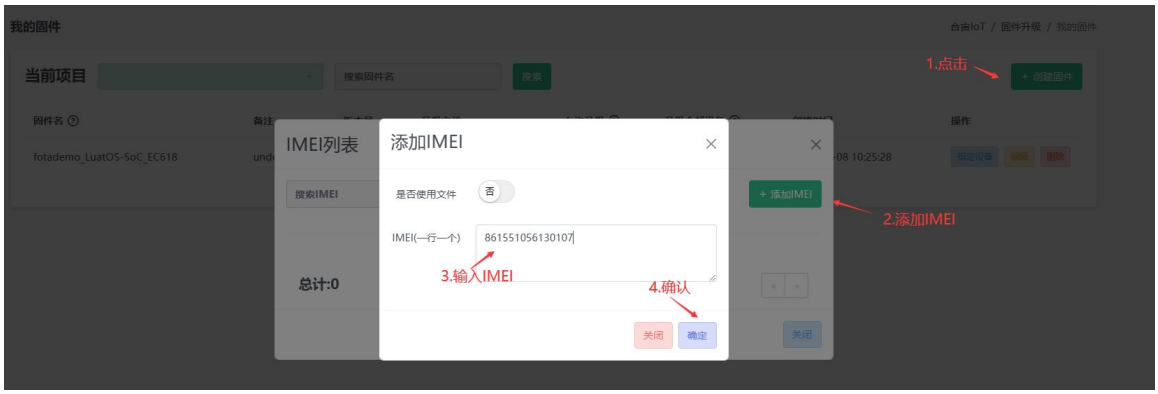
3）生成的对应差分文件见下图所示：

5.在合宙 iot 平台配置升级包

进入 iot 平台创建的产品，打开我的项目->固件列表->创建固件，如下图所示

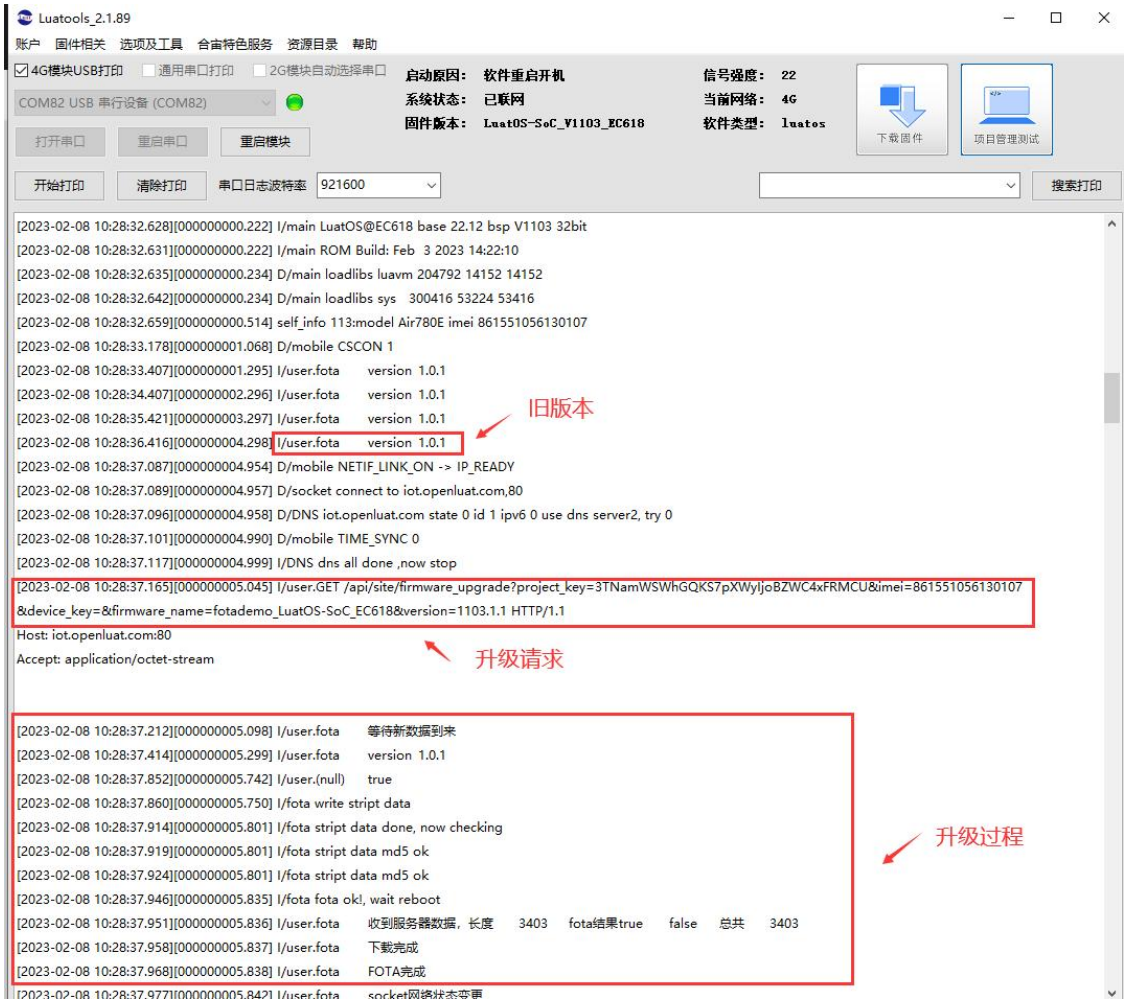
点击确定之后，升级配置完成，如下图所示

添加需要升级的模块 imei，如下图所示



7. 模块开机，完成升级

烧录旧版本的模块开机后，连接升级服务器，下载差分升级包，自动重启，重启过程中完成固件升级，整个过程的关键日志如下图所示



```
[2023-02-08 10:28:37.977][000000005.842] I/user.fota      socket网络状态变更
[2023-02-08 10:28:37.983][000000005.844] I/user.fota      socket网络状态变更
[2023-02-08 10:28:38.008][000000005.893] I/user.fota      0
[2023-02-08 10:28:38.038] DBG: diag com USB 断开连接 COM83 CommError,[WinError 22] 设备不识别此命令。
[2023-02-08 10:28:38.749] DBG: ap log port COM83打开成功
[2023-02-08 10:28:38.780] DBG: soc log port COM82打开成功
[2023-02-08 10:28:39.057][000000000.231] I/pm pm mode 1
[2023-02-08 10:28:39.063][000000000.231] I/pm poweron: Power/Reset
[2023-02-08 10:28:39.069][000000000.232] I/main LuatOS@EC618 base 22.12 bsp V1103 32bit
[2023-02-08 10:28:39.078][000000000.232] I/main ROM Build: Feb  3 2023 14:22:10
[2023-02-08 10:28:39.086][000000000.241] D/main loadlibs luavm 204792 14152 14152
[2023-02-08 10:28:39.090][000000000.241] D/main loadlibs sys  300416 53224 53416
[2023-02-08 10:28:39.095][000000000.522] self_info 113:model Air780E imei 861551056130107
[2023-02-08 10:28:39.582][000000001.170] D/mobile CSCON 1
[2023-02-08 10:28:39.720][000000001.305] I/user.fota      version 1.0.2
[2023-02-08 10:28:40.366][000000001.933] D/mobile NETIF_LINK_ON -> IP_READY
```

7.云平台查看升级日志

登录云平台，选择对应设备升级的产品->固件升级->升级日志->输入对应的 imei 并选择日期->搜索

可以看到设备在此日期下的空中升级日志。正常升级成功的情况下，短时间内会有两次升级请求，第一次请求升级获取到升级包之后，设备重启升级成功会再次向云平台请求是否有更高版本的固件可以升级。直到升级结果为“已是最新版本”服务器没有更高版本的固件为止。

设备升级失败等情况，可对照以下表格寻求解决方案。

响应码	响应信息	问题以及解决办法	云平台日志结果
3	无效的设备	检查请求键名(imei 小写)正确性	无
17	无权限	设备会上报 imei、固件名、项目 key。 服务器会以此查出设备、固件、项目三 条记录， 如果 这三者不在同一个用户名下，就会认为无权限。 设备不在项目 key 对应的账户下， 可寻找合宙技术支持查询该设备在哪个账户下， 核实情况后可修改设备归属	无权限
21	21	不允许从 1.0.2 升到 1.0.3	21
25	无效的项目	productkey 不一致，检查是否存在拼写错误， 检查模块是否在本人账户下，若不在本人账户下，请联系合宙工作人员处理	未找到项目
26	无效的固件	固件名称错误，项目中没有对应的固件。用户自己修改了固件名称，可对照升级日志中设备当前固件名与升级配置中固件名是否相同（固件名称，固件功能要完全一致，只是版本号不同）。	找不到固件
27	已是最新	1、设备固件版本高于云平台 2、固件版本设备脚本版本高于云平台版本	已是最新版本

响应码	响应信息	问题以及解决办法	云平台日志结果
	版本	3、用户项目升级配置中未添加该设备 4、云平台升级配置中，是否升级配置为否	设备不在配置的升级设备列表中
40	循环升级	云平台进入设备列表搜索被禁止的 imei，解除禁止升级就好了。 云平台防止模块在升级失败后，反复请求升级导致流量耗尽， 在模块一天请求升级六次后会禁止模块升级。可在平台解除。	检测到循环升级（已禁止升级，请到设备列表解除）
43	请等待	云平台生成差分升级包需要等待， 一到三分钟后云平台生成完成差分包便可以请求成功。	正在生成版本，请等待
44	基础版本缺失	平台未收录设备固件（多为定制固件， 可寻找合宙技术支持人员添加，需提供定制固件邮件）	基础 CORE 版本缺失

四. 自建第三方服务器升级过程

注意：只有 Luatools 版本 2.1.89 以上支持生成 bin 文件差分包，低版本仅支持生成 sota 文件

手动生成 core 差分包+script 全量包。例如 core 版本有 1001、1002、1003，现在要统一升级到 1004，则需要按照如下步骤操作：

1、手动生成：从 1001 到 1004 的 core 差分包+script 全量包、从 1002 到 1004 的 core 差分包+script 全量包、从 1003 到 1004 的 core 差分包+script 全量包

2、模块请求升级时，服务器根据模块中的 core 版本返回对应的升级包文件

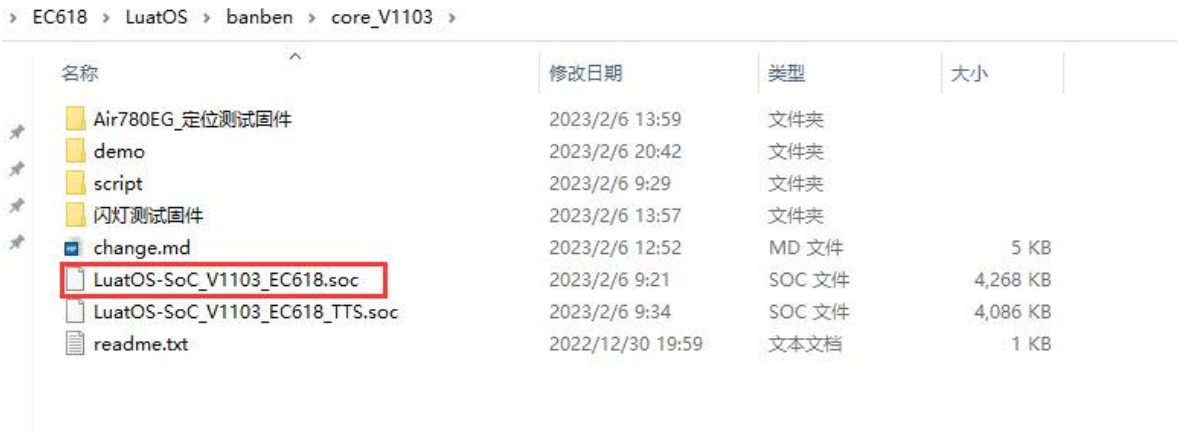
手动生成差分包的过程如下：

用合宙提供的 luatools 中的 soc 差分工具生成差分包

操作步骤：

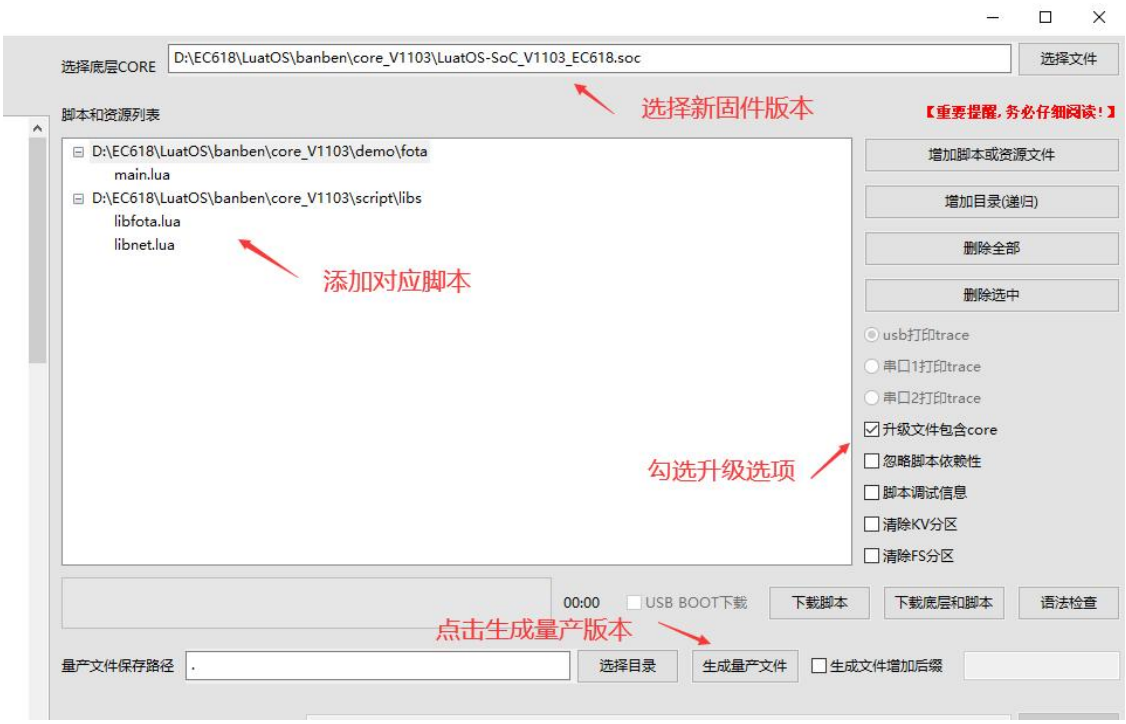
1、准备：模块中使用的旧版本 core 升级文件

此文件指的是模块中旧版本 core 中的.soc 文件（从合宙官方渠道下载），例如以 LuatOS-SoC_V1103_EC618.soc 为例

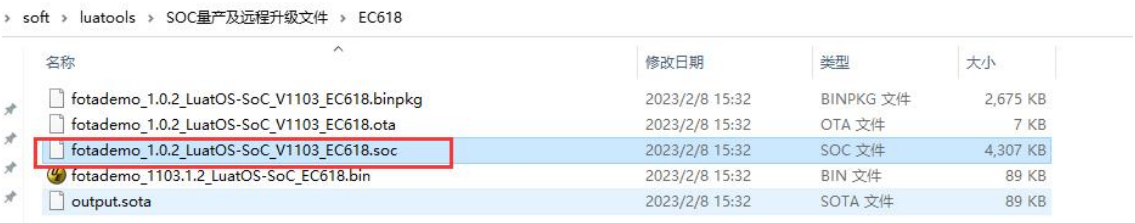


2. 使用 Luatools 分别生成新旧版本的升级包

如下图所示



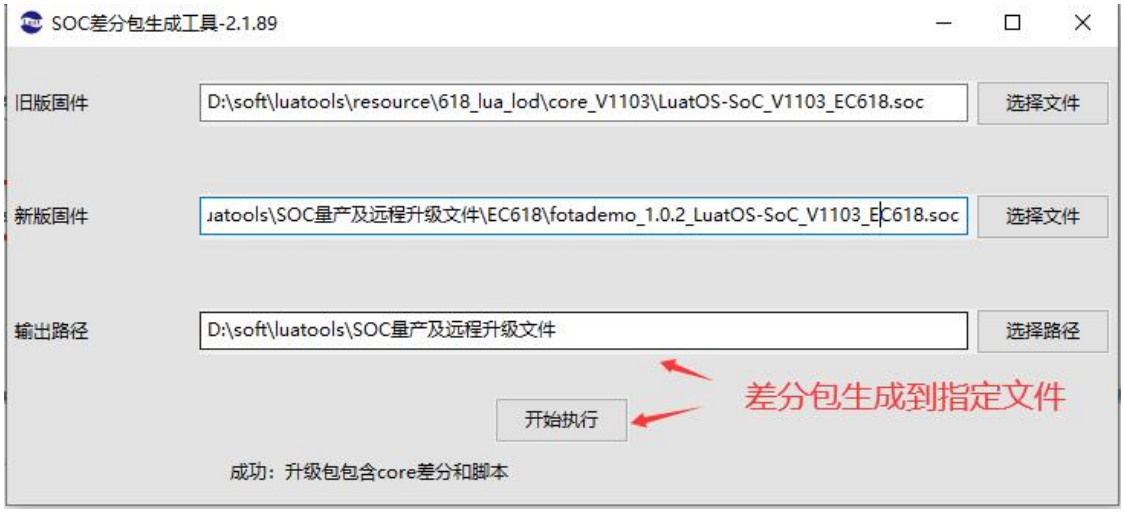
在 Luatools 目录下 SOC 量产及远程升级文件下的 EC618 下，生了一个后缀名为.soc 的文件 fotademo_1.0.2_LuatOS-SoC_V1103_EC618.soc，这个就是新版本的升级包



3、生成差分包

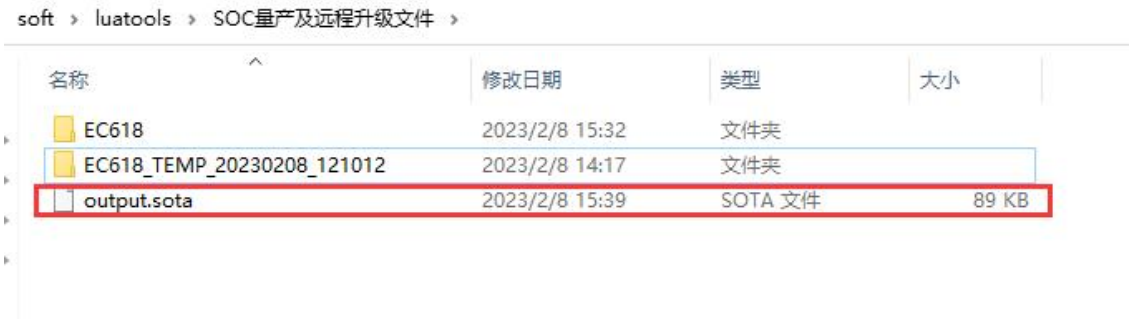
使用 luatools 生成升级文件，无论 core 是否需要升级，升级文件必须包含 core，因为差分会用到

使用 luatools 内的 soc 差分工具生成即可



生成对应差分包如下图所示：

注意：差分包不能超过 480k



4、差分包上传到自建服务器

lua 固件默认支持 HTTP 获取升级包

启动远程升级功能，demo/fota/mian.lua 文件做如下修改，注释掉使用合宙 iot 平台进行升级，打开使用自建服务器进行升级部分代码，填写对应的 ota_url 。

注意：服务器端需要有判断版本号的能力。

```
-- 使用合宙 iot 平台进行升级
--libfota.request(fota_cb)--sys.timerLoopStart(libfota.request, 3600000, fota_cb)

-- 使用自建服务器进行升级

 local ota_url = "http://myserv.com/myapi/version=" .. _G.VERSION .. "&imei=" .. mobile.imei()
```

```
libfota.request(fota_cb, ota_url)

sys.timerLoopStart(libfota.request, 3600000, fota_cb, ota_url)
```

5、使用其他协议实现自建服务器远程升级

可参考脚本库中配置。无论使用哪种协议，远程升级的关键是获取正确的升级固件包。

10.5.2 SOCKET 开发应用指南

一. 概述

socket 是物联网中最常用的功能，本文介绍如何用 Air780EP(V)开发板，演示用阻塞方式做串口透传远程服务器，简单的串口 DTU。

LuatOS-Air 开发模式下，固件分为两部分：core 和 script

二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑，以及官方 TCP 测试 web 页面：https://netlab.luatos.com/
3. 固件和 DEMO 下载地址 LuatOS 软件包

三. 演示步骤

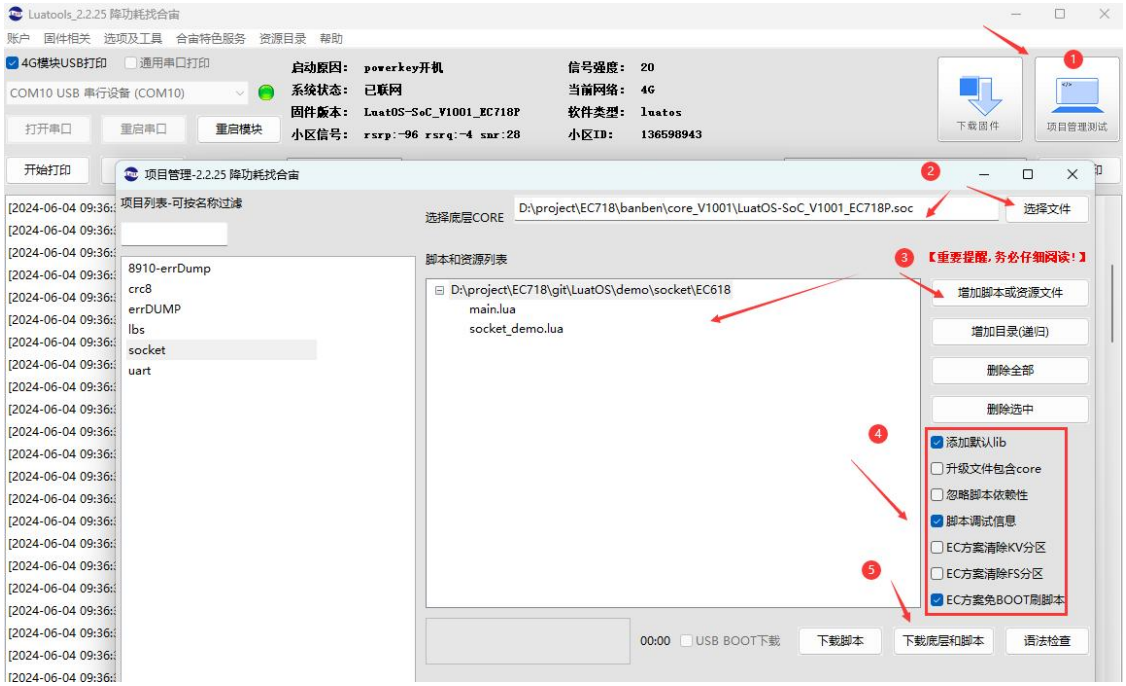
本文以 demo\socket\EC618 这个 DEMO 为例做演示

1. 打开 TCP 测试页面 https://netlab.luatos.com/点击打开 TCP 记录对应的 ip 地址和端口号。

2. 在 DEMO 最后加入如下脚本，第 1 个参数为串口 id，这里用的 main 主串口 id 为 1，第 2 个参数为 TCP 的 IP 地址，这里为'112.125.89.8'，第 3 个参数为 TCP 的 PORT 口，这里为'45087'。

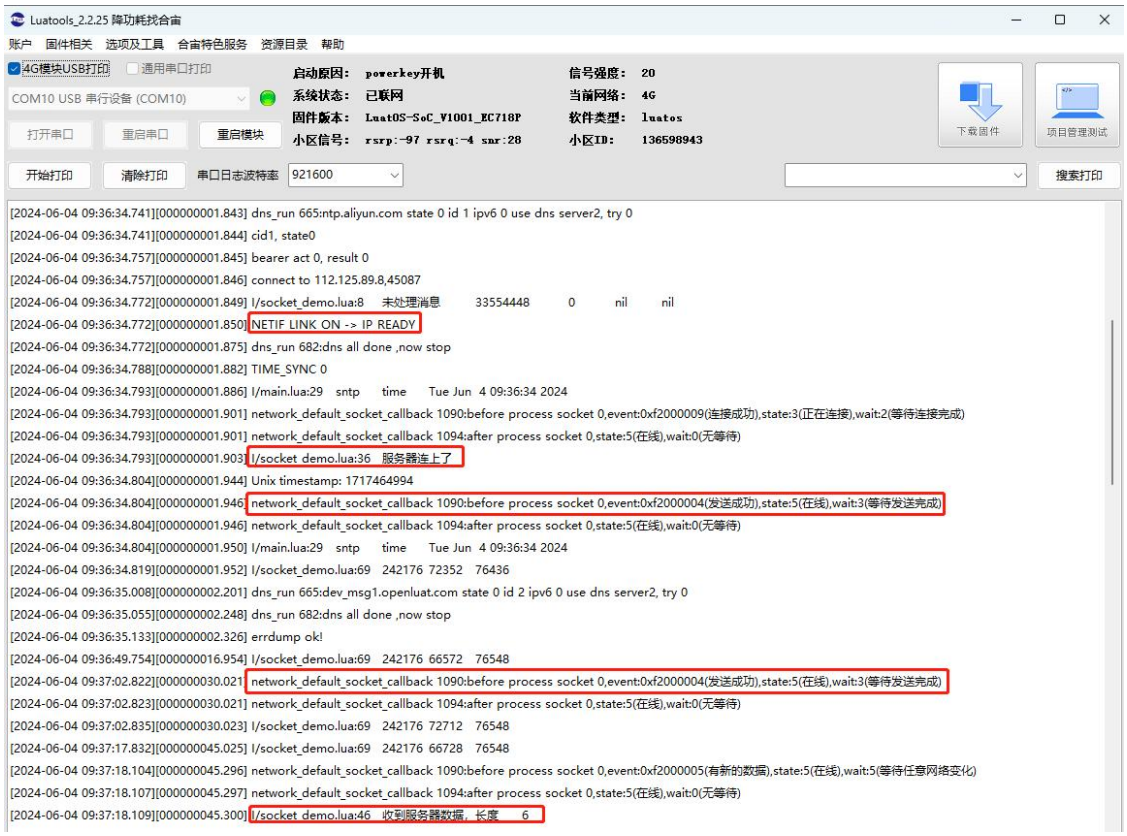
```
83
84  function dtuDemo(uart_id, ip, port)
85      sysplus.taskInitEx(dtuTask, dlName, netCB, uart_id, ip, port)
86  end
87  dtuDemo(1,'112.125.89.8','45087')
```

3. 下载对应 core 和 script 到模块，第 1 次下载选择下载底层和脚本。后面下载底层固件不不变化的情况下可以选择免 BOOT 下载的方式，只下载对应的脚本就可以了。

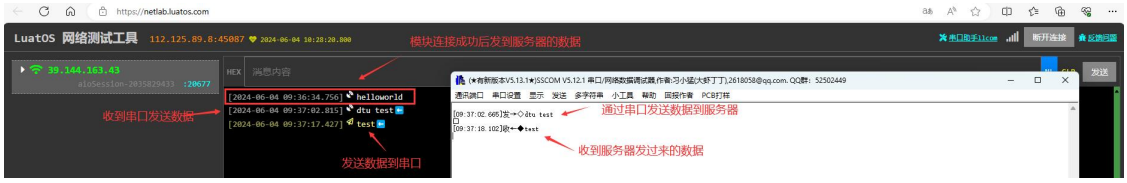


4. 下载后模块运行效果

模块 log



服务器和串口工具对应 log



四．代码解析

完整脚本代码点击右侧查看 demo/socket/EC618/socket_demo.lua

下面的脚本是串口数据的处理，串口数据通过 buff 的方式进行收发的。

uart.setup：配置串口参数

uart.rx：buff 形式读串口，一次读出全部数据存入 buff 中

```lua
--下面演示用阻塞方式做串口透传远程服务器，简单的串口DTU
local dlOnline = false
local com_buff = zbuff.create(4096)
local dlName = "D1_TASK"
local function netCB(msg)
    log.info("未处理消息", msg[1], msg[2], msg[3], msg[4])
end

local function dtuTask(uart_id, ip, port)
    dlOnline = false
    local tx_buff = zbuff.create(1024)
    local rx_buff = zbuff.create(1024)
    local netc
    local result, param, is_err
    result = uart.setup(uart_id,115200,8,1)
    uart.on(uart_id, "receive", function(id, len)
        uart.rx(id, com_buff)
        if dlOnline then    -- 如果已经在线了，则发送socket.EVENT消息来打断任务里的阻塞等待状态，让任务循环继续
            sys_send(dlName, socket.EVENT, 0)
        end
    end)
```

```lua
local function dtuTask(uart_id, ip, port)
    dlOnline = false
    local tx_buff = zbuff.create(1024)
    local rx_buff = zbuff.create(1024)
    local netc
    local result, param, is_err
    result = uart.setup(uart_id,115200,8,1)
    uart.on(uart_id, "receive", function(id, len)
        uart.rx(id, com_buff)
        if dlOnline then    -- 如果已经在线了，则发送socket.EVENT消息来打断任务里的阻塞等待状态，让任务循环继续
            sys_send(dlName, socket.EVENT, 0)
        end
    end)
```

下面的脚本是 SOCKET 连接服务器和发送数据部分的处理。

socket.create：在某个适配的网卡上申请一个 socket_ctrl

socket.debug(ctrl, true)：打开调试信息,默认是关闭状态

socket.config：配置 network 一些信息

libnet.waitLink：阻塞等待网卡的网络连接上，只能用于 sysplus.taskInitEx 创建的任务函数中

libnet.connect：阻塞等待 IP 或者域名连接上，如果加密连接还要等握手完成，只能用于 sysplus.taskInitEx 创建的任务函数中

libnet.tx(d1Name, 0, netc, "helloworld")：连接成功后发送"helloworld"到服务器

```
netc = socket.create(nil, dlName)
socket.debug(netc, true)
socket.config(netc, nil, nil, nil)
-- socket.config(netc, nil, true)
while true do

    log.info(rtos.meminfo("sys"))
    result = libnet.waitLink(dlName, 0, netc)
    result = libnet.connect(dlName, 15000, netc, ip, port)
    -- result = libnet.connect(dlName, 5000, netc, "112.125.89.8",34607)
    dlOnline = result
    if result then
        log.info("服务器连上了")
        libnet.tx(dlName, 0, netc, "helloworld")
    end
```

下面的脚本是接收部分数据处理，服务器连接成功后，通过 while 循环进行数据接收。出现异常情况后会跳出循环。

socket.rx(netc, rx_buff)：接收对端发出的数据，注意数据已经缓存在底层，使用本函数只是提取出来，UDP 模式下一次只会取出一个数据包

uart.tx(uart_id, rx_buff)：接收到的同时发到串口

libnet.tx(d1Name, 15000, netc, tx_buff)：检查到串口的数据后，把串口数据发送服务器

```
while result do
    succ, param, _, _ = socket.rx(netc, rx_buff)
    if not succ then
        log.info("服务器断开了", succ, param, ip, port)
        break
    end
    if rx_buff:used() > 0 then
        log.info("收到服务器数据，长度", rx_buff:used())
        uart.tx(uart_id, rx_buff)
        rx_buff:del()
    end
    tx_buff:copy(nil, com_buff)
    com_buff:del()
    if tx_buff:used() > 0 then
        result, param = libnet.tx(dlName, 15000, netc, tx_buff)
    end
    if not result then
        log.info("发送失败了", result, param)
        break
    end
    tx_buff:del()
    if com_buff:len() > 4096 then
        com_buff:resize(4096)
    end
    if tx_buff:len() > 1024 then
        tx_buff:resize(1024)
    end
    if rx_buff:len() > 1024 then
        rx_buff:resize(1024)
    end
    log.info(rtos.meminfo("sys"))
    -- 阻塞等待新的消息到来，比如服务器下发，串口接收到数据
    result, param = libnet.wait(dlName, 15000, netc)
    if not result then
        log.info("服务器断开了", result, param)
        break
    end
end
```

异常处理，出现异常情况后会端口网络连接，1000ms 会重新连接。

libnet.close：阻塞等待网络断开连接，只能用于 sysplus.taskInitEx 创建的任务函数中

```
dlOnline = false
libnet.close(dlName, 5000, netc)
log.info(rtos.meminfo("sys"))
sys.wait(1000)
```

以上是 SOCKET 阻塞方式，进行数据接收和发送。

SOCKET 异步方式的 DEMO 见 demo/socket/EC618/async_socket_demo.lua

五．常见问题

1、tcp 连接，心跳包建议多长时间一次

因为基站资源有限，如果不发心跳包保活，基站会主动断掉链路，回收资源，模块和服务器无感，并不知道链路已经断开。建议心跳包的频率不要超过 4 分钟，一般都是建议使用 2 分钟。

2、数据收发延迟大、速度慢、经常失败、掉线断开

1. 检查下是否存在代码逻辑错误，导致异常
2. 检查下是否不断重启，导致异常
3. 检查下服务器网络是否稳定，不要用内网穿透方式搭建服务器
4. 检查下使用环境是否网络覆盖不好，例如车库、地下、电梯、山区等
5. 检查下模块信号、网络注册、网络附着、PDP 激活状态
6. 排查是否为设备天线问题：发出来设备的天线调试指标参数给合宙技术支持人员
7. 如果经常出现连接被动断开：
8. 1) 如果直接使用的是 tcp udp socket 连接，检查下心跳包的频率，基站策略会关闭长时间没有数据传输的连接，建议心跳包的频率不要超过 4 分钟，一般都是建议使用 2 分钟
9. 2) 如果使用的是 mqtt 连接，检查下 mqtt keep alive 的时间，基站策略会关闭长时间没有数据传输的连接，建议心跳包的频率不要超过 4 分钟，一般都是建议使用 2 分钟
10. 3) 如果使用的是 mqtt 连接，检查下是否在 1.5 倍的 mqtt keep alive 的时间，没有成功发送数据到服务器，就会被被服务器主动断开，这种情况一般都是发送数据超时引起的
11. 排查是否为网络问题：如果同一地点的所有设备都有问题，可能和网络环境有关系；使用手机设置为同样的网络模式，对比测试确认下；或者把设备移动到其他距离此地点比较远的地方对比测试。
12. 排查是否设备单体问题：如果同一地点，某些设备正常，某些设备异常，按照如下几种情况分析
13. 1) 分析正常设备和异常设备的使用环境是否相同：
14. A. 如果不同，例如异常设备固定在钢制墙壁上，正常设备放置在桌子上，钢制墙壁可能对天线射频有干扰，将异常设备和正常设备放置在同样的使用环境中，再对比测试
15. B. 如果相同，参考第 2)步
16. 2) 分析正常和异常的设备，驻留的小区是否相同：
17. A. 如果相同，重点排查异常设备的天线射频部分，分析不出结果的话，异常设备寄给合宙分析

18. B. 如同不同，多测试几次，确认下，是不是在异常小区内很容易出问题，如果异常小区很容易出问题，可能就是小区拥堵造成的
19. 提供日志给合宙技术支持人员

3、如何统计流量

运营商按照 ip 包来统计流量，ip 包包含：ip 包头+tcp 包头+用户数据，以 IPv4 为例：

ip 包头包含固定的 20 字节+可选的 4 字节，至少 20 字节

tcp 包头包含固定的 20 字节+可选的 4 字节，至少 20 字节

用户数据就是用户能够感知到的数据内容了，如果直接使用 socket，就是用户感知的数据；如果使用 http、mqtt、ssl，这部分数据就不是用户能够感知的原始数据了，http 会加上 http 包头，mqtt 会加入 mqtt 的包裹部分，ssl 会加密数据

tcp 数据收发时，有 ack 确认机制，例如设备发数据给服务器后，还会收到服务器返回的 tcp ack 包，这个 tcp ack 包（至少 40 字节）也是计算在流量之内的。接收到服务器下发的数据时，设备也会回复一个 tcp ack 包，同样这个 ack 包仍然计算在流量之内

另外，socket 连接以及断开连接，都有多次数据收发，这部分也会消耗流量

例如有一个 tcp socket 连接，连接成功后，每分钟设备发送 2 字节的心跳数据到服务器，服务器收到数据后，再回复 2 字节的心跳应答数据。这个过程中，每分钟消耗的流量至少有 42（设备发送）+40（服务器回复 ack）+42（服务器发送）+40（设备回复 ack）=164 字节，实际处理中，服务器回复 ack 和服务器发送可能合并成一个 IP 包 42 字节，这样的话，至少也要 42+42+40 = 124 字节。这是最简单的算法，实际应用中，还要考虑到重传、包头中的可选字节，应该会比这里计算的流量多一些。如果要准确计算，建议在服务器端用 wireshark 抓包分析

另外一个常见的例子是，为什么通过 http 下载一个文件，实际消耗的流量比文件本身要多呢？跟上个例子类似，文件本身大小仅仅是用户数据，除了用户数据外，还有如下几部分的流量消耗：

1. 连接服务器消耗流量
2. http 请求时有 http 头，服务器下发数据时也有 http 头
3. 每包数据都有 tcp 头和 ip 头
4. 重传也会消耗流量
5. 与服务器断开连接也需要消耗流量

4、为什么我一包数据只有不到 50B 的数据，为什么一天消耗的流量要远远大于实际传输值

例如：如果使用的是 TCP 协议，需要三次握手四次挥手才算完成了一次数据交互，原始数据不多但是由于 TCP 协议决定的一包数据必须要加包头包尾帧校验等，所以实际消耗的流量不止 50B，部分运营商有限制每一包数据必须 1KB 起发，不足 1KB 也会加各种校验凑足 1kb。

10.5.3 HTTP 开发应用指南

一. 概述

HTTP 是 Hyper Text Transfer Protocol（超文本传输协议）的缩写。HTTP 是一个应用层协议，由请求和响应构成，是一个标准的客户端服务器模型。HTTP 是一个无状态的协议。HTTP 协议通常承载于 TCP 协议之上，有时也承载于 TLS 或 SSL 协议层之上，这个时候，就成了我们常说的 HTTPS，所以 HTTPS 相关的指令只需要参考 SSL 部分配置连接，其他和 http 都是一样的。

http 是物联网中常用的功能，本文介绍如何用 Air780EP(V)开发板，通过 HTTP 获取和风天气的演示和简要说明。

LuatOS-Air 开发模式下，固件分为两部分：core 和 script

二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

三. 演示步骤

本文以 demo/http 这个 DEMO 为例做演示

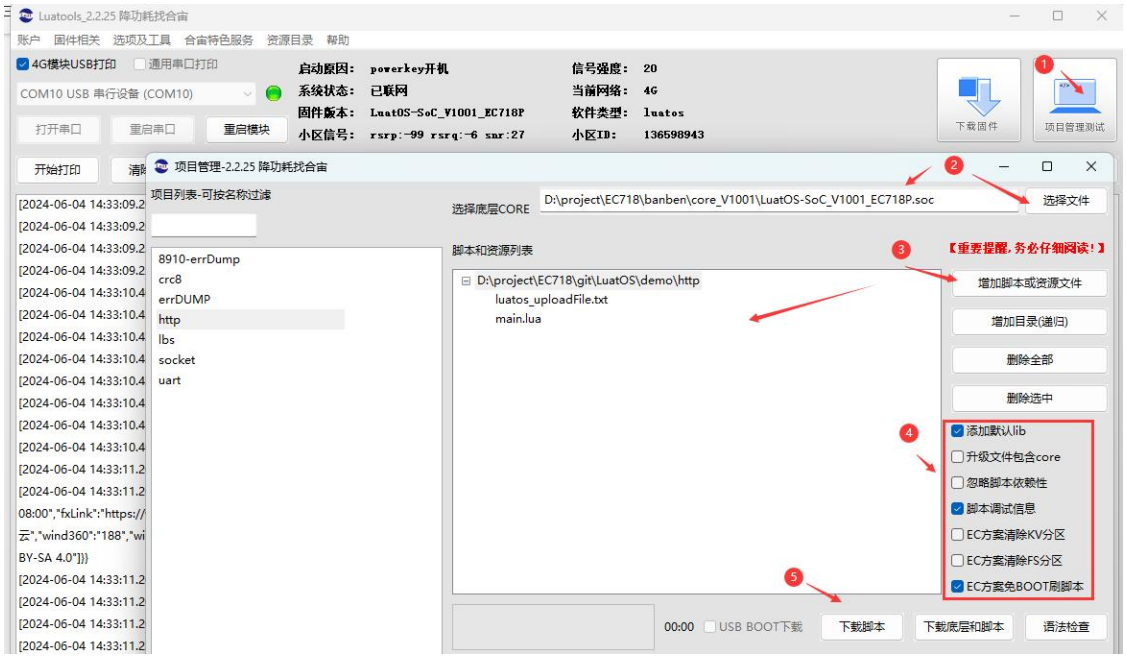
DEMO 默认演示的是 HTTP GET 请求，这里改成获取和风天气的演示

```lua
sys.taskInit(function()
    sys.wait(100)
    -- 打印一下支持的加密套件，通常来说，固件已包含常见的99%的加密套件
    -- if crypto.cipher_suites then
    --     log.info("cipher", "suites", json.encode(crypto.cipher_suites()))
    -- end

    ---------------------------------
    -------- HTTP 演示代码 ----------
    ---------------------------------
    sys.waitUntil("net_ready") -- 等联网

    while 1 do
        -- 演示GET请求
        --demo_http_get()
        -- 表单提交
        -- demo_http_post_form()
        -- POST一个json字符串
        -- demo_http_post_json()
        -- 上传文件，mulitform形式
        -- demo_http_post_file()
        -- 文件下载
        -- demo_http_download()
        -- gzip压缩的响应，以和风天气为例
        demo_http_get_gzip()

        sys.wait(1000)
        -- 打印一下内存状态
        log.info("sys", rtos.meminfo("sys"))
        log.info("lua", rtos.meminfo("lua"))
        sys.wait(600000)
    end
end)
```

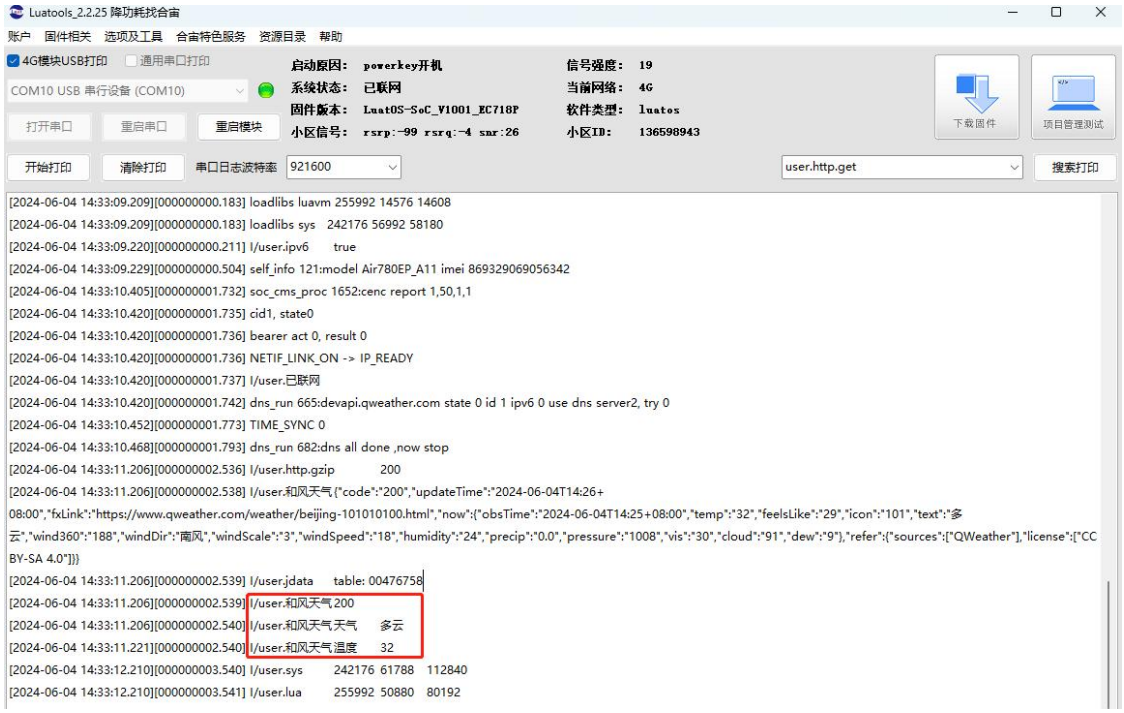
注释掉

打开注释，进行演示

下载对应 core 和 script 到模块



下载后模块运行效果



四．代码解析

完整脚本代码点击右侧查看 demo/http/main.lua

sys.waitUntil("net_ready") ：等联网成功后，做如下调用

http.request：http 客户端请求

code == 200：表示请求成功

miniz.uncompress：这个 API 的响应，总会 gzip 压缩过，需要配合 miniz 库进行解压

json.decode：解码返回数据

log.info("和风天气", "天气", jdata.now.text)：打印对应天气

log.info("和风天气", "温度", jdata.now.temp)：打印对应温度

```lua
local function demo_http_get_gzip()
    -- 这里用 和风天气 的API做演示
    -- 这个API的响应, 总会gzip压缩过, 需要配合miniz库进行解压
    local code, headers, body = http.request("GET", "https://devapi.qweather.com/v7/weather/now?location=101010100&key=0e8c72015e2b4a1dbff1688ad54053de").wait()
    log.info("http.gzip", code)
    if code == 200 then
        local re = miniz.uncompress(body:sub(11), 0)
        log.info("和风天气", re)
        if re then
            local jdata = json.decode(re)
            log.info("jdata", jdata)
            if jdata then
                log.info("和风天气", jdata.code)
                if jdata.now then
                    log.info("和风天气", "天气", jdata.now.text)
                    log.info("和风天气", "温度", jdata.now.temp)
                end
            end
        end
    end
end
```

五．常见问题

返回 404 原因，说明访问的网页不存在

10.5.4 HTTPDNS 开发应用指南

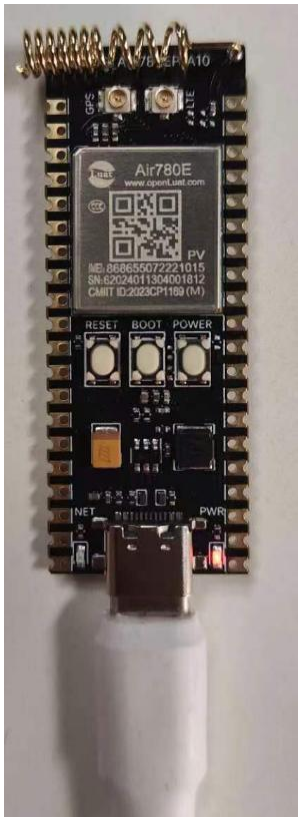
一. 概述

httpdns 是物联网中常用的功能，本文介绍如何用 Air780EP(V)开发板，通过 HTTP 进行域名解析的演示和简要说明。

LuatOS-Air 开发模式下，固件分为两部分：core 和 script

二. 材料准备

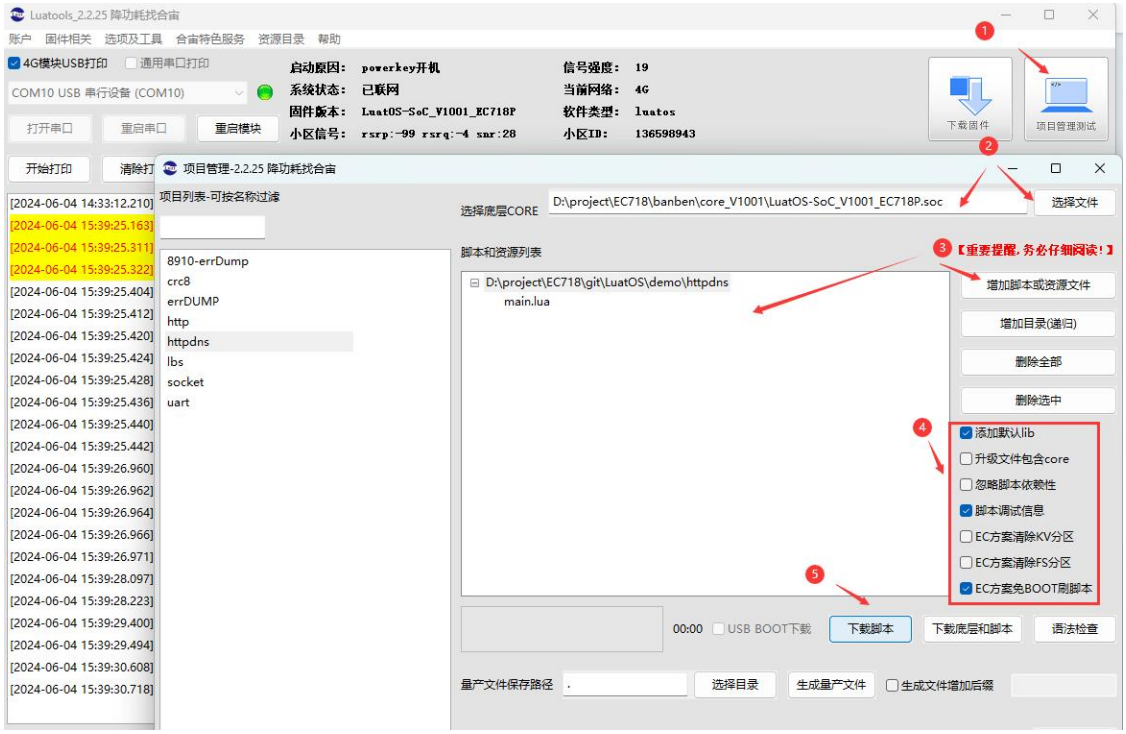
4. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
5. PC 电脑
6. 固件和 DEMO 下载地址 LuatOS 软件包



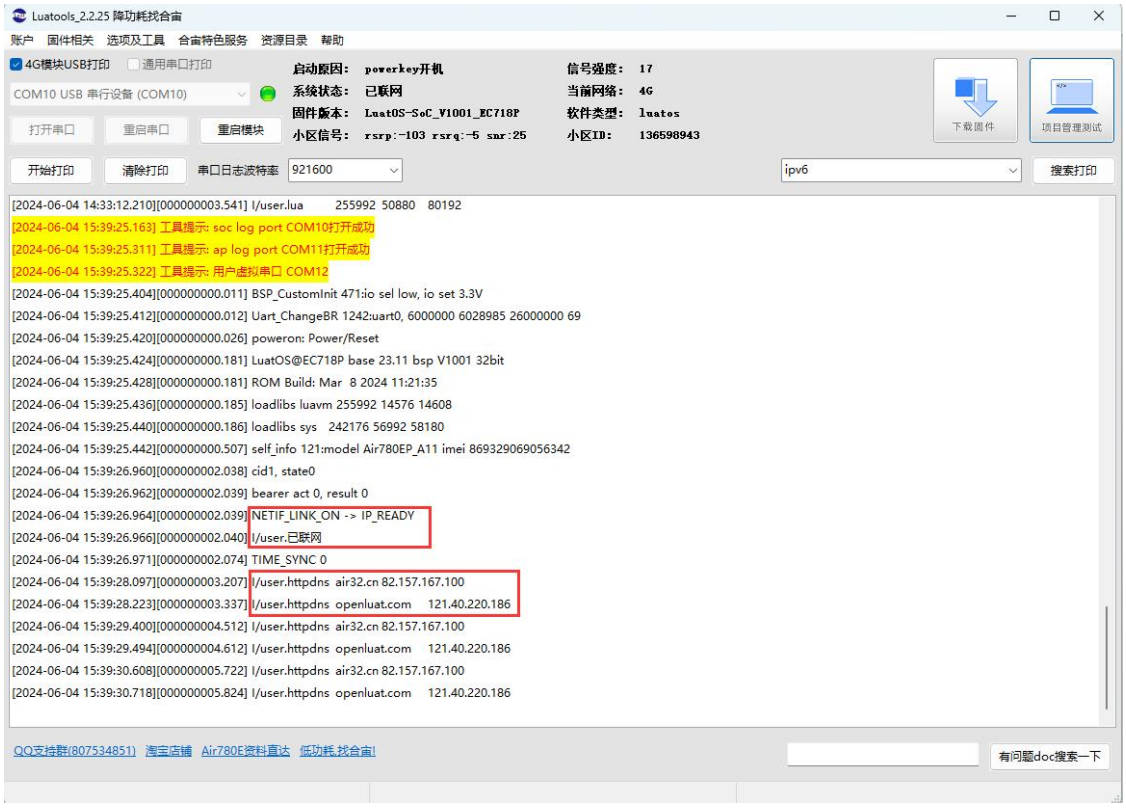
三. 演示步骤

本文以 demo/httpdns 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



下载后模块运行效果



四．代码解析

完整脚本代码点击右侧查看 demo/httpdns/main.lua

等待联网成功后，通过调用 httpdns 接口，获取对应 IP 地址

httpdns.ali：通过阿里 DNS 获取结果

httpdns.tx：通过腾讯 DNS 获取结果

```
62  sys.taskInit(function()
63      sys.waitUntil("net_ready")
64      while 1 do
65          sys.wait(1000)
66          -- 通过阿里DNS获取结果
67          local ip = httpdns.ali("air32.cn")
68          log.info("httpdns", "air32.cn", ip)
69
70
71          -- 通过腾讯DNS获取结果
72          local ip = httpdns.tx("openluat.com")
73          log.info("httpdns", "openluat.com", ip)
74      end
75  end)
76
```

五. 常见问题

获取不到对应 IP，确认模块是否正常注册上网络。如果没有

确认模块是否识别到卡

SIM 卡是否欠费，流量是否用完

当前信号是否正常

卡是不是专网卡，是否机卡绑定

10.5.5 MQTT 开发应用指南

一. 概述

优势：长连接，低带宽，高可靠。

实用场景：需要服务器下发消息给设备，需要及时收到。例如，远程开关，充电桩等

几个重要名词

1. **username** 用户名，**password** 密码，**clientid** 用户标识，mqtt 可以通过前两个参数保证连接的安全，通过 clientid 确保设备唯一性。

2. **qos** 消息质量，分为 0，1，2 三个等级，分别表示只发一次，至少收到一次和只收到一次，通过 qos 可以根据数据的重要性灵活选择发送方式以节省带宽和保证数据可靠。例如不重要的数据采集可使用 qos0 只发一次不关心服务器是否收到，重要的消息通过 qos1 确保能收到，付款信息危险设备开关等场景使用 qos2 保证数据到达的同时且不重复。

3. **host** 主机，**port** 端口，mqtt 需要连接的服务器参数。host 可以说 ip 或者域名。

4. **topic** 主题，根据主题区别消息类型和来源，主要用来分类数据。同时 mqtt 是发布订阅模型，topic 是发布和订阅者通信的重要通道。

5. **payload** 消息内容，发布和订阅的具体数据。

6. **retain** 保留消息，保留消息是一条将保留标志（retained flag）置为 true 的普通 MQTT 消息。broker 会针对主题依照 QoS 级别保留最后一条保留消息，当订阅者订阅主题时会立即收到保留消息。broker 仅为每个主题保留一条保留消息。

本文介绍如何用 Air780EP(V)开发板，进行 mqtt 数据订阅，发布，接收的功能演示。

LuatOS-Air 开发模式下，固件分为两部分：core 和 script

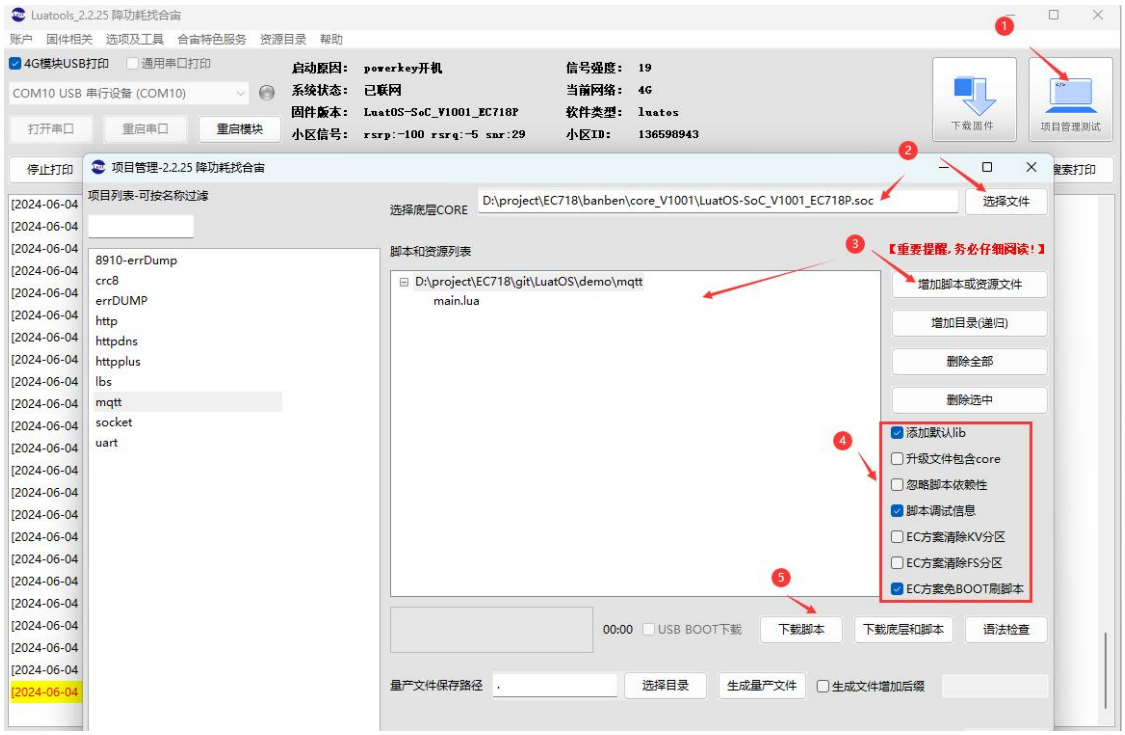
二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑，MQTT.fx 工具
3. 固件和 DEMO 下载地址 LuatOS 软件包

三. 演示步骤

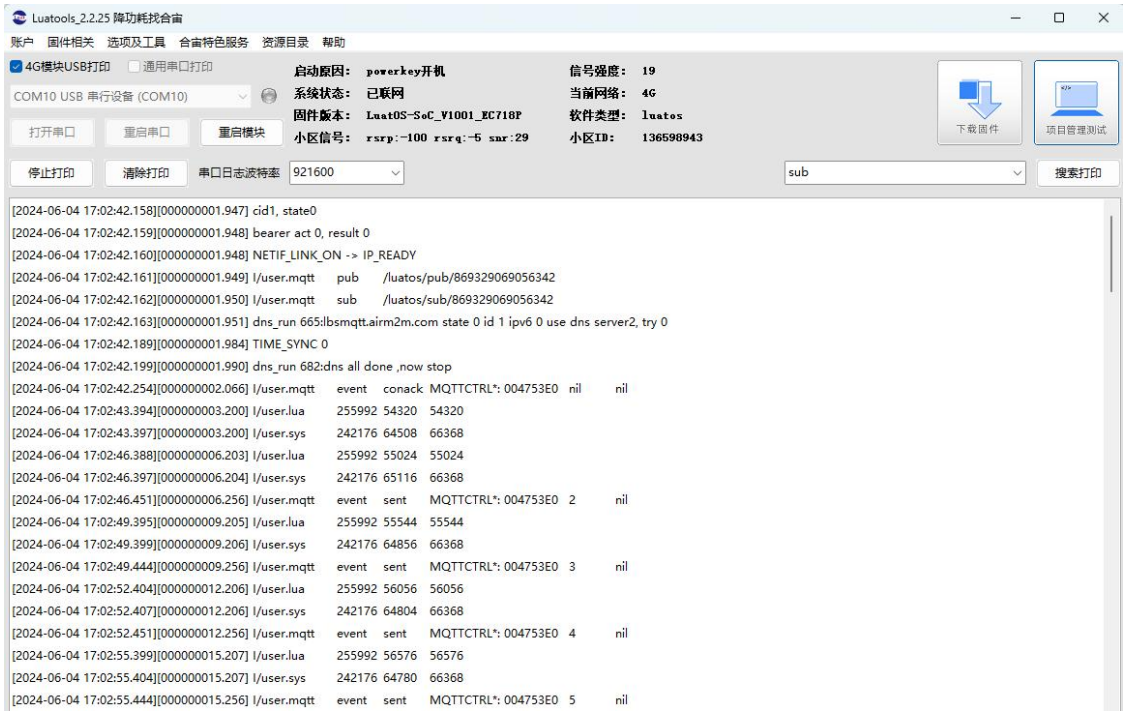
本文以 demo/mqtt 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本

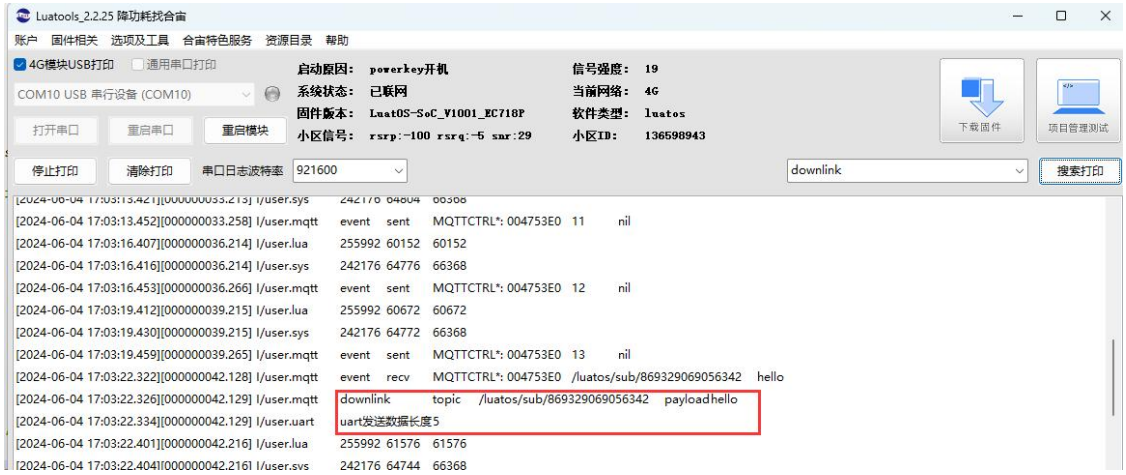


下载后模块运行效果

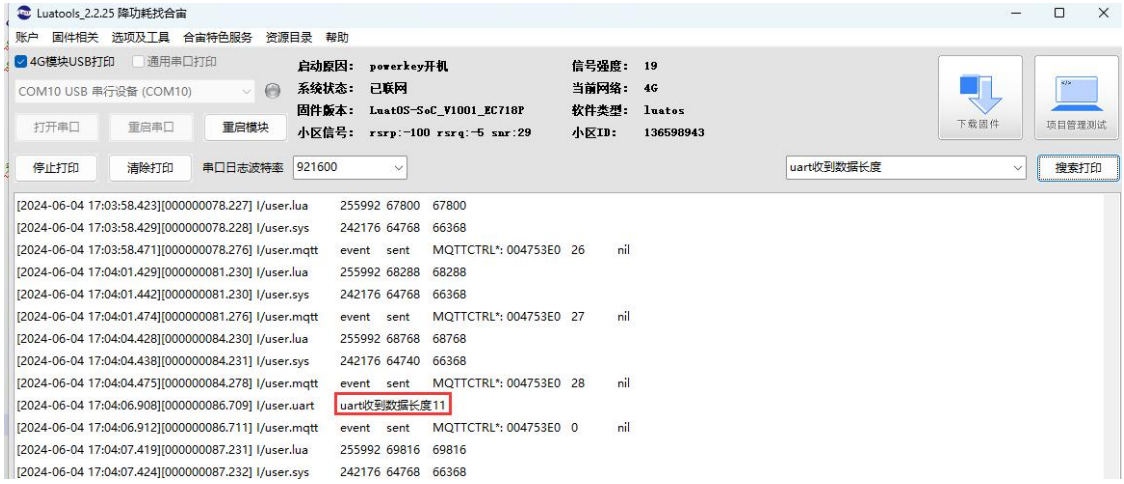
以下是 3 秒中定时上报的数据



以下是接收到 mqtt.fx 发过来的数据，并同时把数据发送到串口

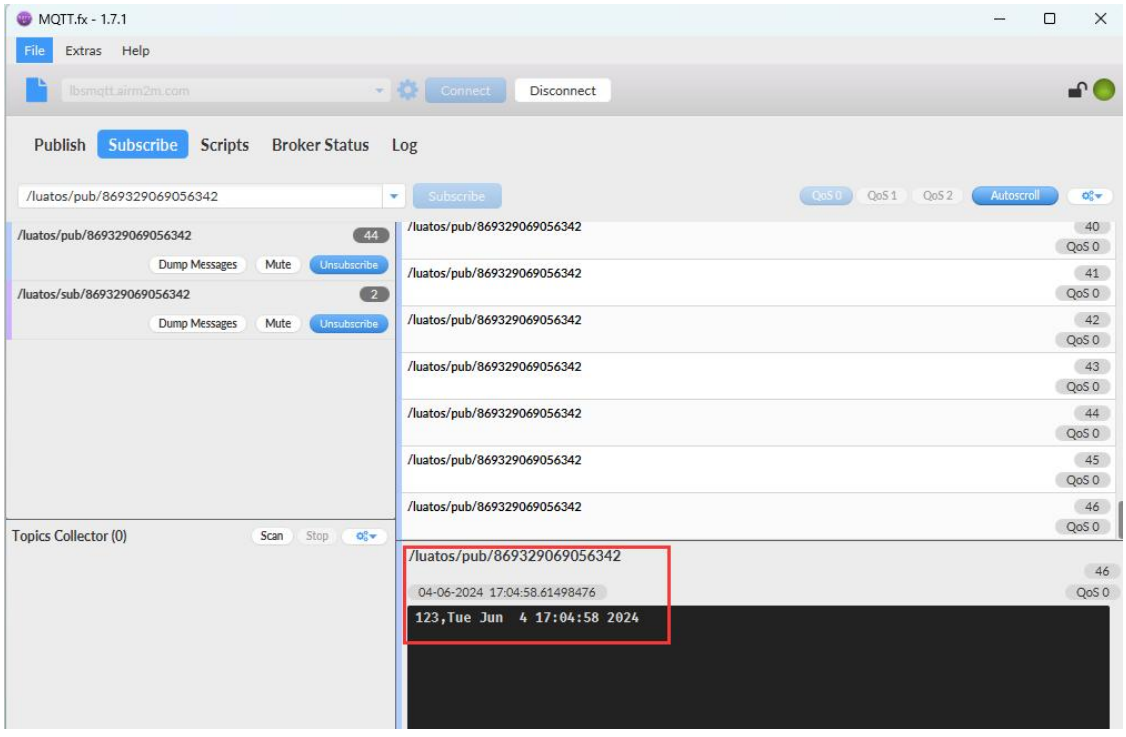


串口接收数据，并同时发布到 mqtt.fx



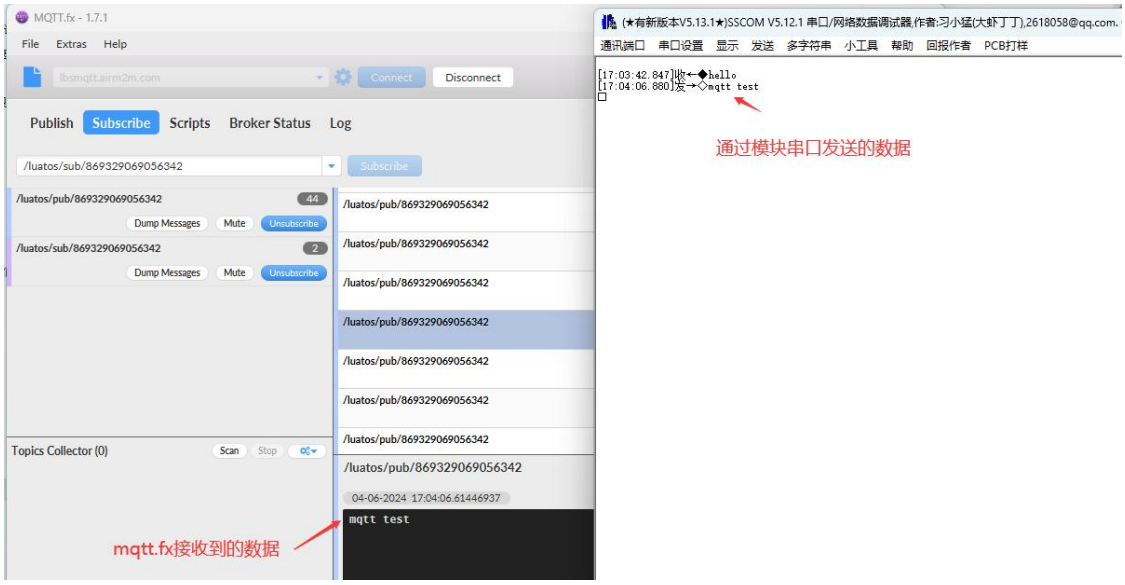
mqtt.fx 工具显示

以上是 3 秒定时上报的数据



mqtt.fx 发布的数据，同时发送到模块主串口上

模块主串口发送的数据，被 mqtt.fx 接收显示效果



四．代码解析

完整脚本代码点击右侧查看　demo/mqtt/main.lua

以下是 MQTT 演示脚本代码

mqtt.create：mqtt 客户端创建

mqttc:auth：mqtt 三元组配置及 cleanSession

mqttc:autoreconn：自动重连

mqttc:on：注册 mqtt 回调

sys.publish("mqtt_conack")：连上后发布"mqtt_conack"消息

mqtt_client:subscribe：订阅主题

sys.publish("mqtt_payload", data, payload)：收到数据后发送"mqtt_payload"消息

```
-------------------------------------
-------- MQTT 演示代码 --------------
-------------------------------------

mqttc = mqtt.create(nil, mqtt_host, mqtt_port, mqtt_isssl, ca_file)

mqttc:auth(client_id,user_name,password) -- client_id必填,其余选填
-- mqttc:keepalive(240) -- 默认值240s
mqttc:autoreconn(true, 3000) -- 自动重连机制

mqttc:on(function(mqtt_client, event, data, payload)
    -- 用户自定义代码
    log.info("mqtt", "event", event, mqtt_client, data, payload)
    if event == "conack" then
        -- 联上了
        sys.publish("mqtt_conack")
        mqtt_client:subscribe(sub_topic)--单主题订阅
        -- mqtt_client:subscribe({[topic1]=1,[topic2]=1,[topic3]=1})--多主题订阅
    elseif event == "recv" then
        log.info("mqtt", "downlink", "topic", data, "payload", payload)
        sys.publish("mqtt_payload", data, payload)
    elseif event == "sent" then
        -- log.info("mqtt", "sent", "pkgid", data)
    -- elseif event == "disconnect" then
        -- 非自动重连时,按需重启mqttc
        -- mqtt_client:connect()
    end
end)

-- mqttc自动处理重连, 除非自行关闭
mqttc:connect()
sys.waitUntil("mqtt_conack")
while true do
    -- 演示等待其他task发送过来的上报信息
    local ret, topic, data, qos = sys.waitUntil("mqtt_pub", 300000)
    if ret then
        -- 提供关闭本while循环的途径, 不需要可以注释掉
        if topic == "close" then break end
        mqttc:publish(topic, data, qos)
    end
    -- 如果没有其他task上报, 可以写个空等待
    --sys.wait(60000000)
end
mqttc:close()
mqttc = nil
```

以下是定时上报数据的脚本，这里通过创建一个 task，3 秒种发布一次 123，当前时间的数据

mqttc:publish：发布消息

```
-- 这里演示在另一个task里上报数据, 会定时上报数据,不需要就注释掉
sys.taskInit(function()
    sys.wait(3000)
    local data = "123,"
    local qos = 1 -- QOS0不带puback, QOS1是带puback的
    while true do
        sys.wait(3000)
        if mqttc and mqttc:ready() then
            local pkgid = mqttc:publish(pub_topic, data .. os.date(), qos)
            -- local pkgid = mqttc:publish(topic2, data, qos)
            -- local pkgid = mqttc:publish(topic3, data, qos)
        end
    end
end)
```

以下是串口部分的脚本，这里串口用的是 main 主串口

sys.publish("mqtt_pub", pub_topic, data)：串口收到数据后，通过 sys.publish 发布"mqtt_pub"消息，另外一个 TASK 收到这个消息后，会发布这个主题对应的数据。

sys.subscribe：订阅"mqtt_payload"消息，当收到其它 TASK 发布的这个消息后，把收到的 payload 数据发到串口。

```lua
-- 以下是演示与uart结合，简单的mqtt-uart透传实现，不需要就注释掉
local uart_id = 1
uart.setup(uart_id, 9600)
uart.on(uart_id, "receive", function(id, len)
    local data = ""
    while 1 do
        local tmp = uart.read(uart_id)
        if not tmp or #tmp == 0 then
            break
        end
        data = data .. tmp
    end
    log.info("uart", "uart收到数据长度", #data)
    sys.publish("mqtt_pub", pub_topic, data)
end)
sys.subscribe("mqtt_payload", function(topic, payload)
    log.info("uart", "uart发送数据长度", #payload)
    uart.write(1, payload)
end)
```

五．常见问题

1、连接服务器失败

1. 检查下模块信号、网络注册、网络附着、PDP 激活状态
2. 检查下 SIM 卡是否欠费
3. 使用 mqtt.fx，连接服务器确认一下是否可以连接成功，排除服务器故障
4. 部分国外的开源项目提供免费的 MQTT 代理服务器，因为网络的原因，国内存在严重的延迟或者丢包现象，导致程序运行出现问题，此现象在 AT 开发时影响严重

2、如何实现掉线自动重连

参考 DEMO 直接调用 mqttc:autoreconn 自动重连接口

3、频繁掉线是什么原因

1. 检查下是否存在代码逻辑错误，导致异常
2. 检查下是否不断重启，导致异常
3. 检查下服务器网络是否稳定，不要用内网穿透方式搭建服务器
4. 检查下使用环境是否网络覆盖不好，例如车库、地下、电梯、山区等
5. 排查是否为设备天线问题

6. 如果经常出现连接被动断开：
7. 1) 检查下 mqtt keep alive 的时间，一般建议使用 2 分钟【如果每 2 分钟内都有应用数据收发，则可以把 mqtt keep alive 的时间设置的长一点儿】，除非有强制要求，否则不能太长，也不能太短。不建议超过 4 分钟，基站策略会关闭长时间没有数据传输的连接，太长时间可能会导致连接被基站关闭；不建议少于 1 分钟，太短时间可能会因为网络环境波动导致上行数据发送超时，可能超过 1.5 倍的心跳时间，从而被服务器主动断开连接
8. 2) 检查下是否在 1.5 倍的 mqtt keep alive 的时间，没有成功发送数据到服务器，就会被被服务器主动断开，这种情况一般都是发送数据超时引起的

4、如果 KeepAlive 设置的时间长，会不会被移动断开？

会，一般建议使用 2 分钟，不建议超过 4 分钟，基站策略会关闭长时间没有数据传输的连接，太长时间可能会导致连接被基站关闭

5、如果有很多个 mqtt 设备连接到服务器，收发数据是怎样区分设备的

通过 clientid 确保设备唯一性，可以设置 IMEI 为 clientid

10.5.6 FTP 开发应用指南

一. 概述

FTP（File Transfer Protocol，文件传输协议） 是 TCP/IP 协议组中的协议之一。FTP 协议包括两个组成部分，其一为 FTP 服务器，其二为 FTP 客户端。其中 FTP 服务器用来存储文件，用户可以使用 FTP 客户端通过 FTP 协议访问位于 FTP 服务器上的资源。在开发网站的时候，通常利用 FTP 协议把网页或程序传到 Web 服务器上。此外，由于 FTP 传输效率非常高，在网络上传输大的文件时，一般也采用该协议。

LuatOS-Air 开发模式下，固件分为两部分：core 和 script

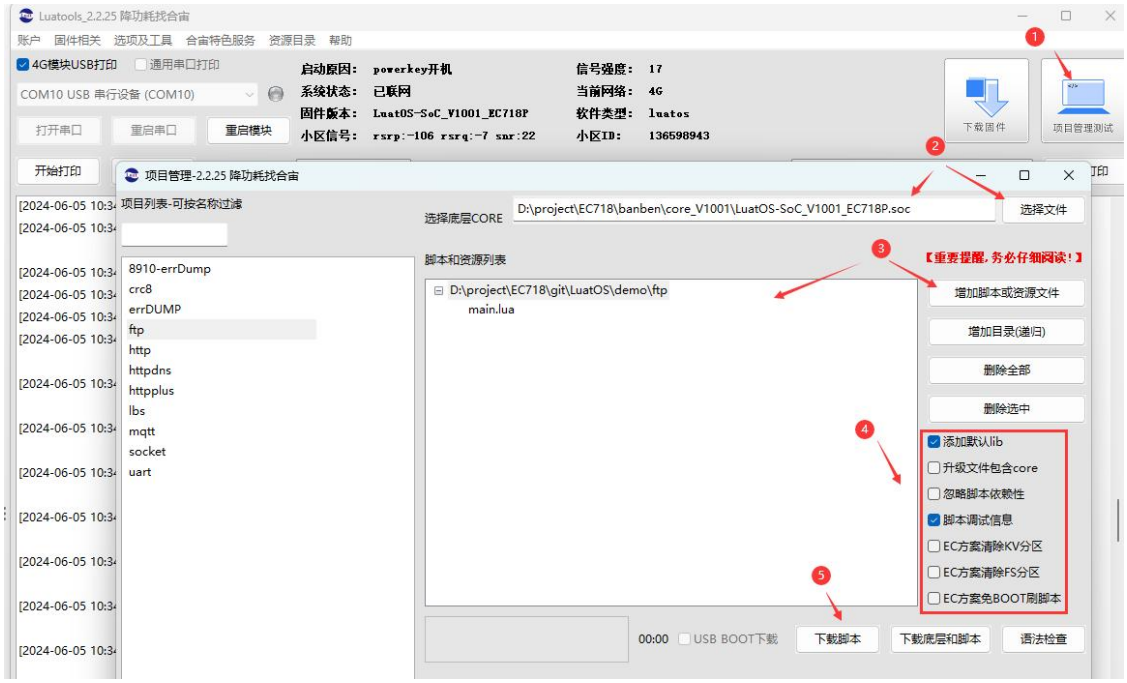
二. 材料准备

4. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
5. PC 电脑，FZ 工具
6. 固件和 DEMO 下载地址 LuatOS 软件包

三．演示步骤

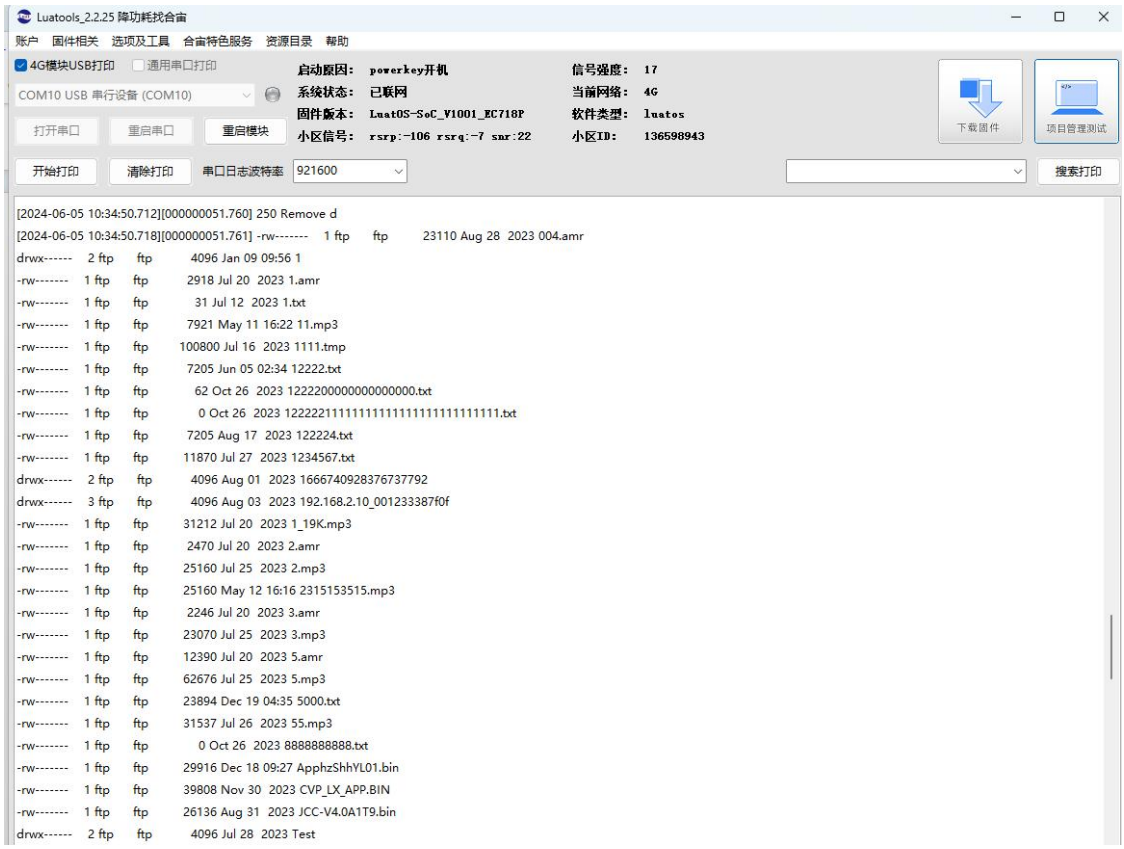
本文以 demo/ftp 这个 DEMO 为例做演示

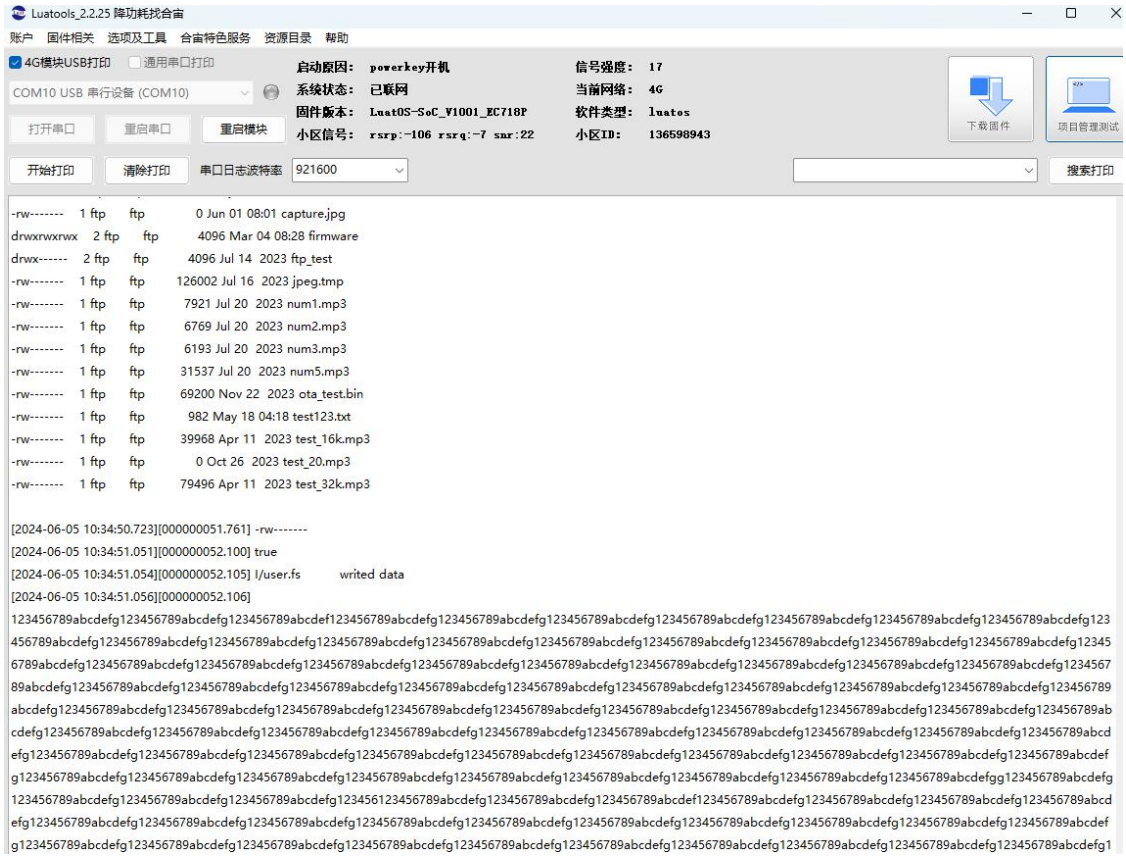
直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本

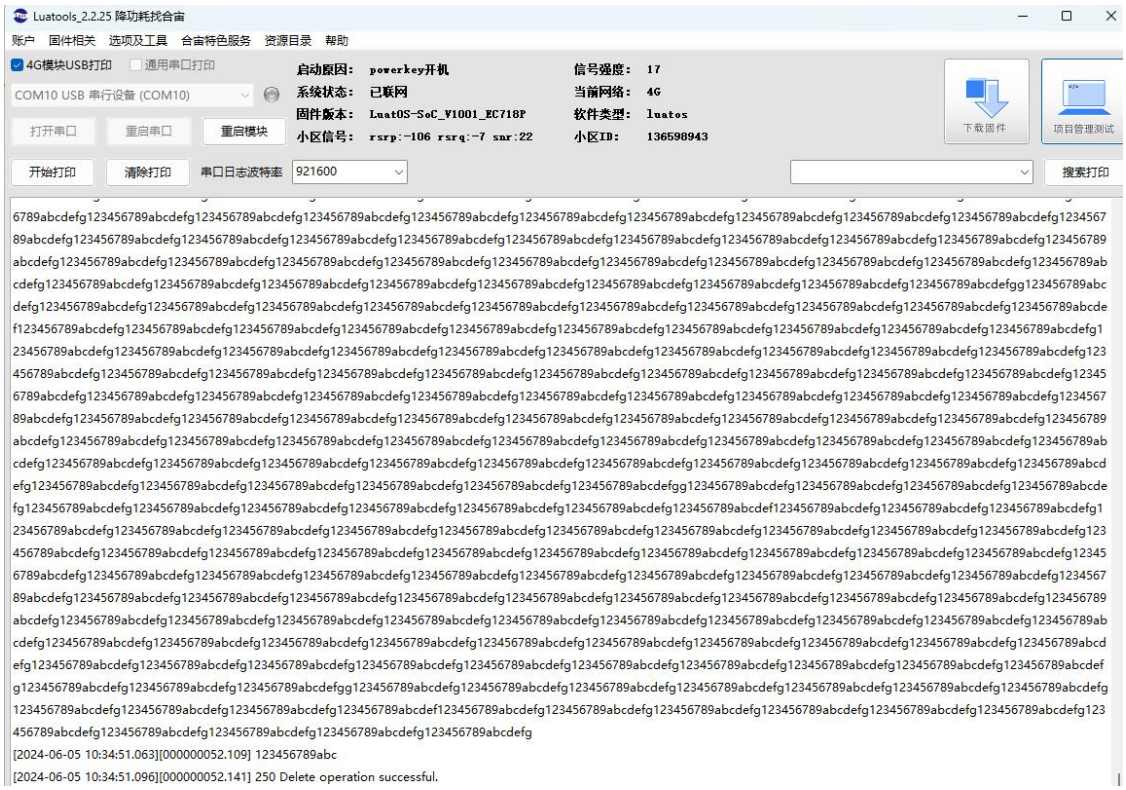


对应模块 log

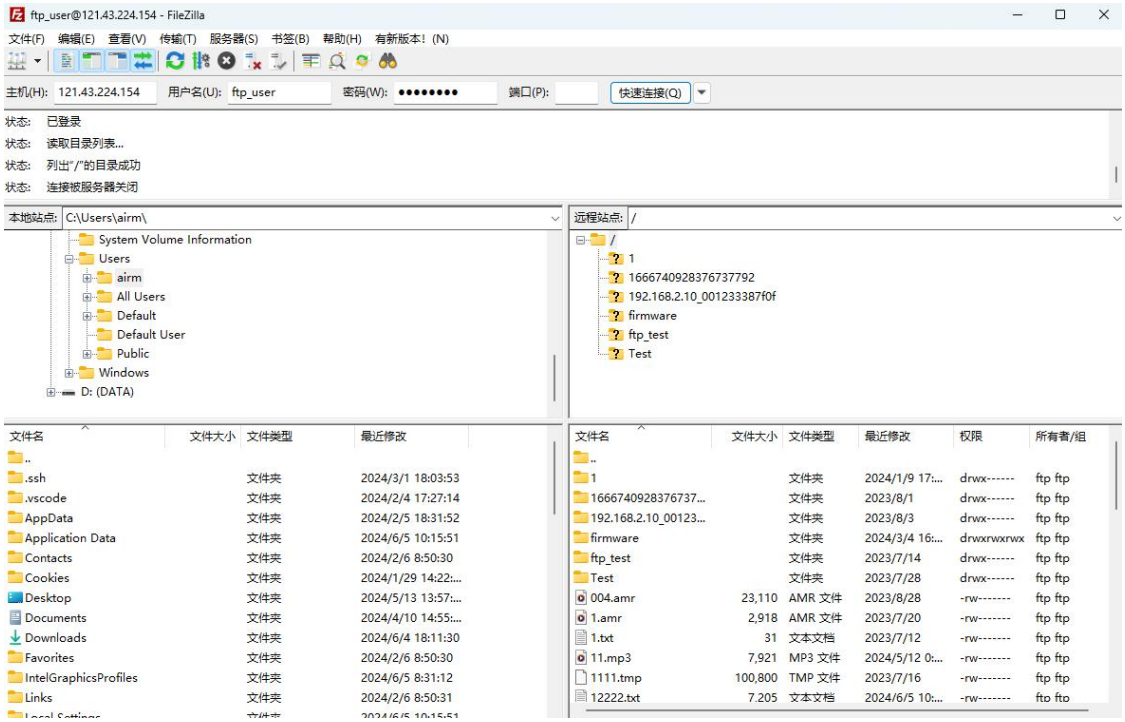
Luatools_2.2.25 降功耗找合宙
账户 固件相关 选项及工具 合宙特色服务 资源目录 帮助
4G模块USB打印
通用串口打印
COM10 USB 串行设备 (COM10)
打开串口
重启串口
重启模块
启动原因: powerkey开机
系统状态: 已联网
固件版本: LuatOS-SoC_V1001_EC718P
小区信号: rsrp:-106 rsrq:-7 snr:22
信号强度: 17
当前网络: 4G
软件类型: luatos
小区ID: 136598943
下载固件
项目管理测试
开始打印
清除打印
串口日志波特率
921600
搜索打印
[2024-06-05 10:34:33.201][000000034.246] true
[2024-06-05 10:34:33.372][000000034.412] 221 Goodbye.
[2024-06-05 10:34:33.380][000000034.412] I/user.meminfo 242176 66308 90376
[2024-06-05 10:34:49.372][000000050.413] I/user.ftp 启动
[2024-06-05 10:34:49.890][000000050.940] true
[2024-06-05 10:34:49.967][000000051.015] 200 NOOP ok.
[2024-06-05 10:34:50.029][000000051.076] 215 UNIX Type: L8
[2024-06-05 10:34:50.108][000000051.160] 200 Switching to Binary mode.
[2024-06-05 10:34:50.185][000000051.230] 257 "/"
[2024-06-05 10:34:50.246][000000051.290] 257 "/QWER" created
[2024-06-05 10:34:50.308][000000051.360] 250 Directory successfully changed.
[2024-06-05 10:34:50.385][000000051.430] 250 Directory successfully changed.
[2024-06-05 10:34:50.463][000000051.505] 250 Remove directory operation successful.


Luatools_2.2.25 降功耗找合宙
账户 固件相关 选项及工具 合宙特色服务 资源目录 帮助
4G模块USB打印
通用串口打印
COM10 USB 串行设备 (COM10)
打开串口
重启串口
重启模块
启动原因: powerkey开机
系统状态: 已联网
固件版本: LuatOS-SoC_V1001_EC718P
小区信号: rsrp:-106 rsrq:-7 snr:22
信号强度: 17
当前网络: 4G
软件类型: luatos
小区ID: 136598943
下载固件
项目管理测试
开始打印
清除打印
串口日志波特率
921600
搜索打印
[2024-06-05 10:34:50.712][000000051.760] 250 Remove d
[2024-06-05 10:34:50.718][000000051.761] -rw------- 1 ftp ftp 23110 Aug 28 2023 004.amr
drwx------ 2 ftp ftp 4096 Jan 09 09:56 1
-rw------- 1 ftp ftp 2918 Jul 20 2023 1.amr
-rw------- 1 ftp ftp 31 Jul 12 2023 1.txt
-rw------- 1 ftp ftp 7921 May 11 16:22 11.mp3
-rw------- 1 ftp ftp 100800 Jul 16 2023 1111.tmp
-rw------- 1 ftp ftp 7205 Jun 05 02:34 12222.txt
-rw------- 1 ftp ftp 62 Oct 26 2023 1222200000000000000.txt
-rw------- 1 ftp ftp 0 Oct 26 2023 12222211111111111111111111111111.txt
-rw------- 1 ftp ftp 7205 Aug 17 2023 122224.txt
-rw------- 1 ftp ftp 11870 Jul 27 2023 1234567.txt
drwx------ 2 ftp ftp 4096 Aug 01 2023 1666740928376737792
drwx------ 3 ftp ftp 4096 Aug 03 2023 192.168.2.10_001233387f0f
-rw------- 1 ftp ftp 31212 Jul 20 2023 1_19K.mp3
-rw------- 1 ftp ftp 2470 Jul 20 2023 2.amr
-rw------- 1 ftp ftp 25160 Jul 25 2023 2.mp3
-rw------- 1 ftp ftp 25160 May 12 16:16 2315153515.mp3
-rw------- 1 ftp ftp 2246 Jul 20 2023 3.amr
-rw------- 1 ftp ftp 23070 Jul 25 2023 3.mp3
-rw------- 1 ftp ftp 12390 Jul 20 2023 5.amr
-rw------- 1 ftp ftp 62676 Jul 25 2023 5.mp3
-rw------- 1 ftp ftp 23894 Dec 19 04:35 5000.txt
-rw------- 1 ftp ftp 31537 Jul 26 2023 55.mp3
-rw------- 1 ftp ftp 0 Oct 26 2023 8888888888.txt
-rw------- 1 ftp ftp 29916 Dec 18 09:27 ApphzShhYL01.bin
-rw------- 1 ftp ftp 39808 Nov 30 2023 CVP_LX_APP.BIN
-rw------- 1 ftp ftp 26136 Aug 31 2023 JCC-V4.0A1T9.bin
drwx------ 2 ftp ftp 4096 Jul 28 2023 Test


Luatools_2.2.25 降功耗找合宙
账户 固件相关 选项及工具 合宙特色服务 资源目录 帮助
4G模块USB打印
通用串口打印
COM10 USB 串行设备 (COM10)
打开串口
重启串口
重启模块
启动原因: powerkey开机
系统状态: 已联网
固件版本: LuatOS-SoC_V1001_EC718P
小区信号: rsrp:-106 rsrq:-7 snr:22
信号强度: 17
当前网络: 4G
软件类型: luatos
小区ID: 136598943
下载固件
项目管理测试
开始打印
清除打印
串口日志波特率
921600
搜索打印
-rw------- 1 ftp ftp 0 Jun 01 08:01 capture.jpg
drwxrwxrwx 2 ftp ftp 4096 Mar 04 08:28 firmware
drwx------ 2 ftp ftp 4096 Jul 14 2023 ftp_test
-rw------- 1 ftp ftp 126002 Jul 16 2023 jpeg.tmp
-rw------- 1 ftp ftp 7921 Jul 20 2023 num1.mp3
-rw------- 1 ftp ftp 6769 Jul 20 2023 num2.mp3
-rw------- 1 ftp ftp 6193 Jul 20 2023 num3.mp3
-rw------- 1 ftp ftp 31537 Jul 20 2023 num5.mp3
-rw------- 1 ftp ftp 69200 Nov 22 2023 ota_test.bin
-rw------- 1 ftp ftp 982 May 18 04:18 test123.txt
-rw------- 1 ftp ftp 39968 Apr 11 2023 test_16k.mp3
-rw------- 1 ftp ftp 0 Oct 26 2023 test_20.mp3
-rw------- 1 ftp ftp 79496 Apr 11 2023 test_32k.mp3
[2024-06-05 10:34:50.723][000000051.761] -rw-------
[2024-06-05 10:34:51.051][000000052.100] true
[2024-06-05 10:34:51.054][000000052.105] I/user.fs writed data
[2024-06-05 10:34:51.056][000000052.106]


Luatools_2.2.25 降功耗找合宙
账户 固件相关 选项及工具 合宙特色服务 资源目录 帮助
4G模块USB打印
通用串口打印
COM10 USB 串行设备 (COM10)
打开串口
重启串口
重启模块
启动原因: powerkey开机
系统状态: 已联网
固件版本: LuatOS-SoC_V1001_EC718P
小区信号: rsrp:-106 rsrq:-7 snr:22
信号强度: 17
当前网络: 4G
软件类型: luatos
小区ID: 136598943
下载固件
项目管理测试
开始打印
清除打印
串口日志波特率
921600
搜索打印
[2024-06-05 10:34:51.063][000000052.109] 123456789abc
[2024-06-05 10:34:51.096][000000052.141] 250 Delete operation successful.

FZ 工具显示



四．代码解析

完整脚本代码点击右侧查看　demo/ftp/main.lua

ftp.login(nil,"121.43.224.154",21,"ftp_user","3QujbiMG").wait()：登录 FTP 客户端

ftp.command("NOOP").wait()：等待，此命令不产生实际动作，它仅使服务器返回 OK

ftp.command("SYST").wait()：查看服务器信息

ftp.command("TYPE I").wait()：文件传输类型为二进制

ftp.command("PWD").wait()：显示工作目录

ftp.command("MKD QWER").wait()：创建目录

ftp.command("CWD /QWER").wait()：切换目录

ftp.command("CDUP").wait()：返回上级工作目录

ftp.command("RMD QWER").wait()：删除目录

ftp.command("LIST").wait()：显示目录下文件

ftp.pull("/122224.txt","/122224.txt").wait()：FTP 文件下载（本地文件，服务器文件）

ftp.command("DELE /12222.txt").wait()：删除文件

ftp.close().wait()：FTP 客户端关闭

```lua
while true do
    sys.wait(1000)
    log.info("ftp 启动")
    print(ftp.login(nil,"121.43.224.154",21,"ftp_user","3QujbiMG").wait())

    print(ftp.command("NOOP").wait())
    print(ftp.command("SYST").wait())

    print(ftp.command("TYPE I").wait())
    print(ftp.command("PWD").wait())
    print(ftp.command("MKD QWER").wait())
    print(ftp.command("CWD /QWER").wait())

    print(ftp.command("CDUP").wait())
    print(ftp.command("RMD QWER").wait())

    print(ftp.command("LIST").wait())

    -- io.writeFile("/1222.txt", "23noianfdiasfhnpqw39fhawe;fuibnnpw3fheaios;fna;osfhisao;fadsfl")
    -- print(ftp.push("/1222.txt","/12222.txt").wait())

    print(ftp.pull("/122224.txt","/122224.txt").wait())

    local f = io.open("/122224.txt", "r")
    if f then
        local data = f:read("*a")
        f:close()
        log.info("fs", "writed data", data)
    else
        log.info("fs", "open file for read failed")
    end

    print(ftp.command("DELE /12222.txt").wait())
    print(ftp.push("/122224.txt","/12222.txt").wait())
    print(ftp.close().wait())
    log.info("meminfo", rtos.meminfo("sys"))
    sys.wait(15000)
end
```

五．常见问题

关于不支持主动模式：

模块 ftp 默认是被动模式，不支持主动模式，无法设置主动模式，公网下要想模块正常连接服务器，服务器肯定不能设置为主动模式，否则哪家模块都不能用。

10.5.7 阿里云开发应用指南

一．概述

物联网平台提供安全可靠的设备连接通信能力，支持设备数据采集上云，规则引擎流转数据和云端数据下发设备端。此外，也提供方便快捷的设备管理能力，支持物模型定义，数据结构化存储，和远程调试、监控、运维。物联网平台文档

消息通信流程

二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

三. 演示步骤

本文以 demo/aliyun 这个 DEMO 为例做演示

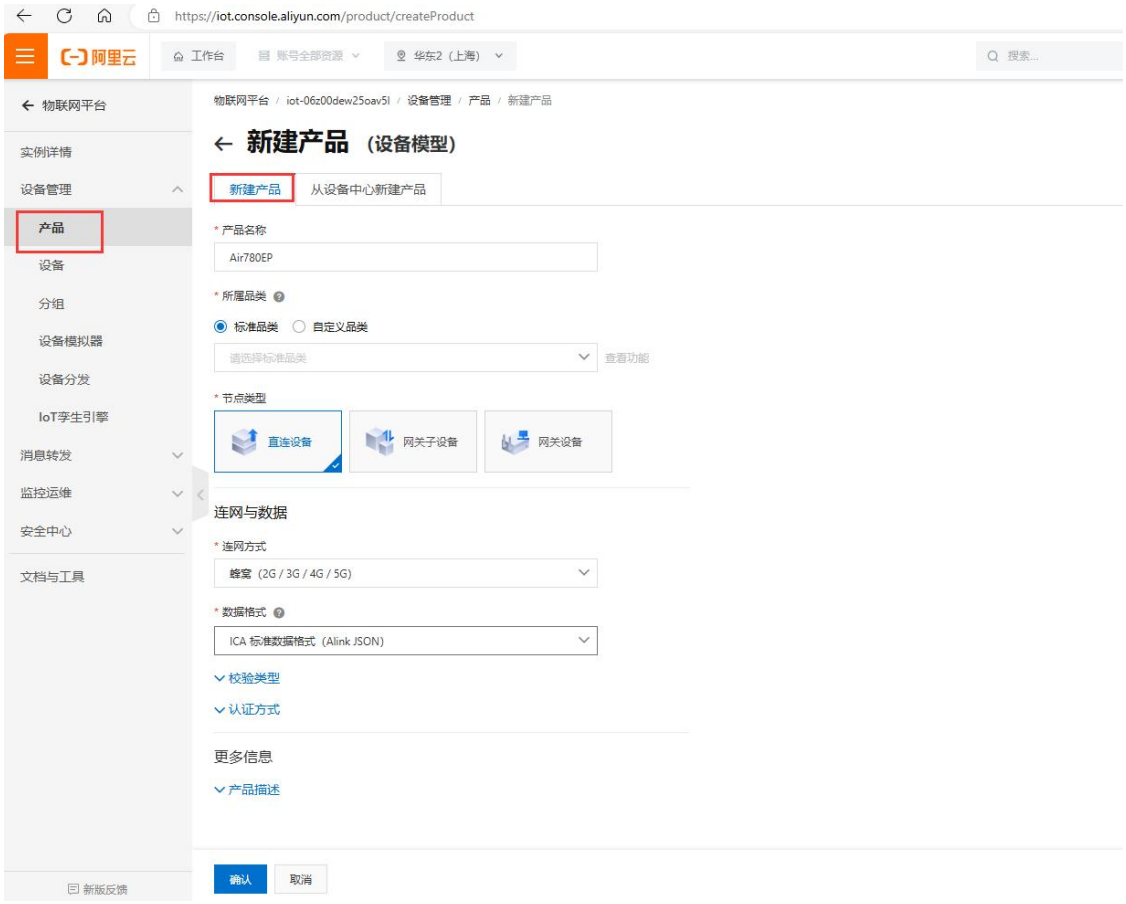
阿里云操作

产品操作

打开阿里云找到物联网平台，开通业务后进入控制台。

点开设备管理的产品页面，点击新建产品。根据需求和图示说明创建产品。

具体详细介绍见阿里云页面

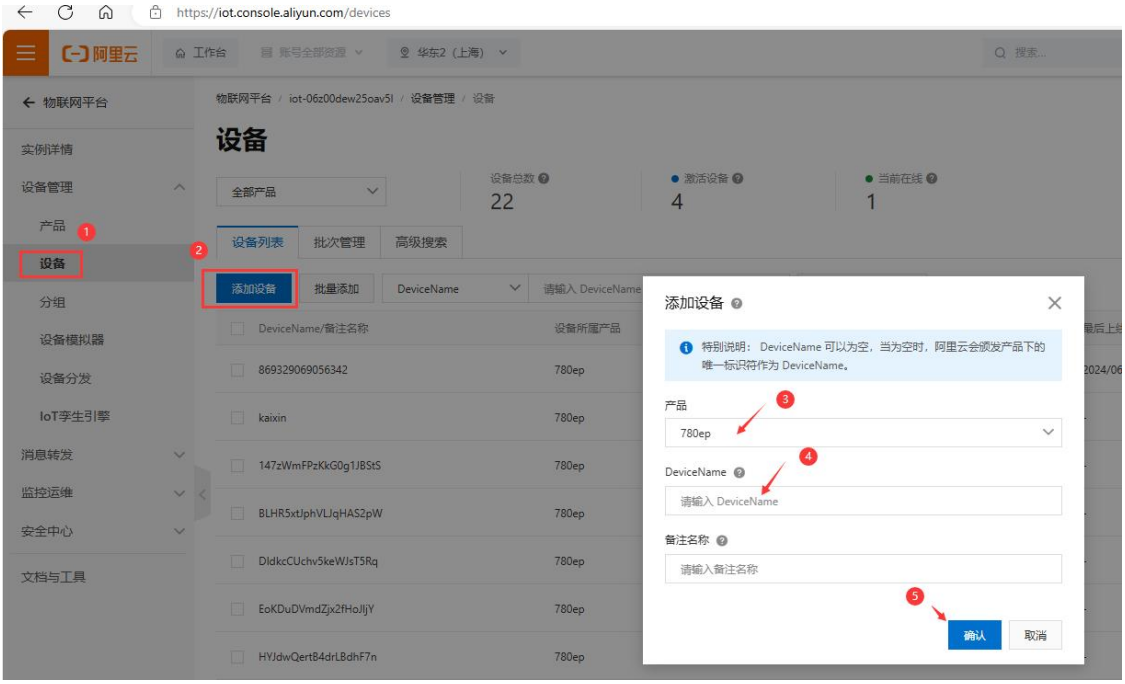


设备操作

创建产品完成后就可以进入设备页面添加设备，在对应产品页面进入设备管理，按照提示添加设备

（在做正式产品时建议使用 imei 为 devicename，方便后期维护）

阿里云设备创建



模块操作

一机一密

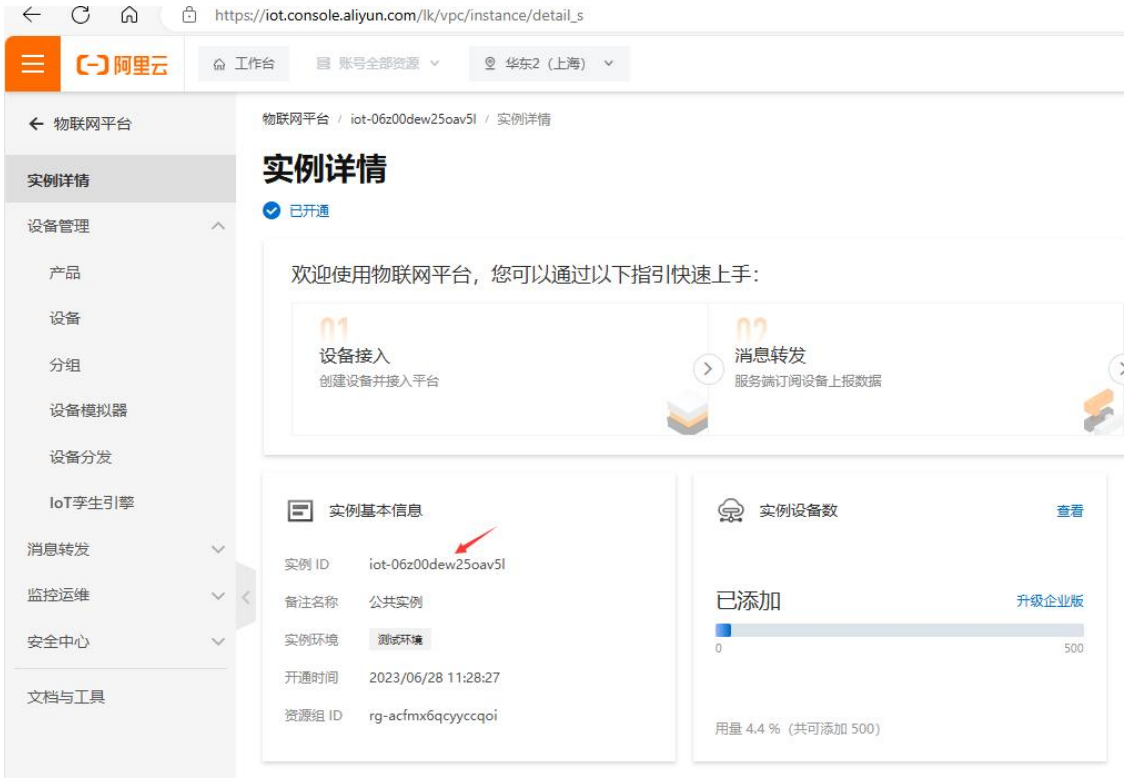
main.lua 做如下修改

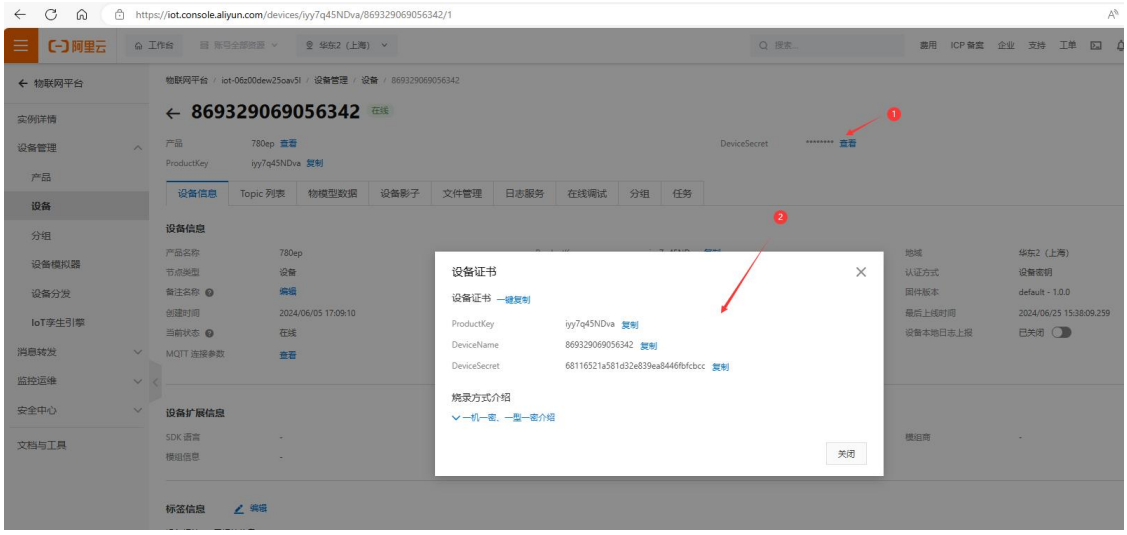
```
PROJECT = "aliyundemo"
VERSION = "1.0.0"
local sys = require "sys"

-- 一型一密优先使用fskv存储密钥
if fskv then
    fskv.init()
end

-- 联网函数，非必须
require "netready"

-- 阿里云事件处理函数
require "testEvt"

-- 一机一密的演示
require "testYjym"

-- 一型一密的演示，一定要修改成自己的信息才能跑通!!!
-- require "testYxym"

-- aliyun+低功耗的演示，需要开启testYjym或者testYxym
-- require "testPm"

-- 用户代码已结束---------------------------------------------
-- 结尾总是这一句
sys.run()
-- sys.run()之后后面不要加任何语句!!!!!
```

一机一密修改 testYjym.lua 里如下数据为自己的参数

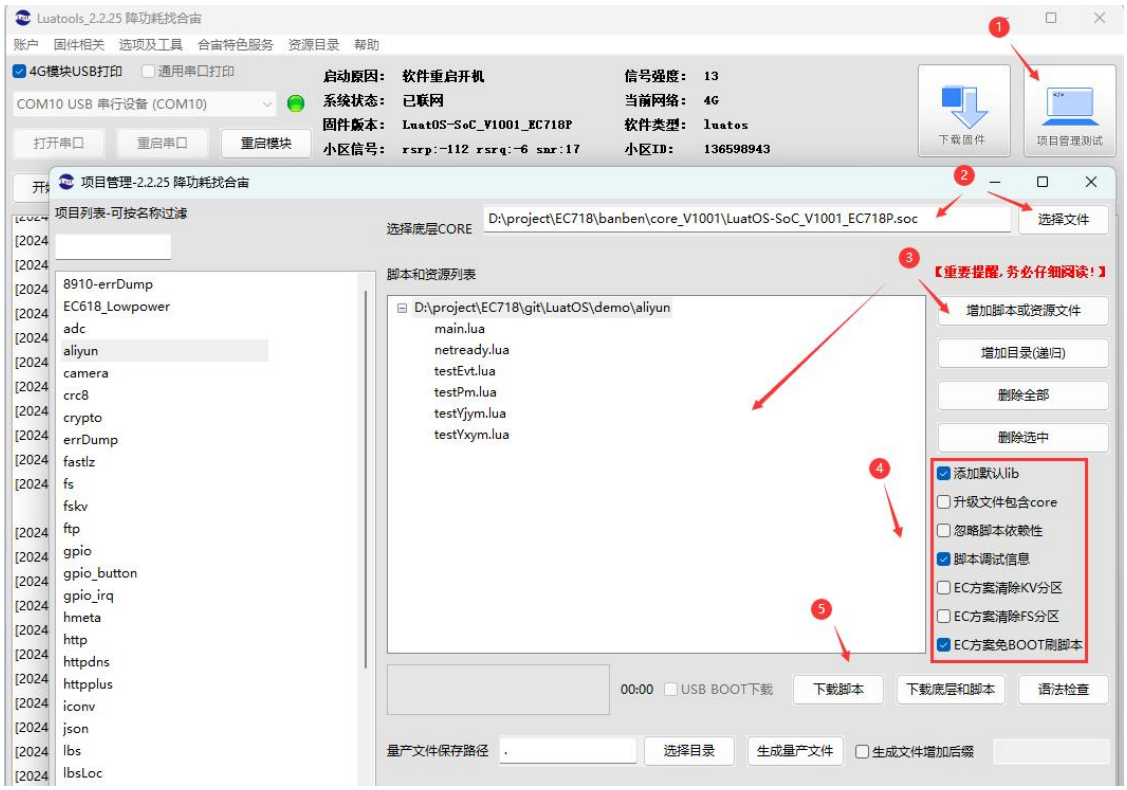
```
--根据自己的服务器修改以下参数
-- 阿里云资料：https://help.aliyun.com/document_detail/147356.htm?spm=a2c4g.73742.0.0.4782214ch6jkXb#section-rtu-6kn-kru
local tPara = {
    -- 一机一密
    -- 设备名名称，必须唯一
    DeviceName = "869329069056342",
    -- 产品key，在产品详情页面
    ProductKey = "iyy7q45NDva",      --产品key
    --设备密钥,一型一密就不填，一机一密(预注册)必须填
    DeviceSecret = "68116521a581d32e839ea8446fbfcbcc", --设备secret
    -- 填写实例id，在实例详情页面，如果是旧的公共实例，请填RegionId参数
    InstanceId = "iot-06z00dew25oav5l",
    RegionId = "cn-shanghai",
    --是否使用ssl加密连接
    mqtt_isssl = false,
}
```

这里对应的数据在云平台上复制

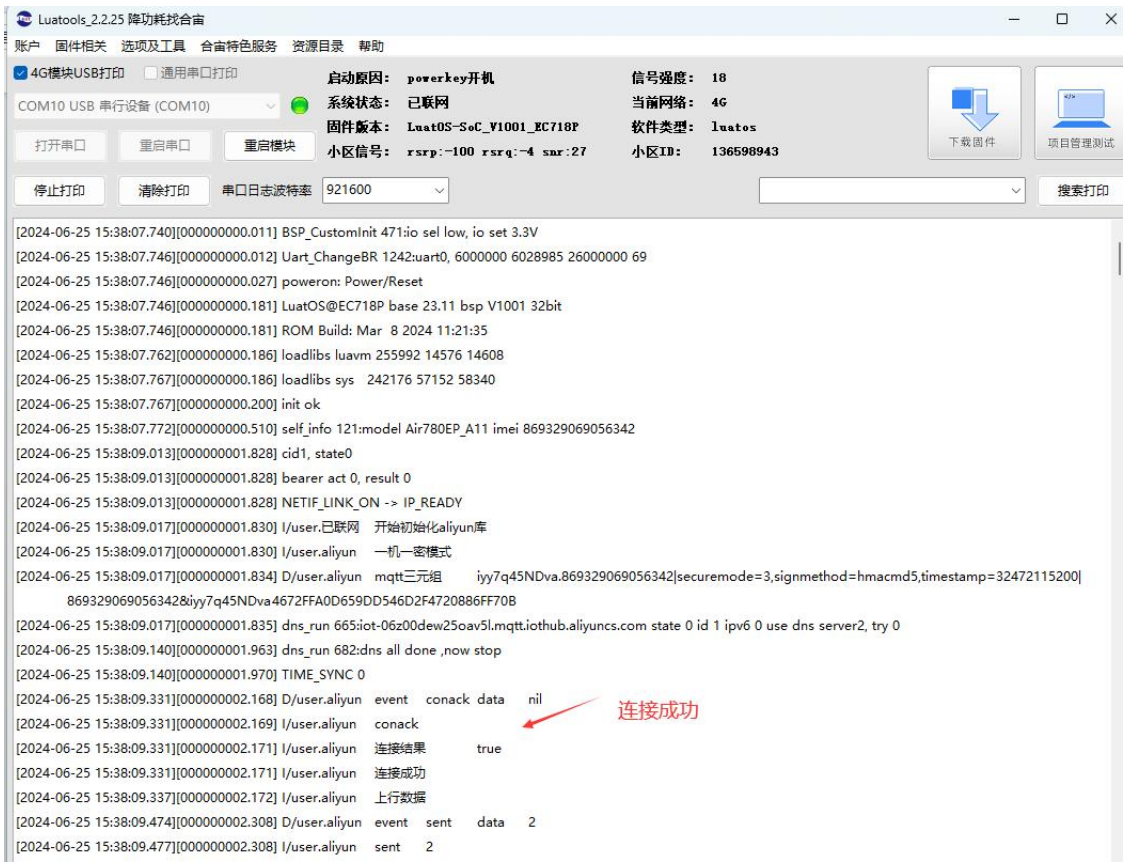




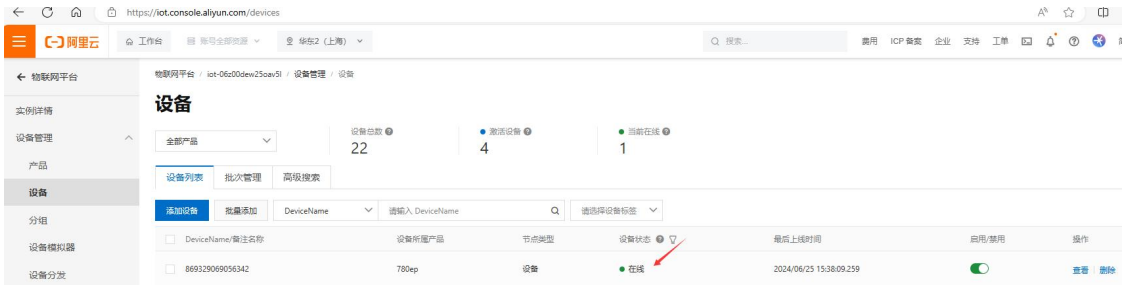
直接下载对应 core 和修改后的 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块对应 log



云平台显示



一型一密

除了需要添加产品和设备还需要在阿里云打开动态注册开关

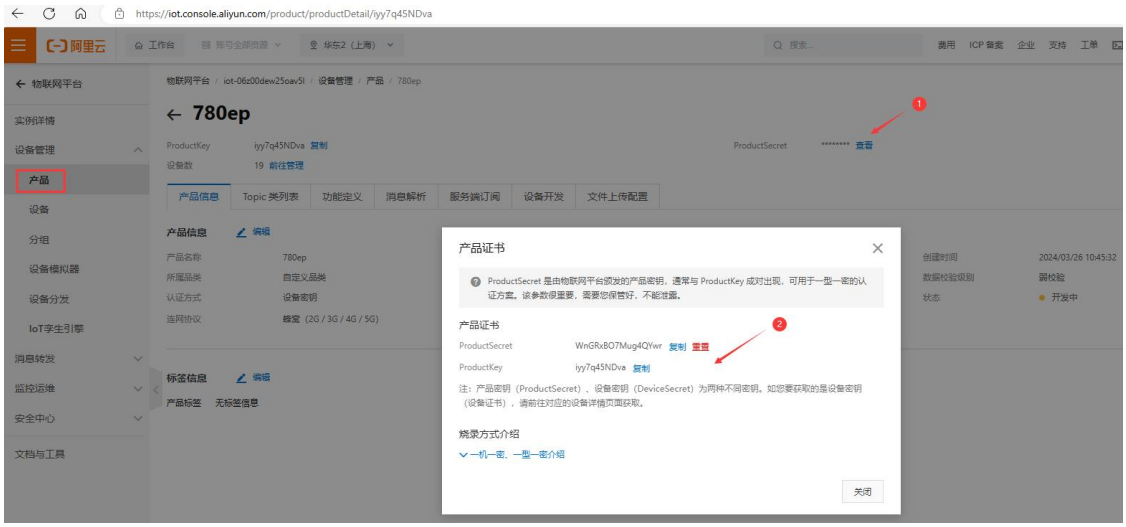
main.lua 做如下修改

```
PROJECT = "aliyundemo"
VERSION = "1.0.0"
local sys = require "sys"

-- 一型一密优先使用fskv存储密钥
if fskv then
    fskv.init()
end

-- 联网函数,非必须
require "netready"

-- 阿里云事件处理函数
require "testEvt"

-- 一机一密的演示
--require "testYjym"

-- 一型一密的演示，一定要修改成自己的信息才能跑通!!!
require "testYxym"

-- aliyun+低功耗的演示，需要开启testYjym或者testYxym
-- require "testPm"

-- 用户代码已结束---------------------------------------------
-- 结尾总是这一句
sys.run()
-- sys.run()之后后面不要加任何语句!!!!!
```

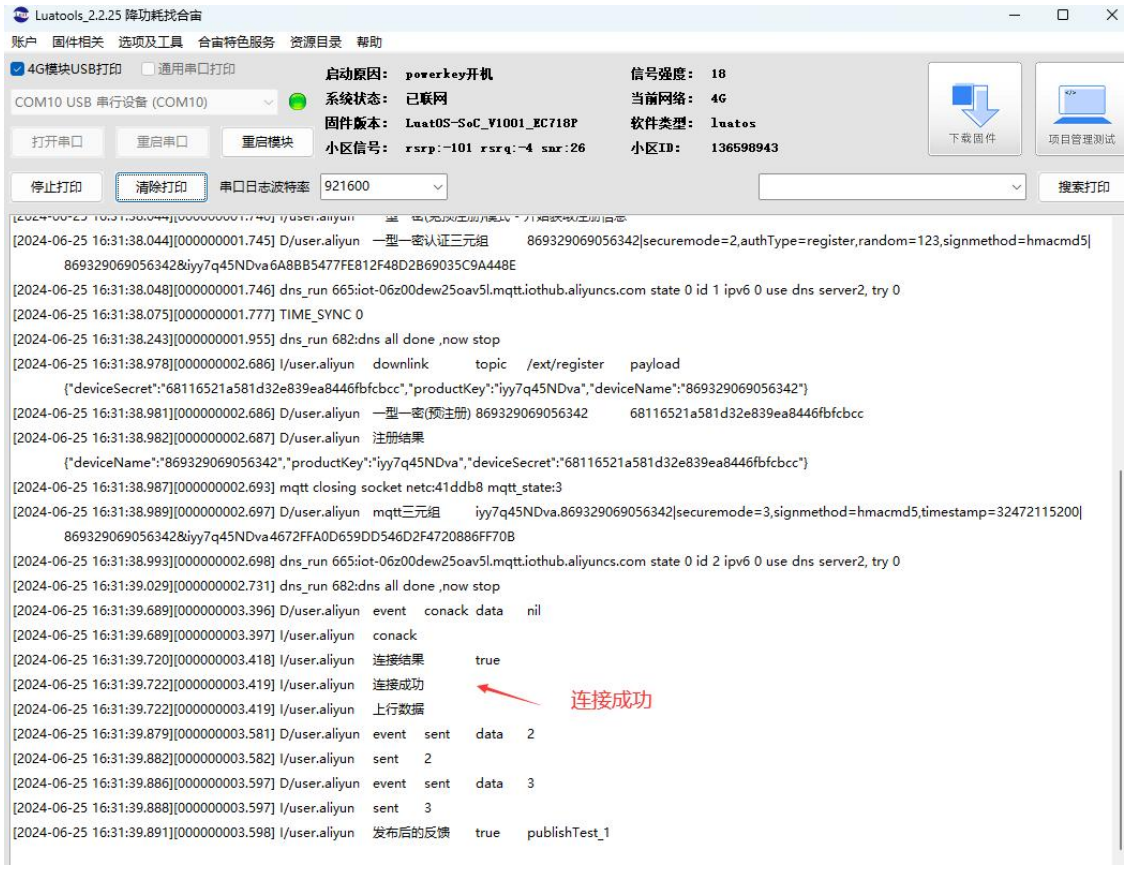
一型一密修改 testYxym.lua 里如下数据为自己的参数

```
--根据自己的服务器修改以下参数
-- 阿里云资料：https://help.aliyun.com/document_detail/147356.htm?spm=a2c4g.73742.0.0.4782214ch6jkXb#section-rtu-6kn-kru
tPara = {
    -- 一型一密 - ProductSecret要填产品secret
    -- 一型一密 分2种：预注册和免预注册
    -- 公共实例只支持 预注册，Registration 填false
    -- 企业实例支持 预注册 和 免预注册，如需使用免预注册，Registration 填true，否则填false
    Registration = false,
    -- 设备名名称，必须唯一
    DeviceName = "869329069056342",
    -- 产品key，在产品详情页面
    ProductKey = "iyy7q45NDva",     --产品key
    --产品secret,一型一密就需要填
    ProductSecret = "WnGRxBO7Mug4QYwr",
    -- 填写实例id，在实例详情页面，如果是旧的公共实例，请填host参数
    InstanceId = "iot-06z00dew25oav5l",
    RegionId = "cn-shanghai",
    --是否使用ssl加密连接
    mqtt_isssl = true,
}
```

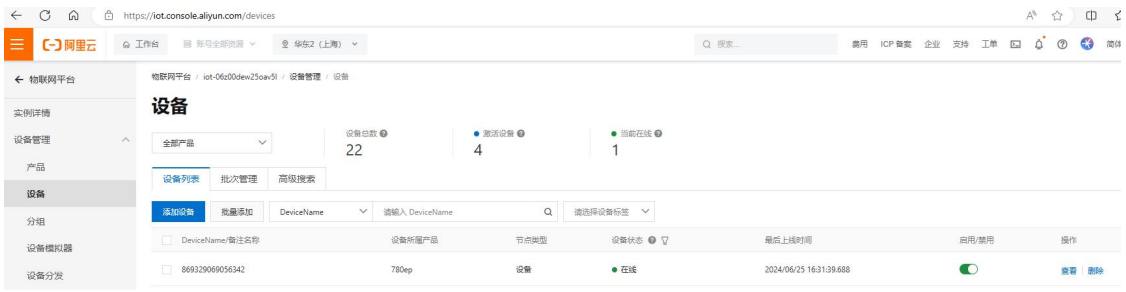
这里对应的数据在云平台上复制



模块对应 log



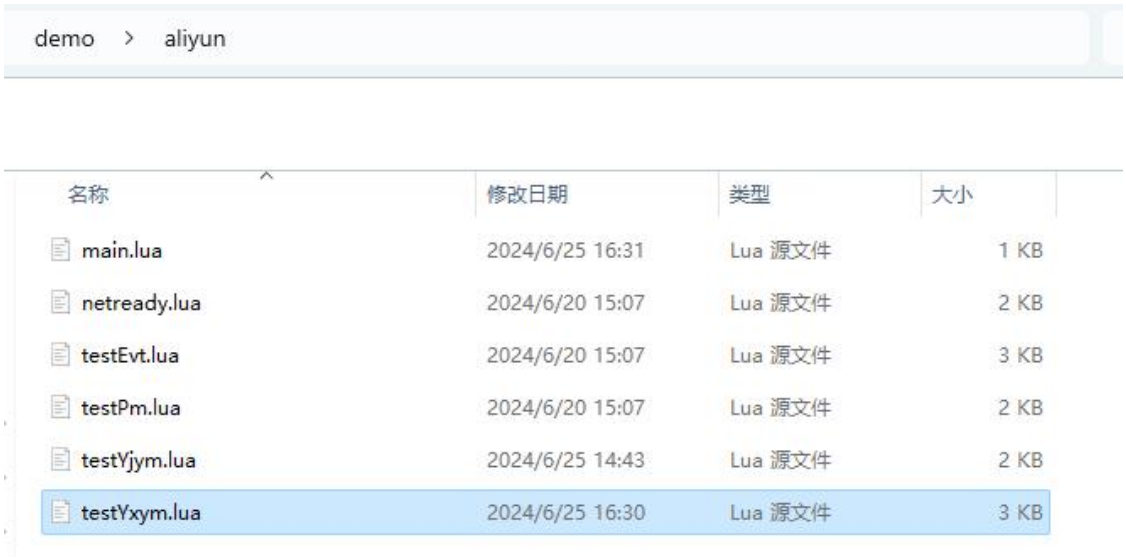
云平台显示



四．代码解析

点击右侧查看 AliYun API 说明：AliYun 阿里云物联网平台

完整脚本代码点击右侧查看　demo/aliyun



main.lua 脚本对应的代码如下所示：

主要是 require 不同的演示模块

```
PROJECT = "aliyundemo"
VERSION = "1.0.0"
local sys = require "sys"

-- 一型一密优先使用fskv存储密钥
if fskv then
    fskv.init()
end

-- 联网函数,非必须
-- require "netready"

-- 阿里云事件处理函数
require "testEvt"

-- 一机一密的演示
require "testYjym"

-- 一型一密的演示, 一定要修改成自己的信息才能跑通!!!
-- require "testYxym"

-- aliyun+低功耗的演示, 需要开启testYjym或者testYxym
-- require "testPm"

-- 用户代码已结束---------------------------------------------
-- 结尾总是这一句
sys.run()
-- sys.run()之后后面不要加任何语句!!!!!
```

netready.lua 脚本对应代码如下所示：

sys.waitUntil("IP_READY")：等待驻网成功

sys.publish("net_ready")：发布"net_ready"消息

```
-- 统一联网函数
sys.taskInit(function()
    ------------------------------
    -- 统一联网函数, 可自行删减
    ------------------------------
    if wlan and wlan.connect then
        -- wifi 联网, ESP32系列均支持
        local ssid = "luatos1234"
        local password = "12341234"
        log.info("wifi", ssid, password)
        -- TODO 改成自动配网
        wlan.init()
        wlan.setMode(wlan.STATION) -- 默认也是这个模式,不调用也可以
        wlan.connect(ssid, password, 1)
    elseif mobile then
        -- Air780E/Air600E系列
        --mobile.simid(2) -- 自动切换SIM卡
        -- LED = gpio.setup(27, 0, gpio.PULLUP)
        -- device_id = mobile.imei()
    elseif w5500 then
        -- w5500 以太网, 当前仅Air105支持
        w5500.init(spi.HSPI_0, 24000000, pin.PC14, pin.PC01, pin.PC00)
        w5500.config() --默认是DHCP模式
        w5500.bind(socket.ETH0)
        -- LED = gpio.setup(62, 0, gpio.PULLUP)
    elseif socket or mqtt then
        -- 适配的socket库也OK
        -- 没有其他操作, 单纯给个注释说明
    else
        -- 其他不认识的bsp, 循环提示一下吧
        while 1 do
            sys.wait(1000)
            log.info("bsp", "本bsp可能未适配网络层, 请查证")
        end
    end
    -- 默认都等到联网成功
    sys.waitUntil("IP_READY")
    sys.publish("net_ready")
end)
```

testEvt.lua 脚本对应代码如下所示：

publishTest()：发布一条 QOS 为 1 的消息

publishTestCb：发布 1 条 qos 为 1 的消息后收到 PUBACK 的回调函数

aliyun.publish：发布一条消息

```lua
--[[
函数名：pubqosltestackcb
功能  ：发布1条qos为1的消息后收到PUBACK的回调函数
参数  ：
        usertag：调用mqttclient:publish时传入的usertag
        result：任意数字表示发布成功，nil表示失败
返回值：无
]]
local publishCnt = 1
local function publishTestCb(result,para)
    log.info("aliyun", "发布后的反馈", result,para)
    sys.timerStart(publishTest,20000)
    publishCnt = publishCnt+1
end

--发布一条QOS为1的消息
function publishTest()
    --注意：在此处自己去控制payload的内容编码，aLiYun库中不会对payload的内容做任何编码转换
    -- aliyun.publish(topic,qos,payload,cbFnc,cbPara)
    log.info("aliyun", "上行数据")
    aliyun.publish("/"..aliyun.opts.ProductKey.."/"..aliyun.opts.DeviceName.."/user/get",1,"LUATOS_CESHI",publishTestCb,"publishTest_"..publishCnt)
end
```

rcvCbFnc：数据接收的处理函数

connectCbFnc：连接结果的处理函数

aliyun.subscribe：根据自己的项目需要订阅主题

aliyun.on("connect",connectCbFnc)：注册连接事件的处理函数

aliyun.on("receive",rcvCbFnc)：注册数据接收事件的处理函数

```lua
---数据接收的处理函数
-- @string topic，UTF8编码的消息主题
-- @string payload，原始编码的消息负载
local function rcvCbFnc(topic,payload,qos,retain,dup)
    log.info("aliyun", "收到下行数据", topic,payload,qos,retain,dup)
end

--- 连接结果的处理函数
-- @bool result，连接结果，true表示连接成功，false或者nil表示连接失败
local function connectCbFnc(result)
    log.info("aliyun","连接结果", result)
    if result then
        sys.publish("aliyun_ready")
        log.info("aliyun", "连接成功")
        --订阅主题
        --根据自己的项目需要订阅主题
        -- aliyun.subscribe(topic,qos)
        -- aliyun.subscribe("/".. aliyun.opts.ProductKey.."/".. aliyun.opts.DeviceName.."/user/ceshi",1)

        --PUBLISH消息测试
        publishTest()
    else
        log.warn("aliyun", "连接失败")
    end
end

-- 连接状态的处理函数
aliyun.on("connect",connectCbFnc)

-- 数据接收的处理函数
aliyun.on("receive",rcvCbFnc)
```

testPm.lua 脚本对应的代码如下所示：

sys.waitUntil("aliyun_ready")：等待阿里云连接成功

sys.wait(5000)：等待 5 秒

pm.power(pm.USB, false)：关闭 USB 电源

pm.force(pm.LIGHT)：强制进入 LIGHT 休眠模式

```lua
-- 低功耗演示
sys.taskInit(function()
    sys.waitUntil("aliyun_ready")
    log.info("aliyun.pm", "阿里云已经连接成功，5秒后请求进入低功耗模式，USB功能会断开")
    sys.wait(5000)
    local bsp = rtos.bsp():upper()
    -- 进入低功耗模式
    if bsp == "EC618" then
        log.info("aliyun.pm", "EC618方案进入低功耗模式")
        -- gpio.setup(23,nil)
        -- gpio.close(33)
        -- mobile.rtime(2)  -- RRC快速释放减少connect时间能大幅降低功耗，但是会带来可能得离线风险，可选择延迟时间或者不用
        pm.power(pm.USB, false)
        pm.force(pm.LIGHT)
    elseif bsp == "EC718P" or bsp == "EC718PV" then
        log.info("aliyun.pm", "EC718P/EC718PV方案进入低功耗模式")
        -- mobile.rtime(2) -- RRC快速释放减少connect时间能大幅降低功耗，但是会带来可能得离线风险，可选择延迟时间或者不用
        pm.power(pm.USB, false)
        pm.force(pm.LIGHT)
    elseif bsp == "AIR101" or bsp == "AIR601" or bsp == "AIR103" then
        log.info("aliyun.pm", "XT804方案进入低功耗模式")
        while 1 do
            pm.dtimerStart(0, 30000)
            pm.request(pm.LIGHT)
            sys.wait(30000)
        end
    end
end)
```

testYjym.lua：脚本对应的代码如下所示：

aliyun.setup：配置阿里云一机一密的产品信息和设备信息

```lua
local sys = require "sys"
local aliyun = require "aliyun"

--根据自己的服务器修改以下参数
-- 阿里云资料：https://help.aliyun.com/document_detail/147356.htm?spm=a2c4g.73742.0.0.4782214ch6jkXb#section-rtu-6kn-kru
local tPara = {
    -- 一机一密
    -- 设备名名称，必须唯一
    DeviceName = "869329069056342",
    -- 产品key，在产品详情页面
    ProductKey = "iyy7q45NDva",      --产品key
    --设备密钥,一型一密就不填，一机一密(预注册)必须填
    DeviceSecret = "68116521a581d32e839ea8446fbfcbcc", --设备secret
    -- 填写实例id，在实例详情页面，如果是旧的公共实例，请填RegionId参数
    InstanceId = "iot-06z00dew25oav5l",
    RegionId = "cn-shanghai",
    --是否使用ssl加密连接
    mqtt_isssl = false,
}

-- 等待联网，然后初始化aliyun库
sys.taskInit(function()
    -- sys.waitUntil("IP_READY")
    sys.waitUntil("net_ready")

    log.info("已联网", "开始初始化aliyun库")
    aliyun.setup(tPara)
end)
```

testYxym.lua：脚本对应的代码如下所示：

aliyun.store()：获取或存储注册信息

aliyun.setup：配置阿里云一型一密的产品信息和设备信息

```lua
--根据自己的服务器修改以下参数
-- 阿里云资料：https://help.aliyun.com/document_detail/147356.htm?spm=a2c4g.73742.0.0.4782214ch6jkXb#section-rtu-6kn-kru
tPara = {
    -- 一型一密 - ProductSecret要填产品secret
    -- 一型一密 分2种：预注册和免预注册
    -- 公共实例只支持 预注册，Registration 填false
    -- 企业实例支持 预注册 和 免预注册，如需使用免预注册，Registration 填true，否则填false
    Registration = false,
    -- 设备名名称，必须唯一
    DeviceName = "869329069056342",
    -- 产品key，在产品详情页面
    ProductKey = "iyy7q45NDva",      --产品key
    --产品secret,一型一密就需要填
    ProductSecret = "WnGRxBO7Mug4QYwr",
    -- 填写实例id，在实例详情页面，如果是旧的公共实例，请填host参数
    InstanceId = "iot-06z00dew25oav5l",
    RegionId = "cn-shanghai",
    --是否使用ssl加密连接
    mqtt_isssl = true,
}
```

```
--根据掉电不消失的kv文件区来储存的deviceSecret,clientid,deviceToken来判断是进行正常连接还是掉电重连
sys.taskInit(function()
    sys.waitUntil("IP_READY")
    log.info("已联网", "开始初始化aliyun库")

    local store = aliyun.store()

    --判断是否是同一三元组，不是的话就重新连接
    if store.deviceName ~= tPara.DeviceName or store.productKey ~= tPara.ProductKey then
        -- 清除fskv区的注册信息
        if fskv then
            fskv.del("DeviceName")
            fskv.del("ProductKey")
            fskv.del("deviceToken")
            fskv.del("deviceSecret")
            fskv.del("clientid")
        else
            os.remove("/alireg.json")
        end
        store = {}
    end

    if store.clientid and store.deviceToken and #store.clientid > 0 and #store.deviceToken > 0 then
        tPara.clientId = store.clientid
        tPara.deviceToken = store.deviceToken
        tPara.reginfo = true
    elseif store.deviceSecret and #store.deviceSecret > 0 then
        tPara.deviceSecret = store.deviceSecret
        tPara.reginfo = true
    end
    aliyun.setup(tPara)
end)
```

五．常见问题

为什么阿里云连接失败

1. 确认模块是否正常联网
2. 确认产品信息和设备信息是否正确

10.5.8 onenet 开发应用指南

一．概述

文章介绍 LuatOS-Air 连接 OneNET Studio。官网地址：https://open.iot.10086.cn/

二．材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

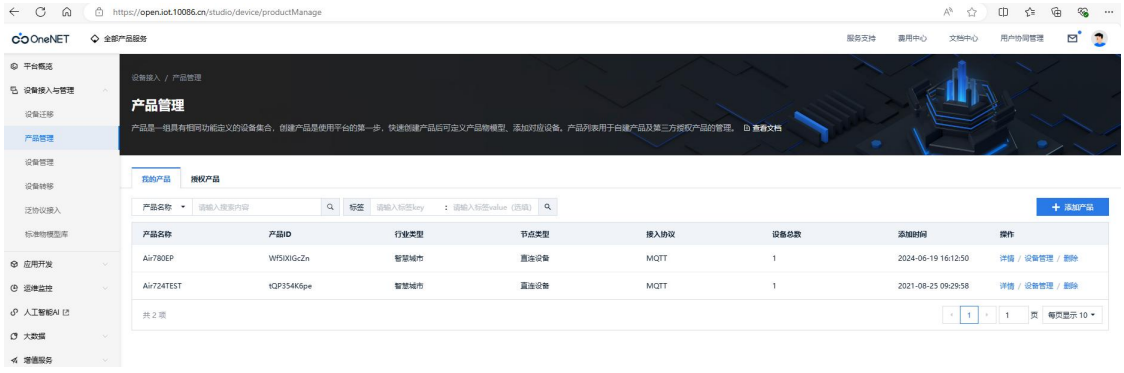
三．演示步骤

本文以 demo/onenet/studio 这个 DEMO 为例做演示

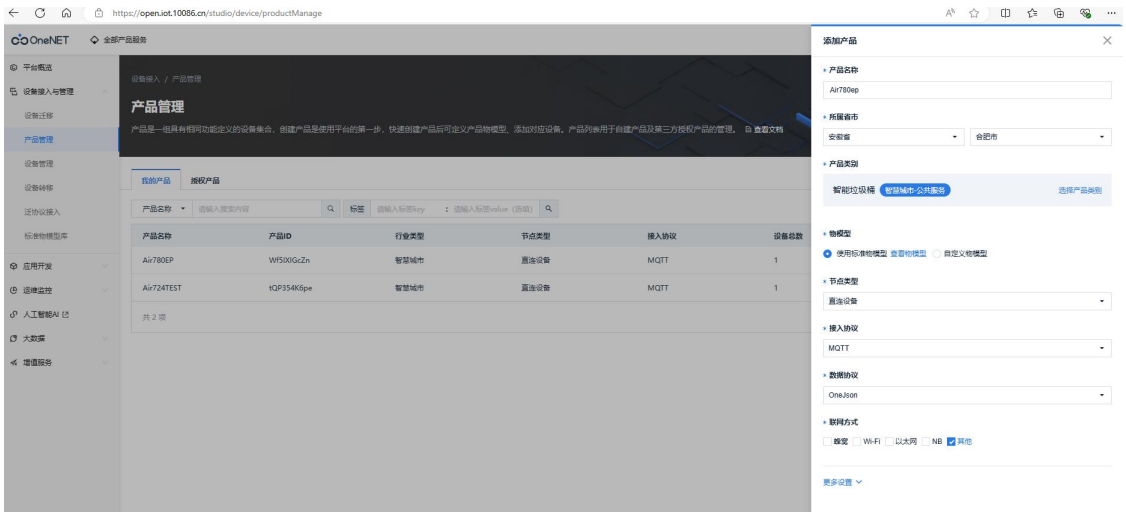
OneNET Studio 操作

产品操作

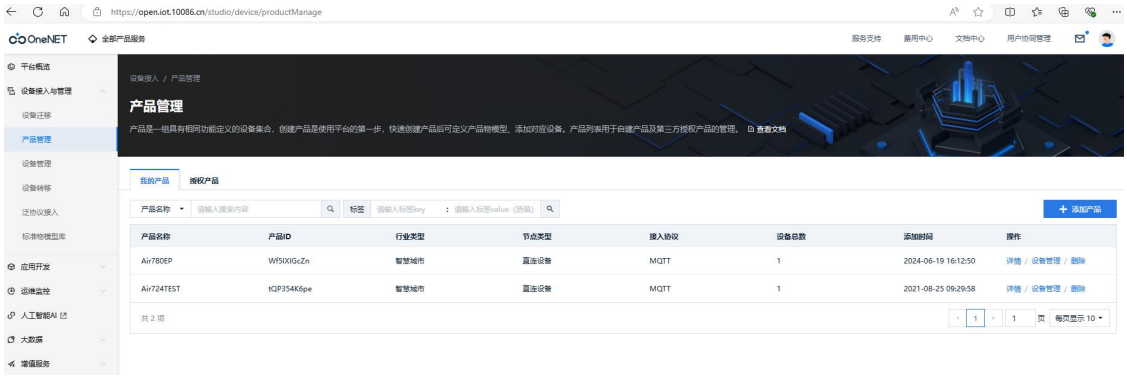
打开 OneNET 官网，进入控制台，进入 Onenet Studio，选择设备接入与管理，点击产品管理。



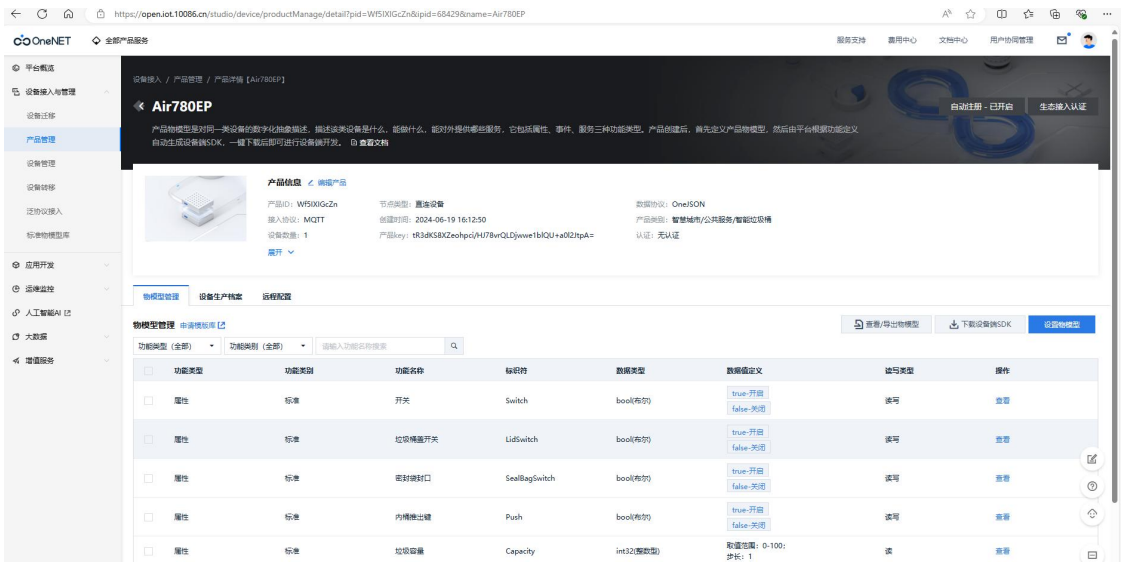
点击添加产品



根据自己产品填写，注意：节点类型选择直连设备，接入协议选择 MQTT，点击确定



点击详情



将右上角自动注册打开

记住产品 ID 和产品 key

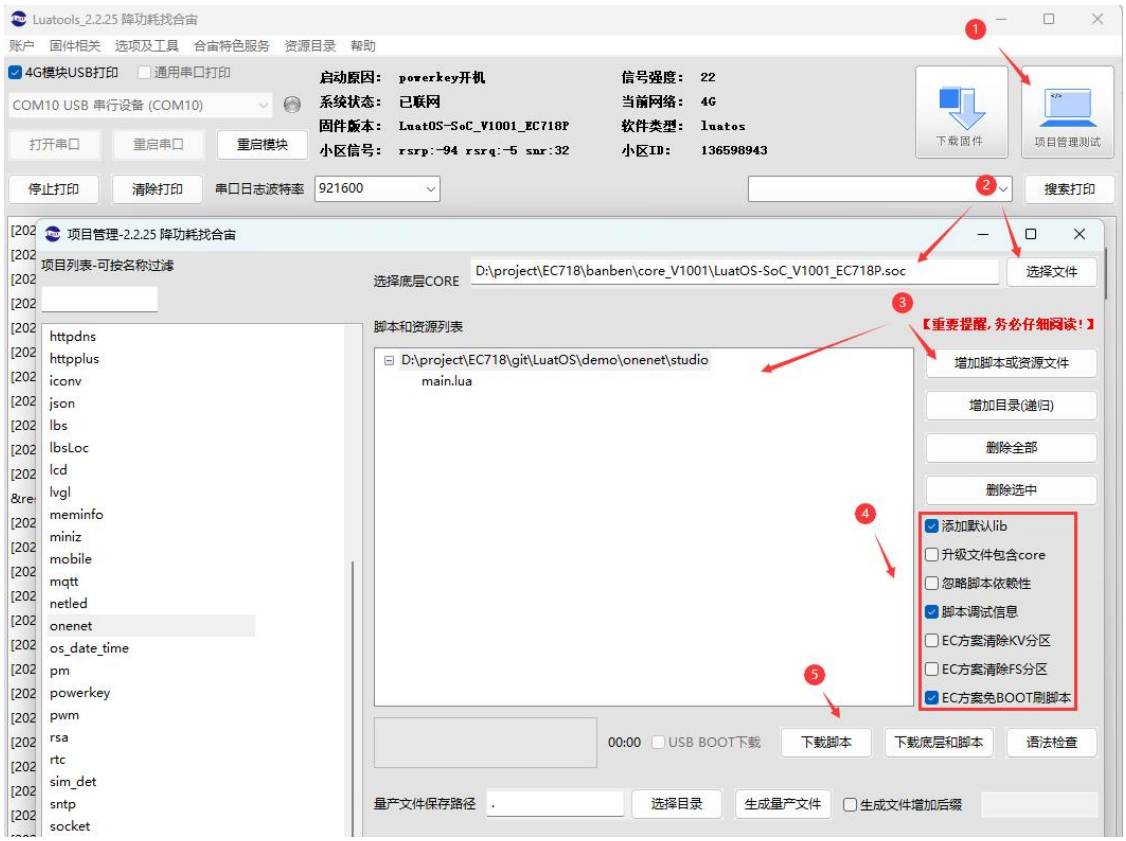
根据自己的设备修改以下参数

```
-- 根据自己的设备修改以下参数
-----------------------------------------------
-- OneNet Studio
mqtt_host = "studio-mqtt.heclouds.com"
mqtt_port = 1883
mqtt_isssl = false
local pid = "Wf5IXIGcZn" -- 产品id
local device = mobile.imei() -- 设备名称，按需设置，如果是Cat.1系列通常用mobile.imei()
local device_secret = "tR3dKS8XZeohpci/HJ78vrQLDjwwe1blQU+a0l2JtpA=" -- 设备密钥
client_id, user_name, password = iotauth.onenet(pid, device, device_secret)
```

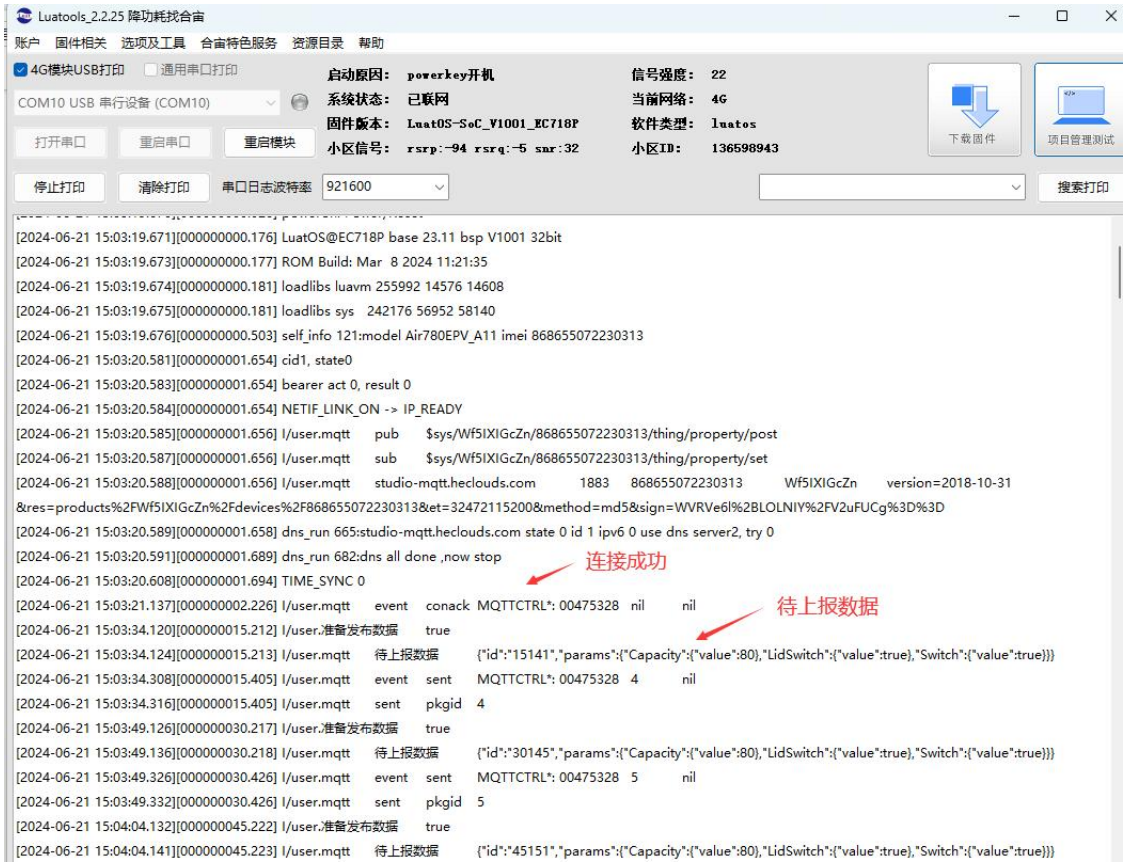
定时上报的数据改成如下内容

```
-- 这里演示在另一个task里上报数据，会定时上报数据，不需要就注释掉
sys.taskInit(function()
    local qos = 1 -- QOS0不带puback，QOS1是带puback的
    while true do
        sys.wait(15000)
        log.info("准备发布数据", mqttc and mqttc:ready())
        -- onenet 使用的是OneJson，就是规范化的Json结构
        -- https://open.iot.10086.cn/doc/v5/develop/detail/508
        if mqttc and mqttc:ready() then
            local data = {}
            data["id"] = tostring(mcu.ticks())
            data["params"] = {}
            -- 业务自定义数据
            -- 例如:
            -- 上传一个Power属性，类型是整型，值为1
            data["params"]["Switch"] = {
                value = true
            }
            -- 上传一个WF属性，类型浮点，值为2.35
            data["params"]["LidSwitch"] = {
                value = true
            }
            -- 上传一个字符串
            data["params"]["Capacity"] = {
                value = 80
            }
```

下载对应 core 和修改后的 script 到模块，这里 core 不变，可以直接下载对应的脚本

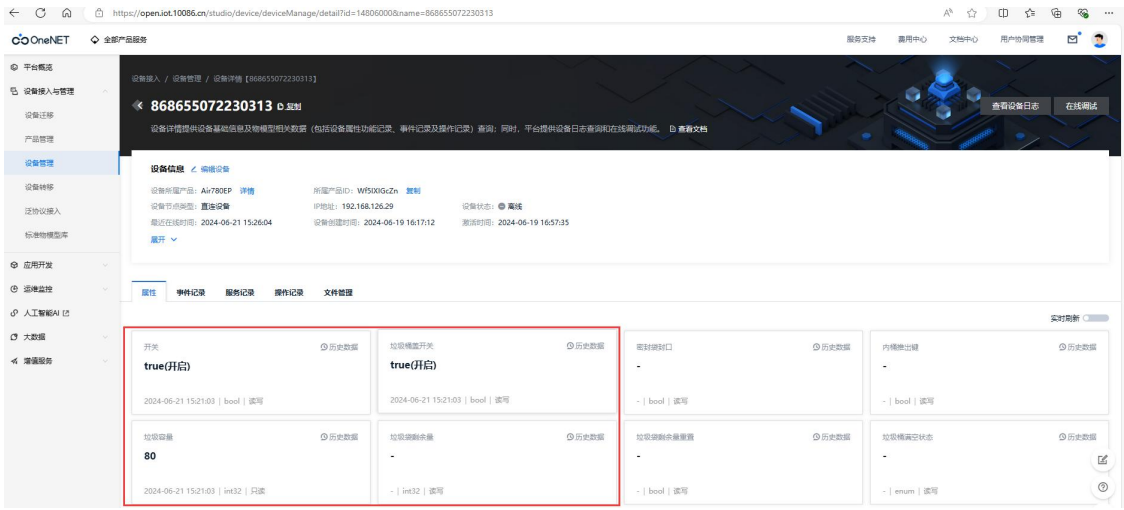


模块对应 log



One NET 平台显示效果

点击详情可以看到模块发送的数据



四．代码解析

点击右侧查看 iotauth API 说明：IoT 鉴权库，用于生成各种云平台的参数

点击右侧查看 mqtt API 说明：mqtt 客户端

完整脚本代码点击右侧查看 demo/onenet/studio/main.lua

根据自己的设备修改以下参数

iotauth.onenet：中国移动物联网平台三元组生成

```lua
-- 根据自己的设备修改以下参数
-----------------------------------------------
-- OneNet Studio
mqtt_host = "studio-mqtt.heclouds.com"
mqtt_port = 1883
mqtt_isssl = false
local pid = "Wf5IXIGcZn" -- 产品id
local device = mobile.imei() -- 设备名称，按需设置，如果是Cat.1系列通常用mobile.imei()
local device_secret = "tR3dKS8XZeohpci/HJ78vrQLDjwwe1blQU+a012JtpA=" -- 设备密钥
client_id, user_name, password = iotauth.onenet(pid, device, device_secret)

-- 下面是常用的topic，完整topic可参考 https://open.iot.10086.cn/doc/v5/develop/detail/639
pub_topic = "$sys/" .. pid .. "/" .. device .. "/thing/property/post"
sub_topic = "$sys/" .. pid .. "/" .. device .. "/thing/property/set"
pub_custome = "$sys/" .. pid .. "/" .. device .. "/custome/up"
pub_custome_reply = "$sys/" .. pid .. "/" .. device .. "/custome/up_reply"
sub_custome = "$sys/" .. pid .. "/" .. device .. "/custome/down/+"
sub_custome_reply = "$sys/" .. pid .. "/" .. device .. "/custome/down_reply/"
-----------------------------------------------
```

接收数据处理函数

```lua
function on_downlink(topic, payload)
    -- 演示多种处理方式

    -- 方式一，将数据publish出去，在另外一个sys.subscribe内处理
    sys.publish("mqtt_payload", data, payload)

    -- 方式二，将数据加入队列，然后通知task处理
    -- table.insert(payloads, {topic, payload})

    -- 方式三，直接处理
    if payload.startsWith("{") and payload.endsWith("}") then
        -- 可能是json,尝试解析
        local jdata = json.decode(payload)
        if jdata then
            -- 属性设置
            if topic == sub_topic then
                local params = jdata.params
                if params then
                    local power_switch = jdata.params.power_switch
                    if power_switch then
                        if power_switch == 1 then
                            log.info("led", "on")
                            LED(1)
                        else
                            log.info("led", "off")
                            LED(1)
                        end
                    end
                    -- 继续处理其他参数
                end
            end
        end
    end
end
```

统一联网函数，可自行删减

Air780EP 对应的是 mobile 选项

sys.waitUntil("IP_READY")：等待模块驻网成功

sys.publish("net_ready", device_id)：发布 net_ready 消息

```lua
-- 统一联网函数，可自行删减
sys.taskInit(function()
    if mqtt == nil then
        while 1 do
            sys.wait(1000)
            log.info("bsp", "本固件未包含mqtt库，请查证")
        end
    end
    local device_id = mcu.unique_id():toHex()
    ------------------------------
    -- 统一联网函数，可自行删减
    ------------------------------
    if wlan and wlan.connect then
        -- wifi 联网，ESP32系列均支持
        local ssid = "luatos1234"
        local password = "12341234"
        log.info("wifi", ssid, password)
        mcu.setClk(240)
        -- TODO 改成自动配网
        -- LED = gpio.setup(12, 0, gpio.PULLUP)
        wlan.init()
        wlan.setMode(wlan.STATION) -- 默认也是这个模式,不调用也可以
        device_id = wlan.getMac()
        wlan.connect(ssid, password, 1)
    elseif mobile then
        -- Air780E/Air600E系列
        -- mobile.simid(2) -- 自动切换SIM卡
        LED = gpio.setup(27, 0, gpio.PULLUP)
        device_id = mobile.imei()
    elseif w5500 then
        -- w5500 以太网，当前仅Air105支持
        w5500.init(spi.HSPI_0, 24000000, pin.PC14, pin.PC01, pin.PC00)
        w5500.config() -- 默认是DHCP模式
        w5500.bind(socket.ETH0)
        LED = gpio.setup(62, 0, gpio.PULLUP)
    elseif mqtt then
        -- 适配的mqtt库也OK
        -- 没有其他操作，单纯给个注释说明
    else
        -- 其他不认识的bsp，循环提示一下吧
        while 1 do
            sys.wait(1000)
            log.info("bsp", "本bsp可能未适配网络层，请查证")
        end
    end
    -- 默认都等到联网成功
    sys.waitUntil("IP_READY")
    sys.publish("net_ready", device_id)
end)
```

sys.waitUntil("net_ready")：等待联网

mqtt.create：创建 mqtt 客户端

mqttc:auth：mqtt 三元组配置

mqttc:autoreconn：自动重连机制

```lua
sys.taskInit(function()
    -- 等待联网
    sys.waitUntil("net_ready")

    -- 打印一下上报(pub)和下发(sub)的topic名称
    -- 上报: 设备 ---> 服务器
    -- 下发: 设备 <--- 服务器
    -- 可使用mqtt.x等客户端进行调试
    log.info("mqtt", "pub", pub_topic)
    log.info("mqtt", "sub", sub_topic)
    log.info("mqtt", mqtt_host, mqtt_port, client_id, user_name, password)

    ---------------------------------------
    -------- MQTT 演示代码 --------------
    ---------------------------------------

    mqttc = mqtt.create(nil, mqtt_host, mqtt_port, mqtt_isssl, ca_file)

    mqttc:auth(client_id, user_name, password) -- client_id必填,其余选填
    -- mqttc:keepalive(240) -- 默认值240s
    mqttc:autoreconn(true, 3000) -- 自动重连机制
```

mqttc:on：注册 mqtt 回调

sys.publish("mqtt_conack")：发布 mqtt_conack 消息

mqtt_client:subscribe(topics) ：多主题订阅

```
mqttc:on(function(mqtt_client, event, data, payload)
    -- 用户自定义代码
    log.info("mqtt", "event", event, mqtt_client, data, payload)
    if event == "conack" then
        -- 联上了
        sys.publish("mqtt_conack")
        local topics = {}
        -- 物模型的topic
        topics[sub_topic] = 2

        -- 透传模式的topic
        -- 首先是 上报后，服务器会回复
        if pub_custome_reply then
            topics[pub_custome_reply] = 1
        end
        -- 然后是 服务器的下发
        if sub_custome then
            topics[sub_custome] = 1
        end
        -- mqtt_client:subscribe(sub_topic, 2)--单主题订阅
        mqtt_client:subscribe(topics) -- 多主题订阅
    elseif event == "recv" then
        -- 打印收到的内容，时间生产环境建议注释掉，不然挺多的
        log.info("mqtt", "downlink", "topic", data, "payload", payload)
        on_downlink(data, payload)
    elseif event == "sent" then
        log.info("mqtt", "sent", "pkgid", data)
        -- elseif event == "disconnect" then
        -- 非自动重连时,按需重启mqttc
        -- mqtt_client:connect()
    end
end)
```

mqttc:connect()：连接服务器

sys.waitUntil("mqtt_conack")：等待 mqtt 连接成功

mqttc:publish：发布消息

```
-- mqttc自动处理重连，除非自行关闭
mqttc:connect()
sys.waitUntil("mqtt_conack")
while true do
    -- 演示等待其他task发送过来的上报信息
    local ret, topic, data, qos = sys.waitUntil("mqtt_pub", 300000)
    if ret then
        -- 提供关闭本while循环的途径，不需要可以注释掉
        if topic == "close" then
            break
        end
        mqttc:publish(topic, data, qos)
    end
    -- 如果没有其他task上报，可以写个空等待
    -- sys.wait(60000000)
end
mqttc:close()
mqttc = nil
```

这里演示在另一个 task 里上报数据，会定时上报数据

mqttc:ready()：mqtt 客户端是否就绪

json.encode(data)：将对象序列化为 json 字符串

mqttc:publish：发布消息

```
-- 这里演示在另一个task里上报数据，会定时上报数据,不需要就注释掉
sys.taskInit(function()
    local qos = 1 -- QOS0不带puback，QOS1是带puback的
    while true do
        sys.wait(15000)
        log.info("准备发布数据", mqttc and mqttc:ready())
        -- onenet 使用的是OneJson，就是规范化的Json结构
        -- https://open.iot.10086.cn/doc/v5/develop/detail/508
        if mqttc and mqttc:ready() then
            local data = {}
            data["id"] = tostring(mcu.ticks())
            data["params"] = {}
            -- 业务自定义数据
            -- 例如:
            -- 上传一个Power属性，类型是整型，值为1
            data["params"]["Switch"] = {
                value = true
            }
            -- 上传一个WF属性，类型浮点，值为2.35
            data["params"]["LidSwitch"] = {
                value = true
            }
            -- 上传一个字符串
            data["params"]["Capacity"] = {
                value = 80
            }
            local updata = json.encode(data)
            log.info("mqtt", "待上报数据", updata)
            local pkgid = mqttc:publish(pub_topic, updata, qos)
        end
    end
end)
```

10.5.9 mobile 开发应用指南

一. 概述

mobile 接口获取模块的各种信息，如：band，imei，imsi，sn，muid，iccid，csq，rssi，rsrq，rsrp，snr，simid，apn，ip。

LuatOS-Air 开发模式下，固件分为两部分：core 和 script

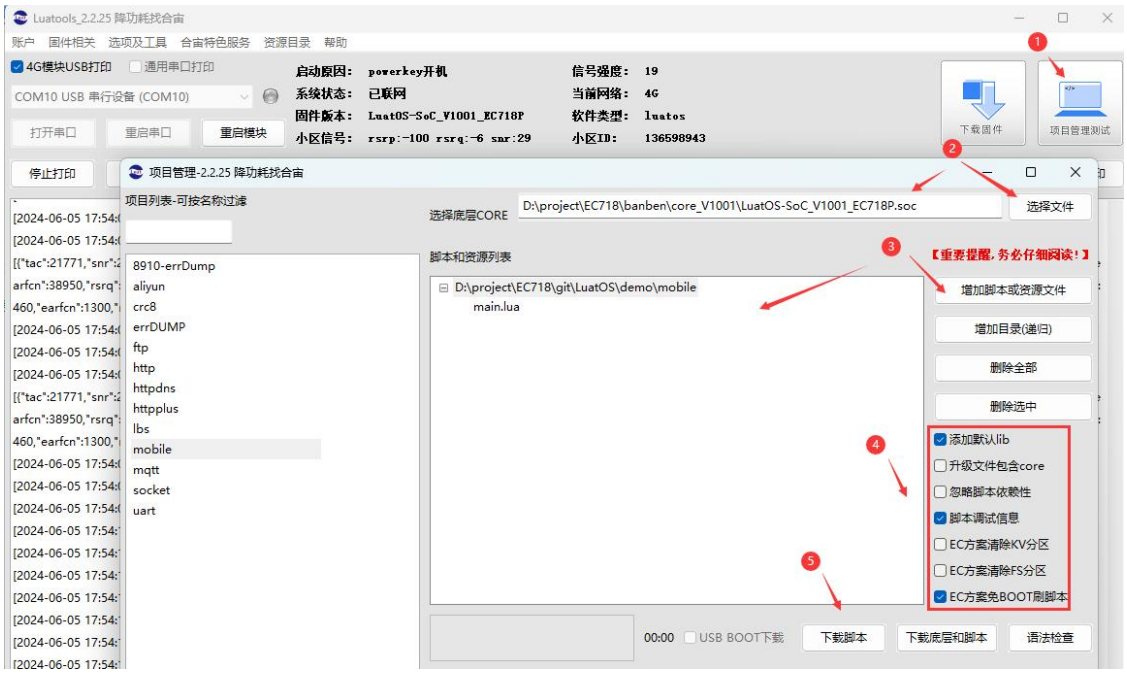
二. 材料准备

4. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
5. PC 电脑
6. 固件和 DEMO 下载地址 LuatOS 软件包

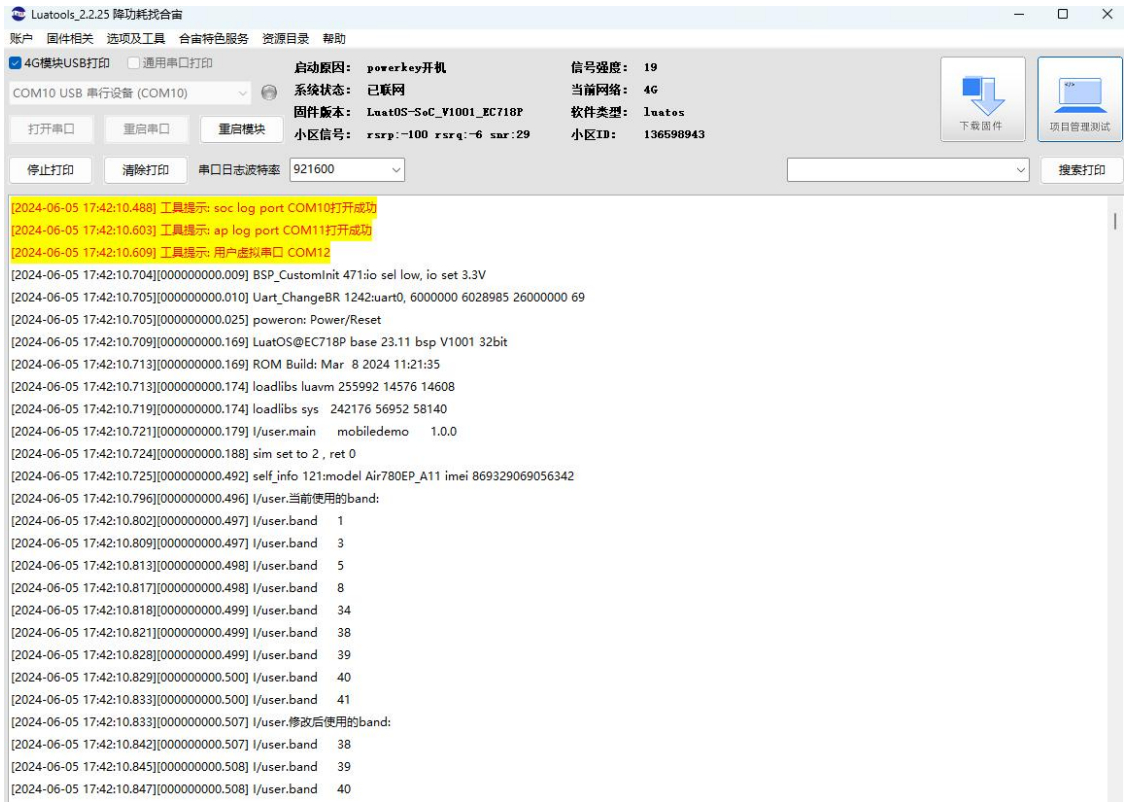
三. 演示步骤

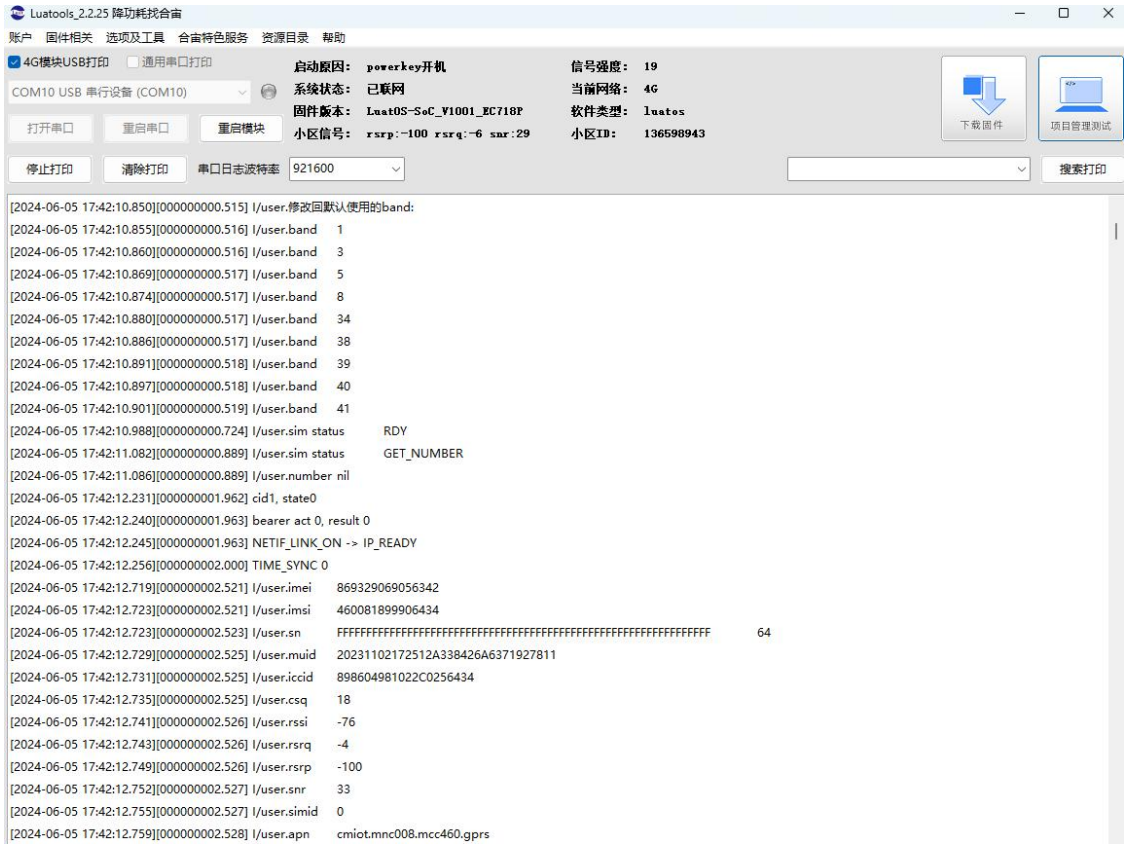
本文以 demo/mobile 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本


Luatools_2.2.25 降功耗找合宙
账户 固件相关 选项及工具 合宙特色服务 资源目录 帮助
4G模块USB打印
通用串口打印
COM10 USB 串行设备 (COM10)
打开串口
重启串口
重启模块
停止打印
启动原因: powerkey开机
系统状态: 已联网
固件版本: LuatOS-SoC_V1001_EC718P
小区信号: rsrp:-100 rsrq:-6 snr:29
信号强度: 19
当前网络: 4G
软件类型: luatos
小区ID: 136598943
下载固件
项目管理测试
项目管理-2.2.25 降功耗找合宙
项目列表-可按名称过滤
8910-errDump
aliyun
crc8
errDUMP
ftp
http
httpdns
httpplus
lbs
mobile
mqtt
socket
uart
选择底层CORE
D:\project\EC718\banben\core_V1001\LuatOS-SoC_V1001_EC718P.soc
选择文件
脚本和资源列表
D:\project\EC718\git\LuatOS\demo\mobile
main.lua
【重要提醒，务必仔细阅读！】
增加脚本或资源文件
增加目录(递归)
删除全部
删除选中
添加默认lib
升级文件包含core
忽略脚本依赖性
脚本调试信息
EC方案清除KV分区
EC方案清除FS分区
EC方案免BOOT刷脚本
00:00
USB BOOT下载
下载脚本
下载底层和脚本
语法检查

模块打印 log


Luatools_2.2.25 降功耗找合宙
账户 固件相关 选项及工具 合宙特色服务 资源目录 帮助
4G模块USB打印
通用串口打印
COM10 USB 串行设备 (COM10)
打开串口
重启串口
重启模块
停止打印
清除打印
串口日志波特率
921600
搜索打印
[2024-06-05 17:42:10.488] 工具提示: soc log port COM10打开成功
[2024-06-05 17:42:10.603] 工具提示: ap log port COM11打开成功
[2024-06-05 17:42:10.609] 工具提示: 用户虚拟串口 COM12
[2024-06-05 17:42:10.704][000000000.009] BSP_CustomInit 471:io sel low, io set 3.3V
[2024-06-05 17:42:10.705][000000000.010] Uart_ChangeBR 1242:uart0, 6000000 6028985 26000000 69
[2024-06-05 17:42:10.705][000000000.025] poweron: Power/Reset
[2024-06-05 17:42:10.709][000000000.169] LuatOS@EC718P base 23.11 bsp V1001 32bit
[2024-06-05 17:42:10.713][000000000.169] ROM Build: Mar 8 2024 11:21:35
[2024-06-05 17:42:10.713][000000000.174] loadlibs luavm 255992 14576 14608
[2024-06-05 17:42:10.719][000000000.174] loadlibs sys 242176 56952 58140
[2024-06-05 17:42:10.721][000000000.179] I/user.main mobiledemo 1.0.0
[2024-06-05 17:42:10.724][000000000.188] sim set to 2 , ret 0
[2024-06-05 17:42:10.725][000000000.492] self_info 121:model Air780EP_A11 imei 869329069056342
[2024-06-05 17:42:10.796][000000000.496] I/user.当前使用的band:
[2024-06-05 17:42:10.802][000000000.497] I/user.band 1
[2024-06-05 17:42:10.809][000000000.497] I/user.band 3
[2024-06-05 17:42:10.813][000000000.498] I/user.band 5
[2024-06-05 17:42:10.817][000000000.498] I/user.band 8
[2024-06-05 17:42:10.818][000000000.499] I/user.band 34
[2024-06-05 17:42:10.821][000000000.499] I/user.band 38
[2024-06-05 17:42:10.828][000000000.499] I/user.band 39
[2024-06-05 17:42:10.829][000000000.500] I/user.band 40
[2024-06-05 17:42:10.833][000000000.500] I/user.band 41
[2024-06-05 17:42:10.833][000000000.507] I/user.修改后使用的band:
[2024-06-05 17:42:10.842][000000000.507] I/user.band 38
[2024-06-05 17:42:10.845][000000000.508] I/user.band 39
[2024-06-05 17:42:10.847][000000000.508] I/user.band 40



四．代码解析

完整脚本代码点击右侧查看 demo/mobile/main.lua

获取和设置 BAND 脚本

mobile.getBand：获取当前使用/支持的 band

mobile.setBand：设置使用的 band

```lua
local band = zbuff.create(40)
local band1 = zbuff.create(40)
mobile.getBand(band)
log.info("当前使用的band:")
for i=0,band:used()-1 do
    log.info("band", band[i])
end
band1[0] = 38
band1[1] = 39
band1[2] = 40
mobile.setBand(band1, 3)     --改成使用38,39,40
band1:clear()
mobile.getBand(band1)
log.info("修改后使用的band:")
for i=0,band1:used()-1 do
    log.info("band", band1[i])
end
mobile.setBand(band, band:used())     --改回原先使用的band，也可以下载的时候选择清除fs

mobile.getBand(band1)
log.info("修改回默认使用的band:")
for i=0,band1:used()-1 do
    log.info("band", band1[i])
end
```

获取模块相关信息

mobile.imei()：获取 IMEI

mobile.imsi()：获取 IMSI

mobile.sn()：获取 SN

mobile.muid()：获取 MUID

mobile.iccid()：获取或设置 ICCID

mobile.csq()：获取 csq，若失败返回 0. 范围 0 - 31, 越大越好

mobile.rssi()：获取 rssi 值,若失败返回 0. 范围 0 到 -114, 越大越好

mobile.rsrq()：获取 rsrp 值（参考信号接收功率）,若失败返回 0. 取值范围: -44 ~ -140 ， 值越大越好

mobile.rsrp()：获取 rsrq 值（参考信号发送功率）,若失败返回 0. 取值范围: -3 ~ -19.5 ， 值越大越好

mobile.snr()：获取 snq 值（信噪比）,若失败返回 0.范围 0 - 30, 越大越好

mobile.simid()：获取当前 SIM 卡槽

mobile.apn(0,1)：获取 apn

```lua
while 1 do
    log.info("imei", mobile.imei())
    log.info("imsi", mobile.imsi())
    local sn = mobile.sn()
    if sn then
        log.info("sn",   sn:toHex())
    end
    log.info("muid", mobile.muid())
    log.info("iccid", mobile.iccid())
    log.info("csq", mobile.csq()) -- 4G模块的CSQ并不能完全代表强度
    log.info("rssi", mobile.rssi()) -- 需要综合rssi/rsrq/rsrp/snr一起判断
    log.info("rsrq", mobile.rsrq())
    log.info("rsrp", mobile.rsrp())
    log.info("snr", mobile.snr())
    log.info("simid", mobile.simid()) -- 这里是获取当前SIM卡槽
    log.info("apn", mobile.apn(0,1))
    log.info("ip", socket.localIP())
    sys.wait(15000)
end
```

基站信息查询脚本

mobile.getCellInfo()：获取基站信息，从 2023.06.20 开始，需要主动请求一次 reqCellInfo 才会有基站数据

mobile.reqCellInfo()：发起基站信息查询,含临近小区

```
-- 轮询式，包含临近小区信息，这是手动搜索，和上面的自动搜索冲突，开启一个就行
sys.taskInit(function()
    sys.wait(5000)
    mobile.config(mobile.CONF_SIM_WC_MODE, 2)
    while 1 do
        mobile.reqCellInfo(10)
        sys.wait(11000)
        log.info("cell", json.encode(mobile.getCellInfo()))
        mobile.config(mobile.CONF_SIM_WC_MODE, 2)
    end
end)
```

mobile.number(0)：获取手机卡号，注意，只有写入了手机号才能读出，因此有可能读出来是空的

```
-- 获取sim卡的状态

sys.subscribe("SIM_IND", function(status, value)
    log.info("sim status", status)
    if status == 'GET_NUMBER' then
        log.info("number", mobile.number(0))
    end
    if status == "SIM_WC" then
        log.info("sim", "write counter", value)
    end
end)
```

六. 常见问题

获取不到相关信息，确认模块是否正常开机，是否识别到 SIM 卡，SIM 卡是否欠费，流量是否用完，天线是否匹配，卡是否为专网卡，是否机卡绑定。

10.5.10 meminfo 开发应用指南

一. 概述

meminfo DEMO 功能获取内存信息

LuatOS-Air 开发模式下，固件分为两部分：core 和 script

二. 材料准备

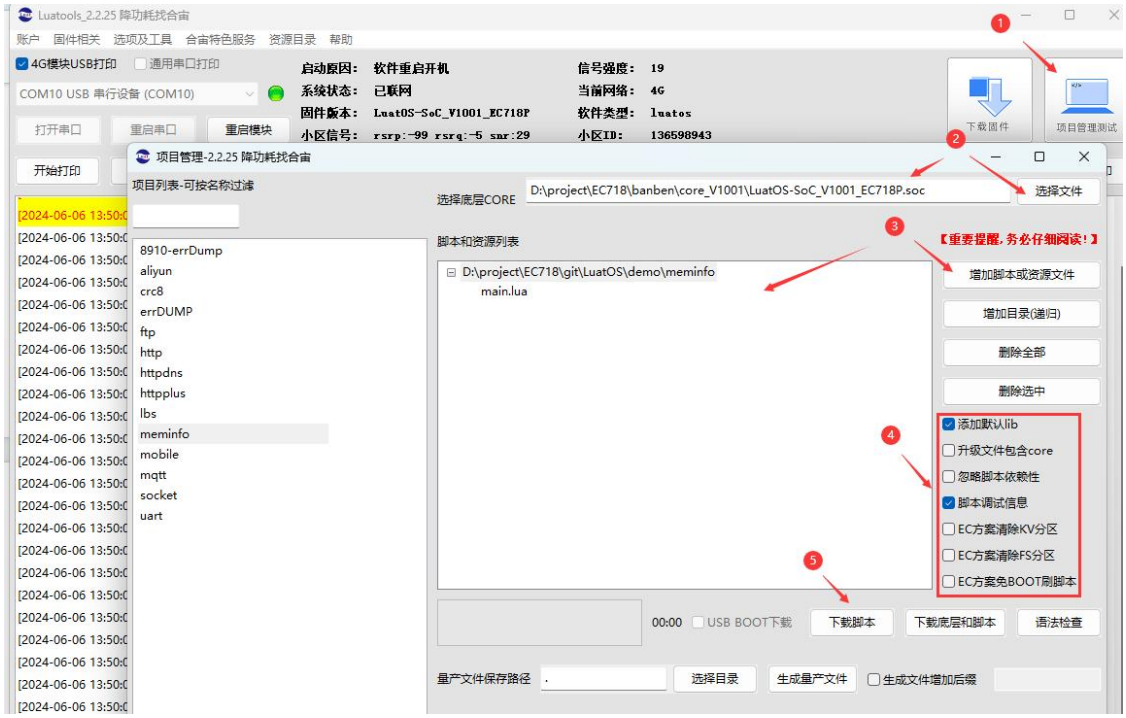
1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑

3. 固件和 DEMO 下载地址 LuatOS 软件包

三．演示步骤

本文以 demo/meminfo 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本

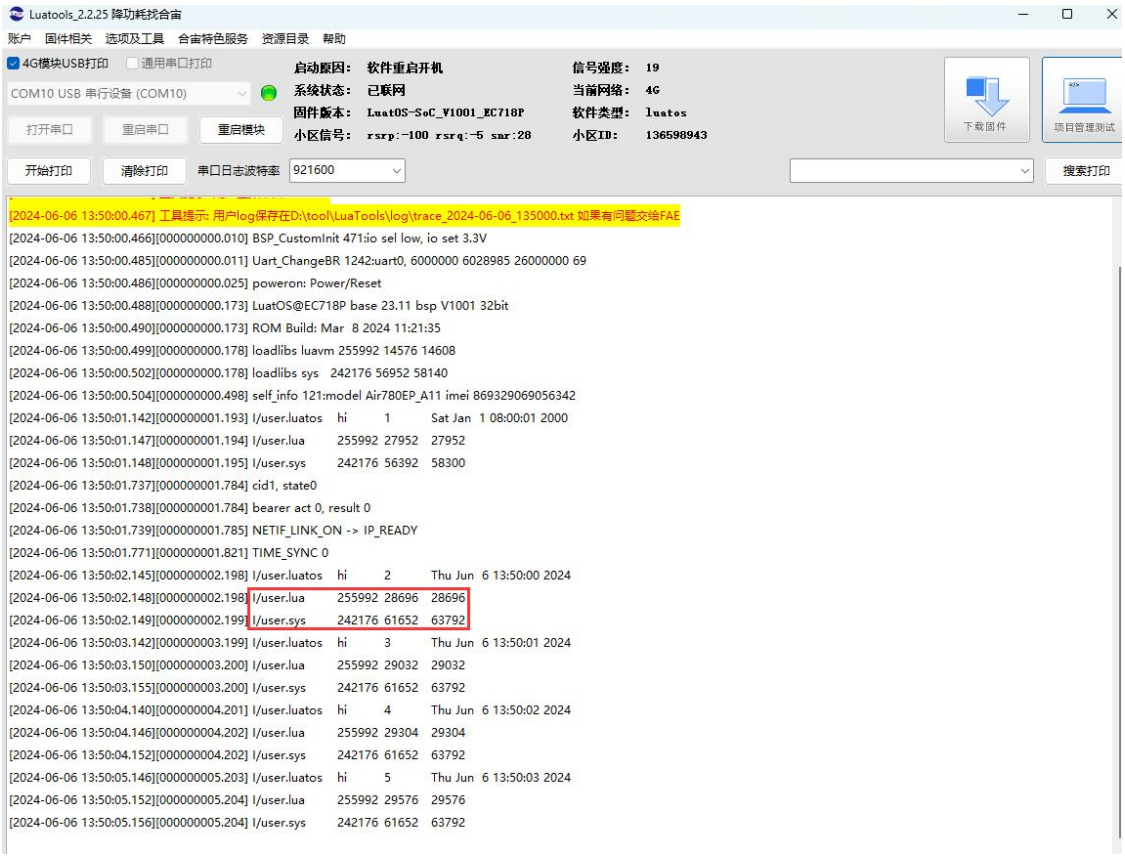


模块打印 log，下面红框内的数据为 lua 内存和 sys 内存的打印数据。

第 1 个值：总内存大小,单位字节

第 2 个值：当前已使用的内存大小,单位字节

第 3 个值：历史最高已使用的内存大小,单位字节



四．代码解析

完整脚本代码点击右侧查看 demo/meminfo/main.lua

创建了 1 个 task 每 1 秒打印 1 次对应的内存

rtos.meminfo()：打印 lua 虚拟机内存

rtos.meminfo("sys")：打印 sys 虚拟机内存

```lua
sys.taskInit(function()
    local count = 1
    while 1 do
        sys.wait(1000)
        log.info("luatos", "hi", count, os.date())
        -- lua内存
        log.info("lua", rtos.meminfo())
        -- sys内存
        log.info("sys", rtos.meminfo("sys"))
        count = count + 1
    end
end)
```

10.5.11 lbsLoc 开发应用指南

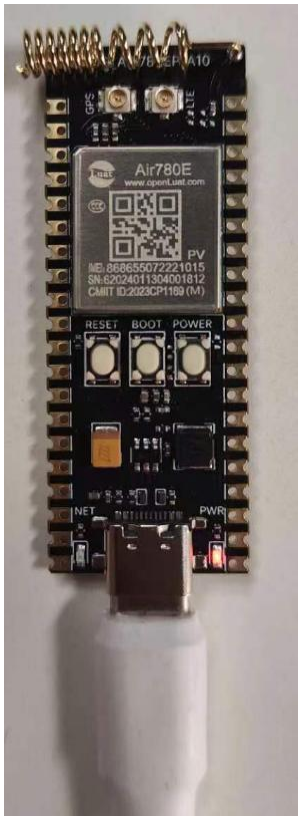
一. 概述

lbsLoc DEMO 功能获取基站对应的经纬度

LuatOS-Air 开发模式下，固件分为两部分：core 和 script

二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

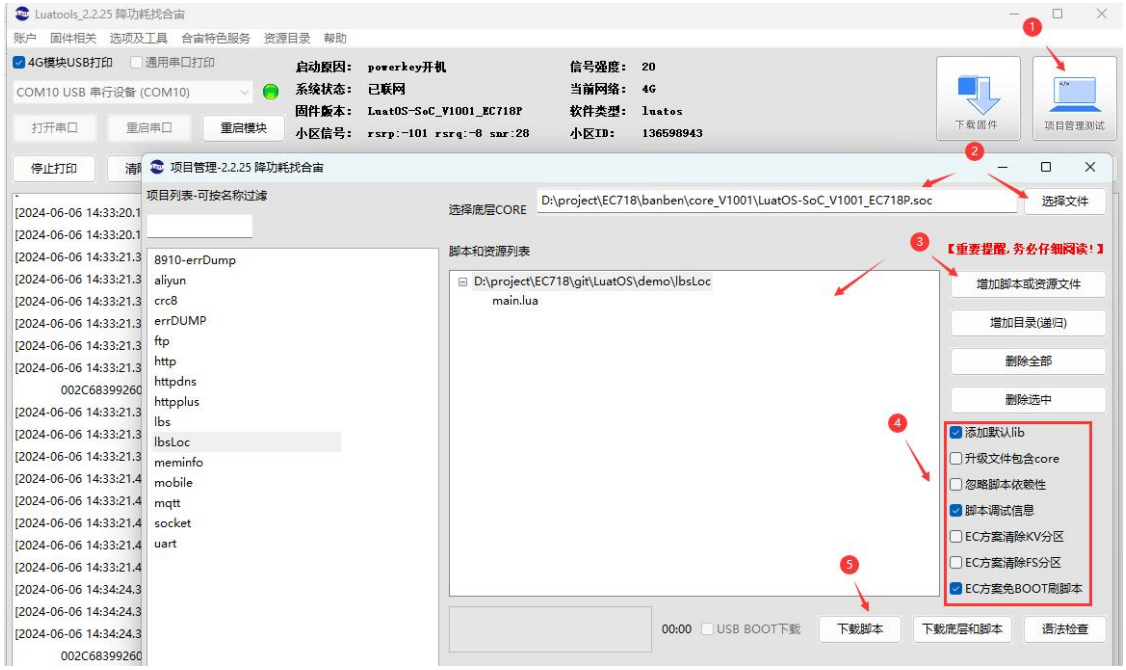


三. 演示步骤

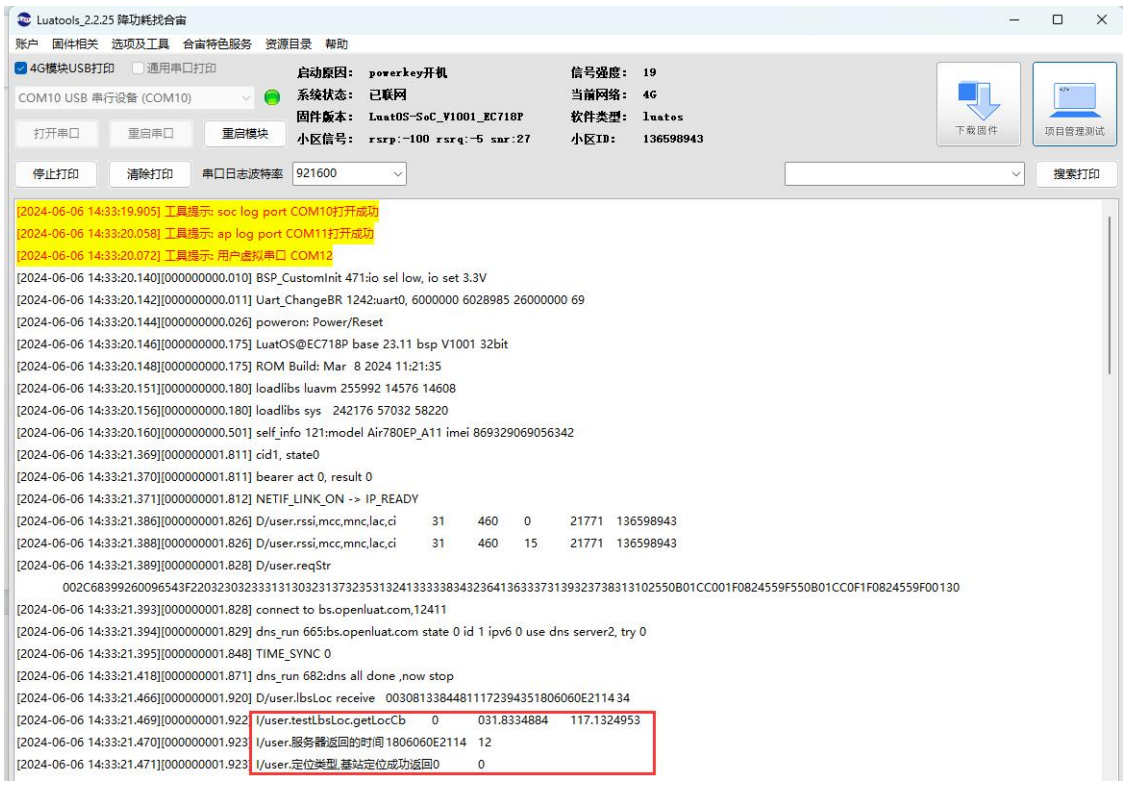
本文以 demo/lbsLoc 这个 DEMO 为例做演示

DEMO 默认演示，只是基站定位。免费版的基站+wifi 混合定位,大部分情况下只会返回基站定位的结果，收费版本请咨询销售。基站+wifi 混合定位方式请根据 DEMO 描述自行测试。

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块对应 log



四．代码解析

完整脚本代码点击右侧查看 demo/lbsLoc/main.lua

mobile.reqCellInfo(15)：发起基站信息查询,含临近小区

lbsLoc.request(getLocCb)：发送基站定位请求

getLocCb(result, lat, lng, addr, time, locType)：获取基站对应的经纬度后的回调函数

--result：number 类型，0 表示成功，1 表示网络环境尚未就绪，2 表示连接服务器失败，3 表示发送数据失败，4 表示接收服务器应答超时，5 表示服务器返回查询失败；为 0 时，后

面的 5 个参数才有意义

-- lat：string 类型，纬度，整数部分 3 位，小数部分 7 位，例如 031.2425864

-- lng：string 类型，经度，整数部分 3 位，小数部分 7 位，例如 121.4736522

-- addr：目前无意义

-- time：string 类型或者 nil，服务器返回的时间，6 个字节，年月日时分秒，需要转为十六进制读取

-- 第一个字节：年减去 2000，例如 2017 年，则为 0x11

-- 第二个字节：月，例如 7 月则为 0x07，12 月则为 0x0C

-- 第三个字节：日，例如 11 日则为 0x0B

-- 第四个字节：时，例如 18 时则为 0x12

-- 第五个字节：分，例如 59 分则为 0x3B

-- 第六个字节：秒，例如 48 秒则为 0x30

-- locType：numble 类型或者 nil，定位类型，0 表示基站定位成功，255 表示 WIFI 定位成功

```lua
-- 功能:获取基站对应的经纬度后的回调函数
-- 参数:-- result：number类型，0表示成功，1表示网络环境尚未就绪，2表示连接服务器失败，3表示发送数据失败，4表示接收服务器应答超时，5表示服务器返回查询失败；为0时，后面的5个参数才有意义
        -- lat：string类型，纬度，整数部分3位，小数部分7位，例如031.2425864
        -- lng：string类型，经度，整数部分3位，小数部分7位，例如121.4736522
        -- addr：目前无意义
        -- time：string类型或者nil，服务器返回的时间，6个字节，年月日时分秒，需要转为十六进制读取
            -- 第一个字节：年减去2000，例如2017年，则为0x11
            -- 第二个字节：月，例如7月则为0x07，12月则为0x0C
            -- 第三个字节：日，例如11日则为0x0B
            -- 第四个字节：时，例如18时则为0x12
            -- 第五个字节：分，例如59分则为0x3B
            -- 第六个字节：秒，例如48秒则为0x30
        -- locType：numble类型或者nil，定位类型，0表示基站定位成功，255表示WIFI定位成功
local function getLocCb(result, lat, lng, addr, time, locType)
    log.info("testLbsLoc.getLocCb", result, lat, lng)
    -- 获取经纬度成功
    if result == 0 then
        log.info("服务器返回的时间", time:toHex())
        log.info("定位类型,基站定位成功返回0", locType)
    end
    -- 广播给其他需要定位数据的task
    -- sys.publish("lbsloc_result", result, lat, lng)
end

sys.taskInit(function()
    sys.waitUntil("IP_READY", 30000)
    while 1 do
        mobile.reqCellInfo(15)
        sys.waitUntil("CELL_INFO_UPDATE", 3000)
        lbsLoc.request(getLocCb)
        sys.wait(60000)
    end
end)
```

五．常见问题

1. 基站定位获取的经纬度是什么格式的？

基站定位获取的经纬度为 WGS-84 格式

各种坐标系说明以及转换方法参考：http://old.openluat.com/GPS-Offset.html

2. 基站定位精度

搜索到的小区越多，定位的精度越高；一般来说，城市中心定位精度比郊区和农村定位精度高，城市中心的定位精度在几十米到几百米不等，郊区和农村的定位精度更低，可能会有几千米甚至更多的误差

基站定位的误差都比较大，如果需要准确定位，请使用支持 GPS 的模块

3. wifi 定位和基站定位的区别

wifi 定位和基站定位都是通过调通 lbsLoc.request 接口来实现定位

主要区别是，wifi 定位要先通过 wlan.scanResult()接口先搜索 WIFI 热点信息(MAC 地址和信号强度)，然后把搜索到的 WIFI 热点信息传入 lbsLoc.request 接口的最后一个参数，后台会查询 WIFI 热点对应的经纬度。

4. wifi 定位精度

根据设备获取的 WiFi 的信息进行定位，WIFI 定位精度一般不受使用环境影响，主要和单一 WIFI 辐射半径、WIFI 覆盖密度有关。一般来说，WIFI 精度在 3 米－200 米左右。

10.5.12 os_date_time 开发应用指南

一. 概述

os_date_time DEMO 功能获取本地和 UTC 时间

LuatOS-Air 开发模式下，固件分为两部分：core 和 script

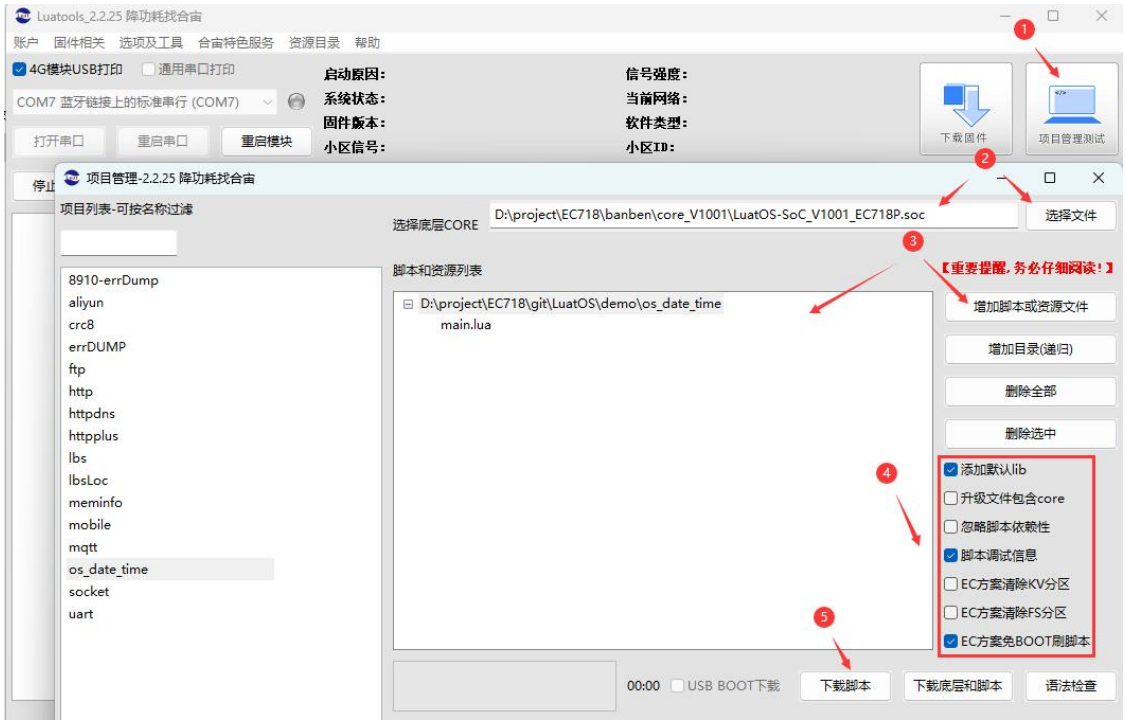
二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

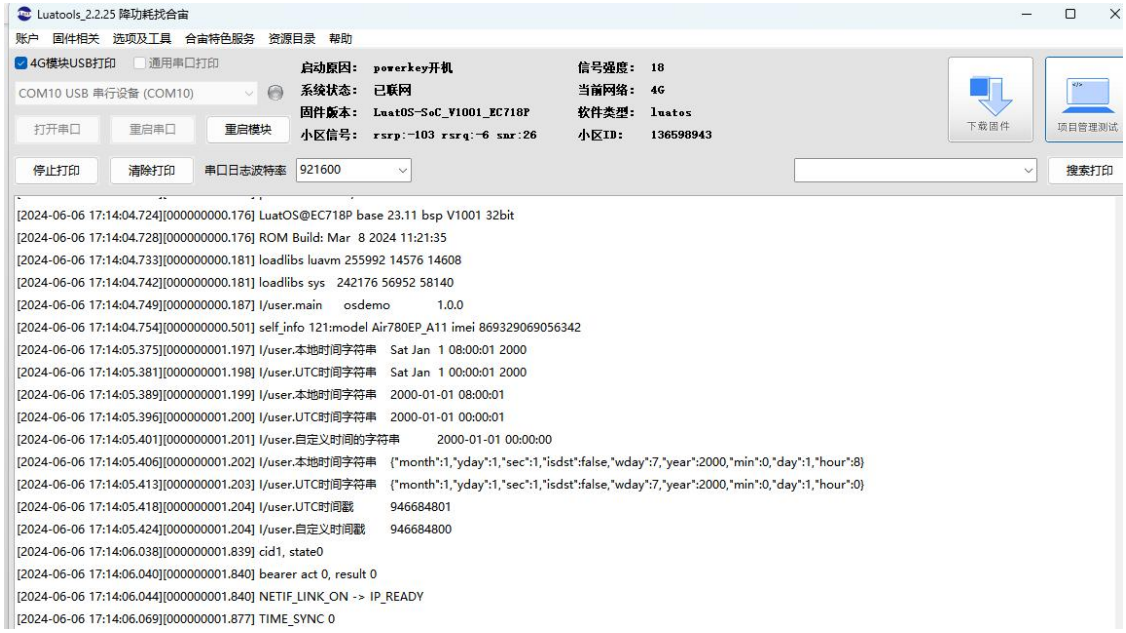
三．演示步骤

本文以 demo/os_date_time 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块对应 log，打印不同格式的本地时间和 UTC 时间



四．代码解析

完整脚本代码点击右侧查看 os_date_time/main.lua

os.date()：获取本地时间字符串

os.date("!%c")：获取 UTC 时间字符串

os.date("%Y-%m-%d %H:%M:%S")：格式化本地时间字符串

os.date("!%Y-%m-%d %H:%M:%S")：格式化 UTC 时间字符串

os.date("!%Y-%m-%d %H:%M:%S", os.time({year=2000, mon=1, day=1, hour=0, min=0, sec=0}))：格式化时间字符串

json.encode(os.date("*t")：获取本地时间的 table

json.encode(os.date("!*t")：获取 UTC 时间的 table

os.time()：UTC 时间戳，但 lua 下的精度只能到秒

os.time({year=2000, mon=1, day=1, hour=0, min=0, sec=0})：自定义时间戳

```lua
sys.taskInit(function()
    sys.wait(1000)

    -- 获取本地时间字符串
    log.info("本地时间字符串", os.date())
    -- 获取UTC时间字符串
    log.info("UTC时间字符串", os.date("!%c"))

    -- 格式化本地时间字符串
    log.info("本地时间字符串", os.date("%Y-%m-%d %H:%M:%S"))
    -- 格式化UTC时间字符串
    log.info("UTC时间字符串", os.date("!%Y-%m-%d %H:%M:%S"))

    -- 格式化时间字符串
    log.info("自定义时间的字符串", os.date("!%Y-%m-%d %H:%M:%S", os.time({year=2000, mon=1, day=1, hour=0, min=0, sec=0})))

    -- 获取本地时间的table
    log.info("本地时间字符串", json.encode(os.date("*t")))
    -- 获取UTC时间的table
    log.info("UTC时间字符串",  json.encode(os.date("!*t")))


    -- 时间戳，但lua下的精度只能到秒
    log.info("UTC时间戳", os.time())
    log.info("自定义时间戳", os.time({year=2000, mon=1, day=1, hour=0, min=0, sec=0}))
end)
```

10.5.13 rtc 开发应用指南

一. 概述

rtc DEMO 功能获取和设置本地时间

LuatOS-Air 开发模式下，固件分为两部分：core 和 script

二. 材料准备

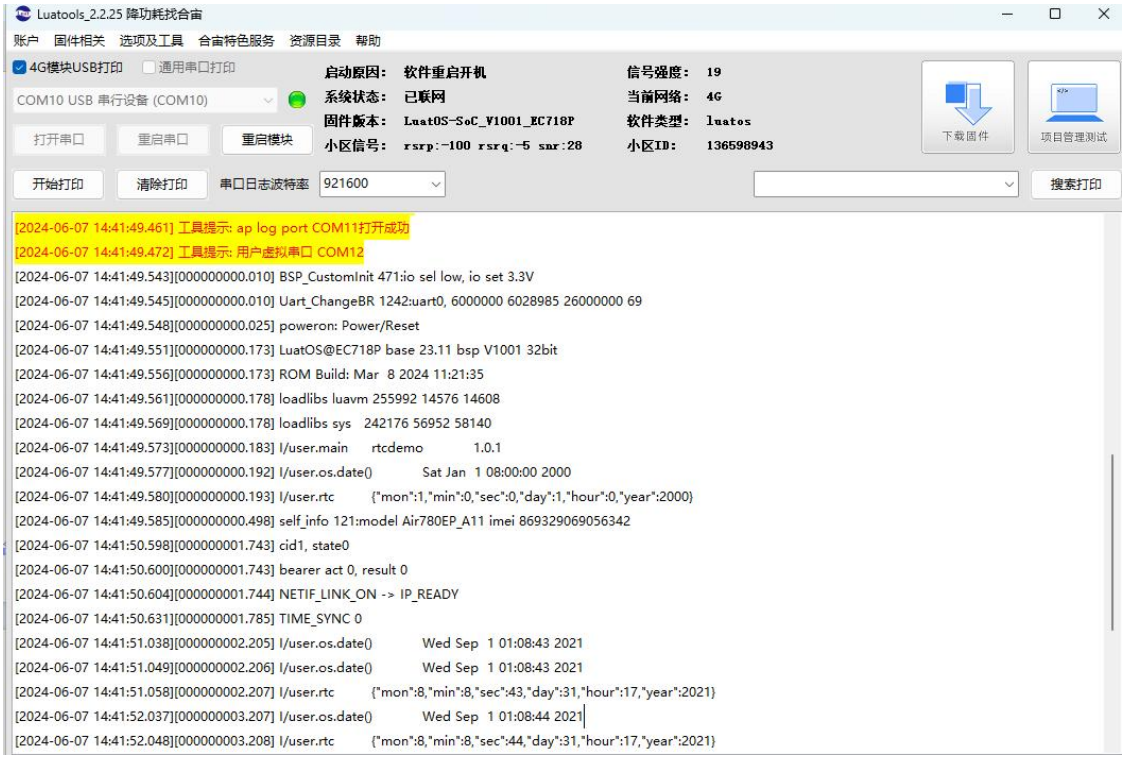
1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

三. 演示步骤

本文以 demo/rtc 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本

模块打印对应 log



四．代码解析

完整脚本代码点击右侧查看 demo/rtc/main.lua

rtc.get()：获取时钟

rtc.set({year=2021,mon=8,day=31,hour=17,min=8,sec=43})：设置时钟

```lua
sys.taskInit(function()
    log.info("os.date()", os.date())
    local t = rtc.get()
    log.info("rtc", json.encode(t))
    sys.wait(2000)
    rtc.set({year=2021,mon=8,day=31,hour=17,min=8,sec=43})
    log.info("os.date()", os.date())
    -- rtc.timerStart(0, {year=2021,mon=9,day=1,hour=17,min=8,sec=43})
    -- rtc.timerStop(0)
    while 1 do
        log.info("os.date()", os.date())
        local t = rtc.get()
        log.info("rtc", json.encode(t))
        sys.wait(1000)
    end
end)
```

10.5.14 sntp 开发应用指南

一. 概述

时间同步有很多种方法，基站授时，GPS 授时，ntp 授时等等多种方法。

NTP 是大家众所周知的、广泛使用的、精确可靠的时钟同步方式。具有高效、准确、流量少，易实现的特点。NTP 这种方式，就是使用“定义”中的蜂窝网络，和远端 NTP 服务器交互数据，收到服务器的回应并解析，重新设定系统时钟，完成同步。

LuatOS-Air 开发模式下，固件分为两部分：core 和 script

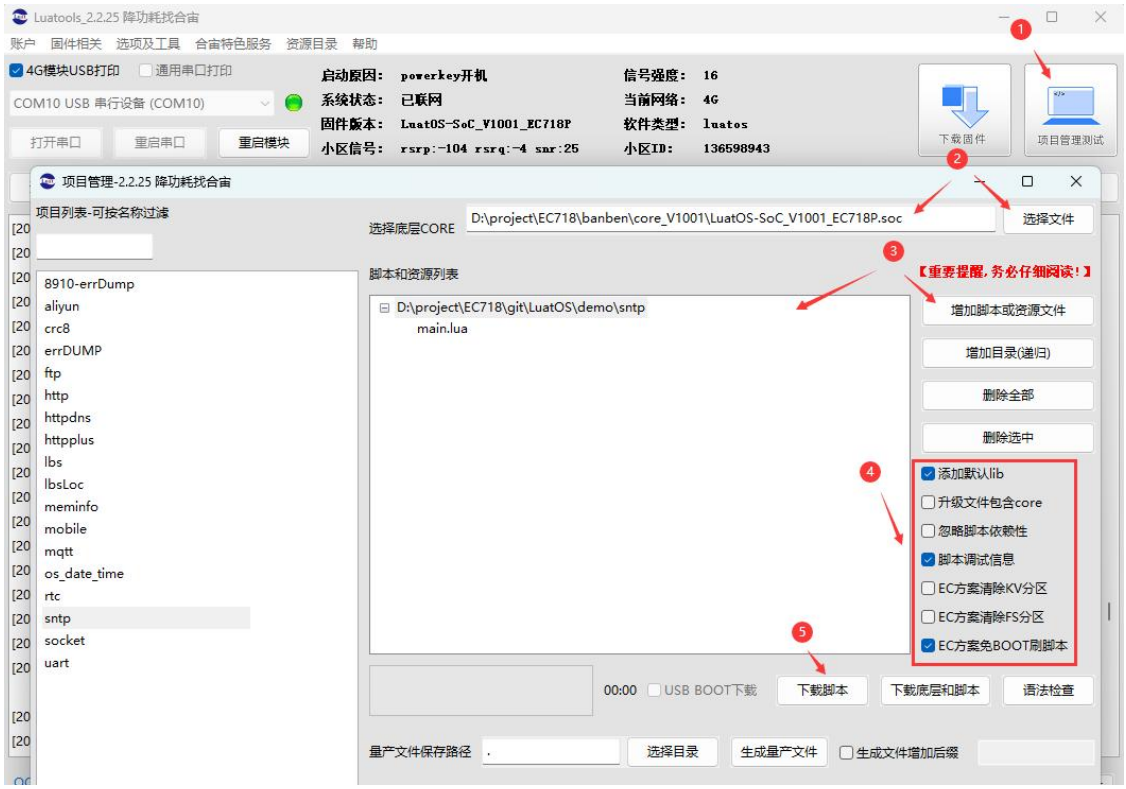
二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

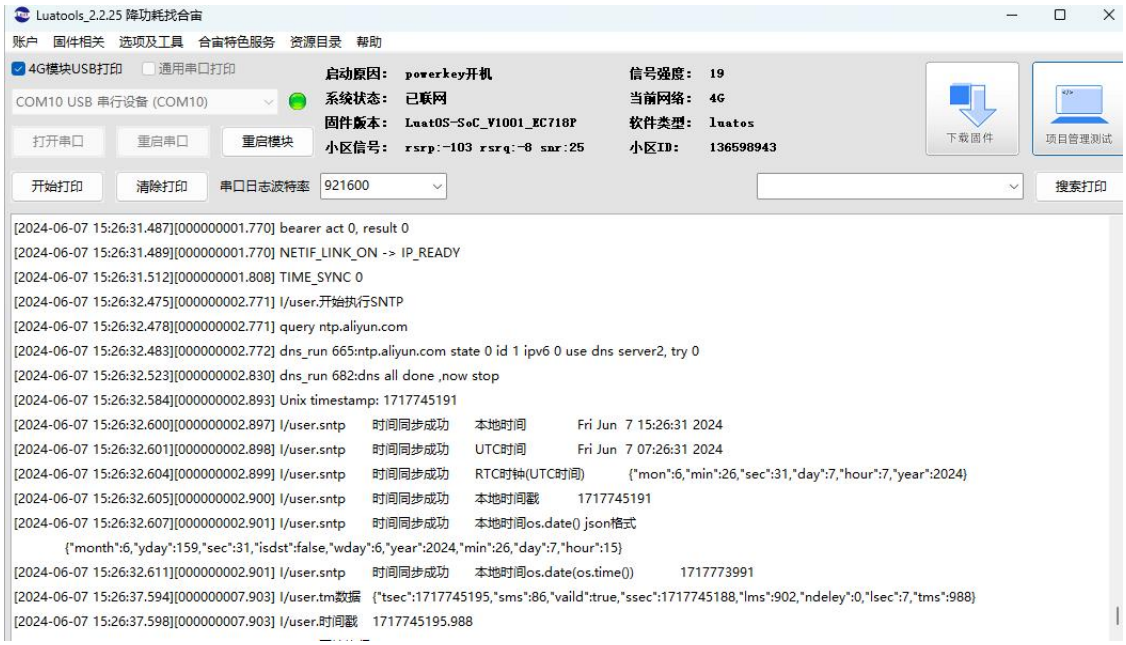
三．演示步骤

本文以 demo/sntp 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块打印 log



四．代码解析

完整脚本代码点击右侧查看 demo/sntp/main.lua

对于 Cat.1 模块，移动/电信卡，通常会下发基站时间， 那么 sntp 就不是必要的，而联通卡通常不会下发，就需要 sntp 了

sntp 内置了几个常用的 ntp 服务器，也支持自选服务器

socket.sntp()：sntp 时间同步，参数 sntp 服务器地址（选填）

时间同步成功，打印不同格式时间

os.date()：本地时间

os.date("!%c")：UTC 时间

json.encode(rtc.get())：RTC 时钟(UTC 时间)

os.time()：本地时间戳

local t = os.date("*t")

json.encode(t)：本地时间 os.date() json 格式

os.time(t)：本地时间 os.date(os.time())

```lua
    -- 对于Cat.1模块，移动/电信卡，通常会下发基站时间，那么sntp就不是必要的，而联通卡通常不会下发，就需要sntp了
    -- 对应ESP32系列模块，固件默认也会执行sntp，所以手动调用sntp也是可选的
    -- sntp内置了几个常用的ntp服务器，也支持自选服务器
    while 1 do
        -- 使用内置的ntp服务器地址，包括阿里ntp
        log.info("开始执行SNTP")
        socket.sntp()
        -- 自定义ntp地址
        -- socket.sntp("ntp.aliyun.com")
        -- socket.sntp({"baidu.com", "abc.com", "ntp.air32.cn"})
        -- 通常只需要几百毫秒就能成功
        local ret = sys.waitUntil("NTP_UPDATE", 5000)
        if ret then
            -- 以下是获取/打印时间的演示，注意时区问题
            log.info("sntp", "时间同步成功", "本地时间", os.date())
            log.info("sntp", "时间同步成功", "UTC时间", os.date("!%c"))
            log.info("sntp", "时间同步成功", "RTC时钟(UTC时间)", json.encode(rtc.get()))
            -- os.time(rtc.get()) 需要 2023.07.21 之后的版本，因为月份的命名差异mon/month
            -- log.info("sntp", "时间同步成功", "utc时间戳", os.time(rtc.get()))
            log.info("sntp", "时间同步成功", "本地时间戳", os.time())
            local t = os.date("*t")
            log.info("sntp", "时间同步成功", "本地时间os.date() json格式", json.encode(t))
            log.info("sntp", "时间同步成功", "本地时间os.date(os.time())", os.time(t))
            -- log.info("sntp", "时间同步成功", "本地时间", os.time())
            -- 正常使用，一小时一次，已经足够了，甚至1天一次也可以
            -- sys.wait(3600000)
            -- 这里为了演示，用5秒一次
            sys.wait(5000)
        else
            log.info("sntp", "时间同步失败")
            sys.wait(60000) -- 1分钟后重试
        end

        -- 时间戳，精确到毫秒. 2023.11.15 新增
        -- 注意，至少成功完成2次sntp，该时间戳才比较准确
        -- 如果仅完成了一次sntp，时间戳比标准时间会慢一个网络延时的时长(10~500ms)不等
        if socket.ntptm then
            local tm = socket.ntptm()
            log.info("tm数据", json.encode(tm))
            log.info("时间戳", string.format("%d.%03d", tm.tsec, tm.tms))
            sys.wait(5000)
        end
    end
end)
```

五．常见问题

本功能模块采用多个免费公共的 NTP 服务器来同步时间，并不能保证任何时间任何地点都能百分百同步到正确的时间，所以，如果用户项目中的业务逻辑严格依赖于时间同步功能，则不要使用使用本功能模块，建议使用自己的应用服务器来同步时间。

10.5.15 crypto 开发应用指南

一．概述

crypto 是加密模块，提供了一套常见的加密算法和校验算法．可以涵盖信息的加解密传输，登陆认证，信息签名等大多数使用场景.主要包含 crc16,crc8,crc7,sha256, sha1,hmac_ md5,md5, base64．这几种算法.

二．材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。

2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

三．演示步骤

加解密算法结果，可对照

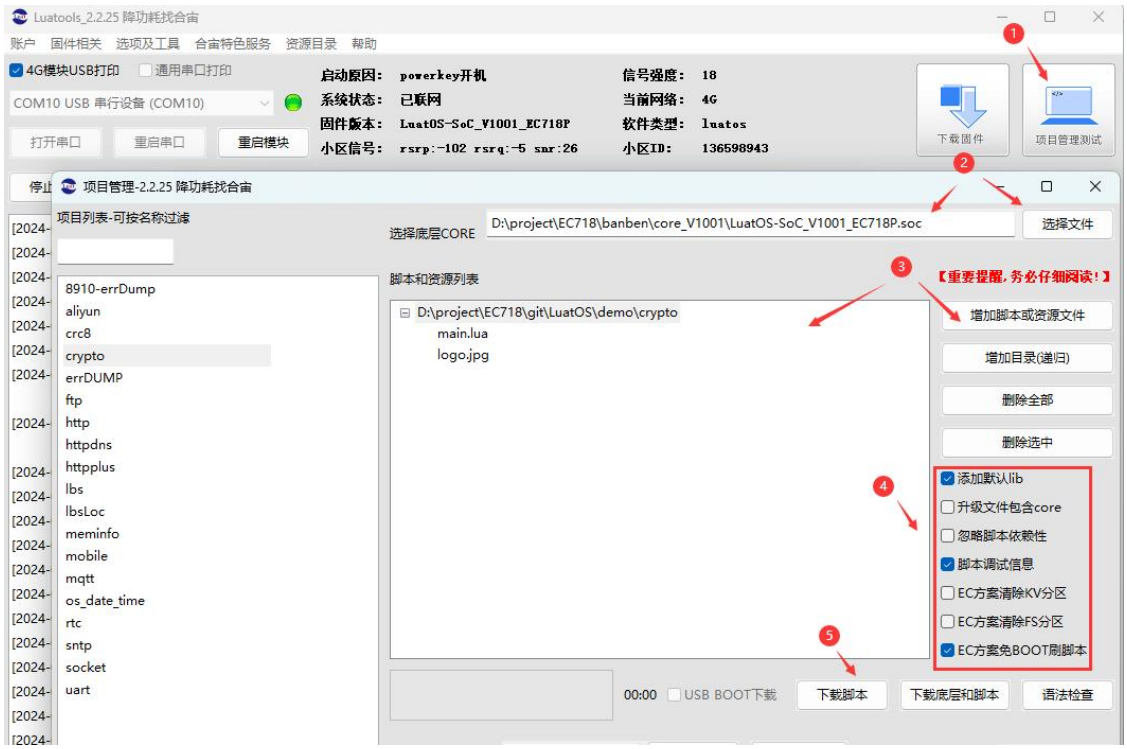
http://tool.oschina.net/encrypt?type=2

http://www.ip33.com/crc.html

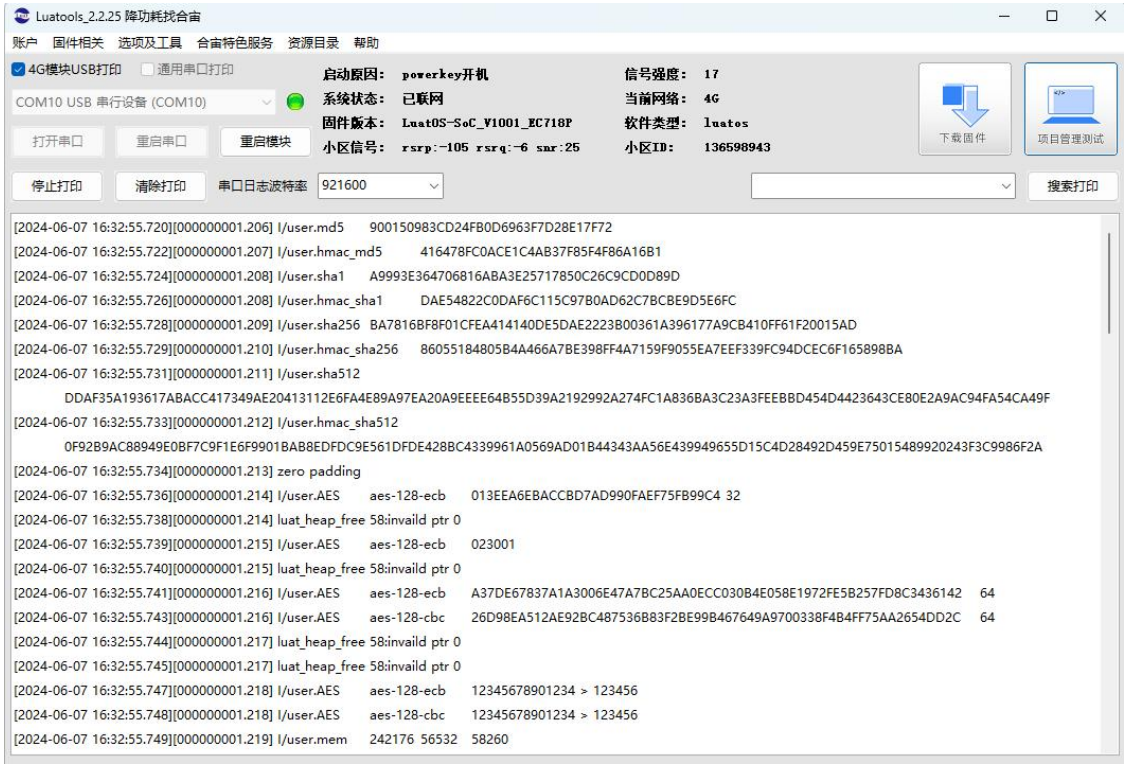
进行测试

本文以 demo/crypto 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本


Luatools_2.2.25 降功耗找合宙
账户 固件相关 选项及工具 合宙特色服务 资源目录 帮助
4G模块USB打印
通用串口打印
COM10 USB 串行设备 (COM10)
打开串口
重启串口
重启模块
启动原因: powerkey开机
系统状态: 已联网
固件版本: LuatOS-SoC_V1001_EC718P
小区信号: rsrp:-102 rsrq:-5 snr:26
信号强度: 18
当前网络: 4G
软件类型: luatos
小区ID: 136598943
下载固件
项目管理测试
项目管理-2.2.25 降功耗找合宙
项目列表-可按名称过滤
8910-errDump
aliyun
crc8
crypto
errDUMP
ftp
http
httpdns
httpplus
lbs
lbsLoc
meminfo
mobile
mqtt
os_date_time
rtc
sntp
socket
uart
选择底层CORE
D:\project\EC718\banben\core_V1001\LuatOS-SoC_V1001_EC718P.soc
选择文件
脚本和资源列表
D:\project\EC718\git\LuatOS\demo\crypto
main.lua
logo.jpg
【重要提醒，务必仔细阅读！】
增加脚本或资源文件
增加目录(递归)
删除全部
删除选中
添加默认lib
升级文件包含core
忽略脚本依赖性
脚本调试信息
EC方案清除KV分区
EC方案清除FS分区
EC方案免BOOT刷脚本
00:00
USB BOOT下载
下载脚本
下载底层和脚本
语法检查

模块打印 log


Luatools_2.2.25 降功耗找合宙
账户 固件相关 选项及工具 合宙特色服务 资源目录 帮助
4G模块USB打印
通用串口打印
COM10 USB 串行设备 (COM10)
打开串口
重启串口
重启模块
启动原因: powerkey开机
系统状态: 已联网
固件版本: LuatOS-SoC_V1001_EC718P
小区信号: rsrp:-105 rsrq:-6 snr:25
信号强度: 17
当前网络: 4G
软件类型: luatos
小区ID: 136598943
下载固件
项目管理测试
停止打印
清除打印
串口日志波特率
921600
搜索打印
[2024-06-07 16:32:55.720][000000001.206] I/user.md5 900150983CD24FB0D6963F7D28E17F72
[2024-06-07 16:32:55.722][000000001.207] I/user.hmac_md5 416478FC0ACE1C4AB37F85F4F86A16B1
[2024-06-07 16:32:55.724][000000001.208] I/user.sha1 A9993E364706816ABA3E25717850C26C9CD0D89D
[2024-06-07 16:32:55.726][000000001.208] I/user.hmac_sha1 DAE54822C0DAF6C115C97B0AD62C7BCBE9D5E6FC
[2024-06-07 16:32:55.728][000000001.209] I/user.sha256 BA7816BF8F01CFEA414140DE5DAE2223B00361A396177A9CB410FF61F20015AD
[2024-06-07 16:32:55.729][000000001.210] I/user.hmac_sha256 86055184805B4A466A7BE398FF4A7159F9055EA7EEF339FC94DCEC6F165898BA
[2024-06-07 16:32:55.731][000000001.211] I/user.sha512
DDAF35A193617ABACC417349AE20413112E6FA4E89A97EA20A9EEEE64B55D39A2192992A274FC1A836BA3C23A3FEEBBD454D4423643CE80E2A9AC94FA54CA49F
[2024-06-07 16:32:55.733][000000001.212] I/user.hmac_sha512
0F92B9AC88949E0BF7C9F1E6F9901BAB8EDFDC9E561DFDE428BC4339961A0569AD01B44343AA56E439949655D15C4D28492D459E75015489920243F3C9986F2A
[2024-06-07 16:32:55.734][000000001.213] zero padding
[2024-06-07 16:32:55.736][000000001.214] I/user.AES aes-128-ecb 013EEA6EBACCBD7AD990FAEF75FB99C4 32
[2024-06-07 16:32:55.738][000000001.214] luat_heap_free 58:invaild ptr 0
[2024-06-07 16:32:55.739][000000001.215] I/user.AES aes-128-ecb 023001
[2024-06-07 16:32:55.740][000000001.215] luat_heap_free 58:invaild ptr 0
[2024-06-07 16:32:55.741][000000001.216] I/user.AES aes-128-ecb A37DE67837A1A3006E47A7BC25AA0ECC030B4E058E1972FE5B257FD8C3436142 64
[2024-06-07 16:32:55.743][000000001.216] I/user.AES aes-128-cbc 26D98EA512AE92BC487536B83F2BE99B467649A9700338F4B4FF75AA2654DD2C 64
[2024-06-07 16:32:55.744][000000001.217] luat_heap_free 58:invaild ptr 0
[2024-06-07 16:32:55.745][000000001.217] luat_heap_free 58:invaild ptr 0
[2024-06-07 16:32:55.747][000000001.218] I/user.AES aes-128-ecb 12345678901234 > 123456
[2024-06-07 16:32:55.748][000000001.218] I/user.AES aes-128-cbc 12345678901234 > 123456
[2024-06-07 16:32:55.749][000000001.219] I/user.mem 242176 56532 58260

```
[2024-06-07 16:32:55.757][000000001.220] I/user.des      486CB8B81CACCDB7     16
[2024-06-07 16:32:55.760][000000001.220] luat_heap_free 58:invaild ptr 0
[2024-06-07 16:32:55.762][000000001.220] I/user.des      abcdefg
[2024-06-07 16:32:55.767][000000001.222] I/user.3des     14619F067B425995D0CD975B85491D98  32
[2024-06-07 16:32:55.769][000000001.223] luat_heap_free 58:invaild ptr 0
[2024-06-07 16:32:55.771][000000001.223] I/user.3des     abcdefg!!--ZZSS
[2024-06-07 16:32:55.773][000000001.225] I/user.cipher   list       ["AES-128-ECB","AES-192-ECB","AES-256-ECB","AES-128-CBC","AES-192-CBC","AES-256-CBC","AES-128-
CTR","AES-192-CTR","AES-256-CTR","AES-128-CCM","AES-192-CCM","AES-256-CCM","DES-ECB","DES-EDE-ECB","DES-EDE3-ECB","DES-CBC","DES-EDE-CBC","DES-EDE3-CBC"]
[2024-06-07 16:32:55.778][000000001.230] I/user.cipher   suites
[2024-06-07 16:32:55.783][000000001.231] ["TLS-ECDHE-ECDSA-WITH-AES-256-CCM","TLS-ECDHE-ECDSA-WITH-AES-256-CBC-SHA384","TLS-ECDHE-RSA-WITH-AES-256-CBC-
SHA384","TLS-ECDHE-ECDSA-WITH-AES-256-CBC-SHA","TLS-ECDHE-RSA-WITH-AES-256-CBC-SHA","TLS-ECDHE-ECDSA-WITH-AES-256-CCM-8","TLS-ECDHE-ECDSA-WITH-
AES-128-CCM","TLS-ECDHE-ECDSA-WITH-AES-128-CBC-SHA256","TLS-ECDHE-RSA-WITH-AES-128-CBC-SHA256","TLS-ECDHE-ECDSA-WITH-AES-128-CBC-SHA","TLS-ECDHE-
RSA-WITH-AES-128-CBC-SHA","TLS-ECDHE-ECDSA-WITH-AES-128-CCM-8","TLS-RSA-WITH-AES-256-CCM","TLS-RSA-WITH-AES-256-CBC-SHA256","TLS-RSA-WITH-AES-256-
CBC-SHA","TLS-RSA-WITH-AES-256-CCM-8","TLS-RSA-WITH-AES-128-CCM","TLS-RSA-WITH-AES-128-CBC-SHA256","TLS-RSA-WITH-AES-128-CBC-SHA","TLS-RSA-WITH-
AES-128-CCM-8","TLS-RSA-PSK-WITH-AES-256-CBC-SHA384","TLS-RSA-PSK-WITH-AES-256-CBC-SHA","TLS-RSA-PSK-WITH-AES-128-CBC-SHA256","TLS-RSA-PSK-WITH-
AES-128-CBC-SHA","TLS-PSK-WITH-AES-256-CCM","TLS-PSK-WITH-AES-256-CBC-SHA384","TLS-PSK-WITH-AES-256-CBC-SHA","TLS-PSK-WITH-AES-256-CCM-8","TLS-PSK-WITH-
AES-128-CCM","TLS-PSK-WITH-AES-128-CBC-SHA256","TLS-PSK-WITH-AES-128-CBC-SHA","TLS-PSK-WITH-AES-128-CCM-8"]
[2024-06-07 16:32:55.795][000000001.231] ["TLS-ECDHE-
[2024-06-07 16:32:55.798][000000001.231] I/user.随机数测试
[2024-06-07 16:32:55.843][000000001.340] I/user.crypto   真随机数          -2111211222     5
[2024-06-07 16:32:55.954][000000001.441] I/user.crypto   真随机数          763572739       5
[2024-06-07 16:32:56.049][000000001.542] I/user.crypto   真随机数          -1683540523     5
```

```
[2024-06-07 16:32:56.143][000000001.643] I/user.crypto   真随机数          -765512938      5
[2024-06-07 16:32:56.250][000000001.744] I/user.crypto   真随机数          1643752047      5
[2024-06-07 16:32:56.359][000000001.845] I/user.crypto   真随机数          -604517806      5
[2024-06-07 16:32:56.386][000000001.862] cid1, state0
[2024-06-07 16:32:56.388][000000001.863] bearer act 0, result 0
[2024-06-07 16:32:56.389][000000001.863] NETIF_LINK_ON -> IP_READY
[2024-06-07 16:32:56.513][000000002.005] I/user.crypto   真随机数          1475639881      5
[2024-06-07 16:32:56.516][000000002.006] TIME_SYNC 0
[2024-06-07 16:32:56.620][000000002.106] I/user.crypto   真随机数          1302425173      5
[2024-06-07 16:32:56.713][000000002.207] I/user.crypto   真随机数          1894895762      5
[2024-06-07 16:32:56.823][000000002.308] I/user.crypto   真随机数          -423085273      5
[2024-06-07 16:32:56.832][000000002.308] I/user.totp的密钥
[2024-06-07 16:32:56.847][000000002.310] I/user.totp     522113 19       29      37
[2024-06-07 16:32:56.854][000000002.311] I/user.totp     964300 19       30      7
[2024-06-07 16:32:56.860][000000002.311] I/user.totp     987714 19       30      37
[2024-06-07 16:32:56.865][000000002.313] I/user.totp     037499 19       31      7
[2024-06-07 16:32:56.870][000000002.314] I/user.totp     699697 19       31      37
[2024-06-07 16:32:56.874][000000002.315] I/user.totp     548191 19       32      7
[2024-06-07 16:32:56.879][000000002.316] I/user.totp     747517 19       32      37
[2024-06-07 16:32:56.883][000000002.317] I/user.totp     243319 19       33      7
[2024-06-07 16:32:56.890][000000002.317] I/user.totp     147474 19       33      37
[2024-06-07 16:32:56.895][000000002.318] I/user.totp     039992 19       34      7
```

```
[2024-06-07 16:32:56.900][000000002.319] I/user.totp     628512 19       34      37
[2024-06-07 16:32:56.904][000000002.320] I/user.totp     529018 19       35      7
[2024-06-07 16:32:56.909][000000002.321] I/user.totp     994006 19       35      37
[2024-06-07 16:32:56.913][000000002.322] I/user.totp     851359 19       36      7
[2024-06-07 16:32:56.917][000000002.323] I/user.totp     943237 19       36      37
[2024-06-07 16:32:56.919][000000002.324] I/user.totp     410702 19       37      7
[2024-06-07 16:32:56.921][000000002.325] I/user.totp     082993 19       37      37
[2024-06-07 16:32:56.923][000000002.326] I/user.totp     193281 19       38      7
[2024-06-07 16:32:56.928][000000002.327] I/user.totp     781573 19       38      37
[2024-06-07 16:32:56.930][000000002.328] I/user.totp     243288 19       39      7
[2024-06-07 16:32:56.933][000000002.328] I/user.文件hash值测试
[2024-06-07 16:32:56.935][000000002.340] I/user.md5      D364D04CCD734D2757B25F3216CC431B
[2024-06-07 16:32:56.937][000000002.347] I/user.sha1     04DA64874D415B1FB9CDD2E89927397D8F48C441
[2024-06-07 16:32:56.940][000000002.362] I/user.sha256   0C601722B4BD2BC1A76BC3701F4EB646F5119C31702852978C326CD8D7C9212C
[2024-06-07 16:32:56.944][000000002.373] I/user.hmac_md5         B5F5A687DB904DB67D3311899932DD61
[2024-06-07 16:32:56.948][000000002.380] I/user.hmac_sha1        2A0B55C290D54ADF2F185EAE460254D267C7CE1F
[2024-06-07 16:32:56.950][000000002.396] I/user.hmac_sha256      6E443C84D4D7A4721A9A195EB9038EDCFA8A8F60A3F1E8A0090B57B3BAB4942E
[2024-06-07 16:32:56.952][000000002.396] I/user.checksum         OK      04      2
[2024-06-07 16:32:56.954][000000002.397] I/user.checksum         357E    E4      2
[2024-06-07 16:32:56.956][000000002.397] I/user.流式hash测试
[2024-06-07 16:32:56.961][000000002.398] I/user.md5_stream       F5BF3E984432AE6F9F98840951E5CEF3
[2024-06-07 16:32:56.963][000000002.398] I/user.md5      F5BF3E984432AE6F9F98840951E5CEF3
```

```
[2024-06-07 16:32:56.966][000000002.399] I/user.hmac_md5_stream        45527D9407615C3A44F475BB7172752A
[2024-06-07 16:32:56.968][000000002.400] I/user.hmac_md5        45527D9407615C3A44F475BB7172752A
[2024-06-07 16:32:56.970][000000002.401] I/user.sha1_stream     C61A2C245CB07A04482CE5B662AE67DBDBE010DB
[2024-06-07 16:32:56.972][000000002.401] I/user.sha1    C61A2C245CB07A04482CE5B662AE67DBDBE010DB
[2024-06-07 16:32:56.974][000000002.402] I/user.hmac_sha1_stream        88471065B8C5F64057418A0A58353A46E7841DE7
[2024-06-07 16:32:56.979][000000002.403] I/user.hmac_sha1       88471065B8C5F64057418A0A58353A46E7841DE7
[2024-06-07 16:32:56.982][000000002.403] I/user.sha256_stream   A4EBDD541454B84CC670C9F1F5508BAF67FFD3FE59B883267808781F992A0B1D
[2024-06-07 16:32:56.984][000000002.404] I/user.sha256  A4EBDD541454B84CC670C9F1F5508BAF67FFD3FE59B883267808781F992A0B1D
[2024-06-07 16:32:56.986][000000002.405] I/user.hmac_sha256_stream      EAF715932F064E462893B7FE04442E2C25ECF2F7C560820A648D0D94BEAEB581
[2024-06-07 16:32:56.989][000000002.406] I/user.hmac_sha256     EAF715932F064E462893B7FE04442E2C25ECF2F7C560820A648D0D94BEAEB581
[2024-06-07 16:32:56.991][000000002.407] I/user.sha512_stream
        3A8529D8F0C7B1AD2FA54C944952829B718D5BEB4FF9BA8F4A849E02FE9A272DAF59AE3BD06DDE6F01DF863D87C8BA4AB016AC576B59A19078C26D8DBE63F79E
[2024-06-07 16:32:56.995][000000002.407] I/user.sha512
        3A8529D8F0C7B1AD2FA54C944952829B718D5BEB4FF9BA8F4A849E02FE9A272DAF59AE3BD06DDE6F01DF863D87C8BA4AB016AC576B59A19078C26D8DBE63F79E
[2024-06-07 16:32:56.997][000000002.409] I/user.hmac_sha512_stream
        18F74340A0048090521796F52A66A45B77AF5981512951B938BFCB026F2D6ED7945B0682731E4E7E3CA72021E5E8AB9810769E48C83F440DC73D6C942215E29E
[2024-06-07 16:32:57.000][000000002.410] I/user.hmac_sha512
        18F74340A0048090521796F52A66A45B77AF5981512951B938BFCB026F2D6ED7945B0682731E4E7E3CA72021E5E8AB9810769E48C83F440DC73D6C942215E29E
[2024-06-07 16:32:57.002][000000002.410] I/user.crc7测试
[2024-06-07 16:32:57.005][000000002.411] I/user.crc7测试124     7C
[2024-06-07 16:32:57.007][000000002.411] I/user.crypto    ALL Done
```

四．代码解析

完整脚本代码点击右侧查看 crypto/main.lua

直接调用对应接口即可，接口说明点击右侧查看 crypto - 加解密和 hash 函数

```lua
-- MD5,输出结果已经hex编码
log.info("md5", crypto.md5("abc"))
log.info("hmac_md5", crypto.hmac_md5("abc", "1234567890"))

-- SHA1,输出结果已经hex编码
log.info("sha1", crypto.sha1("abc"))
log.info("hmac_sha1", crypto.hmac_sha1("abc", "1234567890"))

-- SHA256,输出结果已经hex编码
log.info("sha256", crypto.sha256("abc"))
log.info("hmac_sha256", crypto.hmac_sha256("abc", "1234567890"))

-- SHA512,输出结果已经hex编码
log.info("sha512", crypto.sha512("abc"))
log.info("hmac_sha512", crypto.hmac_sha512("abc", "1234567890"))

local data_encrypt = crypto.cipher_encrypt("AES-128-ECB", "ZERO", "023001", "HZBIT@WLW/YSBKEY")
log.info("AES", "aes-128-ecb", data_encrypt:toHex())
local data_decrypt = crypto.cipher_decrypt("AES-128-ECB", "ZERO", data_encrypt, "HZBIT@WLW/YSBKEY")
log.info("AES", "aes-128-ecb", data_decrypt)

-- AES加密, 未经Hex编码. AES-128-ECB 算法,待加密字符串如果超过32字节会报错,徒蚕. by wendal 20200812
local data_encrypt = crypto.cipher_encrypt("AES-128-ECB", "PKCS7", "12345678901234 > 123456", "1234567890123456")
local data2_encrypt = crypto.cipher_encrypt("AES-128-CBC", "PKCS7", "12345678901234 > 123456", "1234567890123456", "1234567890666666")
log.info("AES", "aes-128-ecb", data_encrypt:toHex())
log.info("AES", "aes-128-cbc", data2_encrypt:toHex())

-- AES解密, 未经Hex编码
local data_decrypt = crypto.cipher_decrypt("AES-128-ECB", "PKCS7", data_encrypt, "1234567890123456")
local data2_decrypt = crypto.cipher_decrypt("AES-128-CBC", "PKCS7", data2_encrypt, "1234567890123456", "1234567890666666")
log.info("AES", "aes-128-ecb", data_decrypt)
log.info("AES", "aes-128-cbc", data2_decrypt)
log.info("mem", rtos.meminfo("sys"))
```

```
-- DES-ECB 加解密
local data1 = crypto.cipher_encrypt("DES-ECB", "PKCS7", "abcdefg", "12345678")
if data1 then -- DES-ECB 在某些平台不支持的
    log.info("des", data1:toHex())
    local data2 = crypto.cipher_decrypt("DES-ECB", "PKCS7", data1, "12345678")
    log.info("des", data2)
else
    log.info("des", "当前固件不支持DES/3DES")
end

-- 3DES-ECB 加解密
local data1 = crypto.cipher_encrypt("DES-EDE3-ECB", "PKCS7", "abcdefg!!--zzss", "123456781234567812345678")
if data1 then -- DES-ECB 在某些平台不支持的
    log.info("3des", data1:toHex())
    local data2 = crypto.cipher_decrypt("DES-EDE3-ECB", "PKCS7", data1, "123456781234567812345678")
    log.info("3des", data2)
else
    log.info("3des", "当前固件不支持DES/3DES")
end


-- 打印所有支持的cipher
if crypto.cipher_list then
    log.info("cipher", "list", json.encode(crypto.cipher_list()))
else
    log.info("cipher", "当前固件不支持crypto.cipher_list")
end
-- 打印所有支持的cipher suites
if crypto.cipher_suites then
    log.info("cipher", "suites", json.encode(crypto.cipher_suites()))
else
    log.info("cipher", "当前固件不支持crypto.cipher_suites")
end


 ----------------------------------------
 log.info("随机数测试")
 for i=1, 10 do
     sys.wait(100)
     log.info("crypto", "真随机数",string.unpack("I",crypto.trng(4)))
     -- log.info("crypto", "伪随机数",math.random()) -- 输出的是浮点数,不推荐
     -- log.info("crypto", "伪随机数",math.random(1, 65525)) -- 不推荐
 end

 -- totp的密钥
 log.info("totp的密钥")
 local secret = "VK54ZXPO74ISEM2E"
 --写死时间戳用来测试
 local ts = 1646796576
 --生成十分钟的动态码验证下
 for i=1,600,30 do
     local r = crypto.totp(secret,ts+i)
     local time = os.date("*t",ts+i + 8*3600)--东八区
     log.info("totp", string.format("%06d" ,r),time.hour,time.min,time.sec)
 end

 -- 文件测试
 log.info("文件hash值测试")
 if crypto.md_file then
     log.info("md5", crypto.md_file("MD5", "/luadb/logo.jpg"))
     log.info("sha1", crypto.md_file("SHA1", "/luadb/logo.jpg"))
     log.info("sha256", crypto.md_file("SHA256", "/luadb/logo.jpg"))

     log.info("hmac_md5", crypto.md_file("MD5", "/luadb/logo.jpg", "123456"))
     log.info("hmac_sha1", crypto.md_file("SHA1", "/luadb/logo.jpg", "123456"))
     log.info("hmac_sha256", crypto.md_file("SHA256", "/luadb/logo.jpg", "123456"))
 else
     log.info("文件hash值测试", "当前固件不支持crypto.md_file")
 end
```

```lua
if crypto.checksum then
    log.info("checksum", "OK", string.char(crypto.checksum("OK")):toHex())
    log.info("checksum", "357E", string.char(crypto.checksum("357E", 1)):toHex())
else
    log.info("checksum", "当前固件不支持crypto.checksum")
end

-- 流式hash测试
log.info("流式hash测试")
if crypto.hash_init then
    -- MD5
    local md5_obj = crypto.hash_init("MD5")
    crypto.hash_update(md5_obj, "1234567890")
    crypto.hash_update(md5_obj, "1234567890")
    crypto.hash_update(md5_obj, "1234567890")
    crypto.hash_update(md5_obj, "1234567890")
    local md5_result = crypto.hash_finish(md5_obj)
    log.info("md5_stream", md5_result)
    log.info("md5", crypto.md5("12345678901234567890123456789012345678901234567890"))

    -- HMAC_MD5
    local hmac_md5_obj = crypto.hash_init("MD5", "1234567890")
    crypto.hash_update(hmac_md5_obj, "1234567890")
    crypto.hash_update(hmac_md5_obj, "1234567890")
    crypto.hash_update(hmac_md5_obj, "1234567890")
    crypto.hash_update(hmac_md5_obj, "1234567890")
    local hmac_md5_result = crypto.hash_finish(hmac_md5_obj)
    log.info("hmac_md5_stream", hmac_md5_result)
    log.info("hmac_md5", crypto.hmac_md5("12345678901234567890123456789012345678901234567890", "1234567890"))

-- SHA1
local sha1_obj = crypto.hash_init("SHA1")
crypto.hash_update(sha1_obj, "1234567890")
crypto.hash_update(sha1_obj, "1234567890")
crypto.hash_update(sha1_obj, "1234567890")
crypto.hash_update(sha1_obj, "1234567890")
local sha1_result = crypto.hash_finish(sha1_obj)
log.info("sha1_stream", sha1_result)
log.info("sha1", crypto.sha1("12345678901234567890123456789012345678901234567890"))

-- HMAC_SHA1
local hmac_sha1_obj = crypto.hash_init("SHA1", "1234567890")
crypto.hash_update(hmac_sha1_obj, "1234567890")
crypto.hash_update(hmac_sha1_obj, "1234567890")
crypto.hash_update(hmac_sha1_obj, "1234567890")
crypto.hash_update(hmac_sha1_obj, "1234567890")
local hmac_sha1_result = crypto.hash_finish(hmac_sha1_obj)
log.info("hmac_sha1_stream", hmac_sha1_result)
log.info("hmac_sha1", crypto.hmac_sha1("12345678901234567890123456789012345678901234567890", "1234567890"))

-- SHA256
local sha256_obj = crypto.hash_init("SHA256")
crypto.hash_update(sha256_obj, "1234567890")
crypto.hash_update(sha256_obj, "1234567890")
crypto.hash_update(sha256_obj, "1234567890")
crypto.hash_update(sha256_obj, "1234567890")
local sha256_result = crypto.hash_finish(sha256_obj)
log.info("sha256_stream", sha256_result)
log.info("sha256", crypto.sha256("12345678901234567890123456789012345678901234567890"))
```

```
    -- HMAC_SHA256
    local hmac_sha256_obj = crypto.hash_init("SHA256", "1234567890")
    crypto.hash_update(hmac_sha256_obj, "1234567890")
    crypto.hash_update(hmac_sha256_obj, "1234567890")
    crypto.hash_update(hmac_sha256_obj, "1234567890")
    crypto.hash_update(hmac_sha256_obj, "1234567890")
    local hmac_sha256_result = crypto.hash_finish(hmac_sha256_obj)
    log.info("hmac_sha256_stream", hmac_sha256_result)
    log.info("hmac_sha256", crypto.hmac_sha256("12345678901234567890123456789012345678901234567890", "1234567890"))

    -- SHA512
    local sha512_obj = crypto.hash_init("SHA512")
    if sha512_obj then
        crypto.hash_update(sha512_obj, "1234567890")
        crypto.hash_update(sha512_obj, "1234567890")
        crypto.hash_update(sha512_obj, "1234567890")
        crypto.hash_update(sha512_obj, "1234567890")
        local sha512_result = crypto.hash_finish(sha512_obj)
        log.info("sha512_stream", sha512_result)
        log.info("sha512", crypto.sha512("12345678901234567890123456789012345678901234567890"))
    end

    -- HMAC_SHA512
    local hmac_sha512_obj = crypto.hash_init("SHA512", "1234567890")
    if hmac_sha512_obj then
        crypto.hash_update(hmac_sha512_obj, "1234567890")
        crypto.hash_update(hmac_sha512_obj, "1234567890")
        crypto.hash_update(hmac_sha512_obj, "1234567890")
        crypto.hash_update(hmac_sha512_obj, "1234567890")
        local hmac_sha512_result = crypto.hash_finish(hmac_sha512_obj)
        log.info("hmac_sha512_stream", hmac_sha512_result)
        log.info("hmac_sha512", crypto.hmac_sha512("12345678901234567890123456789012345678901234567890", "1234567890"))
    end
else
    log.info("crypto", "当前固件不支持crypto.hash_init")
end
```

```
log.info("crc7测试")
if crypto.crc7 then
    local result = crypto.crc7(string.char(0xAA), 0xE5, 0x00)
    log.info("crc7测试", result, string.format("%02X", result))
else
    log.info("crypto", "当前固件不支持crypto.crc7")
end
```

10.5.16 rsa 开发应用指南

一. 概述

RSA 加密算法是一种公钥加密算法，由 Ron Rivest、Adi Shamir 和 Leonard Adleman 在 1977 年发明。RSA 算法的安全性基于数论中的两个重要问题：质因数分解和求离散对数。RSA 算法通过将一组大素数与其他参数结合起来生成公钥和私钥，用公钥加密数据，用私钥解密数据，实现了信息的安全传输。

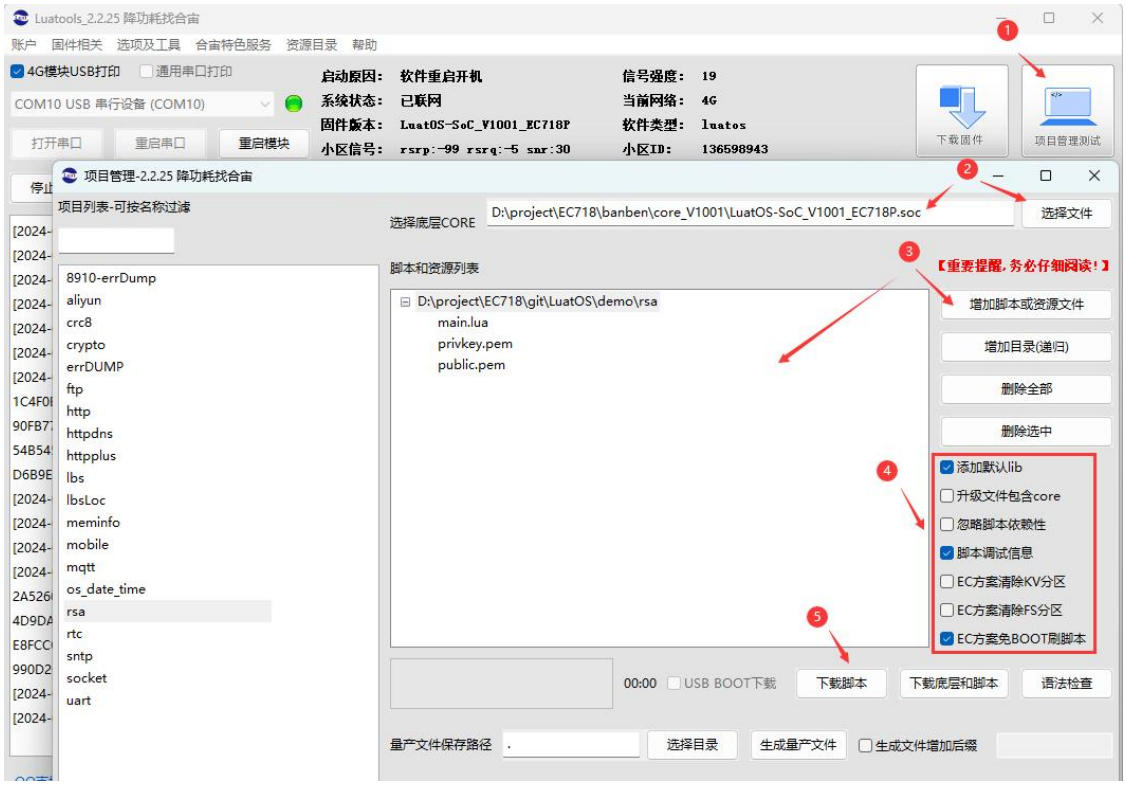
二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

三．演示步骤

本文以 demo/rsa 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块对应 log

四．代码解析

完整脚本代码点击右侧查看 demo/rsa

rsa.encrypt: 读取公钥并马上加密数据

第 1 个参数：公钥数据,仅支持 PEM 格式，第 2 个参数：待加密数据，不能超过公钥位数的一半，例如 2048bit 的公钥，只能加密 128 字节的数据。

返回值：加密成功后的数据,若失败会返回 nil。

rsa.decrypt：读取私钥,然后解码数据

第 1 个参数：私钥数据,仅支持 PEM 格式，第 2 个参数：待解密数据，第 3 个参数：私钥的密码,可选。

返回值：解密成功后的数据,若失败会返回 nil。

rsa.sign：签名

第 1 个参数私钥数据,仅支持 PEM 格式，第 2 个参数签名模式，例如 rsa.MD_SHA1，rsa.MD_SHA256，第 3 个参数 hash 数据，如果是 HEX 字符串,记得 fromHex 转二进制数据，第 4 个参数私钥的密码,可选。

返回值：成功返回 sig 数据，否则返回 nil

rsa.verify：验签

第 1 个参数私钥数据,仅支持 PEM 格式，第 2 个参数签名模式，例如 rsa.MD_SHA1，rsa.MD_SHA256，第 3 个参数 hash 数据，如果是 HEX 字符串,记得 fromHex 转二进制数据，第 4 个参数 ig 数据，如果是 HEX 字符串,记得 fromHex 转二进制数据。

返回值：有效返回 true,否则为 false,出错返回 nil

```
-- 读取公钥并马上加密数据
local res = rsa.encrypt((io.readFile("/luadb/public.pem")), "abc")
-- 打印结果
log.info("rsa", "encrypt", res and #res or 0, res and res:toHex() or "")

-- 下面是解密，通常不会在设备端进行，这里主要是演示用法，会很慢
if res then
    -- 读取私钥，然后解码数据
    local dst = rsa.decrypt((io.readFile("/luadb/privkey.pem")), res, "")
    log.info("rsa", "decrypt", dst and #dst or 0, dst and dst:toHex() or "")
end

-- 演示签名和验签
local hash = crypto.sha1("1234567890"):fromHex()
-- 签名通常很慢，通常是服务器做
local sig = rsa.sign((io.readFile("/luadb/privkey.pem")), rsa.MD_SHA1, hash, "")
log.info("rsa", "sign", sig and #sig or 0, sig and sig:toHex() or "")
if sig then
    -- 验签是很快的
    local ret = rsa.verify((io.readFile("/luadb/public.pem")), rsa.MD_SHA1, hash, sig)
    log.info("rsa", "verify", ret)
end
```

五．常见问题

如何生成公钥以及私钥文件

如果在线测试功能，可以访问网页 https://www.ssleye.com/pass_double.html

10.5.17 fskv 开发应用指南

一. 概述

fskv - kv 数据库,掉电不丢数据，功能上与 EEPROM 是类似的。本库的目标是替代 fdb 库。

fskv 与 fdb 的实现机制导致的差异

	fskv	fdb
1. value 长度	4096	255
2. key 长度	63	64
3. 空间利用率(对比)	较低	较高
4. 读取速度	恒定	脏数据影响速度,非恒
5. 写入数据	恒定	脏数据影响速度,非恒定
6. 均衡擦除	自动	自动

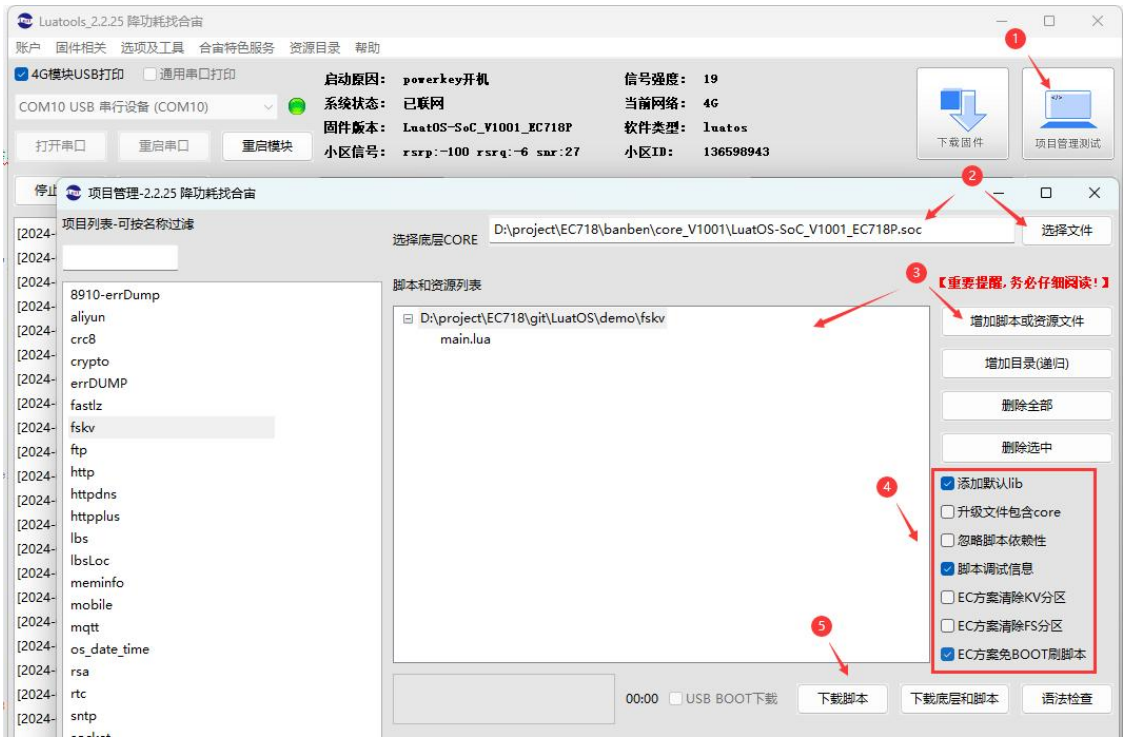
二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

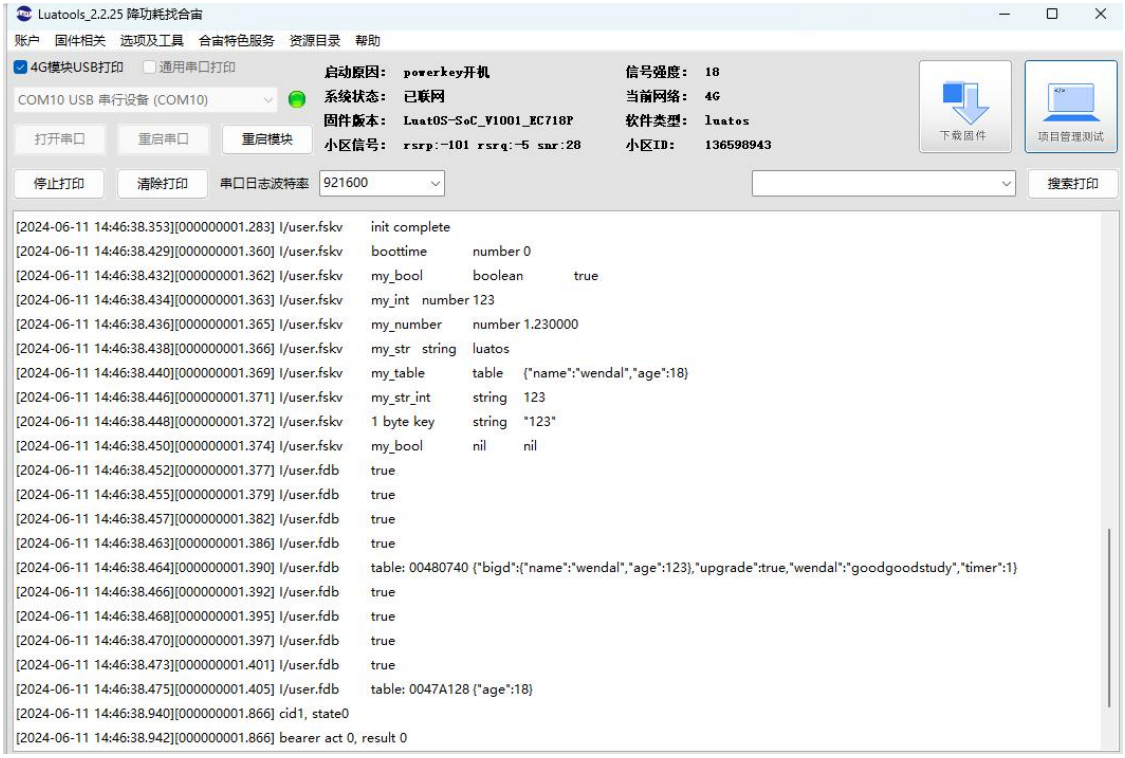
三. 演示步骤

本文以 demo/fskv 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块打印 log



四．代码解析

完整脚本代码点击右侧查看 demo/fskv/main.lua

fdb.kvdb_init：初始化 kv 数据库

fdb.kv_get：根据 key 获取对应的数据

第 1 个参数：key 的名称,必填,不能空字符串

第 2 个参数：可选的次级 key,仅当原始值为 table 时有效,相当于 fskv.get(key)[skey]

返回值：存在则返回数据,否则返回 nil

fdb.kv_set：设置一对 kv 数据

第 1 个参数：key 的名称,必填,不能空字符串

第 2 个参数：用户数据,必填,不能 nil，支持字符串/数值/table/布尔值，数据长度最大 4095 字节

返回值：成功返回 true,否则返回 false

```
-- 初始化kv数据库
fdb.kvdb_init("onchip_flash")
log.info("fdb", "init complete")
-- 先放入一堆值
local bootime = fdb.kv_get("boottime")
if bootime == nil or type(bootime) ~= "number" then
    bootime = 0
else
    bootime = bootime + 1
end
fdb.kv_set("boottime", bootime)

fdb.kv_set("my_bool", true)
fdb.kv_set("my_int", 123)
fdb.kv_set("my_number", 1.23)
fdb.kv_set("my_str", "luatos")
fdb.kv_set("my_table", {name="wendal",age=18})

fdb.kv_set("my_str_int", "123")
fdb.kv_set("1", "123") -- 单字节key
--fdb.kv_set("my_nil", nil) -- 会提示失败,不支持空值


log.info("fdb", "boottime",      type(fdb.kv_get("boottime")),    fdb.kv_get("boottime"))
log.info("fdb", "my_bool",      type(fdb.kv_get("my_bool")),    fdb.kv_get("my_bool"))
log.info("fdb", "my_int",       type(fdb.kv_get("my_int")),     fdb.kv_get("my_int"))
log.info("fdb", "my_number",    type(fdb.kv_get("my_number")),  fdb.kv_get("my_number"))
log.info("fdb", "my_str",       type(fdb.kv_get("my_str")),     fdb.kv_get("my_str"))
log.info("fdb", "my_table",     type(fdb.kv_get("my_table")),   json.encode(fdb.kv_get("my_table")))
log.info("fdb", "my_str_int",     type(fdb.kv_get("my_str_int")),   fdb.kv_get("my_str_int"))
log.info("fdb", "1 byte key",     type(fdb.kv_get("1")),   json.encode(fdb.kv_get("1")))
```

fdb.kv_del：根据 key 删除数据

参考：key 的名称,必填,不能空字符串

返回值：成功返回 true,否则返回 false

```
-- 删除测试
fdb.kv_del("my_bool")
local t = fdb.kv_get("my_bool")
log.info("fdb", "my_bool",       type(t),     t)
```

fdb.sett：设置 table 内的键值对数据

第 1 个参数：key 的名称,必填,不能空字符串

第 2 个参数：table 的 key 名称，必填，不能是空字符串

第 3 个参数：用户数据,必填,支持字符串/数值/table/布尔值，数据长度最大 4095 字节

返回值：成功返回 true,否则返回 false/nil

```lua
if fskv.sett then
    -- 设置数据，字符串,数值,table,布尔值,均可
    -- 但不可以是nil，function，userdata，task
    log.info("fdb", fskv.sett("mytable", "wendal", "goodgoodstudy"))
    log.info("fdb", fskv.sett("mytable", "upgrade", true))
    log.info("fdb", fskv.sett("mytable", "timer", 1))
    log.info("fdb", fskv.sett("mytable", "bigd", {name="wendal",age=123}))

    -- 下列语句将打印出4个元素的table
    log.info("fdb", fskv.get("mytable"), json.encode(fskv.get("mytable")))
    -- 注意：如果key不存在，或者原本的值不是table类型，将会完全覆盖
    -- 例如下列写法，最终获取到的是table，而非第一行的字符串
    log.info("fdb", fskv.set("mykv", "123"))
    log.info("fdb", fskv.sett("mykv", "age", "123")) -- 保存的将是 {age:"123"}

    -- 删除测试
    log.info("fdb", fskv.set("mytable", {age=18, name="wendal"}))
    log.info("fdb", fskv.sett("mytable", "name", nil))
    log.info("fdb", fskv.get("mytable"), json.encode(fskv.get("mytable")))
end
```

10.5.18 fs 开发应用指南

一．概述

Air780EP 模组内置的 Flash 总空间为 4MB，**文件系统分区**：用户代码运行过程中动态创建的文件存储在此分区，总空间 128KB。可以通过 fs.fsstat(path)获取文件系统信息。

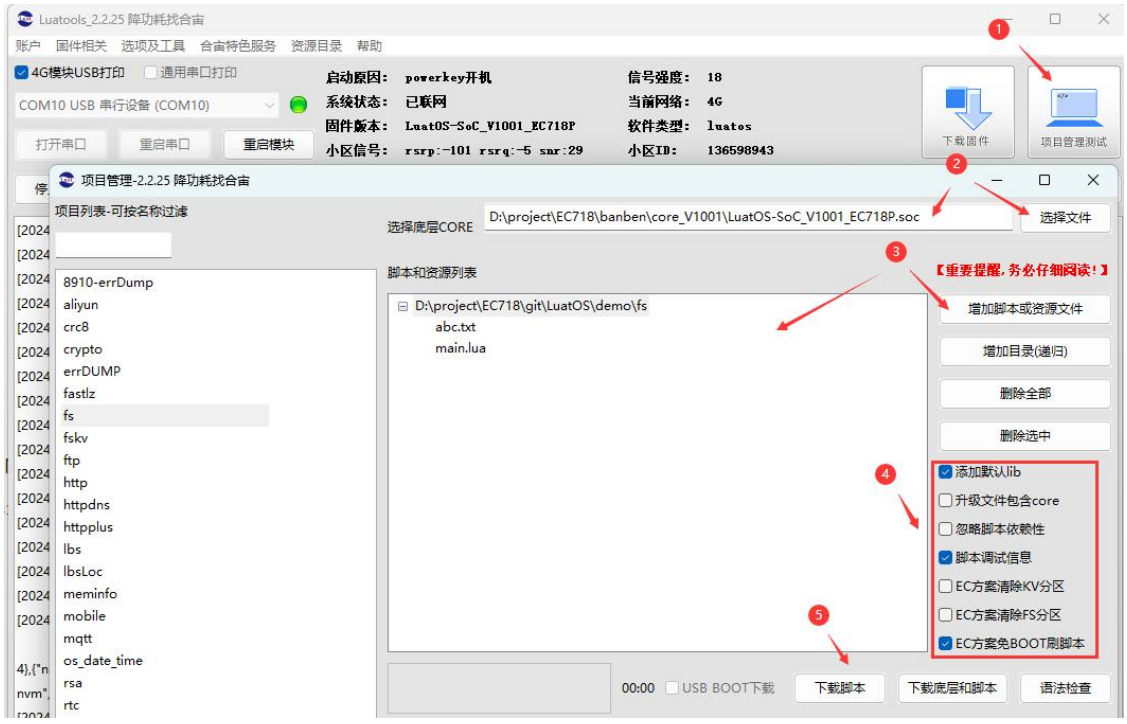
二．材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

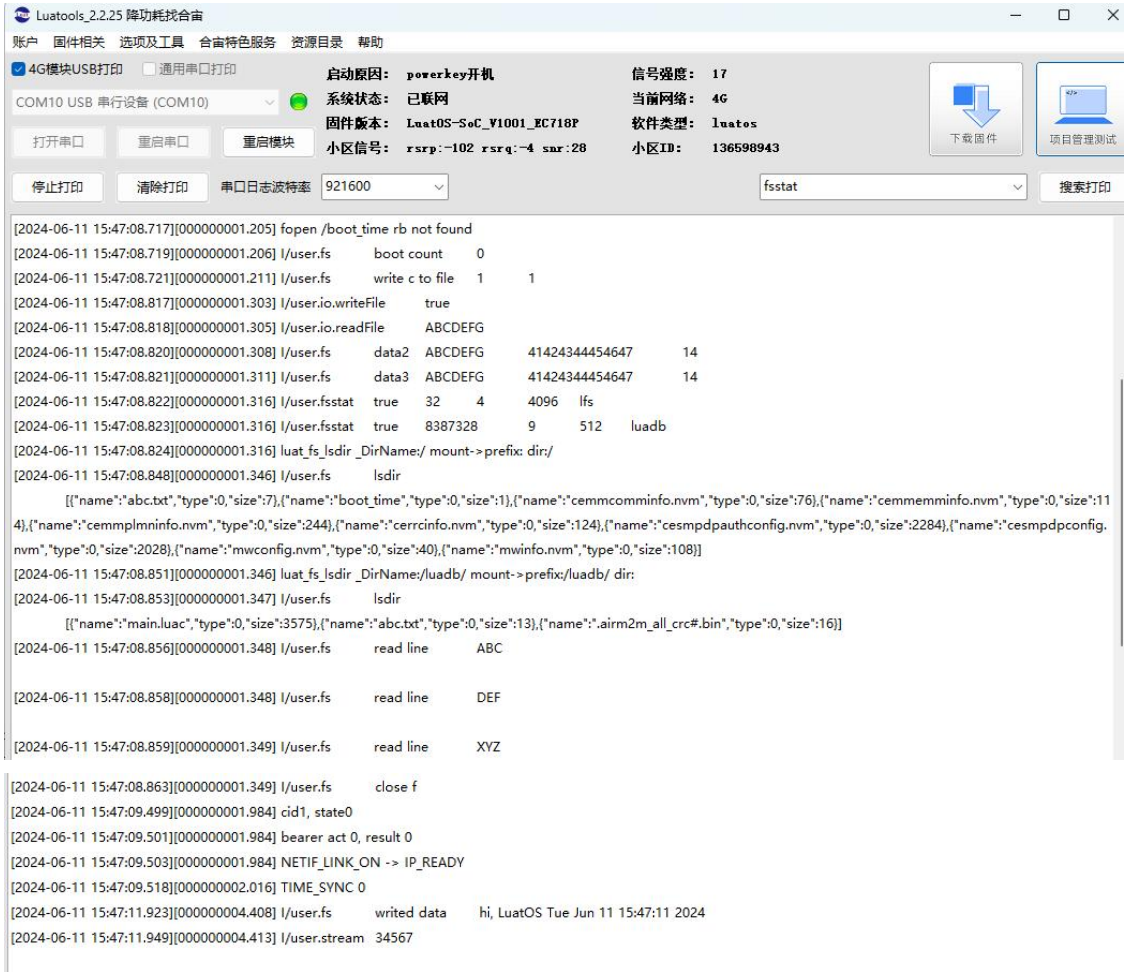
三．演示步骤

本文以 demo/fs 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块打印 log



四．代码解析

完整脚本代码点击右侧查看 fs/main.lua

io.open：打开文件

第 1 个参数，文件路径

第 2 个参数，打开模式, "rb"只读模式, 打开文件，"wb"读写默认,打开文件，"wb+"写入文件, 且截断为 0 字节，"a"追加模式

模式字符串可以是下列任意值：

“r”: 读模式（默认）；

“w”: 写模式；

“a”: 追加模式；

“r+”: 更新模式，所有之前的数据都保留；

“w+”: 更新模式，所有之前的数据都删除；

“a+”: 追加更新模式，所有之前的数据都保留，只允许在文件尾部做写入。

模式字符串可以在最后加一个 ‘b’ ， 这会在某些系统上以二进制方式打开文件。

返回值：成功文件句柄，失败 nil

io.writeFile：将数据写入文件

第 1 个参数，文件路径

第 2 个参数，数据

返回值：成功返回 true，否则返回 false

io.readFile：读取整个文件,请注意内存消耗

第 1 个参数，文件路径

第 2 个参数，读取模式，默认 “rb”

第 3 个参数，起始位置,默认 0

第 4 个参数，读取长度,默认整个文件

返回值：文件数据,若文件不存在会返回 nil

```
-- 根目录/是可写
local f = io.open("/boot_time", "rb")
local c = 0
if f then
    local data = f:read("*a")
    log.info("fs", "data", data, data:toHex())
    c = tonumber(data)
    f:close()
end
log.info("fs", "boot count", c)
c = c + 1
f = io.open("/boot_time", "wb")
--if f ~= nil then
log.info("fs", "write c to file", c, tostring(c))
f:write(tostring(c))
f:close()
--end

log.info("io.writeFile", io.writeFile("/abc.txt", "ABCDEFG"))

log.info("io.readFile", io.readFile("/abc.txt"))
local f = io.open("/abc.txt", "rb")
local c = 0
if f then
    local data = f:read("*a")
    log.info("fs", "data2", data, data:toHex())
    f:close()
end
```

```
-- 根目录/是可写
local f = io.open("/boot_time", "rb")
local c = 0
if f then
    local data = f:read("*a")
    log.info("fs", "data", data, data:toHex())
    c = tonumber(data)
    f:close()
end
log.info("fs", "boot count", c)
c = c + 1
f = io.open("/boot_time", "wb")
--if f ~= nil then
log.info("fs", "write c to file", c, tostring(c))
f:write(tostring(c))
f:close()
```

f:seek：设置及获取基于文件开头处计算出的位置。

第 1 个参数：指定的基点（默认值是 "cur"）。基点可以是：

“set”: 基点为 0 （文件开头）；

“cur”: 基点为当前位置；

“end”: 基点为文件尾；

第 2 个参数：偏移值（默认为 0）

返回值：当 seek 成功时，返回最终从文件开头计算起的文件的位置。 当 seek 失败时，返回 nil

调用 f:seek() 可以返回文件当前位置，并不改变它； 调用 f:seek("set") 将位置设为文件开头（并返回 0）； 调用 f:seek("end") 将位置设到文件末尾，并返回文件大小。

```
-- seek和tell测试
local f = io.open("/abc.txt", "rb")
local c = 0
if f then
    f:seek("end", 0)
    f:seek("set", 0)
    local data = f:read("*a")
    log.info("fs", "data3", data, data:toHex())
    f:close()
end
```

fs.fsstat：获取文件系统信息

参数：路径,默认”/”,可选

返回值：

第 1 个返回值：获取成功返回 true,否则返回 false

第 2 个返回值：总的 block 数量

第 3 个返回值：已使用的 block 数量

第 4 个返回值：block 的大小,单位字节

第 5 个返回值：文件系统类型,例如 lfs 代表 littlefs

```
if fs then
    -- 根目录是可读写的
    log.info("fsstat", fs.fsstat("/"))
    -- /luadb/ 是只读的
    log.info("fsstat", fs.fsstat("/luadb/"))
end
```

io.lsdir：列出目录下的文件

参数：需要枚举的目录路径

返回值：成功与否

```
local ret, files = io.lsdir("/")
log.info("fs", "lsdir", json.encode(files))

ret, files = io.lsdir("/luadb/")
log.info("fs", "lsdir", json.encode(files))
```

f:read("l")：读取刷机时加入的文件，并演示按行读取，刷机时选取的非 lua 文件，均存放在/luadb/目录下，单层无子文件夹

```
-- 读取刷机时加入的文件，并演示按行读取
-- 刷机时选取的非lua文件，均存放在/luadb/目录下，单层无子文件夹
f = io.open("/luadb/abc.txt", "rb")
if f then
    while true do
        local line = f:read("l")
        if not line or #line == 0 then
            break
        end
        log.info("fs", "read line", line)
    end
    f:close()
    log.info("fs", "close f")
else
    log.info("fs", "pls add abc.txt!!")
end
```

io.mkdir：创建文件夹

参数：需要建立的目录路径

第 1 个返回值：成功与否

第 2 个返回值：底层返回值

```
-- 文件夹操作
sys.wait(3000)
io.mkdir("/iot/")
f = io.open("/iot/1.txt", "w+")
if f then
    f:write("hi, LuatOS " .. os.date())
    f:close()
else
    log.info("fs", "open file for write failed")
end
f = io.open("/iot/1.txt", "r")
if f then
    local data = f:read("*a")
    f:close()
    log.info("fs", "writed data", data)
else
    log.info("fs", "open file for read failed")
end

-- 2023.6.6 新增 io.readFile支持配置起始位置和长度
io.writeFile("/test.txt", "0123456789")
log.info("stream", io.readFile("/test.txt", "rb", 3, 5))
```

五．常见问题

1. 不同的 core 固件分配的文件系统空间不尽相同，可通过 fs.fsstat(path)获取文件系统信息，单位为字节。

2. 为什么读文件失败

- 如果读取的是本地烧录的文件，检查下文件有没有烧录以及完整路径是否正确
- 如果读取的是程序运行过程中动态创建的文件，检查下文件是否创建成功【文件系统空间不足可能会导致文件创建失败】以及完整路径是否正确

3. luatools 烧录本地的文件在那个目录？

   luatools 烧录本地的文件在/luadb/这个目录下

   例子:在 luatools 烧录 1.txt 文件后读取文件路径如下

   `io.readFile("/luadb/1.txt")`

4. luatools 烧录本地的文件可以读写吗?

5. 可以读，但是不可写

10.5.19 miniz 开发应用指南

一. 概述

miniz 能解压标准 zlib 数据流，快速解压,需要 32kb 的 LuaVM 内存。

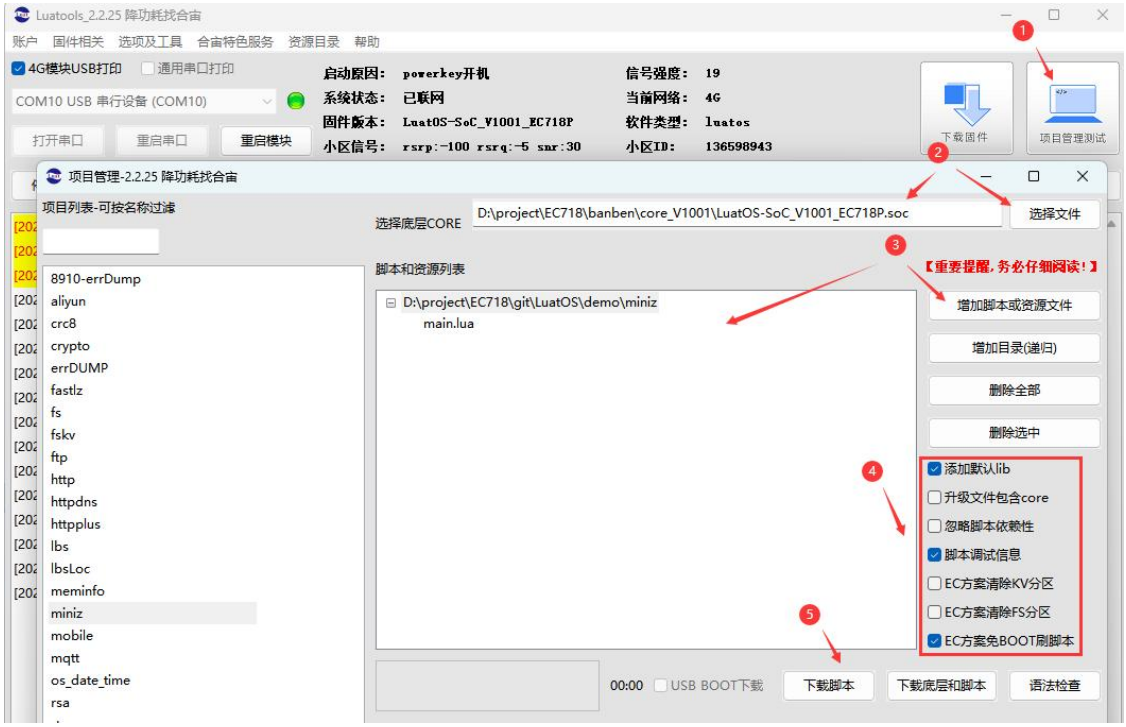
二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

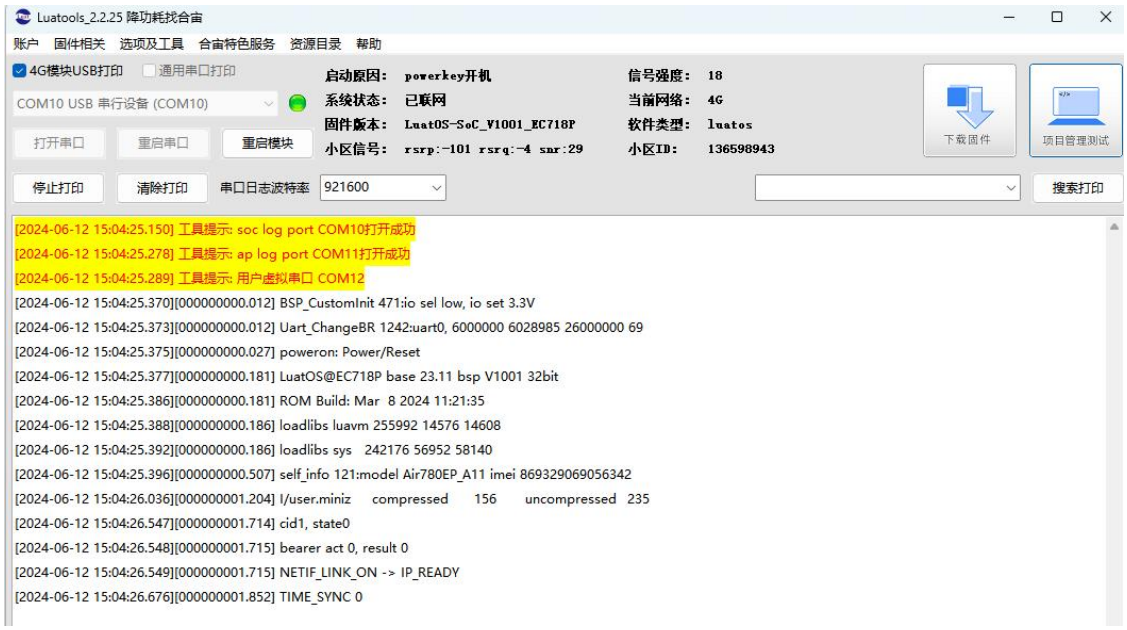
三. 演示步骤

本文以 demo/miniz 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块对应 log



四．代码解析

完整脚本代码点击右侧查看 demo/miniz/main.lua

miniz.uncompress：快速解压,需要 32kb 的 LuaVM 内存

第 1 个参数：待解压的数据，解压后的数据不能大于 32k

第 2 个参数：解压参数,默认是 miniz.PARSE_ZLIB_HEADER，即解析 zlib 头部

返回值：若解压成功,返回数据字符串，否则返回 nil

```lua
sys.taskInit(function()
    sys.wait(1000)
    -- 压缩过的字符串，为了方便演示，这里用了base64编码
    -- 大部分MCU设备的内存都比较小，miniz.compress 通常在服务器端完成，这里就不演示了
    -- miniz能解压标准zlib数据流
    local b64str = "eAEFQIGNwyAMXOUm+E2+OzjhCCiOjYyhyvbVR7K7IR0l+iau8G82eIW5jXVoPzF5ps
    local str = b64str:fromBase64()

    local dstr = miniz.uncompress(str)
    -- 压缩过的数据长度 156
    -- 解压后的数据长度,即原始数据的长度 235
    log.info("miniz", "compressed", #str, "uncompressed", #dstr)
end)
```

10.5.20 errDump 开发应用指南

一. 概述

LuatOS 错误日志上报功能模块名叫：errDump，errDump 对"量产投放市场的设备，远程调试初步定位问题"至关重要， 强烈建议客户一定要使用此功能。

二. 材料准备

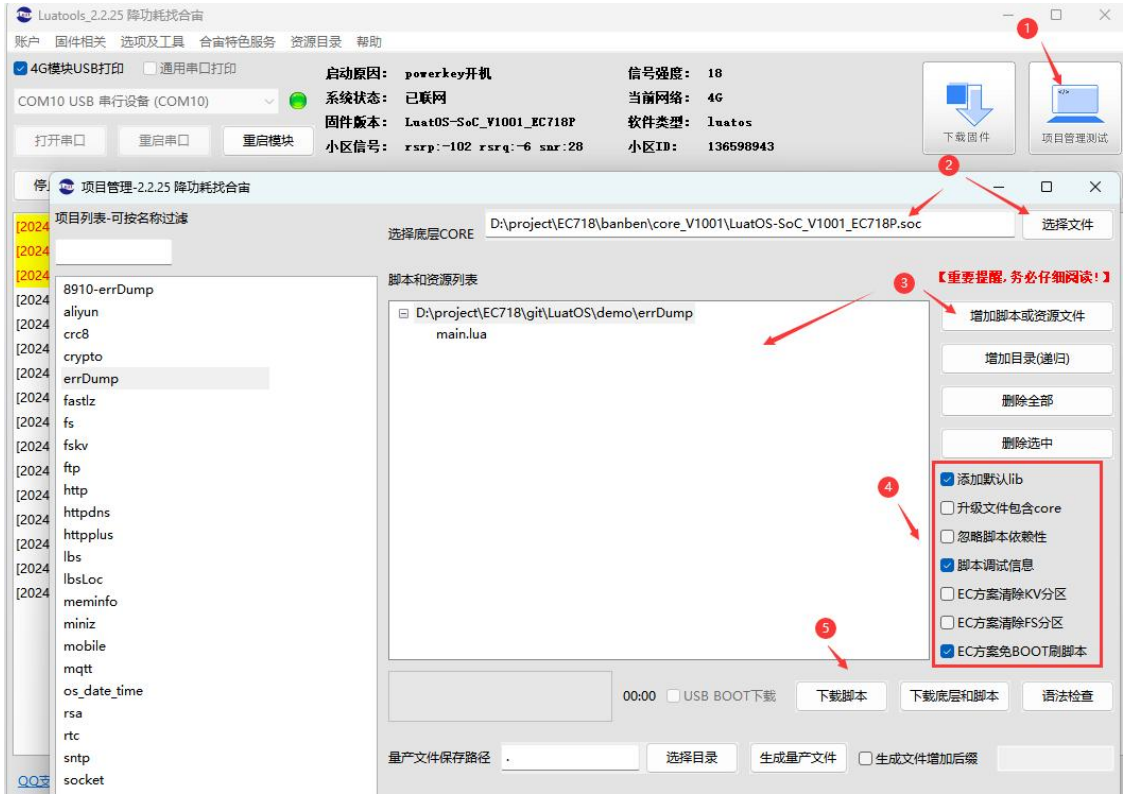
1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

三．演示步骤

本文以 demo/errDump 这个 DEMO 为例做演示

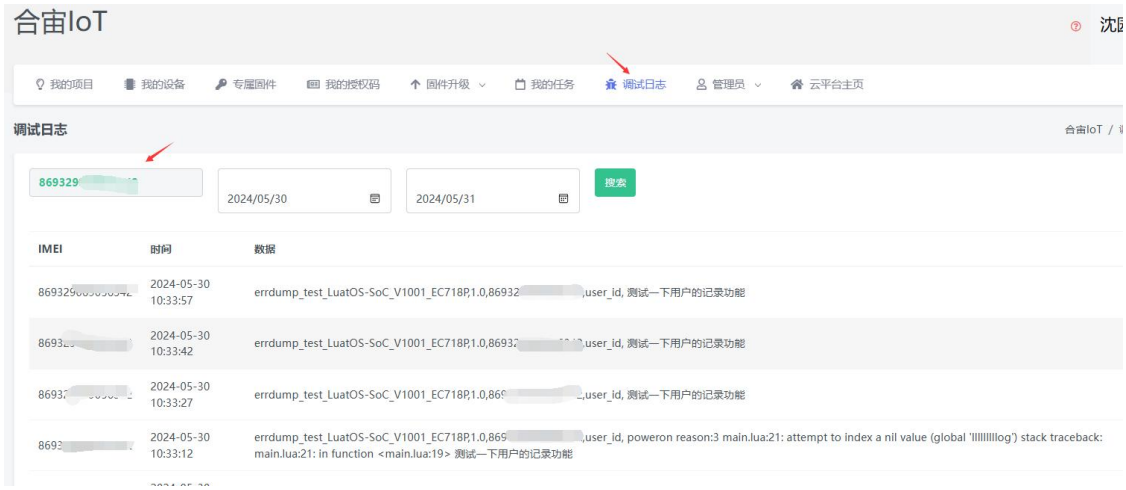
下面是用 DEMO 中的自动发送作为演示,首先 PRODUCT_KEY 把改成自己的，打开演示自动发送部分的脚本，注释掉演示手动获取信息部分脚本。

然后下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块打印日志

云平台对应日志



四．代码解析

完整脚本代码点击右侧查看 demo/errDump/main.lua

errDump.config：配置关键日志上传 IOT 平台，这里的日志包括引起 luavm 异常退出的日志和用户通过 record 写入的日志，类似于 air 的 errDump

第 1 个参数：是否启用记录功能，false 的话将不会记录任何日志

第 2 个参数：定时上传周期，单位秒，默认 600 秒，这个是自动上传时候后的重试时间时间，在开机后或者有 record 操作后会很快尝试上传到合宙 IOT 平台一次，如果为 0，则不会上传，由用户 dump 后自己上传自己的平台

第 3 个参数：用户的特殊标识，可以为空

第 4 个参数：设备识别号, 4G 设备默认是 imei

第 5 个参数：服务器域名,默认 dev_msg1.openluat.com

第 6 个参数：服务器端口,默认

返回值：nil 无返回值

errDump.record：写入用户的异常日志，注意最大只有 4KB，超过部分新的覆盖旧的，开启自动上传后会上传到合宙 IOT 平台

参数：日志

返回值：nil 无返回值

```
PROJECT = "errdump_test"
VERSION = "1.0"
PRODUCT_KEY = "y                    M1" --换成自己的
-- sys库是标配
_G.sys = require("sys")
_G.sysplus = require("sysplus")
log.style(1)

--下面演示自动发送
errDump.config(true, 600, "user_id")    -- 默认是关闭，用这个可以额外添加用户标识，比如用户自定义的ID之类

local function test_user_log()
    while true do
        sys.wait(15000)
        errDump.record("测试一下用户的记录功能")
    end
end

local function test_error_log()
    sys.wait(60000)
    lllllllllog.record("测试一下用户的记录功能") --默认写错代码死机
end
```

10.5.21 zbuff 开发应用指南

一. 概述

zbuff 可以理解为 char[], char*, uint8_t*，为了与 lua 更好的融合, zbuff 带长度,带指针位置,可动态扩容。可当成数组直接赋值和取值。

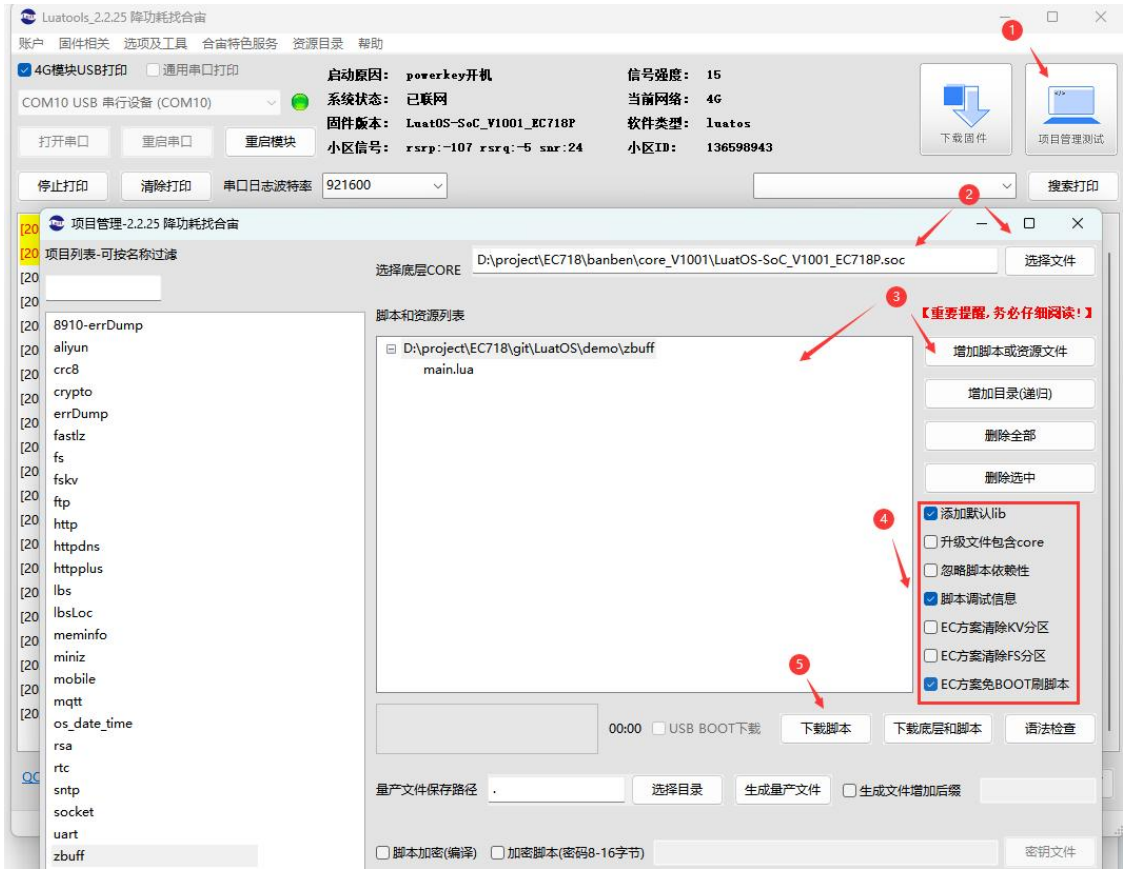
二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

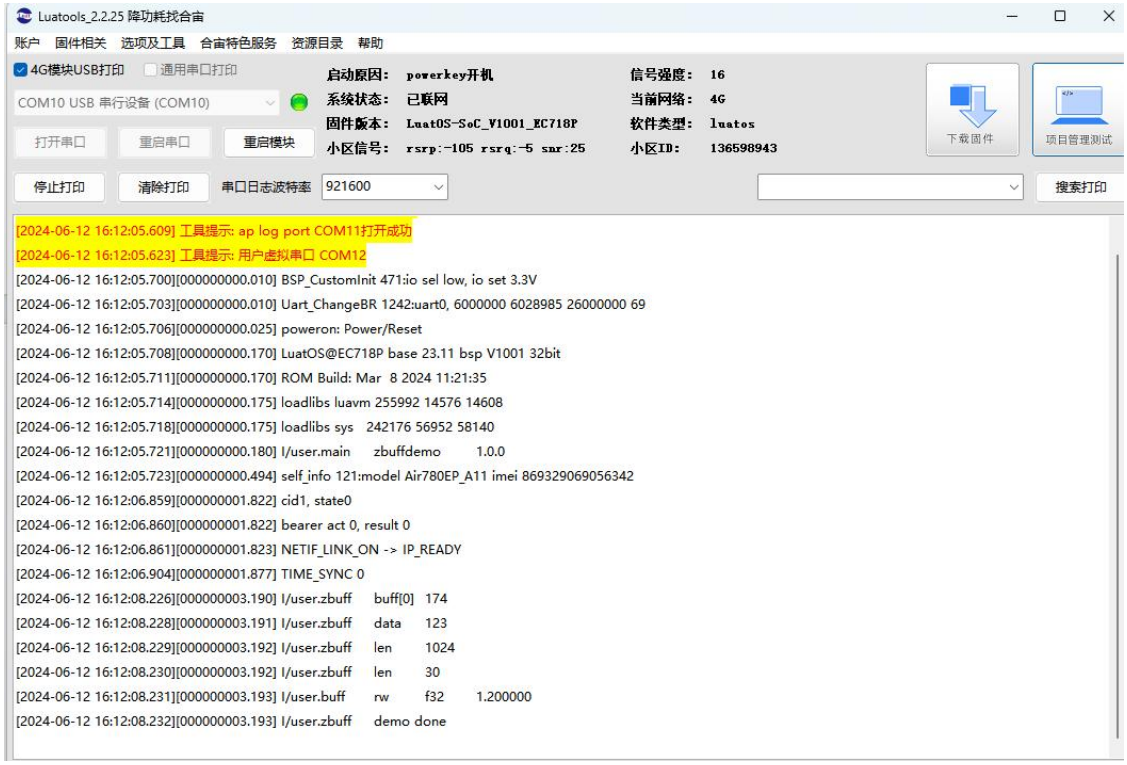
三．演示步骤

本文以 demo/zbuff 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块对应 log



四．代码解析

完整脚本代码点击右侧查看 demo/zbuff/main.lua

zbuff.create：创建 zbuff

第 1 个参数：字节数

第 2 个参数：可选参数，number 时为填充数据，string 时为填充字符串

第 3 个参数：可选参数，内存类型，可选：zbuff.HEAP_SRAM(内部 sram,默认) zbuff.HEAP_PSRAM(外部 psram) zbuff.HEAP_AUTO(自动申请,如存在 psram 则在 psram 进行申请,如不存在或失败则在 sram 进行申请) 注意:此项与硬件支持有关

返回值：zbuff 对象，如果创建失败会返回 nil

buff:write：zbuff 写数据（从当前指针位置开始；执行后指针会向后移动）

参数：写入 buff 的数据，string 时为一个参数，number 时可为多个参数

返回值：数据成功写入的长度

buff:seek：zbuff 设置光标位置（可能与当前指针位置有关；执行后指针会被设置到指定位置）

第 1 个参数：偏移长度

第 2 个参数：where，基点，默认 zbuff.SEEK_SET。zbuff.SEEK_SET: 基点为 0 （文件开头），zbuff.SEEK_CUR: 基点为当前位置，zbuff.SEEK_END: 基点为文件尾

返回值：设置光标后从 buff 开头计算起的光标的位置

buff:read：zbuff 读数据（从当前指针位置开始；执行后指针会向后移动）

参数：读取 buff 中的字节数

```lua
-- zbuff可以理解为char[], char*, uint8_t*
-- 为了与lua更好的融合, zbuff带长度,带指针位置,可动态扩容
local buff = zbuff.create(1024)
-- 可当成数组直接赋值和取值
buff[0] = 0xAE
log.info("zbuff", "buff[0]", buff[0])

-- 以io形式操作

-- 写数据write, 操作之后指针会移动,跟文件句柄一样
buff:write("123") -- 字符串
buff:write(0x12, 0x13, 0x13, 0x33) -- 直接写数值也可以

-- 设置指针位置, seek
buff:seek(5, zbuff.SEEK_CUR) -- 指针位置+5
buff:seek(0)                 -- 绝对地址

-- 读数据read, 指针也会移动
local data = buff:read(3)
log.info("zbuff", "data", data)
```

buff:clear：zbuff 清空数据（与当前指针位置无关；执行后指针位置不变）

参数：可选，默认为 0。要设置为的值，不会改变 buff 指针位置

buff:pack：将一系列数据按照格式字符转化，并写入（从当前指针位置开始；执行后指针会向后移动）

第 1 个参数：后面数据的格式（符号含义见下面的例子）

第 2 个参数：传入的数据，可以为多个数据

返回值：成功写入的数据长度

例子

```
buff:pack(">IIHA", 0x1234, 0x4567, 0x12,"abcdefg") -- 按格式写入几个数据
-- A string
-- f float
-- d double
-- n Lua number
-- c char
-- b byte / unsignen char
-- h short
-- H unsigned short
-- i int
-- I unsigned int
-- l long
-- L unsigned long
-- < 小端
-- > 大端
-- = 默认大小端
```

buff:unpack：将一系列数据按照格式字符读取出来（从当前指针位置开始；执行后指针会向后移动）

参数：数据的格式（符号含义见上面 pack 接口的例子）

返回值 1：成功读取的数据字节长度

返回值 2：按格式读出来的数据

```
-- 支持以pack库的形式写入或读取数据
buff:seek(0)
buff:pack(">IIHA", 0x1234, 0x4567, 0x12,"abcdefg")
buff:seek(0)
local cnt,a,b,c,s = buff:unpack(">IIHA10")
```

直接按类型读写数据

```
-- 也可以直接按类型读写数据
local len = buff:writeI8(10)
local len = buff:writeU32(1024)
local i8data = buff:readI8()
local u32data = buff:readU32()
```

buff:toStr：按起始位置和长度取出数据（与当前指针位置无关；执行后指针位置不变）

第 1 个参数：数据的起始位置（起始位置为 0）,默认值也是 0

第 2 个参数：数据的长度,默认是全部数据

返回值：读出来的数据

例子

```
local s = buff:toStr(0,5)--读取开头的五个字节数据
local s = buff:toStr() -- 取出整个zbuff的数据
local s = buff:toStr(0, buff:used()) -- 取出已使用的部分，与buff:query()一样
```

buff:len()：获取 zbuff 对象的长度（与当前指针位置无关；执行后指针位置不变）

返回值：zbuff 对象的长度

buff:used()：获取 zbuff 里最后一个数据位置指针到首地址的偏移量，来表示 zbuff 内已有有效数据量大小，注意这个不同于分配的空间大小，由于 seek()会改变最后一个数据位置指针，因此也会影响到 used()返回值。

返回值：有效数据量大小

```
-- 取出指定区间的数据
local fz = buff:toStr(0,5)

-- 获取其长度
log.info("zbuff", "len", buff:len())
-- 获取其指针位置
log.info("zbuff", "len", buff:used())

-- 测试writeF32, 注意, EC618系列(Air780E等), V1106会死机, 在V1107修复
buff:seek(0, zbuff.SEEK_SET)
buff:writeF32(1.2)
buff:seek(0, zbuff.SEEK_SET)
log.info("buff", "xx", "f32", buff:readF32())
```

10.5.22 hmeta 开发应用指南

一．概述

hmeta 识别底层模组类型的，不同的模组可以使用相同的 bsp,但根据封装的不同,根据内部数据仍可识别出具体模块。

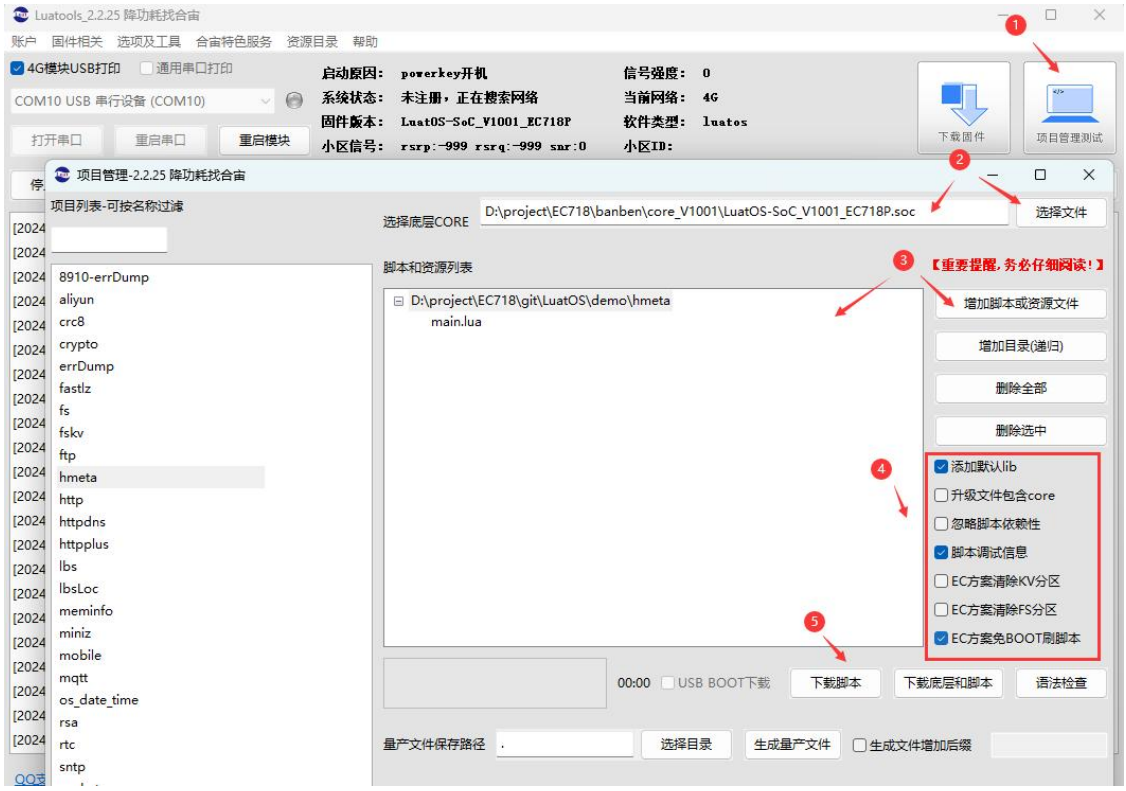
二．材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

三．演示步骤

本文以 demo/hmeta 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块对应 log

四．代码解析

完整脚本代码点击右侧查看 demo/hmeta/main.lua

hmeta.model()：获取模组名称，若能识别到,返回模组类型，否则会是 nil

hmeta.hwver()：获取模组的硬件版本号，若能识别到,返回模组类型，否则会是 nil

rtos.bsp()：获取芯片平台

```lua
sys.taskInit(function()
    while hmeta do
        -- hmeta识别底层模组类型的
        -- 不同的模组可以使用相同的bsp,但根据封装的不同,根据内部数据仍可识别出具体模块
        log.info("hmeta", hmeta.model(), hmeta.hwver and hmeta.hwver())
        log.info("bsp",   rtos.bsp())
        sys.wait(3000)
    end
    log.info("这个bsp不支持hmeta库哦")
end)
```

10.5.23 json 开发应用指南

一．概述

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格局。它基于 ECMAScript 的一个子集。 JSON 选用完全独立于言语的文本格局，但是也使用了类似于 C 言语宗族的习气（包含 C、C++、C#、Java、JavaScript、Perl、Python 等）。这些特性使 json 调试成为抱负的数据交换言语。 易于人阅览和编写，同时也易于机器解析和生成(一般用于提高网络传输速率)。

1.JSON 的格式

JSON 数据的书写格式是键（名称）/值对。

JSON 键值对是用来保存 JS 对象的一种方式，和 JS 对象的写法也大同小异，键/值对包括字段名称（在双引号中），后面写一个冒号，然后是值。

JSON 值可以是：字符串（在双引号中）、数组（在中括号中）、数字（整数或浮点数）、逻辑值（true 或 false）、对象（在大括号中）、 null。

2.JSON 结构

JSON 结构有两种结构，就是对象和数组。通过这两种结构可以表示各种复杂的结构。

{“province”: “Shanxi”} 可以理解为是一个包含 province 为 Shanxi 的对象，

[“Shanxi”,“Shandong”]这是一个包含两个元素的数组

而 [{“province”: “Shanxi”},{“province”: “Shandong”}] 就表示包含两个对象的数组。

当然了,你也可以使用 {“province”:[“Shanxi”,“Shandong”]} 来简化上面的 JSON,这是一个拥有一个 name 数组的对象。

下面是一小段 JSON 代码：

{“skillz”: {“web”:[{“name”: “html”， “years”: “5” }， {“name”: “css”， “years”: “3” }],”database”:[{“name”: “sql”， “years”: “7” }]}}

花括弧，方括弧，冒号和逗号：花括弧表示一个“容器”，方括号装载数组，名称和值用冒号隔开，数组元素通过逗号隔开。

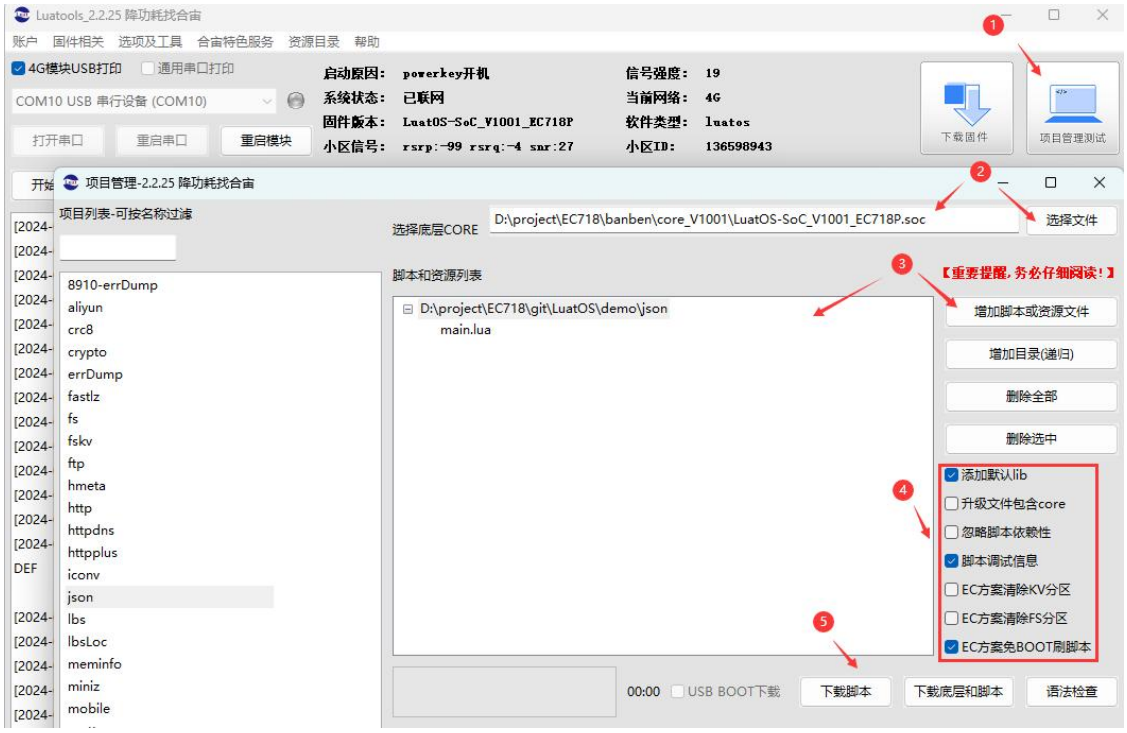
二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

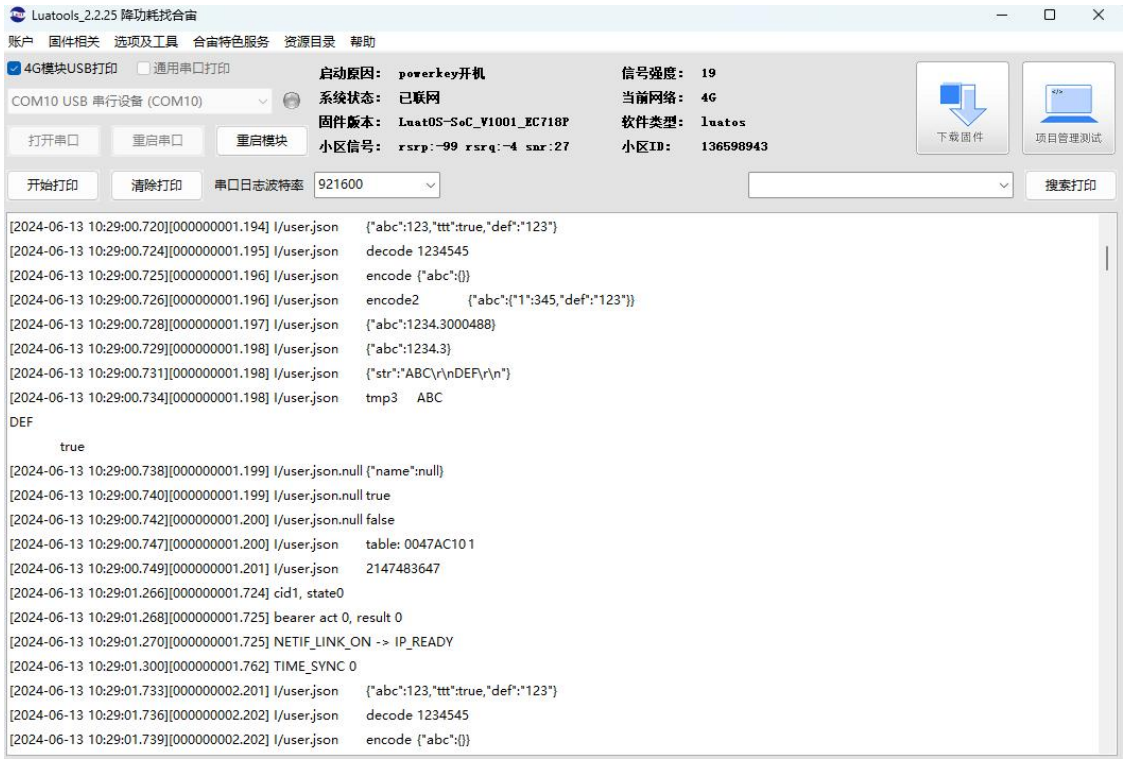
三. 演示步骤

本文以 demo/json 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块对应 log



四．代码解析

完整脚本代码点击右侧查看 demo/json/main.lua

json.encode：将对象序列化为 json 字符串

第 1 个参数：需要序列化的对象

第 2 个参数：浮点数精度和模式,这项不存在的时候,为默认值”7f”,数字只支持”0~14”,模式只支持”f/g

返回值 1：序列化后的 json 字符串，失败的话返回 nil

返回值 2：序列化失败的报错信息

json.decode：将字符串反序列化为对象

参数：需要反序列化的 json 字符串

返回值 1：反序列化后的对象(通常是 table)，失败的话返回 nil

返回值 2：成功返回 1,否则返回 0

返回值 3：反序列化失败的报错信息

```
-- table 转为 字符串
local t = {abc=123, def="123", ttt=true}
local jdata = json.encode(t)
log.info("json", jdata)

-- 字符串转table
local str = "{\"abc\":1234545}" -- 字符串可以来源于任何地方,网络,文本,用户输入,都可以
local t = json.decode(str)
if t then -- 若解码失败, 会返回nil
    log.info("json", "decode", t.abc)
end

-- lua中的table是 数组和hashmap的混合体
-- 这对json来说会有一些困扰, 尤其是空的table
local t = {abc={}}
-- 假设从业务上需要输出 {"abc":[]}
-- 实际会输出 {"abc": {}} , 空table是优先输出 hashmap 形式, 而非数组形式
log.info("json", "encode", json.encode(t))
-- 混合场景, json场景应避免使用
t.abc.def = "123"
t.abc[1] = 345
-- 输出的内容是 {"abc":{"1":345,"def":"123"}}
log.info("json", "encode2", json.encode(t))

-- 浮点数演示
-- 默认%.7g
log.info("json", json.encode({abc=1234.300}))
-- 限制小数点到1位
log.info("json", json.encode({abc=1234.300}, "1f"))

-- 2023.6.8 处理\n在encode之后变成\b的问题
local tmp = "ABC\r\nDEF\r\n"
local tmp2 = json.encode({str=tmp})
log.info("json", tmp2)
local tmp3 = json.decode(tmp2)
log.info("json", "tmp3", tmp3.str, tmp3.str == tmp)
-- break

log.info("json.null", json.encode({name=json.null}))
log.info("json.null", json.decode("{\"abc\":null}").abc == json.null)
log.info("json.null", json.decode("{\"abc\":null}").abc == nil)
```

五．常见问题

1.自己添加的 JSON 字符串解析失败

搜索 JSON 解析网页，先把这段 JSON 字符串放进网页里看 JSON 格式是否正确。如果正确，

在看看自己代码里的转义字符有没有加对加全。

2.json.encode（）这个 api 生成的 json 顺序是随机的吗？

是的

10.5.24 gpio 开发应用指南

一．概述

GPIO(General-Purpose Input/Output) 指通用输入输出口。在软件运行期间能够动态配置和控制的通用引脚，实现某时刻电平状态的输出与输入，即可以通过它们输出高低电平或者通过它们读入引脚的状态-是高电平或是低电平。除了通用输入输出功能外，还可能有上拉、下拉、中断或其它复用功能。LuatOS 固件下的 IO 复用配置

GPIO 额外说明

1. AONGPIO 是休眠时仍可维持高电平的 GPIO,但驱动能力很弱

2. 普通 GPIO 在配置成输入/中断模式时，上下拉无法设置，如果默认上下拉不能满足要求，可以设置成 0 来取消默认上下拉，然后外部加上下拉

虚拟 GPIO

Air780EP(EC718P/EC718PV 全系)支持多个虚拟的 GPIO，将非 GPIO 管脚通过软件模拟成 GPIO 来使用

编号	名称	功能	备注
39	wakeup0	仅支持输入和中断	wakeup0 休眠唤醒脚
40	wakeup1/vbus	仅支持输入和中断	USB 的 VBUS，检测 USB 是否是插入状态
41	wakeup2	仅支持输入和中断	wakeup2 休眠唤醒脚, USIM_DET
46	pwrkey	仅支持输入和中断	即开机键，开机之后当普通 GPIO 使用

说明:

1. vbus 与 USB 功能是解耦的

2. 与常规认识不同，在不接 vbus 的情况下, USB 功能依然可用

3. 在进入休眠前，将上述 wakeup0/wakeup1/wakeup2 设置成中断状态，即可实现管脚唤醒功能

4. 非 wakeup 的普通 GPIO，是不支持休眠唤醒的

5. wakeup3/wakeup4/wakeup5 在模块未引出

例如将 wakup0 设置为唤醒脚，中断回调可以是空函数

gpio.setup(39, function() end, gpio.PULLUP)

二. 材料准备

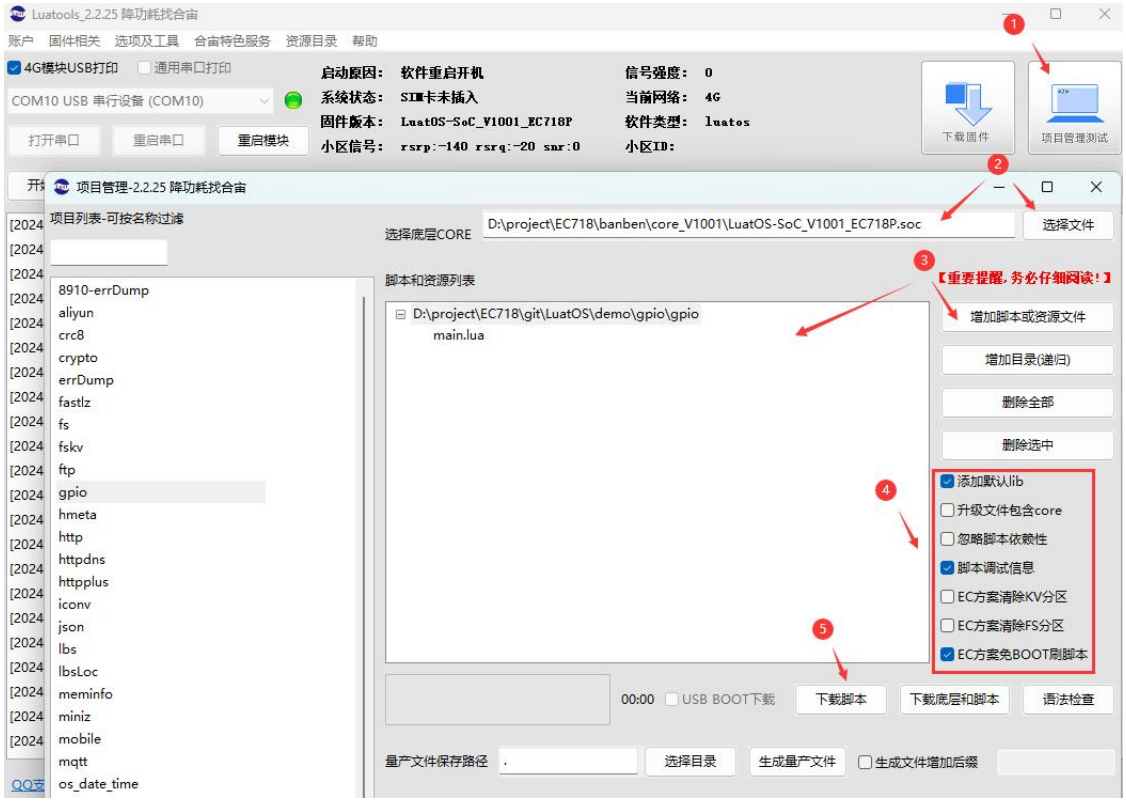
1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

三. 演示步骤

本文以 demo/gpio 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本

下面是 demo\gpio\gpio 下载的演示，其它下载方式一样，选择对应的脚本就可以了。

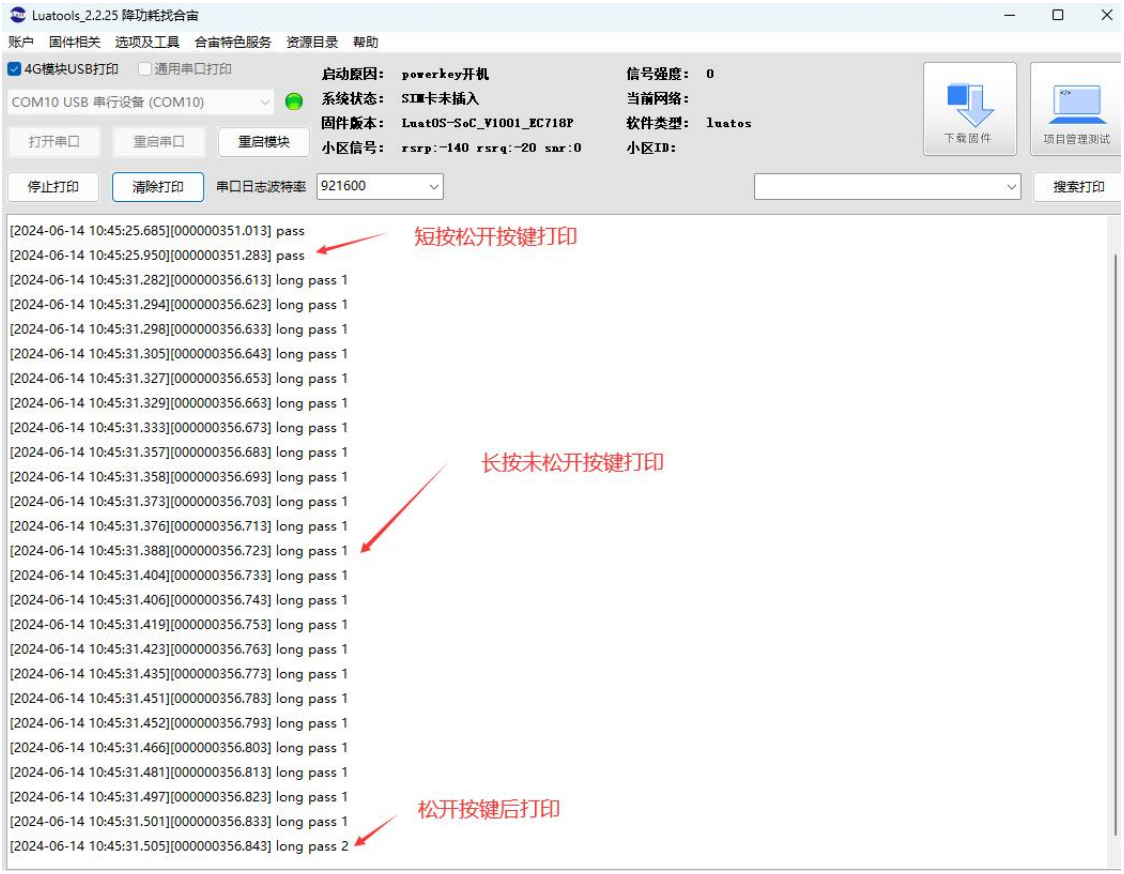


demo\gpio\gpio 下载到模块后，开发板对应的绿色 NET 灯会闪烁，并打印对应的 log，模块打印 log 如下所示:

demo\gpio\gpio_button 修改 BTN_PIN = 46 对应开发板上的开机按键，为了更好的显示演示效果，下面打印加了序号

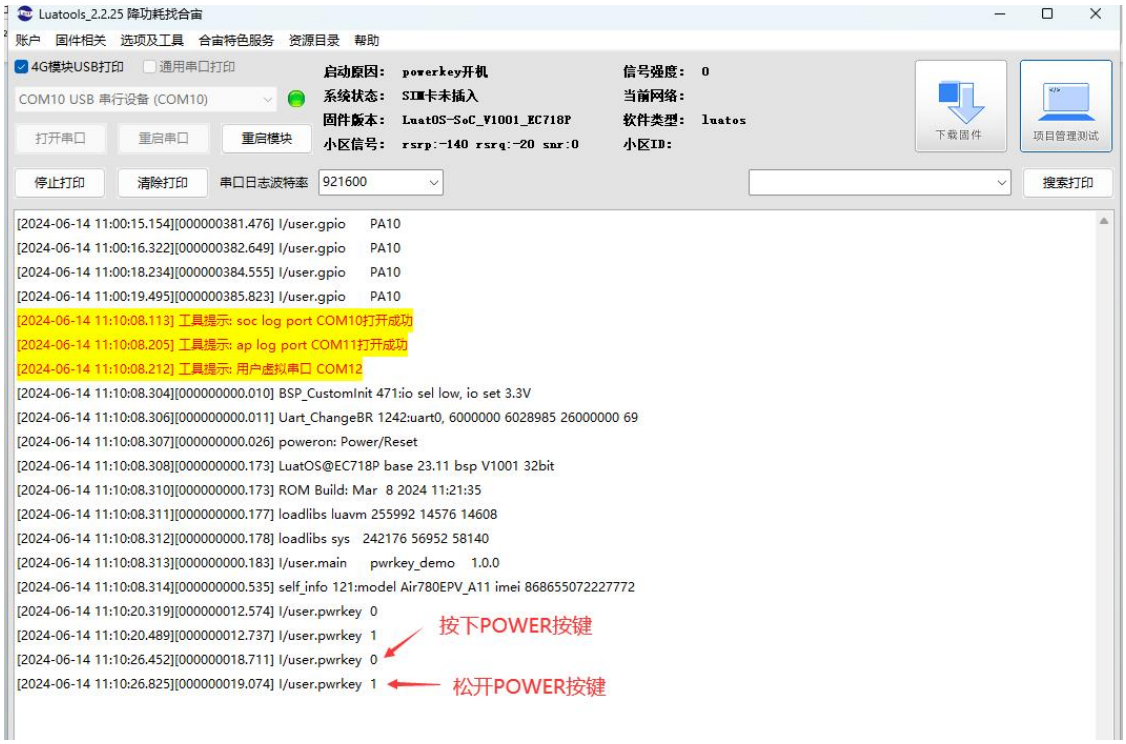
```
button_timer = sys.timerLoopStart(function()
    if button_state then
        if button() == 0 then
            button_cont = button_cont + 1
            if button_cont > button_long_time then
                print("long pass 1")
            end
        else
            if button_cont < button_shake_time then
            else
                if button_cont < button_long_time then
                    print("pass")
                else
                    print("long pass 2")
                end
            end
            button_cont = 0
            button_state = false
            button_detect = true
        end
    end
end,button_timer_outtime)
```

修改后的代码下载到开发板后，按开机键对应 log 如下：



demo\gpio\gpio_irq 修改 gpio_pin=35 对应开发板 LCD_CS 管脚，下载到模块后，用 LCD_CS 和 GND 短接和断开后打印 log 如下所示：

demo\gpio\powerkey 直接下载到模块后，操作 POWER 按键打印 log：



四．代码解析

点击右侧查看 GPIO API 说明：GPIO API

demo\gpio\gpio 完整脚本代码点击右侧查看 demo/gpio/gpio/main.lua

Air780UM 开发板上就一个灯对应的 GPIO 为 GPIO27

```
    elseif rtos_bsp == "EC718P" then -- Air780E开发板引脚
        return 27, 255, 255 -- AIR780EP开发板上就一个灯
```

赋值开发板 LED 引进编号

```
--LED引脚判断赋值结束

local P1,P2,P3=pinx()--赋值开发板LED引脚编号
local LEDA= gpio.setup(P1, 0, gpio.PULLUP)
```

单独用一个 task 来做流水灯控制，Air780EP 开发板就 1 个灯，所以只会看到 1 个灯闪烁的效果

```lua
sys.taskInit(function()
--开始流水灯
    local count = 0
    while 1 do
    --流水灯程序
        sys.wait(500) --点亮时间
        -- 轮流点灯
        LEDA(count % 3 == 0 and 1 or 0)
        if P2 and P2 ~=255 then
            LEDB(count % 3 == 1 and 1 or 0)
        end
        if P3 and P3 ~= 255 then
            LEDC(count % 3 == 2 and 1 or 0)
        end
        log.info("GPIO", "Go Go Go", count, rtos.bsp())
        log.info("LuatOS:", "https://wiki.luatos.com")
        count = count + 1
    end
end)
```

demo\gpio\gpio_button 完整脚本代码点击右侧查看 demo/gpio/gpio_button/main.lua

gpio.debounce(BTN_PIN, 5)：设置按键防抖时间为 5 毫秒

gpio.setup：设置管脚功能，这里配置为中断回调模块

sys.timerLoopStart：循环定器来没 10 毫秒查看一次按键状态，只按键状态，还是长按中，还是长按抬起状态，并打印对应的状态

```lua
local BTN_PIN = 46 -- 按实际开发板选取

-- 若固件支持防抖，启用防抖
if gpio.debounce then
    gpio.debounce(BTN_PIN, 5)
end

button = gpio.setup(BTN_PIN, function()
        if not button_detect then return end
        button_detect = false
        button_state = true
    end,
    gpio.PULLUP,
    gpio.FALLING)

button_timer = sys.timerLoopStart(function()
    if button_state then
        if button() == 0 then
            button_cont = button_cont + 1
            if button_cont > button_long_time then
                print("long pass 1")
            end
        else
            if button_cont < button_shake_time then
            else
                if button_cont < button_long_time then
                    print("pass")
                else
                    print("long pass 2")
                end
            end
            button_cont = 0
            button_state = false
            button_detect = true
        end
    end
end,button_timer_outtime)
```

配置为POWER按键

demo\gpio\gpio_irq 完整脚本代码点击右侧查看 demo/gpio/gpio_irq/main.lua

修改 gpio_pin=35 对应开发板 LCD_CS 管脚

gpio.debounce(gpio_pin, 100)：防抖设置为 100 毫秒

gpio.setup：配置中断回调函数

```lua
--配置gpio7为输入模式，下拉，并会触发中断
--请根据实际需求更改gpio编号和上下拉
local gpio_pin = 35
gpio.debounce(gpio_pin, 100)
gpio.setup(gpio_pin, function()
    log.info("gpio", "PA10")
end, gpio.PULLUP)
```

demo\gpio\powerkey 完整脚本代码点击右侧查看 demo/gpio/powerkey/main.lua

Air780EP 开机键对应的 GPIO 为 46

gpio.setup：配置对应回调函数

```lua
local function pinx()
    if rtos_bsp == "EC618" then -- AIR780E                    -- 35是虚拟GPIO，见https
        return 35
    elseif rtos_bsp == "EC718P" then -- AIR780EP              -- 46是虚拟GPIO
        return 46
    else
        return 255
    end
end


local powerkey_pin = pinx()                                  -- 赋值powerkey引脚编号

if powerkey_pin ~= 255 then
    gpio.setup(powerkey_pin, function()
        log.info("pwrkey", gpio.get(powerkey_pin))
    end, gpio.PULLUP)
else
    log.info("bsp not support")
end
```

五．常见问题

1. GPIO 开机电平变化，GPIO 复用关系，GPIO IO 特性

点击 Air780EP 文件中心查看硬件设计相关手册对应内容

2. LuatOS 固件下的 IO 复用配置，点击下面链接查看

LuatOS 固件下的 IO 复用配置

10.5.25 adc 开发应用指南

一．概述

可读取硬件 adc 通道，也支持读取 CPU 温度和 VBAT 供电电源。读取 CPU 温度，单位为 0.001 摄氏度，是内部温度，非环境温度。读取 VBAT 供电电压，单位为 mV。默认不开启分压，范围是 0-1.6v 精度高，开启分压后，外部输入最大不可超过 3.3V。

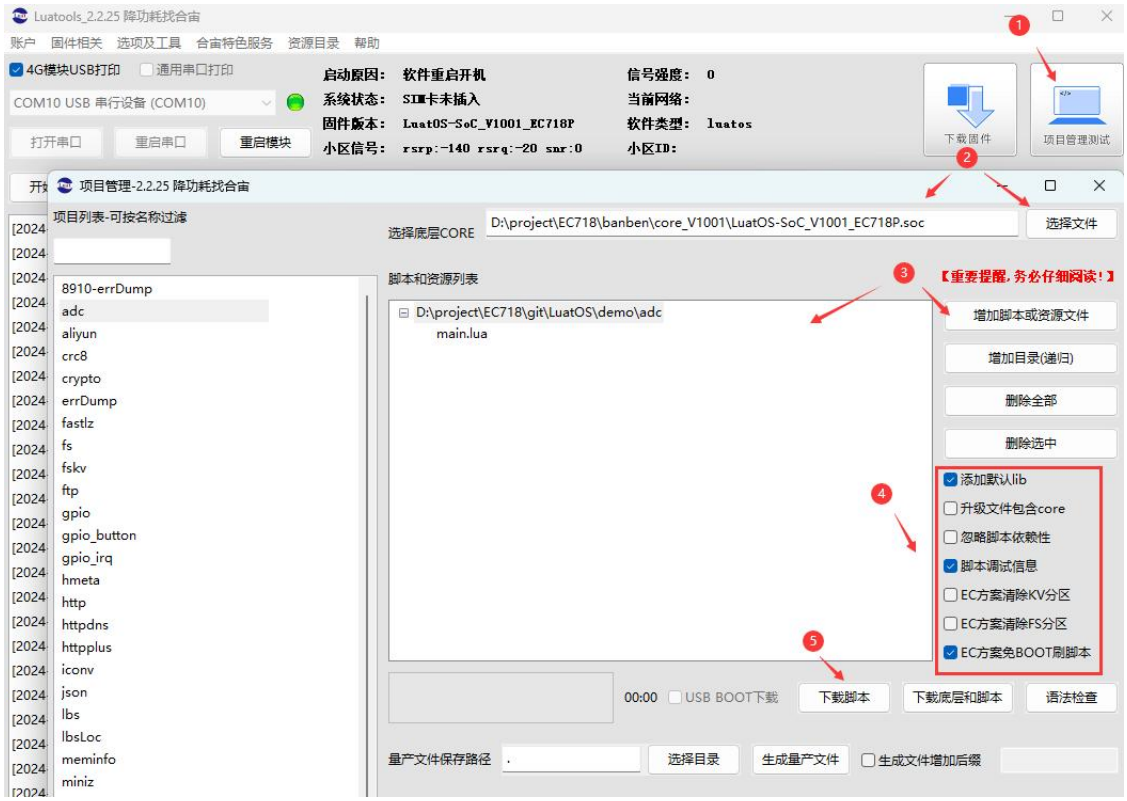
二．材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

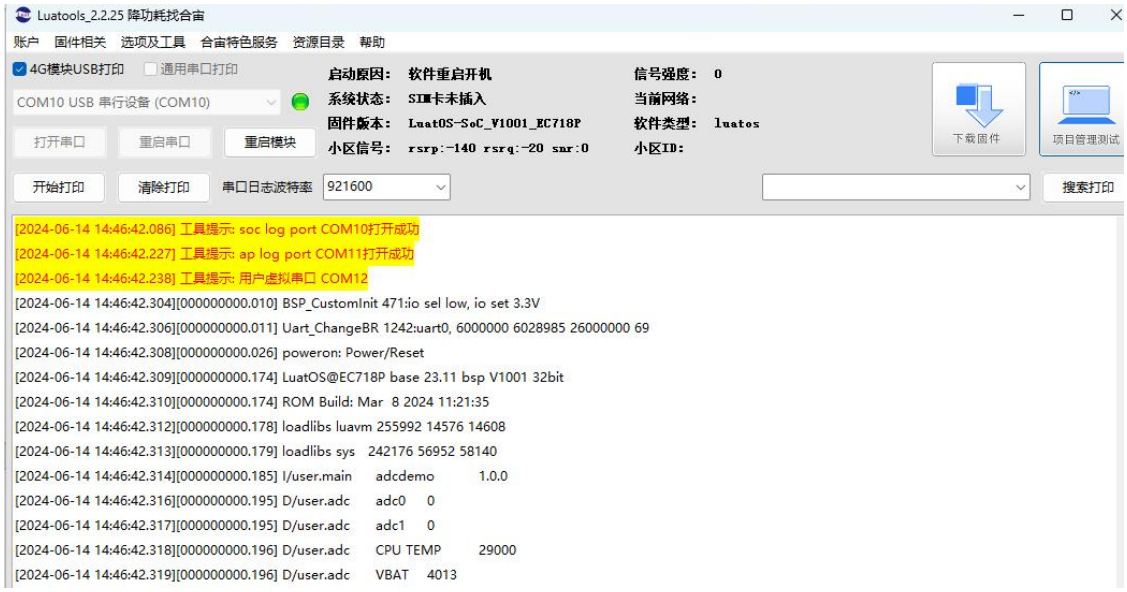
三．演示步骤

本文以 demo/adc 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块对应 log



四．代码解析

点击右侧查看 ADC API 说明：ADC API

完整脚本代码点击右侧查看 demo/adc/main.lua

配置对应通道 id

```
elseif rtos_bsp == "EC718P" then --Air780EP开发板ADC编号
    -- 默认不开启分压,范围是0-1.6v精度高
    -- 开启分压后, 外部输入最大不可超过3.3v
    -- adc.setRange(adc.ADC_RANGE_MAX)
    return 0,1,255,255,adc.CH_CPU ,adc.CH_VBAT
```

adc.open：打开 adc 通道

```
if adc_pin_0 and adc_pin_0 ~= 255 then adc.open(adc_pin_0) end
if adc_pin_1 and adc_pin_1 ~= 255 then adc.open(adc_pin_1) end
if adc_pin_2 and adc_pin_2 ~= 255 then adc.open(adc_pin_2) end
if adc_pin_3 and adc_pin_3 ~= 255 then adc.open(adc_pin_3) end
if adc_pin_temp and adc_pin_temp ~= 255 then adc.open(adc_pin_temp) end
if adc_pin_vbat and adc_pin_vbat ~= 255 then adc.open(adc_pin_vbat) end
```

adc.get：获取 adc 计算值

```
-- 下面是循环打印，接地不打印0也是正常现象
-- ADC的精度都不会太高，若需要高精度ADC，建议额外添加adc芯片
while true do
    if adc_pin_0 and adc_pin_0 ~= 255 then
        log.debug("adc", "adc" .. tostring(adc_pin_0), adc.get(adc_pin_0)) -- 若adc.get报nil，改成adc.read
    end
    if adc_pin_1 and adc_pin_1 ~= 255 then
        log.debug("adc", "adc" .. tostring(adc_pin_1), adc.get(adc_pin_1))
    end
    if adc_pin_2 and adc_pin_2 ~= 255 then
        log.debug("adc", "adc" .. tostring(adc_pin_2), adc.get(adc_pin_2))
    end
    if adc_pin_3 and adc_pin_3 ~= 255 then
        log.debug("adc", "adc" .. tostring(adc_pin_3), adc.get(adc_pin_3))
    end
    if adc_pin_temp and adc_pin_temp ~= 255 then
        log.debug("adc", "CPU TEMP", adc.get(adc_pin_temp))
    end
    if adc_pin_vbat and adc_pin_vbat ~= 255 then
        log.debug("adc", "VBAT", adc.get(adc_pin_vbat))
    end
    sys.wait(1000)
end
```

adc.close：关闭 adc 通道

```
-- 若不再读取，可关掉adc，降低功耗，非必须
if adc_pin_0 and adc_pin_0 ~= 255 then adc.close(adc_pin_0) end
if adc_pin_1 and adc_pin_1 ~= 255 then adc.close(adc_pin_1) end
if adc_pin_2 and adc_pin_2 ~= 255 then adc.close(adc_pin_2) end
if adc_pin_3 and adc_pin_3 ~= 255 then adc.close(adc_pin_3) end
if adc_pin_temp and adc_pin_temp ~= 255 then adc.close(adc_pin_temp) end
if adc_pin_vbat and adc_pin_vbat ~= 255 then adc.close(adc_pin_vbat) end
```

10.5.26 netled 开发应用指南

一. 概述

点击 netLed 网络状态指示灯对应 API 可以配置网络状态指示灯工作状态

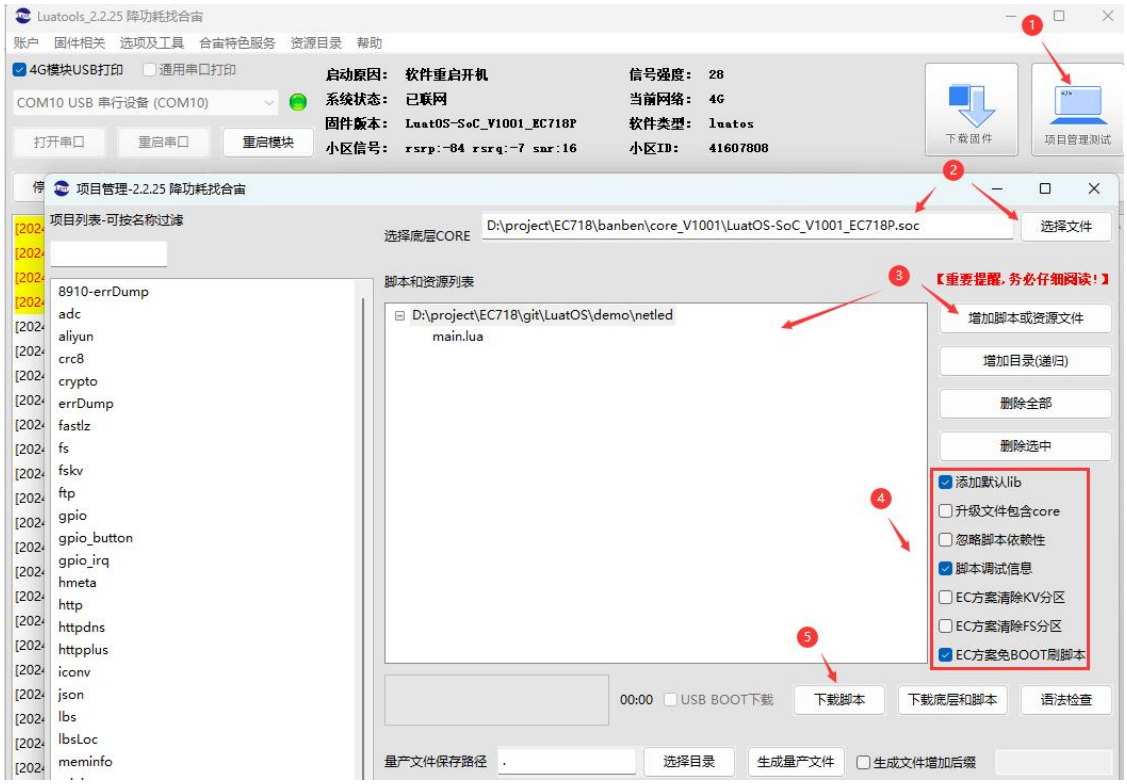
二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

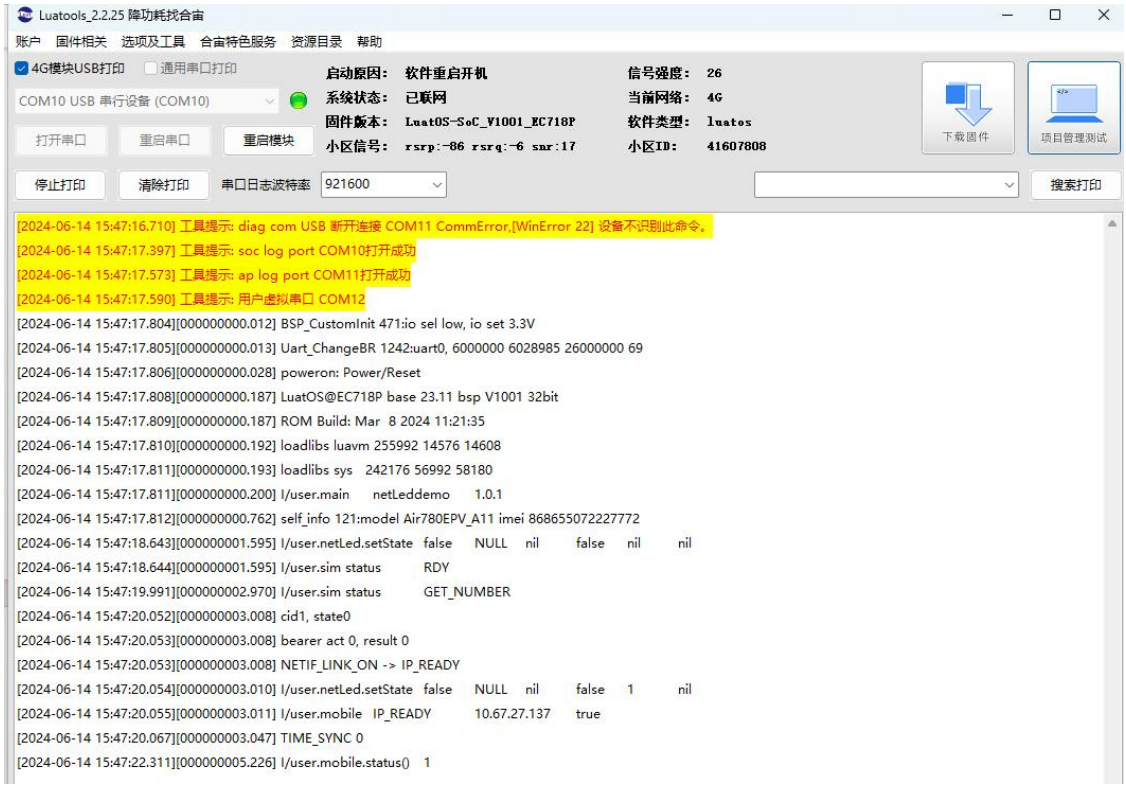
三．演示步骤

本文以 demo/netled 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



上面的 DEMO 下载到模块后，模块注册网络后，模块对应绿色网络状态指示灯会显示呼吸灯效果，对应 log 如下：



四．代码解析

点击右侧查看 netled API 说明：netled API

完整脚本代码点击右侧查看 demo/netled/main.lua

gpio.setup：配置 nedled 对应的 GPIO

mobile.status() == 1：注册网络成功后，显示呼吸灯效果

netLed.setupBreateLed(LEDA)：呼吸灯，LEDA 为 gpio.setup 返回值

```
--LED引脚判断赋值结束

local LEDA= gpio.setup(27, 0, gpio.PULLUP)


sys.taskInit(function()
    --流水灯程序
    sys.wait(5000) --延时5秒等待网络注册
    log.info("mobile.status()", mobile.status())
    while true do
        if mobile.status() == 1 then
            sys.wait(600)
            netLed.setupBreateLed(LEDA)
        else
            sys.wait(3000)
            log.info("net fail")
        end
    end
end)
```

10.5.27 uart 开发应用指南

一. 概述

串口通讯(Serial Communication)是一种设备间非常常用的串行通讯方式，主要按位(bit)发送和接收字节，典型地，串口用于 ASCII 码字符的传输。通信使用 3 根线完成，分别是**地线、发送、接收**。由于串口通信是异步的，端口能够在一根线上发送数据同时在另一根线上接收数据。其他线用于握手，但不是必须的。串口通信最重要的参数是**波特率、数据位、停止位和奇偶校验**。对于两个进行通信的端口，这些参数必须匹配。因为它简单便捷，因此大部分电子设备都支持该通讯方式，电子工程师在调试设备时也经常使用该通讯方式输出调试信息。

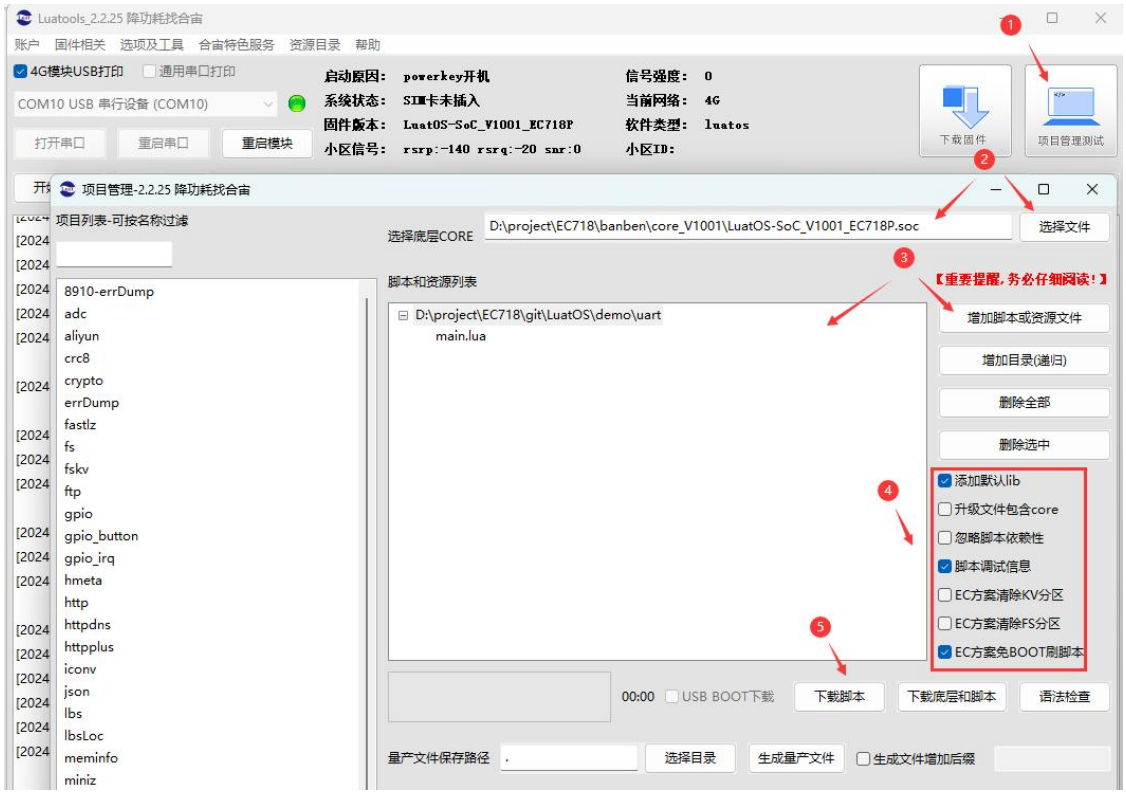
二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

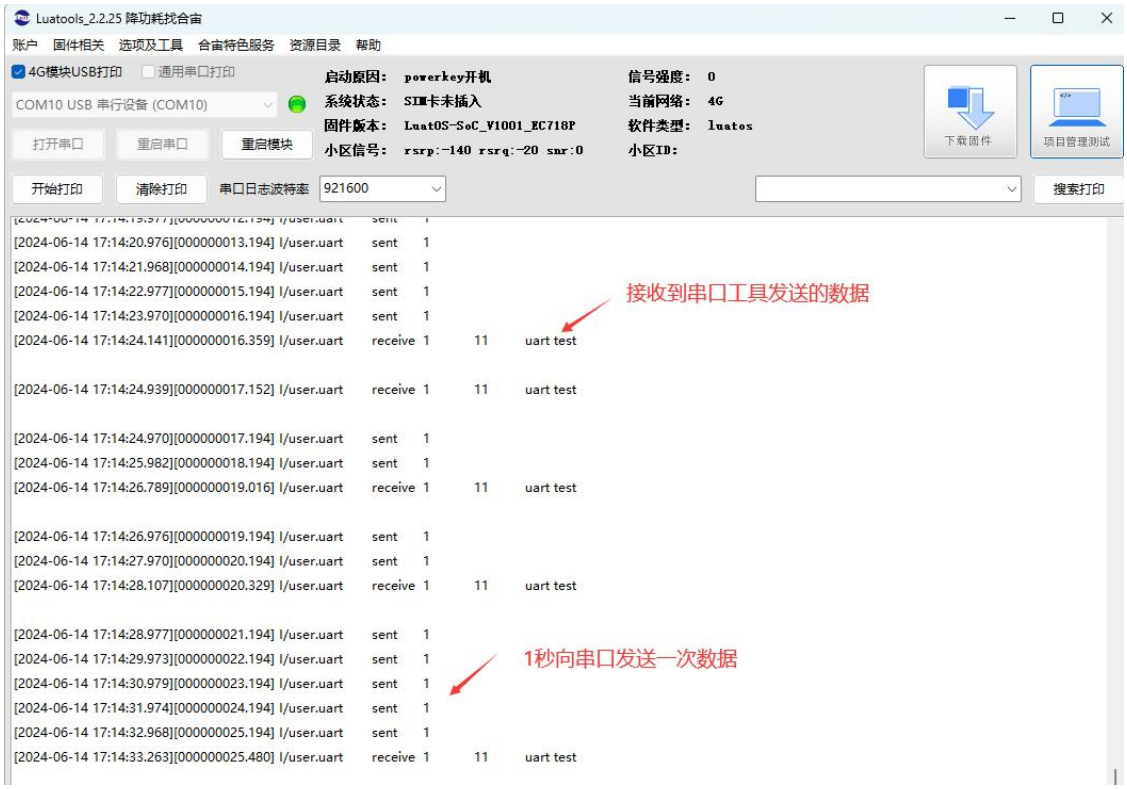
三．演示步骤

本文以 demo/uart 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块打印 log



四．代码解析

点击右侧查看 uart API 说明：uart API

完整脚本代码点击右侧查看 demo/uart/main.lua

uart.setup：配置串口参数，这用的是 mian 主串口，串口 id 为 1

```
local uartid = 1 -- 根据实际设备选取不同的uartid

--初始化
uart.setup(
    uartid,--串口id
    115200,--波特率
    8,--数据位
    1--停止位
)
```

uart.write：写串口数据

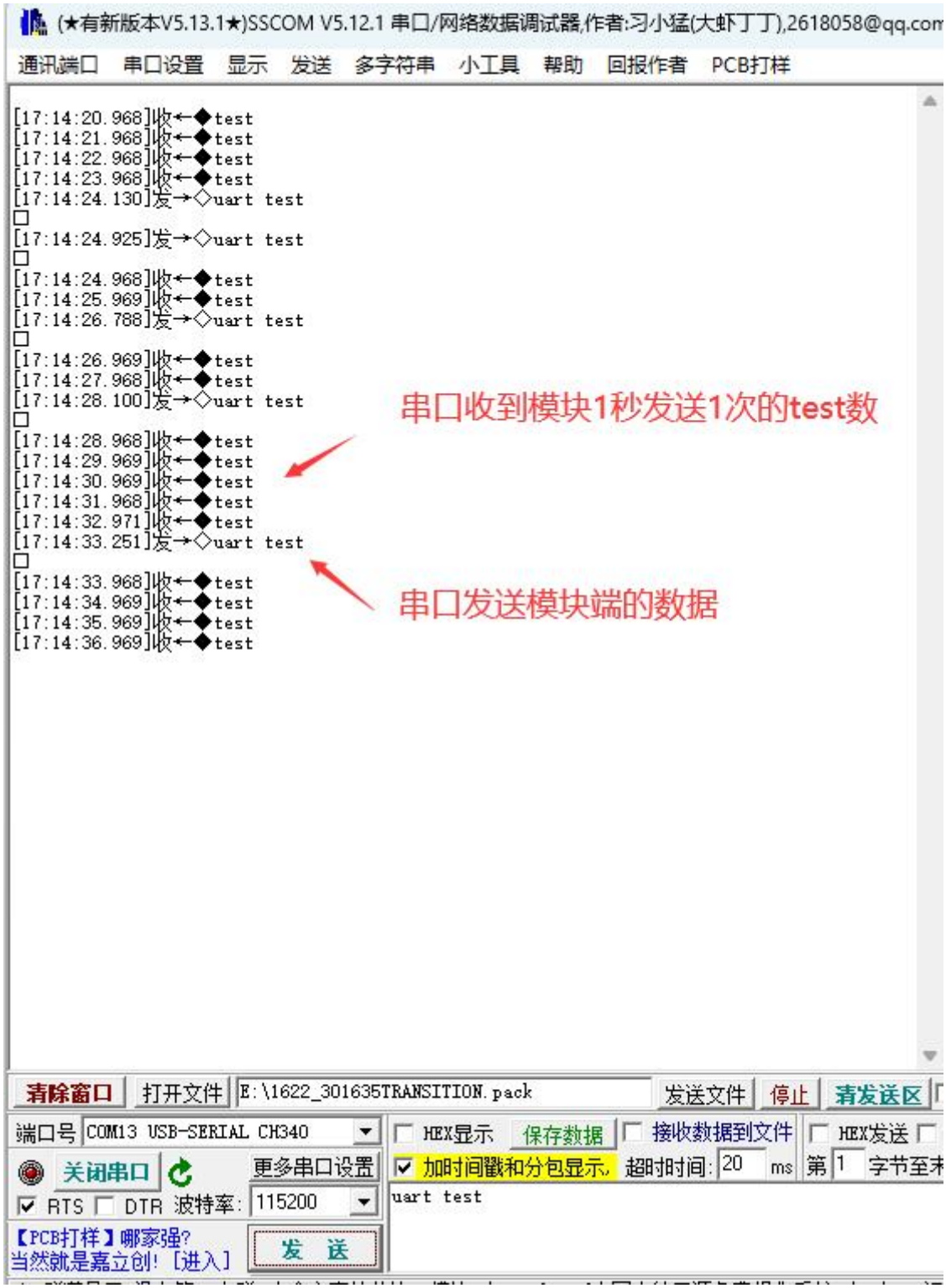
```
--循环发数据
sys.timerLoopStart(uart.write,1000, uartid, "test")
```

uart.on：注册串口事件回调

uart.read(id, 128)：每次读取 128 字节的串口数据

```lua
-- 收取数据会触发回调，这里的"receive" 是固定值
uart.on(uartid, "receive", function(id, len)
    local s = ""
    repeat
        s = uart.read(id, 128)
        if #s > 0 then -- #s 是取字符串的长度
            -- 如果传输二进制/十六进制数据，部分字符不可见，不代表没收到
            -- 关于收发hex值，请查阅 https://doc.openluat.com/article/583
            log.info("uart", "receive", id, #s, s)
            -- log.info("uart", "receive", id, #s, s:toHex())
        end
        -- 如使用2024.5.13之前编译的ESP32C3/ESP32S3固件，恢复下面的代码可以正常工作
        -- if #s == len then
        --     break
        -- end
    until s == ""
end)
```

串口工具发送或接收的数据



五．常见问题

为什么串口通信中会出现一些乱码数据？

串口波特率不正确或者串口通信的电平不匹配。

首先确认代码中所配置的通信的波特率与用户使用通信设备的波特参数设置的是否一直，然后确认用户所使用的通信设备的串口通信电平是否与模块串口电平一致，如不匹配，请确保

一致。不能过高或过低，过低有可能会导致电平电压不能被识别，过高有可能会损坏 UART 通信接口。

10.5.28 pwm 开发应用指南

一．概述

PWM（Pulse Width Modulation）是利用微处理器的数字输出来对模拟电路进行控制的一种非常有效的技术，广泛应用在从测量、通信到功率控制与变换的许多领域中。

二．材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

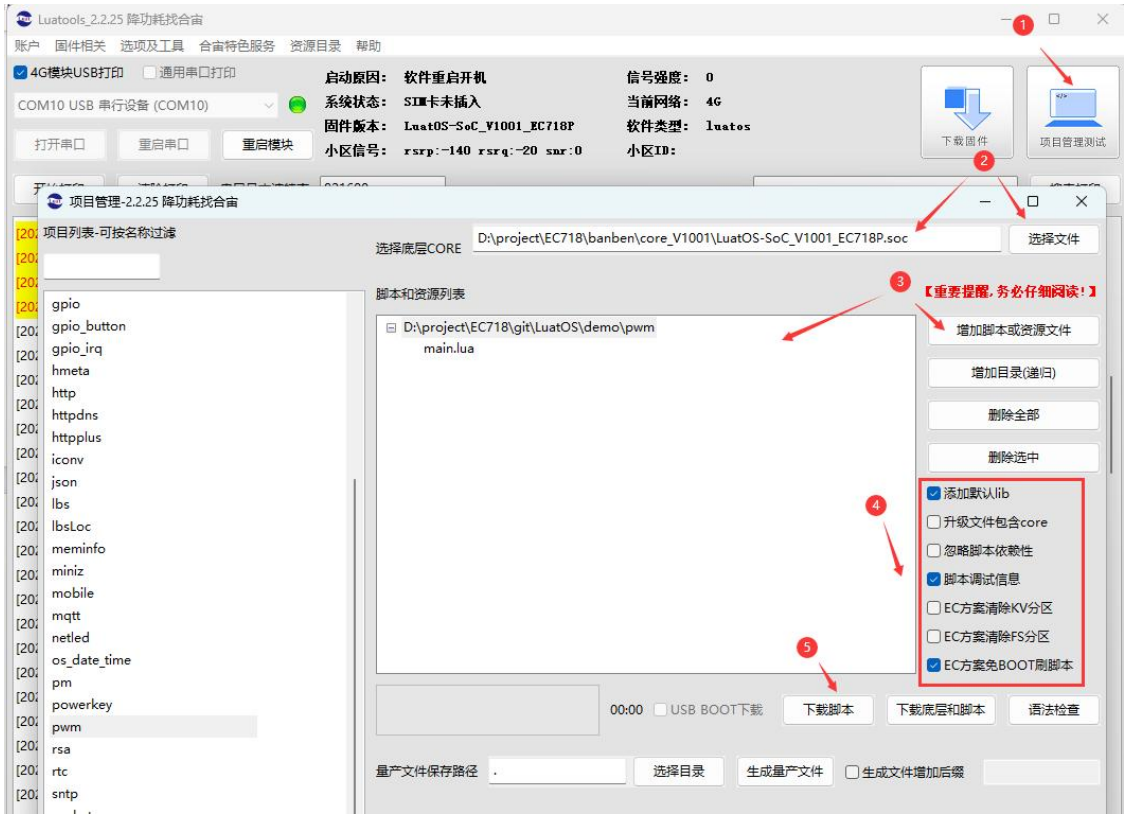
三．演示步骤

本文以 demo/pwm 这个 DEMO 为例做演示

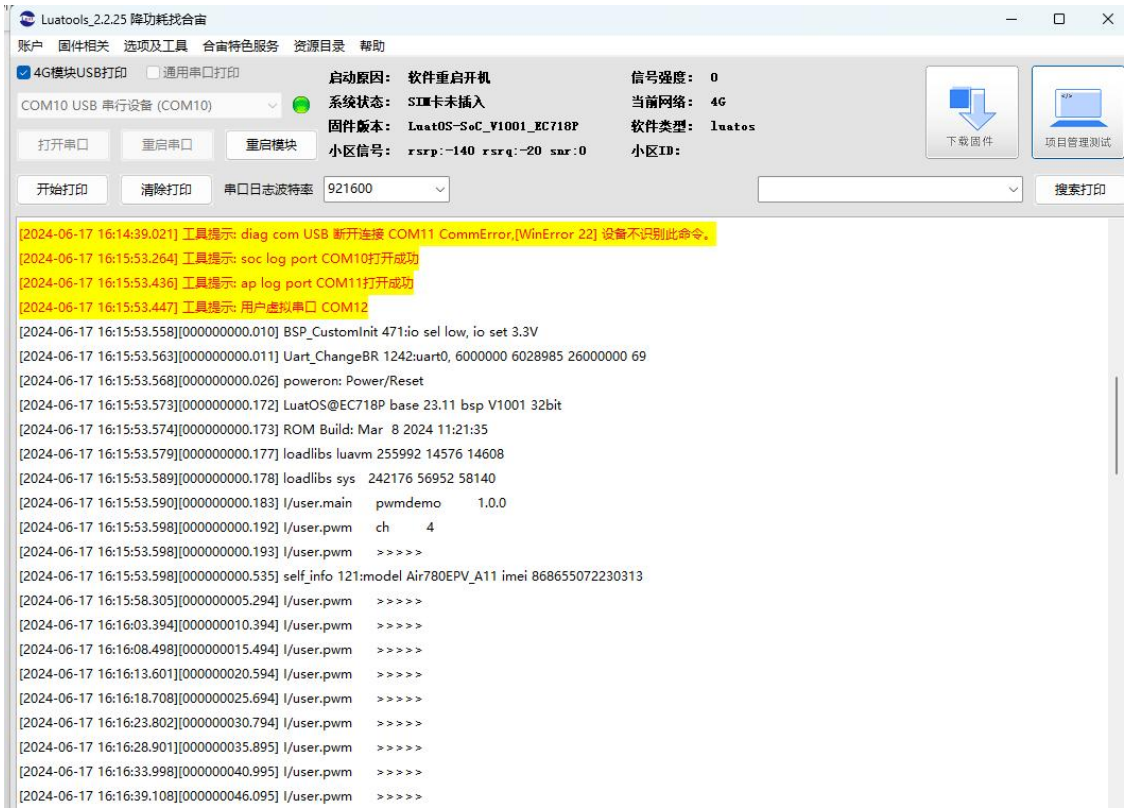
开发板对应的网络灯为 PWM4（GPIO 27）,这里修改 PWM_ID 为 4

```
elseif rtos.bsp() == "EC718P" then
    PWM_ID = 4 -- GPIO 27, NetLed
```

下载修改后对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



下面到开发板后可以看到左侧绿色网络灯呼吸灯效果，模块对应 log



四．代码解析

点击右侧查看 PWM API 说明：pwm API

完整脚本代码点击右侧查看 demo/pwm/main.lua

设置对应的 PWM_ID，这里设置为 4

```lua
local PWM_ID = 0
if rtos.bsp() == "EC618" then
    PWM_ID = 4 -- GPIO 27, NetLed
elseif rtos.bsp() == "EC718P" then
    PWM_ID = 4 -- GPIO 27, NetLed
```

仿呼吸灯效果，通过 pwm.open 来设置不同的占空比，来实现仿呼吸灯效果

```
sys.taskInit(function()
    log.info("pwm", "ch", PWM_ID)
    while 1 do
        -- 仿呼吸灯效果
        log.info("pwm", ">>>>>")
        for i = 10,1,-1 do
            pwm.open(PWM_ID, 1000, i*9) -- 频率1000hz, 占空比0-100
            sys.wait(100 + i*10)
        end
        for i = 10,1,-1 do
            pwm.open(PWM_ID, 1000, 100 - i*9)
            sys.wait(100 + i*10)
        end
        sys.wait(2000)
    end
end)
```

五. 常见问题

为什么配置后输出没波形

看通道是否配置正确，，gpio 口是否被占用。LuatOS 固件下的 IO 复用配置

10.5.29 string 开发应用指南

一. 概述

字符串（即 string），就是一串文本数据，可以存储你要的文本。

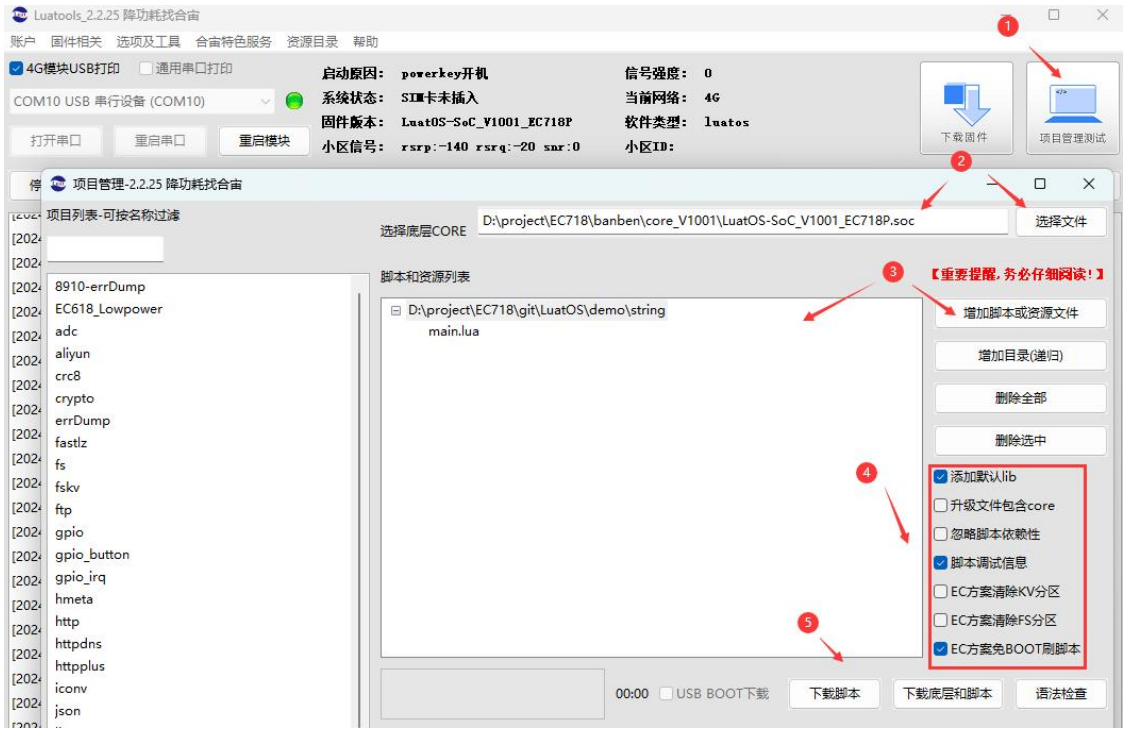
二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

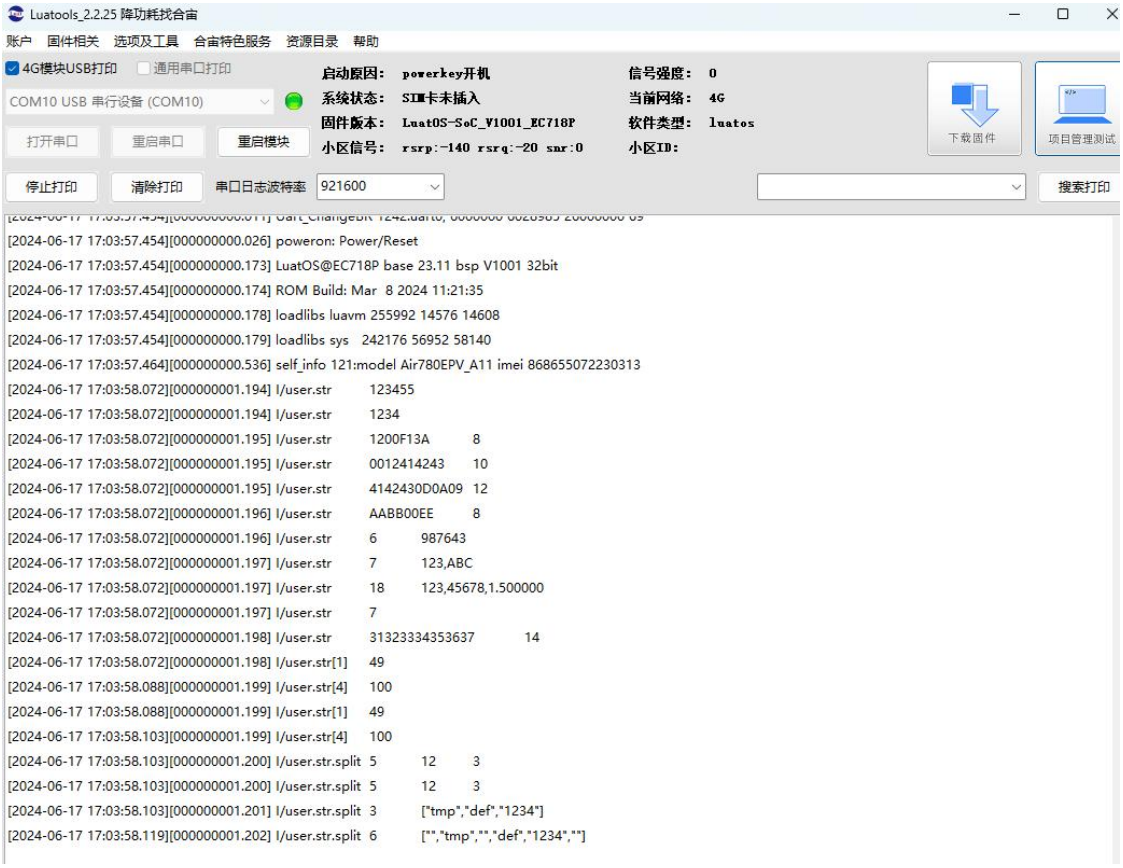
三．演示步骤

本文以 demo/string 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



模块对应 log



四．代码解析

点击右侧查看 string API 说明：sring API 详解 string 库

完整脚本代码点击右侧查看 demo/string/main.lua

Lua 语言中字符串可以使用以下三种方式来表示：

1. 单引号间的一串字符

2. 双引号间的一串字符

3. [[和]]间的一串字符

DEMO 中的脚本用的是双引号间的一串字符的方式来表示的。

string.char：接收 0 个或更多的整数（整数范围：0~255），返回这些整数所对应的 ASCII 码字符组成的字符串。当参数为空时，默认是一个 0。

str:toHex()：将字符串转成 HEX

string.fromHex：将 HEX 转成字符串

使用..来表示字符串拼接符号

```lua
-- 常量声明
local str = "123455"
log.info("str", str)

-- 合成式
str = string.char(0x31, 0x32, 0x33, 0x34)
log.info("str", str)
-- lua的字符串可以包含任意数据，包括 0x00
str = string.char(0x12, 0x00, 0xF1, 0x3A)
log.info("str", str:toHex()) -- 注意，这里用toHex()，因为包含了不可见字符

-- 使用转义字符
str = "\x00\x12ABC"
log.info("str", str:toHex()) -- 注意，这里用toHex()，因为包含了不可见字符
str = "ABC\r\n\t"
log.info("str", str:toHex()) -- 注意，这里用toHex()，因为包含了不可见字符


-- 解析生成
str = string.fromHex("AABB00EE")
log.info("str", str:toHex())
str = string.fromHex("393837363433")
log.info("str", #str, str)

-- 连接字符串，操作符 ".."
str = "123" .. "," .. "ABC"
log.info("str", #str, str)
```

string.format(formatstring, …)

按照格式化参数 formatstring，返回后面...内容的格式化版本。

编写格式化字符串的规则与标准 c 语言中 printf 函数的规则基本相同：

它由常规文本和指示组成，这些指示控制了每个参数应放到格式化结果的什么位置，及如何放入它们。

一个指示由字符%加上一个字母组成，这些字母指定了如何格式化参数，例如 d 用于十进制数、x 用于十六进制数、o 用于八进制数、f 用于浮点数、s 用于字符串等。

```lua
-- 格式化生成
str = string.format("%s,%d,%f", "123", 45678, 1.5)
log.info("str", #str, str)
```

#str：获取长度

str:byte(1)：获取指定位置的所对应的 ASCII 码，注意 lua 的下标是 1 开始的

这个函数功能刚好和前面的 string.char 相反，是提取字符串中实际的数值

str:split：按照指定分隔符分割字符串

```
--------------------------------------------------
-- 字符串的解析与处理
--------------------------------------------------
-- 获取长度
str = "1234567"
log.info("str", #str)
-- 获取字符串的HEX字符串显示
log.info("str", str:toHex())

-- 获取指定位置的值，注意lua的下标是1开始的
str = "123ddss"
log.info("str[1]", str:byte(1))
log.info("str[4]", str:byte(4))
log.info("str[1]", string.byte(str, 1))
log.info("str[4]", string.byte(str, 4))

-- 按字符串分割
str = "12,2,3,4,5"
tmp = str:split(",")
log.info("str.split", #tmp, tmp[1], tmp[3])
tmp = string.split(str, ",") -- 与前面的等价
log.info("str.split", #tmp, tmp[1], tmp[3])
str = "/tmp//def/1234/"
tmp = str:split("/")
log.info("str.split", #tmp, json.encode(tmp))
-- 2023.04.11新增的，可以保留空的分割片段
tmp = str:split("/", true)
log.info("str.split", #tmp, json.encode(tmp))
```

五．常见问题

字符串打印为空。原因是字符串里有不可见字符比如“空字符”，解决方式 toHex()后打印。

10.5.30 sim_det 开发应用指南

一．概述

SIM 卡热插拔检测功能是指模块开机后 SIM 插上或拔去，模块能够检测判断 SIM 卡是否在位。

sim 卡座的 cd 检测管脚接到 gpio 口，可以指定任意没被复用的 gpio 做 sim 卡检测，gpio 配置为中断模式，可以在 demo 中改为上升沿检测拔出，或下降沿拔出都可以。

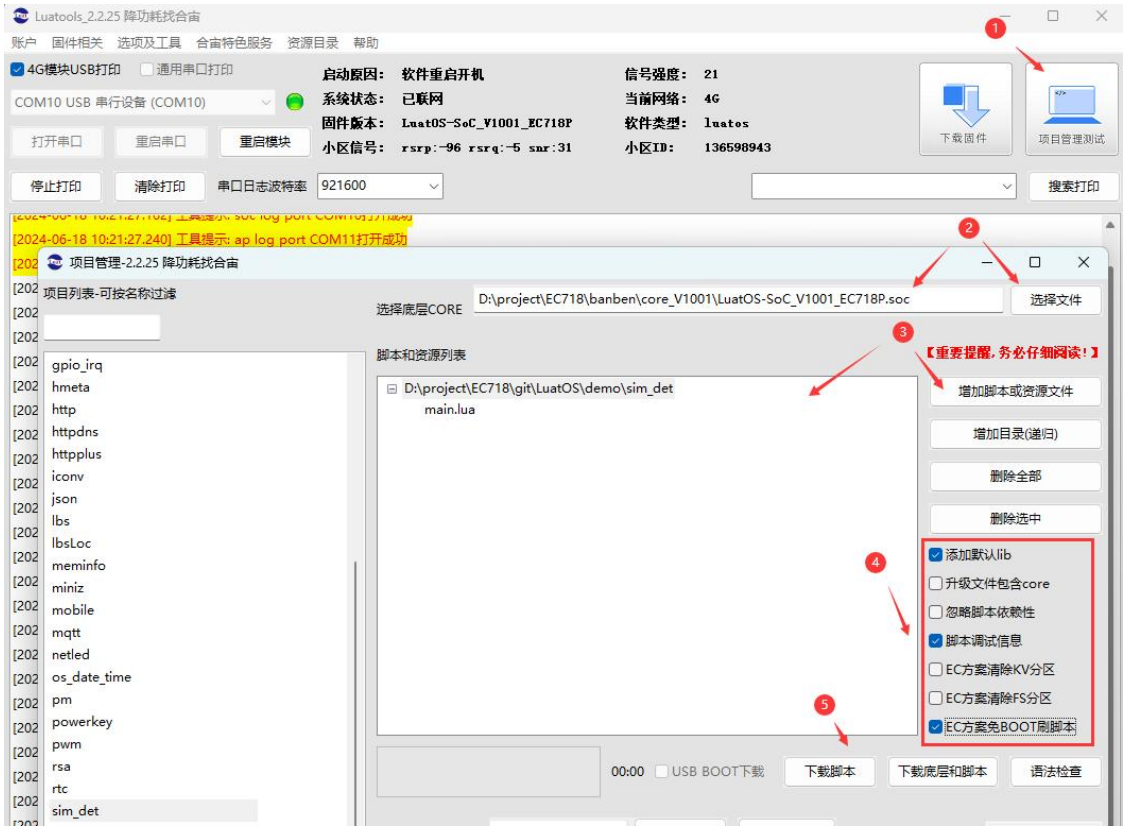
二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

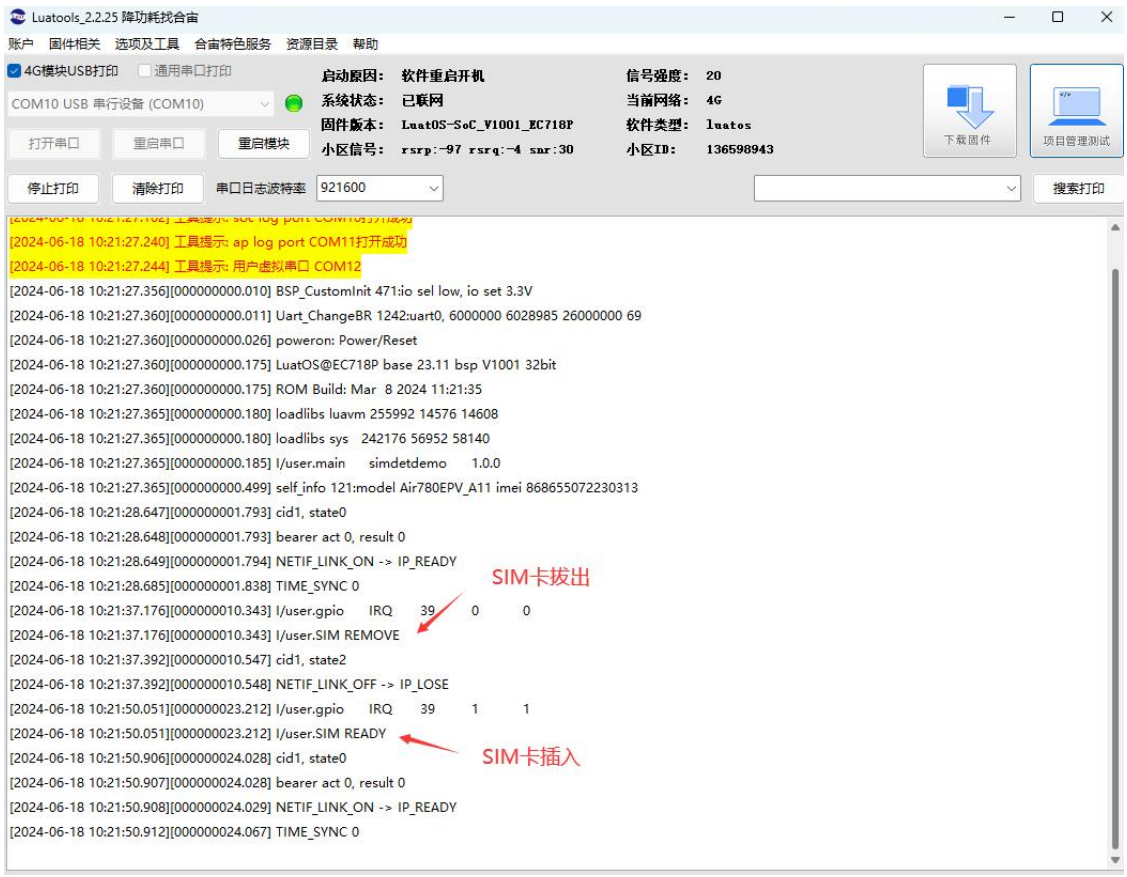
三. 演示步骤

本文以 demo/sim_det 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本



修改 DEMO 中的 gpio_pin 为 39，对应 wakeup0 管脚，下载对应固件和脚本后，把 wakeup0 管脚与 GND 短接，模拟 SIM 卡把出效果，断开 wakeup0 管脚与 GND 短接，模拟 SIM 卡插入效果。模块打印 log 如下：



四．代码解析

完整脚本代码点击右侧查看 demo/sim_det/main.lua

这个里 gpio_pin 配置为 39，为 wakeup0 管脚 LuatOS 固件下的 IO 复用配置

gpio.setup：设置为中断模式，并在中断回调函数中，打印对应状态和做相应的处理。

SIM 卡拔出的时候进入飞行模块，SIM 卡插入的时候退出飞行模式，重新驻网。

```
--配置gpio为输入模式，下拉，并会触发中断
--请根据实际需求更改gpio编号和上下拉
--SIMDET配置为39
local gpio_pin = 39
gpio.debounce(gpio_pin, 200, 1)
gpio.setup(gpio_pin, function(val)
    local status = gpio.get(gpio_pin)
    log.info("gpio", "IRQ",gpio_pin, val, status)
    if status == 0 then
        log.info("SIM REMOVE")
        mobile.flymode(0, true)
    else
        log.info("SIM READY")
        mobile.flymode(0, false)
    end
end, gpio.PULLUP)
```

五．常见问题

sim 卡不识别问题？

sim 卡不识别按以下步骤进行：

1：看模块开机没？at 指令能不能用？at+cpin?返回 ready,代表读到卡，如果返回 error,按以下几点排查：

2：sim 卡是不是 Ok？sim 卡方向有没反，注意直插的卡座缺口是朝外

3：每个脚对地用万用表，打到二级管档位量下，红接地，黑接每个管脚，值是 400-500 之间正常。

4：看模块与卡座之间连线有没有问题，引脚对地阻值是不是正常，引脚间有没短路，

原理图和 Pcb 网络名与 sim 卡座实物对不对，卡座是否接触良好，如果 sim 卡一插入，vdd 和 clk 就对地短路，一般是卡座问题。

5：示波器测试下 sim_vdd 波形，有个 1.8-3.3v 的高电平跳变，如果波形正常，一般是卡座和模块引脚之间问题，要识别到卡，才会有电压输出。

6：如果是贴片卡，可以先拆下贴片卡，飞线到一个可以识别到卡的卡座到模块管脚，看下是否贴片卡问题。

10.5.31 usb_uart 开发应用指南

一. 概述

若使用普通串口调试功能，需要以下条件之一才能收到数据

1. 把 DTR 设置为高电平
2. 先发送一次数据
3. USB 虚拟串口的固定 id：uart.VUART_0

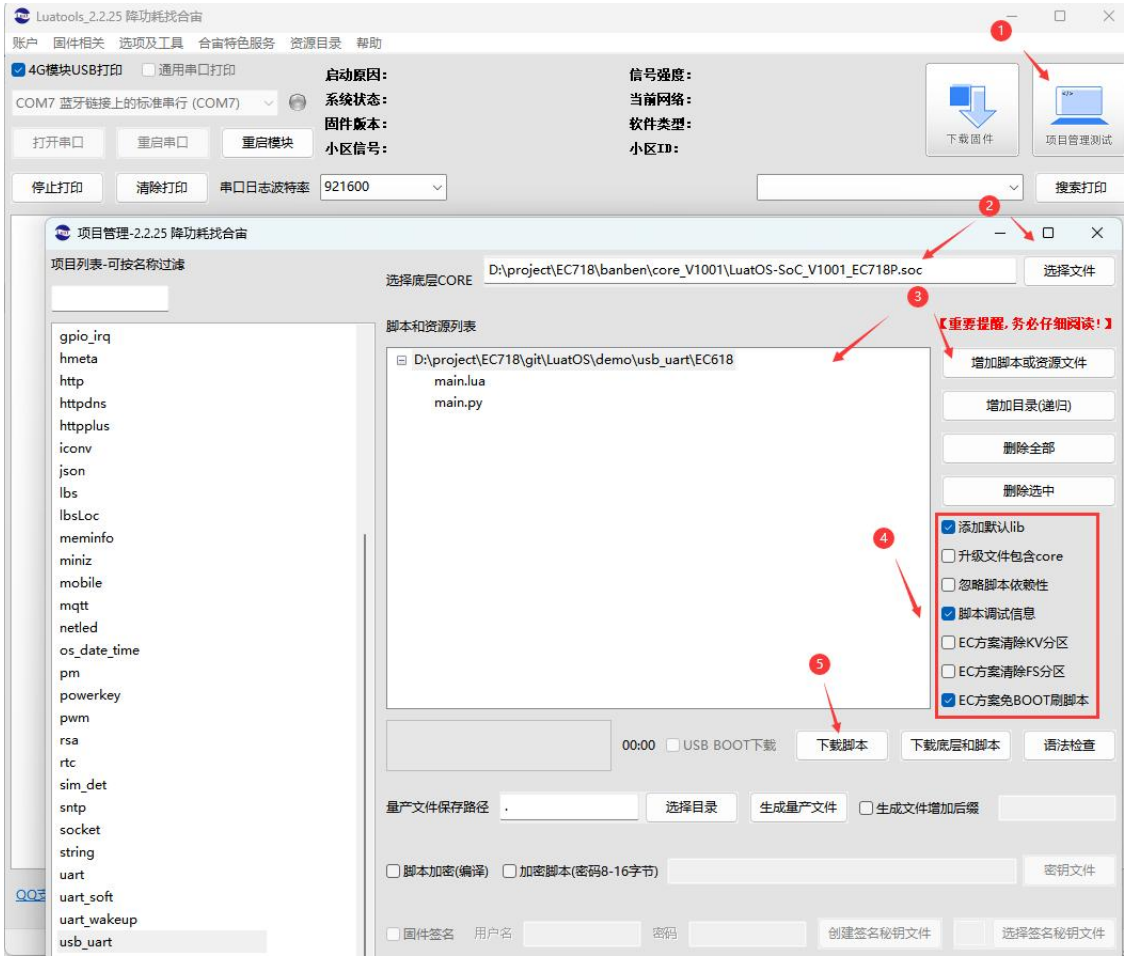
二. 材料准备

1. Air780EP(V)开发板一套，包括天线 SIM 卡，USB 线。
2. PC 电脑
3. 固件和 DEMO 下载地址 LuatOS 软件包

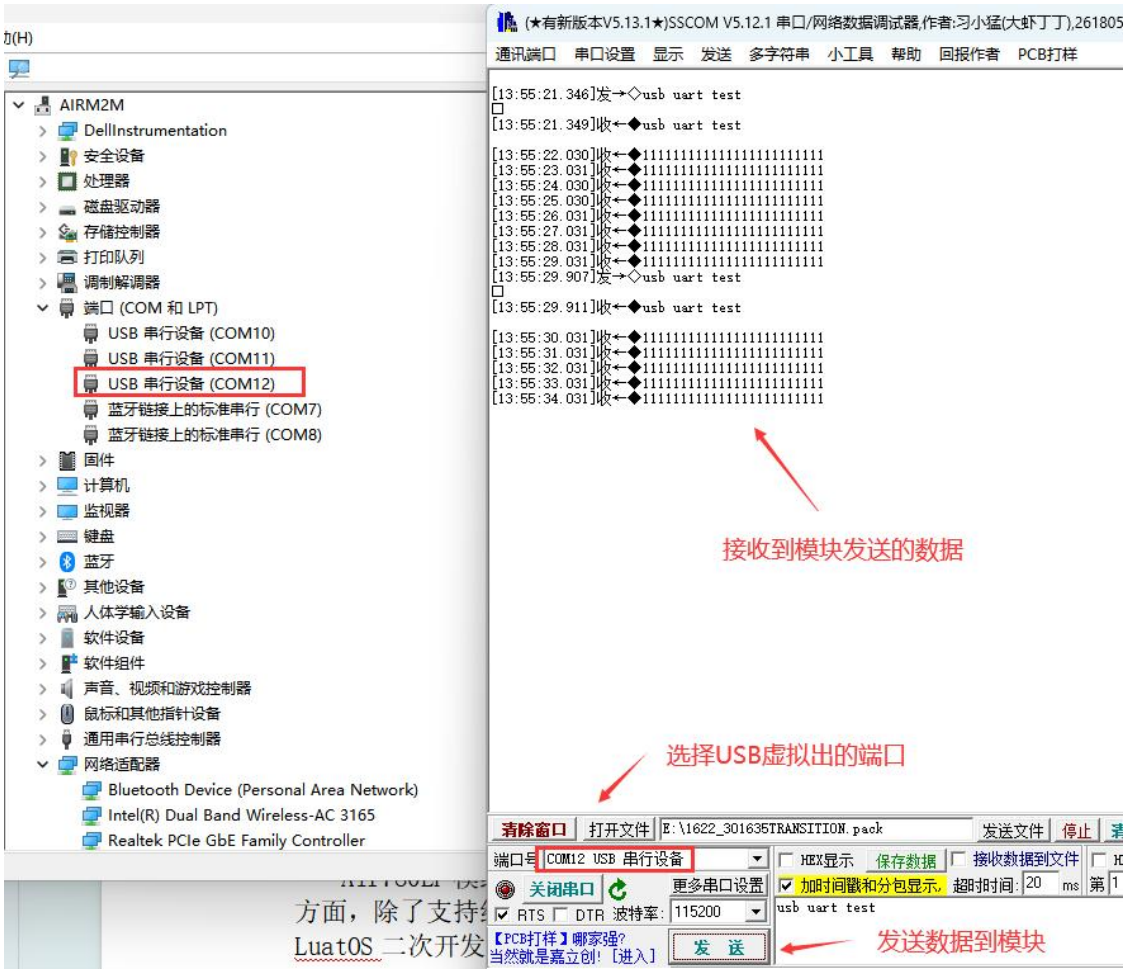
三. 演示步骤

本文以 demo/usb_uart/EC618 这个 DEMO 为例做演示

直接下载对应 core 和 script 到模块，这里 core 不变，可以直接下载对应的脚本


Luatools_2.2.25 降功耗找合宙
账户 固件相关 选项及工具 合宙特色服务 资源目录 帮助
4G模块USB打印
通用串口打印
COM7 蓝牙链接上的标准串行 (COM7)
打开串口
重启串口
重启模块
停止打印
清除打印
串口日志波特率
921600
启动原因:
系统状态:
固件版本:
小区信号:
信号强度:
当前网络:
软件类型:
小区ID:
下载固件
项目管理测试
搜索打印
项目管理-2.2.25 降功耗找合宙
项目列表-可按名称过滤
gpio_irq
hmeta
http
httpdns
httpplus
iconv
json
lbs
lbsLoc
meminfo
miniz
mobile
mqtt
netled
os_date_time
pm
powerkey
pwm
rsa
rtc
sim_det
sntp
socket
string
uart
uart_soft
uart_wakeup
usb_uart
选择底层CORE
D:\project\EC718\banben\core_V1001\LuatOS-SoC_V1001_EC718P.soc
选择文件
脚本和资源列表
【重要提醒，务必仔细阅读！】
D:\project\EC718\git\LuatOS\demo\usb_uart\EC618
main.lua
main.py
增加脚本或资源文件
增加目录(递归)
删除全部
删除选中
添加默认lib
升级文件包含core
忽略脚本依赖性
脚本调试信息
EC方案清除KV分区
EC方案清除FS分区
EC方案免BOOT刷脚本
00:00
USB BOOT下载
下载脚本
下载底层和脚本
语法检查
量产文件保存路径
选择目录
生成量产文件
生成文件增加后缀
脚本加密(编译)
加密脚本(密码8-16字节)
密钥文件
固件签名
用户名
密码
创建签名秘钥文件
选择签名秘钥文件

串口操作效果



四．代码解析

完整脚本代码点击右侧查看 demo/usb_uart/EC618/main.lua

local uartid = uart.VUART_0 -- USB 虚拟串口的固定 id

uart.setup：配置串口参数

```
local uartid = uart.VUART_0 -- USB虚拟串口的固定id

--初始化
local result = uart.setup(
    uartid,--串口id
    115200,--波特率,其实无所谓, 纯虚拟串口
    8,--数据位
    1--停止位
)
```

uart.on：注册串口事件回调

```
-- 收取数据会触发回调，这里的"receive" 是固定值
uart.on(uartid, "receive", function(id, len)
    local s = ""
    repeat
        -- s = uart.read(id, 1024)
        s = uart.read(id, len)
        if s and #s > 0 then -- #s 是取字符串的长度
            -- 如果传输二进制/十六进制数据，部分字符不可见，不代表没收到
            -- 关于收发hex值，请查阅 https://doc.openluat.com/article/583
            log.info("uart", "receive", id, #s, s)
            uart.write(uart.VUART_0, s)
            -- log.info("uart", "receive", id, #s, s:toHex())
        end
    until s == ""
end)
```

uart.tx：buff 形式写串口,等同于 c 语言 uart_tx(uart_id, &buff[start], len);

```
local tx_buff = zbuff.create(24)
tx_buff:set(0, 0x31)
-- 并非所有设备都支持sent事件
uart.on(uartid, "sent", function(id)
    log.info("uart", "sent", id)
end)

sys.taskInit(function()

    while 1 do
        -- uart.write(uart.VUART_0, "hello test usb-uart\r\n")
        uart.tx(uart.VUART_0, tx_buff,0, tx_buff:len())
        sys.wait(1000)
    end
end)
```

五．常见问题

模块不开机&不识别 usb 端口&烧录不了固件处理步骤如下：

1.正常情况，模块 vbat 供电，pwrkey 管脚拉到地 1.5s 以上就可以开机，VDD_EXT 管脚输出 1.8v(IO_SEL 管脚悬空)/3.3v (IO_SEL 管脚接地)就是开机标志，注意：如果 vbat 管脚供电了，pwrkey 接地了，模块 VDD_EXT 管脚也没输出，或者 VDD_EXT 管脚输出电压异常，可能 spi 或者串口，usb，sim 卡，VDD_EXT 管脚被静电浪涌打坏，注意：复位脚不能上拉，上拉开不了机

2.先看是什么情况下导致的？

（1） 如果是烧录程序过程中导致不识别端口，大概率是变砖，用 uboot 方式烧录,烧录方法参考第 8 点。

（2） 如果是第一次打样，按如下方式排查：

3.接个好的板子看能不能识别到端口，排除电脑环境和 usb 线和 usb 驱动问题,检查下 usb 座

子到模块之间线路有没问题？

4.也可以把 usb 线路上的 tvs 管和电容都去掉，直接飞线到模块 usb 脚到电脑 usb 口测试。

5.模块供电脚 vbat，复位脚电压，VDD_EXT 脚电压是不是 ok ，VDD_EXT 有没对地短路，断电后万用表打到二极管档，红接地，黑接 vbat,VDD_EXT ,usb 三个管脚，串口，sim 卡几个管脚测量，数值在 400-600 左右是正常，如果偏差大可能有硬件损坏。

6.pwrkey 一直接地，vbat 供电后，VDD_EXT 出来没，如果有 1.8v(IO_SEL 管脚悬空)/3.3v(IO_SEL 管脚接地)输出说明开机了

7.上面都没问题，用 uboot 方式烧录个 at 固件试下。

8.uboot 强制烧录步骤：先把模块 pwrkey 脚对地短接，USB_BOOT(注意如果是 700e 要串个 51k 上拉到 vbat 才会进 boot)和 VDD_EXT 短接不松开，接好 usb 线到电脑，然后电脑 Luatools 工具上先选个 at 固件,点下载，注意:是最后给 vbat 管脚上电开机,看到 进度条在跑了就行了，烧录完后要断开 VDD_EXT 和 USB_BOOT 管脚，选一个最新的 at 固件，链接如下：https://doc.openluat.com/article/4922#AT_1

9.如果能烧录成功，但是不能识别到 usb 端口，一般是 tvs 管和电容导致，可以去掉 usb 上的 tvs 管和电容测试。

10.如果设备管理器一直看不到 usb 端口：

(1) 检查 vbus，dp,dm 数据线有没对地短路，阻值异常，是否被浪涌打坏。

(2）看是否 USB_BOOT 和 VDD_EXT 短路。

(3）看串口的 rx 和 tx 是否短路，串口是否对地短路。

(4) 看 VDD_EXT 是否对地短路。

(5)USB_BOOT 管脚是否一直被拉高，如果都 ok，就用 USB_BOOT 短接 VDD_EXT 重新烧录个 at 固件，参考第 8 点。

11，如果开机能看到 usb 三个端口，按住 boot 按键开机，进不了 boot 模式，usb 出现感叹号，不识别，（1）看是否 USB_BOOT 和 VDD_EXT 假焊，或者 USB_BOOT 和 VDD_EXT 之间的串联电阻是否过大，按键开关是否开路。（2）usb，dp,dm 到模块数据线之间串的电阻是否过大，建议 10 欧以下。(3)如果上面都排查了，一般是 usb 线的问题，usb 要求走差分，90 欧姆阻抗，不要用杜邦线转接，建议直接找个好的 usb 数据线剪断飞到模块 usb 管脚测试。

十一 项目实战（后续补充）

根据 https://gitee.com/openLuat/luatos-private-product 整理；
定位器（luatos）、iRTU 等完整的软硬件项目或者完整的软件项目文档、源码、设计说明；

定位器 luatos：1500 行代码，50 页；
定位器文字说明：50 页；

云喇叭和定位器类似，也是 300 页；

iRTU 代码：3400 行，110 页；
irtu 代码文字说明：50 页；

十二 云平台服务

差分包生成工具
云编译系统
iRTU 管理系统
errDump 远程日志
FOTA 远程升级
MES 查询
基站查询工具
基站 WIFI 定位
坐标转换
网络测试工具

12.1 差分包生成工具

Air780EP 模组的只有 AT 看法方式会用到云平台的差分工具，其它开发方式的差分用不到这个工具，直接使用 luatools。

AT 开发方式参考：

9.5.5 远程升级(FOTA)应用指南

LuatOS 开发方式参考：

10.5.1 FOTA 开发应用指南

12.2 云编译系统

还在困扰如何搭建编译环境？芯片越来越多，环境也越来越多？电脑好卡呀~好，云编译来为你排忧解难~。同时云编译定制固件功能，用户可以根据自己的项目需求，自行编译定制固，能更大限度的利用有限的 Flash 和 RAM 资源。

12.2.1 已支持的平台:

1.air101 - Air101 的 LuatOS 固件

2.air103 - Air103 的 LuatOS 固件

3.air105 - Air105 的 LuatOS 固件

4.idf5-esp32c3 - ESP32C3 的 LuatOS 固件

5.idf5-esp32s3 - ESP32S3 的 LuatOS 固件

6.ec618-luatos - ec618 系列(Air780E/Air600E 等)的 LuatOS 固件

7.ec618-csdk - ec618 系列(Air780E/Air600E 等)的 CSDK

8.air601 - Air601 的 LuatOS 固件

9.air601-at - Air601 的 AT 固件

10.ec718p-luatos - ec718P 系列(Air780EP/Air780EPV)的 LuatOS 固件

12.2.2 提醒

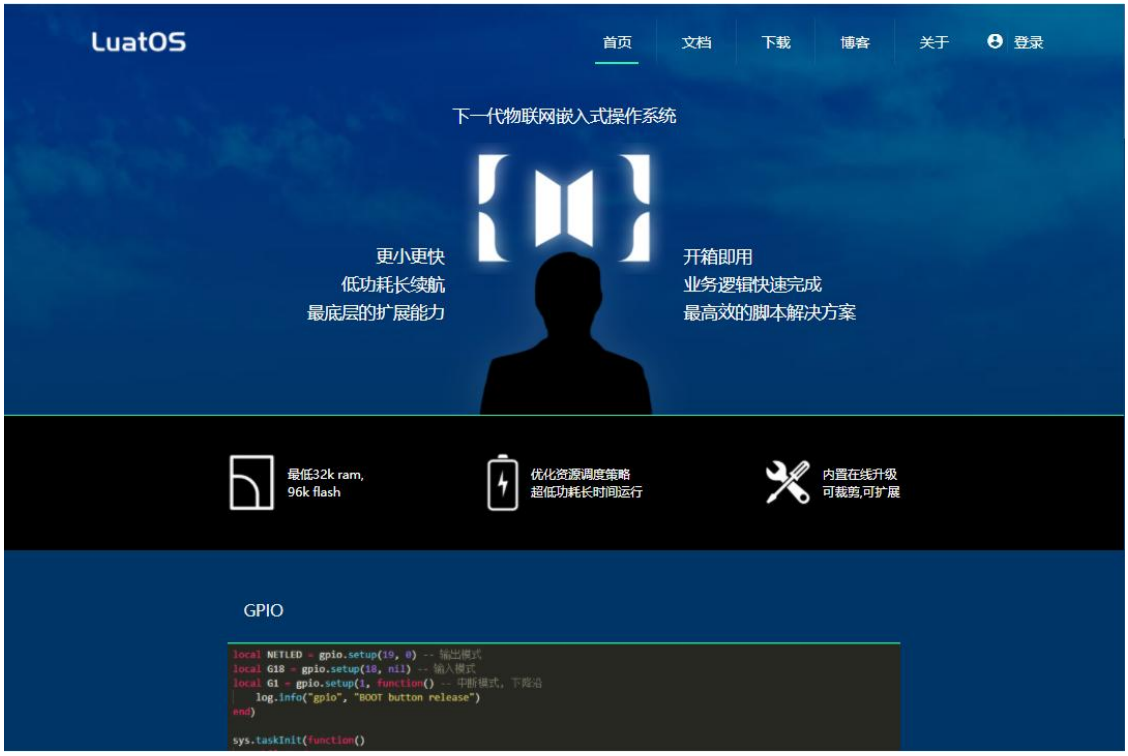
1.页面显示的 flash 占用为估算值,若编译成功就是能放得下

2.soc 文件本身也是压缩包(但不需要解压它),**soc 文件的大小并不代表底层固件的真实大小**

这里我们以 Air101 为展示

12.2.3 访问构建网站

https://luatos.com



12.2.4 点击登录

使用合宙 erp 账号登录即可. 若没有账号, 可免费注册. 本服务免费使用.

12.2.5 点击右上角构建

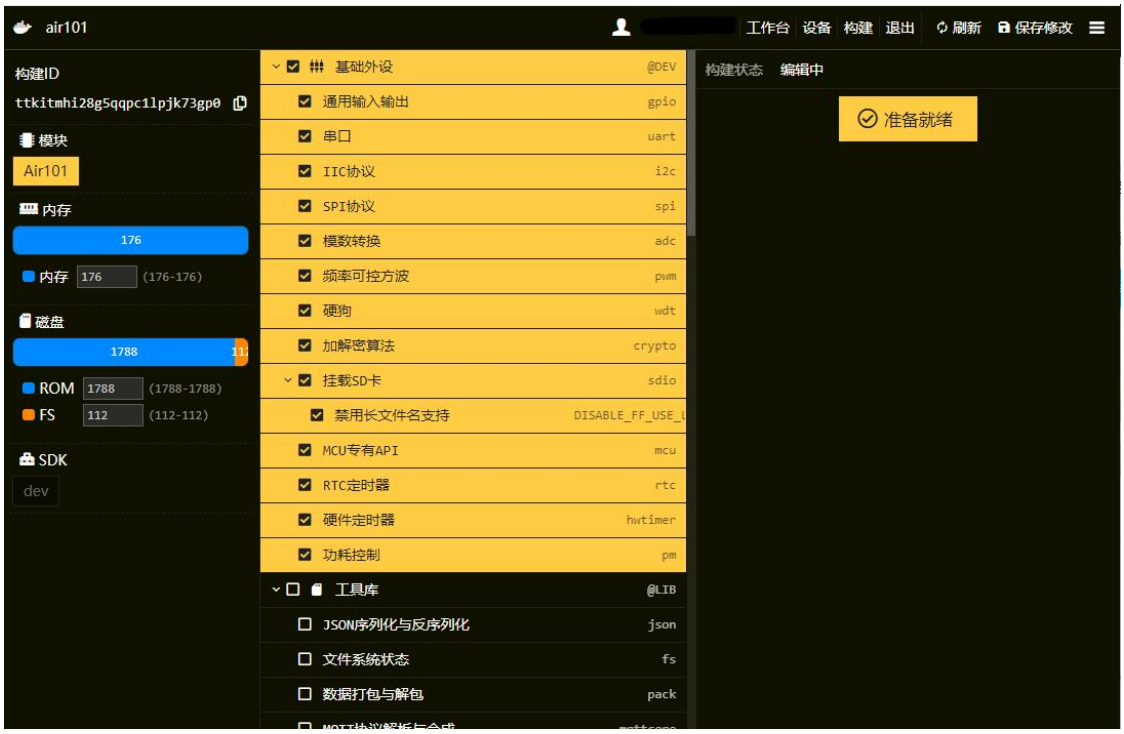
点击右上角菜单新构建，自定义构建名称，可以是任意英文字符，与具体构建类型无关.

设备
构建
退出
刷新
保存修改
编辑中
新构建...
切换当前构建...
删除当前构建...
查看源代码
ALT+SHIFT+V

询问
请输入新构建名称
air101
确定
取消

air101
工作台
设备
构建
退出
刷新
保存修改
构建ID
ttkitmhi28g5qqpc1lpjk73gp0
模块
Air101
内存
176
内存 176 (176-176)
磁盘
1788
ROM 1788 (1788-1788)
FS 112 (112-112)
SDK
dev
基础外设 @DEV
通用输入输出 gpio
串口 uart
IIC协议 i2c
SPI协议 spi
模数转换 adc
频率可控方波 pwm
硬狗 wdt
加解密算法 crypto
挂载SD卡 sdio
禁用长文件名支持 DISABLE_FF_USE_L
MCU专有API mcu
RTC定时器 rtc
硬件定时器 hwtimer
功耗控制 pm
工具库 @LIB
JSON序列化与反序列化 json
文件系统状态 fs
数据打包与解包 pack
MQTT协议解析与合成 mqttcore
构建状态 编辑中
准备就绪

12.2.6 勾选想要的组件



12.2.7 点击右上角保存修改，之后点击准备就绪

注意：一定要点击保存修改

等待编译，过一段时间刷新一下结果

编译成功，点击下载即可

12.2.8 常见问题：

为何我编译失败？

编译失败的问题有很多，通常有这几种比较常见：

1. 为未点击保存修改就点击准备就绪。
2. 组件选择过多，或字体的选择导致 flash 大小不够,导致编译失败
3. 固件包都是压缩包, soc 也是压缩包,并不直接代表固件大小
4. 若持续失败，请联系我们 或者报 issue

自定义字库怎么用

自定义字库的名称，与云编译页面上字库的“命名”值相同. 即 字体英文名_字体类型_字号

1. 自定义字库与原有字库不冲突
2. 自定义字库与原有字库的引用方式差异

```
-- 原有字库的引用方式
lcd.font_opposansm12_chinese
-- 自定义字库的引用方式
fonts.get("oppo_bold_12")
```

使用方式

```
-- 原有字库的引用方式，各自引用
lcd.setFont(lcd.font_opposansm12_chinese)

eink.setFont(eink.font_opposansm12_chinese)

u8g2.setFont(u8g2.font_opposansm12_chinese)

-- 自定义字库的引用方式，统一使用
lcd.setFont(fonts.get("oppo_bold_12"))

eink.setFont(fonts.get("oppo_bold_12"))

u8g2.setFont(fonts.get("oppo_bold_12"))
```

12.3 iRTU 管理系统

12.3.1iRTU 开发简介

一、LuatOS-iRTU 是什么

LuatOS-IRTU 是合宙采用 LuatOS 架构自主研发、并且免费开源的系统级主控 DTU 软件解决方案，实现了传统 DTU 的所有功能；除此之外，传统 DTU 不支持但市场需要的功能，LuatOS-iRTU 可能也支持，比如丰富的外设驱动库(ADC、I2C、SPI 等接口，支持传感器、LCD 等)，比如数据流模板(可自定义解析处理网络端和主控 MCU 端的双向数据)，比如自定义任务功能(可服务器配置编程脚本远程 OTA 更新，功能即时扩充)。

目前，LuatOS-iRTU 已经被众多 DTU 厂商采用，经过百万级装机设备的验证，稳定可靠，广泛应用于物联网领域的各行各业。

二、LuatOS-iRTU 支持什么功能

1. 支持 WEB 端和 PC 端配置功能

支持 TCP/UDP socket、HTTP、MQTT 等常见透传和非透传模式；

支持阿里云、OneNET、百度云、腾讯云、天翼云等常见公有云；

支持自动心跳，保持永久在线；

支持数据流模板，可自定义解析处理网络端和主控 MCU 端的双向数据；

支持消息推送(电话，短信，网络通知)；

支持 GPS 数据以及相关数据采集；

支持自定义任务功能，可以配置编程脚本，为软件系统的扩充提供便利；

支持 ADC、I2C、SPI 等外设接口，配合任务功能，可以方便的扩充传感器、LCD 等外设；

更多详情参考：实例

2. 支持远程升级功能

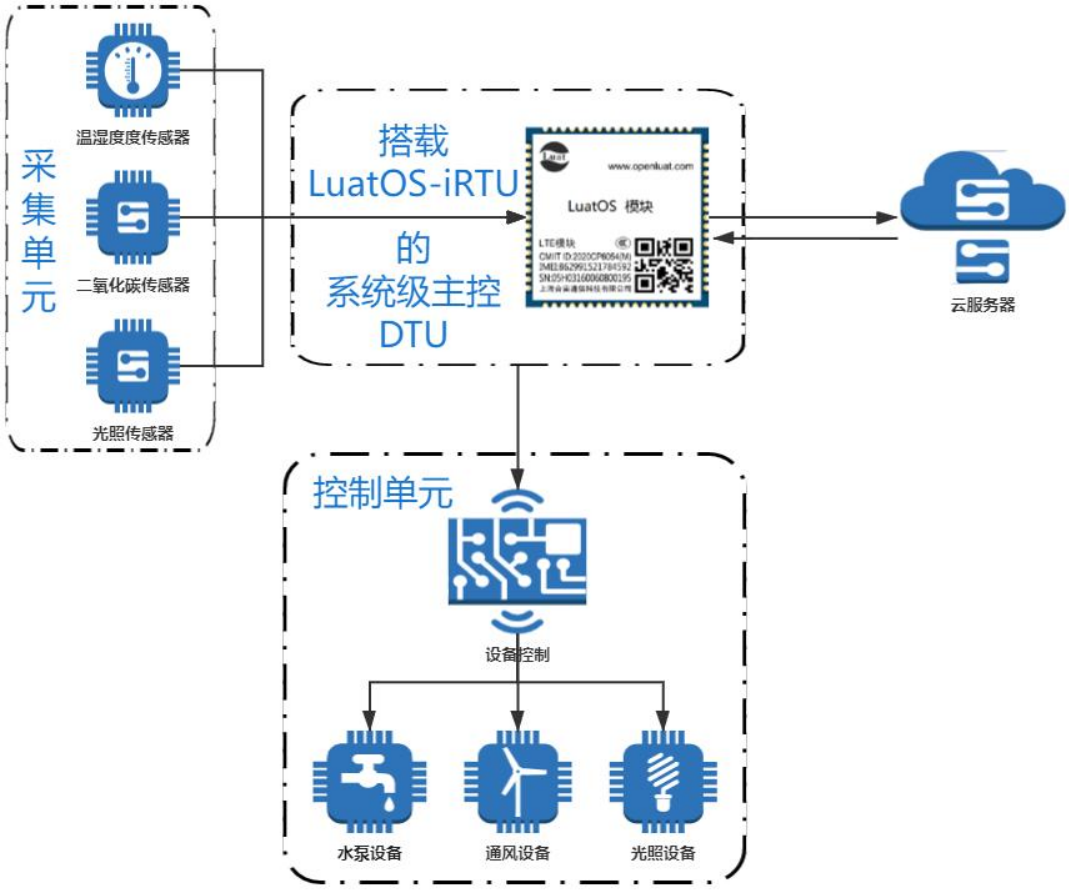
与其他厂商的远程升级收费服务不同，LuatOS-iRTU 提供永久免费的远程升级服务，在升级云平台简单配置几步，就可以对 DTU 固件进行远程升级。

3. 支持 DTU 管理后台域名定制

三、具有什么优势

搭载 LuatOS-iRTU 软件解决方案的 DTU，不仅仅是一种传统 DTU，更是一种“硬件成本大幅降低、业务功能扩充方便、脚本源码免费开放”的系统级主控 DTU。

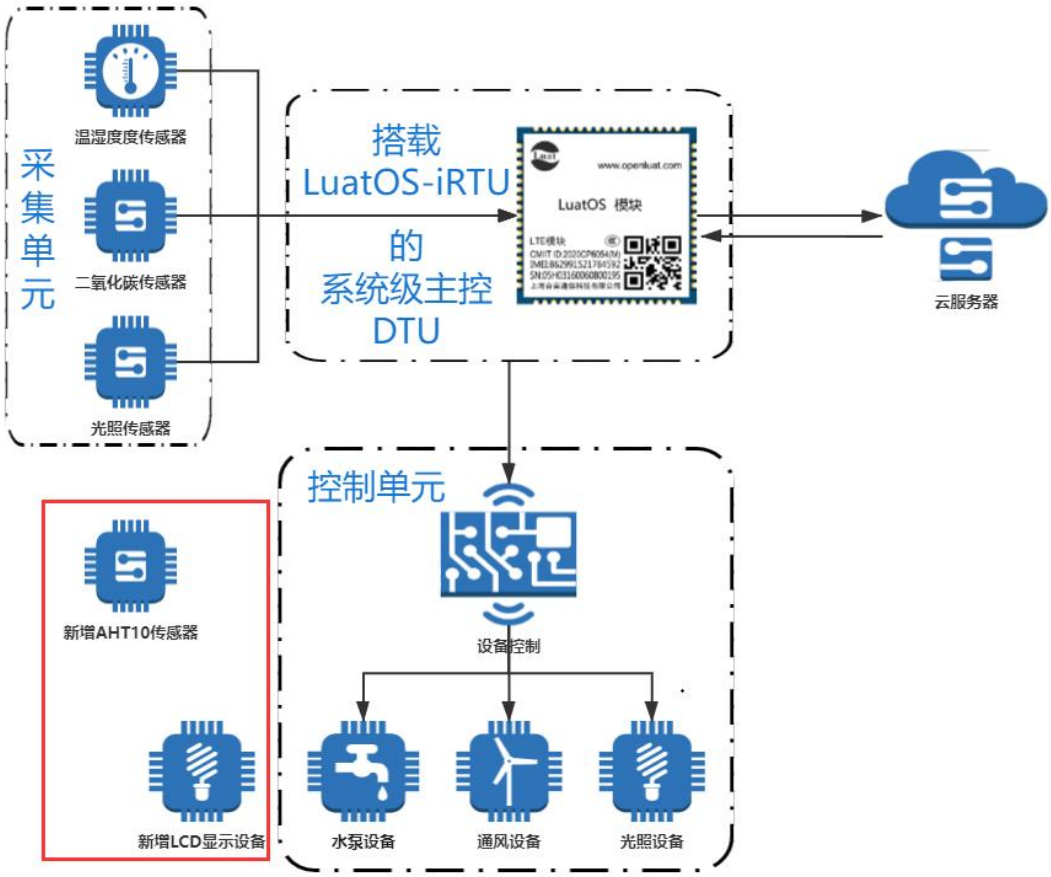
1. 硬件成本大幅降低



- 传统 DTU，由 MCU 和传感器通信，获取采集到的数据；搭载 LuatOS-iRTU 的系统级主控 DTU，由 LuatOS 模块直接和传感器通信，获取采集到的数据；可以省去一个 MCU 的硬件成本。

- 传统可商业化的 DTU 产品，售价基本在大几百元左右；而搭载 LuatOS-iRTU 的 DTU 产品，模组和外围器件的成本在 100 元左右，可以大幅提高 DTU 企业的盈利能力。

2. 业务功能扩充方便



```lua
--任务1：每隔5秒查询一次温湿度数据，然后通过网络通道1上报给服务器
function
    sys.wait(5000)
    local i2cId = 2

    --i2c初始化
    local init = i2c.setup(i2cId, i2c.SLOW)
    log.info("AHT10 init",init==i2c.SLOW)

    while true do
        --发送查询温湿度命令
        i2c.send(i2cId, 0x38, {0xAC, 0x22, 0x00})
```

```
        --读取温湿度数据
        local data = i2c.recv(i2cId, 0x38, 6)


        --解析温湿度数据
        local temprature,humanity = 0,0
        if #data == 6 then
            local _, _, data2, data3, data4, data5, data6 = pack.unpack(data,
"b6")
            humanity = bit.bor(bit.bor(bit.lshift(data2, 12),
bit.lshift(data3, 4)), bit.rshift(data4, 4))/ 1048576 * 10000
            log.info("AHT10 humanity", humanity/100 )
            temprature = bit.bor(bit.bor(bit.lshift(bit.band(data4, 0x0f),
16), bit.lshift(data5, 8)), data6) / 1048576 * 20000 - 5000
            log.info("AHT10 temprature", temprature/100)
        end


        --温湿度数据通过网络通道1发送到服务器
        sys.publish("NET_SENT_RDY_1","温度："..(temprature/100).."，湿度：
"..(humanity/100))


        --间隔5秒后，再次查询
        sys.wait(5000)
    endend
```

```
--任务2：解析网络通道1，服务器下发的数据，然后驱动LCD显示数据内容
function
    local function init()
        local para =
        {
            width = 128, --分辨率宽度，128像素；用户根据屏的参数自行修改
            height = 160, --分辨率高度，160像素；用户根据屏的参数自行修改
            bpp = 16, --位深度，彩屏仅支持16位
            bus = disp.BUS_SPI4LINE, --LCD专用SPI引脚接口，不可修改
            xoffset = 2, --X轴偏移
            yoffset = 1, --Y轴偏移
            freq = 26000000, --spi时钟频率，支持110K到13M（即110000到13000000）之间的整数（包含110000和13000000）
            pinrst = pio.P0_6, --reset，复位引脚
            pinrs = pio.P0_1, --rs，命令/数据选择引脚
            --初始化命令
            --前两个字节表示类型：0001表示延时，0000或者0002表示命令，0003表示数据
            --延时类型：后两个字节表示延时时间（单位毫秒）
            --命令类型：后两个字节命令的值
            --数据类型：后两个字节数据的值
            initcmd =
            {
                0x00020011,
```

```
            0x00010078,
            --0x00020021, -- 反显
            0x000200B1,
            0x00030002,
            0x00030035,
            0x00030036,
            0x000200B2,
            0x00030002,
            0x00030035,
            0x00030036,
            0x000200B3,
            0x00030002,
            0x00030035,
            0x00030036,
            0x00030002,
            0x00030035,
            0x00030036,
            0x000200B4,
            0x00030007,
            0x000200C0,
            0x000300A2,
            0x00030002,
            0x00030084,
```

```
            0x000200C1,
            0x000300C5,
            0x000200C2,
            0x0003000A,
            0x00030000,
            0x000200C3,
            0x0003008A,
            0x0003002A,
            0x000200C4,
            0x0003008A,
            0x000300EE,
            0x000200C5,
            0x0003000E,
            0x00020036,
            0x000300C0,
            0x000200E0,
            0x00030012,
            0x0003001C,
            0x00030010,
            0x00030018,
            0x00030033,
            0x0003002C,
            0x00030025,
```

```
    0x00030028,
    0x00030028,
    0x00030027,
    0x0003002F,
    0x0003003C,
    0x00030000,
    0x00030003,
    0x00030003,
    0x00030010,
    0x000200E1,
    0x00030012,
    0x0003001C,
    0x00030010,
    0x00030018,
    0x0003002D,
    0x00030028,
    0x00030023,
    0x00030028,
    0x00030028,
    0x00030026,
    0x0003002F,
    0x0003003B,
    0x00030000,
```

```
            0x00030003,
            0x00030003,
            0x00030010,
            0x0002003A,
            0x00030005,
            0x00020029,
        },
        --休眠命令
        sleepcmd = {
            0x00020010,
        },
        --唤醒命令
        wakecmd = {
            0x00020011,
        }
    }
    disp.init(para)
    disp.clear()
    disp.update()
end

--控制SPI 引脚的电压域
pmd.ldoset(15,pmd.LDO_VLCD)
```

```
init()



--订阅网络通道 1 下发的数据处理函数

--将收到的网络通道数据，通过 LCD 显示出来

sys.subscribe("NET_RECV_WAIT_1",function(id,data)

    log.info("NET_RECV_WAIT_1",id,data)

    disp.clear()

    disp.puttext(common.utf8ToGb2312(data),0,0)

    disp.update()

end)end
```

任务-脚本代码.lua

如果要支持一款新增的采集设备或者控制设备：

- 传统 DTU，配置功能中不支持编程，基本上都要修改 DTU 固件，然后本地烧录或者远程升级固件，执行难度较大；
- 搭载 LuatOS-iRTU 的系统级主控 DTU，配置功能中包含一个强大的任务配置项，支持脚本编程，在不修改 DTU 固件的前提下，可以在任务中添加脚本代码配置；DTU 下载最新配置参数，就可以轻松支持。

3. 脚本源码免费开放

LuatOS-iRTU 的系统级主控 DTU 软件解决方案源码全部开放。

代码仓库地址：https://gitee.com/openLuat/luatos-irtu

用户可以在源码的基础上进行二次开发，根据自己的需求进行高度定制化，为未来 DTU 的发展赋予更多的可能性。

四、技术支持体系

- gitee 代码：https://gitee.com/openLuat/luatos-irtu
- 发行版固件：https://gitee.com/openLuat/luatos-irtu/releases
- 文档教程：https://gitee.com/shengxia666/application-manual-irtu
- 视频教程：https://www.bilibili.com/video/av41012302

- 社区支持：https://doc.openluat.com
- 企业微信小群：合宙 FAE 和客户之间进行技术沟通的核心渠道；客户采用合宙模块，即建立由销售、FAE 及 CPM 组成的支持团队在内的支持小群，永不解散。
- 现场支持：如果线上方式无法有效解决问题，合宙将毫不犹豫的通过现场沟通的方式，直至问题解决。

五、开发什么难度

功能部分	开发周期	基本配置用户	基本配置+数据流+任务 用户	源码基础上自定义功能二次开发用户
基本配置功能的使用	1小时到1天	☑	☑	☑
LuatOS-iRTU代码学习	1周到2周		☑	☑
音频播放业务	1到2天		根据自己的业务需求 自行选择第一列对应的功能部分	根据自己的业务需求 自行选择第一列对应的功能部分
蓝牙业务	1到5天			
GPS业务	1到2天			
GPIO、键盘阵列	1天			
I2C业务	1天			
SPI业务	1天			
UART业务	1到5天			
NVM、文件系统业务	1天			
短信、通话业务	1天			
基站WIFI定位、NTP时间同步	1天			
录音	1天			
远程升级	1天			
UI界面	1到5天			

12.3.2 固件版本及源码获取

gitee 代码：https://gitee.com/openLuat/luatos-irtu

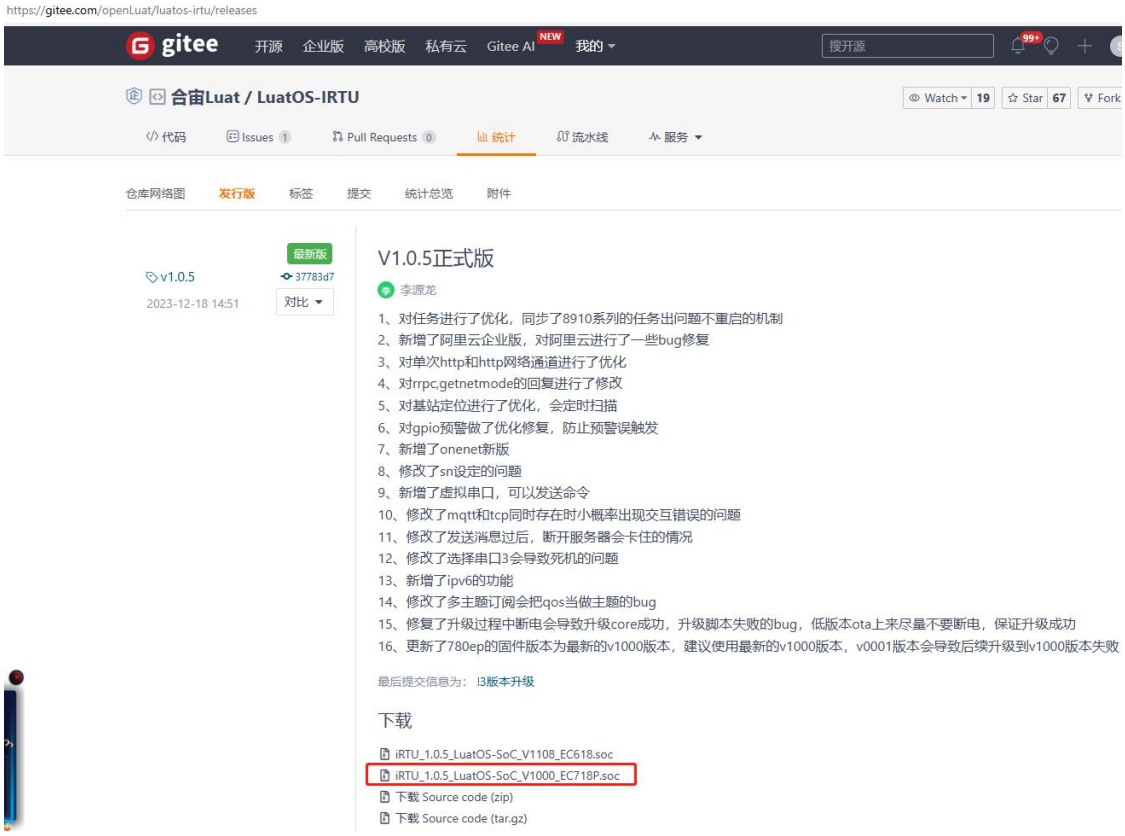
发行版固件：https://gitee.com/openLuat/luatos-irtu/releases

12.3.3 iRTU 新手教程

1、iRTU 固件的获取：

获取连接：https://gitee.com/openLuat/luatos-irtu/releases

下载最新版本的 soc 固件即可。

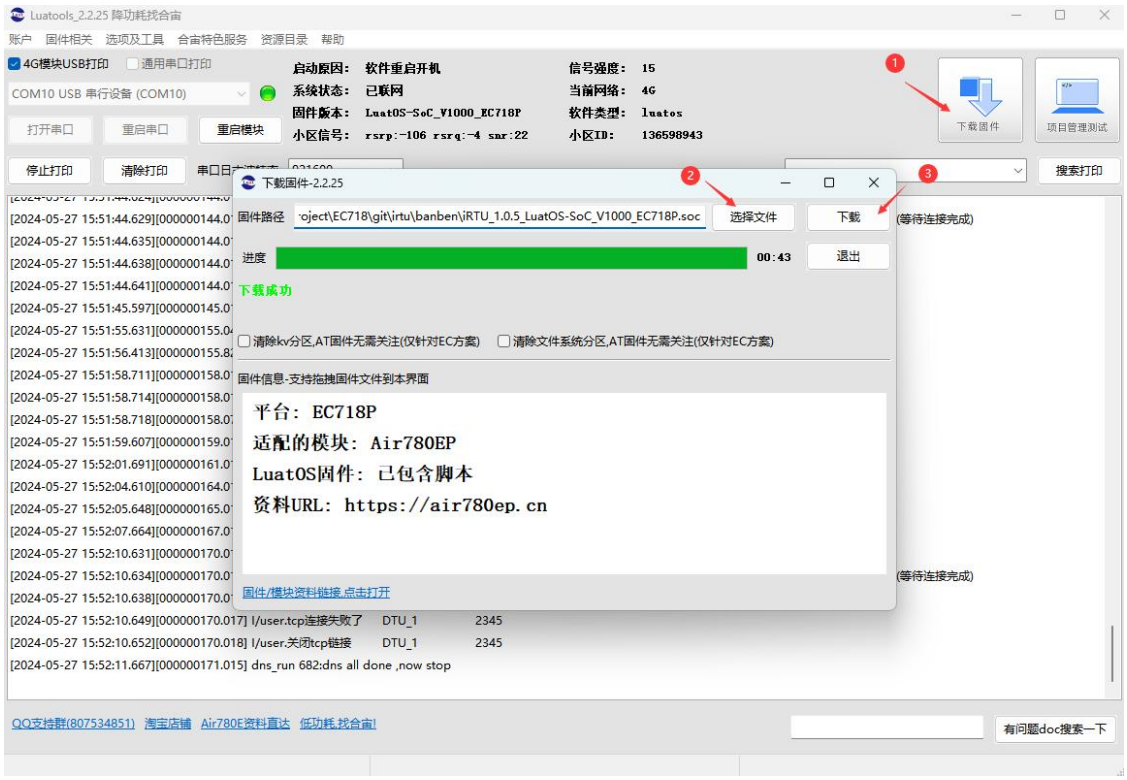


2、固件如何烧录：

下载烧录工具 luatools

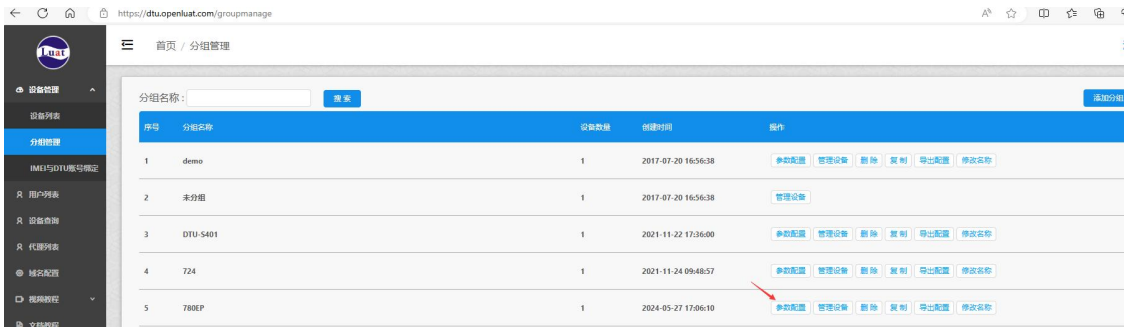
链接:https://doc.openluat.com/wiki/52?wiki_page_id=5071

点击下载固件



3、网页端配置：

登录 dtu.openluat.com 在分组管理里面添加分组，点击参数配置即可配置



模块联网上电自动获取配置。

配置获取前保证模块在分组下面，单个模块选择 IMEI 与 DTU 账号绑定，选择好分组获取专属固件，然后给模块烧录专属固件，然后再烧录 git 上面下载的固件即可获取配置正常使用。

12.3.4 iRTU 使用手册

DTU 固件参考手册 V3.9.pdf

12.3.5 iRTU 使用实例

实例

12.3.5.1 GPIO

GPIO 设置分为三种设置

NETLED:此项可设置网络指示灯对应的 GPIO 引脚，用户根据实际需求，自行选择。

NETRDY:此项可设置 网络准备通知对应的 GPIO 引脚，用户根据实际需求，自行选择。

```
LED 闪烁规则如下：100ms 闪烁 —— 注册 GSM500ms 闪烁 —— 附着 GPRS100ms 亮，1900ms 灭 —— 已连接到服务器
```

RSTCNF:此项可设置硬件上**重置 iRTU 参数**对应的 GPIO 引脚，实际使用时，将该 GPIO 引脚拉低进行**重置 iRTU 参数**操作，用户根据实际需求，自行选择。

12.3.5.2 GPS

启用 GPS 功能

GPS 功能目前支持 Air530/800 /801/868/820 以及相应的开发板。

启用 GPS 功能会占用 1 个串口，另外一个串口依旧可以配置 DTU 的各种功能

GPS 报文 JSON 定义

信息 = [是否有效,时间戳,经度,纬度,海拔,方位角,速度,载噪比,定位卫星]

设备 = [是否打开，震动，开锁，点火，充电，剪线，外电电压，电池电压，GPRS 信号值]

```
{"msg":[true,1547272725,1136037366,348581216,114,354,1,43,4]}
{"sta":[true,false,false,false,false,false,65535,4101,11]}
```

GPS 报文 HEX 定义（报文长度固定大端）

msg 报文 0xAA 开头，设备信息报文 0x55 开头

[信息 | 是否有效|时间戳|经度|纬度|海拔|方位角|速度|载噪比|定位卫星]

[1B | 1B| 4B | 4B| 4B |2B | 2B | 1B | 1B | 1B] = 21 Byte

```
[AA 01 5C399812 43B68DF6 14C6ED60 0016 001E 0F 32 0E]
```

[设备 |是否打开|震动|开锁|点火|充电|剪线|外电电压|电池电压|GPRS 信号值]

[1B | 1B | 1B | 1B| 1B | 1B| 1B |4B |2B |1B] = 13 byte

```
[55 00 00 00 00 00 00 00006A10 1011 15]
```

GPS 的 GPIO 配置

demo: "config, gps, pio, pio8 ,pio9, pio10, 0, 16"

字段	值	含义
config	config	配置指令标识
id	gps	通道编号gps表示配置gps
type	pio	GPS 的配置 GPIO 的标志位
netdrdy	pio0-pio128	GPS 定位成功指示灯 GPIO 编号，例如 GPIO_3 就填 pio3，下同
vib	pio0-pio128	振动传感器信号输入GPIO编号
acc	pio0-pio128	ACC开锁信号输入GPIO编号注意ACC信号电压很高要用分压电阻
chg	pio0-pio128	内置锂电池充电状态输入GPIO编号,可为空
adc	0 or 1	检测VCC的ADC通道编号，默认0
ratio	1-50	VCC/1.8V + 1的值，是值ADC采集电压和VCC实际电压的比值

GPS 的报文配置

demo: "config, gps, fun, 2, 115200, 0, 5, 1, json, 100, ; ,60"

字段	值	含义
config	config	配置指令标识
id	gps	通道编号gps表示配置gps
type	fun	GPS 的配置 fun参数 的标志位
串口ID	1 or 2	GPS 启用的串口波特率，注意不要和socket通道同时启用
波特率	115200	9600-115200，注意和模块的GPS实际波特率一致
功耗模式	0 or 2 or 8	GPS工作模式: 正常，低功耗，低功耗跟踪模式
采集间隔	1-600	单位秒，多久采集并记录一次GPS报文
采集方式	0 or 1	0 布防采集（触发上报），1持续采集
报文格式	json or hex	设备信息和GPS定位信息上报的报文格式，支持JSON和HEX
缓冲发送	0 - 1000	0 表示不缓冲，采集后立刻上报，其他表示先加入缓冲区，然后1次上报
分隔符	除,外的可见字符	启用缓冲发送，用来分割报文的标记
状态间隔	0-1440	单位分钟，设备信息报文上报频率，0为不上报
上报通道	上报网络通道	捆绑用于GPS上报的网络通道id,串口和GPS可以公用1个socket/mqtt/http 通道

12.3.5.3 HTTP 透传

操作步骤：

1. 打开 DTU 配置界面，选择**网络通道参数**，打开任意通道，通道类型选择 **HTTP**，绑定的串口 ID 根据用户实际需求进行选择。

2. 选择提交请求的方法

   2.1 **method**：有 get 和 post 两种类型供用户选择，默认为 post

   2.2 **url** ：HTTP 请求的地址和参数

   2.3 **超时时间**：（HTTP 请求最长等待时间，超过这个时间，HTTP 将返回，默认为 30S）

   2.4 **请求类型**：用户通过串口发送给服务器的数据的数据类型

   2.5 **type**：body 的提交类型，有 urlencode(普通字符串)、json(json 字符串)、stream(二进制文本，图片、音频文件等)

i.

3. 通过串口发送 body，即可获得 HTTP 的 server 的回应。

日志打印： -[---------- 网络注册已成功 ----------]
-[---------- HTTP is start! ----------]

截图：

```
[2021-11-13 16:18:00.216] [I]-[ril.proatc] +CSQ: 24,99
[2021-11-13 16:18:00.216] [I]-[ril.proatc] OK
[2021-11-13 16:18:01.217] [E]-[uart.read--->] 1 1
[2021-11-13 16:18:01.217] [I]-[UART_1 read:] 16 7072696E74282268656C6C6F22290D0A
[2021-11-13 16:18:01.217] [I]-[串口流量统计值:] 16
[2021-11-13 16:18:01.217] [I]-[user header:]
[2021-11-13 16:18:01.248] [I]-[socket:connect-coreid,prot,addr,port,cert,timeout] 0 TCPSSL www.baidu.com 443 nil 120
[2021-11-13 16:18:01.590] [I]-[打印占用的内存:] 704.9599609375
[2021-11-13 16:18:01.590] [I]-[打印可用的空间] 1296000
[2021-11-13 16:18:01.808] [I]-[socket:on_response:] 0 SOCKET_CONNECT 0
[2021-11-13 16:18:01.808] [I]-[socket:connect: connect ok]
[2021-11-13 16:18:01.808] [I]-[发送的http报文:] POST / HTTP/1.0
[2021-11-13 16:18:01.808] Connection: close
[2021-11-13 16:18:01.808] Content-Type: application/x-www-form-urlencoded
[2021-11-13 16:18:01.808] Accept: */*
[2021-11-13 16:18:01.808] User-Agent: Mozilla/4.0
[2021-11-13 16:18:01.808] Content-Length: 16
[2021-11-13 16:18:01.808] Accept-Language: zh-CN,zh,cn
[2021-11-13 16:18:01.808] Host: www.baidu.com
[2021-11-13 16:18:01.808] print("hello")
[2021-11-13 16:18:01.808] [D]-[socket.send] total 214 bytes first 30 bytes POST / HTTP/1.0
[2021-11-13 16:18:01.808] Connection: c
[2021-11-13 16:18:01.886] [I]-[socket:on_response:] 0 SOCKET_SEND 0
[2021-11-13 16:18:01.886] [D]-[socket.recv] 488 nil
```

```
[2021/11/13 16:18:01.1492] ← print("hello")

[2021/11/13 16:18:01.9810] → 302
Content-Length: 154
Traceid: 163679148028491701861093818744108369258l
Date: Sat, 13 Nov 2021 08:18:00 GMT
Set-Cookie: BDSVRTM=0; path=/
X-Frame-Options: sameorigin
X-Ua-Compatible: IE=Edge,chrome=1
Location: https://www.baidu.com/search/error.html
Server: BWS/1.1
Content-Type: text/html
Bdpagetype: 3
<html>
<head><title>302 Found</title></head>
<body bgcolor="white">
<center><h1>302 Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

注意事项

1. HTTP 是无状态的短链接，服务器无法主动给模块下发，由模块主动给 HTTP 的 Server 发消息，而 HTTP 的 Server 被动的返回一个参数。

2. URL 这里，如果是 https 的话，一定要仔细写好，默认添加 SSL，若不填的话就是默认的 http 请求。

12.3.5.4 MQTT 透传

操作步骤：

1. 基本参数配置前面已有所介绍，在这重点讲一下网络通道参数里的几个重要参数：**MQTT 心跳包的间隔**默认为 300 秒(保活)

2. **MQTT 的地址或域名**，以及 **MQTT 服务器的端口号**，为了方便测试，我们提供了一个免费的 MQTT 服务器供用户测试，无账号和密码限制，实际项目中，用户根据自己的 MQTT 服务器设置进行选择填写。（MQTT 服务器的地址为 lbsmqtt.airm2m.com，端口号为 1884）

i.

ii.

3. **MQTT 保存会话标志位**：根据用户实际需求选持久会话或离线自动销毁

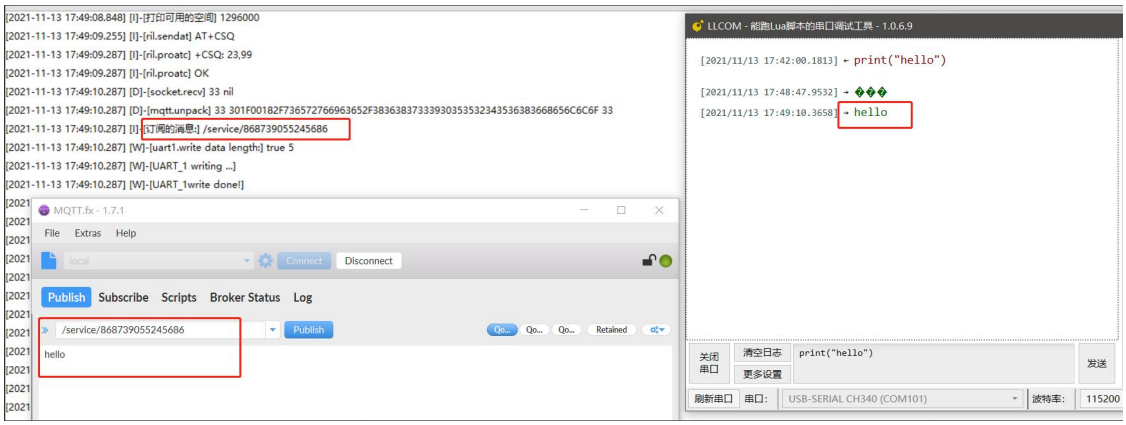
4. **消息订阅与发布**：本文中为了演示方便，选择了\device\和\service\，实际项目中根据用户实际需求设置

5. **主题添加 IMEI**：为了区分发布的数据时从哪台设备中来的，可以勾选**主题添加 IMEI**，如果客户端只想订阅某一模块发布的消息，可以在 MQTT 客户端中订阅如下主题：**/自定义主题/IMEI**，默认主题添加 IMEI，用户根据实际需求自行选择。

6. 配置成功后点击确定，重启设备，打开 MQTT 工具（这里使用的是 MQTT.FX），连接上对应的地址，配置 MQTT 的订阅主题为 **/用户自定义主题/#**,配置 MQTT 发布主题为 **/用户自定义主题/IMEI**(注：#为订阅上级主题发布的所有消息，通常用于调试过程)

日志打印： -[---------- 网络注册已成功 ----------] -[---------- MQTT is start! ----------] -[----- 发布的主题:] /device/imei -[订阅的消息:] /service/imei

截图：



12.3.5.5 TCP/UDP 透传

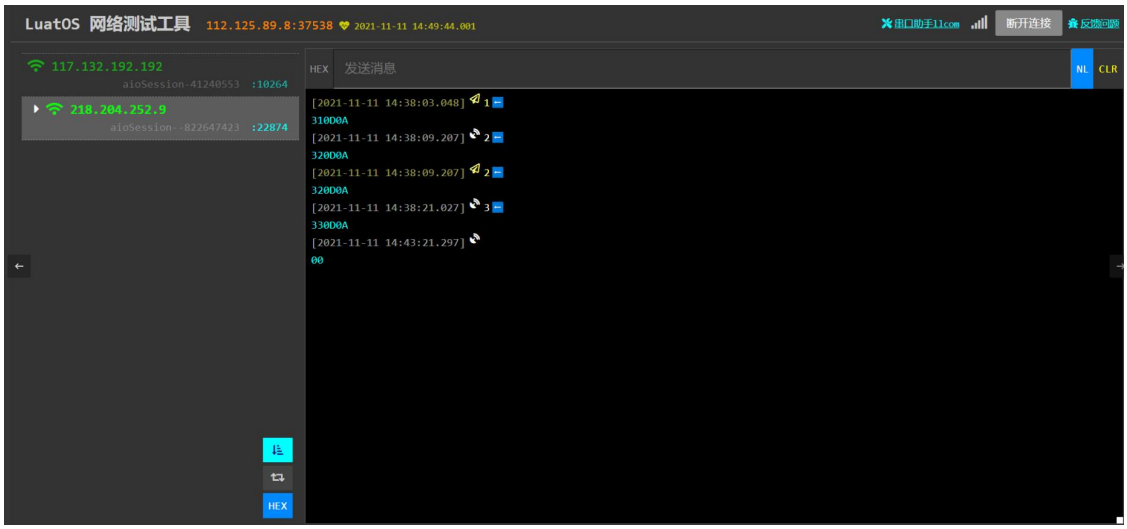
注：合宙提供了一个免费的 TCP/UDP 测试服务器，用户可用来测试 TCP/udp 服务器的操作。https://netlab.luatos.com/

操作步骤：

1. DTU 管理系统： DTU 管理系统->设备管理->分组管理->添加分组 添加分组后点击查看，通过绑定 IMEI 号来绑定设备。
2. 打开 DTU 管理系统，基本参数配置前面已经有所介绍，这里重点介绍网络通道参数：

选择 socket 通道、TCP 协议，心跳包可以自定义，图中是选择了自定义，每隔一段时间会向 TCP 服务器发送 IMEI 号，下面的心跳间隔时间与此对应。 接着填上 TCP 服务器的地址以及端口号，**TCP 通道捆绑的串口 ID 即为基本参数配置时开启的串口通道**，要一一对应。 最后点击确定就完成了 DTU 参数配置。

3. 给板子烧录 DTU 固件，打开串口工具，以及 TCP 服务器页面，可以看到，串口能发送数据到 TCP 服务器端，TCP 服务器也能下发数据到模块串口，而且模块会每隔一分钟 向 TCP 服务器发送 0X00，如图所示



日志打印： -[---------- 网络注册已成功 ----------]

-[---------- TCP/UDP is start! ----------]

注意事项：

注意添加设备时确保 IMEI 的正确，否则设备无法绑定 DTU，相应的功能就无法实现。

12.3.5.6 任务

基础概念介绍

任务在脚本层可以解释为一个 task，当用户开启一个**任务**等同与在脚本层新建了一个 task，所以在用户启用**任务**功能之前，建议用户先熟悉 luat 的基础语法，点击此处即可进行 luat 基础语法的学习。

使用方法

任务的使用需要谨慎，按照如下固定格式来写

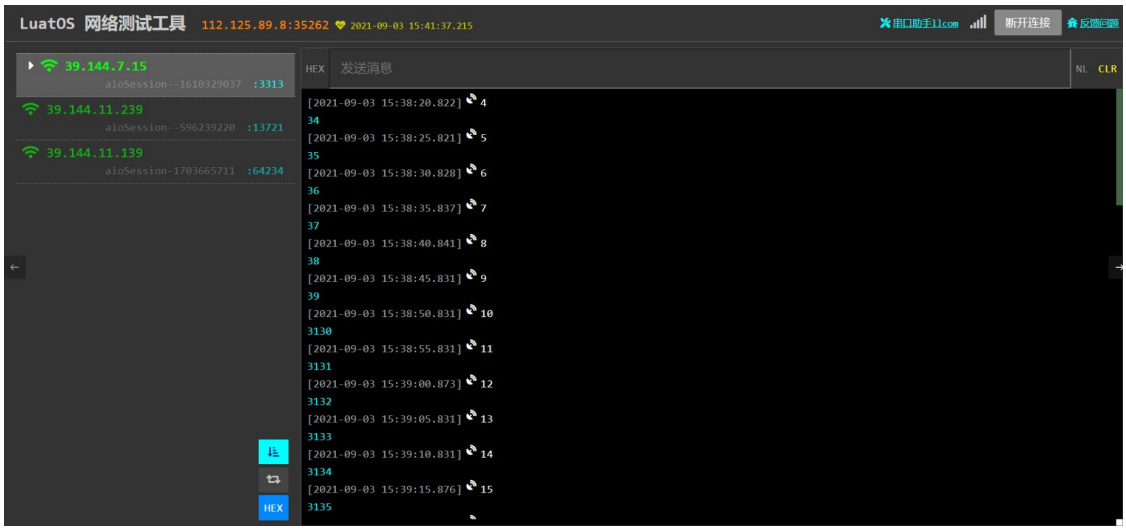
```
function

--用户自定义代码
```

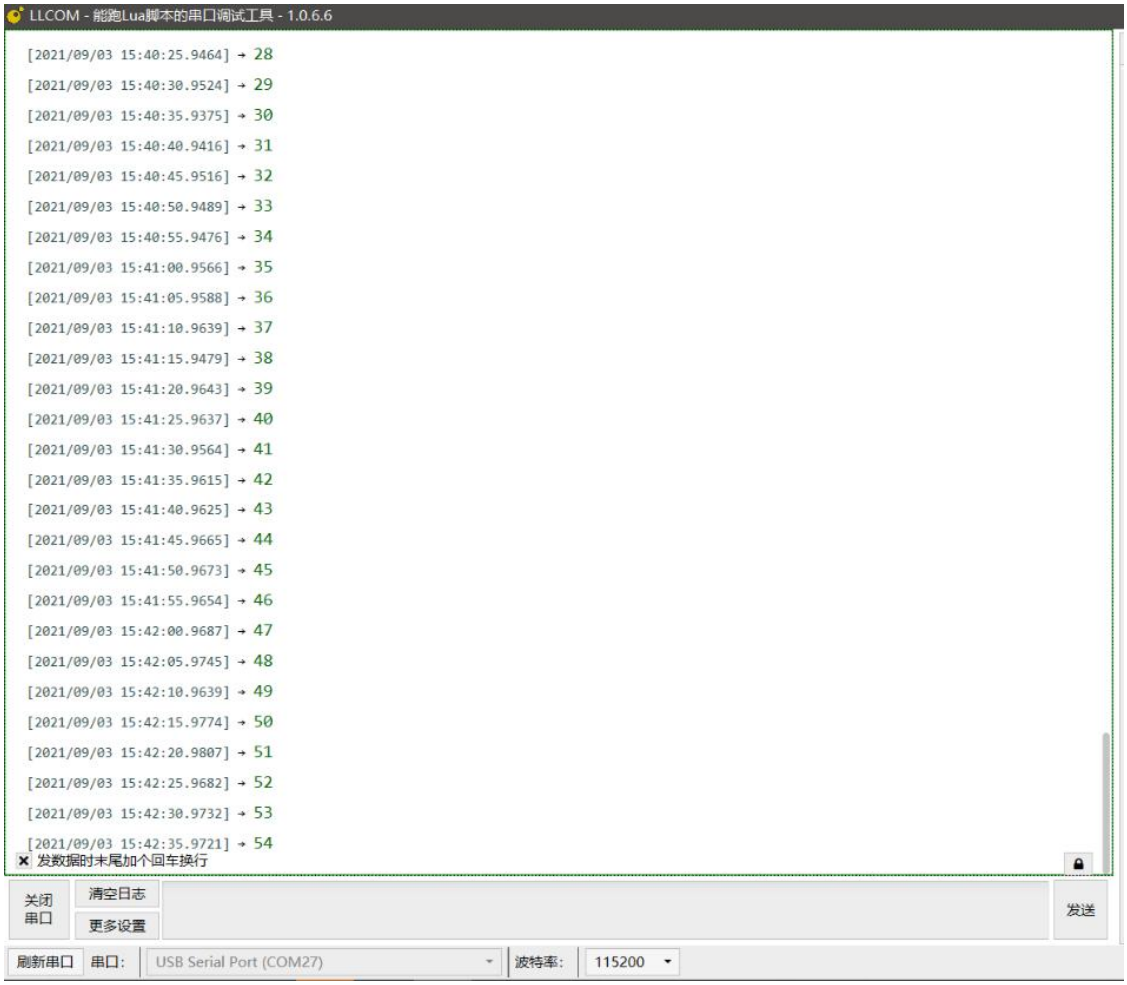
```
end
```

例如

```
function
--定义一个叫做 count 的变量，并使它的初始值为 0
local count=0
--进入一个循环
while true do
    --发布一个消息给串口 1，输出 count 变量的值
    --如需发布消息给其他串口，仅需把下文中的 NET_RECV_WAIT_1 中的 1 和后面逗号的 1 换成其他串口号
    sys.publish("NET_RECV_WAIT_1",1,tostring(count))
    --令 count 的值自增 1
    count = count +1
    --发布一个消息到网络通道 1，同样输出 count 变量的值
    --如需发送到其他网络通道，仅需把下文中的 ET_SENT_RDY_1 中的 1，换成其他网络通道号
     sys.publish("NET_SENT_RDY_1",tostring(count))
    --延时 5s
    sys.wait(5000)
    end
end
```

最终效果如下图所示





在**任务**里可以完成一些拓展功能，比如**免 MCU** 使用模块的 iic 接口去读取并处理传感器数据，就可以在**任务**里处理（建议将传感器驱动放在 iic.lua 文件里）。例如：客户在 iic.lua 中写了一个叫 sht30 的函数，并 return 了返回值，我们就可以在**任务**中去读取到返回值，**任务**代码如下所示**。

```
function
--定义一个叫做 temp 和 hump 的变量，并使它们的初始值为 0
    local temp=0
    local hump=0
```

```
    --进入一个循环
    while true do
        temp,hump = iic.sht30()
        --发布一个消息给串口，输出 temp 和 hump 的值
        sys.publish("NET_RECV_WAIT_1",tostring(temp),tostring(hump))
        --发布一个消息到服务器，同样输出 temp 和 hump 的值
        sys.publish("NET_SENT_RDY_1",tostring(temp),tostring(hump))
        --延时 5s
        sys.wait(5000)
    end
end
```

当传感器和模块连线正确且传感器和模块均在正常工作的时候，服务器和串口都能收到这条带有 temp 和 hump 的消息。 以上两个演示仅是**任务**的实例 demo，用户可根据自己的实际需要去写自己的**任务**。

12.3.5.7 数据流

数据流可以将串口接收到的数据进行处理（例如将串口接收到的数据进行加减某个数，给数据帧前后添加固定的校验码等）后发送给服务器，总共有七路数据流，对应七路网络通道。

操作步骤

此次以 mqtt 向多主题发送数据为例：

mqtt 多主题格式：

topic1;qos;topic2;qos...(每个参数之间必须加；最后一个参数后不需要加；)

1 2 3 4（每个参数对应一位，依次递增）

测试代码如下：

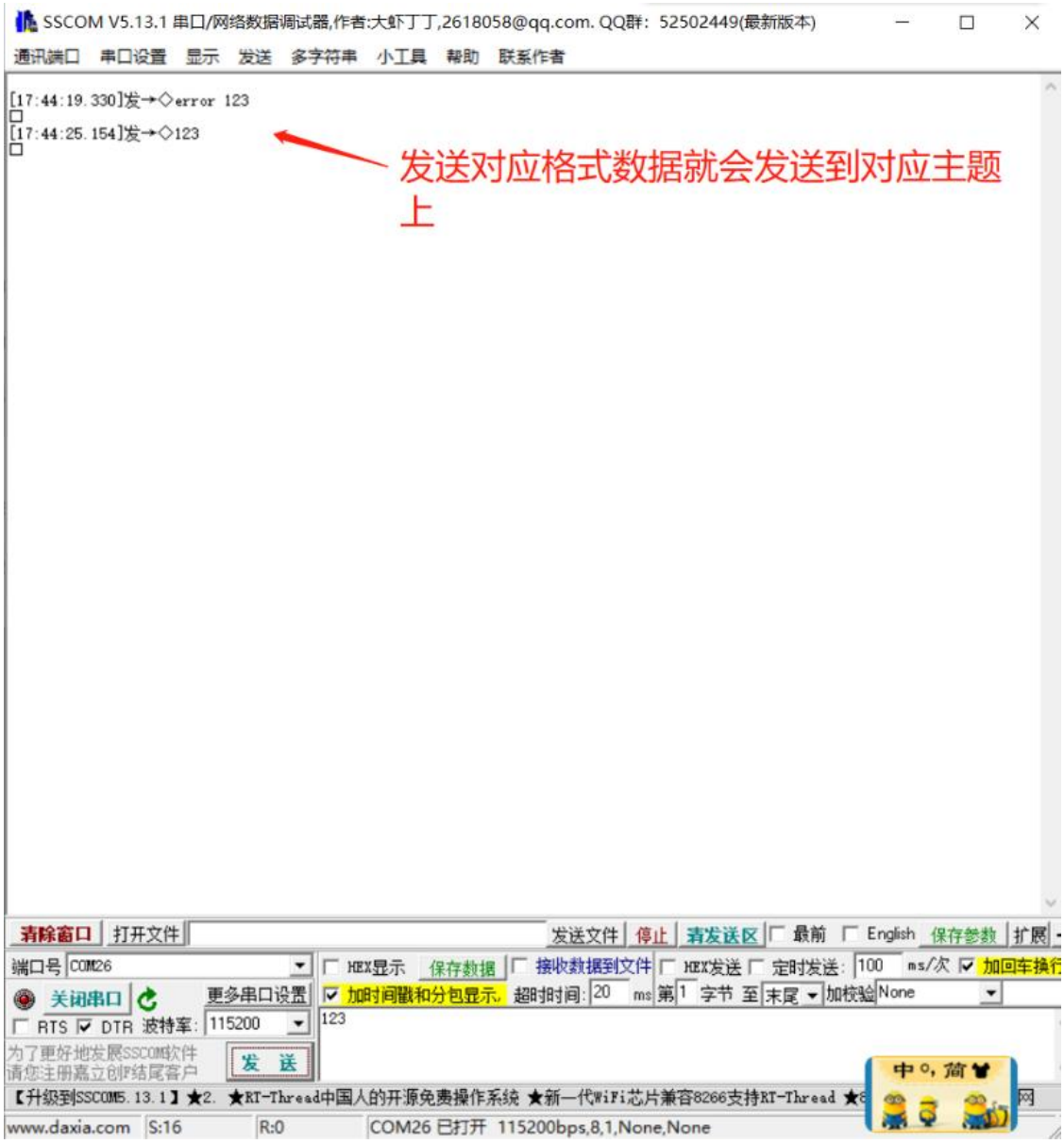
```
function   --dtu 固定格式（function  代码 end）
        local str=...--接收的字符串
```

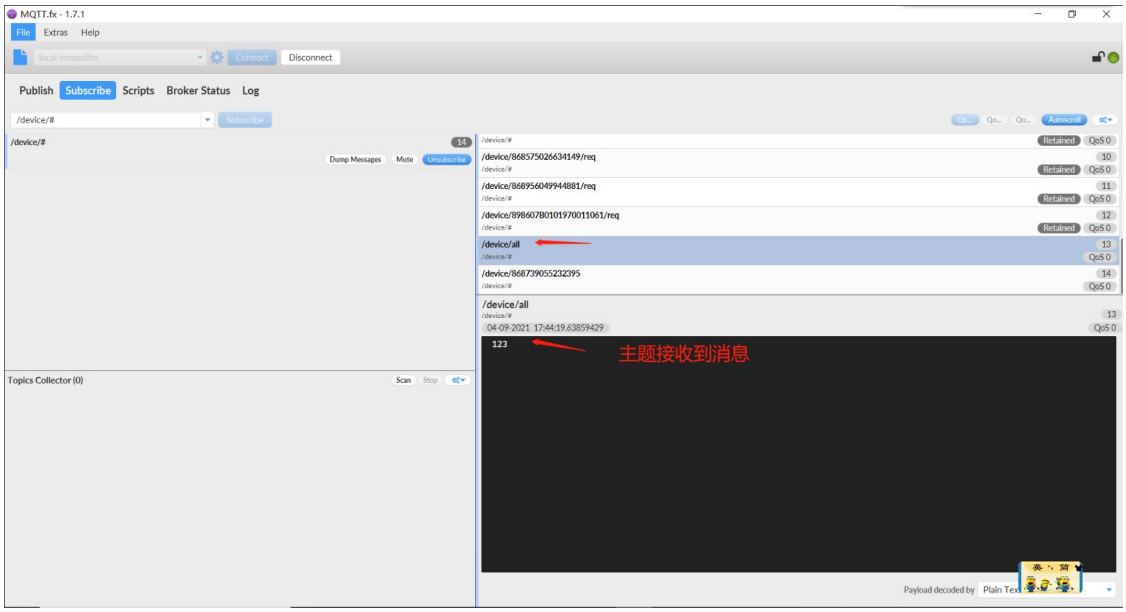
```
        if str:sub(1,5)=="error" then  --自定义格式
                return str:sub(6,-1),3  --返回值
        end
        return str,1  -- 返回值
end
```

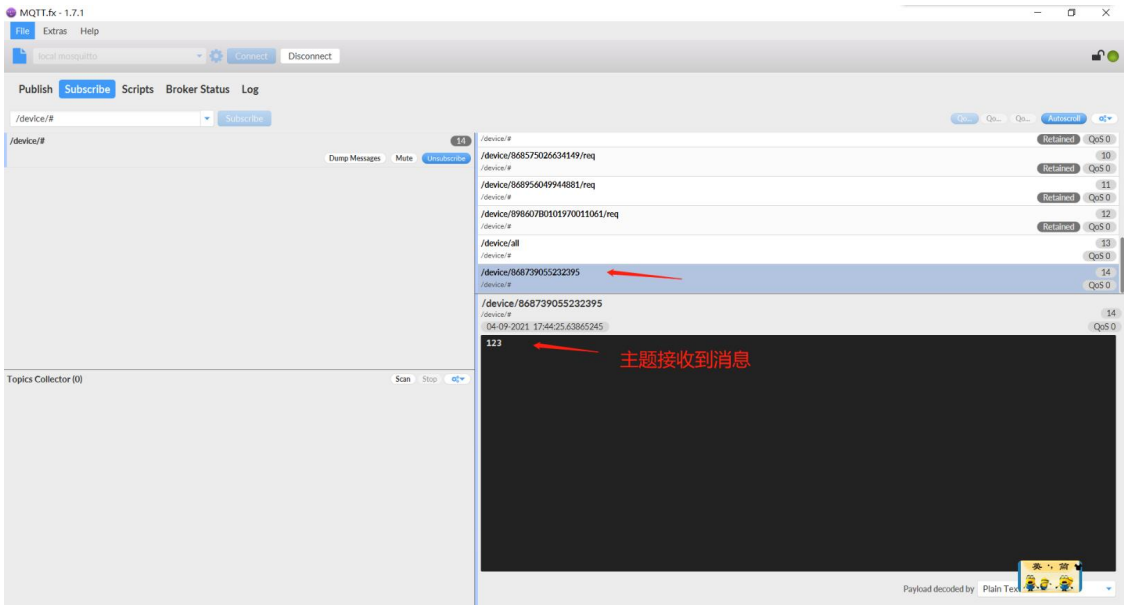
修改参数

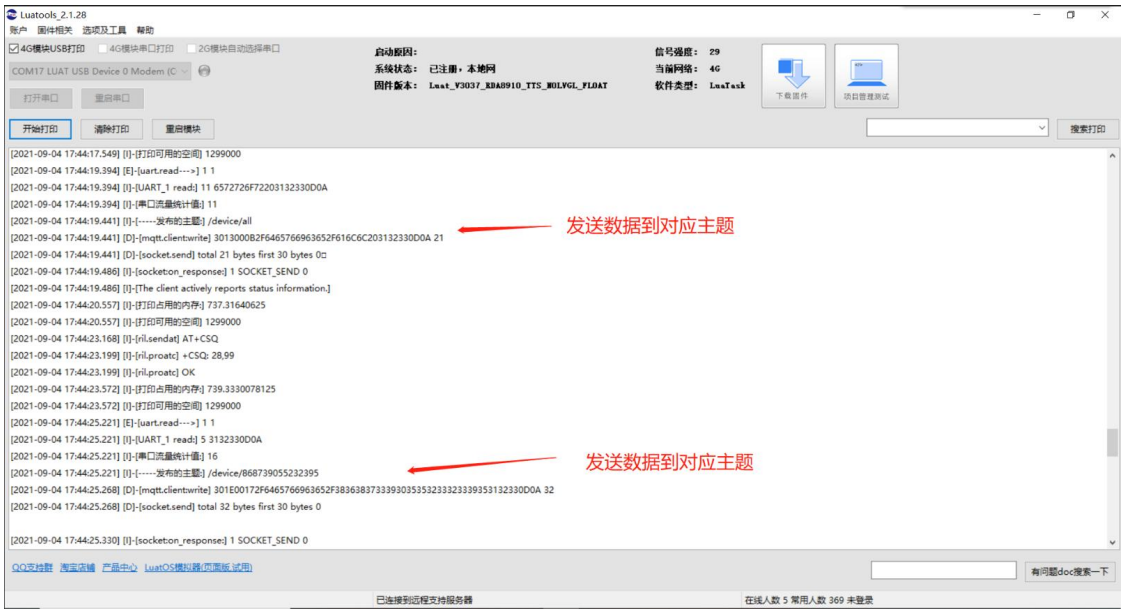
日志打印

重启模块后：


SSCOM V5.13.1 串口/网络数据调试器,作者:大虾丁丁,2618058@qq.com. QQ群: 52502449(最新版本)
通讯端口 串口设置 显示 发送 多字符串 小工具 帮助 联系作者
[17:44:19.330]发→◇error 123
[17:44:25.154]发→◇123
发送对应格式数据就会发送到对应主题上
清除窗口
打开文件
发送文件
停止
清发送区
最前
English
保存参数
扩展
端口号 COM26
HEX显示
保存数据
接收数据到文件
HEX发送
定时发送:
100
ms/次
加回车换行
关闭串口
更多串口设置
加时间戳和分包显示,
超时时间:
20
ms
第
1
字节
至
末尾
加校验
None
RTS
DTR
波特率:
115200
123
为了更好地发展SSCOM软件
请您注册嘉立创结尾客户
发 送
【升级到SSCOM5.13.1】★2. ★RT-Thread中国人的开源免费操作系统 ★新一代WiFi芯片兼容8266支持RT-Thread
www.daxia.com
S:16
R:0
COM26 已打开 115200bps,8,1,None,None


MQTT.fx - 1.7.1
File
Extras
Help
Connect
Disconnect
Publish
Subscribe
Scripts
Broker Status
Log
/device/#
Dump Messages
Mute
Unsubscribe
/device/868575026634149/req
/device/868956049944881/req
/device/898607B0101970011061/req
/device/all
/device/868739055232395
/device/all
04-09-2021 17:44:19.6859429
123
主题接收到消息
Topics Collector (0)
Scan
Stop
Payload decoded by



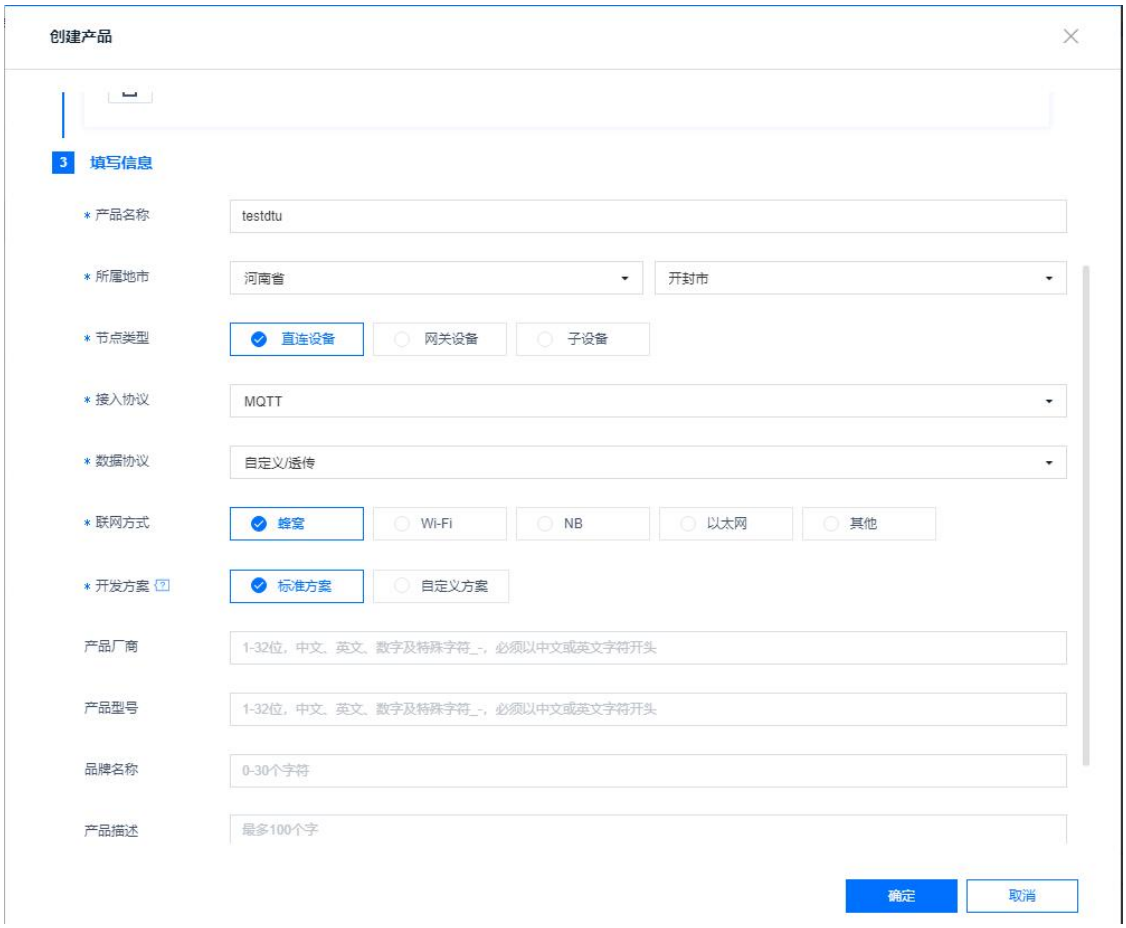


12.3.5.8 新版 onenet

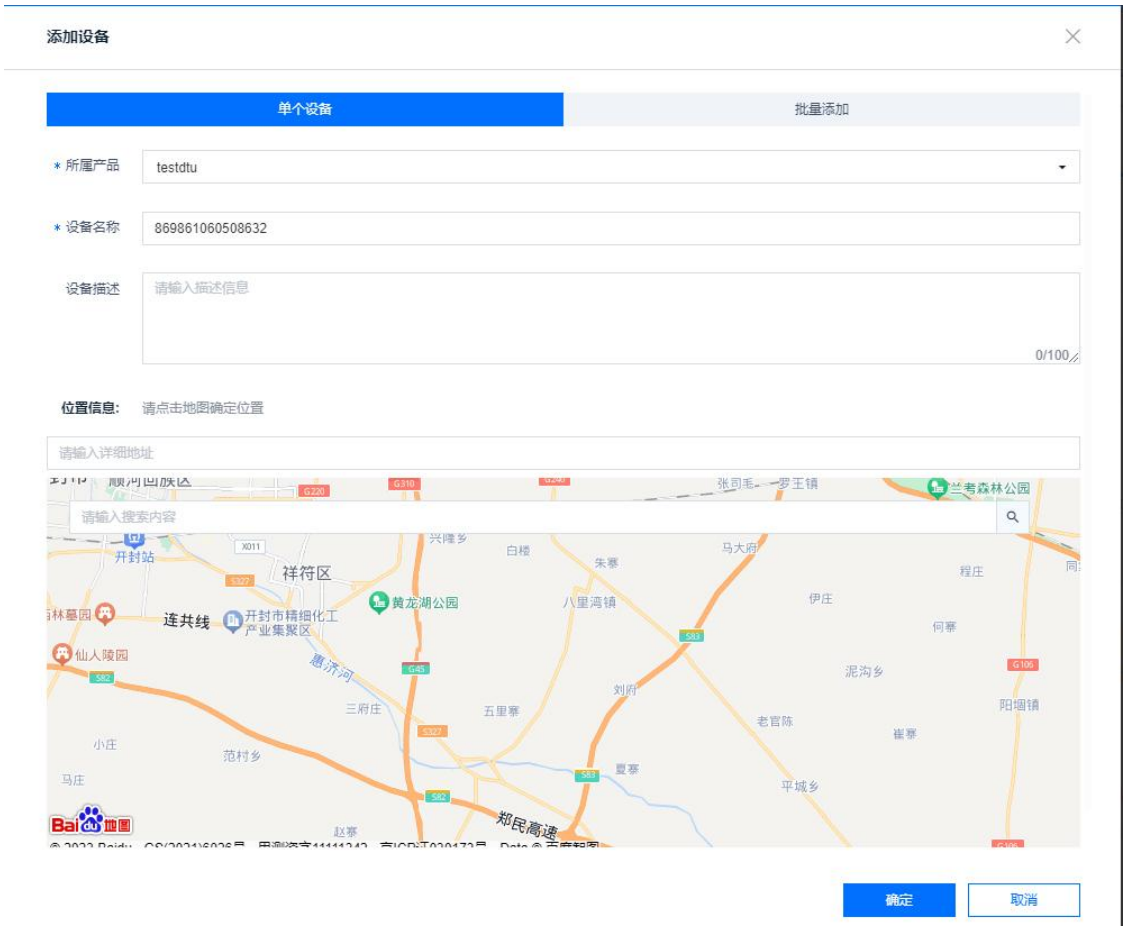
最新版 onenet 需要项目的 **ProductId**，**ProductSecret** 还有设备的 **ID**（默认为设备的 IMEI，也就是 **DeviceName**）

创建一型一密的项目

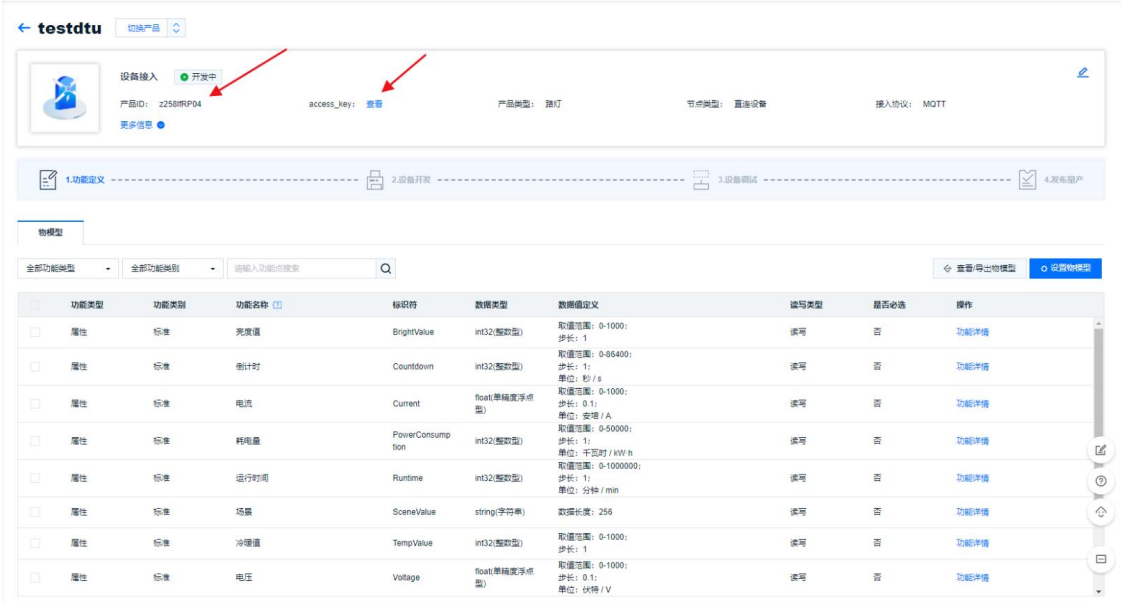
新建产品如下所示即可


创建产品
3 填写信息
* 产品名称
testdtu
* 所属地市
河南省
开封市
* 节点类型
直连设备
网关设备
子设备
* 接入协议
MQTT
* 数据协议
自定义/透传
* 联网方式
蜂窝
Wi-Fi
NB
以太网
其他
* 开发方案
标准方案
自定义方案
产品厂商
1-32位，中文、英文、数字及特殊字符_-，必须以中文或英文字符开头
产品型号
1-32位，中文、英文、数字及特殊字符_-，必须以中文或英文字符开头
品牌名称
0-30个字符
产品描述
最多100个字
确定
取消

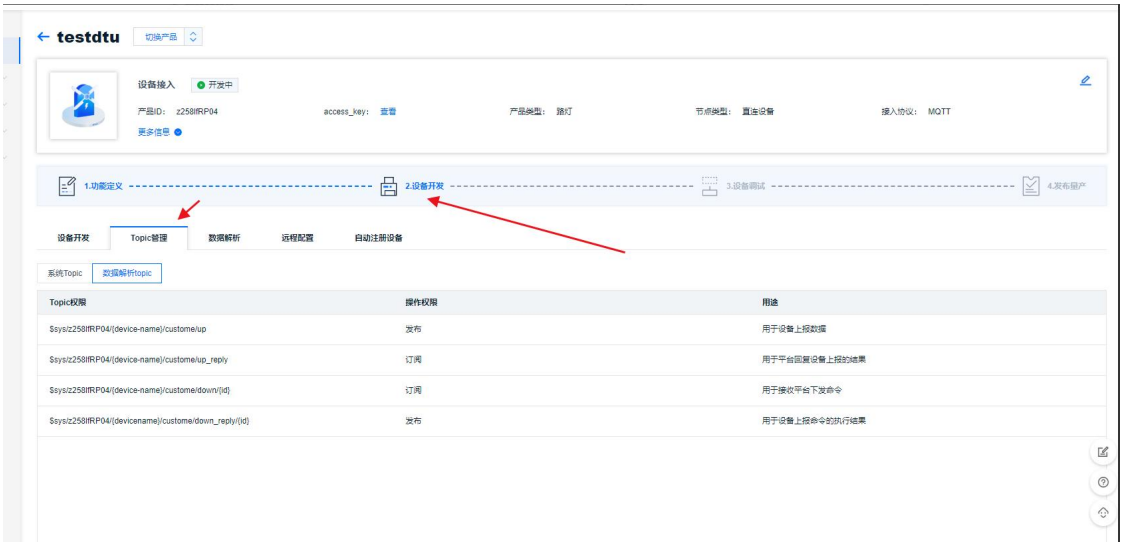
新建设备


添加设备
单个设备
批量添加
* 所属产品
testdtu
* 设备名称
869861060508632
设备描述
请输入描述信息
0/100
位置信息:
请点击地图确定位置
请输入详细地址
请输入搜索内容
祥符区
连共线
黄龙湖公园
开封市精细化工产业集聚区
仙人陵园
确定
取消

点击左侧产品开发选择产品可获得产品的产品 ID 和 access_key，因为 onenet 的生成三元组的方式想需要产品级/设备级的秘钥，类似于阿里云一型一密和一机一密的区别，dtu 配置的话大多数是产品级别，所以我们这边获取这两个值。



主题的获取方法在上面页面中，点击设备开发，topic 管理，即可获取 topic，在 web 配置的页面输入正确的 topic 才可以连接成功，否则会连接失败



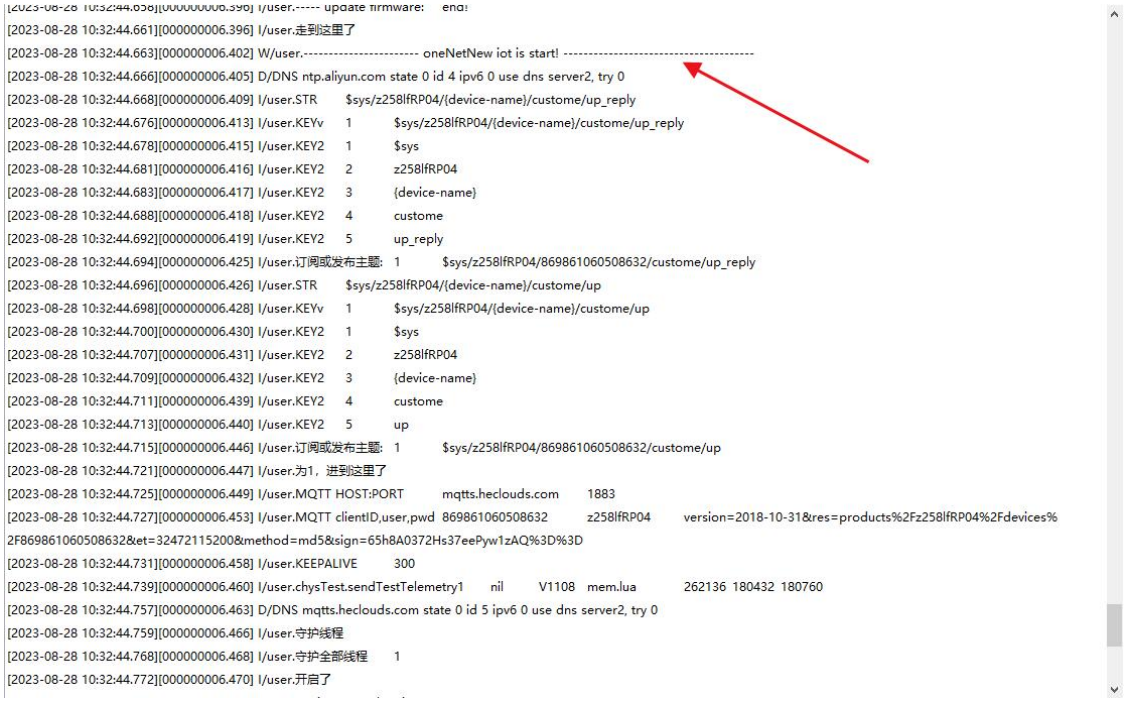
iRTU web 端配置

基本参数的配置和网络通道参数中个参数的含义与解释在前面的章节中有详细的说明，在这里不做过多讲解，用户根据自己情况进行选择即可。 onenet 最为重要的两个参数 **ProductID** 和 **ProductSecret** 前面讲过怎么获取了，将这两个参数填入对应位置即可。需要注意的是主题需要填写正确。

单击确定，即可完成所有配置。

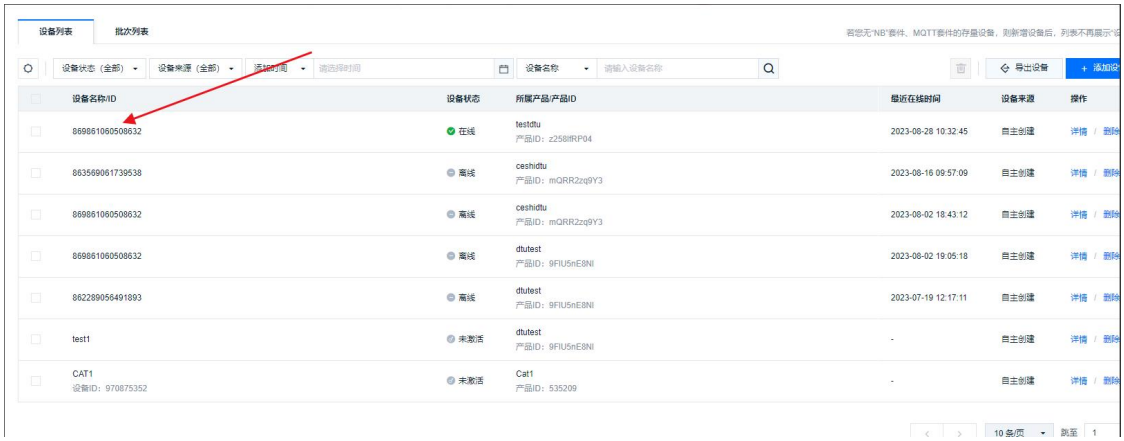
接下来给模块上电/重启，在设备上电情况下也可以通过串口/其他服务器端给模块发送 rrpc,upconfig 命令（串口端还可以发送+++命令），让设备读取到最新的参数配置即可。

当模块检测到参数配置有更新的情况下，会去跟新自己的参数，然后重启设备。重启之后就会去链接 onenet，如果这时你的设备开启了日志打印且设备直连电脑，luatools 打开且日志输出正常的情况下，会观察到 luatools 打印了下面两张截图的红框部分，如果都打印了，即证明你的设备连接 onenet 成功。



```
[2023-08-28 10:32:44.731][000000006.458] I/user.KEEPALIVE 300
[2023-08-28 10:32:44.739][000000006.460] I/user.chysTest.sendTestTelemetry1 nil V1108 mem.lua 262136 180432 180760
[2023-08-28 10:32:44.757][000000006.463] D/DNS mqtts.heclouds.com state 0 id 5 ipv6 0 use dns server2, try 0
[2023-08-28 10:32:44.759][000000006.466] I/user.守护线程
[2023-08-28 10:32:44.768][000000006.468] I/user.守护全部线程 1
[2023-08-28 10:32:44.772][000000006.470] I/user.开启了
[2023-08-28 10:32:44.791][000000006.481] D/user.rssi,mcc,mnc,lac,ci 31 460 0 18511 192756611
[2023-08-28 10:32:44.794][000000006.483] D/user.reqStr
207A314F6F44664150324C44744F5374694D5154564466584F36526B72576542472C688916605080 36F22032303233303533313131313231364135363534363941313331323230323235
3401484F01CC001F0B7D3B8300 174
[2023-08-28 10:32:44.807][000000006.487] D/socket connect to bs.openluat.com,12411
[2023-08-28 10:32:44.809][000000006.488] D/DNS bs.openluat.com state 0 id 6 ipv6 0 use dns server2, try 0
[2023-08-28 10:32:44.826][000000006.564] D/sntp Unix timestamp:1693189965
[2023-08-28 10:32:44.858][000000006.584] I/DNS dns all done ,now stop
[2023-08-28 10:32:44.906][000000006.638] D/user.lbsLoc receive 00307489502411340294561708 1C0A202D 34
[2023-08-28 10:32:44.911][000000006.644] I/user.定位类型,基站定位成功返回0 0
[2023-08-28 10:32:44.914][000000006.645] I/user.基站定位请求的结果: 034.7980542 114.3204965
[2023-08-28 10:32:44.984][000000006.725] I/user.mqtt event conack MQTTCTRL*: 004505C0 nil nil
[2023-08-28 10:32:44.991][000000006.727] I/user.mqtt连接成功
[2023-08-28 10:32:44.994][000000006.730] I/user.mqtt订阅成功
[2023-08-28 10:32:44.996][000000006.738] I/user.mqtt event sent MQTTCTRL*: 004505C0 0 nil
[2023-08-28 10:32:44.998][000000006.739] I/user.mqtt sent pkgid 0
[2023-08-28 10:32:55.289][000000016.998] D/mobile CSCON 0
[2023-08-28 10:32:59.843][000000021.570] D/mobile CSCON 1
[2023-08-28 10:32:59.921][000000021.654] D/mobile CSCON 0
[2023-08-28 10:33:13.559][000000035.286] I/user.libnet + csq 27 -59 -87 -8
[2023-08-28 10:33:43.556][000000065.286] I/user.libnet + csq 27 -59 -87 -8
[2023-08-28 10:34:13.554][000000095.286] I/user.libnet + csq 27 -59 -88 -9
[2023-08-28 10:34:43.553][000000125.286] I/user.libnet + csq 27 -59 -88 -9
```

当然，你也可以在自己的 onenet 平台的项目列表里看见你的在线设备。



至此，onenet 注册全过程完成，后续根据 onenet 规定的格式发送数据即可。

12.3.5.9 腾讯云（新版）自动注册

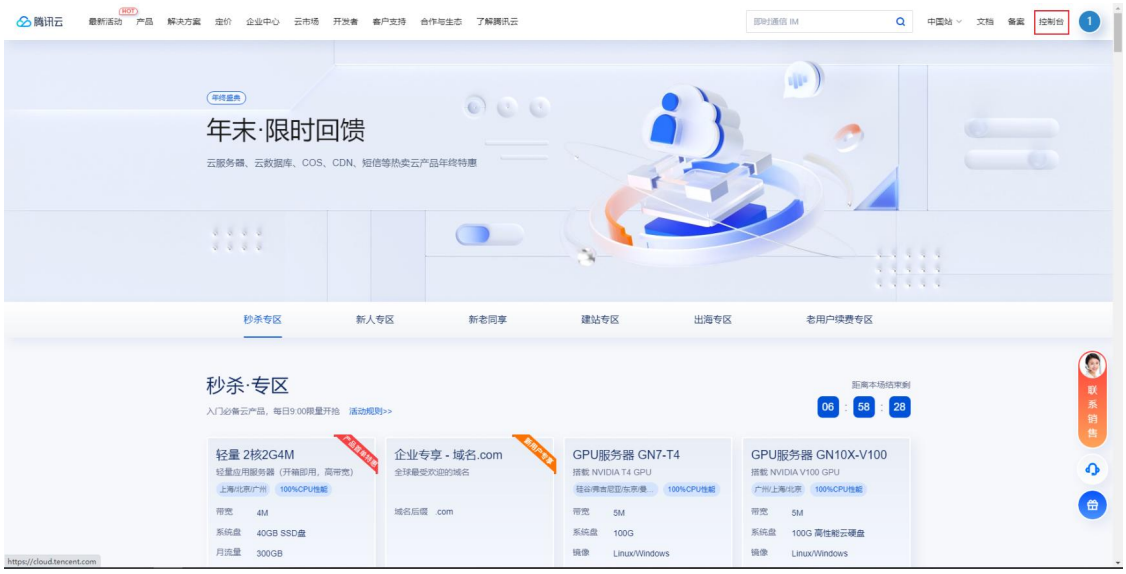
腾讯云端的配置

自动注册设备需要用到三个很重要的参数 **DeviceName**、**ProductID** 和 **ProductSecret**。

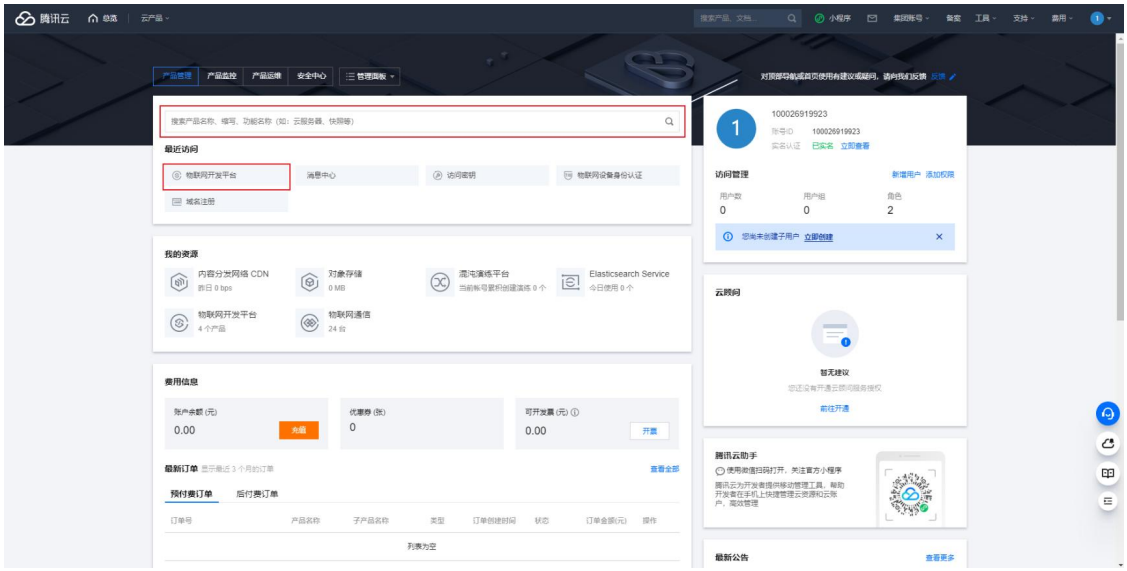
腾讯云 ProductKey 的产生

1.进入腾讯云物联网开发平台

登录进腾讯云 web 端后，单击控制台



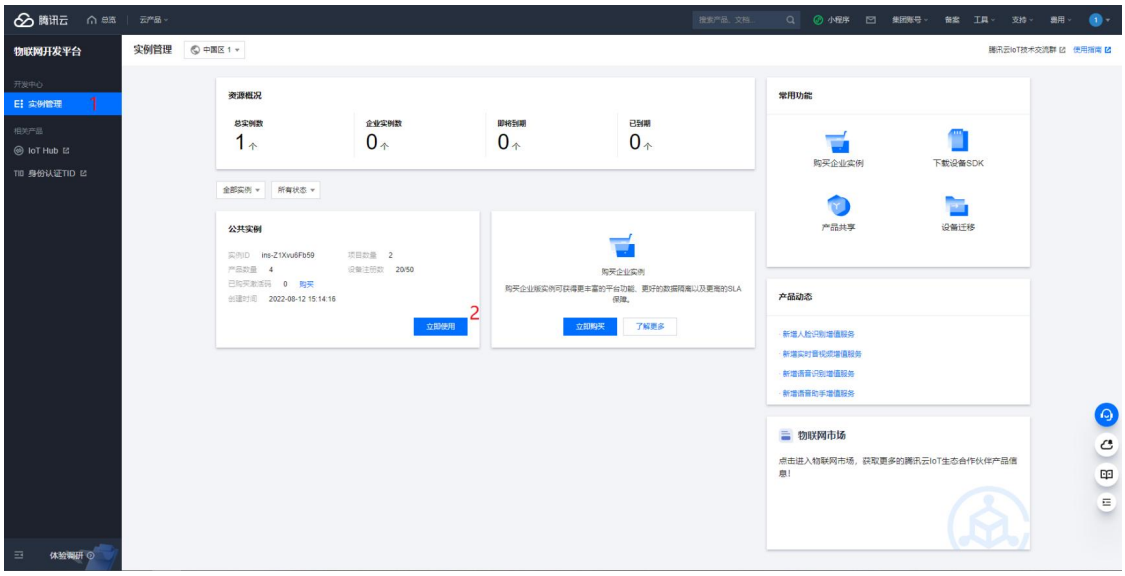
进入控制台后，在**搜索栏**中搜索物联网平台，点击进入即可



进入平台后点击**实例管理**

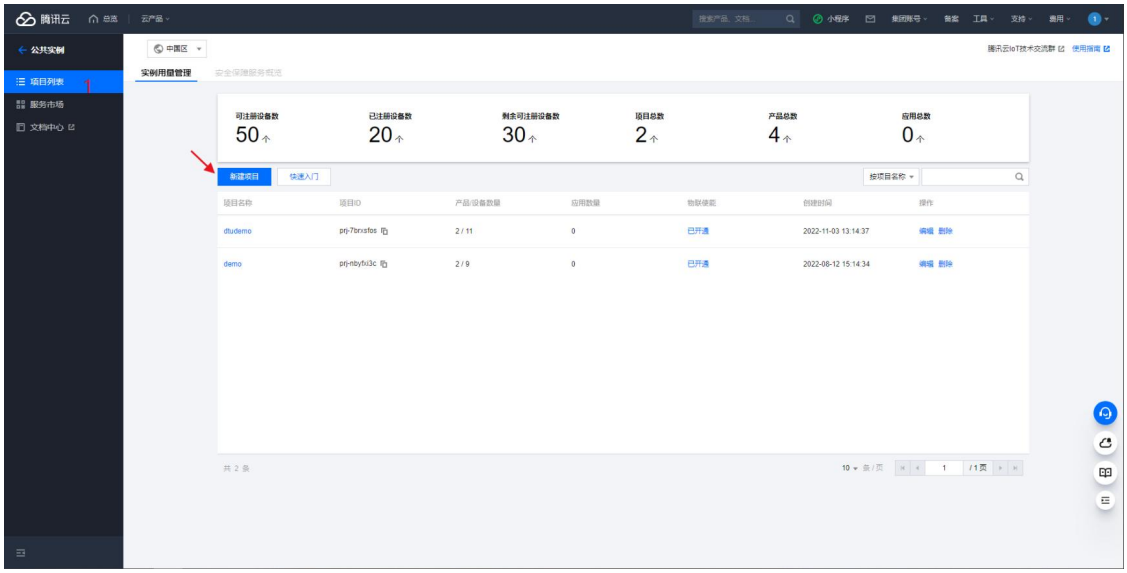
2.公共实例与企业实例的选择

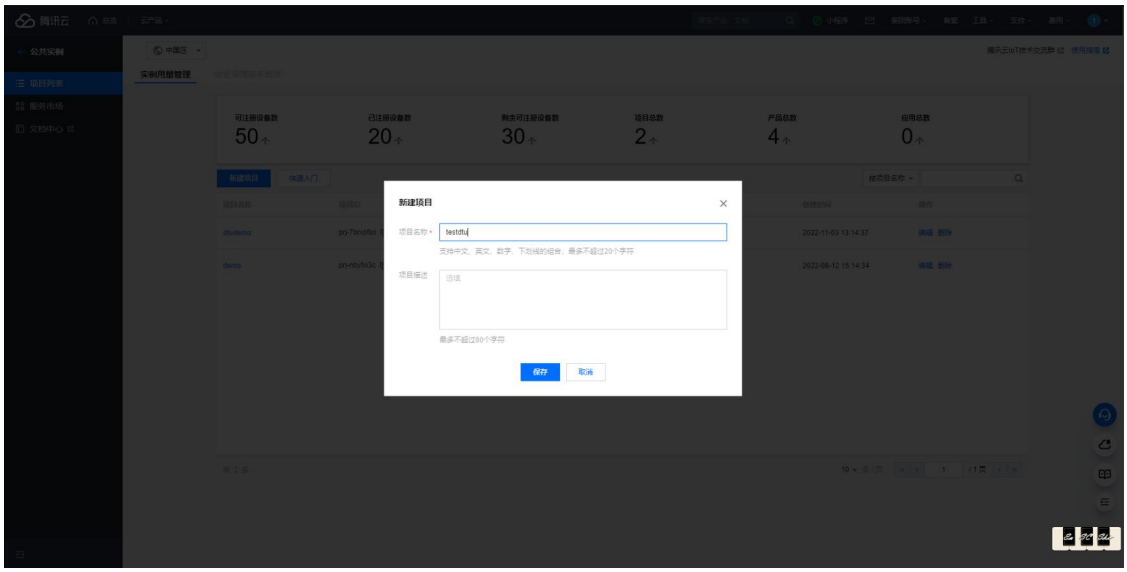
关于公共版实例与企业版实例的区别可以点击这里查看， 这里为了演示方便，我们选择了公共版实例



3.创建项目

进入公共版实例以后点击**项目列表--->创建项目**，默认无项目，截图中的几个项目是我自己创建的

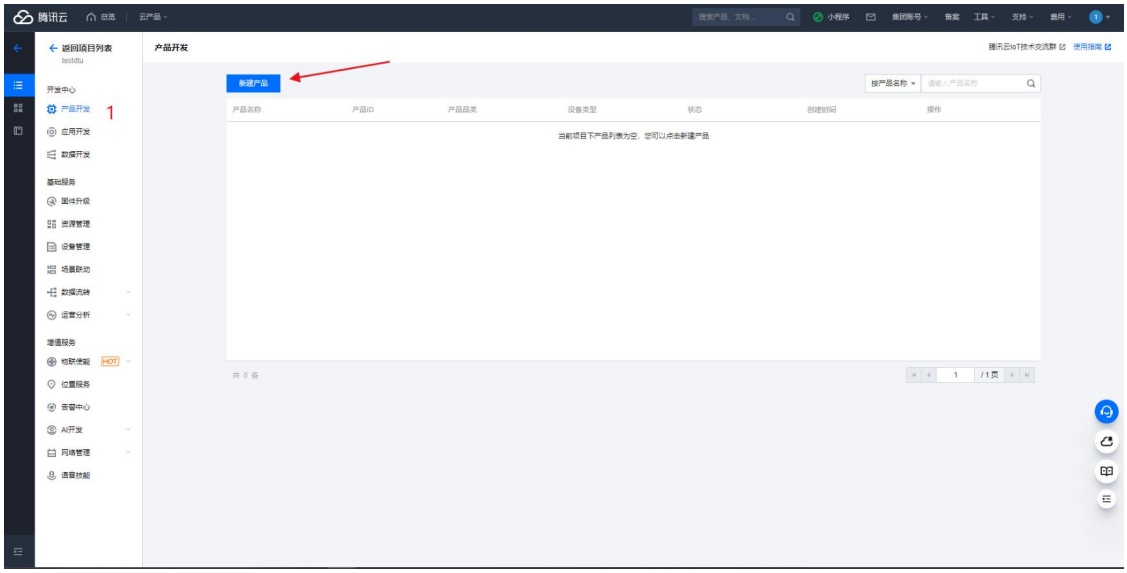


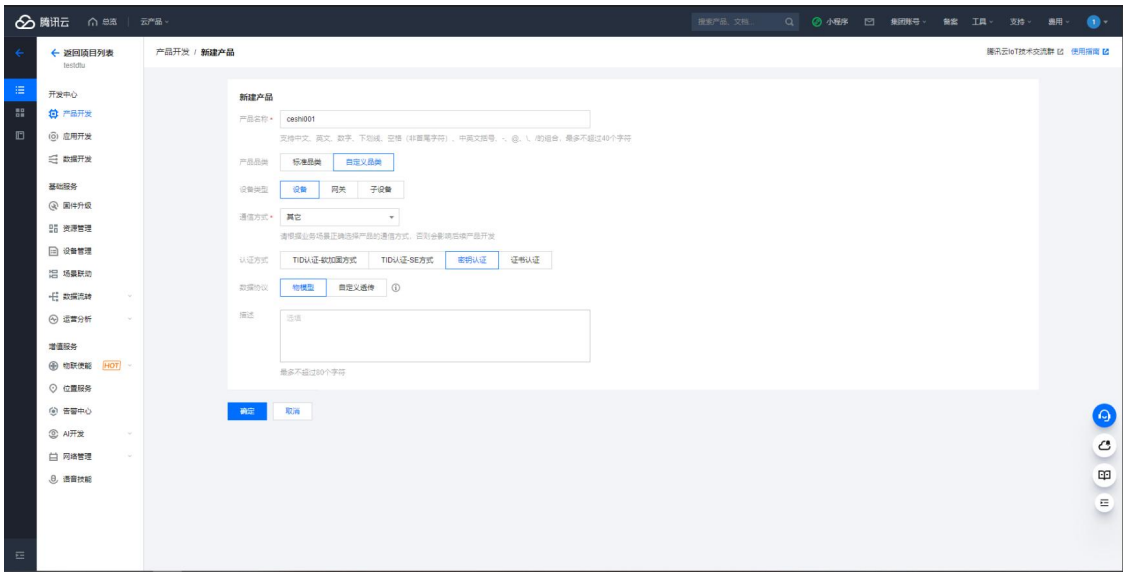


项目名称用户自定义即可 **项目描述**用户自定义即可

4.创建产品

进入项目以后点击**产品开发--->创建产品**，默认无产品，截图中的几个产品是我自己创建的

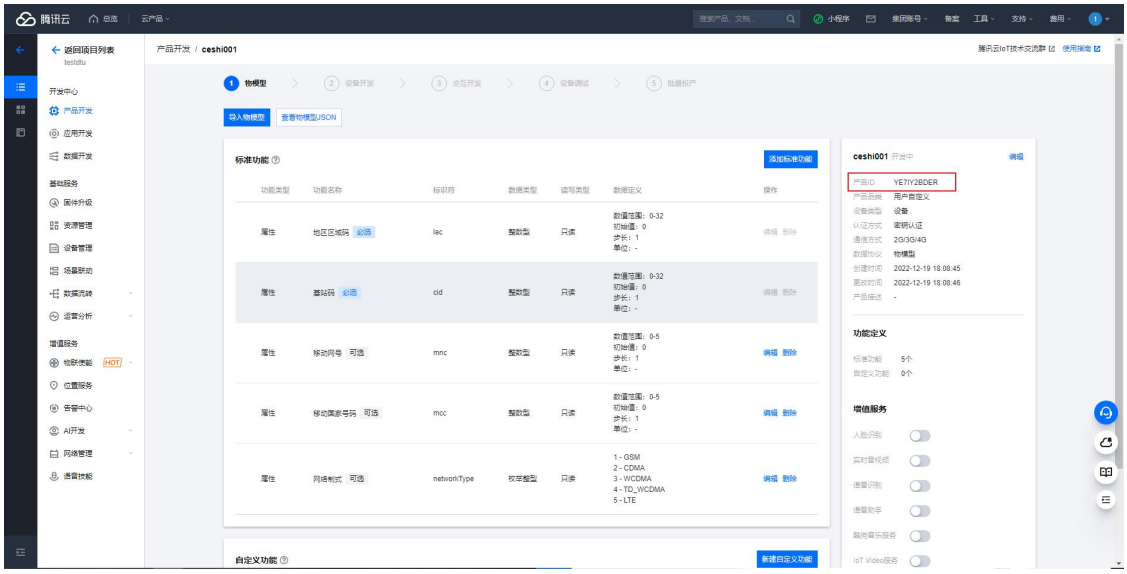




产品名称用户自定义即可 **产品品类**根据用户使用场景自由选择，方便演示这里选择第二个

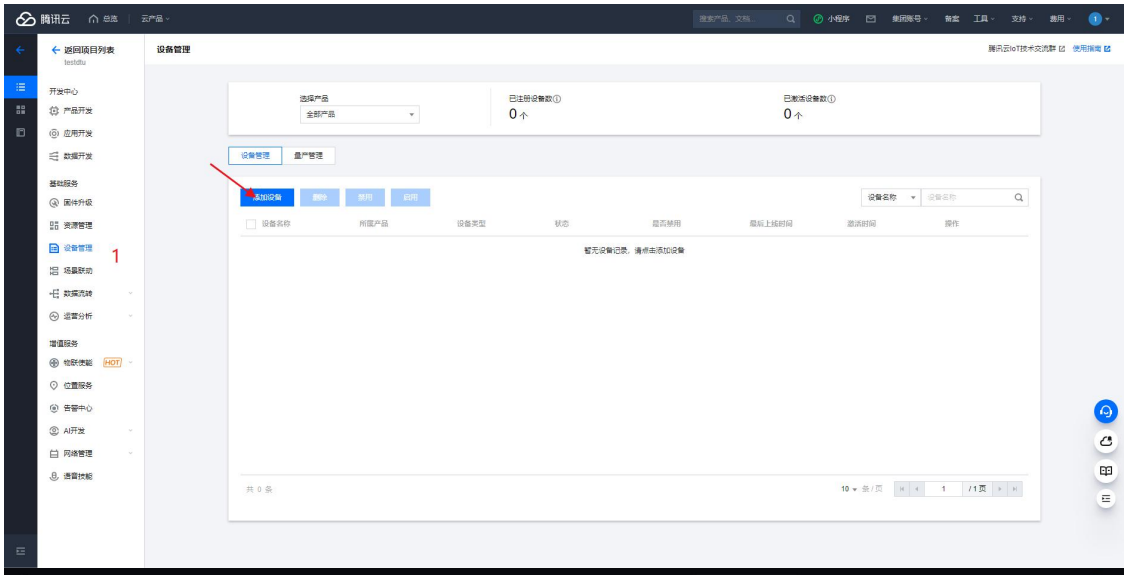
设备类型选择**设备**。 **通信方式**这里，因为我们使用的是 Cat.1/2G 模块，所以要选择 **2G/3G/4G**。 **认证方式**这里，由于本文主要讲的是**自动注册**，所以选择**密钥认证**。 **数据协议**这里，可以选择**物模型**和**自定义透传**。 物模型协议是腾讯云物联网开发平台定义的物模型协议，基于 JSON 格式可方便的进行设备向云端进行数据传输，并在云端进行存储与处理。 自定义透传协议是基于 MQTT 协议将用户私有业务数据上云。 **描述**用户自定义即可 点击**确定**后即可创建自己的产品。 创建成功后，因为本章讲解的主要是自动注册功能，直接点击**设备名称**即可。

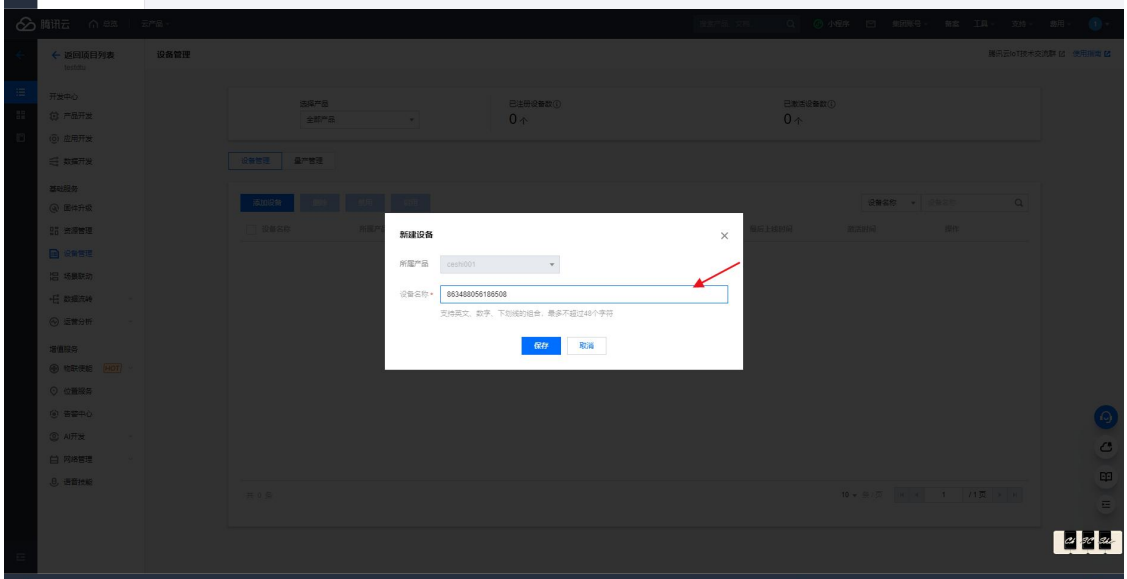
5.查询 productID

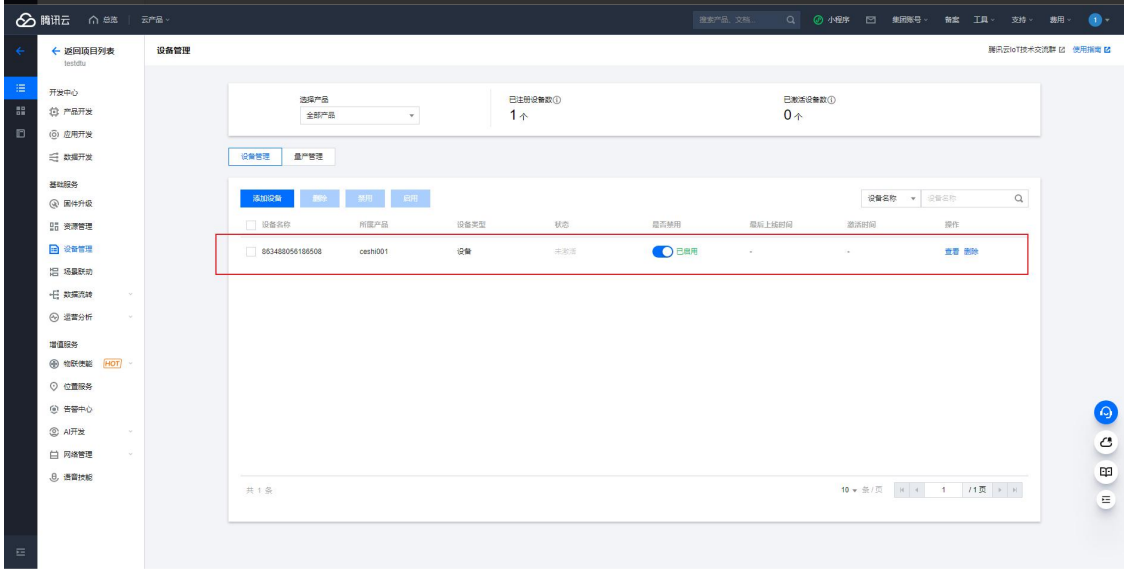


在项目名下即可看到自动注册的 **ProductSecret** 点击动态注册将出现 ProductSecret，点击显示可以查看 ProductSecret。 点击自动创建设备可以根据 **DeviceName** 自动创建设备，自动创建设备如果不填 **DeviceName** 会默认 IEMI 作为设备名称创建设备，不点击则要在设备管理里面创建设备。

6.创建设备


腾讯云
设备管理
添加设备
1


腾讯云
设备管理
新建设备
86348056186508
保存
取消

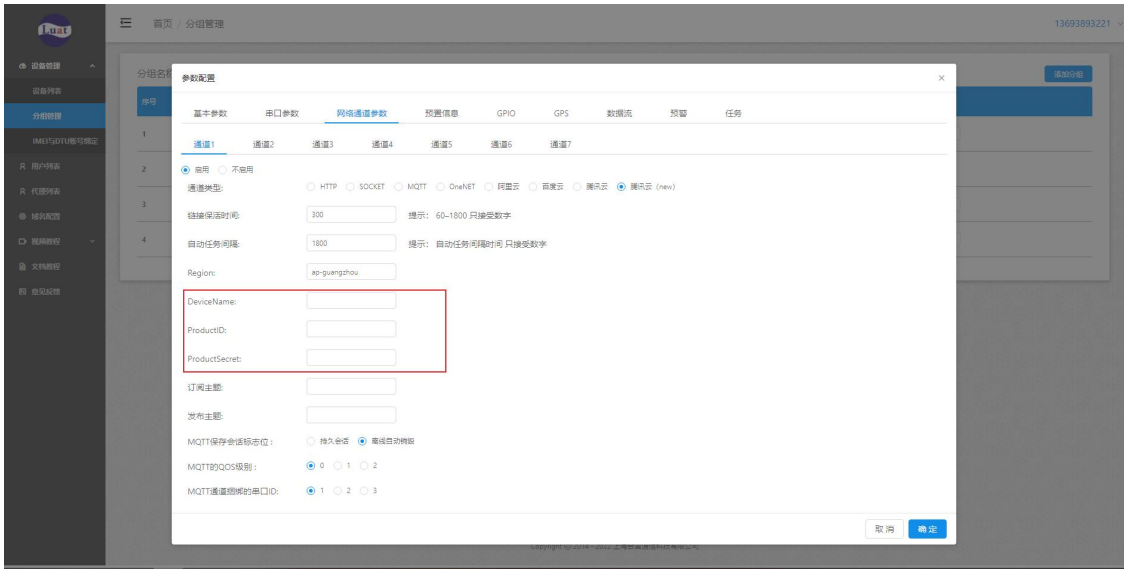

腾讯云
设备管理
86348056186508
ceshi001

如果没有选择自动创建设备，则需要在设备管理里面创建设备，一般用设备 IMEI 作为设备名称。

至此，腾讯云平台配置工作完成，接下来做 iRTU 的 web 端配置

iRTU web 端配置

基本参数的配置和网络通道参数中个参数的含义与解释在前面的章节中有详细的说明，在这里不做过多讲解，用户根据自己情况进行选择即可。 自动注册到腾讯云最为重要的三个参数，**DeviceName、ProductID、ProductSecret** 前面讲过怎么获取了，将这三个参数填入对应位置即可，如果选择了自动创建设备可以不填，默认模块的 IMEI 号为设备名称。



单击确定，即可完成所有配置。

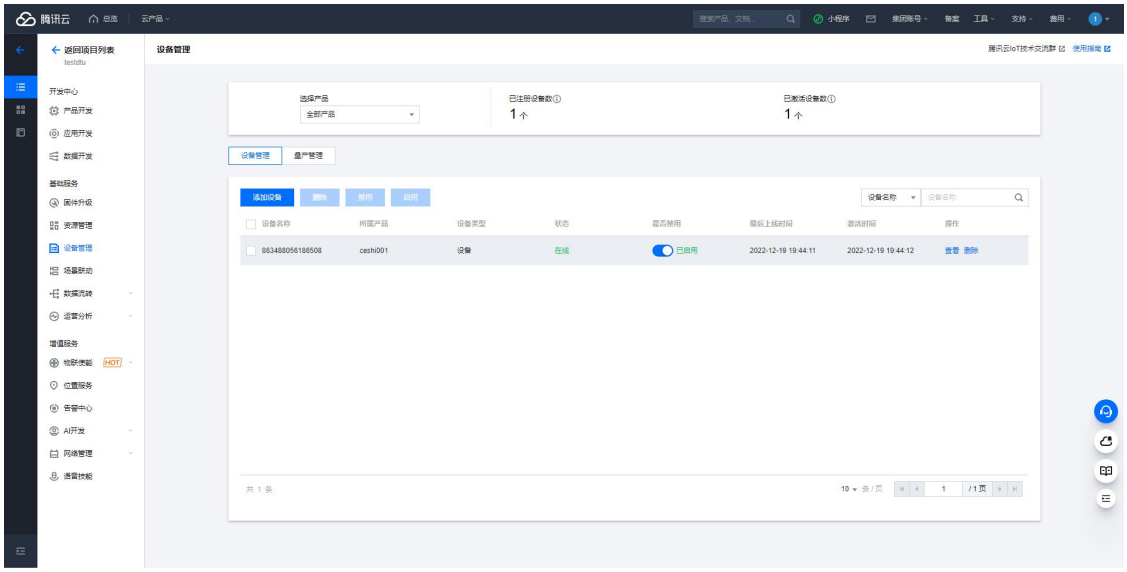
接下来给模块上电/重启，在设备上电情况下也可以通过串口/其他服务器端给模块发送 rrpc,upconfig 命令（串口端还可以发送+++命令），让设备读取到最新的参数配置即可。

当模块检测到参数配置有更新的情况下，会去跟新自己的参数，然后重启设备。重启之后就会去链接腾讯云，如果这时你的设备开启了日志打印且设备直连电脑，luatools 打开且日志输出正常的情况下，会观察到 luatools 打印了下面两张截图的红框部分，如果都打印了，即证明你的设备连接腾讯云成功。

```
[2022-12-19 19:21:54.111] [I]-[socket:sock_close] 2
[2022-12-19 19:21:54.111] [I]-[socket:on_response:] 2 SOCKET_CLOSE 0
[2022-12-19 19:21:54.111] [W]-[---------- tencent iot is start! ----------]
[2022-12-19 19:21:54.111] [I]-[UNPACK1] txiotnew 300 1800 ap-guangzhou  YE7IY2BDER RnBjvjUnKYE8WnsppUZuhQeW  1 0 1
[2022-12-19 19:21:54.111] [I]-[UNPACK] 300 1800 ap-guangzhou  YE7IY2BDER RnBjvjUnKYE8WnsppUZuhQeW  1 0 1
[2022-12-19 19:21:54.111] [I]-[KKKK] 1
[2022-12-19 19:21:54.111] [I]-[PIOS] table: 0x80a83670
[2022-12-19 19:21:54.111] [I]-[REG] ffffffff
[2022-12-19 19:21:54.111] [I]-[convert] 0
[2022-12-19 19:21:54.111] [I]-[passon] false
[2022-12-19 19:21:54.111] [I]-[upprot] table: 0x80a98070
[2022-12-19 19:21:54.111] [I]-[dwprot] table: 0x80ab0930
[2022-12-19 19:21:54.111] [D]-[socket.recv] 48 nil
[2022-12-19 19:21:54.111] [I]-[socket:sock_close] 1
[2022-12-19 19:21:54.111] [I]-[ril.proatc] ERROR
[2022-12-19 19:21:54.111] [I]-[ril.defrsp] AT+EEMGINFO? false ERROR nil
[2022-12-19 19:21:54.111] [I]-[ril.sendat] AT+CSQ
[2022-12-19 19:21:54.111] [I]-[socket:on_response:] 1 SOCKET_CLOSE 0
[2022-12-19 19:21:54.111] [I]-[ntp.timeSync is date:] 2022/12/19,19:21:54
[2022-12-19 19:21:54.111] [I]-[link.pdpCmdCnf] CONNECT_DELAY true nil nil
[2022-12-19 19:21:54.111] [I]-[publish IP_READY_IND ready:] true
[2022-12-19 19:21:54.111] [I]-[ril.proatc] +CSQ: 30,99
[2022-12-19 19:21:54.111] [I]-[ril.proatc] OK
[2022-12-19 19:21:54.111] [I]-[ril.sendat] AT+CCLK="22/12/19,19:21:54+32"
[2022-12-19 19:21:54.427] [I]-[ril.proatc] OK
[2022-12-19 19:21:54.427] [I]-[ AutoTimeSync is Done!]
[2022-12-19 19:21:54.427]
```

```
[2022-12-19 19:22:03.295] [I]-[socket:connect-coreid,prot,addr,port,cert,timeout] 0 TCP YE7IY2BDER.iotcloud.tencentdevices.com 1883 nil 120
[2022-12-19 19:22:03.454] [I]-[socket:on_response:] 0 SOCKET_CONNECT 0
[2022-12-19 19:22:03.454] [I]-[socket:connect: connect ok]
[2022-12-19 19:22:03.454] [D]-[mqtt.client:write] 10A70100044D51545404C2012C0019594537495932424445523836333438383035363138363530380033594537495932424 4 50
[2022-12-19 19:22:03.454] [D]-[socket.send] total 170 bytes first 30 bytes □□

[2022-12-19 19:22:03.578] [I]-[socket:on_response:] 0 SOCKET_SEND 0
[2022-12-19 19:22:03.578] [D]-[socket.recv] 4 nil
[2022-12-19 19:22:03.578] [D]-[mqtt.unpack] 4 20020000 4
[2022-12-19 19:22:03.578] [I]-[mqtt连接成功]
[2022-12-19 19:22:03.578] [I]-[sub1] nil
[2022-12-19 19:22:03.609] [I]-[pub1] YE7IY2BDER/863488056186508/event/863488056186508
[2022-12-19 19:22:03.609] [I]-[qos1] 0
[2022-12-19 19:22:03.609] [D]-[mqtt.client:write] 8237000200325945374959324244455 22F38363334383830353631383635303820F636F6E74726F6C2F383633343838303536 50
[2022-12-19 19:22:03.609] [D]-[socket.send] total 57 bytes first 30 bytes 7

[2022-12-19 19:22:03.737] [I]-[socket:on_response:] 0 SOCKET_SEND 0
[2022-12-19 19:22:03.737] [D]-[socket.recv] 5 nil
[2022-12-19 19:22:03.737] [D]-[mqtt.unpack] 5 9003000280 5
[2022-12-19 19:22:03.737] [I]-[mqtt订阅成功]
[2022-12-19 19:22:03.737] [D]-[mqtt.client:write] 303A0030594537495932424445522F38363334383830353631383635303820F6576656E742F383633343838303536313836 35 50
[2022-12-19 19:22:03.737] [D]-[socket.send] total 60 bytes first 30 bytes 0:
[2022-12-19 19:22:03.766] [I]-[isrndis] 0 nil
[2022-12-19 19:22:03.766] - "dtu.warn.gpio里的元素有" = {
[2022-12-19 19:22:03.766] -    1 = 0
[2022-12-19 19:22:03.766] -    2 = 0
[2022-12-19 19:22:03.766] -    3 = 0
```

当然，你也可以在自己的腾讯云平台的项目列表里看见你的在线设备。



至此，腾讯云自动注册全过程完成。

12.3.5.10 自动任务采集

web 端配置

自动任务采集：当用户觉得 iRTU 现有的功能不能满足要求，想获取模块的各种信息，例如基站定位的结果，获取 ADC 的电压值，获取模块 SN/IMEI 号，获取 GSM 网络注册状态等，就可以在自动采集任务中去实现。下面简单介绍下自动采集任务如何操作。

打开**预置信息-->自动采集任务**-->串口 1/串口 2(根据用户实际情况进行选择)，勾选**启用选项**。

采集等待：部分信息（例如基站定位结果）是需要延时等待才会有结果的（例如基站定位需要模块去和基站进行通讯，获取到基站返回的原始数据后，并不能直接解析，需要把数据发送给解析服务器，服务器解析好了以后，再把解析后的数据返回给模块），所以需要设置采集等待时间。

CMD：此项即是**自动采集**需要发送的命令了，命令格式如下

```
--"用户自定义字符串"和用户想要执行的函数中间必须要加“ .. ”，这种写法在 lua 中表示将前后的字符或变量拼接起来

function  return "用户自定义字符串"..用户想要执行的函数 end
```

提示：

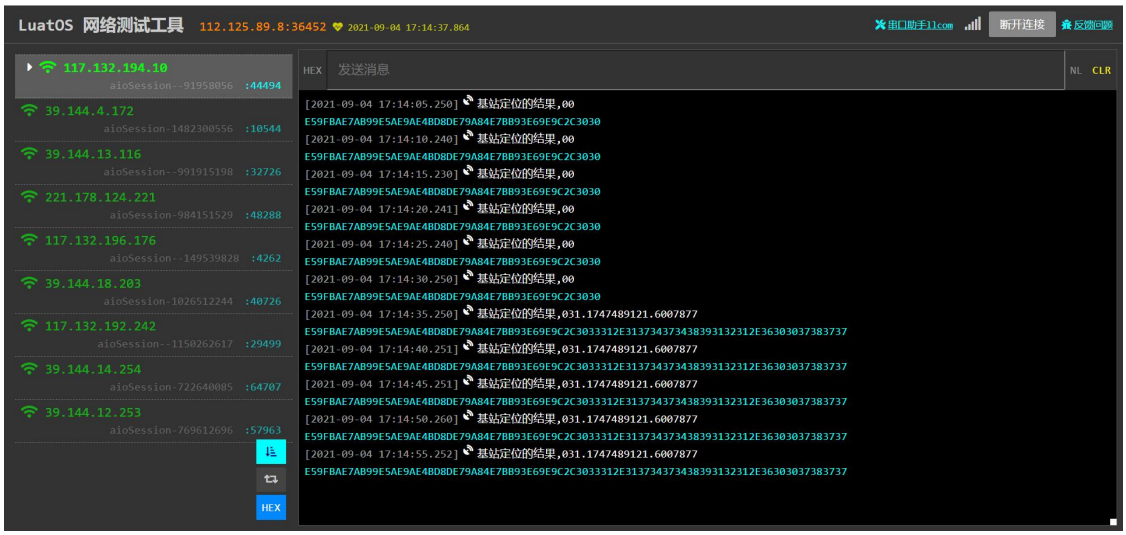
1. 函数需要用 function 开头，用 end 结尾,用来区别 HEX 指令。用串口写入该配置参数的时候，注意函数中的分号要转义处理。

2. 函数结尾 return 返回的数据会被发往用户配置的服务器，用户可以自定义通信报文协议。

3. 如果函数需要写入数据到串口，直接调在函数调用“uart.write(uid,str)” 类似的 Luat-API。

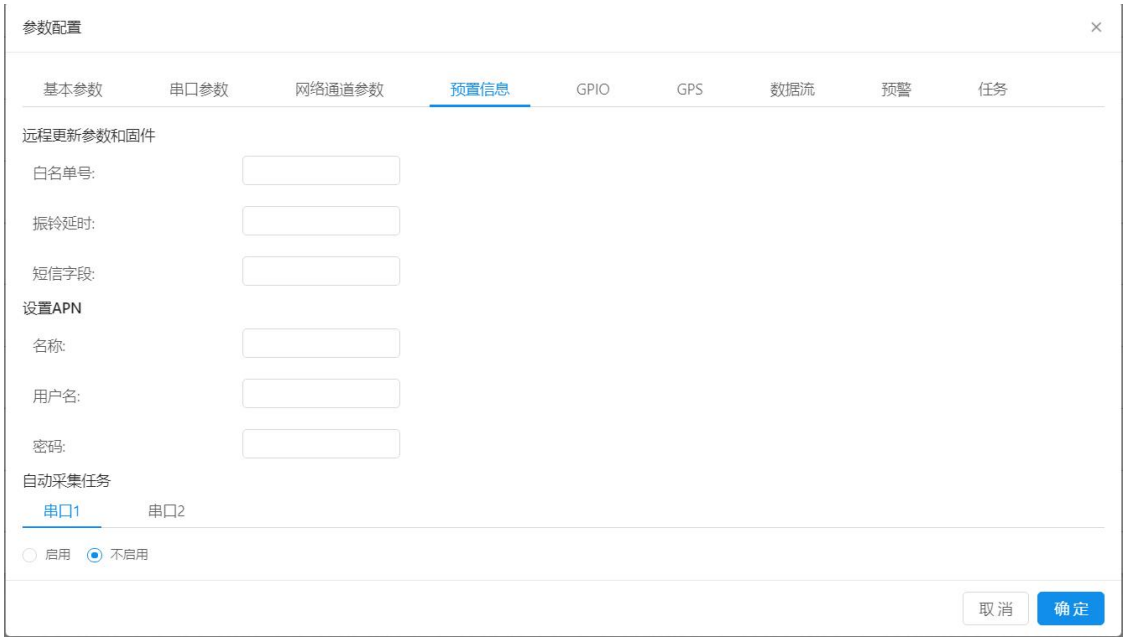
例如

```
function return "基站定位的结果,"..create.getLat()..create.getLng() end
```



以上演示仅是**自动采集任务**的实例 demo，用户可根据自己的实际需要去写自己的**自动采集任务**

12.3.5.11 设置 APN



目前 web 端和串口配置只能设置名称，用户名和密码三项，加密方式默认为 0，不加密，本地配置工具里可以配置加密方式为 0:不加密、1:PAP、 2:CHAP 三种。用户根据实际需要，自行配置即可。

12.3.5.12 阿里云一型一密

阿里云一型一密注册

阿里云一机一密与一型一密注册过程相似 区别是一机一密需要项目的 **ProductKey**，**DeviceSecret** 还有设备的 **ID**（默认为设备的 IMEI，也就是 **DeviceName**），而一型一密不需要设备的 **DeviceName** 一型一密需要打开**自动注册** 其他过程完全一致,本文介绍阿里云的一型一密注册过程。

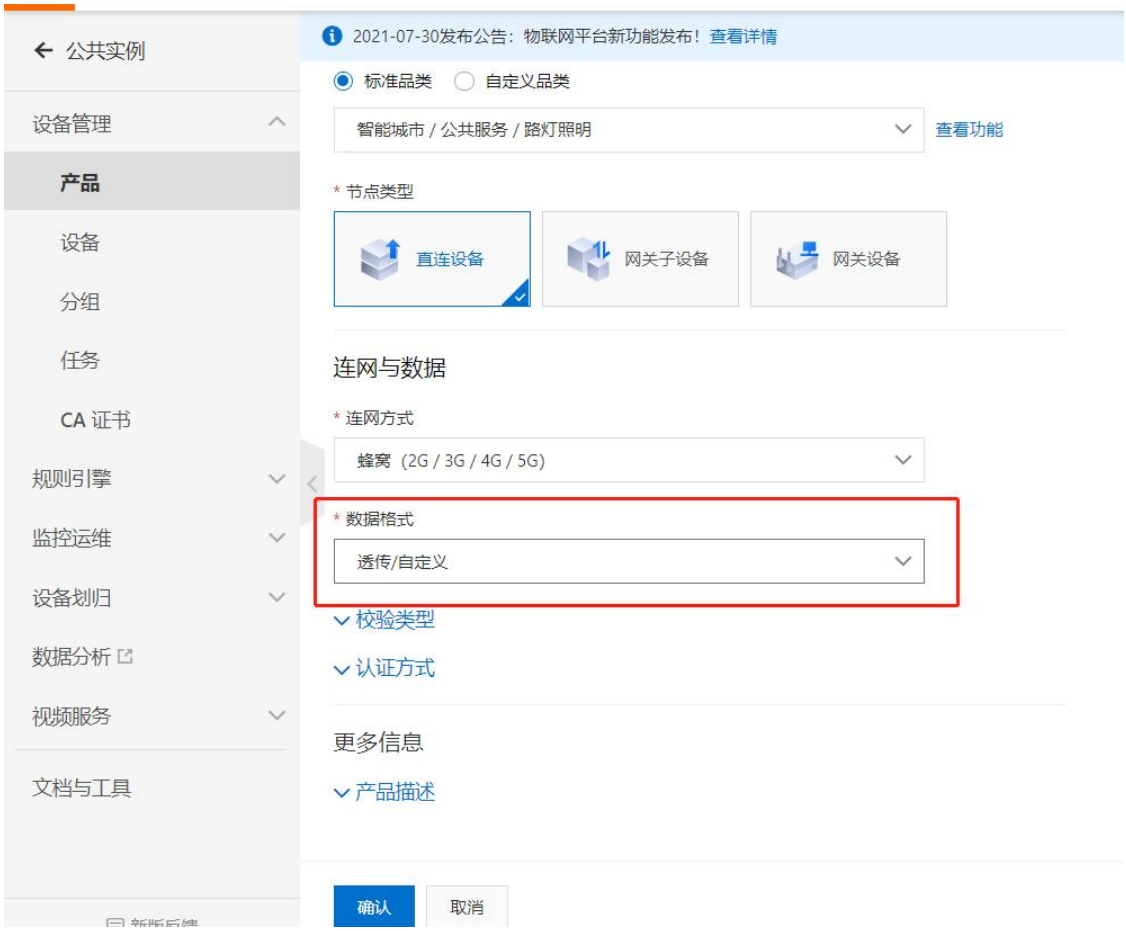
阿里云界面配置过程

关于如何打开控制台，如何进入阿里云物联网平台，如何新建产品，在"阿里云自动注册"一文中有详细的描述在这里不做过多赘述。注：本文为了演示方便，演示过程均为为公共实例的注册。

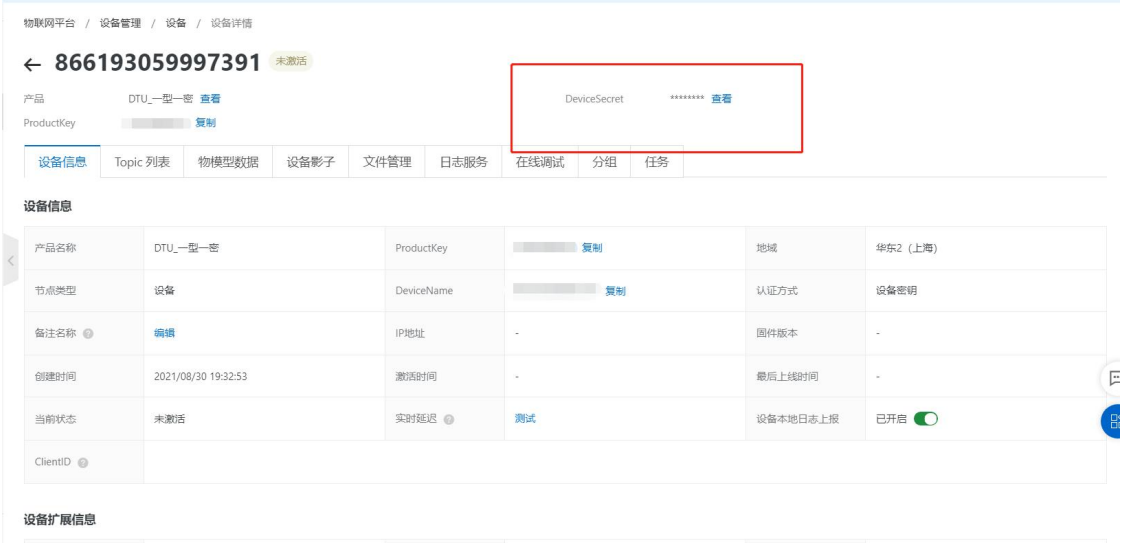
创建一型一密的项目

新建产品如下所示即可（注：所属品类根据用户实际需求进行选择，本文为了掩饰方便，选择了默认的第一个）

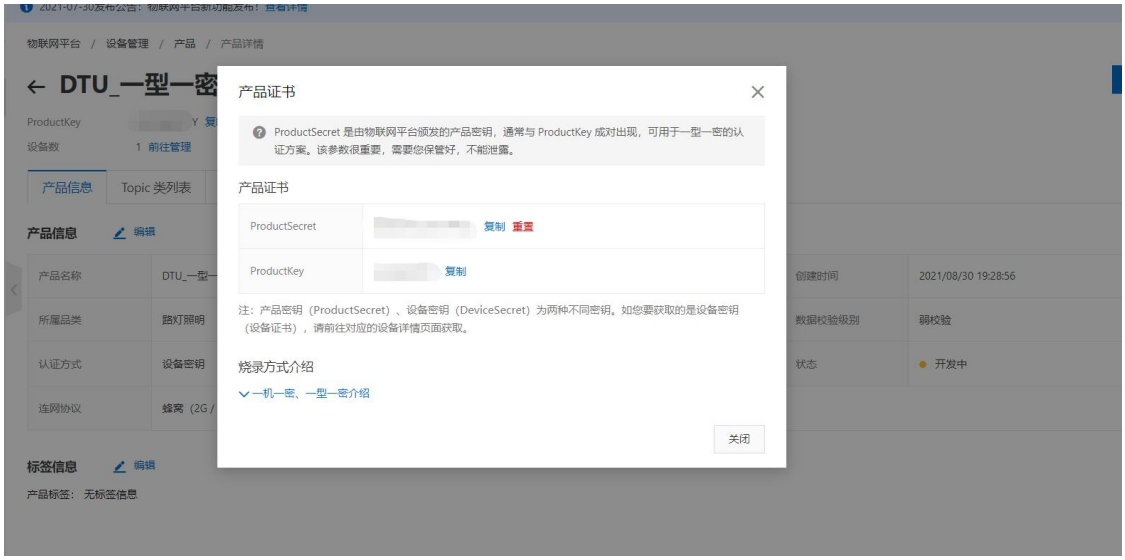
注意这里的数据格式，选择透传和 Alink_JSON 都可以，用户根据实际需求进行选择，如果需要使用到物模型功能，勾选 Alink_JSON，如果只是透传数据则勾选透传即可。



由于一机一密和一型一密没有自动注册这个功能，所以我们需要在创建好的产品下手动添加设备。（注：一定要打开**动态注册**选项） 设备名建议用 IMEI 号来命名，方便用户识别。(注：虽然合宙提供了修改 IMEI 的方法，但是仅作为调试使用，不建议用户量产使用，会影响到后期的空中升级，并违反相关法律法规) 添加完设备后回到**产品**，点击你的**一型一密产品**，进入产品界面,点击红框内的**查看**。



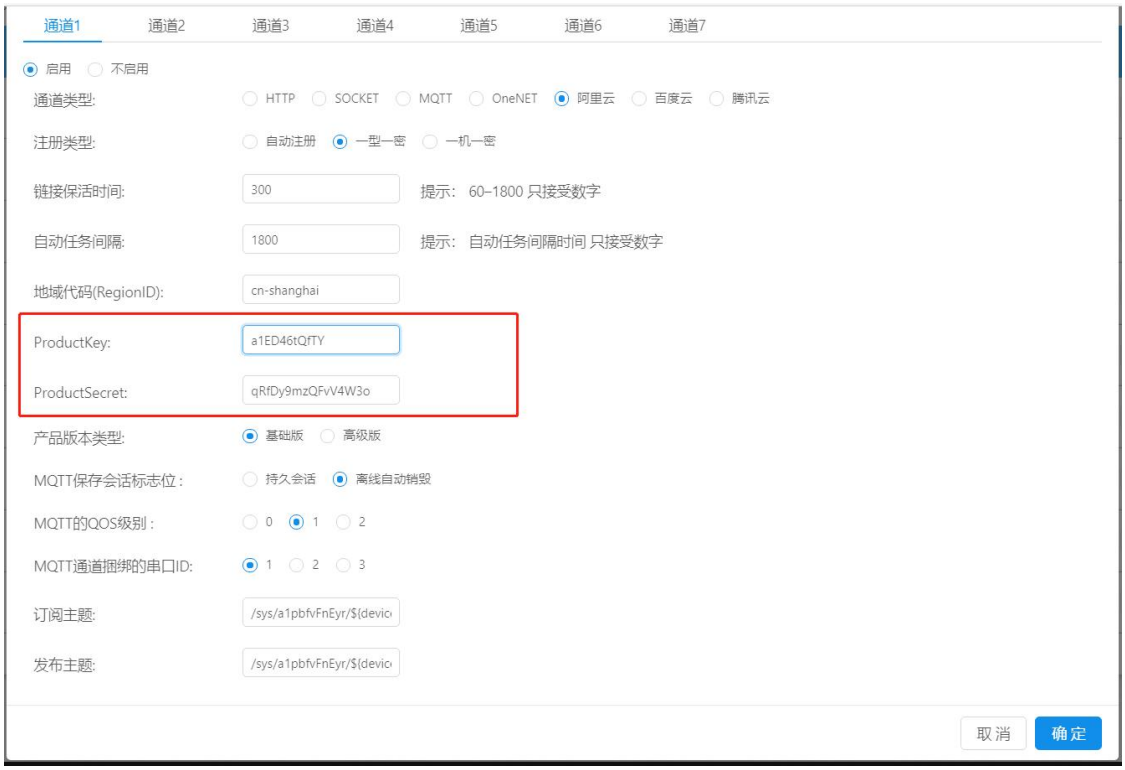
将自己 **ProductSecret** 和 **ProductKey** 保存起来，后面 iRTU 的 web 端配置需要。



一型一密最为重要的两个个参数，**ProductSecret** 和 **ProductKey** 到这里就获取完成了，接下来做 iRTU 的 web 端配置。

iRTU web 端配置

基本参数的配置和网络通道参数中个参数的含义与解释在前面的章节中有详细的说明，在这里不做过多讲解，用户根据自己情况进行选择即可。 阿里云一型一密最为重要的两个参数 **ProductSecret** 和 **ProductKey** 前面讲过怎么获取了，将这两个参数填入对应位置即可。



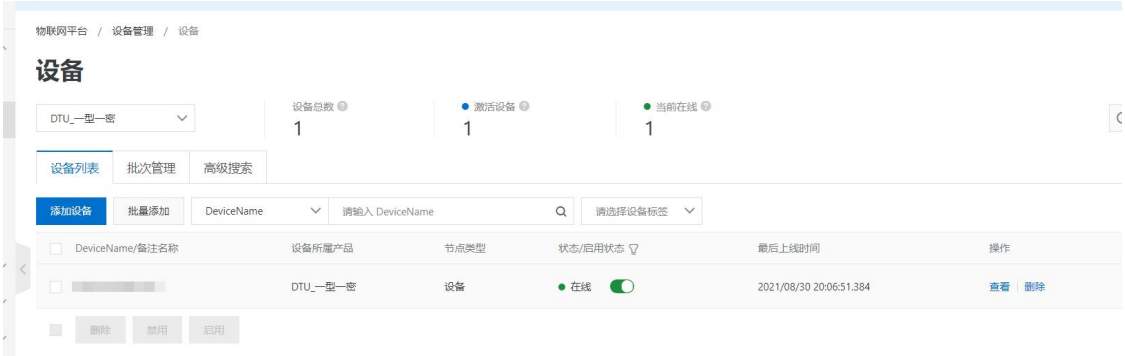
单击确定，即可完成所有配置。

接下来给模块上电/重启，在设备上电情况下也可以通过串口/其他服务器端给模块发送 rrpc,upconfig 命令（串口端还可以发送+++命令），让设备读取到最新的参数配置即可。

当模块检测到参数配置有更新的情况下，会去跟新自己的参数，然后重启设备。重启之后就会去链接阿里云，如果这时你的设备开启了日志打印且设备直连电脑，luatools 打开且日志输出正常的情况下，会观察到 luatools 打印了下面两张截图的红框部分，如果都打印了，即证明你的设备连接阿里云成功。

```
[2021-08-30 20:06:51.957] [I]-[socket:on_response:] 2 SOCKET_CLOSE 0
[2021-08-30 20:06:51.957] [W]-[---------- Aliyun iot is start! ----------]
[2021-08-30 20:06:51.957] lualibc_fopen fail -4200062,0,/alikey.cnf
[2021-08-30 20:06:51.957] [I]-[io.exists:] /alikey.cnf: strerror:-4200062
[2021-08-30 20:06:51.957] [I]-[socket:connect-coreid,prot,addr,port,cert,timeout] 0 TCPSSL iot-auth.cn-shanghai.aliyuncs.com 443 nil 120
[2021-08-30 20:06:52.017] [I]-[ril.proatc] +CSQ: 22,99
[2021-08-30 20:06:52.017] [I]-[ril.proatc] OK
[2021-08-30 20:06:52.577] [I]-[socket:on_response:] 0 SOCKET_CONNECT 0
[2021-08-30 20:06:52.577] [I]-[socket:connect: connect ok]
[2021-08-30 20:06:52.577] [I]-[发送的http报文:] POST /auth/register/device HTTP/1.0
[2021-08-30 20:06:52.577] Connection: close
[2021-08-30 20:06:52.577] Content-Type: application/x-www-form-urlencoded
[2021-08-30 20:06:52.577] Accept: */*
[2021-08-30 20:06:52.577] User-Agent: Mozilla/4.0
[2021-08-30 20:06:52.577] Content-Length: 118
[2021-08-30 20:06:52.577] Accept-Language: zh-CN,zh,cn
[2021-08-30 20:06:52.577] Host: iot-auth.cn-shanghai.aliyuncs.com
[2021-08-30 20:06:52.577] productKey=a1ED46tQfTY&deviceName[illegible]&random[illegible]3&sign=[illegible]3&signMethod=HmacMD5
[2021-08-30 20:06:52.637] [D]-[socket.send] total 357 bytes first 30 bytes POST /auth/register/device HTT
[2021-08-30 20:06:52.637] [I]-[socket:on_response:] 0 SOCKET_SEND 0
[2021-08-30 20:06:52.717] [D]-[socket.recv] 345 nil
[2021-08-30 20:06:52.717] [I]-[httpv2.response code:] 200
[2021-08-30 20:06:52.717] [I]-[socket:sock_close] 0
[2021-08-30 20:06:52.717] [I]-[socket:on_response:] 0 SOCKET_CLOSE 0
[2021-08-30 20:06:52.717] [W]-[一型一密动态注册返回三元组:] true true
[2021-08-30 20:06:52.757] [I]-[订阅或发布主题:] /sys/[illegible]/thing/event/property/post_reply
[2021-08-30 20:06:52.757] [I]-[订阅或发布主题:] /sys/[illegible]/thing/event/property/post
[2021-08-30 20:06:52.757] [I]-[MQTT HOST:PORT] a1ED46tQfTY.iot-as-mqtt.cn-shanghai.aliyuncs.com 1883
[2021-08-30 20:06:52.757] [I]-[MQTT clientID,user,pwd] 89860446101980203924|securemode=3,signmethod=hmacsha1[illegible]F7A4
[2021-08-30 20:06:52.757] 2A44A0E3AAB723010FF26F04E
[2021-08-30 20:06:52.757] [I]-[socket:connect-coreid,prot,addr,port,cert,timeout] 0 TCP a1ED46tQfTY.iot-as-mqtt.cn-shanghai.aliyuncs.com 1883 nil 120
[2021-08-30 20:06:52.917] [I]-[socket:on_response:] 0 SOCKET_CONNECT 0
[2021-08-30 20:06:52.917] [I]-[socket:connect: connect ok]
[2021-08-30 20:06:52.917] [D]-[mqtt.client:write] 10890100044D51545404C2012C00363839383630343436313031393830323033393234 7C7365637572656D6F64653D332C73 50
[2021-08-30 20:06:52.917] [D]-[socket.send] total 140 bytes first 30 bytes □
[2021-08-30 20:06:52.957] [I]-[socket:on_response:] 0 SOCKET_SEND 0
[2021-08-30 20:06:52.997] [D]-[socket.recv] 4 nil
[2021-08-30 20:06:52.997] [D]-[mqtt.unpack] 4 20020000 4
[2021-08-30 20:06:52.997] [D]-[mqtt.client:write] 8245000200402F7379732F6131706266764 66E4579722F383636313933303539393937333931 2F7468696E672F6576656E74 50
[2021-08-30 20:06:52.997] [D]-[socket.send] total 71 bytes first 30 bytes 倀
[2021-08-30 20:06:53.037] [I]-[socket:on_response:] 0 SOCKET_SEND 0
[2021-08-30 20:06:53.037] [D]-[socket.recv] 5 nil
[2021-08-30 20:06:53.037] [D]-[mqtt.unpack] 5 9003000280 5
[2021-08-30 20:06:53.437] [I]-[打印占用的内存:] 715.6640625
```

当然，你也可以在自己的阿里云平台的项目列表里看见你的在线设备。



至此，阿里云一型一密注册全过程完成。

12.3.5.13 阿里云一机一密

阿里云一机一密与一型一密注册过程相似，唯一区别是一机一密需要项目的 **ProductKey**，**DeviceSecret** 还有设备的 **ID**（默认为设备的 IMEI，也就是 **DeviceName**），而一型一密不需要设备的 **DeviceName**，其他过程完全一致，本文介绍阿里云的的一机一密注册过程。

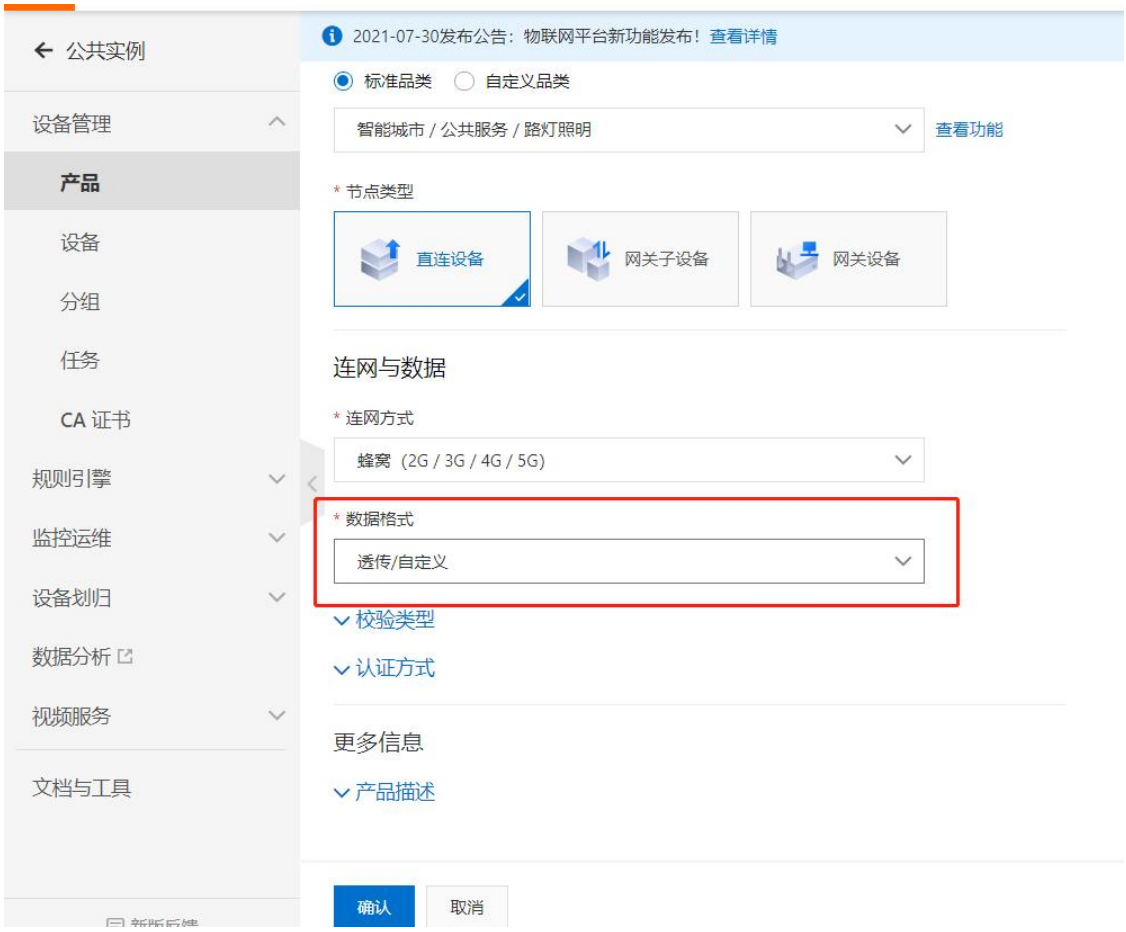
阿里云界面配置过程

关于如何打开控制台，如何进入阿里云物联网平台，如何新建产品，在“阿里云自动注册”一文中有详细的描述在这里不做过多赘述。注：本文为了演示方便，演示过程均为为公共实例的注册。

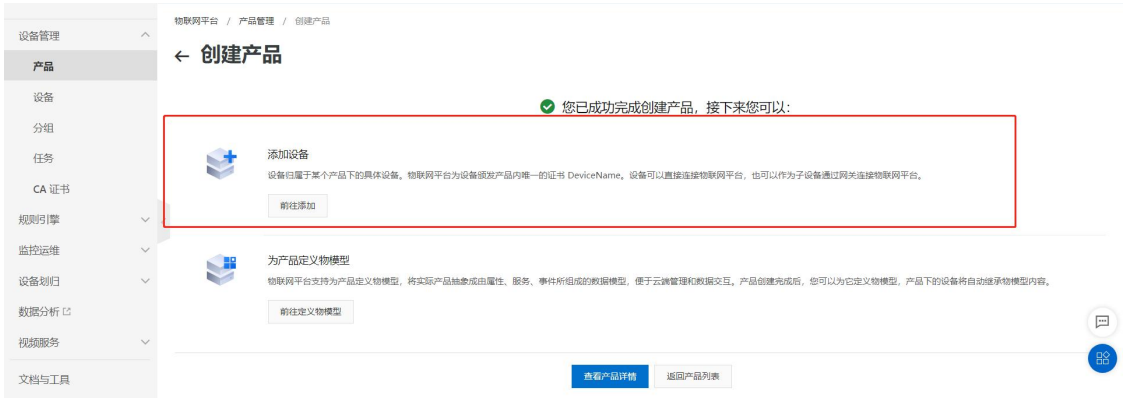
创建一机一密的项目

新建产品如下所示即可（注：所属品类根据用户实际需求进行选择，本文为了掩饰方便，选择了默认的第一个）

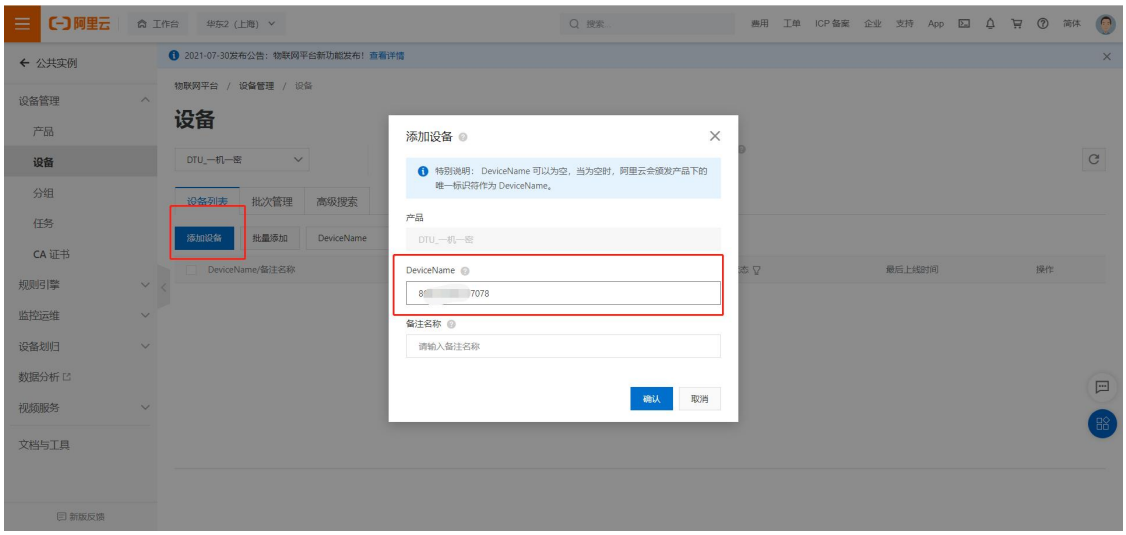
注意这里的数据格式，选择透传和 Alink_JSON 都可以，用户根据实际需求进行选择，如果需要使用到物模型功能，勾选 Alink_JSON，如果只是透传数据则勾选透传即可。



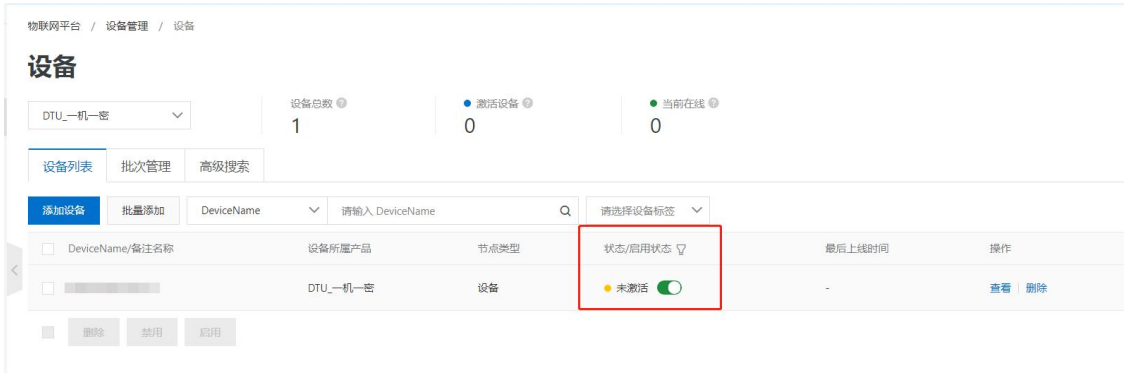
由于一机一密和一型一密没有自动注册这个功能，所以我们需要在创建好的产品下手动添加设备。



设备名建议用 IMEI 号来命名，方便用户识别。(注：虽然合宙提供了修改 IMEI 的方法，但是仅作为调试使用，不建议用户量产使用，会影响到后期的空中升级，并违反相关法律法规)

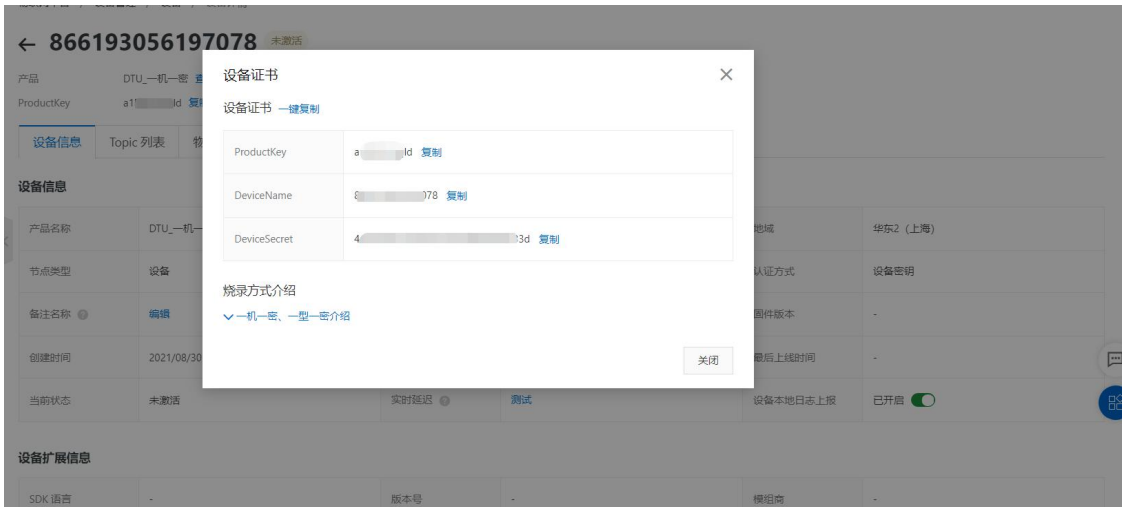


添加设备成功后，可以在平台上看到如下设备



显示未注册是因为我们在 WEB 端还没有配置模块的参数。 点击设备进行查看，可以看到如下图所示界面

点击上图中红框中的**查看**，即可查看本设备的三元组。



将自己三元组（即 **ProductKey，DeviceName，DeviceSecret**）保存起来，后面 iRTU 的 web 端配置需要。 一机一密最为重要的三个参数，**ProductKey，DeviceName，DeviceSecret** 到这里就获取完成了，接下来做 iRTU 的 web 端配置。

iRTU web 端配置

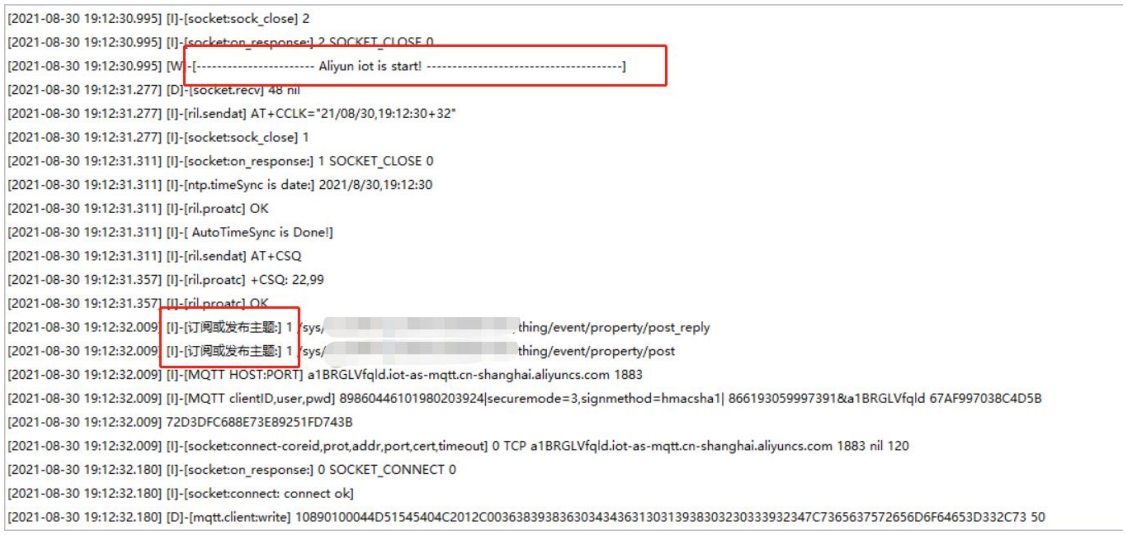
基本参数的配置和网络通道参数中个参数的含义与解释在前面的章节中有详细的说明，在这里不做过多讲解，用户根据自己情况进行选择即可。 阿里云一机一密最为重要的三个参数，**ProductKey，DeviceName，DeviceSecret** 前面讲过怎么获取了，将这三个参数填入对应位置即可。

单击确定，即可完成所有配置。

接下来给模块上电/重启，在设备上电情况下也可以通过串口/其他服务器端给模块发送 rrpc,upconfig 命令（串口端还可以发送+++命令），让设备读取到最新的参数配置即可。

当模块检测到参数配置有更新的情况下，会去跟新自己的参数，然后重启设备。重启之后就会去链接阿里云，如果这时你的设备开启了日志打印且设备直连电

脑，luatools 打开且日志输出正常的情况下，会观察到 luatools 打印了下面截图的红框部分，如果都打印了，即证明你的设备连接阿里云成功。



当然，你也可以在自己的阿里云平台的项目列表里看见你的在线设备。

至此，阿里云一机一密注册全过程完成。

12.3.5.14 阿里云的自动注册

阿里云端的配置

RAM 平台子用户的创建与权限

先从 RAM 平台子用户的创建与权限分配说起。

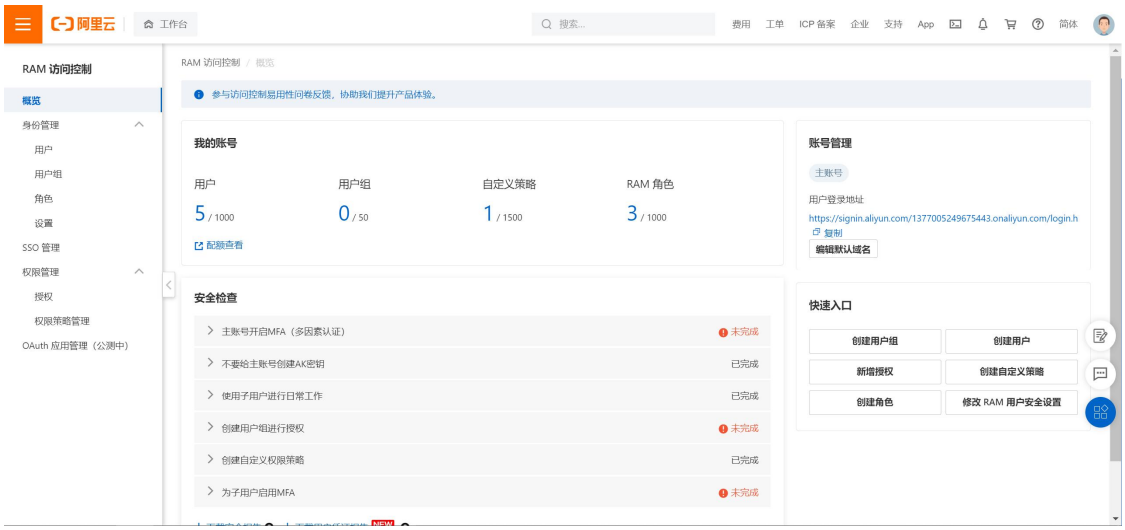
因为自动注册设备需要用到两个阿里云很重要的参数 **AccessKey ID** 和 **Access Key Secret**，而这两个参数在阿里云中的地位可不一般，他几乎可以对你一切产品进行操作，所以阿里云推出了 RAM（Resource Access Management）管理平台，RAM 的主要作用是控制账号系统的权限。通过使用 RAM，创建、管理子账号，并通过给子账号授予不同的权限，控制子账号对资源的操作权限，今天用到的就是这个。

1.开通

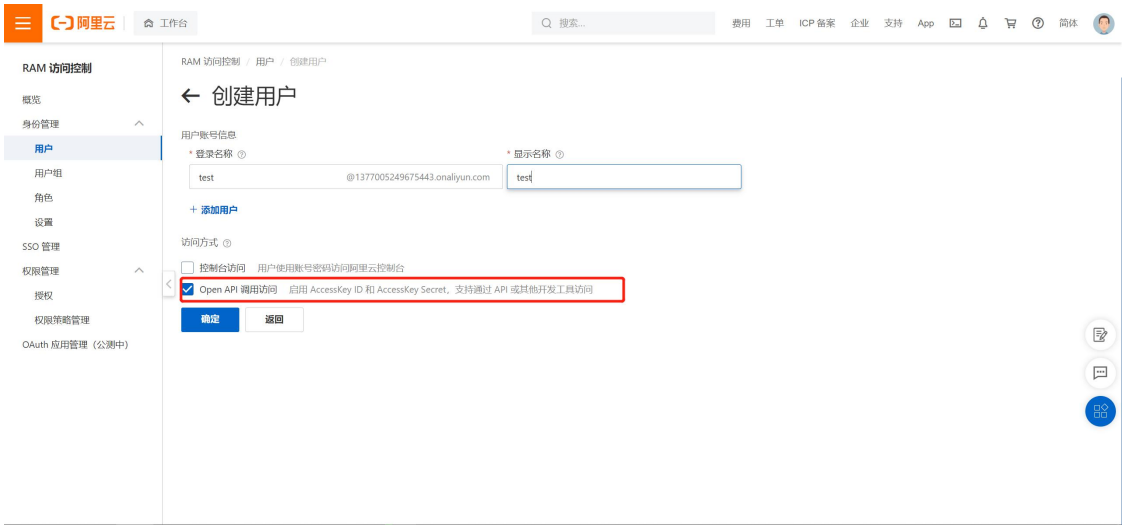
直接点这里就可以开通，前提是你要有阿里云账号(此处注册阿里云平台)

2.新建用户

开通界面如下图所示（默认没有用户那 5 个是我加的），点右侧**用户**--->新建用户

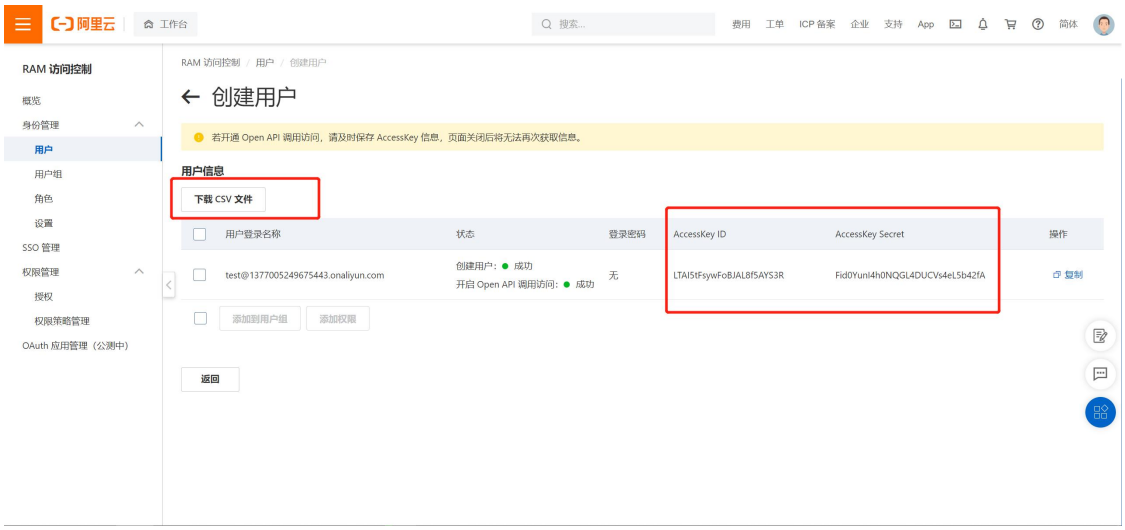


起个名，随便写，下面 **Open API** 调用访问一定勾选，后面说了启用 **AccessKeyID 和 AccessKeySecret**，支持通过 API 或其他开发工具访问，要的就是这个



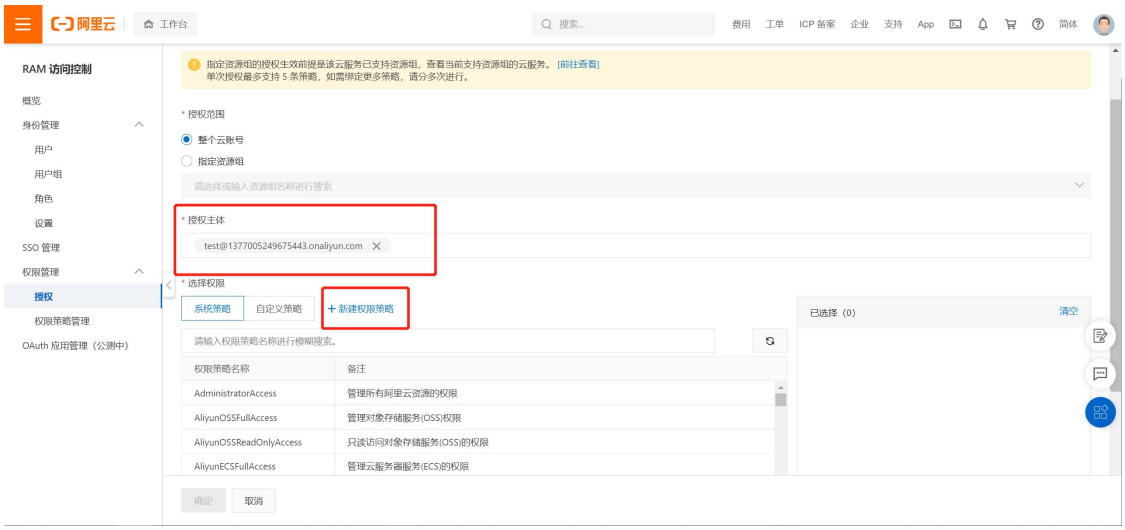
3.保存参数

保存好这几个参数备用，建议直接下载，这个只能查看一次忘了只能重置



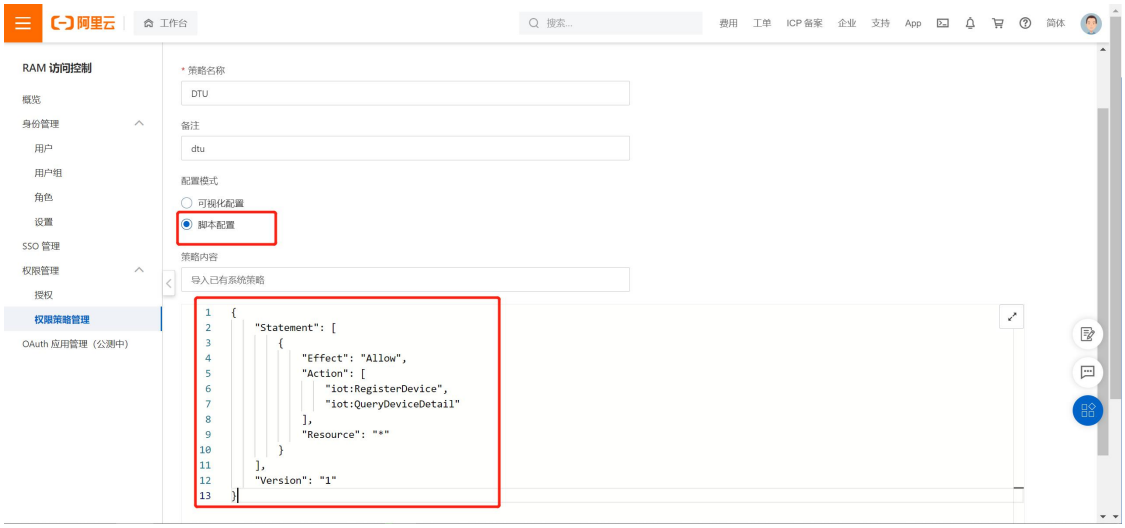
4.新增授权

新增一个**授权权限策略**,授权--->授权主体（输入你刚刚创建的用户名）--->新建权限策略



5.配置策略

名称随意，点击**脚本配置**



具体脚本如下

```
{
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
                        "iot:RegisterDevice",
                        "iot:QueryDeviceDetail"
        ],
        "Resource": "*"
    }
],
"Version": "1"
}
```

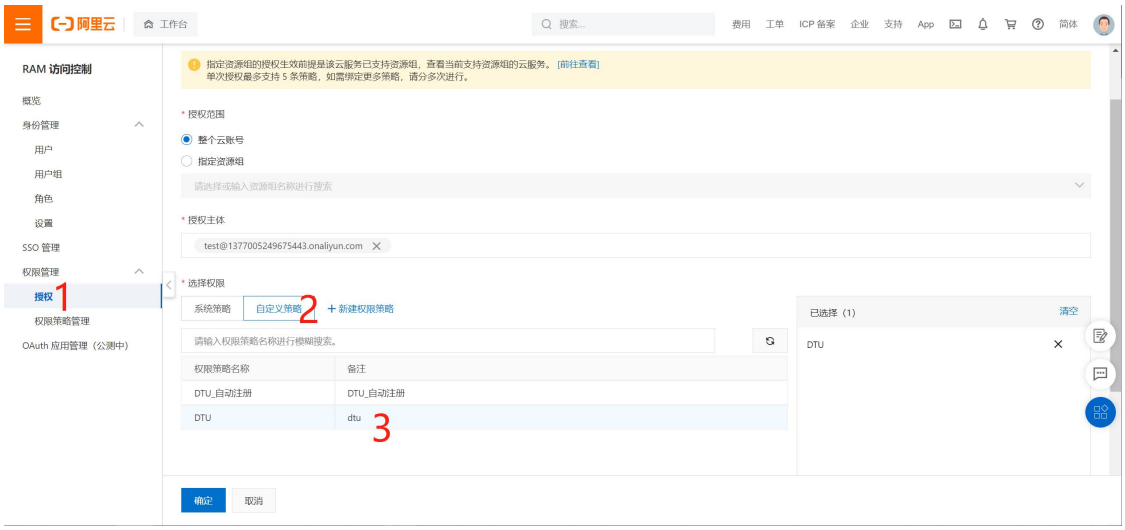
如何编写这个脚本可以看这里

这个脚本 iot:RegisterDevice 是注册设备，iot:QueryDeviceDetail 是查询设备详情，iRTU 只用了这两个权限。

6.添加权限

回到刚才创建的用户为他添加权限

选自定义，点击该策略即可添加至列表，然后点击确定即可



阿里云 ProductKey 的产生

1.进入阿里云物联网平台

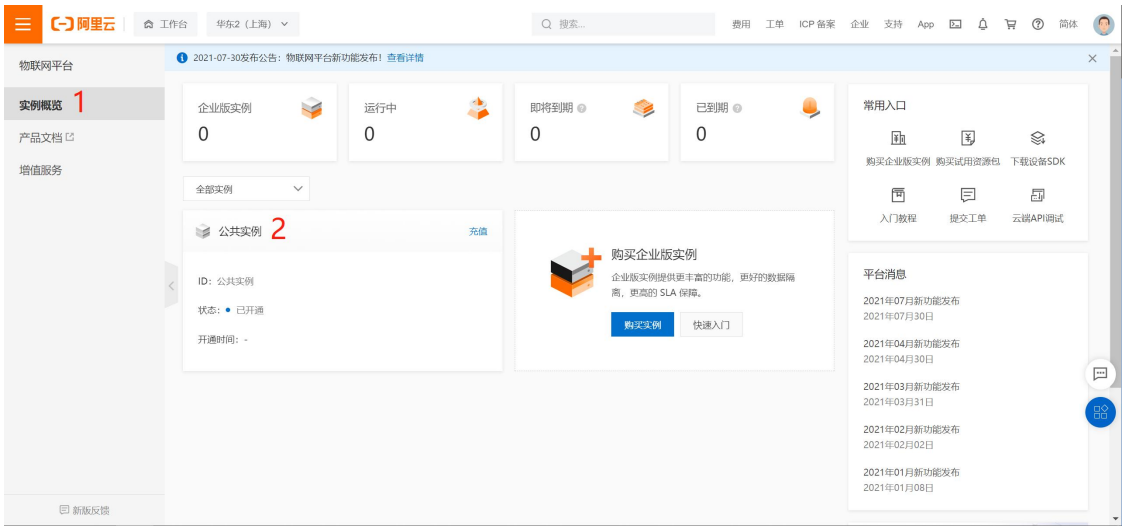
登录进阿里云 web 端后，单击控制台

进入控制台后，在**全部产品与服务**中搜索物联网平台，点击进入即可

进入平台后点击**实例概览**

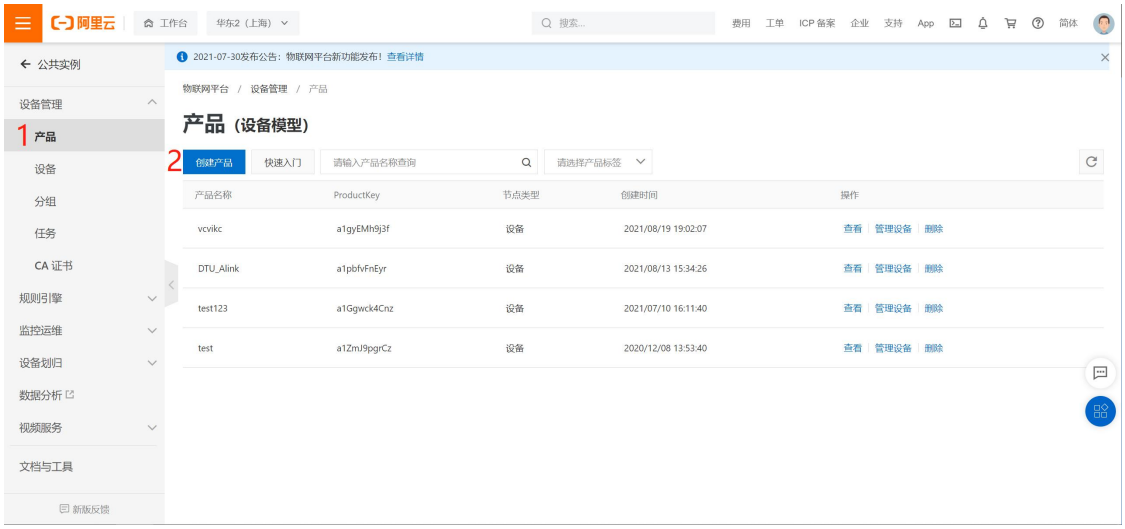
2.公共实例与企业实例的选择

关于公共版实例与企业版实例的区别可以点击这里查看， 这里为了演示方便，我们选择了公共版实例



3.创建产品

进入公共版实例以后点击**产品--->创建产品**，默认无产品，截图中的几个产品是我自己创建的

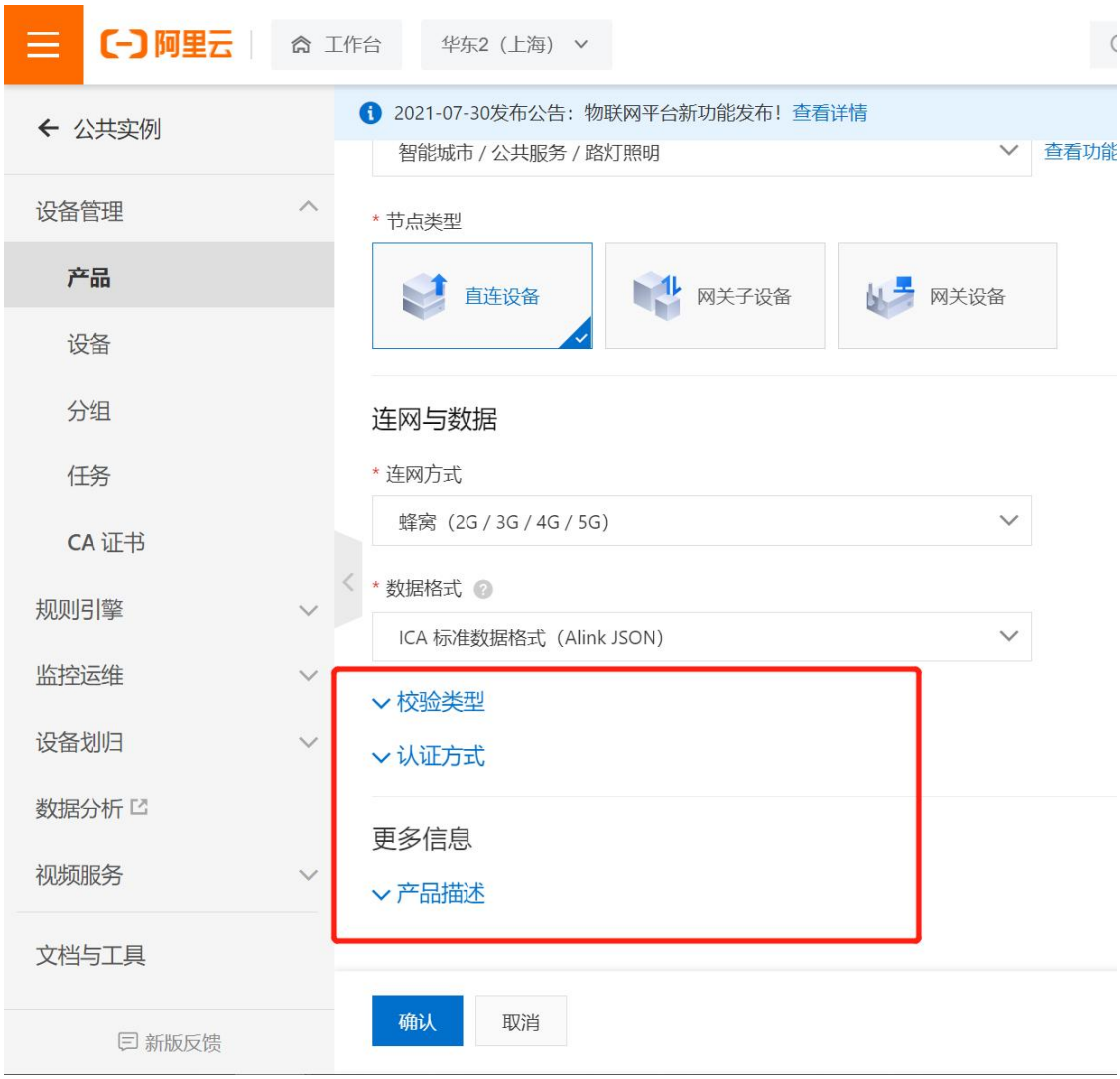


产品名称用户自定义即可 **所属品类**根据用户使用场景自由选择，方便演示这里选择第一个

节点类型选择**直连设备**，如下是三种设备的简介。

直连设备：直连物联网平台，但不能挂载子设备的设备网关子设备：作为网关的子设备，由网关代理连接物联网平台。网关设备：可以挂载子设备的直连设备。网关可以管理子设备，维持与子设备的拓扑关系，并将该拓扑关系同步到云端。

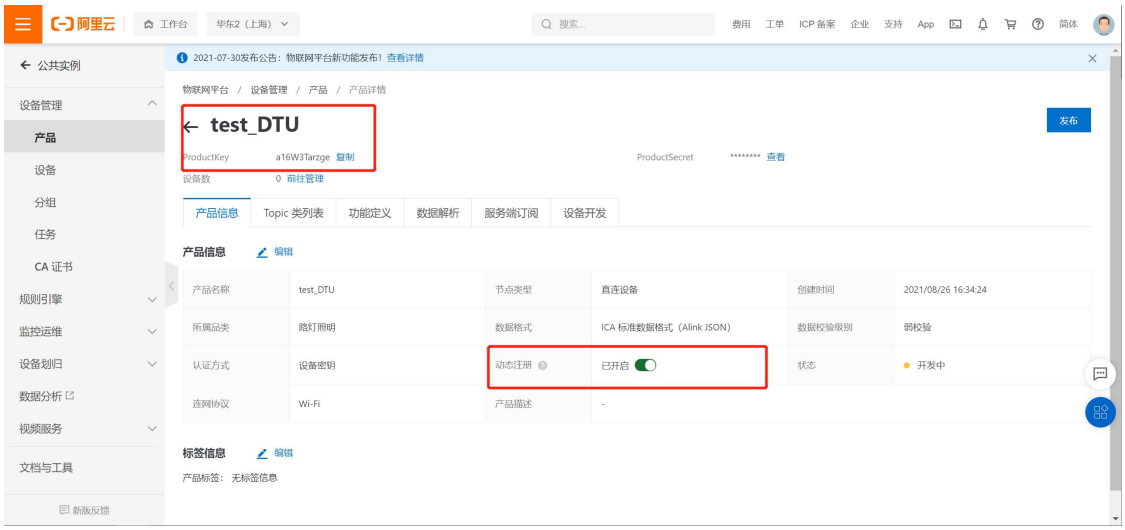
联网方式这里，因为我们使用的是 Cat.1/Cat.4/2G 模块，所以要选择**蜂窝(2G/3G/4G/5G)**。 **数据格式**这里，由于本文主要讲的是**自动注册**，所以选择**透传**或者 **Alink JSON** 没有区别,如果要使用阿里云的物模型功能，需要勾选 **Alink JSON**。 校验类型，验证方式，产品描述这三部分为选填，非必填项，客户根据实际需求选填即可。



点击**确定**后即可创建自己的产品。

创建成功后，因为本章讲解的主要是自动注册功能，所以不用添加任何设备和物模型，直接点击**查看设备详情**即可。

4.查询 productkey



在项目名下即可看到自动注册最为重要的一个 key---**ProductKey** 将 productkey 复制出来备用。

至此，阿里云平台配置工作完成，接下来做 iRTU 的 web 端配置

iRTU web 端配置

基本参数的配置和网络通道参数中个参数的含义与解释在前面的章节中有详细的说明，在这里不做过多讲解，用户根据自己情况进行选择即可。 自动注册到阿里云最为重要的三个参数，**ProductKey、AccessKey ID、Access Key Secret** 前面讲过怎么获取了，将这三个参数填入对应位置即可。

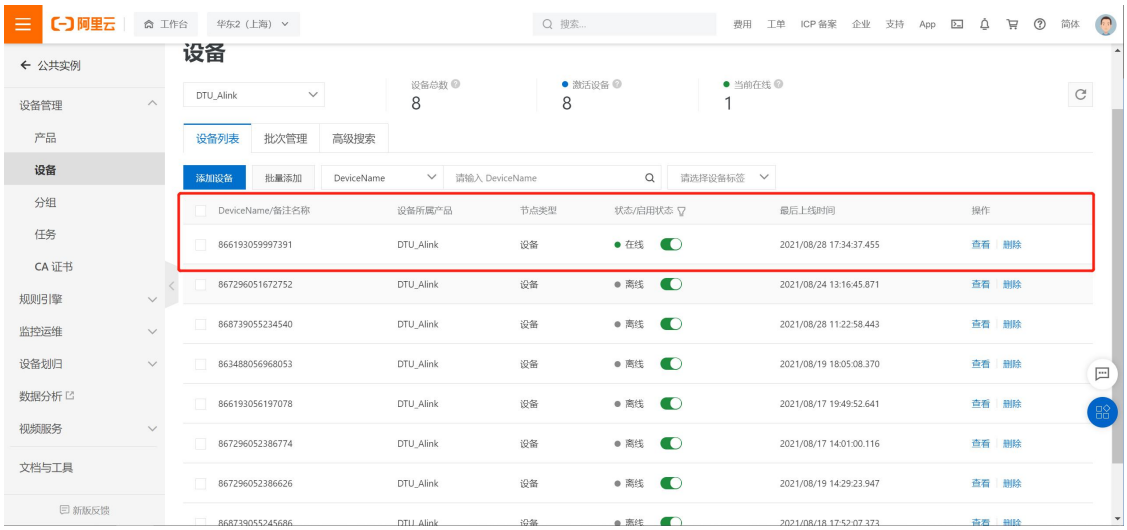
单击确定，即可完成所有配置。

接下来给模块上电/重启，在设备上电情况下也可以通过串口/其他服务器端给模块发送 rrpc,upconfig 命令（串口端还可以发送+++命令），让设备读取到最新的参数配置即可。

当模块检测到参数配置有更新的情况下，会去跟新自己的参数，然后重启设备。重启之后就会去链接阿里云，如果这时你的设备开启了日志打印且设备直连电脑，luatools 打开且日志输出正常的情况下，会观察到 luatools 打印了下面两张截图的红框部分，如果都打印了，即证明你的设备连接阿里云成功。

```
[2021-08-28 17:21:25.353] [D]-[socket.recv] 312 nil
[2021-08-28 17:21:25.359] [I]-[httpv2.response code:] 404
[2021-08-28 17:21:25.365] [I]-[socket:sock_close] 2
[2021-08-28 17:21:25.365] [I]-[socket:on_response:] 2 SOCKET_CLOSE 0
[2021-08-28 17:21:25.365] [W]-[---------- Aliyun iot is start! ----------]
[2021-08-28 17:21:25.430] [D]-[socket.recv] 48 nil
[2021-08-28 17:21:25.461] [I]-[ril.sendat] AT+CCLK="21/08/28,17:21:26+32"
[2021-08-28 17:21:25.461] [I]-[socket:sock_close] 1
[2021-08-28 17:21:25.461] [I]-[socket:on_response:] 1 SOCKET_CLOSE 0
[2021-08-28 17:21:25.461] [I]-[ntp.timeSync is date:] 2021/8/28,17:21:26
[2021-08-28 17:21:25.509] [I]-[ril.proatc] OK
[2021-08-28 17:21:25.509] [I]-[ AutoTimeSync is Done!]
[2021-08-28 17:21:26.004] [W]-[uart1.write data length:] true 1
[2021-08-28 17:21:26.004] [W]-[UART_1 writing ...]
[2021-08-28 17:21:26.004] [W]-[UART_1write done!]
[2021-08-28 17:21:26.390] lualibc_fopen fail -4200062,0,/alikey.cnf
[2021-08-28 17:21:26.390] [I]-[io.exists:] /alikey.cnf: strerror:-4200062
[2021-08-28 17:21:26.467] [I]-[socket:connect-coreid,prot,addr,port,cert,timeout] 0 TCPSSL iot.cn-shanghai.aliyuncs.com 443 nil 120
[2021-08-28 17:21:26.999] [I]-[打印占用的内存:] 743.3984375
[2021-08-28 17:21:26.999] [I]-[打印可用的空间] 1298000
[2021-08-28 17:21:27.231] [I]-[socket:on_response:] 0 SOCKET_CONNECT 0
[2021-08-28 17:21:27.231] [I]-[socket:connect: connect ok]
[2021-08-28 17:21:27.231] [I]-[发送的http报文:] GET /?AccessKeyId=[illegible]Action=RegisterDevice&DeviceName=[illegible]&Format=J
[2021-08-28 17:21:27.231] SON&ProductKey=[illegible]gionId=cn-shanghai&Signature=[illegible]&SignatureMethod=HMAC-SHA1&SignatureNo
```

当然，你也可以在自己的阿里云平台的项目列表里看见你的在线设备。



至此，阿里云自动注册全过程完成。

12.3.5.15 预警

iRTU 提供了 GPIO/ADC/VBATT 触发三种预警方式供用户选择，预警又可以发送到服务器(又称互联网 88)/手机(短信/电话)，本文会详细的讲解这三种预警方式的使用方法，并提供教程。

本文提及的所有触发，为方便演示，链接的服务器都是合宙测试服务器，用户根据当前项目实际需求，自行选择服务器发送预警信息。 关于电话和短信的预警信息提示，需要当前设备上插的卡为特殊物联网卡(可以打电话或者发短信)或者手机卡，如果任意用户设置过的预警方式被激活，都会发送对应的信息到**白名单客户**的卡上去。 白名单客户的设置在下图所示位置，**预警信息---->远程更新参数和固件**里，用户根据实际需求，自行设置即可。

GPIO 触发

web 端配置

GPIO 触发需要在 web 端内配置如下图所示，点击**预警**，**GPIO 触发上报**勾选启用，具体的 GPIO 号根据客户实际需求自行更改（需要去看对应模块以及自己设计的板子的原理图确定实际 GPIO，这里演示的时候使用的是 724 A13 官方开发板）

触发模式也是根据用户实际需要进行选择，这里为了演示方便，上升与下降沿触发都勾选上了

上报消息可以根据用户服务器能接受并处理的消息自行添加与更改，默认为字符串格式。

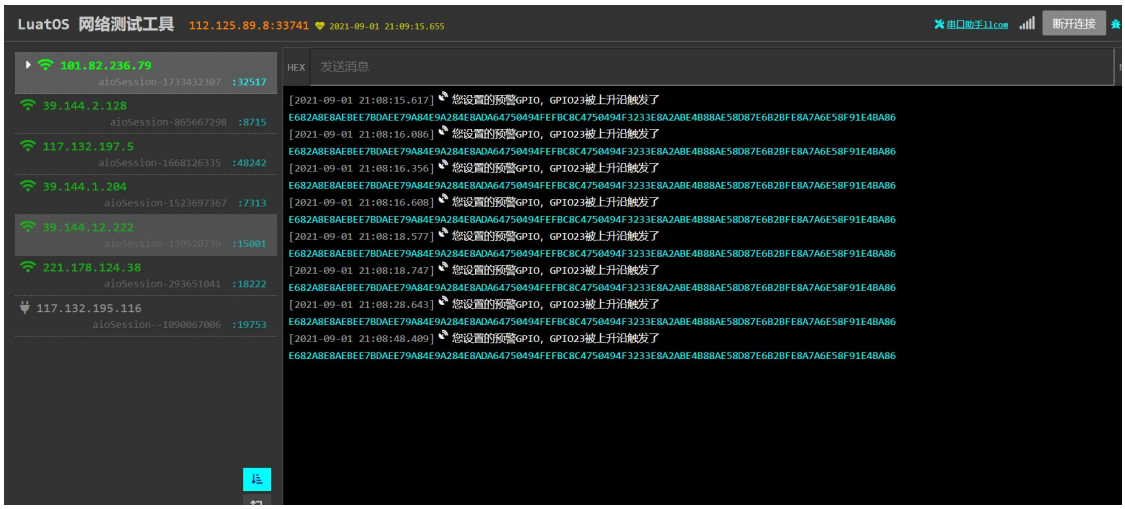
上报通道本项必须根据和用户在**网络通道参数**中设置的通道 ID 一致，否则用户服务器收不到数据

上报方式上报方式有**互联网**(即用户设置的服务器)**短信/电话**(会发送到用户白名单设置的卡号上)

单击确定，即可完成所有配置。

接下来给模块上电/重启，在设备上电情况下也可以通过串口/其他服务器端给模块发送 rrpc,upconfig 命令（串口端还可以发送+++命令），让设备读取到最新的参数配置即可。

当模块检测到参数配置有更新的情况下，会去跟新自己的参数，然后重启设备。重启之后就会去链接对应的服务器，如果此时用户触发了预警的 GPIO 方式且你的设备开启了日志打印并直连电脑，luatools 打开且日志输出正常的情况下，会观察到 luatools 打印了下面截图的红框部分，，即证明你的设备发送预警成功，当然你也可以在对应的**服务器日志**里找到刚刚设备发的信息，如果你设置了**电话**或者**短信**，相应**白名单**对应的号主，就能收到**电话**或者**短信**。


LuatOS 网络测试工具
112.125.89.8:33741
2021-09-01 21:09:15.655
串口助手1lcom
断开连接
101.82.236.79
aioSession-1733432307 :32517
39.144.2.128
aioSession-865667298 :8715
117.132.197.5
aioSession-1668126335 :48242
39.144.1.204
aioSession-1523697367 :7313
39.144.12.222
aioSession-139520739 :15001
221.178.124.38
aioSession-293651041 :18222
117.132.195.116
aioSession--1090067006 :19753
HEX
发送消息
[2021-09-01 21:08:15.617] 您设置的预警GPIO, GPIO23被上升沿触发了
E682A8E8AEBEE7BDAEE79A84E9A284E8ADA64750494FEFBC8C4750494F3233E8A2ABE4B88AE58D87E6B2BFE8A7A6E58F91E4BA86
[2021-09-01 21:08:16.086] 您设置的预警GPIO, GPIO23被上升沿触发了
E682A8E8AEBEE7BDAEE79A84E9A284E8ADA64750494FEFBC8C4750494F3233E8A2ABE4B88AE58D87E6B2BFE8A7A6E58F91E4BA86
[2021-09-01 21:08:16.356] 您设置的预警GPIO, GPIO23被上升沿触发了
E682A8E8AEBEE7BDAEE79A84E9A284E8ADA64750494FEFBC8C4750494F3233E8A2ABE4B88AE58D87E6B2BFE8A7A6E58F91E4BA86
[2021-09-01 21:08:16.608] 您设置的预警GPIO, GPIO23被上升沿触发了
E682A8E8AEBEE7BDAEE79A84E9A284E8ADA64750494FEFBC8C4750494F3233E8A2ABE4B88AE58D87E6B2BFE8A7A6E58F91E4BA86
[2021-09-01 21:08:18.577] 您设置的预警GPIO, GPIO23被上升沿触发了
E682A8E8AEBEE7BDAEE79A84E9A284E8ADA64750494FEFBC8C4750494F3233E8A2ABE4B88AE58D87E6B2BFE8A7A6E58F91E4BA86
[2021-09-01 21:08:18.747] 您设置的预警GPIO, GPIO23被上升沿触发了
E682A8E8AEBEE7BDAEE79A84E9A284E8ADA64750494FEFBC8C4750494F3233E8A2ABE4B88AE58D87E6B2BFE8A7A6E58F91E4BA86
[2021-09-01 21:08:28.643] 您设置的预警GPIO, GPIO23被上升沿触发了
E682A8E8AEBEE7BDAEE79A84E9A284E8ADA64750494FEFBC8C4750494F3233E8A2ABE4B88AE58D87E6B2BFE8A7A6E58F91E4BA86
[2021-09-01 21:08:48.409] 您设置的预警GPIO, GPIO23被上升沿触发了
E682A8E8AEBEE7BDAEE79A84E9A284E8ADA64750494FEFBC8C4750494F3233E8A2ABE4B88AE58D87E6B2BFE8A7A6E58F91E4BA86

9:08
9月1日 周三 晚上
2.4K/s
下午9:08
[4条]您设置的预警GPIO，GPIO23被上升沿触发了
已读
删除
回复
电信
忽略
接听

```
[2021-09-01 21:08:15.618] [D]-[socket.send] total 52 bytes first 30 bytes 您设置的预警GPIO，GPIO2
[2021-09-01 21:08:15.649] [I]-[ril.proatc] >
[2021-09-01 21:08:15.649] [I]-[ril.procatc.send] 0011100B81[illegible]008003060A88BBE7F6E76849848B66004700500049004FFF0C004700500049004F0032003388AB4E
[2021-09-01 21:08:15.649] 0A53476CBF89E653D14E86
[2021-09-01 21:08:15.681] [I]-[socket:on_response:] 0 SOCKET_SEND 0
[2021-09-01 21:08:15.898] [I]-[ril.proatc] +CMGS: 1
[2021-09-01 21:08:15.898] [I]-[ril.proatc] OK
[2021-09-01 21:08:15.898] [I]-[lib_sms rsp] +CMGS AT+CMGS=62 true OK +CMGS: 1
[2021-09-01 21:08:15.928] [I]-[sendcnf] true
[2021-09-01 21:08:15.928] [I]-[ril.sendat] ATD1[illegible];
[2021-09-01 21:08:15.928] [I]-[ril.proatc] OK
[2021-09-01 21:08:15.959] [I]-[ril.defrsp] ATD1[illegible]; true OK nil
[2021-09-01 21:08:15.959] [I]-[ril.sendat] ATD1[illegible];
[2021-09-01 21:08:16.113] [D]-[socket.send] total 52 bytes first 30 bytes 您设置的预警GPIO，GPIO2
[2021-09-01 21:08:16.144] [I]-[ril.proatc] ERROR
[2021-09-01 21:08:16.144] [I]-[ril.defrsp] ATD1[illegible]; false ERROR nil
[2021-09-01 21:08:16.144] [I]-[ril.sendat] ATD1[illegible];
[2021-09-01 21:08:16.144] [I]-[socket:on_response:] 0 SOCKET_SEND 0
[2021-09-01 21:08:16.175] [I]-[ril.proatc] ERROR
[2021-09-01 21:08:16.175] [I]-[ril.defrsp] ATD1[illegible]; false ERROR nil
[2021-09-01 21:08:16.362] [D]-[socket.send] total 52 bytes first 30 bytes 您设置的预警GPIO，GPIO2
[2021-09-01 21:08:16.362] [I]-[ril.sendat] ATD1[illegible];
```

ADC 触发

WEB 端配置

ADC 有两个通道，ADC0 和 ADC1，两个触发的配置方式完全一致，在这里就只描述一组，如果用户想使用两组 ADC 通道，按照第一组的描述自由选择即

可。 和上文 GPIO 触发相同，ADC 触发也在**预警**下进行设置。 点击启用后，即可进行 ADC 触发的配置。

触发模式:有低电压触发和高电压触发两种模式，用户根据实际情况需要，自行选择。 在这里解释下**回差电压**的意思：当你设置了低电压预警后，设备在 ADC 采集到的电压低于预设的电压时会进行预警，回差电压的意思是，当你给设备进行充电操作一段时间后可能会发现 ADC 采集到的电压已经高于预警电压，但是设备仍然在发送预警信息给预设的服务器，这时候就可以设置一个回差电压，当 ADC 采集到的电压高于回差电压后，设备就不会发送预警信息到预设服务器了，所以这里的回差电压，建议设置的比预警电压高至少 100mv。

上报消息：这个和上文的上报消息一样，用户根据实际情况进行设置即可。

上报通道：本项必须根据和用户在**网络通道参数**中设置的通道 ID 一致，否则用户服务器收不到数据

上报方式:上报方式有**互联网**(即用户设置的服务器)**短信/电话**(会发送到用户白名单设置的卡号上)

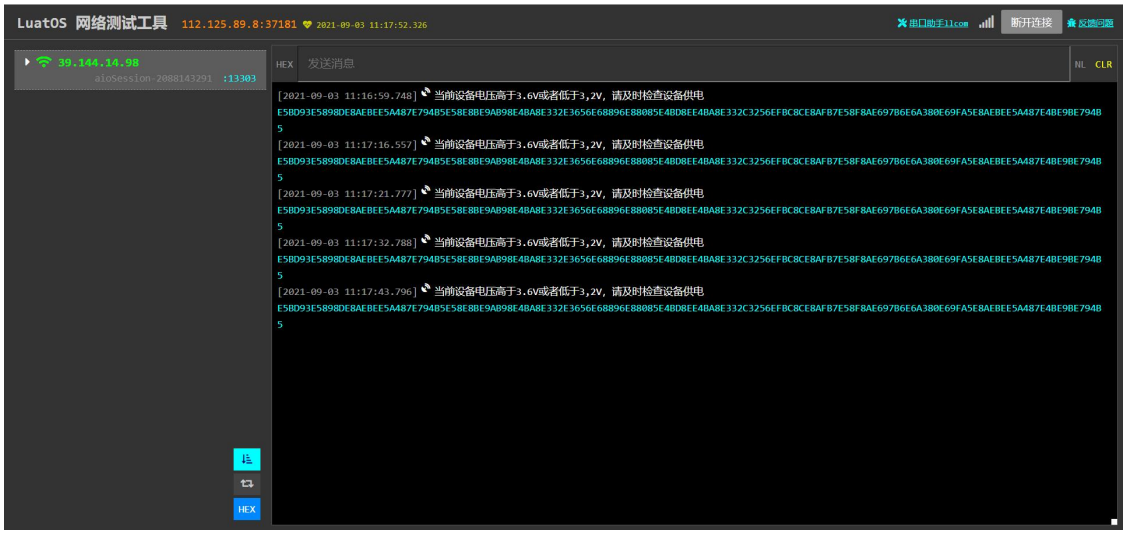
检测频率：设置检测 ADC 采集电压值的检测频率，例如这里设置的 10S，在 10S 内，电压从 3400mv 跌落到 3100mv，10s 内无预警信息，等到 10s 后，才会发送预警信息。

警报间隔:设置上报信息的发送间隔，例如这里设置的 10S，那么当电压低于/高于预警值，每隔 10s，设备就会上发一次预警信息给预设服务器。

全部配置完成后点击确定即可。

接下来给模块上电/重启，在设备上电情况下也可以通过串口/其他服务器端给模块发送 rrpc,upconfig 命令（串口端还可以发送+++命令），让设备读取到最新的参数配置即可。

当模块检测到参数配置有更新的情况下，会去跟新自己的参数，然后重启设备。重启之后就会去链接对应的服务器，如果此时用户触发了预警的 **ADC 方式**且你的设备开启了日志打印并直连电脑，luatools 打开且日志输出正常的情况下，会观察到 luatools 打印了下面截图的红框部分，，即证明你的设备发送预警成功，当然你也可以在对应的**服务器日志**里找到刚刚设备发的信息，如果你设置了**电话**或者**短信**，相应**白名单**对应的号主，就能收到**电话**或者**短信**,短信和电话方式通知结果和上文 GPIO 触发一致，这里就不演示效果了。



VBATT 触发上报

所有配置和 ADC 触发相同，在这里仅放一张配置截图和实现效果截图，用户根据 ADC 触发配置，自行设置各项参数即可。

至此，预警功能演示完毕，用户根据实际需要进行预警方式的选择。

12.4 errDump 远程日志

12.4.1 概述

LuatOS 错误日志上报功能模块名叫：errDump，errDump 对"量产投放市场的设备，远程调试初步定位问题"至关重要， 强烈建议客户一定要使用此功能。

12.4.2 原理

errDump 就是将模块运行过程中产生的错误信息或者应用日志通过 TCP/UDP 上报到互联网上的指定服务器，技术人员可以在服务器上查阅日志，协助远程了解设备运行情况，或者故障诊断。

12.4.3 测试脚本

可以直接用 DEMO 测试：errDump

接口说明：errDump - 错误上报

下面是用 DEMO 中的自动发送作为演示：

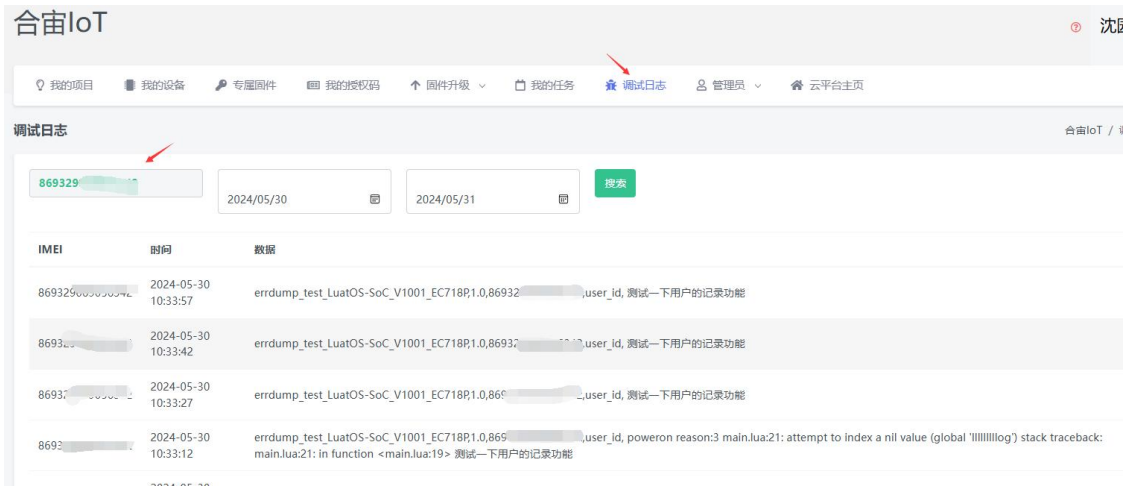
测试脚本

```
PROJECT = "errdump_test"
VERSION = "1.0"
PRODUCT_KEY = "XXX" --换成自己的
-- sys 库是标配
_G.sys = require("sys")
_G.sysplus = require("sysplus")
log.style(1)

--下面演示自动发送
 errDump.config(true, 600, "user_id")    -- 默认是关闭，用这个可以额外添加用户标识，比如用户自定义的 ID 之类

 local function test_user_log()
     while true do
         sys.wait(15000)
         errDump.record("测试一下用户的记录功能")
     end
 end

 local function test_error_log()
     sys.wait(60000)
      lllllllllog.record("测试一下用户的记录功能") --默认写错代码死机
 end
sys.taskInit(test_user_log)
sys.taskInit(test_error_log)
sys.run()
```

12.4.4 测试效果

模块打印日志

云平台对应日志



12.5 FOTA 远程升级

合宙 IOT 平台，点此登录

12.5.1 帮助中心

登录系统后，可以点击后右上角的？进入帮助中心，阅读相关的说明，如下所示：

12.5.2 FAOTA 应用指南

AT 开发方式参考：

9.5.5 远程升级(FOTA)应用指南

LuatOS 开发方式参考：

10.5.1 FOTA 开发应用指南

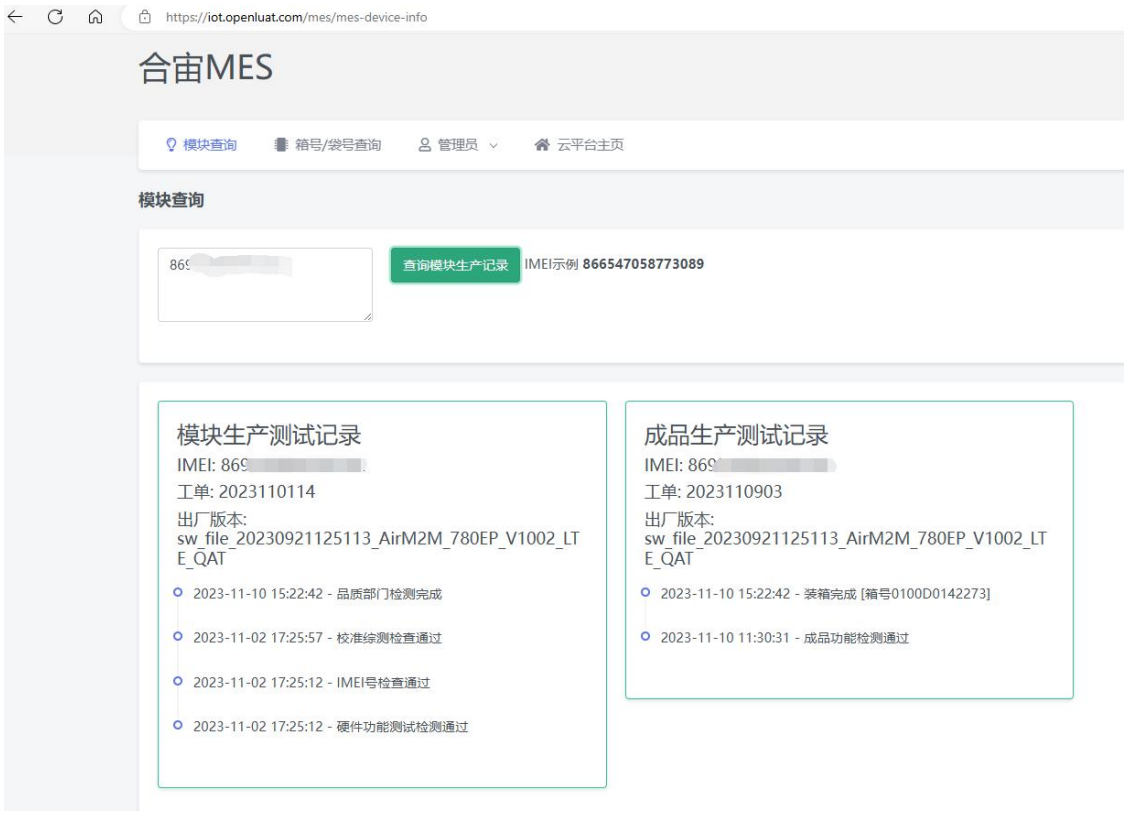
CSDK 开发方式参考：

11.8.2 FOTA 指南（全量升级-合宙 IOT 服务器）

11.8.3 FOTA 指南（差分升级-合宙 IOT 服务器）

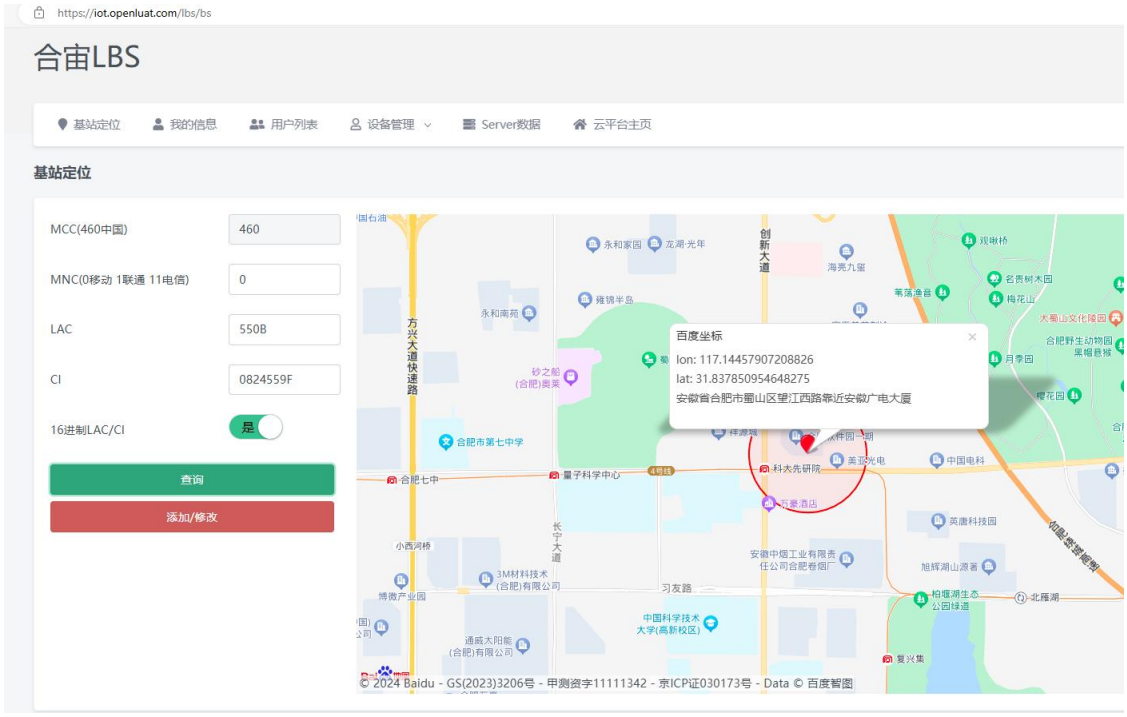
12.6 MES 查询

合宙批量发货的每一个包装箱都有一个唯一的箱号标签。每一个包装箱内的包装的包装袋都有一个唯一的袋号便签。通过 IMEI 或箱号/袋号，您可以查阅模组的生产、测试等相关信息。合宙 MES



12.7 基站查询工具

点击 合宙 LBS 可以根据小区信息查看模块对应位置，如下所示：



注意：任何一个在线查询网站，查询结果仅供参考；无法保证 100%可以成功，因为小区 id 不断在新增，查询网站的后台数据库很难做到实时更新小区 id 数据库。

12.8 基站 WIFI 定位

12.8.1 基站/WIFI 定位原理

模块正常联网后，搜索附近的小区信息，上报给后台；后台查询小区经纬度，并结合信号强度等因素进行计算，最后返回计算出来的经纬度给模块。

WIFI 定位的原理和基站定位原理类似，搜索附近 WIFI 的 MAC 地址信息，上报给后台。后台查询 MAC 地址经纬度，并结合信号强度等因素进行计算，最后返回计算出来的经纬度给模块。

12.8.2 免费和收费服务区别

目前合宙的 4G 模块都支持基站定位，此服务免费。但免费的服务器承受不了高并发压力，为了提供更好的服务，可以联系销售选择收费服务。

免费服务

1. 免费服务仅支持单基站定位,无 wifi 定位，同一 IMEI 的请求间隔需大于 15 秒。
2. 超过频次的定位请求均会返回定位失败。返回错误码，不包含位置信息。

收费服务

1. 收费服务支持多基站定位和 wifi 定位，同一 IMEI 的请求间隔需大于 5 秒。

12.8.3 各固件的配置方式

免费服务

沿用现有的接入参数和流程，但要控制访问频次

1. AT 固件，现有固件均支持
2. Lua 固件，无需修改

收费服务

计费方式：联系销售按 IMEI 数量购买，由销售导入 IMEI 列表，或使用项目 key 的方式批量购买

12.8.4 使用方法

LuatOS 使用方法

demo 见 demo/lbsLoc/main.lua

免费

脚本逻辑是调用 lbsLoc.request(getLocCb)，在 getLocCb 回调函数里就会有查询到的经纬度。

收费

脚本逻辑是设置正确的项目 key（XXX），在 getLocCb 回调函数里就会有查询到的经纬度。

WIFI 定位，需要通过 wlan.scan()获取对应 WIFI 地址，然后在调用 lbsLoc.request()，在 getLocCb 回调函数里就会有查询到的经纬度。

```
-- 以下为基站+wifi 混合定位

sys.subscribe("WLAN_SCAN_DONE", function ()

   local results = wlan.scanResult()

    log.info("scan", "results", #results)

   if #results > 0 then

        local reqWifi = {}

       for k,v in pairs(results) do

            log.info("scan", v["ssid"], v["rssi"], v["bssid"]:toHex())

            local bssid = v["bssid"]:toHex()

            bssid = string.format ("%s:%s:%s:%s:%s:%s", bssid:sub(1,2),
bssid:sub(3,4), bssid:sub(5,6), bssid:sub(7,8), bssid:sub(9,10),
bssid:sub(11,12))

           reqWifi[bssid]=v["rssi"]

        end

        lbsLoc.request(getLocCb,nil,nil,"XXX",nil,nil,nil,reqWifi)

    else
```

```
        lbsLoc.request(getLocCb) -- 没有wifi 数据,进行普通定位
    end
end)


sys.taskInit(function()
    sys.waitUntil("IP_READY", 30000)
    wlan.init()
    while 1 do
        wlan.scan()
        sys.wait(60000)
    end
end)
```

AT 使用方法

AT 使用方法见：Luat4G 模块 EC618&EC716&EC718 系列 AT 命令手册

免费

参考如下章节说明

5.17 读取基站定位（LBS）信息和时间：AT+CIPGSMLOC

收费

发 AT+WIFILOC=1,1,1 这个定位需要 WIFI 扫描，要选择 WIFI 优先，不然模块处于连接态时 WIFI 扫不到

12.8.5 常见问题

1. 基站定位精度如何？

搜索到的小区越多，定位的精度越高；一般来说，城市中心定位精度比郊区和农村定位精度高，城市中心的定位精度在几十米到几百米不等，郊区和农村的定位精度更低，可能会有几千米甚至更多的误差。

通过多小区定位，将实时搜索到的所有小区同时上报给后台参与定位，基站定位的误差都比较大，如果需要准确定位，请使用支持 GPS 的模块，例如 Air780EG、Air510U。

2. wifi 定位精度？

根据设备获取的 WiFi 的信息进行定位，WIFI 定位精度一般不受使用环境影响，主要和单一 WIFI 辐射半径、WIFI 覆盖密度有关。一般来说，WIFI 精度在 3 米－200 米左右。

3. 为什么基站定位失败？

a、 main.lua 中的 PRODUCT_KEY 和此设备在 iot.openluat.com 中所属项目的 ProductKey 必须一致，请去检查。

b、 后台基站数据库查询不到所有小区的位置信息；在 trace 中向上搜索 mcc,mnc,lac,ci，然后在电脑浏览器中打开定位查询，手动查找 mcc,mnc,lac,ci 后的所有小区位置；如果手动可以查到位置，则服务器存在 BUG，直接向技术支持人员反映问题；如果手动无法查到位置，则基站数据库还没有收录当前设备的小区位置信息，向技术支持人员反馈，我们会尽快收录。

c、达到了总的 qps 限制数量。

4. 合宙官方的基站/WIFI 定位能在国外使用吗？

不能，没有国外的基站数据库。

5. 基站/WIFI 定位获取的经纬度是什么格式的？

基站定位获取的经纬度为 WGS-84 格式。

各种坐标系说明以及转换方法参考：http://old.openluat.com/GPS-Offset.html

6. 当同时使用基站定位和 wifi 定位怎么判断是基站定位成功了还是 wifi 定位成功了？

根据返回的 locType 判断，为 0 表示基站定位成功，返回 255 表示 WIFI 定位成功。

12.9 坐标转换

OpenLuat 的所有 GNSS 模块 (立即购买) 均使用国际标准 **WGS-84** 坐标系，所以开发者在国内常见地图定位时，会发现与实际情况有几十米的误差。这并非模块问题， 而是国内地图采用了**非标坐标系**所致。

国内常见地图如高德地图使用 **GCJ-02** 坐标系， 百度地图使用 **BD-09** 坐标系，故此开发者需要对模块输出的经纬度进行加偏处理，才能在国内的地图上实现精确定位。

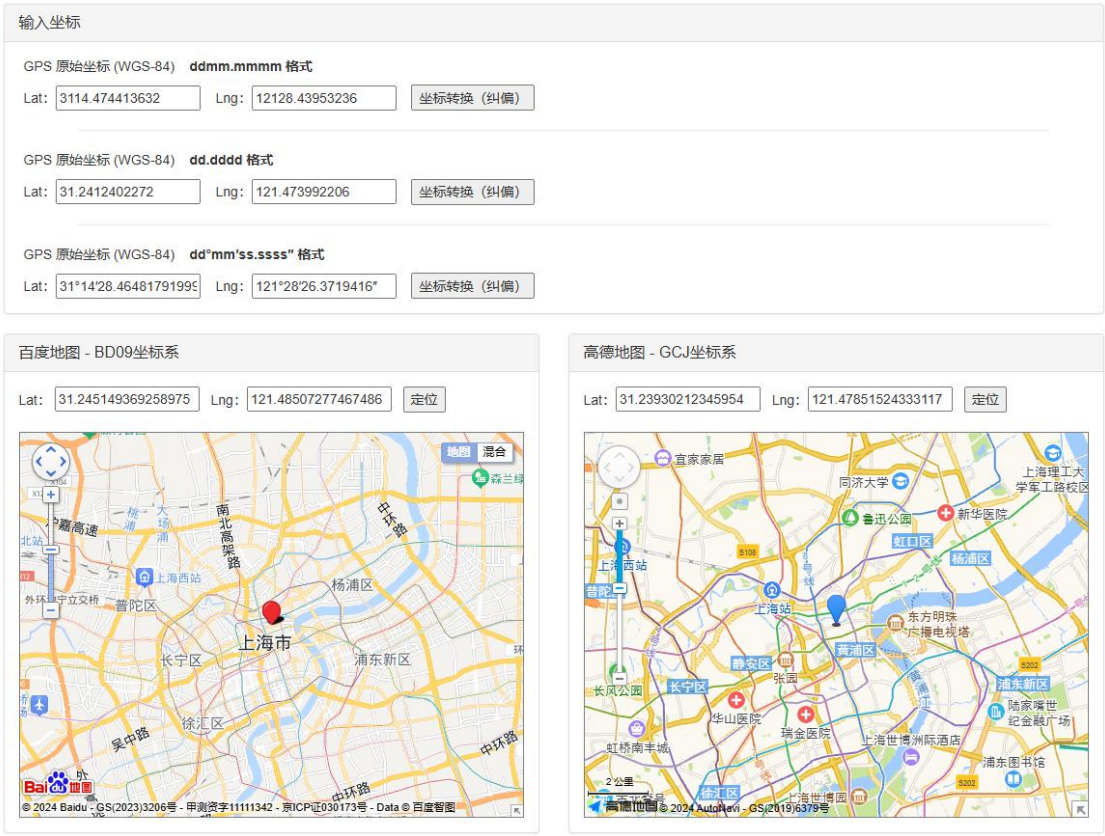
本页面展示了原始 GPS 坐标 (WGS-84)，分别转换为 GCJ02 坐标 和 BD09 坐标 的定位效果 (代码下载)

1、WGS-84 通常叫做“原始 GPS 坐标”。是从硬件设备默认采集到的坐标。GGA 输出格式为 ddmm.mmmm，应转换为 dd.dddd 格式。

2、国外的地图软件均使用该坐标系（如 Google Maps、Bing Maps, Here Maps），中国区的“谷歌地图”除外。

3、GCJ02 俗称“火星坐标”。高德地图，腾讯地图，谷歌地图（中国区域）使用该坐标。

4、其他搜狗地图、百度地图、图吧地图等，均采用其他自定义坐标。



12.10 网络测试工具

开发调试平台（不保证平台系统的稳定性，仅方便开发者开发调试使用）

TCP 测试平台：https://netlab.luatos.com

十三 合宙产品选型手册

可通过点击右方链接在线阅读：合宙产品选型手册

也可直接阅读本章节以下内容：

合宙
通信
上海合宙Cat.1模组/功耗分析仪
产品选型手册
Air700ECQ
超小模组
让你的产品设计更灵活
Air9000P
功耗分析仪
让你的产品更有价值
低功耗模式
超低功耗方案
让你的产品续航更持久
FOTA
远程升级
让你的产品运维更便捷
降功耗,找合宙!
WWW.OPENLUAT.COM

模组型号	AT指令	LuatOS 二次开发	USB上网 RNDIS/PPP	VoLTE 语音通话	GNSS定位	超薄1.7mm	低功耗	适用区域
Air700ECQ	√	√				√	★★★	中国移动
Air700EAQ	√	√				√	★★★	中国电信
Air700EMQ	√	√				√	★★★	中国联通
Air780EQ	√	√					★★★	中国
Air780EP	√	√					★★★★★	中国
Air780EPS	√	√				√	★★★★★★	中国
Air201		√			√		★★★★★★	中国
Air780E	√	√	√				★★★★	中国
Air780EX	√	√	√				★★★★	中国
Air780EG	√	√	√		√		★★★★	中国
Air724UG	√	√	√	√			★★	中国
Air780ER	√		√				★★★	中国
Air780EEN	√		√				★★★	北美
Air780EEU	√		√				★★★	欧亚
Air780EEJ	√		√				★★★	日本

怎么选择4G模组?
合宙
通信
怎么选择4G模组
首先需要说明的是， 没有十全十美的模组；每个模组都有各自的特色，都有各自的不可替代性；同时，每个模组也都有一定的局限性。
您可以根据自己产品的需要，从如下几个维度来做选择：
OpenCPU开发能力排行
AT指令模组推荐
模组功耗排行
模组大小排行
模组厚度排行
模组价格排行
海外模组推荐
USB上网模组
通信定位二合一模组
综合型号推荐
降功耗,找合宙!
WWW.OPENLUAT.COM

怎么选择4G模组?
合宙
通信
OpenCPU开发能力强弱排行
1
Air724UG
2
Air780E
Air780EX
Air780EG
3
Air780EPS
4
Air780EP
5
Air700ECQ
Air700EAQ
Air700EMQ
6
Air780EQ
Air201
特别说明:
这里所说的OpenCPU，特指合宙基于Lua脚本虚拟机研发的LuatOS开发方式;
Air780ER/Air780EEN/Air780EEU/Air780EEJ，只能支持AT命令，不支持OpenCPU;
Air201具备支持AT命令接口的能力，但其因为集成外设功能较多，因此只推荐LuatOS开发方式;
除Air201外，所有其它型号都支持AT命令开发方式。
降功耗,找合宙!
WWW.OPENLUAT.COM

功能点	Air780EQ	Air700ECQ/Air700EAQ/Air700EMQ	Air780E	Air780EG	Air780EP	Air780EPS	Air724UG	备注
总Flash	4M	4M	5MB	5MB	4MB	8MB	8M	
用户代码可用Flash（含脚本和脚本FOTA）	1.4M	1.4M	>1.6MB	>1.6MB	<1.6MB	<5MB	>4M	
总RAM	1M	1M	2MB(SRAM)	2MB(SRAM)	2MB(SRAM)+2MB(PSRAM)	2MB(SRAM)+2MB(PSRAM)	16M(SRAM)	
TCP/UDP	2路	2路	6路	6路	6路	6路	8路	
TCP-SSL/TCP-TLS	1路	1路	2路	2路	2路	2路	4路	
MQTT	√	√	√	√	√	√	√	mqtt通信
HTTP	√	√	√	√	√	√	√	http通信
FTP	√	√	√	√	√	√	√	ftp通信
WEBSOCKET	√	√	√	√	√	√	√	websocket通信
JSON	√	√	√	√	√	√	√	json
NTP网络对时	√	√	√	√	√	√	√	时钟,必带
SMS短信	√（不支持电信）	√（不支持电信）	√（不支持电信）	√（不支持电信）	√（不支持电信）	√（不支持电信）	√	短信功能
阿里云/腾讯云/百度云/华为云	√	√	√	√	√	√	√	各种云平台的密钥计算
REPL控制台repl							√	控制台repl
PROTOBUF			√	√	√	√	√	谷歌PB编解码
RSA加密			√	√	√	√	√	RSA加密
XXTEA加密			√	√	√	√	√	XXTEA加密
国密算法			√	√	√	√	√	国密加密
加密解密md5/sha1/aes	√	√	√	√	√	√	√	加密/解密,md5/sha1之类
64位数据处理	√	√	√	√	√	√		64位数据的处理库
ICONV字符集转换	√	√	√	√	√	√	√	字符编码转换
ZBUFF(C内存数组)	√	√	√	√	√	√		c风格内存数组, tcp/udp需要
PACK数据编解码	√	√	√	√	√	√	√	数据打包解包
zlib解压	√	√	√	√	√	√	√	zlib压缩
内部WDT硬狗	√	√	√	√	√	√	√	硬狗,必带
PM功耗管理	√	√	√	√	√	√	√	功耗管理
低功耗模式	√	√	√	√	√	√		长连接低功耗模式
接口类								
GPIO	√（仅ADNGPIO）	√	√	√	√	√	√	复用较多,数量见具体硬件手册
UART	√（2个）	√（2个）	√（2个）	√（1个）	√（3个）	√（3个）	√（3个）	串口通信/485/少量传感器
I2C		√	√	√	√	√	√	传感器居多
SPI		√	√	√	√	√	√	传感器/各种外设/SPI屏幕
ADC		√	√	√	√	√	√	测外部电压/供电电压/CPU温度
PWM		√	√	√	√	√	√	输出方波,驱动马达
GPS定位				√				GPS定位功能
WIFISCAN	√(共用天线)	√(共用天线)	√(共用天线)	√(共用天线)	√(共用天线)	√(共用天线)	√	WIFI扫描,用于WiFi定位
音频类								
音频播放(MP3)							√	播放AMR/MP3
录音							√	录音
TTS							√	TTS
VOLTE							√	VoLTE
UI类								
SPI串口屏							√	SPI屏幕
墨水屏							√	墨水屏驱动
OLED单色屏							√	OLED驱动
中文字体							√	字体管理
LVGL							√	LVGL
摄像头							√	摄像头功能
扫码							√	

怎么选择4G模组?
合宙
通信
合宙Cat.1模组支持三种低功耗模式
功耗模式 | 常规模式 | 低功耗模式 | PSM+模式
4G在线状态 | 在线，长连接 | 在线，长连接 | 离线，飞行模式
定时器唤醒 | 支持 | 支持 | 支持
中断唤醒 | 支持 | 支持 | 支持
串口唤醒 | 支持 | 支持，唤醒时波特率需先设置为9600bps | 支持，唤醒时波特率需先设置为9600bps
服务器4G唤醒 | 支持，1秒内 | 支持，1秒内 | 不支持
上行发送 | 1秒内响应 | 1秒内响应 | 1.5秒内响应
VEXT电源输出状态 | 保持输出 | 不能保持输出，也不能保持关闭，间歇性输出状态 | 不能保持输出，也不能保持关闭，间歇性输出状态
所有GPIO管脚是否可以控制输出电平 | 可以 | 不可以 | 不可以
常规GPIO管脚是否可以保持电平 | 可以 | 不可以 | 不可以
特殊AGPIO管脚是否可以保持电平 | 可以 | 可以 | 可以
RAM供电及唤醒后软件运行状态 | RAM供电，正常工作，满血状态 | RAM供电，唤醒后保持原状态运行 | RAM掉电，唤醒后程序从初始状态运行(PSM+状态前运行数据丢失)
典型功耗表现 | 较低（4.5mA） | 均衡（0.3mA） | 极低（3μA）
测试环境：
Air780EPS：供电电压3.8V，USB功能关闭，移动网络，频段B3，RSRP值-88附近，DRX 2.56秒，心跳间隔5分钟，心跳数据100Byte，TCP协议，合宙服务器，回环测试；
Air780EPS：同等环境下，低功耗模式，DRX 1.28秒时，平均电流0.5mA，DRX 0.64秒时，平均电流0.7mA；
Air780EPS：同等环境下，常规模式，DRX 1.28秒时，平均电流4.7mA，DRX 0.64秒时，平均电流4.9mA；
DRX，Discontinuous Reception，非连续接收，可简单理解为模块与基站之间保持心跳的间隔，一般为0.64秒/1.28秒/2.56秒，需要注意的是，DRX由基站根据网络实际情况而定，模组无法自行控制；
除Air724UG外，合宙其它所有型号的4G Cat.1模组全部支持低功耗模式和PSM+模式，Air780EPS/Air201表现最佳。
关于常规GPIO和特殊AGPIO的说明：
合宙Air700系列/Air780系列模组在不同功耗模式下不同GPIO的功能表现不一致，其中：
一部分在低功耗模式和PSM+模式下可以保持电平的为特殊AGPIO（有些文档中有时也会被写为AON_GPIO，AlwaysON，一直保持的意思）；
一部分常规GPIO的电平状态在低功耗模式和PSM+模式下不可保持（相对常规模式下常规GPIO电平的运行状态而言），请务必注意；
关于常规GPIO和特殊AGPIO的相关说明，详见具体型号的硬件设计手册或其它资料。
降功耗,找合宙!
WWW.OPENLUAT.COM

怎么选择4G模组?
合宙
通信
模组尺寸从小到大排行
1
对模组大小要求极致的场景推荐使用
超小超薄
10mm × 13mm × 1.7mm
Air700ECQ
中国移动专用
Air700EAQ
中国电信专用
Air700EMQ
中国联通专用
3
集成封装
16mm×32mm
Air201
资产定位模组
4
经典封装
24mm×24mm
Air724UG
多功能模组
2
通用封装16mm×18mm
Air780E
Air780EX
Air780EP
Air780EQ
Air780EPS
Air780EEN
Air780EEU
Air780EEJ
Air780ER
Air780EG
降功耗,找合宙!
WWW.OPENLUAT.COM

怎么选择4G模组?
合宙
通信
合宙USB上网模组推荐
Air780ER
Air780ER
合宙USB上网模组
支持RNDIS/PPP拨号上网
可搭配Windows/Linux操作系统
降功耗,找合宙!
WWW.OPENLUAT.COM

怎么选择4G模组?

合宙 通信

通信定位二合一模组

Air201除了支持4G Cat.1通信和极致低功耗定位二合一之外，还支持G-Sensor震动检测、锂电池充电管理、SIM卡座、USB-C连接器等功能；Air780EG只支持4G Cat.1通信和定位的功能。

Air780EG

16mm×18mm×2.3mm

Air201

16mm×32mm×4.0mm

您也可以选择其它合适的通信模组搭配GNSS定位模组Air510U或Air530Z使用；
Air530Z和Air510U，二者除了尺寸不同之外，没有明显的差异，可以根据自己的尺寸需求选择不同的型号。

Air530Z

12.9mm×9.9mm×2.3mm

Air510U

10.1mm×9.7mm×2.0mm

降功耗,找合宙!

WWW.OPENLUAT.COM

怎么选择4G模组?
合宙
通信
模组支持AT指令的推荐选择
推荐型号
Air700ECQ Air700EAQ Air700EMQ Air780EQ
Air780EP
Air780EPS
Air780E Air780EX
Air780ER
Air780EEN Air780EEU Air780EEJ 【海外版】
Air724UG
Air780EG
AT指令
TTS
VoLTE
GPS
USB上网
模组资料
Air700ECQ
Air700EAQ
Air700EMQ
Air780EQ
Air780EP
Air780EPS
Air780E
Air700EX
Air700ER
Air780EEN
Air780EEU
Air780EEJ
Air724UG
Air780EG
降功耗,找合宙!
WWW.OPENLUAT.COM

怎么选择4G模组?
合宙 通信
模组厚度从薄到厚排行
1
1.7mm超薄系列
对模组厚度要求极致的场景推荐使用
Air700ECQ
中国移动专用
Air700EAQ
中国电信专用
Air700EMQ
中国联通专用
Air780EPS
4G全网通系列
3
厚度4mm
Air201
4G资产定位模组
2
厚度2.3mm
Air780E
Air780EX
Air780EP
Air780EQ
Air780EG
Air780EEN
Air780EEU
Air780EEJ
Air780ER
Air724UG
Air724UG
Air724UG-NFM
降功耗,找合宙!
WWW.OPENLUAT.COM

怎么选择4G模组?
合宙
通信
综合型号推荐
首先，合宙推出的所有模组型号，都是工业品质，都是经过严格的供应链管理和出厂检测，都可以放心大批量使用。
01 如果是追求最极致的低功耗的话，选择Air780EPS和Air201；
02 如果是开发相对复杂的应用，要在模组里面运行相对多的算法，控制更多的外设：Air724UG最合适；
03 如果是想最快速度开发一个定位+通信的完整方案的话：Air201最合适；
04 根据只需要"通信协议+UART通信+GPIO控制"最基础的OpenCPU能力，可以选择Air780EQ；如需再增加SPI、I2C、ADC、PWM等功能，可以选择：Air700ECQ/Air700EAQ/Air700EMQ；
05 如果需要相对复杂的二次开发：Air780EP的价格比 Air780E、Air780EX相对要低一点；Air780EPS的功耗表现最极致；
06 如果只需要最基础的AT指令功能，推荐选择：Air700ECQ/Air700EAQ/Air700EMQ/Air780EQ；
07 通信+定位二合一模组：除Air201外，您还可以选择Air780EG，既支持AT指令，也支持LuatOS二次开发；您也可以选择其它合适的通信模组搭配GNSS定位模组Air510U或Air530Z使用；
08 如果您需要海外模组：可以根据具体的使用区域选择Air780EEN、Air780EEU或Air780EEJ；
09 如果您需要USB上网，国内推荐选择Air780ER，海外推荐选择Air780EEN/Air780EEU/Air780EEJ；
10 Air780EX在LuatOS/AT能力方面与Air780E相当，但Air780EX为LCC封装(邮票孔)更方便手工焊接，管脚数量方面Air780EX要比Air780E少一些。
注： 合宙模组的二次开发，指的都是基于Lua脚本语言研发的LuatOS二次开发。
降功耗,找合宙!
WWW.OPENLUAT.COM

功能点	Air700ECQ Air700EAQ Air700EMQ Air780EQ	Air780ER	Air780EP	Air780E Air780EX	Air780EPS	Air724UG	Air780EEN Air780EEU Air780EEJ	Air780EG	备注说明
基础指令	√	√	√	√	√	√	√	√	
TCP/UDP	√	√	√	√	√	√	√	√	
HTTP	√		√	√	√	√	√	√	
MQTT	√		√	√	√	√	√	√	
FTP	√		√	√	√	√	√	√	
NTP	√	√	√	√	√	√	√	√	NetworkTime Protocol网络时间协议
SSL	√	√	√	√	√	√	√	√	TLS/SSL配置
FS	√		√	√	√	√	√	√	建立文件，枚举文件，传输TLS/SSL证书
SMS	√		√	√	√	√	√	√	短信功能，Air724UG支持移动联通电信，其余型号仅支持移动联通，电信不支持。
Wi-Fi Scan	√		√	√	√	√	√	√	Wi-Fi扫描，用于定位
LBS	√		√	√	√	√	√	√	基站定位
GPS								√	定位功能
TTS						√			Text To Speech
VoTLE						√			语音通话
PPP		√		√		√	√	√	PPP拨号上网
USB_RNDIS_ECM		√		√		√	√	√	USB网络驱动

怎么选择4G模组?
合宙
通信
模组价格从低到高排行
-以下价格来自于合宙官方淘宝店 Luat.taobao.com，更多信息请联系合宙销售经理-
Air780EQ
合宙4G模组
¥17.9
合宙通用经典封装
Air700ECQ
合宙4G模组
¥19.9
超小超薄，合宙专属
Air700EAQ
合宙4G模组
¥19.9
超小超薄，合宙专属
Air700EMQ
合宙4G模组
¥19.9
超小超薄，合宙专属
Air780ER
合宙4G模组
¥19.9
合宙通用经典封装
Air780EP
合宙4G模组
¥19.9
合宙通用经典封装
Air780E
合宙4G模组
¥25.9
累计2000万+出货
Air780EX
合宙4G模组
¥26.9
合宙通用经典封装
Air780EPS
合宙4G模组
¥33.9
极致低功耗，更低更稳定
Air724UG
合宙4G模组
¥42.9
累计服务数万家客户
Air780EG
合宙4G模组
¥49.9
文档完善，开发无忧
超低功耗
资产定位开发包
¥299
超低功耗、全套硬件、赠送配件
降功耗,找合宙!
WWW.OPENLUAT.COM

合宙
通信
除了Cat.1模组，
合宙还有哪些产品线？
简单来说，可以分为如下4个部分：
01
DTU
整机+后台+小程序
02
资产定位
整机+后台+小程序+APP
03
仪 表
功耗分析仪+上位机
04
云服务
合宙云端增值服务系列
降功耗,找合宙!
WWW.OPENLUAT.COM

怎么选择4G模组?

合宙 通信

综合功耗从低到高排行

合宙低功耗4G Cat.1模组支持三种功耗模式（下一页详细介绍）：

功耗模式	常规模式	低功耗模式	PSM+模式
网络状态	在线	在线	离线
服务器命令	随时响应	随时响应	无法响应
CPU	满频运行	满频运行	降频运行
外设功能	全部可用 所有GPIO电平都可以控制	部分可用 仅有AGPIO可以保持电平	部分可用 仅有AGPIO可以保持电平

降功耗,找合宙!

WWW.OPENLUAT.COM

降功耗,找合宙!

WWW.OPENLUAT.COM

合宙 通信

合宙功耗分析仪Air9000

01. 电源功能

电压输出	不支持
电流限制	不支持
供电方式	USB-C/电源线/5~24V
特别说明	可搭配快充充电头(支持PD/PPS协议)或直流稳压电源作为电源使用

02. 测试功能

电压量程	0~24V
电压精度	10mV
电流量程	0~5A
电流精度	1μA
最大功率	120W
供电方式	USB-C/电源线/5~24V
硬件采样率	80KHz
上位机采样率	10KHz
上位机软件	Windows7及以上
API接口	开放动态库API接口

降功耗,找合宙!

WWW.OPENLUAT.COM

合宙 通信

合宙功耗分析仪Air9000P

01. 电源功能

电压输出	0~5V
电压可调	1mV
电流限制	0~2A
电流可调	0.1μA
最大功率	10W
供电方式	4.5V~5.5V

02. 测试功能

电压量程	0~5V
电压精度	1mV
电流量程	0~2A
电流精度	0.1μA
最大功率	10W
供电方式	USB-C/5V
硬件采样率	100KHz
上位机采样率	10KHz
上位机软件	Windows7及以上
API接口	开放动态库API接口

降功耗,找合宙!

WWW.OPENLUAT.COM

合宙
通信
可调稳压直流电源
可搭配Air9000使用，也可以独立使用
超轻超小，单手可握，独立风扇，长时间工作无压力
输出电压：0~30V
输出电流：0~5A
输出功率：150W
降功耗,找合宙!
WWW.OPENLUAT.COM

合宙
通信
合宙功耗分析仪Air9000/Air9000P
PC端软件支持
www.hezhouyibiao.com
电流实时动态显示/功耗自动计算
DC Analyzer Air9000
3.3320 V
3.6735 μA
12.240 μW
00:01:37
降功耗,找合宙!
WWW.OPENLUAT.COM

合宙
通信
云服务
合宙云端增值服务系列
DTU后台
远程管理
详见dtu.openluat.com
LBS定位服务
特色服务
根据需要自由选择
OTA远程升级
极简操作
详见iot.openluat.com
降功耗,找合宙!
WWW.OPENLUAT.COM

合宙
通信
联络合宙
官网：www.openluat.com
14:05
合宙 LuatOS
志愿者
报名入口
招募20名！合宙杯物联网大赛志愿者
燃情赛事等你来~
找资料
买产品
看文章
15:26
上海合宙 LuatOS...
+关注
11085
粉丝数
6年好店
店铺特色
1.4万
回头客
推荐
宝贝
分类
作品
¥ 19.90
合宙Air780EU 4G海外Cat1模
¥ 19.90
合宙Air780E 4G Cat.1模组全
¥ 28.00
合宙Air780EG · 1618超小封
¥ 16.80
合宙Air700E模块 · 黄豆大小
¥ 4.80
合宙Air32F103芯片 软硬件
首页
宝贝
作品
客服
合宙官方淘宝店
Luat.taobao.com
合宙官方公众号
HelloLuatOS
降功耗,找合宙!
WWW.OPENLUAT.COM

十四 合宙功耗分析仪

合宙功耗分析仪 Air9000、Air9000P 是一款功能齐全、稳定可靠。且支持 PC 端软件的小型手持式高精度功耗测试仪表。具有高速动态变化的微弱电流的抓取和上位机软件电流波形分析功能。适合于各类电池供电产品的动态功耗测试。解决各类企业低功耗测试难题。

Air9000：电流范围 0-5A，最小分辨率 1uA;电压范围 0-24V，最小分辨率 10mV。

Air9000P：电流范围 0-2A，最小分辨率 0.1uA;电压范围 0-5V，最小分辨率 1mV。

Air9000P 独有：内置可调输出高精度电源，出厂默认精确校准。

14.1 Air9000 产品手册

Air9000 使用手册可通过点击右方链接在线阅读：合宙功耗分析仪 Air9000 产品手册

也可直接阅读本章节以下内容：

合宙功耗分析仪使用手册 V1.03

Air9000

高精度动态功耗测试

多功能PC端软件支持

适用于各类电池供电产品的动态功耗测试

电流范围0~5A，最小分辨率1μA；电压范围0~24V，最小分辨率10mV

解决各类企业低功耗测试难题

降功耗，找合宙！ www.hezhouyibiao.com

合宙功耗分析仪——手册目录

一. 产品介绍 …… 01
二. 技术指标 …… 02
三. 接口与按键 …… 04
四. 供电方式 …… 05
五. 连接方式 …… 06
六. 操作说明
6.01 LCD显示之数据概览页 …… 07
6.02 LCD显示之详细数据页 …… 08
6.03 LCD显示之波形预览页 …… 08
6.04 LCD显示之PD切换页 …… 09
6.05 关于PD功能的相关说明 …… 10
6.06 LCD显示之设置页 …… 11
6.07 LCD显示之手动升级页 …… 12
七. PC端软件功能 …… 13
八. PC端软件界面
8.01 界面功能说明 …… 14
8.02 数据浮窗区域 …… 15
8.03 当前显示区域 …… 15
8.04 平均数据区域 …… 16
8.05 其他数据区域 …… 16
8.06 图表显示区域 …… 17
8.07 切换数据显示方式 …… 18
8.08 实时数据/历史数据 …… 19
8.09 选取某个特定区域 …… 20
8.10 分析数据 …… 21
8.11 控制设备输出 …… 22
8.12 导出数据 …… 23
8.13 导入数据 …… 23
8.14 强制升级软件 …… 24
8.15 数据缓存方式 …… 25

点击 01-25 任意页右下角【降功耗，找合宙】区域，即可返回本目录页。

降功耗，找合宙！ www.hezhouyibiao.com

一. 合宙功耗分析仪——产品介绍

合宙功耗分析仪Air9000——是一款功能齐全、稳定可靠，且支持PC端软件的小型手持式高精度功耗测试仪表。

- 输入端可连接直流数字电源，由直流电源控制Air9000输入电压，进而控制电压输出；
- 输入端也可连接PD快充充电头（支持PPS），进而Air9000诱骗设定的电压与电流输出；
- 输出端连接被测设备，通过测量输出端的电压和电流，进而测量被测设备的功耗；
- 电压测量范围0~24V，测量误差＜0.5%，最小分辨率10mV；
- 电流测量范围0~5A，测量误差＜0.5%，最小分辨率1μA；
- 电流多通道同步采样，无换挡延迟，采样速率高达80KHz，满足低功耗测试需求；
- 可连接PC端软件查看与分析电流波形，满足蜂窝模组、蓝牙芯片、Wi-Fi芯片等功耗测试需求；
- 可使用命令行工具在任意设备（Win/Linux/Mac）上抓取串口数据，使用PC端软件导入查看。

www.hezhouyibiao.com

- 让 你 的 硬 件 更 值 钱 -
降功耗，找合宙！ 01

二. 合宙功耗分析仪——技术指标

合宙 仪表

电源功能

电压输出	不支持
电流限制	不支持
最大功率	120W
供电方式	USB-C/电源线/5~24V
特别说明	可搭配快充充电头(支持PD/PPS协议)或直流稳压电源，作为电源使用

测量功能

电压量程	0~24V
电压精度	10mV
电流量程	0~5A
电流精度	1μA
最大功率	120W
供电方式	USB-C/电源线/5~24V
硬件采样率	80KHz
PC端采样率	10KHz
PC端软件	Windows7及以上
API接口	开放动态库，支持全平台

注：相关信息如有异动，请以最新使用手册为准。

www.hezhouyibiao.com

- 让 你 的 硬 件 更 值 钱 -
降功耗，找合宙！

02

二. 合宙功耗分析仪——技术指标

- **最大测量范围：**电压最大24V，精度最高可达10mV，电流最大5A，精度最高可达1μA；
- 使用0.4% ±20ppm精度基准芯片，预留0.1% 基准位置，0.1%精度精密电阻，12位ADC设计；
- **自动测量范围：**电流回显档位分别为500μA、50mA、5A三个档位，根据当前输出电流自动切换；
- **档位切换速度：**Air9000采用多通道同步采样，无换挡延迟；
- **压降切换速度：**Air9000采用2/3通道采样电阻，采样电阻被短路时最大延时为100uS；
- 固定多通道10KHz采样速率，USB高速数据传输；
- **电流测量精度：**500μA档位时分辨率1μA，50mA档位时分辨率100μA，见下表。

类型	范围	分辨率	标称精度	实际精度	类型
电压设定	0V-21V	20mV	2%	PD诱骗PPS决定	
电压回读	0V-24V	10mV	1%	< 0.5%	
电流设定	0-3A	50mA	2%	PD诱骗PPS决定	
500μA	1μA-500μA	1μA	1%	< 0.5%	自动档位
50mA	500μA-50mA	15μA	1%	< 0.5%	自动档位
5A	50mA-5A	1.5mA	1%	< 1%	自动档位

三. 合宙功耗分析仪——接口与按键

接口

接口	功能说明
PC口	设备自身供电/PC客户端通讯
电源输入	USB TYPE-C 供电输入
电源输出	USB TYPE-C 供电输出
小拨码	供电来源切换，电源输入/PC口

按键

按键	功能说明
翻页键	翻页键/强制进入升级模式
O	确定键
<	左移键
>	右移键
X	取消键

www.hezhouyibiao.com

- 让 你 的 硬 件 更 值 钱 -
降功耗，找合宙！ 04

四. 合宙功耗分析仪——供电方式

合宙功耗分析仪Air9000自身支持以下两种方式供电：

PC口供电方式：

当设备左上角【**小拨码拨向→右边**】时，则选择通过PC口向Air9000自身供电，电源输出口电压与电源输入口电压压降接近为零。

特别注意：

PC口严禁反接或电压大于5.5V，反接或大于5.5V会导致设备必烧！

部分品牌快充充电头(如华为、小米等私有协议快充头)可能会错误识别快充协议，导致向设备提供超过5.8V的高电压进而导致瞬间烧毁Air9000，推荐使用合宙官方标配支持PD协议的快充充电头。

电源输入口供电方式：

当设备左上角【**小拨码拨向←左边**】时，则选择通过电源输入口向Air9000自身供电，电源输出口电压相对电源输入口电压有一定的压降；

由于Air9000测量的电压值为输出端电压(电源输出口)，即被测设备实际供电电压，所以虽然输出端相比输入端有一定的压差，但被测设备的实际电源电压依旧是真实值，测试结果不会失真。

特别注意：

当拨码键拨向←左边时，如果电源输入口使用的设备为支持PD协议的充电头且为从USB-C口取电，则需要提前在Air9000设置页将开机自动启用PD选项打开，否则Air9000将无法取电开机；而且，USB-C口供电电压低于5V时，Air9000也无法正常开机。

如果电源输出口连接的被测设备是支持PD快充协议的手机/充电宝/电脑时，请不要使用并开启电源输入口快充充电器的PD功能，即不要在Air9000设置页将开机自动启用PD选项打开，电源输入口和电源输出口两台设备同时支持PD诱骗功能会导致Air9000不可预知的干扰。

五. 合宙功耗分析仪——连接方式

合宙功耗分析仪Air9000支持两种连接方式：四线模式与双线模式。

四线模式：

该模式可以在测量电流的同时，同步测量电源输出口的电压。

所谓四线——指的是电源输入口的正极、电源输入口的负极、电源输出口的正极、电源输出口的负极。四线模式下，既可以测试电源输出口的电流，也可以测试电源输出口的电压。

双线模式：

该模式可以直接将被测设备串联入供电电路正极回路，但缺点是Air9000将必须通过PC口供电且无法测量电源输出口电压（此时Air9000及PC端电压显示值为环境噪音）。

所谓双线——指的是电源输入口的正极、电源输出口的正极。双线模式下，只可以测试电源输出口的电流。

六. 合宙功耗分析仪——操作说明

Air9000支持中英文双语切换，手册仅以英文界面截图示例说明。

6.01 LCD显示之数据概览页

LCD显示-数据概览页由上到下显示的数值分别为：

- 电压瞬时值
- 电流瞬时值
- 当前耗电量或当前功率
- 设备统计时间

该页相关操作说明：

- 按下**O键**，可旋转屏幕；
- 按下**<键**，可切换Wh、Ah与W的显示；
- 按下**>键**，可切换显示风格；
- 按下**X键**，可清空累加数据(时间、电量等)，从头重新统计。

六. 合宙功耗分析仪——操作说明

Air9000支持中英文双语切换，手册仅以英文界面截图示例说明。

6.02 LCD显示之详细数据页

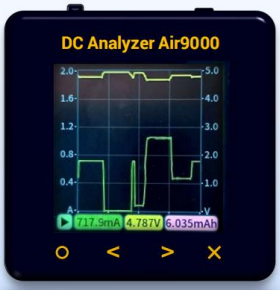
该页显示了平均电流、峰谷电流与更加详细的电量统计（安时与瓦时）。

相关操作说明：

- 按下X键，可清空所有数据，从头重新统计。

6.03 LCD显示之波形预览页

该页可预览当前的电流波形数据。



相关操作说明：

- 按下O键，可重新开始或暂停当前的波形刷新；
- 按下<键，可切换Wh、Ah与 W的显示；
- 按下>键，可更改电流波形纵坐标范围，固定或自适应；
- 按下X键，可清空当前数据，重新抓取。

六. 合宙功耗分析仪——操作说明

Air9000支持中英文双语切换，手册仅以英文界面截图示例说明。

6.04 LCD显示之PD切换页

该页可设置PD诱骗的目标电压与电流值。

相关操作说明：

- 按下O键，可开启PD功能，请按弹窗描述继续操作。

- 按下O键，可执行当前选中项目的操作；
- 按下<键，可按顺序切换需要选择的值按钮；
- 按下X键，可关闭PD诱骗功能。

六. 合宙功耗分析仪——操作说明

Air9000支持中英文双语切换，手册仅以英文界面截图示例说明。

6.05 关于PD功能的相关说明：

- 当前仅支持PD/PPS快充协议的诱骗，其他快充协议尚不支持；
- PD使用USB-C引脚进行通信，标准的连接线为USB-C口–USB-C口，一般USB-A口的数据线可能无法通信；
- 即使能使用品牌手机触发快充，也不能说明此充电头支持标准PD快充协议，有可能是品牌手机的私有协议；

- 必须要使用**六根线芯的USB-C数据线**，部分USB-C数据线只有VBUS、GND、 USB-C口–USB-C口四根线芯，这种USB-C数据线无法支持使能充电头PD快充功能，建议使用CC专用USB-C数据线；
- 当PD功能设置页开启**【启用PPS功能】**选项后，将使支持PPS功能的充电头自动进入PPS模式，因部分品牌充电头的PD与PPS支持电压范围不同，如需使用充电头纯PD模式固定电压档位模式，请关闭PPS功能。

风险提示：

Air9000开启PD/PPS功能且设置电压输出小于5V时（比如MCU常见供电电压3.3V），充电头电压输出可能会有跳到5V的突变。

因此，建议在充电头电压输出稳定前先断开电源输出口被测设备。

六. 合宙功耗分析仪——操作说明

Air9000支持中英文双语切换，手册仅以英文界面截图示例说明。

6.06 LCD显示之设置页

该页可设置设备中的各项参数。

相关操作说明：

- 按下O键，可开启功能，或将值向左调整；
- 按下<键，可向上下切换项目；
- 按下X键，可关闭功能，或将值向右调整。

Air9000

www.hezhouyibiao.com

- 让 你 的 硬 件 更 值 钱 -
降功耗，找合宙！

六. 合宙功耗分析仪——操作说明

Air9000支持中英文双语切换，手册仅以英文界面截图示例说明。

6.07 LCD显示之手动升级页：

如果Air9000在升级过程中出现错误导致无法正常开机，可以进入DFU升级模式手动升级。

相关操作说明：

- 断开Air9000全部电源连接；
- 按住**翻页键强制进入升级模式**不松开；
- **PC口**插入USB-C数据线供电；
- 此时可以看到Air9000屏幕上显示DFU模式的信息。

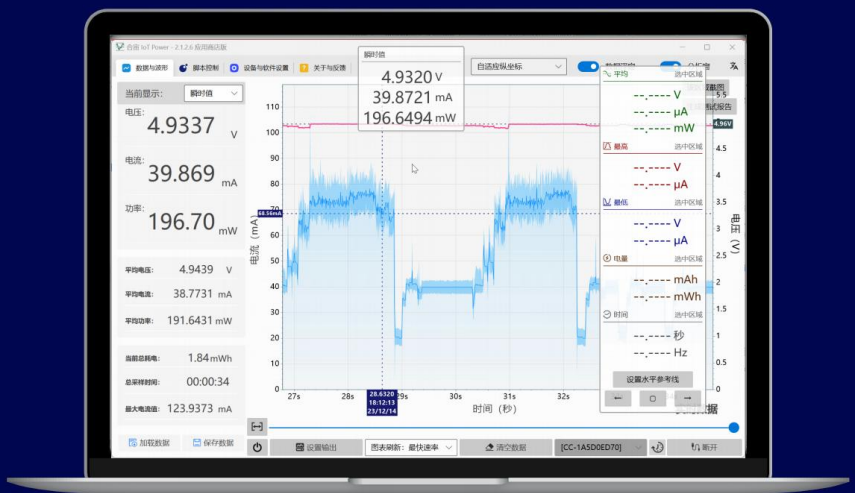
强制升级：

除DFU模式手动升级外，还可以通过如下方式强制升级：

在电脑**设备管理器**的**通用串行总线设备**中找到名为 IoTPower DFU的设备；

打开PC端–设备与软件设置–强制升级固件，进行升级操作即可。

七. 合宙功耗分析仪——PC端软件功能



PC端软件主要功能：

- 实时显示电压电流动态波形
- **分析指定时间段的功耗数据**
- 记录指定时间段的历史数据
- 导出历史数据且支持导出为独立.exe文件
- 导入历史数据并回放波形动态变化全过程
- 运行Lua脚本控制电源输出，可模拟电池放电曲线等
- 为合宙功耗分析仪各型号升级固件
- **微软商店下载：**点此打开微软商店页面
- DLL/SO动态运行库：Gitee iot-power dll/so

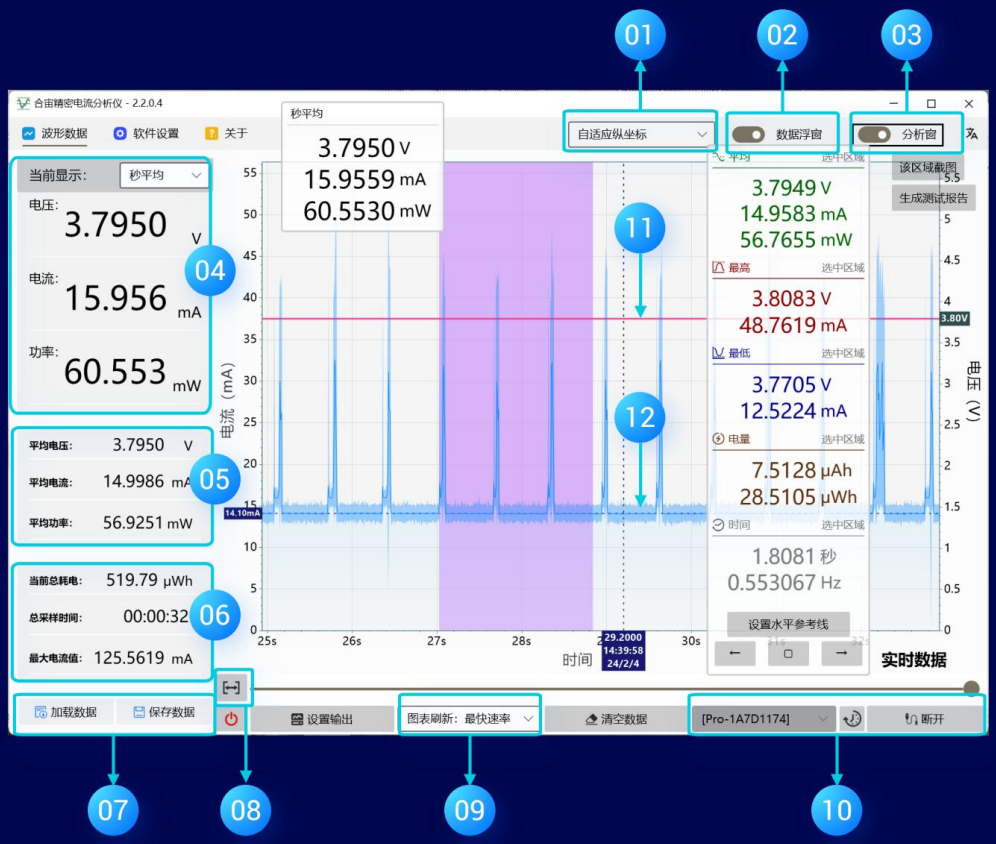
- 平均电流
- 平均电压
- 平均功率
- 峰值、最小电流
- 峰值、最小电压

www.hezhouyibiao.com

- 让 你 的 硬 件 更 值 钱 -
降功耗，找合宙！

13

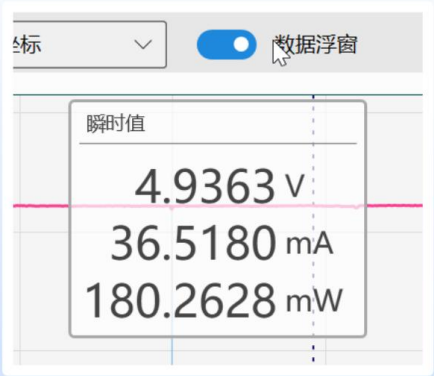
八. 合宙功耗分析仪——PC端软件界面



8.01 界面功能说明（对应图示序号）

01. 图表展示方式的偏好
02. 当前数据的置顶悬浮窗
    方便在软件最小化时查看数据
03. 选定区域数据分析
04. 当前数据可选择显示：
    瞬时值/秒平均值/分钟平均值
05. 本次抓取数据的平均值
06. 其他的各项信息
07. 导出或导入数据
08. 切换按钮：当前/圈选
09. 确定波形显示区域的下拉选项
10. 控制设备的连接与断开
11. 粉色曲线为电压
12. 蓝色曲线为电流

八. 合宙功耗分析仪——PC端软件界面



8.02 数据浮窗区域

界面右上方的数据浮窗开关可以打开或关闭浮窗；

数据浮窗打开后此区域的数据与当前显示区域的数据完全相同且始终置顶显示，方便软件最小化后查看实时数据。

特别说明：

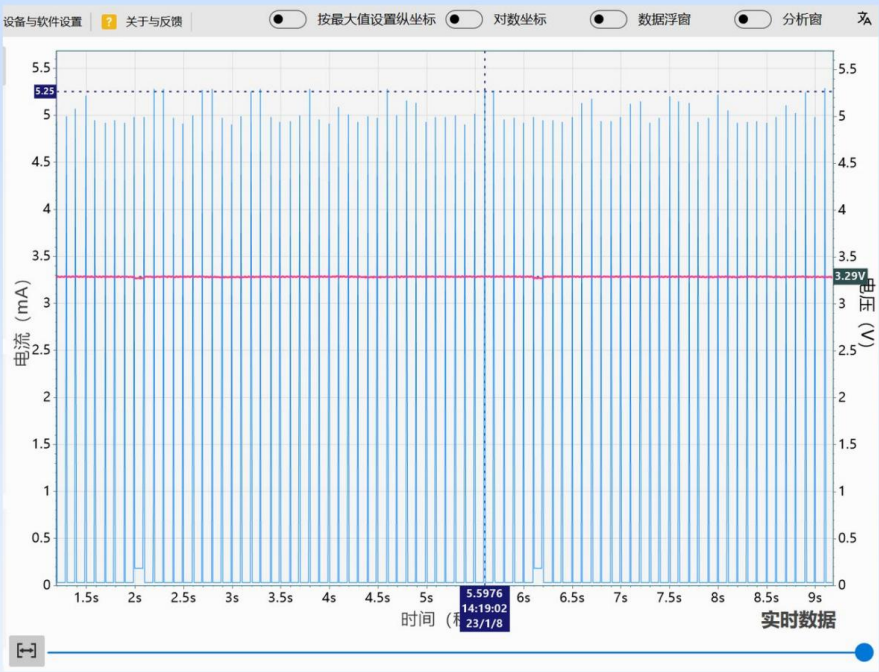
当前显示数据若选择瞬时值档位，可能因刷新时间点不完全一致而导致PC端与仪表端数据不一致，为正常现象。

8.03 当前显示区域

当前显示区域数值为设备连接后当前的数据，可以切换为滑动平均值或瞬时值。

八. 合宙功耗分析仪——PC端软件界面

八. 合宙功耗分析仪——PC端软件界面



8.06 图表显示区域

该区域为图表显示区域，可以预览被测设备实时电压电流动态波形。

调整显示方式

- **自适应坐标：**按照当前显示区域的最大值，动态地决定纵坐标范围；
- **按最大值设置纵坐标：**按此次抓取数据的最大值，固定纵坐标范围；
- **对数坐标：**当电流数据绝对值相差极大时可启用该功能，将以10的倍数来显示纵坐标数值。

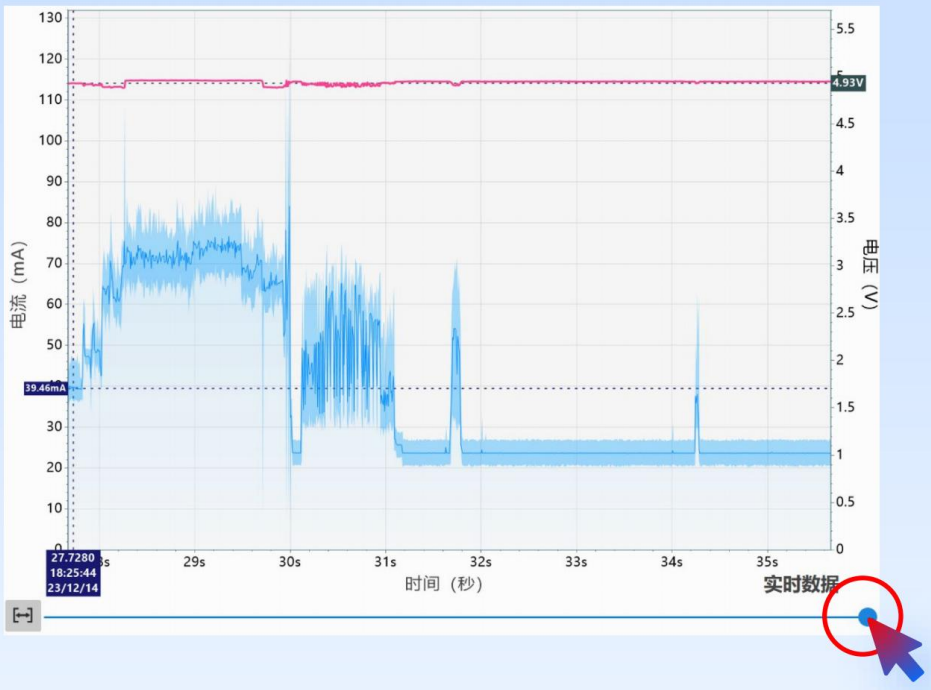
八. 合宙功耗分析仪——PC端软件界面

8.07 切换数据显示按钮

按下该按钮，图表显示区域可在**实时数据/历史数据**和**选取某个区域**两种显示模式中切换。

Air9000

八. 合宙功耗分析仪——PC端软件界面



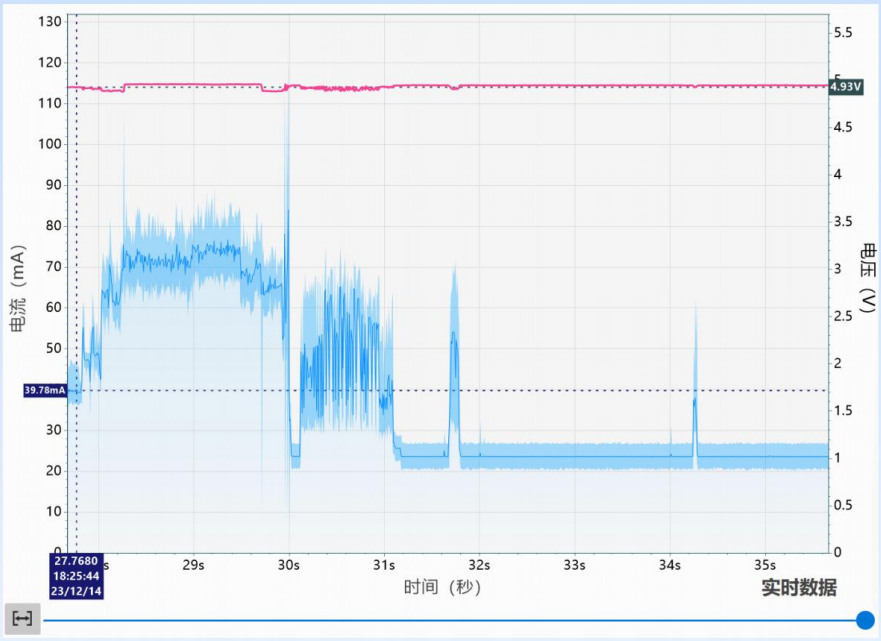
8.08 实时数据/历史数据（不能缩放显示和分析）

当鼠标在图表显示区域或在下方拖动轴上时，可以使用鼠标滚轮，调整显示不同时间段的波形；

也可以直接拖动下方拖动轴上的操纵杆，调整显示不同时间段的波形。

Air9000

八. 合宙功耗分析仪——PC端软件界面



8.09 选取某个特定区域（可以缩放显示和分析）

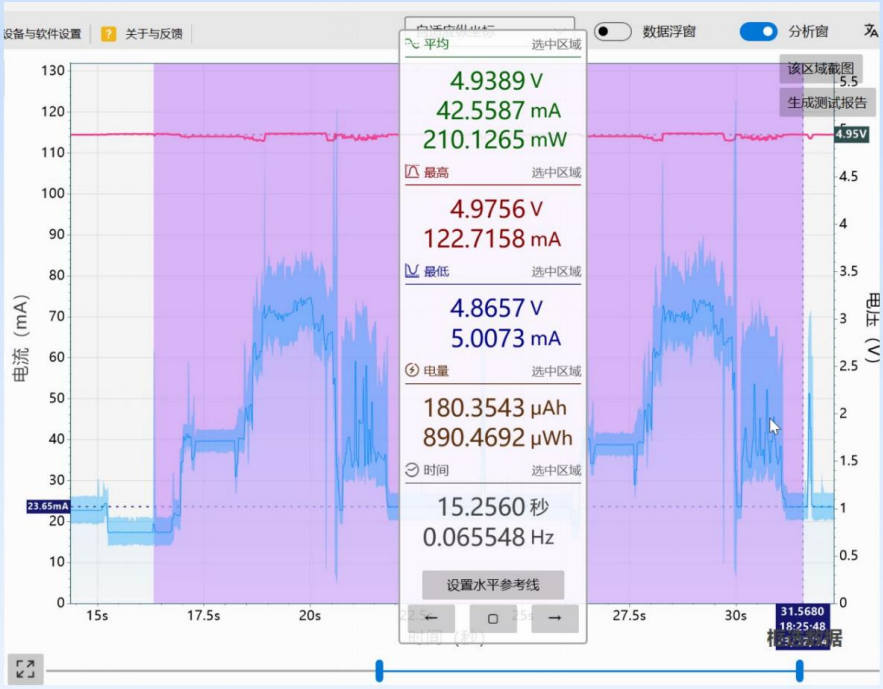
使用**鼠标左键**拖动，可以选取查看一个区域的数据，并进入**选取区域数据模式**。

当处于选取区域数据模式时，可以使用鼠标滚轮调整波形的时间位置，也可以按住CTRL键并滚动鼠标滚轮放大或缩小当前显示的区域。

下方拉杆同样也可以用于控制选择显示的范围：

- **鼠标拖动中间：**可以调整波形的时间位置；
- **鼠标拖动左边或右边的竖线：**可以调整波形显示的开始或结束时间点；
- **鼠标双击中间：**可以直接切换为查看全局数据。

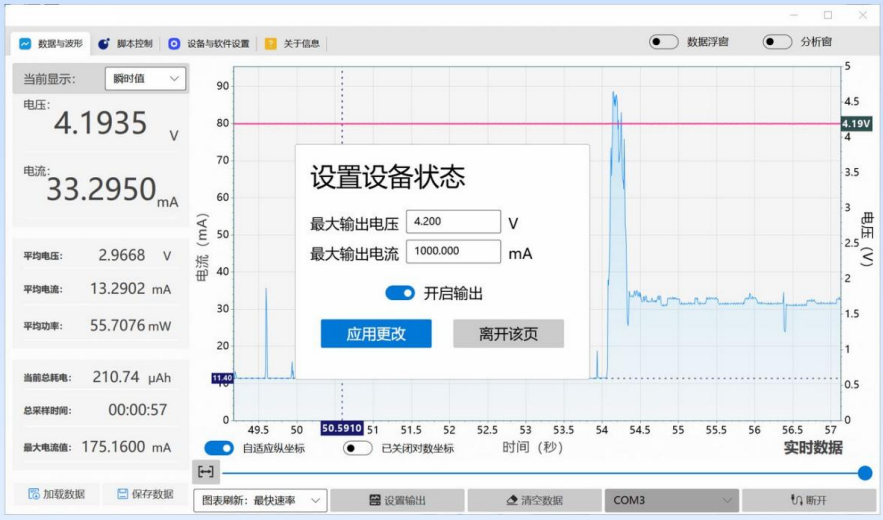
八. 合宙功耗分析仪——PC端软件界面



8.10 分析数据

任意模式下，**右击鼠标拖动并滑动图标显示区域，**即可选定一个时间段并分析其数据，分析窗将弹出并显示分析的结果。

八. 合宙功耗分析仪——PC端软件界面



8.11 控制设备输出

- 通过电源符号按钮，控制设备的输出、关闭状态;
- 通过设备输出按钮，设置设备的输出电压与电流。

特别说明：

Air9000需配合**支持PD/PPS快充协议的充电头**使用，推荐使用合宙随机附带快充充电头。

Air9000

八. 合宙功耗分析仪——PC端软件界面

8.12 导出数据

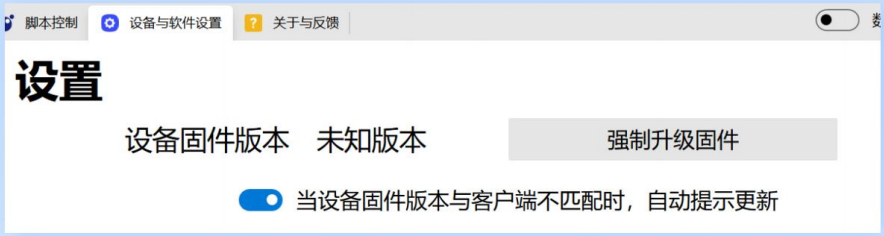
数据可自由地导出为.iotpl文件、.csv表格文件、.exe自执行文件；

- .csv表格文件用于用户自行分析原始数据，无法再重新导入，.iotpl文件可以再次导入PC端软件；
- .exe为自执行文件，运行后可以再次导出为.iotpl文件用于导入PC端软件。

8.13 导入数据

- 手动选择导入Air9000专用的.iotpl文件；
- 导入从自动保存的文件夹中的.iotpl文件，该功能通常出现在功能关闭或重新抓取时。

八. 合宙功耗分析仪——PC端软件界面



8.14 强制升级软件

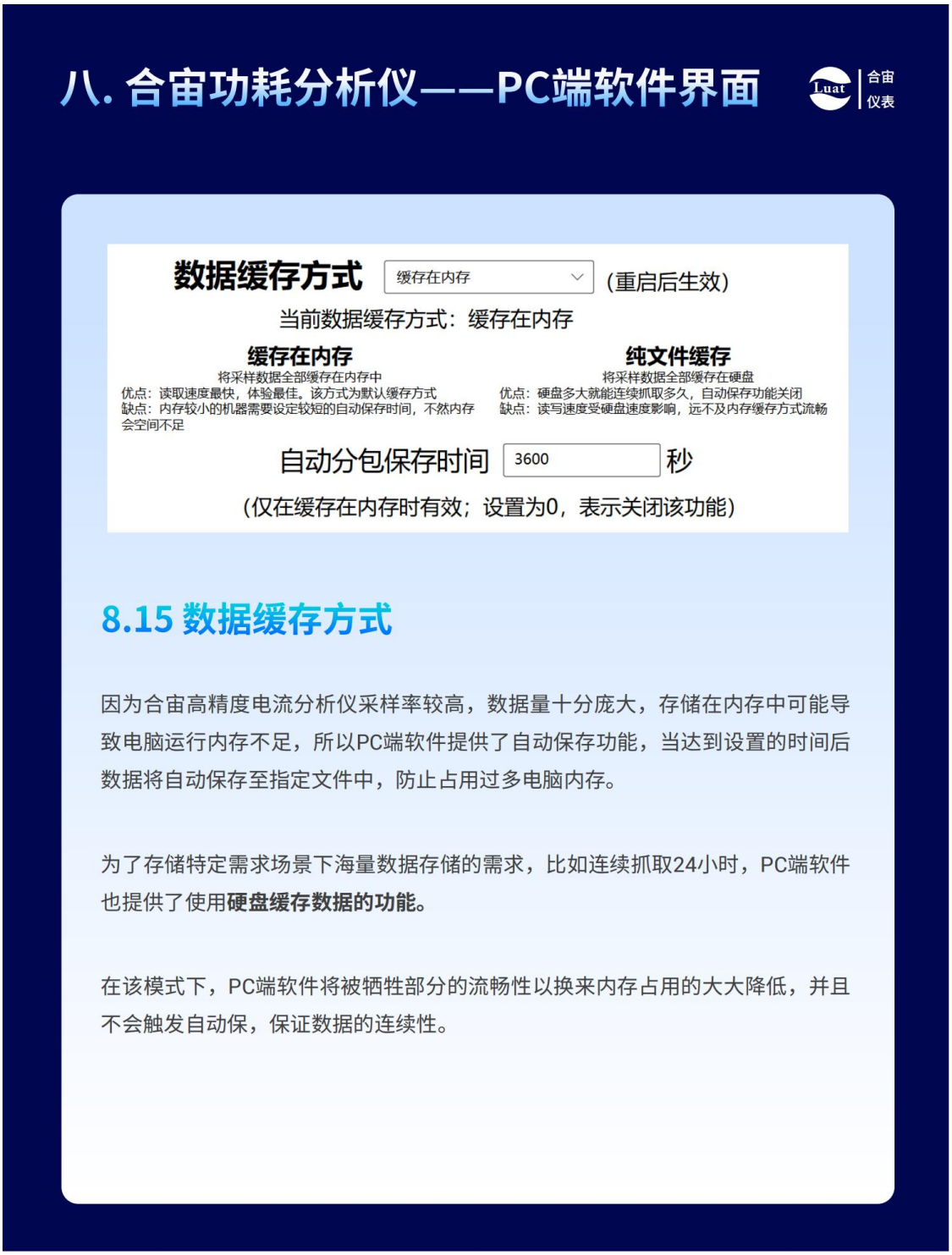
通常来说，设备固件会自动提示升级，但当设备升级途中中断或因其他原因导致固件损坏时，可手动刷入固件；

- 单击强制升级固件按钮，可强制刷入客户端自带的固件。

  注意：若设备有升级模式，请先进入升级模式后再刷固件。

- 右击强制升级固件按钮，可强制刷入用户手动选择的固件文件，用于测试功能、验证BUG修复等。

Air9000



14.2 Air9000 上手指南

Air9000 上手指南可通过点击右方链接在线阅读：合宙功耗分析仪 Air9000 上手指南

也可直接阅读本章节以下内容：

Air9000

合宙功耗分析仪 Air9000 上手指南

上海合宙通信科技有限公司

1. 收到快递

顺丰快递，时效性还是很不错的；

2. 拆快递箱

我购买了两台，一台 Air9000，一台 Air9000P；

3. 包装展示

看起来还挺高级的，跟仪表类产品很搭；

4. 打开包装

左边 Air9000，右边 Air9000P；

5. 先聊 Air9000

给 Air9000 来个特写；

6. 配件展示

1 台 Air9000+1 个 65W 充电头+1 个 A2C 数据线+1 个 C2C 数据线+2 个 USB-C 转鳄鱼夹电源线+说明书卡片；

7. 特写展示

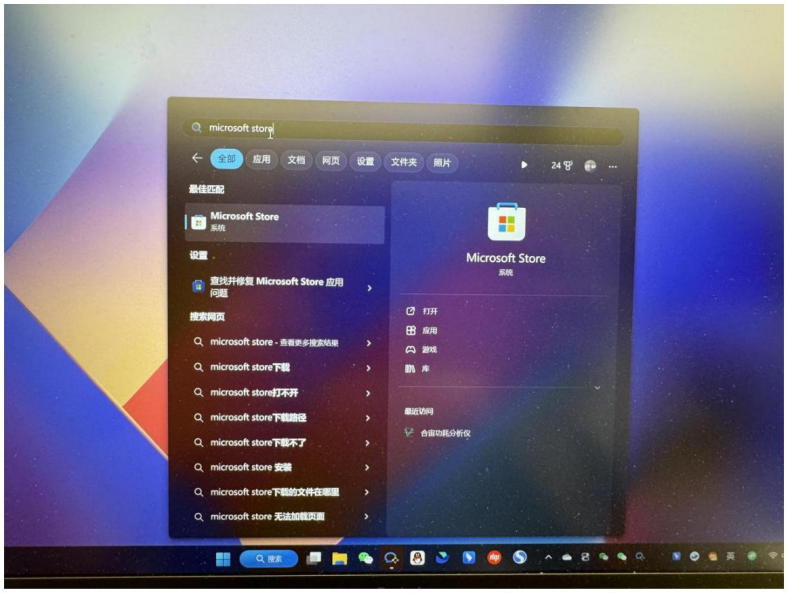
降功耗，找合宙！镭雕丝印效果看起来还不错；

8. 说明书卡片

卡片正面是 Air9000 经典图示，背面是说明书链接二维码，大家也可以直接打开 www.hezhouyibiao.com 查看产品说明；

9. 下载上位机

打开 Microsoft Store；



10. 合宙功耗分析仪

Microsoft Store 搜索框内输入“合宙功耗分析仪”，安装并打开；

11. 连接电脑

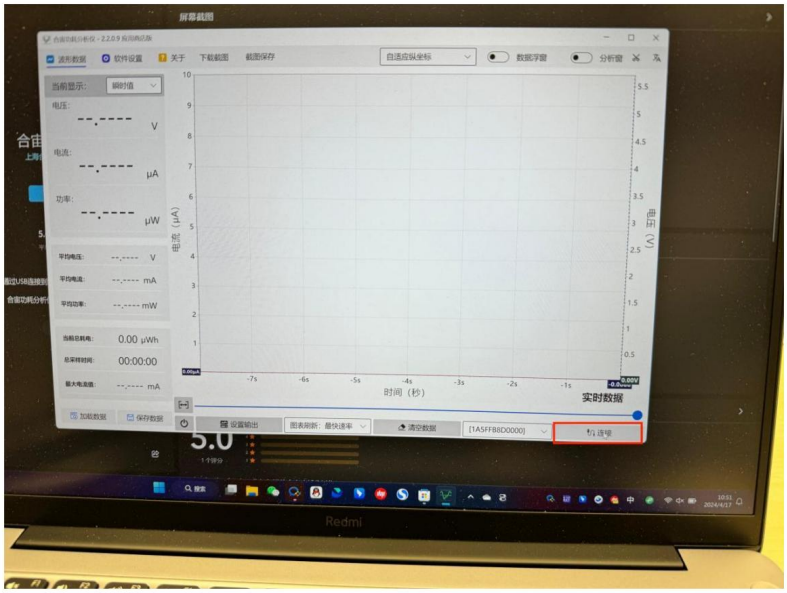
用 A2C 数据线连接电脑；

12. 特别提醒

小拨码开关建议拨到右侧，默认由电脑 USB 供电；

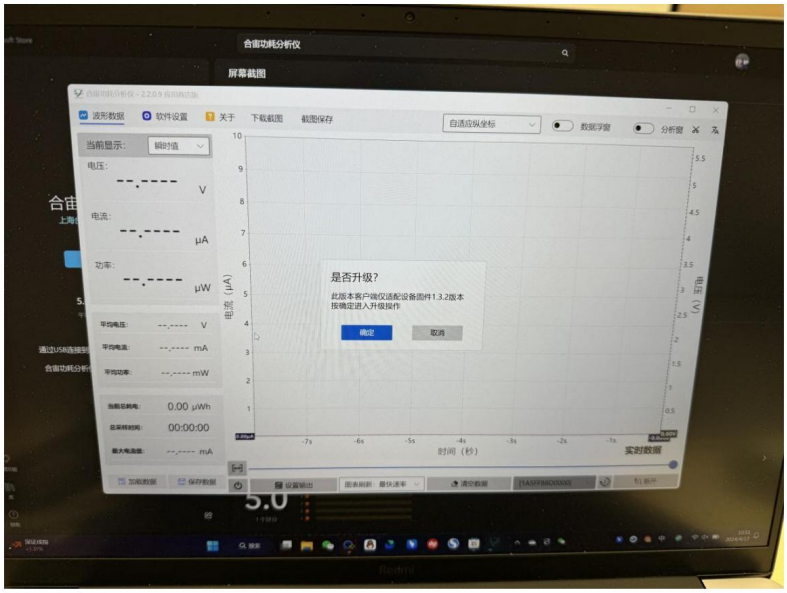
13. 点击连接

点击上位机右下角“连接”按钮，使 Air9000 与上位机连接通信；



14. 升级软件

首次连接上位机会提示升级 Air9000 最新固件，点击“确定”即可；

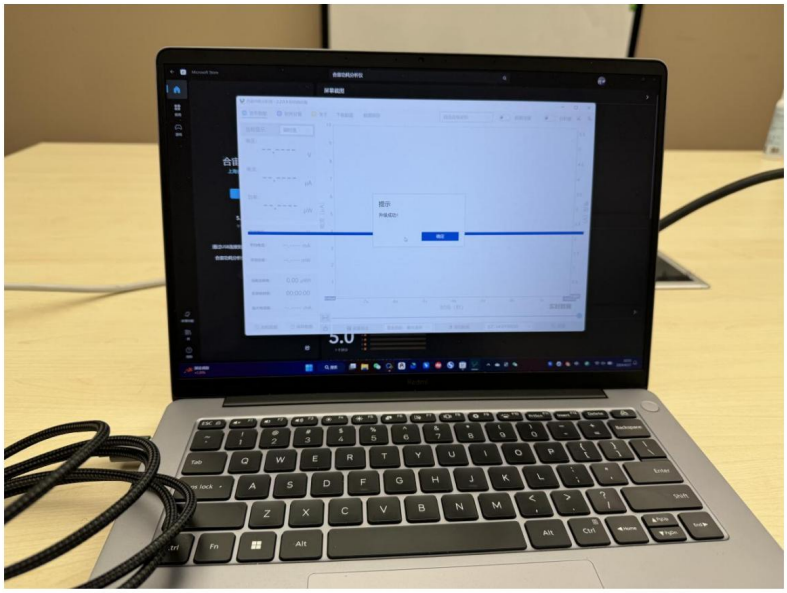


15. 升级固件中

保持连接，不要断开；

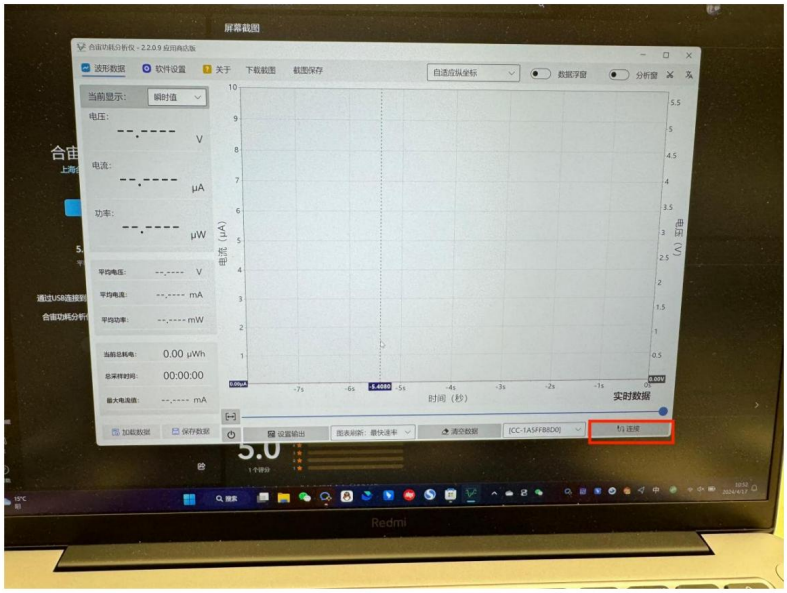
16. 升级成功

固件升级成功，点击“确定”即可；



17. 再次连接

软件升级成功后需再点击一次“连接”；

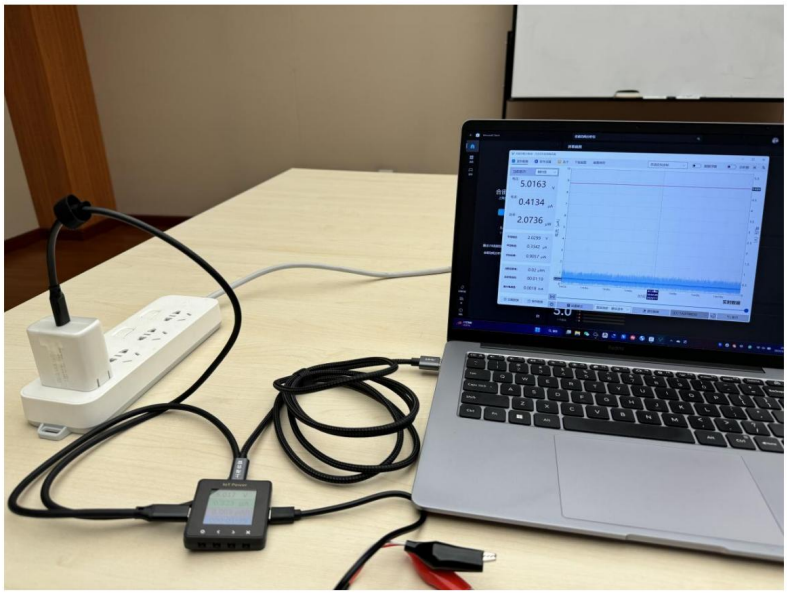


18. 连接成功

连接成功后，无论 Air9000 还是上位机显示的都是电路纹波随机值，可以忽略；

19. 硬件全家桶

1 台 Air9000+1 个 65W 充电头+1 个 A2C 数据线+1 个 C2C 数据线+1 个 USB-C 转鳄鱼夹电源线+1 台电脑；

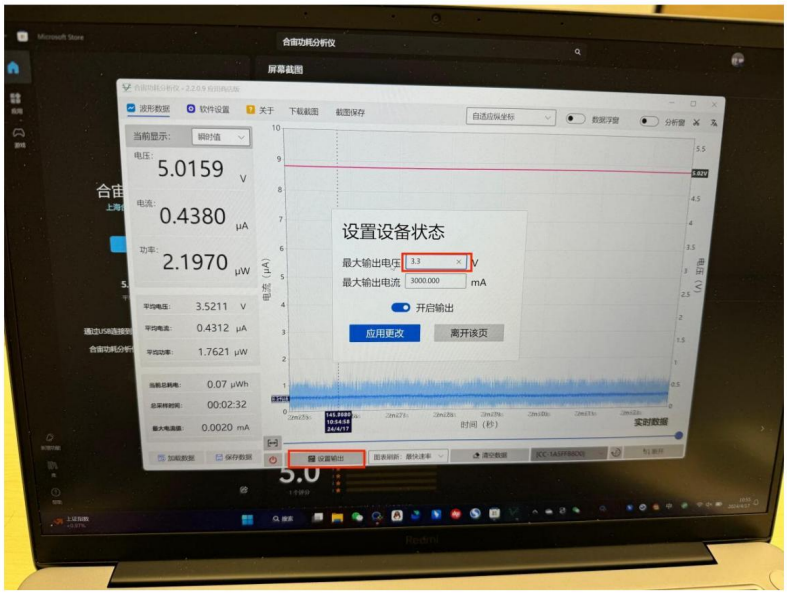


20. PD 功能

Air9000 翻到 PD 功能页，可以看到 PD/PPS 功能均已默认打开，这是调整充电头电压输出值的前提；

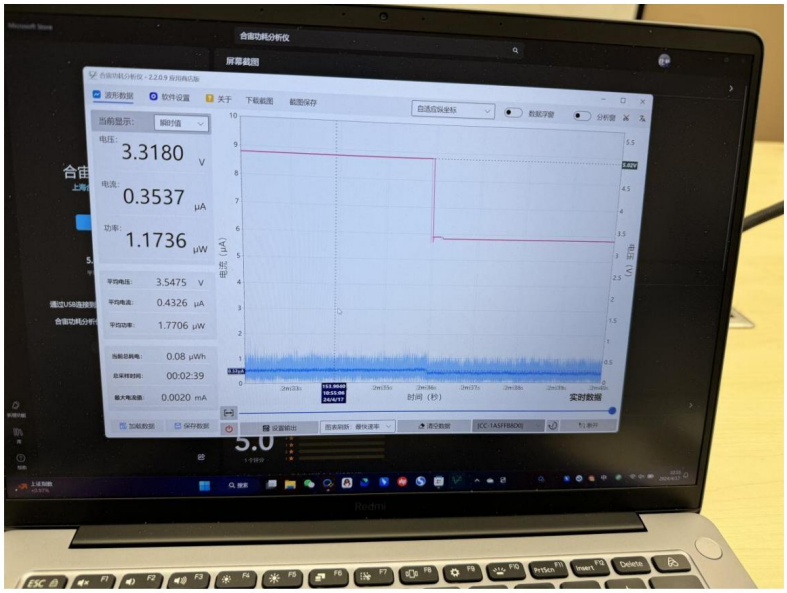
21. 输出值调整

充电头默认电压输出值 5V，以调整充电头输出值 3.3V 为例，如下：



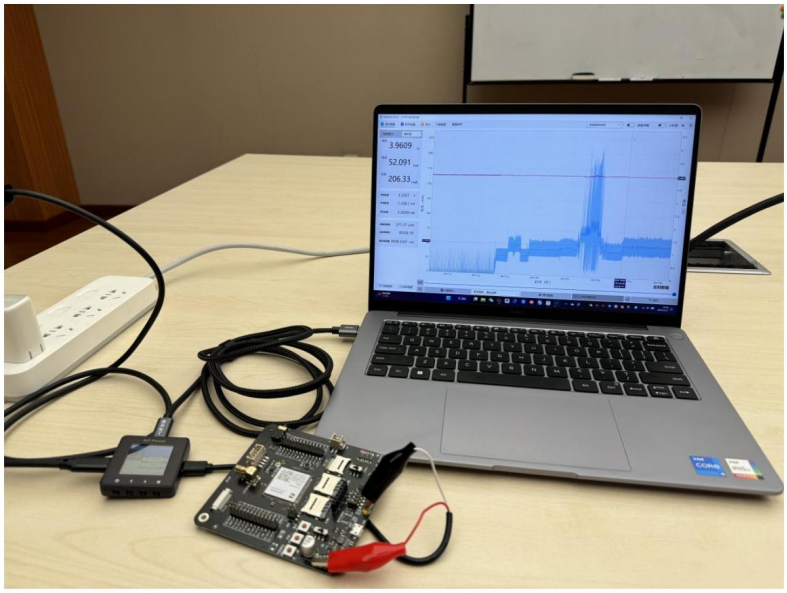
22. 调整成功

3.3V 输出设置成功，见下图。



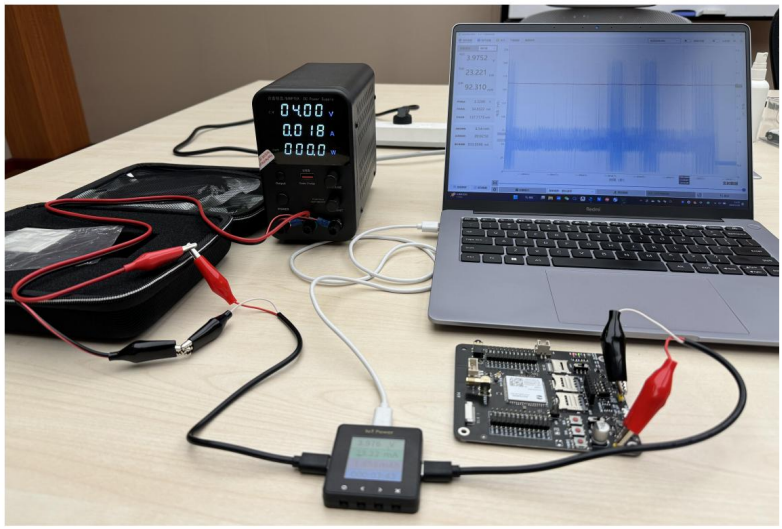
23. 接上被测设备

以合宙 4G Cat.1 模组 Air724UG 开发板为例；



24. 直流电源

Air9000 除支持 PD/PPS 充电头作为外接电源使用外，也支持直接外接直流稳压电源；



特别说明：

Air9000 配套用直流稳压电压合宙官方淘宝店有售，也可直接联系合宙销售经理购买。

25. 更多疑问

请点击 www.hezhouyibiao.com 查阅详细资料。

14.3 Air9000P 产品手册

Air9000P 使用手册可通过点击右方链接在线阅读：合宙功耗分析仪 Air9000P 产品手册

也可直接阅读本章节以下内容：

降功耗，找合宙！ www.hezhouyibiao.com

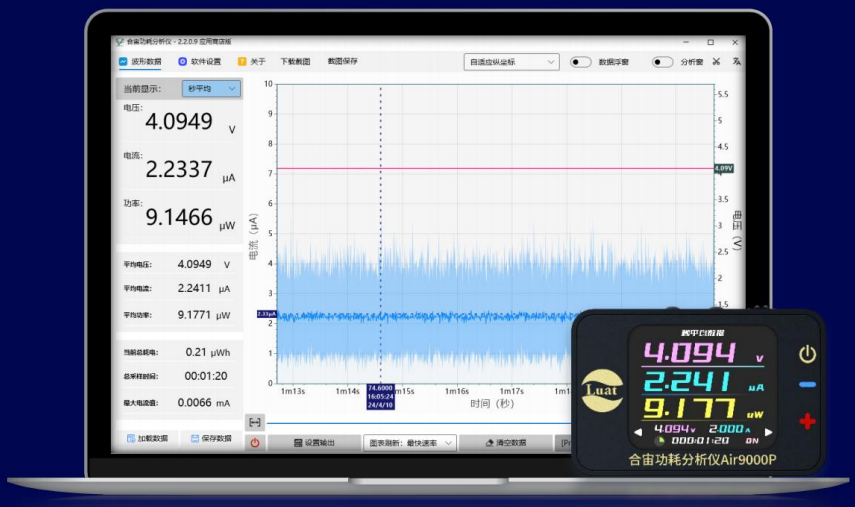
合宙功耗分析仪——手册目录

一. 产品介绍 …… 01
二. 技术指标 …… 02
三. 接口与按键 …… 04
四. 供电方式 …… 05
五. 连接方式 …… 06
六. 操作说明
6.01 切换语言 …… 07
6.02 翻页切换 …… 08
6.03 控件选择与修改 …… 09
6.04 输出供电 …… 10
6.05 修改电压 …… 11
6.06 修改电流 …… 12
6.07 开启和关闭输出 …… 13
6.08 查看秒平均数据 …… 14
6.09 切换功耗单位 …… 15
6.10 查看详细数据 …… 16
6.11 查看波形/暂停刷新 …… 17
6.12 切换电流坐标范围 …… 18
6.13 模拟电池 …… 19
七. PC端软件功能 …… 20
八. PC端软件界面
8.01 界面功能说明 …… 21
8.02 数据浮窗区域 …… 22
8.03 当前显示区域 …… 22
8.04 平均数据区域 …… 23
8.05 其他数据区域 …… 23
8.06 图表显示区域 …… 24
8.07 切换数据显示按钮 …… 25
8.08 实时数据/历史数据 …… 26
8.09 选取某个特定区域 …… 27
8.10 分析数据 …… 28
8.11 控制设备输出 …… 29
8.12 导出数据 …… 30
8.13 导入数据 …… 30
8.14 强制升级软件 …… 31
8.15 数据缓存方式 …… 32

点击 01-32 任意页右下角【降功耗，找合宙】区域，即可返回本目录页。

降功耗，找合宙！ www.hezhouyibiao.com

一. 合宙功耗分析仪——产品介绍



合宙功耗分析仪Air9000P——是一款功能齐全、稳定可靠、支持PC端软件的小型手持式高精度功耗测试仪表。

- 通过USB-C接口供电，测量低功耗设备的电压与电流；
- 可做电池类产品电源使用，支持指定电压输出(0~5V)；
- 支持电流限流输出，可设置0~2A范围内的任意限流值；
- 电流测量范围0~2A，测量精度±0.1%±3字，最小分辨率0.1μA；
- 电压测量范围0~5V，测量精度±0.1%±3字，最小分辨率1mV；
- OCP过流保护，电路短路时将电流输出自动限制至1.5A以下；
- 电流多通道同步采样，无换挡延迟，采样速率高达10KHz，满足低功耗测试需求；
- 可连接PC端软件查看与分析电流波形，满足蜂窝模组、蓝牙芯片、Wi-Fi芯片等功耗测试需求；
- 可使用命令行工具在任意设备(Win/Linux/Mac)上抓取串口数据，使用PC端软件导入查看。

www.hezhouyibiao.com

- 让 你 的 硬 件 更 值 钱 -
降功耗，找合宙！ 01

二. 合宙功耗分析仪——技术指标

01. 电源功能

电压输出	0~5V
电压可调	1mV
电流限制	0~2A
电流可调	0.1μA
最大功率	10W
供电方式	4.5V~5.5V

02. 测试功能

电压量程	0~5V
电压精度	1mV
电流量程	0~2A
电流精度	0.1μA
最大功率	10W
供电方式	USB-C/5V
硬件采样率	100KHz
上位机采样率	10KHz
上位机软件	Windows7及以上
API接口	开放动态库API接口

www.hezhouyibiao.com

- 让 你 的 硬 件 更 值 钱 -
降功耗，找合宙！ 02

二. 合宙功耗分析仪——技术指标

核心技术指标说明：

- **最大测量范围：**电压最大5V，最小分辨率1mV；电流最大2A，最小分辨率0.1μA；
- **最大测量精度：**采用高精度基准芯片，12位ADC设计，默认出厂校准，精度可达±0.1%±3字%；
- **电流测量范围：**电流回显200μA、20mA、2A三个档位，根据当前测试电流大小自动切换；
- **档位切换速度：**Air9000P采用多通道同步采样，无换挡延迟；
- **压降切换速度：**Air9000P采用2/3通道采样电阻，采样电阻被短路时最大延时为100μs；
- **电流测量精度：**200μA档位时分辨率0.1μA，20mA档位时分辨率10μA，更多数据参见下表。

类型	范围	分辨率	实际精度	说明
电压设定	0~5V	1mV	±0.1%±3字	
电压回读	0~5V	10mV	±0.1%±3字	
电流设定	0~2A	1μA	±0.1%±3字	
200μA	0.1uA-200μA	0.1μA	±0.1%±3字	自动档位
20mA	200μA-20mA	10μA	±0.1%±3字	自动档位
2A	20mA-2A	1mA	±0.1%±3字	自动档位

三. 合宙功耗分析仪——接口与按键

操作说明	普通模式	编辑模式
左翻页键	切换到左边的页面	切换到左边的一位数字上
右翻页键	切换到右边的页面	切换到右边的一位数字上
滚动滚轮	切换页面中选中的控件	调整选中数字的数值
按下滚轮	进入或切换选中的控件	退出编辑模式
输出按钮	开启或关闭电源输出	开启或关闭电源输出

四. 合宙功耗分析仪——供电方式

供电方式：

Air9000P通过PC口进行供电，可连接电脑USB-C口，由电脑USB-C口供电；
也可将连接PC口的USB-C线正负极接电源供电。

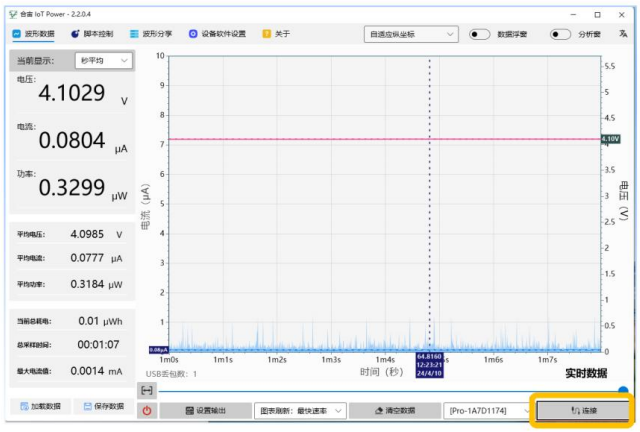
特别注意：

- PC口，即USB-C口，严禁正负反接或电压大于6V，反接或大于6V将会烧毁Air9000；
- 部分品牌快充充电头(如华为、小米等私有协议快充头)可能会错误识别快充协议，导致向设备提供超过5.8V的高电压进而瞬间烧毁Air9000P，推荐使用默认5V输出的充电头或接电脑USB-C口；
- 电源输出接口，严禁反接其他电源/电池，或连接5V以上电压的其他电源/电池，以免烧毁设备。

五. 合宙功耗分析仪——连接方式

首次使用开机提示【用前须知】，须“连接一次PC软件后即可使用”。

配套PC软件可从Windows Store搜索下载：合宙功耗分析仪



安装打开PC软件，点击右下角【**连接**】按钮，即可开始使用。

六. 合宙功耗分析仪——操作说明

6.01 切换语言

Air9000P支持中英文双语切换，设备默认中文显示，可自行更改。

中文切换为英文

英文切换为中文

相关操作说明：

- 开机后按左翻页键进入【设置】界面；
- 向右转动滚轮至【其他设置】选项，按下滚轮，则进入语言设置界面；
- 向右转动滚轮，选中**Chinese UI开关**（默认开启），按下滚轮关闭中文选项。
- 按左右翻页键，即可发现已切换至英文显示。
- 如需再切换至中文显示，则重复上述步骤，开启**中文界面开关**即可。

六. 合宙功耗分析仪——操作说明

6.02 翻页切换

通过顶部左/右翻页键，可以切换到不同功能界面。

相关操作说明：

此处以开机动画后进入的默认界面【实时数据】为例说明。

- 按左翻页键可依次进入：
  设置页—波形分析页—数据列表页—实时数据页，如此循环。
- 按右翻页键可依次进入：
  数据列表页—波形分析页—设置页—实时数据页，如此循环。

六. 合宙功耗分析仪——操作说明

合宙 仪表

6.03 控件选择与修改

通过滚轮的左/右转动、按下操作，可以选择选择或修改控件值。

相关操作说明：

此处【设置】页面为例说明。

- 向右转动滚轮键可向下切换所需控件，反之向左转动滚轮键可向上切换；
- 在所需控件，按下滚轮即可选中控件，部分控件可通过滚轮键更改控件值。

  例如：屏幕亮度的调整

  选中【基本功能】按下滚轮，右转滚轮选中【屏幕亮度】，控件显示蓝色边框；

  按下滚轮，控件显示红色边框；此时左右转动滚轮即可调整屏幕亮度。

- 再次按下滚轮，则退出修改。

按下滚轮选定屏幕亮度

转动滚轮调整亮度

六. 合宙功耗分析仪——操作说明

6.04 输出供电

相关数据说明：

- 秒平均数据界面从上至下依次为电压值、电流值、电量/功率值；
- 电压值下方的进度条，表示当前电压值占设置电压值的占比；
- 电流值下方的进度条，表示当前电流值占设置电流值的占比；
- 电量/功率值下方的进度条，表示当前耗电量占设置模拟电池电量的占比。

六. 合宙功耗分析仪——操作说明

6.05 修改电压

在实时数据页，通过左/右翻页键以及滚轮的配合操作，可以修改电压数值。

相关操作说明：

- 右转滚轮选中屏幕左下方的电压值，再按下滚轮即可进入编辑模式；
- 通过左右翻页键切换需改的数位，再通过左右转动滚轮键更改数值；
- 再次按下滚轮键即可退出更改。

翻页键选定电压数值　　转动滚轮调整电压数值

六. 合宙功耗分析仪——操作说明

6.06 修改电流

在实时数据页，通过左/右翻页键以及滚轮的配合操作，可以修改电流数值。

相关操作说明：

- 右转滚轮选中屏幕右下方的电流值，再按下滚轮即可进入编辑模式；
- 通过左右翻页键切换需改的数位，再通过左右转动滚轮键更改数值；
- 再次按下滚轮键即可退出更改。
- 选中电流值边上的A或mA，按下滚轮即可切换电流单位。

翻页键选定电流数值

转动滚轮调整电流数值

六. 合宙功耗分析仪——操作说明

6.07 开启和关闭输出

在实时数据页，按下右侧的【输出按钮】即可开启或关闭电源输出。

开启输出

按下输出按钮

右下角显示ON

关闭输出

按下输出按钮

右下角显示OFF

六. 合宙功耗分析仪——操作说明

6.08 查看秒平均数据

在实时数据页，使用滚轮选中上方【实时数据】，按下滚轮即可切换至秒平均显示。

01

转动滚轮选定实时数据

02

按下滚轮切换为秒平均数据

六. 合宙功耗分析仪——操作说明

6.09 切换功耗单位

在实时数据或秒平均数据界面，右转滚轮至功耗数据单位位置，再按下滚轮即可切换功耗单位： μW、 μWh、 μAh三种可选。

六. 合宙功耗分析仪——操作说明

6.10 查看详细数据

通过顶部的左/右翻页键，可切换至详细数据列表页。

当前可查看的相关测试数据：

- 秒平均电流
- 分钟平均电流
- 总平均电流
- RTC统计时间
- 最大峰值电流
- 最小非零电流
- 总容量
- 总能量

六. 合宙功耗分析仪——操作说明

6.11 查看波形/暂停刷新

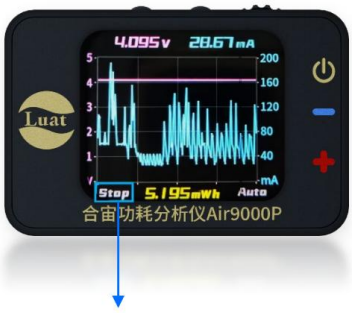
相关操作说明：

- 通过左/右翻页键，即可切换进入电压电流波形页面。
- 右转滚轮选中左下方的Run，按下滚轮文字变为Stop，即可暂停波形打印。
- 再次按下滚轮则恢复查看，如是循环。

翻页键进入波形页面

状态显示为显示【Run】

按下滚轮暂停刷新



状态显示为【Stop】

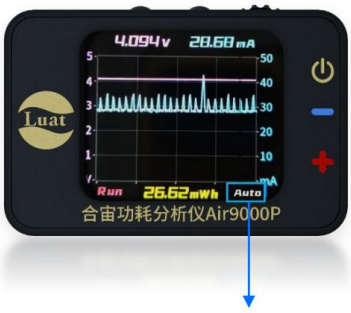
六. 合宙功耗分析仪——操作说明

6.12 切换电流坐标范围

相关操作说明：

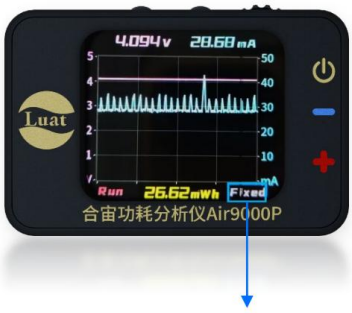
- 通过左/右翻页键，进入电压电流波形页面。
- 右转滚轮选中右下方的Auto，按下滚轮文字变为Fixed，即可切换电流坐标范围。
- 多次按下滚轮切换不同的固定坐标方式，如是循环可恢复自动坐标范围功能。

翻页键进入波形页面



状态显示为显示【Auto】

按下滚轮切换坐标范围



状态显示为【Fixed】

六. 合宙功耗分析仪——操作说明

6.13 模拟电池

Air9000P模拟电池功能，可模拟电压下降波形与容量，输出时将根据耗电量自动更改输出电压，并在电量耗尽时关闭输出。

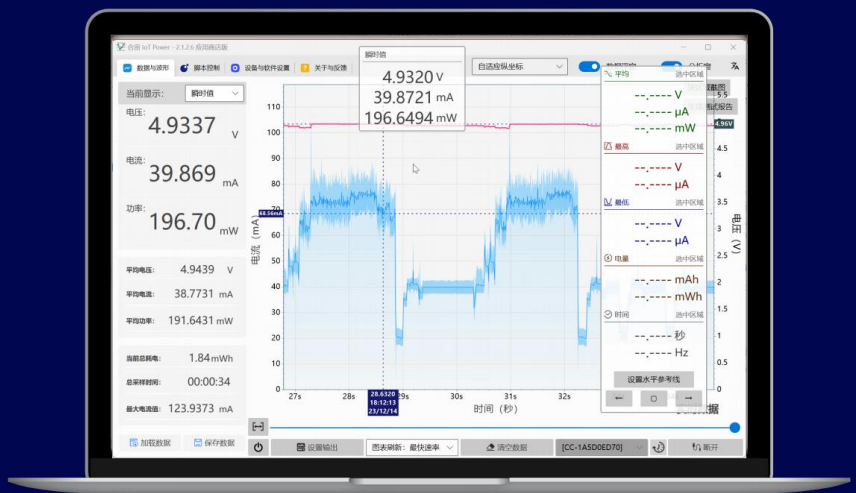
开启模拟电池功能

- 可在【设置】页面查看当前池设置信息；
- 使用滚轮选中启用模拟电池控件；
- 按下滚轮即可切换开启与关闭。

修改电池电压曲线与电量

- 将设备通过USB连接至PC；
- 通过PC软件设置页面，修改模拟电池相应的参数，并更新到设备中。

七. 合宙功耗分析仪——PC端软件功能



PC端软件主要功能：

- 实时显示电压电流动态波形
- **分析指定时间段的功耗数据**
- 记录指定时间段的历史数据
- 导出历史数据且支持导出为独立.exe文件
- 导入历史数据并回放波形动态变化全过程
- 运行Lua脚本控制电源输出，可模拟电池放电曲线等
- 为合宙功耗分析仪各型号升级固件
- **微软商店下载：**点此打开微软商店页面
- DLL/SO动态运行库：Gitee iot-power dll/so

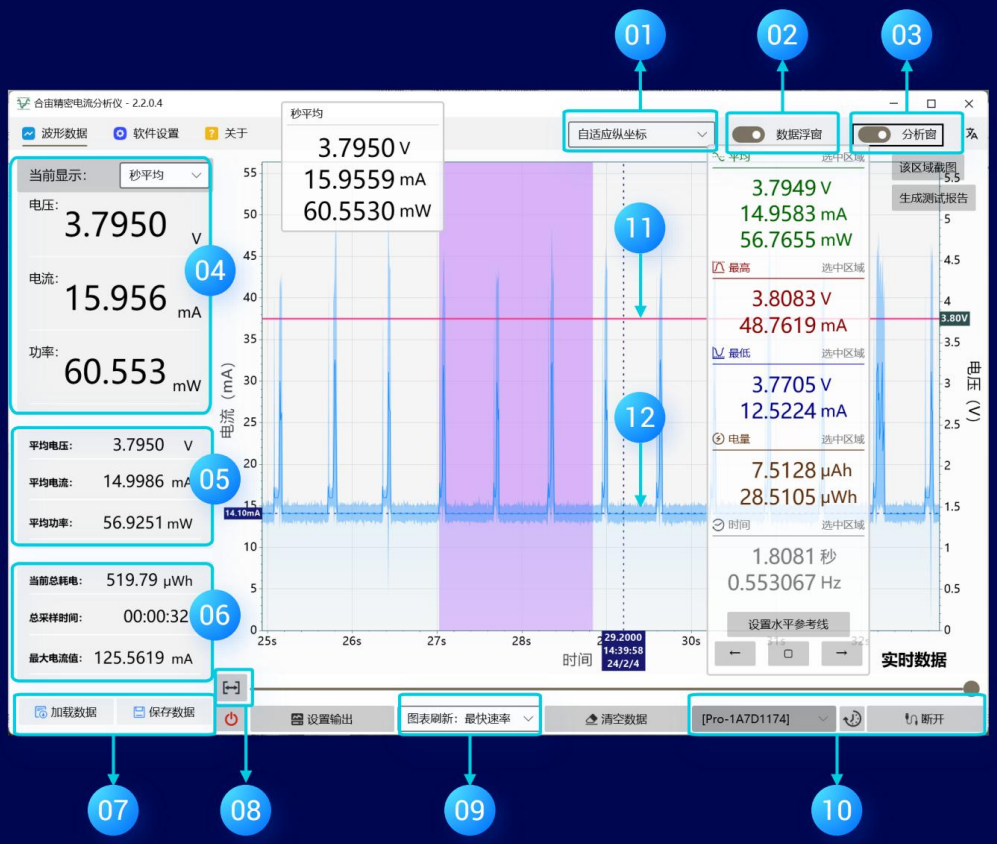
- 平均电流
- 平均电压
- 平均功率
- 峰值、最小电流
- 峰值、最小电压

www.hezhouyibiao.com

- 让 你 的 硬 件 更 值 钱 -
降功耗，找合宙！

20

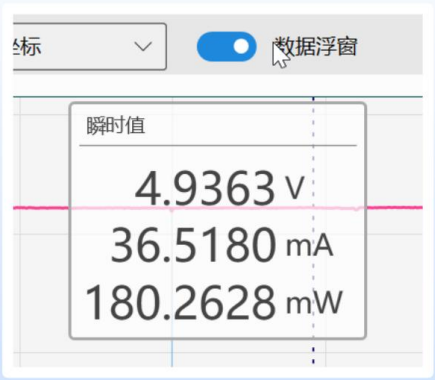
八. 合宙功耗分析仪——PC端软件界面



8.01 界面功能说明（对应图示序号）

01. 图表展示方式的偏好
02. 当前数据的置顶悬浮窗
    方便在软件最小化时查看数据
03. 选定区域数据分析
04. 当前数据可选择显示：
    瞬时值/秒平均值/分钟平均值
05. 本次抓取数据的平均值
06. 其他的各项信息
07. 导出或导入数据
08. 切换按钮：当前/圈选
09. 确定波形显示区域的下拉选项
10. 控制设备的连接与断开
11. 粉色曲线为电压
12. 蓝色曲线为电流

八. 合宙功耗分析仪——PC端软件界面



8.02 数据浮窗区域

界面右上方的数据浮窗开关可以打开或关闭浮窗；

数据浮窗打开后此区域的数据与当前显示区域的数据完全相同且始终置顶显示，方便软件最小化后查看实时数据。

特别说明：

当前显示数据若选择瞬时值档位，可能因刷新时间点不完全一致而导致PC端与仪表端数据不一致，为正常现象。

8.03 当前显示区域

当前显示区域数值为设备连接后当前的数据，可以切换为滑动平均值或瞬时值。

八. 合宙功耗分析仪——PC端软件界面

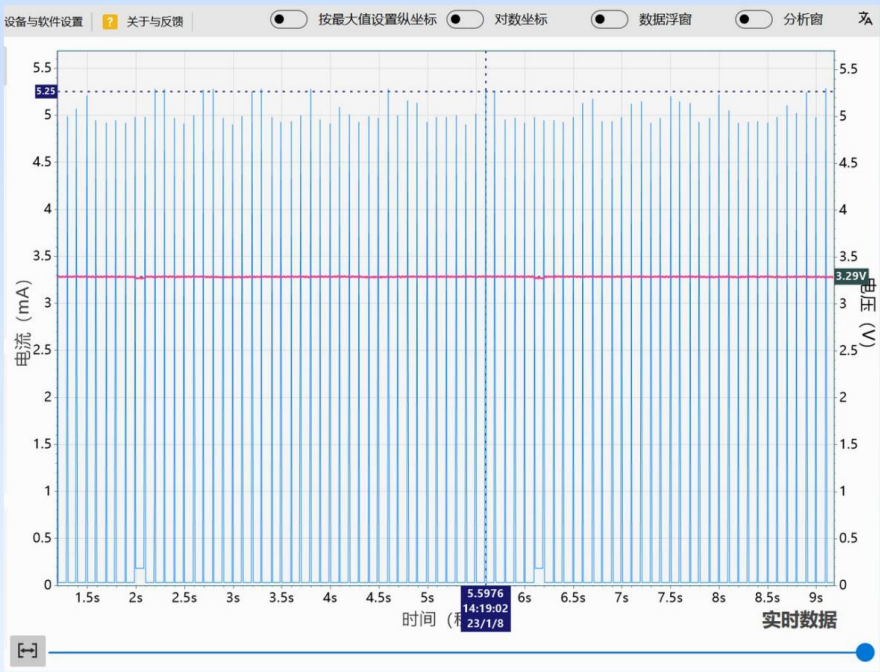
8.04 平均数据区域

当前显示区域数值为设备连接后当前的数据，可以切换为滑动平均值或瞬时值。

8.05 其他数据区域

其它数据区域的数据为被测设备本次连接后所有数据的累计数据，单击当前总耗电区域的数值区域，可以切换Wh与Ah不同电量单位。

八. 合宙功耗分析仪——PC端软件界面



8.06 图表显示区域

该区域为图表显示区域，可以预览被测设备实时电压电流动态波形。

调整显示方式

- **自适应坐标：**按照当前显示区域的最大值，动态地决定纵坐标范围；
- **按最大值设置纵坐标：**按此次抓取数据的最大值，固定纵坐标范围；
- **对数坐标：**当电流数据绝对值相差极大时可启用该功能，将以10的倍数来显示纵坐标数值。

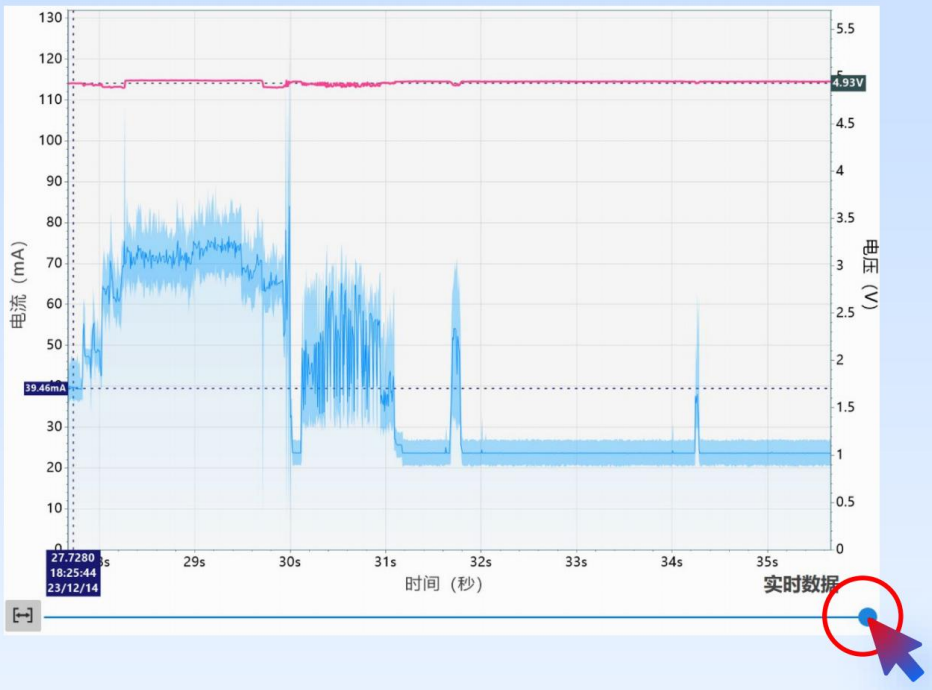
八. 合宙功耗分析仪——PC端软件界面

8.07 切换数据显示按钮

按下该按钮，图表显示区域可在**实时数据/历史数据**和**选取某个区域**两种显示模式中切换。

Air9000P

八. 合宙功耗分析仪——PC端软件界面



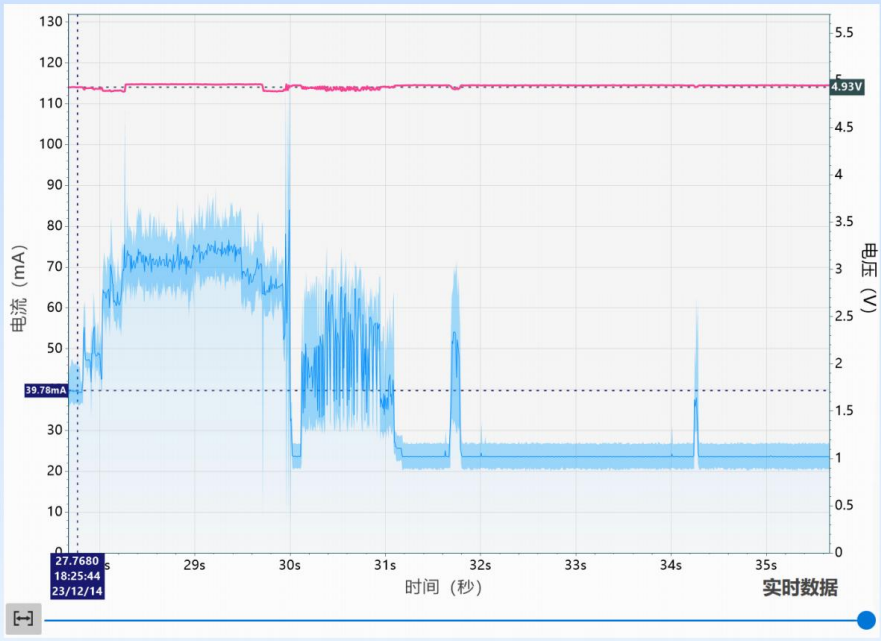
8.08 实时数据/历史数据（不能缩放显示和分析）

当鼠标在图表显示区域或在下方拖动轴上时，可以使用鼠标滚轮，调整显示不同时间段的波形；

也可以直接拖动下方拖动轴上的操纵杆，调整显示不同时间段的波形。

Air9000P

八. 合宙功耗分析仪——PC端软件界面



8.09 选取某个特定区域（可以缩放显示和分析）

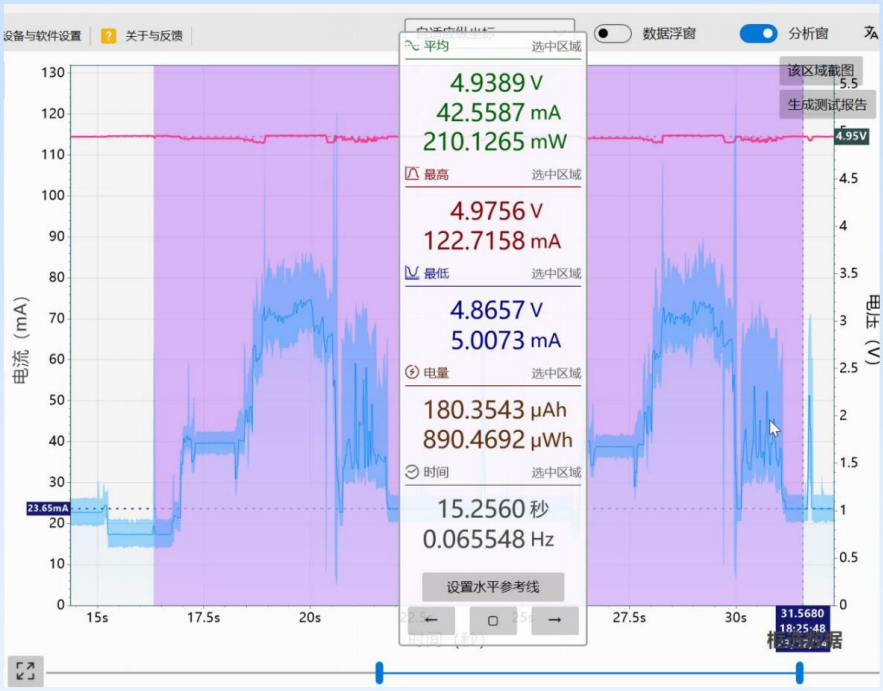
使用**鼠标左键**拖动，可以选取查看一个区域的数据，并进入**选取区域数据模式**。

当处于选取区域数据模式时，可以使用鼠标滚轮调整波形的时间位置，也可以按住CTRL键并滚动鼠标滚轮放大或缩小当前显示的区域。

下方拉杆同样也可以用于控制选择显示的范围：

- **鼠标拖动中间：**可以调整波形的时间位置；
- **鼠标拖动左边或右边的竖线：**可以调整波形显示的开始或结束时间点；
- **鼠标双击中间：**可以直接切换为查看全局数据。

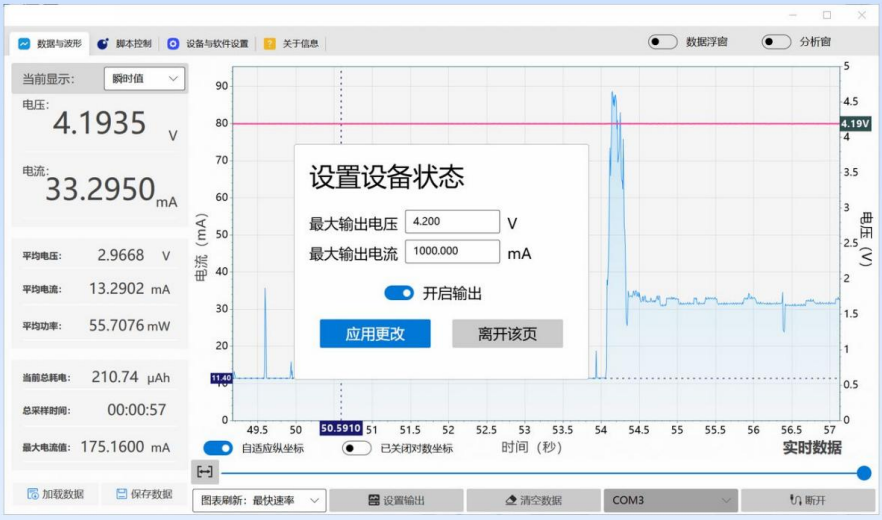
八. 合宙功耗分析仪——PC端软件界面



8.10 分析数据

任意模式下，**右击鼠标拖动并滑动图标显示区域，**即可选定一个时间段并分析其数据，分析窗将弹出并显示分析的结果。

八. 合宙功耗分析仪——PC端软件界面



8.11 控制设备输出

- 通过电源符号按钮，控制设备的输出、关闭状态;
- 通过设备输出按钮，设置设备的输出电压与电流。

特别说明：

Air9000需配合**支持PD/PPS快充协议的充电头**使用，推荐使用合宙随机附带快充充电头。

Air900P

八. 合宙功耗分析仪——PC端软件界面

8.12 导出数据

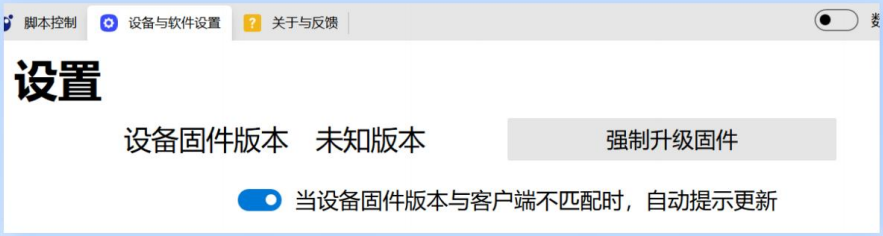
数据可自由地导出为.iotpl文件、.csv表格文件、.exe自执行文件；

- .csv表格文件用于用户自行分析原始数据，无法再重新导入，.iotpl文件可以再次导入PC端软件；
- .exe为自执行文件，运行后可以再次导出为.iotpl文件用于导入PC端软件。

8.13 导入数据

- 手动选择导入Air9000专用的.iotpl文件；
- 导入从自动保存的文件夹中的.iotpl文件，该功能通常出现在功能关闭或重新抓取时。

八. 合宙功耗分析仪——PC端软件界面



8.14 强制升级软件

通常来说，设备固件会自动提示升级，但当设备升级途中中断或因其他原因导致固件损坏时，可手动刷入固件；

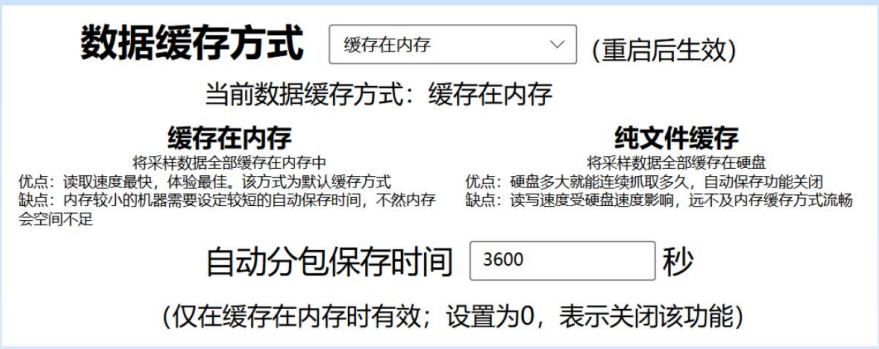
- 单击强制升级固件按钮，可强制刷入客户端自带的固件。

  注意：若设备有升级模式，请先进入升级模式后再刷固件。

- 右击强制升级固件按钮，可强制刷入用户手动选择的固件文件，用于测试功能、验证BUG修复等。

Air9000P

八. 合宙功耗分析仪——PC端软件界面



8.15 数据缓存方式

因为合宙高精度电流分析仪采样率较高，数据量十分庞大，存储在内存中可能导致电脑运行内存不足，所以PC端软件提供了自动保存功能，当达到设置的时间后数据将自动保存至指定文件中，防止占用过多电脑内存。

为了存储特定需求场景下海量数据存储的需求，比如连续抓取24小时，PC端软件也提供了使用**硬盘缓存数据的功能。**

在该模式下，PC端软件将被牺牲部分的流畅性以换来内存占用的大大降低，并且不会触发自动保，保证数据的连续性。

Air9000P

14.4 Air9000P 上手指南

Air9000P 上手指南可通过点击右方链接在线阅读：合宙功耗分析仪 Air9000P 上手指南

也可直接阅读本章节以下内容：

合宙功耗分析仪 Air9000P 上手指南

上海合宙通信科技有限公司

一、Air9000P 展示

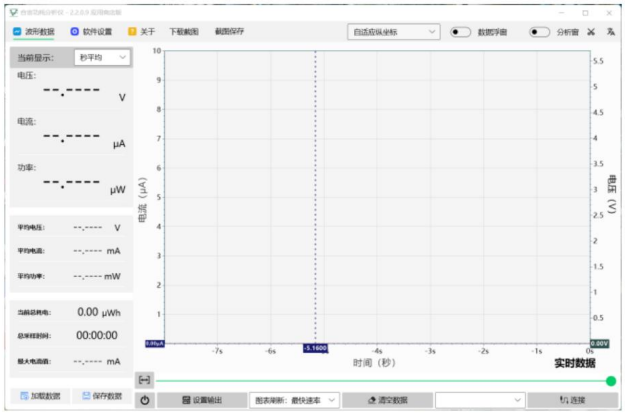
产品展示

> Air9000P*1（金色 logo）
>
> A2C 数据线*1（印有“上海合宙”字样）
>
> 2.0mm 枪式香蕉插头转鳄鱼夹测试线*2
>
> 定制版高档硬壳收纳包（防摔防水特厚耐磨便携）*1

二、上位机软件-合宙功耗分析仪

下载安装

打开 Microsoft Store，下载“合宙功耗分析仪”



合宙功耗分析仪-界面

高速动态变化的微弱电流抓取
电流波形分析
上位机软件电流波形分析
PC端软件主要功能
实时显示电压电流动态波形
分析指定时间段的功耗数据
记录指定时间段的历史数据
导出历史数据且支持导出为独立 .exe 文件
导入历史数据并回放波形动态变化全过程
运行 Lua 脚本控制电源输出，可模拟电池放电曲线等
为合宙功耗分析仪各型号升级固件

界面功能说明
01 图表展示方式的偏好
02 当前数据的置顶悬浮窗
方便在软件最小化时查看数据
03 选定区域数据分析
04 当前数据可选择显示
瞬时值/秒平均值/分钟平均值
05 本次抓取数据的平均值
06 其他的各项信息
07 导出或导入数据
08 切换按钮:当前/圈选
09 确定波形显示区域的下拉选项
10 控制设备的连接与断开
11 粉色曲线为电压
12 蓝色曲线为电流

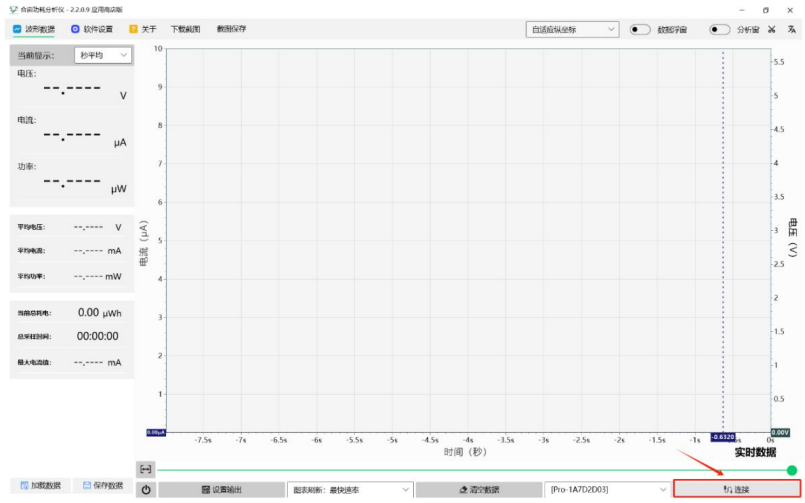
三、连接 Air9000P

首次连接

> 自动弹出“用前须知”

固件升级

> 点击“连接”-弹出“是否升级”窗口，点击“确定”-升级完毕
> 设备固件升级过程中，请保持连接，切勿断开设备



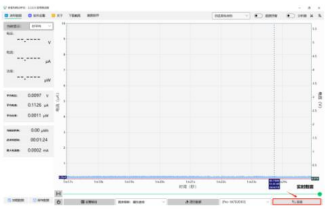
是否升级?
此版本客户端仅适配设备固件1.0.1版本
按确定进入升级操作
确定
取消
设备固件升级中

提示
升级成功！
确定

四、开始测试（以测“合宙门铃”为例）

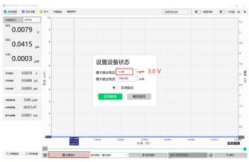
连接设备

> 固件升级完成后，需再次连接设备，点击“连接”



调整输出电压与电流

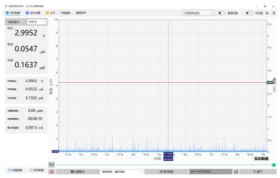
> 方法一：在上位机软件点击“设置输出”-设置“最大输出电压/最大输出电流”；显示屏会同步更改
>
> 方法二：使用 Air9000P 上的轮盘进行调整；软件会同步更改

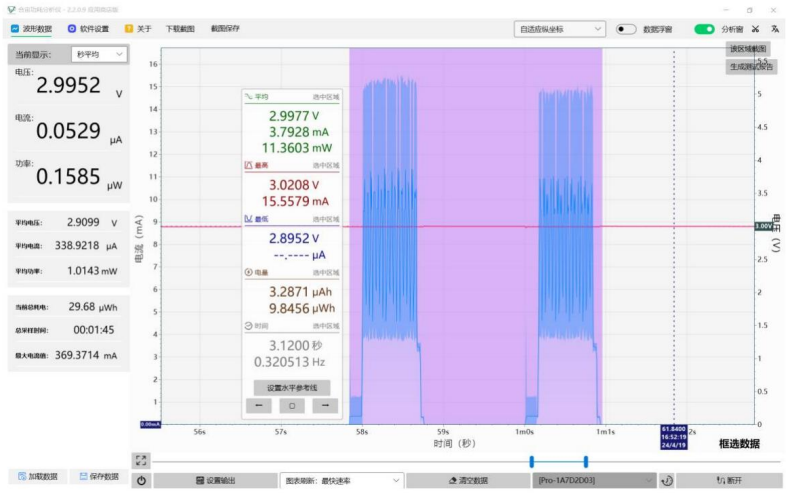


> 其他界面

连接待测物

> 连接待测物，按下开关





五、其他资料

有关“Air9000P 测功耗”的更多内容，请在“降功耗找合宙”公众号阅读

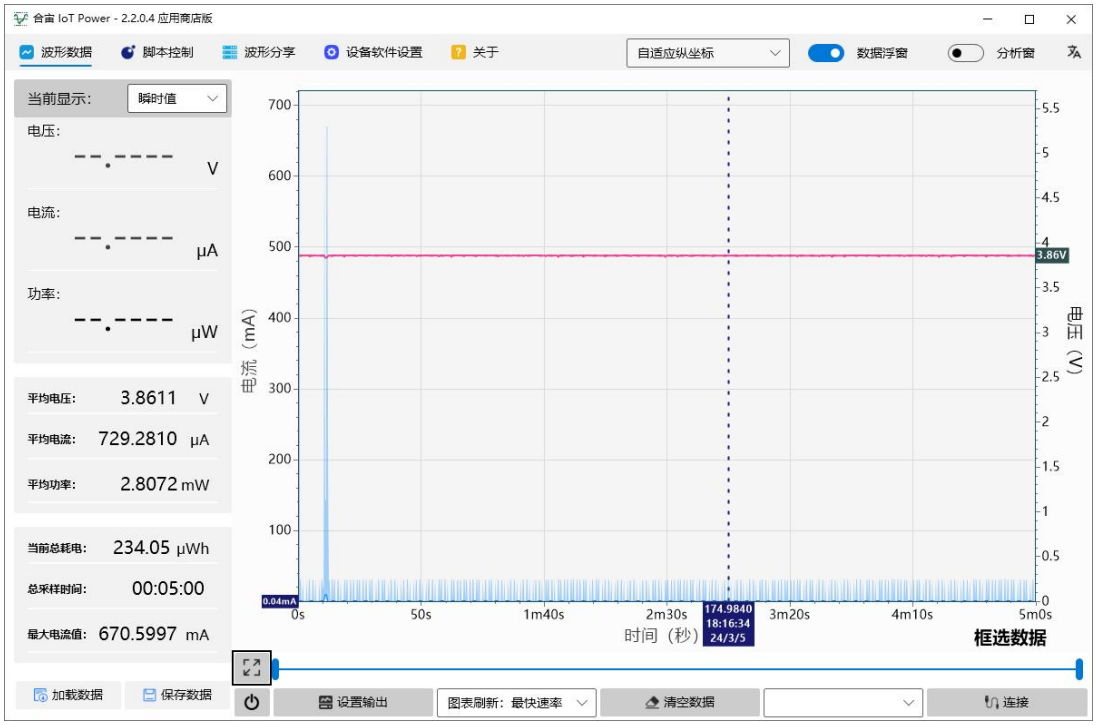
> 小白测功耗|蓝牙温湿度传感器：
> https://mp.weixin.qq.com/s/kE9PTPWxEzFg2kk_-fmleg
> 小白测功耗|门铃：https://mp.weixin.qq.com/s/X26xLpZ9ZXHKDX5toSvSZA

十五 实网环境功耗测试电流曲线

移动卡+500 微安模式+每隔 5 分钟发送 1KB TCP 数据

平均电压：3.8611V，平均电流：729.2810uA，平均功率：2.8072mW，

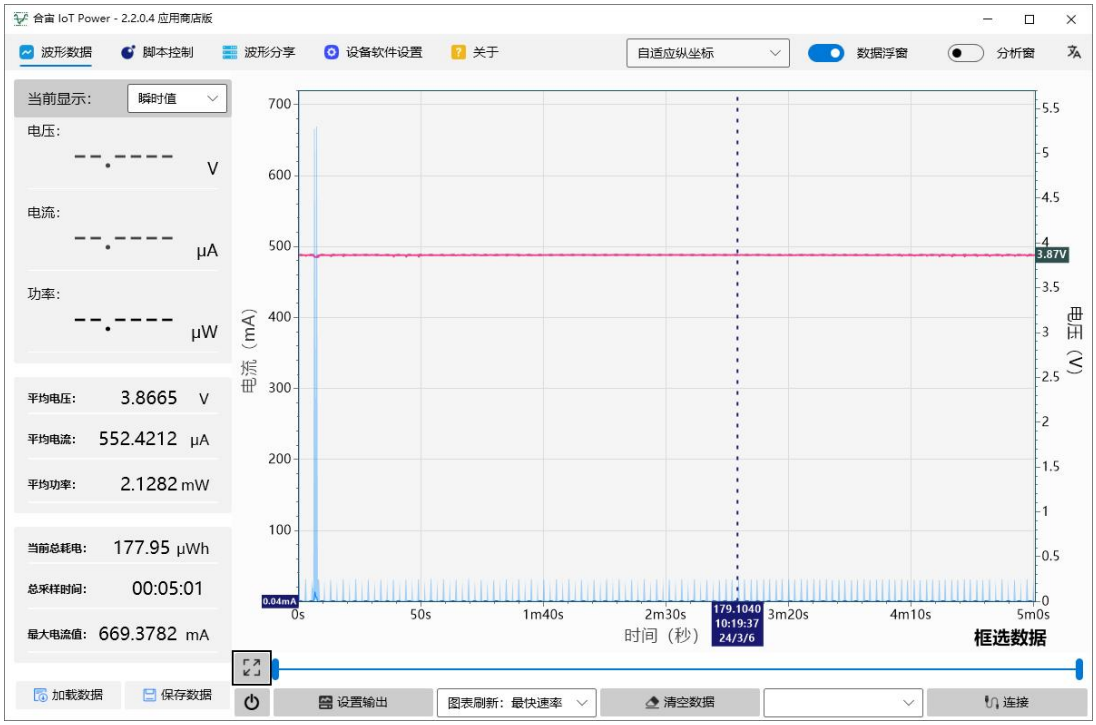
当前总耗电：234.05uWh，总采样时间 5 分钟，最大电流值：670.5997mA。



移动卡+200 微安模式+每隔 5 分钟发送 1KB TCP 数据

平均电压：3.8665V，平均电流：552.4212uA，平均功率：2.1282mW，

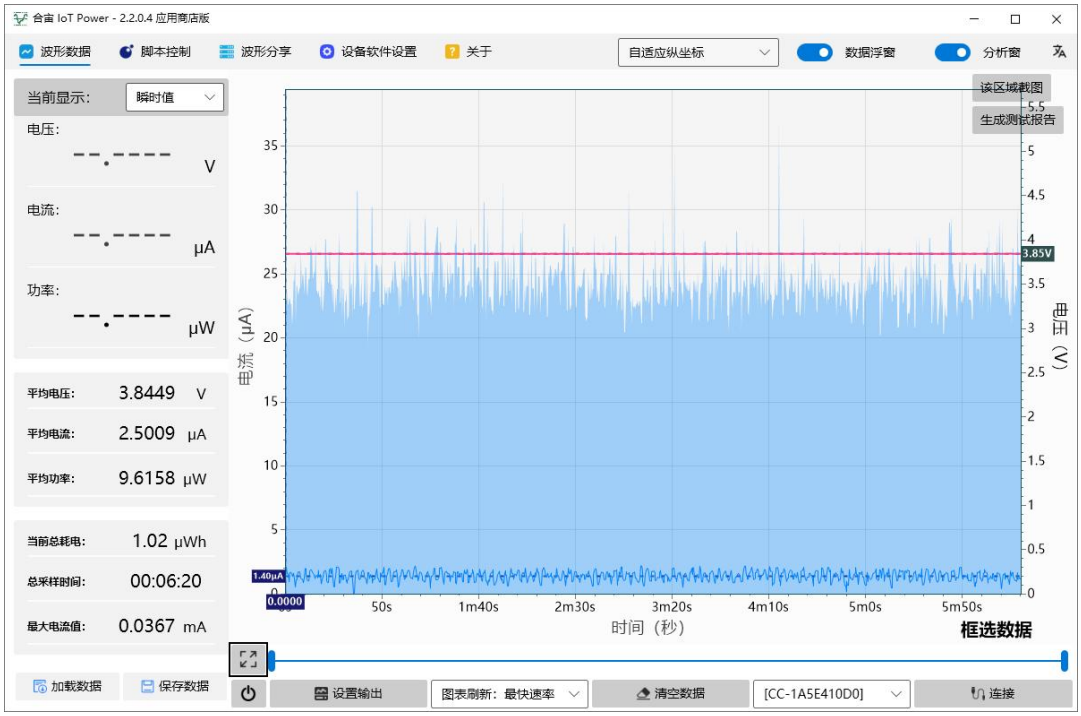
当前总耗电：177.95uWh，总采样时间 5 分钟零 1 秒，最大电流值：669.3782mA。



移动卡+2 微安模式+无业务数据交互

平均电压：3.8449V，平均电流：2.5009uA，平均功率：9.6158uW，

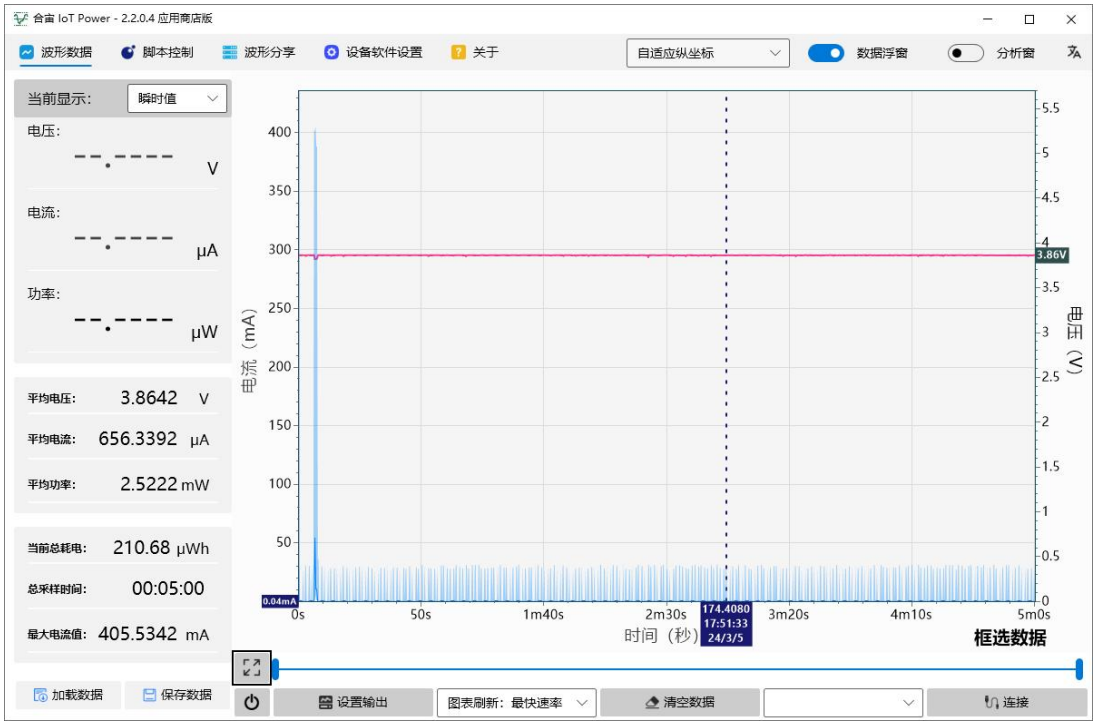
当前总耗电：1.02uWh，总采样时间 6 分钟 20 秒，最大电流值：0.0367mA。



电信卡+500 微安模式+每隔 5 分钟发送 1KB TCP 数据

平均电压：3.8642V，平均电流：656.3392uA，平均功率：2.5222mW，

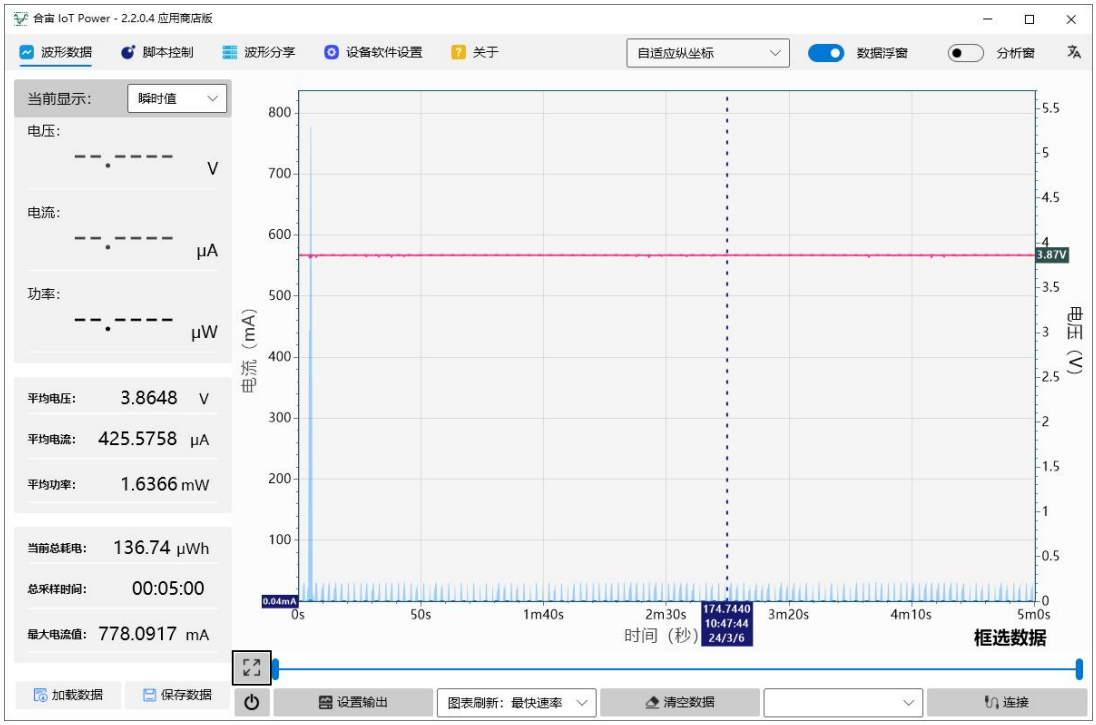
当前总耗电：210.68uWh，总采样时间 5 分钟，最大电流值：405.5342mA。



电信卡+200 微安模式+每隔 5 分钟发送 1KB TCP 数据

平均电压：3.8648V，平均电流：425.5758uA，平均功率：1.6366mW，

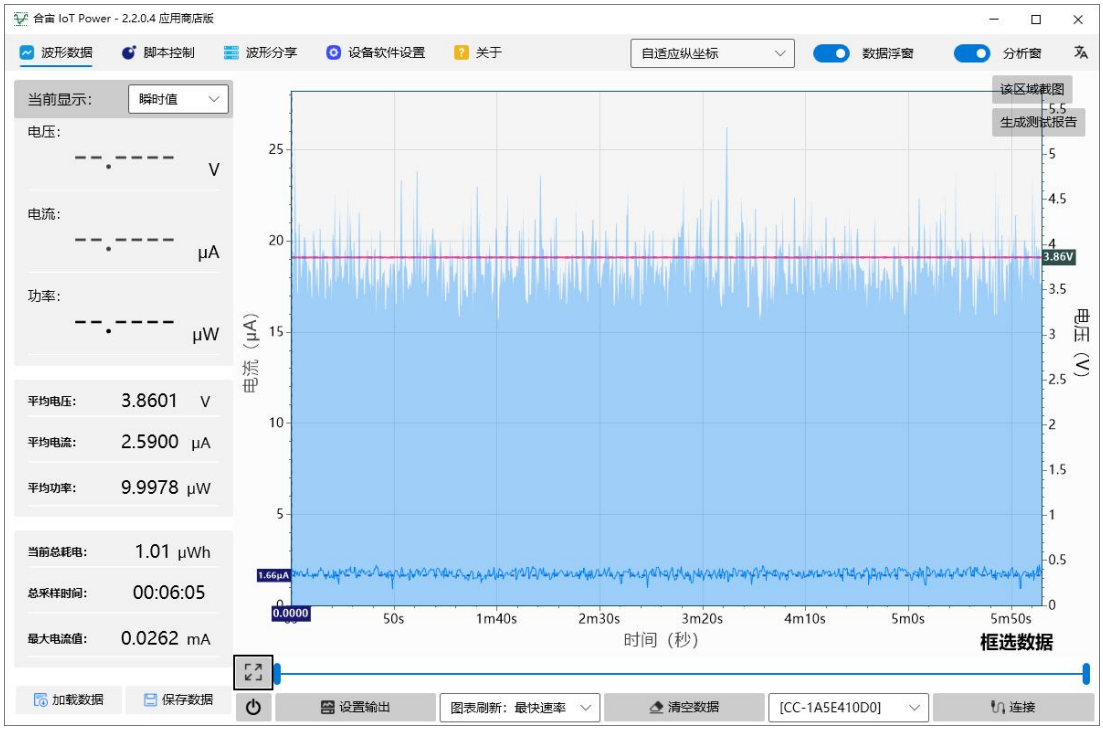
当前总耗电：136.74uWh，总采样时间 5 分钟，最大电流值：778.0917mA。



电信卡+2 微安模式+无业务数据交互

平均电压：3.8601V，平均电流：2.5900uA，平均功率：9.9978uW，

当前总耗电：1.01uWh，总采样时间 6 分钟零 5 秒，最大电流值：0.0262mA。



十六 Lua 语言学习教程

1 序言

1.1 初识 Lua

1.2 第一个 Lua 程序

1.3 输出数据

1.4 代码注释

1.1 初识 Lua

Lua 是一种轻量小巧的脚本语言，它用标准 C 语言编写并以源代码形式开放。这意味着什么呢？这意味着 Lua 虚拟机可以**很方便的嵌入别的程序里**，从而为应用程序提供灵活的扩展和定制功能。而整个 Lua 虚拟机编译后仅仅一百余 K，经过适当的裁剪还能做到更小，十分适合嵌入式的开发。

同时，在目前脚本引擎中，Lua 的运行速度占有绝对优势。这些都决定了 Lua 是作为嵌入式脚本的最佳选择。

在我们编写代码之前，需要做一下准备，就是：**确保你输入标点符号时，用的不是中文输入法。**

1.2 第一个 Lua 程序

几乎所有语言的第一行代码，都是输出 hello world，本教程也不意外。

在 Lua 中，打印结果只需要使用 print 这个函数即可。同时，如果需要使用函数，只需要在函数名后加上双括号，同时传入你想传入的值即可。

所以，我们来执行下面的代码，打印出 hello world 吧！

点我快速测试下面的代码

```
print("hello world!")
```

1.3 输出数据

上一部分，我们知道了，在 Lua 中，可以使用 print 函数来打印你想要得到的结果。

并且还知道了，函数是指可以实现某些功能的子程序，可以使用函数名(参数)来执行。

让我们试着输出一些其他东西吧！使用多个 print 函数，输出自己想输出的数据。

点我快速测试下面的代码

```
print("测试")
print("aabcdefg")
print("xxxxx","第二个参数","第三个参数")
```

1.4 代码注释

代码注释就是在代码里，不会运行的部分。注释完全不会被运行。

这部分是为了在查看代码时，可以更好地立即现有代码含义用的。

我们可以用--开头，来写一段单行注释

也可以用--[[开头，]]结尾，写一段多行注释。

下面是注释的例子：

点我快速测试下面的代码

```
print("这段代码会运行")
--print("我被注释掉了，所以不会运行")
--[[
    我是多行注释
    不管我写多少行
    都不会影响代码运行
]]
```

2 变量

2.1 number 变量
2.2 了解 nil
2.3 赋值语句
2.4 交换变量
2.5 输出变量
2.6 算数运算符

2.1 number 变量

变量，可以看作是一个桶，在里面装你想要装的内容。这些内容可以是 Lua 包含的所有合法类型。

例如：我想要新建一个桶，名叫 bucket，在里面放入 233 这个数字，就可以像下面一样：

```
bucket = 233
```

让我们试着自己新建几个变量吧！

- 新建变量 year，并将变量的值设置为 1926
- 新建变量 month，并将变量的值设置为 8
- 新建变量 day，并将变量的值设置为 7

点我快速测试下面的代码

```
--新建三个变量，并赋值
year = 1926
month = 8
```

```
day = 7
--打印出三个变量的值
print(year,month,day)
```

2.2 了解 nil

nil 类型表示没有任何有效值，只要是没有声明的值，它就是 nil

比如我打印一个没有声明的值，便会输出 nil：

点我快速测试下面的代码

```
ccc = 233
print(ccc)
print(aaa)
```

这里需要你思考一下，运行以下代码，将会输出什么结果？

点我快速测试下面的代码

```
a = 1
b = '2'
c = a
print(a,b,c,d)
```

2.3 赋值语句

赋值是改变一个变量值的最基本的方法。

如下面一样，使用等号对左边的变量进行赋值

```
n = 2
n = 3
n = n + 1
b = n
```

Lua 可以对多个变量同时赋值，变量用逗号分开，赋值语句右边的值会依次赋给左边的变量。

```
n = 1
a, b = 10, 2*n
```

当左右值的数量不一致时，Lua 会进行下面的设定：

- 变量个数 > 值的个数：按变量个数补足 nil
- 变量个数 < 值的个数：多余的值会被忽略

下面的例子可以展示这种设定：

点我快速测试下面的代码

```
a, b, c = 0, 1
print(a,b,c)
--输出 0   1   nil

a, b = a+1, b+1, b+2
print(a,b)
--输出 1   2

a, b, c = 0
print(a,b,c)
--输出 0   nil   nil
```

这里需要你思考一下，运行以下代码，将会输出什么结果？

点我快速测试下面的代码

```
a,b,c = 1,2,3
a,c = a+1,b
d = c,b
print(a,b,c,d)
```

2.4 交换变量

这部分需要你自己完成一个任务：

已知下面的代码，并且已知 a 和 b 的值，请在交换他们的值，使打印输出 12 34

点我快速测试下面的代码

```
a = 34
b = 12

--你需要在这里进行交换变量的操作
--提示：新建一个变量来存放临时的值

print(a,b)
```

2.5 输出变量

我们已经知道了，在 Lua 中，可以使用 print 函数来打印你想要得到的结果。

同时在上一节，我们学会了新建变量和设置变量的值。

让我们试着输出某个变量吧！使用 print 函数，输出已知变量。 我们已知变量 num 为某个数字，试着输出它的值吧！

点我快速测试下面的代码

```
num = 123
--请补全代码，输出 num 的值
print(你要在这里填东西)
```

2.6 算数运算符

运算符是一个特殊的符号，用于告诉解释器执行特定的数学或逻辑运算。

上一节中，新建的数字变量，我们称之为 number 类型的变量。

本节我们来学习使用算术运算符，如下所示：

+ 加法
- 减法
* 乘法
/ 除法

% 取余，求出除法的余数

^ 乘幂，计算次方

- 负号，取负值

我们可以通过以下实例来理解算术运算符的应用：

点我快速测试下面的代码

```
a = 21
b = 10
c = a + b
print('a + b 的值为 ', c )
c = a - b
print('a - b 的值为 ', c )
c = a * b
print('a * b 的值为 ', c )
c = a / b
print('a / b 的值为 ', c )
c = a % b
print('a % b 的值为 ', c )
c = a^2
print('a^2 的值为 ', c )
c = -a
print('-a 的值为 ', c )
c = a * (b - a)
print('a * (b - a) 的值为 ', c )
```

你需要完成下面的任务：

已知，一个长方体的长宽高分别为 a、b、c（单位米），且这个物体重量为 m（单位克）

请打印出物体的密度（单位 g/m³）

注：密度计算公式 密度 = 质量 / 体积

```
a,b,c = 1,2,3 --长宽高
m = 10        --重量
--请打印出物体的密度
```

---

3 字符串

3.1 string 类型变量
3.2 转义字符
3.3 string 拼接
3.4 number 转 string
3.5 string 转 number

3.1 string 类型变量

字符串（即 string），就是一串文本数据，可以存储你要的文本。

在第二节中，print 出的数据就是一个字符串。

Lua 语言中字符串可以使用以下三种方式来表示：

1. 单引号间的一串字符
2. 双引号间的一串字符
3. [[和]]间的一串字符

你可以参考下面的例子来深入理解：

点我快速测试下面的代码

```
--双引号间的一串字符
str1 = "Lua"
--单引号间的一串字符
str2 = 'Lua'
--[[和]]--间的一串字符
str3 = [[Lua]]
str4 = [[使用双括号时，甚至能包含换行数据
换行了
最后一行]]
```

```
--输出所有字符串
print(str1)
print(str2)
print(str3)
print(str4)
```

接下来你需要完成下面的练习：

新建三个变量 s1、s2、s3

分别存入字符串数据：str、abc、233，使输出打印正确

点我快速测试下面的代码

```
--请在空白处补全代码

print("s1,s2,s3 的值：",s1,s2,s3)
print("s1,s2,s3 的类型：",type(s1),type(s2),type(s3))
```

3.2 转义字符

在上一节中，我们学习了如何声明字符串。

但是我们有时候会遇到一些特殊的问题，如：**如何输出单引号和双引号？如何输出回车换行？**

也许我们可以用下面的方式简单规避，输出单引号时，声明字符串用双引号括起来，像下面这样

```
str = "'"
```

同理，输出双引号时，声明字符串用单引号括起来，像下面这样

```
str = '"'
```

但是，这样会出现一个问题：如何同时显示单引号和双引号？这里就需要转义字

符登场了。

转义字符用于表示不能直接显示的字符，比如后退键、回车键、等。

以 \ 开头的都是转义字符，下面时常用的转义字符格式：

转义字符	含义
\n	换行(LF),将当前位置移到下一行开头
\r	回车(CR),将当前位置移到本行开头
\\	反斜杠字符\
‘	单引号
“	双引号
\0	空字符(NULL)
\ddd	1 到 3 位八进制数所代表的任意字符
\xhh	1 到 2 位十六进制所代表的任意字符

例如，如果我们想给 str 赋值一个单引号，一个双引号（”），那么我们可以这样写：

```
str = "\'\""
```

下面需要你来完成一个简单的任务：

新建一个变量 str，给 str 赋值为

```
ab\cd"ef'g\h]]
```

并打印出来

点我快速测试下面的代码

```
str = --补全代码
print(str)
--输出应为 ab\cd"ef'g\h]]
```

3.3 string 拼接

字符串和字符串可以相加吗？可以！我们可以用拼接符号来将两个独立的字符串拼起来。

我们使用..来表示字符串拼接符号，如下面的示例代码：

点我快速测试下面的代码

```
print('abc'..'def')
str1 = '123'
str2 = '999'
print(str1..str2)
```

下面你要完成这个任务：

已知三个字符串变量 s1、s2、s3

请将他们按顺序拼接起来，存入 all，并使用 print 输出结果

点我快速测试下面的代码

```
s1,s2,s3 = "aaa","bbb","ccc"
all = --请补全代码
print(all)
```

3.4 number 转 string

上面一节学习了如何拼接字符串，这个方法固然很好用，但是有时候我们会遇到一个需求，那就是把 number 类型的变量和 string 类型的变量拼接起来，组成一个新的 string

比如下面的变量 n 和 s，如何拼接起来呢？

```
n = 123
s = 'm/s'
```

我们可以直接将 number 类型的变量 n 转换成 string 类型的值，这样就可以拼接了

使用 tostring(value)函数即可实现这一操作：

点我快速测试下面的代码

```
n = 123
s = 'm/s'

result = tostring(n)..s
print(result)
```

下面你要完成这个任务：

已知三个变量 n1、s、n2

然后将他们按顺序拼接起来，存入变量 result，使输出结果正确

小提示：在某些情况下，Lua 会自动将 number 类型转换成 string 类型

点我快速测试下面的代码

```
n1,s,n2 = 1,"abc",2
result = --请补全代码
print(result)
```

3.5 string 转 number

上面一节学习了如何将 number 转成 string，这一节我们来学习如何将 string 转成 number

比如下面的变量 s，存储的内容是一个字符串，但是代表了一个数字，如何转成 number 再与 n 相加计算呢？

```
n = 123
s = '2333'
```

我们可以直接将 string 类型的变量 s 转换成 number 类型的值，这样就可以计算了

使用 tonumber(value)函数即可实现这一操作：

点我快速测试下面的代码

```
n = 123
s = '2333'

result = tonumber(s) + n
print(result)
```

下面你要完成这个任务：

已知三个字符串变量 s1、s2、s3，其内容均为纯数字

请计算他们的算术和，赋值给新建的变量 result，使下面代码输出正确结果

点我快速测试下面的代码

```
s1,s2,s3 = "11","12","100"
result = --请补全代码
print(result)
```

---

4 逻辑运算

4.1 布尔型和比较运算
4.2 逻辑运算符
4.3 检验大小（自测题）

4.1 布尔型和比较运算

布尔型（boolean）只有两个可选值：true（真） 和 false（假）

Lua 把 false 和 nil 看作是 false，其他的都为 true（包括 0 这个值，也是相当于 true）

Lua 中也有许多的关系运算符，用于比较大小或比较是否相等，符号及其含义如下表：

符号	含义
==	等于，检测两个值是否相等，相等返回 true，否则返回 false

符号	含义
~=	不等于，检测两个值是否相等，相等返回 false，否则返回 true
>	大于，如果左边的值大于右边的值，返回 true，否则返回 false
<	小于，如果左边的值大于右边的值，返回 false，否则返回 true
>=	大于等于，如果左边的值大于等于右边的值，返回 true，否则返回 false
<=	小于等于， 如果左边的值小于等于右边的值，返回 true，否则返回 false

我们可以通过以下实例来更加透彻的理解关系运算符的应用：

点我快速测试下面的代码

```
a = 21
b = 10
print('==的结果',a==b)
print('~=的结果',a~=b)
print('>的结果',a>b)
print('<的结果',a<b)
print('>=的结果',a>=b)
print('<=的结果',a<=b)
```

下面问题来了，运行以下代码，将会输出什么结果？请自行思考

点我快速测试下面的代码

```
a = 1
b = '1'
c = a
d = 2

print(a == b)
print(c == a)
print(a ~= b)
print(d <= c)
```

4.2 逻辑运算符

逻辑运算符基于布尔型的值来进行计算，并给出结果，下表列出了 Lua 语言中的常用逻辑运算符：

符号	含义
and	逻辑与操作符。 若 A 为 false，则返回 A，否则返回 B
or	逻辑或操作符。 若 A 为 true，则返回 A，否则返回 B
not	逻辑非操作符。与逻辑运算结果相反，如果条件为 true，逻辑非为 false

我们可以通过以下实例来更加透彻的理解逻辑运算符的应用：

点我快速测试下面的代码

```
print('true and false 的结果',true and false)
print('true or false 的结果',true or false)
print('true and true 的结果',true and true)
print('false or false 的结果',false or false)
print('not false 的结果',not false)
print('123 and 345 的结果',123 and 345)
print('nil and true 的结果',nil and true)
```

下面问题来了，运行以下代码，将会输出什么结果？

点我快速测试下面的代码

```
a = 1
b = '1'
c = 0

print(a and b)
print(c or a)
print(not b)
print(d and c)
print(1 < 2 and 3 > 2)
```

4.3 检验大小（自测题）

题目：如果已知 number 变量 n，那么如果需要判断 n 是否符合下面的条件：

3<n≤10

以下四行判断代码，正确的是？

（返回 true 即表示变量 n 符合要求）

点我快速测试下面的代码

```
n = 1--这个数可能是任意数字
print(n > 10 or n <= 3)
print(n <= 10 or n > 3)
print(n < 3 and n >= 10)
print(n <= 10 or n > 3)
```

---

5 分支判断

5.1 条件判断
5.2 多条件判断
5.3 判断三角形合法性（自测题）
5.4 if 的判断依据（自测题）

5.1 条件判断

上面一节学习了布尔类型，那么这个需要用到哪里呢？我们需要用它来进行某些判断。

在 Lua 中，可以使用 `if` 语句来进行判断，如下面所举例的代码，可以判断 n 是否为小于 10 的数：

点我快速测试下面的代码

```
n = 5
if n < 10 then
    print('n 小于 10')
end
```

我们整理一下，实际上 if 语句就是如下结构：

```
if 条件 then
    符合条件的代码
end
```

下面是你需要完成的事：

已知变量 n，请判断 n 是否为奇数，如果是，请给 n 的值加上 1

如果你觉得有难度，请查看下面的提示：

- 求出 n 除以 2 的余数：n % 2
- 给 n 的值加上 1：n = n + 1

点我快速测试下面的代码

```
--已知一个 number 变量 n
n = 5   --这个数字是几都有可能
print("n 在判断前的值",n)

if 你的条件 then
    要做的事
end
print("n 在判断后的值",n)
```

5.2 多条件判断

上面一节学习了简单的 if 语句写法，这一节我们来学习多条件分支语句

在 Lua 中，可以使用 if 语句来进行判断，同时可以使用 else 语句，表示多个分支判断

```
if 条件 1 then
    满足条件 1
elseif 条件 2 then
    不满足条件 1，但是满足条件 2
```

```
else
    前面条件全都不满足
end
```

举个例子，比如有一个数字 n：

- 当它大于等于 0、小于 5 时，输出太小，
- 当它大于等于 5、小于 10 时，输出适中，
- 当它大于等于 10 时，输出太大，

那么代码就像如下这样：

点我快速测试下面的代码

```
n = 1--更改这个数多运行几次试试
if n >= 0 and n < 5 then
    print('太小')
elseif n >= 5 and n < 10 then
    print('适中')
elseif n >= 10 then
    print('太大')
end
```

注意：else 和 elseif 都是可选的，可有可无，但是 end 不能省略

下面是你需要完成的事：

已知变量 n，请判断 n 是否为奇数，

- 如果是，请给 n 的值加上 1
- 如果不是，请将 n 的值改为原来的两倍

点我快速测试下面的代码

```
--已知一个 number 变量 n
n = 5    --这个数字是几都有可能
print("n 在判断前的值",n)


if 你的条件 then
```

```
    要做的事
else
    要做的事
end
print("n 在判断后的值",n)
```

5.3 判断三角形合法性（自测题）

你需要使用前面几章的知识，来完成下面的题目

已知三个 number 类型的变量 a、b、c，分别代表三根木棒的长度

请判断，使用这三根木棒，是否可以组成一个三角形（两短边之和大于第三边）

- 如果可以组成，就打印出 true
- 如果不可以，就打印 false

```
a,b,c = 1,2,3--多改几个数值自行测试

--补全代码
```

5.4 if 的判断依据（自测题）

我们在前面了解到，Lua 把 false 和 nil 看作是 false，其他的都为 true（包括 0 这个值，也是相当于 true）

那么问题来了，执行下面的代码，将会输出什么？

点我快速测试下面的代码

```
result = ''
if 0 then
    result = result..'T,'
else
    result = result..'F,'
end
if a then
```

```
    result = result..'T'
else
    result = result..'F'
end
print(result)
```

---

6 函数

6.1 初识函数
6.2 local 变量
6.3 函数参数
6.4 函数返回值
6.5 判断三角形合法性 2（自测题）
6.6 返回多个值（自测题）

6.1 初识函数

函数是指一段在一起的、可以做某一件事儿的程序，也叫做子程序。

在前面的内容中，我们已经接触过了函数的调用，这个函数就是前面用到了很多次的 print(...)。

调用函数只需要按下面的格式即可：

函数名(参数 1,参数 2,参数 3,.....)

为何要使用函数？因为很多事情都是重复性操作，我们使用函数，可以快速完成这些操作

下面我们举一个最简单的函数例子，这个函数没有传入参数、没有返回值

它实现了一个简单的功能，就是输出 Hello world!：

点我快速测试下面的代码

```
function hello()
    print('Hello world!')
end
```

这个函数名为 hello，我们可以按下面的方法进行调用（执行）：

点我快速测试下面的代码

```
function hello()
    print('Hello world!')
end
hello()
```

这行代码会输出 Hello world!。

同时，在 Lua 中，函数也是一种变量类型，也就是说，hello 实际上也是一个变量，里面存储的是一个函数，我们可以用下面的代码来理解：

点我快速测试下面的代码

```
function hello()
    print('Hello world!')
end
a = hello
--把 hello 函数同时赋值给 a 变量
a()
hello()
--a 和 hello 变量指向同一个函数
--所以执行结果和 hello()相同
```

因为函数只是个变量，你甚至在一开始可以这样声明 hello 函数：

点我快速测试下面的代码

```
hello = function()
    print('Hello world!')
end
hello()
```

下面你需要做一件简单的事情：

- 新建一个函数变量 biu，使其执行后会打印 biubiubiu 这个字符串，
- 新建一个函数变量 pong，使其与 biu 指向的函数相同

点我快速测试下面的代码

```
--请在此处补全代码

--请使下面的调用可以正常打印出 biubiubiu
biu()
pong()
print("biu 和 pong 相等吗？",biu==pong)
```

6.2 local 变量

之前我们创建的变量，都是全局变量，这种变量在代码运行周期从头到尾，都不会被销毁，而且随处都可调用。

但是当我们代码量增加，很多时候大量新建全局变量会导致内存激增，我们需要一种可以临时使用、并且可以自动销毁释放内存资源的变量，要怎么解决呢？

我们可以使用 `local` 标志来新建临时变量，使用 `local` 创建一个局部变量，与全局变量不同，局部变量只在被声明的那个代码块内有效。

参考下面的代码：

点我快速测试下面的代码

```
a = 123
function add()
    local n = a+2
    print(n)
end
add()
print(n)
```

上面的代码中，n 就是一个局部变量，它只在这个 funcion 中有效，并且函数运行完后会自动回收这部分的内存。

我们应该尽可能的使用局部变量，以方便 lua 虚拟机自动回收内存空间，同时减少资源占用提高运行速度。

下面请阅读以下代码，思考一下，正确的输出结果是什么：

点我快速测试下面的代码

```
str = 'abc'
function connect()
    local s = str..'def'
end
print(s,str)
```

6.3 函数参数

在前几章的使用中，我们知道函数是可以传入参数的，如 print(123)

那么，我们如何编写可以传入参数的函数呢？可以按下面的模板来写

```
function 函数名(参数 1,参数 2,...)
    代码内容
end
```

这里传入的参数，等价于在函数内部新建了一个 local 的变量，修改这些数据不会影响外部的数据（除了后面还没有讲到的 table 等类型）

举个例子，比如下面的函数，可以实现打印出两个传入值的和：

点我快速测试下面的代码

```
function add(a,b)
    print(a+b)
end
add(1,2)
--会输出 3
```

这段代码其实等价于：

点我快速测试下面的代码

```
function add()
    local a = 1
```

```
    local b = 2
    print(a+b)
end
add()
```

下面问题来了，请设计一个函数 p，可以按下面的调用方式来打印出物体的密度：

点我快速测试下面的代码

```
--补全这个函数的代码，满足题目要求
function p(a,b,c,m)

    print()
end

--一个长方体的长宽高分别为 a、b、c（单位米）
a = 1
b = 2
c = 3
--这个物体重量为 m（单位克）
m = 230
--下面打印出密度
--注：密度计算公式 密度 = 质量 / 体积
p(a,b,c,m)
```

6.4 函数返回值

在前面的代码中，我们实现了一个函数，输入变量 a、b，函数会自动输出两个数值的和。

但是一般来说，我们的需求远远不止这些，我们可能需要一个如下功能的函数：

执行函数，输入两个值，获取这两个值的和

如果还是按上面几节的内容，我们只会输出这个值，并不能把这个值传递给其他的变量进行后续使用，如何解决这个需求呢？

我们可以使用函数的返回值来实现这个需求，结合上面的需求，我们可以用下面的代码实现：

点我快速测试下面的代码

```
function add(a,b)
    return a+b
end
all = add(1,2)
--这里 all 的值就是 3 了
print(all)
```

这里的 return 表示返回一个值，并且立刻结束这个函数的运行

同时，和输入值可以有多个一样，返回值也可以有多个

点我快速测试下面的代码

```
function add(a,b)
    return a+b,"ok"
end
all, result = add(1,2)
--这里 all 的值就是 3 了
--这里 result 的值就是 string "ok"
print(all,result)
```

下面问题来了，请设计一个函数 p，可以按下面的调用方式来返回出物体的密度，返回值为 number 类型：

点我快速测试下面的代码

```
function p(a,b,c,m)
    --请补全代码
end

--一个长方体的长宽高分别为 a、b、c（单位米）
a = 1
b = 2
```

```
c = 3
--这个物体重量为 m（单位克）
m = 230
--下面返回密度值
--注：密度计算公式 密度 = 质量 / 体积
result = p(a,b,c,m)
print(result)
```

6.5 判断三角形合法性 2（自测题）

你需要使用前面几章的知识，来完成下面的题目

- 已知三个 number 类型的变量，分别代表三根木棒的长度
- 请判断，使用这三根木棒，是否可以组成一个三角形（两短边之和大于第三边）
- 请新建一个函数 triangle，并可以用如下形式调用（如果可以组成，就返回 true）：

点我快速测试下面的代码

```
function triangle(a,b,c)
    --请补全代码
end

result = triangle(1,2,3)--传入值为三边长度，多改几个测试下
print(result)
```

6.6 返回多个值（自测题）

你需要使用前面几章的知识，来完成下面的题目

- 已知 2 个 number 类型的变量，分别代表一个长方体的长和宽
- 请计算这个长方形的周长和面积
- 请新建一个函数 rectangle，并可以用如下形式调用：

点我快速测试下面的代码

```
function rectangle(a,b)
    --补全代码
```

```
end

area,len = rectangle(1,2)
--结果：
--面积 area 为 2
--周长 len 为 6
print(area,len)
```

---

7 Table

7.1 认识数组
7.2 简单 table
7.3 table 下标
7.4 下标进阶
7.5 table 小测验
7.6 table 小测验 2
7.7 Lua 全局变量与 table
7.8 table 小测试 3
7.9 table.concat
1.10 table 删减

7.1 认识数组

数组，使用一个变量名，存储一系列的值

很多语言中都有数组这个概念，在 Lua 中，我们可以使用 table（表）来实现这个功能

在 Lua 中，table 是一个一系列元素的集合，使用大括号进行表示，其中的元素之间以逗号分隔，类似下面的代码：

```
t = {1,3,8,5,4}
```

我们可以直接使用元素的下标，来访问、或者对该元素进行赋值操作。

在上面的 table 变量 t 中，第一个元素的下标是 1，第二个是 2，以此类推。

我们可以用变量名+中括号，中括号里加上下标，来访问或更改这个元素，如下面的例子：

点我快速测试下面的代码

```
t = {1,3,8,5,4}
print(t[1]) --打印 1
print(t[3]) --打印 8

t[2] = 99 --更改第二个元素的值
print(t[2]) --打印 99

t[6] = 2 --凭空新建第六个元素并赋值
print(t[6]) --打印 2

print(t[10])
--因为不存在，打印 nil
```

以上就是 table 最简单的一个例子了，就是当作数组来用（注意，一般语言中的数组基本都为不可变长度，这里的 table 为可变长度）

下面你需要完成：

- 新建一个 table，名为 cards，存入 1-10 十个数字
- 将第 3 个元素与第 7 个元素交换
- 将第 9 个元素与第 2 个元素交换
- 增加第 11 个变量，值为 23

```
--请补全代码
cards =
```

7.2 简单 table

上一节里，我们将 table 来表示数组，实际上，table 中可以包括任意类型的数据

比如我们可以在 table 中放置 number 和 string 数据，类似下面的代码：

```
t = {"abc",223,",..a",123123}
```

我们甚至能在里面放 function 变量

点我快速测试下面的代码

```
t = {
    function() return 123 end,
    function() print("abc") end,
    function(a,b) return a+b end,
    function() print("hello world") end,
}
t[1]()
t[2]()
t[3](1,2)
t[4]()
```

这些 table 访问每个元素的方式仍然是直接用下标，并且也能用下标来进行修改

下面你需要完成：

- 新建一个 table，名为 funcList，并实现以下功能
- 调用 funcList[1](a,b)，返回 a 和 b 的乘积
- 调用 funcList[2](a,b)，返回 a 减 b 的差
- 调用 funcList[3](a)，返回 a 的相反数（-a）

点我快速测试下面的代码

```
--请补全代码
funcList = {


}


a,b = 1,2--提供两个数
print("a,b 值为",a,b)
print("a 和 b 的乘积：",funcList[1](a,b))
print("a 和 b 的差：",funcList[2](a,b))
print("a 和相反数：",funcList[3](a))
```

7.3 table 下标

在前两节，我们的 table 都只是一些简单的 List（列表），每个元素的下标都是自动从 1 排列的

实际上，Lua 中，下标可以直接在声明时进行指定，像下面这样：

点我快速测试下面的代码

```
t = {6,7,8,9}
--上面和下面的代码等价
t = {
    [1] = 6,
    [2] = 7,
    [3] = 8,
    [4] = 9,
}

--甚至你可以跳过某些下标
t = {
    [1] = 6,
    [3] = 7,
    [5] = 8,
    [7] = 9,
}
print(t[7])
--输出 9

--在声明后赋予元素值也是可以的
t = {}--空的 table
t[101] = 10
print(t[101])
--输出 10
```

下面你需要：

- 新建一个变量 t，并按下面的格式声明
- 下标为 1 的元素，值为 123（number）
- 下标为 13 的元素，值为"abc"（string）
- 下标为 666 的元素，值为"666"（string）

点我快速测试下面的代码

```
--请补全代码
t = {

}

print("下标为 1 的元素：",t[1],type(t[1]))
print("下标为 13 的元素：",t[13],type(t[13]))
print("下标为 666 的元素：",t[666],type(t[666]))
```

7.4 下标进阶

在上一节，我们学习了如何自定义下标，其实在 Lua 中，下标也可以是字符串，如下面的例子

点我快速测试下面的代码

```
t = {
    ["apple"] = 10,
    banana = 12,
    pear = 6,
}
--使用["下标"] = 值
--和  下标 = 值
--都是正确写法
--当第二种方式有歧义时，应该用第一种方式

--可以用下面两种方式访问：
print(t["apple"])
--输出 10
print(t.apple)
```

```
--输出 10
--当第二种方式有歧义时，应该用第一种方式
```

可见，在使用 string 作为下标时，table 的灵活性提升了一个数量级。

string 作为下标时，也可以动态赋值：

点我快速测试下面的代码

```
t = {} -- 空 table
t["new"] = "新的值"
print(t.new)
--输出 新的值
```

下面你需要完成：

- 新建 table 变量 t
- 下标为 apple 的元素，值为 123（number）
- 下标为 banana 的元素，值为"abc"（string）
- 下标为 1@1 的元素，值为"666"（string）

点我快速测试下面的代码

```
--请补全代码
t = {

}

print("下标为 apple 的元素：",t["apple"],type(t["apple"]))
print("下标为 banana 的元素：",t["banana"],type(t["banana"]))
print("下标为 1@1 的元素：",t["1@1"],type(t["1@1"]))
```

7.5 table 小测验

下面的代码，将会打印什么？

点我快速测试下面的代码

```
t = {
    apple = {
```

```
        price = 7.52,
        weight = 2.1,
    },
    banana = {
        price = 8.31,
        weight = 1.4,
        year = '2018'
    },
    year = '2019'
}
print(
    t.price,
    t.apple.price,
    t.banana.weight,
    t.year
)
```

7.6 table 小测验 2

下面的代码，将会打印什么？

点我快速测试下面的代码

```
t = {
    {
        price = 7.52,
        weight = 2.1,
    },
    {
        price = 8.31,
        weight = 1.4,
        year = '2018'
    },
    year = '2019'
}
print(
```

```
    t["price"],
    t[1].price,
    t[2].weight,
    t["year"]
)
```

7.7 Lua 全局变量与 table

在前面我们知道了，在table中，可以直接用table名[下标]或table名.string下标来访问元素

实际上，在 Lua 中，所有的全局变量全部被存放在了一个大 table 中，这个 table 名为：_G

我们可以用下面的例子来示范：

点我快速测试下面的代码

```
n = 123--新建变量
print(n)--输出 123
print(_G.n)--输出 123

_G.abc = 1--相当于新建全局变量
print(abc)--输出 1

_G["def"] = 23--相当于新建全局变量
print(def)--输出 23

--甚至你可以像下面这样
_G.print("hello")
_G["print"]("world")
```

现在，你明白为什么说万物基于 table 了吧？

你需要完成下面的任务：

- 已知有一个全局变量，名为@#$
- 请新建一个变量 result
- 将@#$变量里的值赋值给 result

点我快速测试下面的代码

```
_G["@#$"] = 123


result = --请补全代码
print("result 值为",result)
```

7.8 table 小测试 3

请新建一个名为 t 的 table，满足以下要求

- t[10]可获得 number 类型数据 100
- t.num 可获得 number 类型数据 12
- t.abc[3]可获得 string 类型数据 abcd
- t.a.b.c 可获得 number 类型数据 789

点我快速测试下面的代码

```
--请补全代码
t = {


}


print("t[10]可获得 number 类型数据 100:",t[10],type(t[10]))
print("t.num 可获得 number 类型数据 12:",t.num,type(t.num))
print("t.abc[3]可获得 string 类型数据
abcd:",t.abc[3],type(t.abc[3]))
print("t.a.b.c 可获得 number 类型数据 789:",t.a.b.c,type(t.a.b.c))
```

7.9 table.concat

table.concat (table [, sep [, i [, j]]])

将元素是 string 或者 number 类型的 table，每个元素连接起来变成字符串并返回。

可选参数 sep，表示连接间隔符，默认为空。

i 和 j 表示元素起始和结束的下标。

下面是例子：

点我快速测试下面的代码

```
local a = {1, 3, 5, "hello" }
print(table.concat(a))
print(table.concat(a, "|"))

-->打印的结果：
--135hello
--1|3|5|hello
```

请完成下面的任务：

- 已知 table 变量 t，
- 将 t 中的结果全部连起来
- 间隔符使用，
- 并使用 print 打印出来

点我快速测试下面的代码

```
t = {"a","b","c","d"}
print("连接结果：")
--补全代码
```

7.10 table 删减

table.insert (table, [pos ,] value)

在（数组型）表 table 的 pos 索引位置插入 value，其它元素向后移动到空的地方。pos 的默认值是表的长度加一，即默认是插在表的最后。

table.remove (table [, pos])

在表 table 中删除索引为 pos（pos 只能是 number 型）的元素，并返回这个被

删除的元素，它后面所有元素的索引值都会减一。pos 的默认值是表的长度，即默认是删除表的最后一个元素。

下面是例子：

点我快速测试下面的代码

```
local a = {1, 8}              --a[1] = 1,a[2] = 8
table.insert(a, 1, 3)   --在表索引为 1 处插入 3
print(a[1], a[2], a[3])
table.insert(a, 10)    --在表的最后插入 10
print(a[1], a[2], a[3], a[4])

-->打印的结果：
--3    1    8
--3    1    8    10


local a = { 1, 2, 3, 4}
print(table.remove(a, 1)) --删除速索引为 1 的元素
print(a[1], a[2], a[3], a[4])

print(table.remove(a))   --删除最后一个元素
print(a[1], a[2], a[3], a[4])

-->打印的结果：
--1
--2    3    4    nil
--4
--2    3    nil    nil
```

请完成下面的任务：

- 已知 table 变量 t，
- 去除 t 中的第一个元素
- 然后这时，在 t 的第三个元素前，加上一个 number 变量，值为 810

点我快速测试下面的代码

```
t = {1,2,3,4,5,6,7,8,9}

--补全代码

print("第一个元素应为 2：",t[1])
print("第三个元素应为 810：",t[3])
```

---

8 循环

8.1 while 循环
8.2 for 循环
8.3 中断循环
8.4 循环测试题 1（自测题）
8.5 循环测试题 2（自测题）
8.6 循环测试题 3（自测题）
8.7 循环测试题 4（自测题）

8.1 while 循环

在实际功能实现中，经常会遇到需要循环运行的代码，比如从 1 到 100 填充 table 数据，我们可以直接用循环语句来实现

我们首先来学习 while 这个循环语法，整体的格式如下：

```
while 继续循环判断依据 do
    执行的代码
end
```

下面举一个例子，我们计算从 1 加到 100 的结果：

点我快速测试下面的代码

```
local result = 0
local num = 1

while num <= 100 do
```

```
    result = result + num
    num = num + 1
end

print(result)
```

上面的代码，就是当 num≤100 时，result 不断地加 num，并且 num 每次循环后自己加 1

理解了上面的代码，我们来完成下面一个简单的任务吧：

- 已知两个 number 类型的变量 min 和 max
- 请计算从 min 与 max 之间，所有 3 的倍数的和
- 打印出结果

点我快速测试下面的代码

```
min,max = 114,514 --这个结果应为 42009
result = 0--结果存放到这个变量

while 请完善 do
    --补全代码
end

print("结果：",result)
```

8.2 for 循环

for 循环在某些程度上，和 while 循环很相似，但是 for 循环可以更加简洁地表达中间累积的量

我们首先来学习 for 这个循环语法，整体的格式如下：

```
for 临时变量名=开始值,结束值,步长 do
    循环的代码
end
```

其中，步长可以省略，默认为 1

临时变量名可以直接在代码区域使用（但不可更改），每次循环会自动加步长值，并且在到达结束值后停止循环

下面举一个例子，我们计算从 1 加到 100 的结果：

点我快速测试下面的代码

```
local result = 0

for i=1,100 do
    result = result + i
end

print(result)
```

上面的代码，就是当 i≤100 时，result 不断地加 i，并且 i 每次循环后增加 1

理解了上面的代码，我们来完成下面一个简单的任务吧：

- 已知两个 number 类型的变量 min 和 max
- 请计算从 min 与 max 之间，所有 7 的倍数的和
- 打印出结果

点我快速测试下面的代码

```
min,max = 114,514 --这个结果应为 17955
result = 0--结果存放到这个变量

for --补全代码


print("结果：",result)
```

8.3 中断循环

前面我们学习了循环语句，有些时候循环运行到一半，我们不想再继续运行了，

怎么办呢？

我们可以在一个循环体中使用 break，来立即结束本次循环，继续运行下面的代码

比如像下面这样，计算 1-100 相加途中，小于 100 的最大的和：

点我快速测试下面的代码

```
result = 0
for i=1,100 do
    result = result + i
    if result > 100 then
        result = result - i
        break
    end
end

print(result)
```

可以看见，当发现和大于 100 后，代码立即把 result 的值还原到了加上当前数字之前的状态，并且调用 break 语句，立即退出了本次循环

在 while 中，我们也可以使用 break:

点我快速测试下面的代码

```
result = 0
c = 1
while true do
    result = result + c
    if result > 100 then
        result = result - c
        break
    end
    c = c + 1
end
```

```
print(result)
```

我们在这里直接使用了死循环（因为 while 的继续运行判断依据始终为 true），整体逻辑也和之前 for 的代码一致，当发现和大于 100 后，代码立即把 result 的值还原到了加上当前数字之前的状态，并且调用 break 语句，立即退出了本次循环

现在你需要完成一项任务：

- 请求出小于变量 max 的 13 的倍数的最大值（max 大于 0）
- 并将结果打印出来
- 本题理论上不用循环就能实现，但是为了练习一下技巧，请用 for 循环来实现

点我快速测试下面的代码

```
max = 810 --结果应为 806
result = 0

for --请补全代码

print(result)
```

8.4 循环测试题 1（自测题）

前面我们学习了循环语句，我们需要完成下面的任务

我们知道，print 函数可以打印一行完整的输出

那么，已知变量 a，请打印出下面的结果：

（a 为大于 0 的整数，且需要输出 a 行数据，数据从 1 开始，每行与上一行的差为 2）

```
1
3
5
7
```

```
9
```

（上面例子为当 a 为 5 的情况）

做题区域：

点我快速测试下面的代码

```
a = 10
--需要用 print 输出要求的结果
print("输出结果：")


for --请补全代码
```

8.5 循环测试题 2（自测题）

我们需要完成下面的任务

那么，已知变量 a，请打印出下面的结果：

（a 为大于 0 的整数，且需要输出 a 行数据，第一行为一个*，后面每行多一个*）

```
*
**
***
****
*****
```

（上面例子为当 a 为 5 的情况）

做题区域：

点我快速测试下面的代码

```
a = 10
--需要用 print 输出要求的结果
print("输出结果：")
```

```
for --请补全代码
```

8.6 循环测试题 3（自测题）

我们需要完成下面的任务

那么，已知变量 a，请打印出下面的结果：

（a 为大于 0 的整数，且需要输出 a 行数据，按图示规律输出）

```
1
12
123
1234
12345
123456
1234567
12345678
123456789
12345678910
1234567891011
```

（上面例子为当 a 为 11 的情况）

做题区域：

点我快速测试下面的代码

```
a = 20
--需要用 print 输出要求的结果
print("输出结果：")


for --请补全代码
```

8.7 循环测试题 4（自测题）

- 有一只猴子，第一天摘了若干个桃子 ，当即吃了一半，但还觉得不过瘾 ，就又多吃了一个。
- 第 2 天早上又将剩下的桃子吃掉一半，还是觉得不过瘾，就又多吃了两个。
- 以后每天早上都吃了前一天剩下的一半加天数个（例如，第 5 天吃了前一天剩下的一半加 5 个）。
- 到第 n 天早上再想吃的时候，就只剩下一个桃子了。
- 那么，已知变量 a 为最后一天的天数，请打印出第一天的桃子数。
- 如：a 为 5 时，输出 114

做题区域：

点我快速测试下面的代码

```
a = 6
--需要用 print 输出要求的结果
print("输出结果：")


for --请补全代码
```

---

9 详解 string 库

string.sub
string.rep
string.len
大小写转换
string.format
string 的本质
string.char
string.byte
string.find
string.gsub

string.sub

接下来几节会讲解 string 库的各种接口

```
string.sub(s, i [, j])
```

返回字符串 s 中，从索引 i 到索引 j 之间的子字符串。

i 可以为负数，表示倒数第几个字符。

当 j 缺省时，默认为 -1，也就是字符串 s 的最后位置。

当索引 i 在字符串 s 的位置在索引 j 的后面时，将返回一个空字符串。

下面是例子：

点我快速测试下面的代码

```
print(string.sub("Hello Lua", 4, 7))
print(string.sub("Hello Lua", 2))
print(string.sub("Hello Lua", 2, 1))
print(string.sub("Hello Lua", -3, -1))

-->打印的结果：
lo L
ello Lua

Lua
```

值得注意的是，我们可以使用冒号来简化语法，像下面这样：

点我快速测试下面的代码

```
s = "12345"
s1 = string.sub(s, 4, 7)
s2 = s:sub(4, 7)
--两种写法是等价关系
print(s1,s2)
```

请完成下面的任务：

- 已知字符串变量 s，请分别打印出（每种一行）：
- s 从第 4 个字符开始，到最后的值
- s 从第 1 个字符开始，到倒数第 3 个字符的值
- s 从倒数第 5 个字符开始，到倒数第 2 个字符的值

点我快速测试下面的代码

```
s = "1919810"

--补全代码
print()
print()
print()
```

---

string.rep

string.rep(s, n)

返回字符串 s 的 n 次拷贝。

示例代码：

点我快速测试下面的代码

```
print(string.rep("abc", 3))
--输出结果：
--abcabcabc
```

请完成下面的任务：

打印一行数据，数据内容为 810 个 114514

点我快速测试下面的代码

```
--补全代码
print()
```

---

string.len

string.len(s)

接收一个字符串，返回它的长度。

示例代码：

点我快速测试下面的代码

```
s = "hello lua"
print(string.len(s))
--输出结果：
9


--同时也可以使用简便语法
print(s:len())
```

请完成下面的任务：

- 新建一个变量 s，使数据内容为 810 个 114514
- 并打印出字符串 s 的长度

点我快速测试下面的代码

```
s = --补全代码
print()
```

大小写转换

string.lower(s)

接收一个字符串 s，返回一个把所有大写字母变成小写字母的字符串。

string.upper(s)

接收一个字符串 s，返回一个把所有小写字母变成大写字母的字符串。

示例代码：

点我快速测试下面的代码

```
s = "hello lua"
print(string.upper(s))
print(string.lower(s))
--输出结果：
HELLO LUA
hello lua
```

```
--同时也可以使用简便语法
print(s:upper())
print(s:lower())
```

请完成下面的任务：

已知一个变量 s，打印出全是大写字母的 s 字符串

点我快速测试下面的代码

```
s = "asd8938KJjsidiajdl;(()k)"
print --补全代码
```

---

string.format

string.format(formatstring, ...)

按照格式化参数 formatstring，返回后面...内容的格式化版本。

编写格式化字符串的规则与标准 c 语言中 printf 函数的规则基本相同：

它由常规文本和指示组成，这些指示控制了每个参数应放到格式化结果的什么位置，及如何放入它们。

一个指示由字符%加上一个字母组成，这些字母指定了如何格式化参数，例如 d 用于十进制数、x 用于十六进制数、o 用于八进制数、f 用于浮点数、s 用于字符串等。

示例代码：

点我快速测试下面的代码

```
print(string.format("%.4f", 3.1415926))     -- 保留 4 位小数
print(string.format("%d %x %o", 31, 31, 31))-- 十进制数 31 转换成不同进制
d,m,y = 29,7,2015
print(string.format("%s %02d/%02d/%d", "today is:", d, m, y))
```

```
--控制输出 2 位数字，并在前面补 0


-->输出
-- 3.1416
-- 31 1f 37
-- today is: 29/07/2015
```

请完成下面的任务：

- 已知一个变量 n，为 number 类型整数
- 打印出 n:连上 n 值的字符串

点我快速测试下面的代码

```
n = 810


print --补全代码
```

---

string 的本质

这一节我们来讲解字符串的本质

字符串，是用来存储一串字符的，但是它的本质就是一串数字。如何用一串数字来代表一串字符呢？

在计算机中，每一个符号都对应着一个数字，但是在讲解这个知识之前，我们了解一下补充知识：

在大多数编程语言中，我们使用 0x 开头来表示这个数字是 16 进制的。
比如
10 等价于 0x0a
256 等价于 0xff

接下来，你需要了解，每一个符号都对应着一个数字，比如：

0 对应着 0x30、1 对应着 0x31 a 对应着 0x61、b 对应着 0x62 A 对应着 0x41、B 对应着 0x42

上面的编码规则，我们称之为 ascii 码，具体想了解可以打开下面的网址查看：

http://ascii.911cha.com/

当然，1 字节最大为 0xff，即 256，只能存下一部分符号，大部分的中文按某些编码，一个中文占用 2 或 3 个字节

计算机如何解析这些数据，我们不需要了解，当你知道了上面的知识后，你应该可以理解下面的描述：

字符串"apple"实际上的内容就是下面的一串数字：
0x61,0x70,0x70,0x6c,0x65

同时，lua 的字符串中可以保存任何数值，即使是 0x00 这种不代表任何含义的数，也可以保存

补充：在其他语言中（如 C），0x00 代表字符串结束，但是在 lua 中并不是这样。lua 的字符串每字节可以存储任意的一字节数据。

比如下面的描述：

有一串 lua 字符串中的数据为：
0x01,0x02,0x30,0x00,0x44
实际人能看到的（不可见字符用�代替）：
��0�D
当然，你不能说你看不见的数据就不存在，他们都完好无损地在这个字符串中

下面你需要思考一个问题：一串字符串数据如下，它的实际内容是什么（指人能看见的字符串内容，如 abcd）？

0x62,0x61,0x6e,0x61,0x6e,0x61

---

string.char

string.char (...)

接收 0 个或更多的整数（整数范围：0~255），返回这些整数所对应的 ASCII 码字符组成的字符串。当参数为空时，默认是一个 0。

如果上一章节有认真学习过了的话，这段话应该是很好理解的。实质上就是把计算机认识的一串数字，变成字符串变量，并且字符串内的数据就是要存的那串数据。

示例代码：

点我快速测试下面的代码

```
str1 = string.char(0x30,0x31,0x32,0x33)
str2 = string.char(0x01,0x02,0x30,0x03,0x44)
print(str1)
print(str2)

-->输出（不可见字符用�代替）
--0123
--??0?D
```

请完成下面的任务：

- 已知一个字符串的每个字符在数组 t 中按顺序排列
- 请根据 t 的值，打印出字符串内容（一行数据）
- 注：这个字符串存储的不一定是可见的字符

点我快速测试下面的代码

```
t =
{0x79,0x6F,0x75,0x20,0x61,0x72,0x65,0x20,0x72,0x69,0x67,0x68,0x
74}
print("真正的字符串内容：")
--补全代码
```

---

string.byte

string.byte(s [, i [, j]])

返回字符 s[i]、s[i + 1]、s[i + 2]、……、s[j] 所对应的 ASCII 码。i 的默认值为 1，

即第一个字节,j 的默认值为 i 。

这个函数功能刚好和前面的 string.char 相反，是提取字符串中实际的数值。

示例代码：

点我快速测试下面的代码

```
str = "12345"
print(string.byte(str,2))
print(str:byte(2))--也可以这样
print(str:byte())--不填默认是 1

-->输出（十进制数据）
--50
--50
--49
```

请完成下面的任务：

- 已知字符串 s
- 请把 s 中代表的数据，全部相加，并打印出来

点我快速测试下面的代码

```
s = string.char(1,2,3,4,5,6,7,8,9)
print("s 内数据的和是：")
--补全代码
```

---

string.find

string.find(s, p [, init [, plain]])

这个函数会在字符串 s 中，寻找匹配 p 字符串的数据。如果成功找到，那么会返回 p 字符串在 s 字符串中出现的开始位置和结束位置；如果没找到，那么就返回 nil。

第三个参数 init 默认为 1，表示从第几个字符开始匹配，当 init 为负数时，表

示从 s 字符串的倒数第-init 个字符处开始匹配。

第四个参数 plain 默认为 false，当其为 true 时，只会把 p 看成一个字符串对待。

可能你会奇怪，第四个参数有什么存在的必要吗？p 不是本来就应该是个字符串吗？ 实际上，lua 中的匹配默认意义是正则匹配，同时，这里的正则与其它语言也有些许不同。

由于篇幅有限，本节和下面的几节涉及匹配内容时，均不会考虑正则的使用方法，Lua 正则教程将会在最后几节单独详细地列出来。

第四个参数为 true 时，便不会使用正则功能。

示例代码：

点我快速测试下面的代码

```
--只会匹配到第一个
print(string.find("abc abc", "ab"))
-- 从索引为 2 的位置开始匹配字符串：ab
print(string.find("abc abc", "ab", 2))
-- 从索引为 5 的位置开始匹配字符串：ab
print(string.find("abc abc", "ab", -3))

-->输出
--1  2
--5  6
--5  6
```

请完成下面的任务：

- 已知字符串 s，里面有很多相同的字符串
- 请找出字符串 s 中，所有字符串 awsl 的位置
- 使用 print 打印结果，结果一行一个
- 如字符串 12awslawslaw，输出 3 和 7

点我快速测试下面的代码

```
s = "12awsaslwlaawsllslllswasllalssawwlawslaw"
print("两个 awsl 的位置分别是：")
--补全代码
```

---

string.gsub

string.gsub(s, p, r [, n])

将目标字符串 s 中所有的子串 p 替换成字符串 r。

可选参数 n，表示限制替换次数。

返回值有两个，第一个是被替换后的字符串，第二个是替换了多少次。

特别提示：这个函数的目标字符串 s，也是支持正则的

下面是例子：

点我快速测试下面的代码

```
print(string.gsub("Lua Lua Lua", "Lua", "hello"))
print(string.gsub("Lua Lua Lua", "Lua", "hello", 2)) --指明第四个参数

-->打印的结果：
-- hello hello hello   3
-- hello hello Lua     2
```

同样的，我们也可以使用冒号来简化语法，像下面这样：

点我快速测试下面的代码

```
s = "12345"
r = s:gsub("2","b")
print(r)
```

请完成下面的任务：

- 已知字符串变量 s，请分别打印出（每种一行）：

- 把字符串 s 中，前 5 个 a，替换为 b
- 把字符串 s 中，前 3 个 c，替换为 xxx
- 把结果打印出来，一行数据

点我快速测试下面的代码

```
s = "asdicagydausckfugdaflgscdabgsdbahhacbshbsd"
print("s 变换前的值：",s)
--补全代码
```

跨文件调用

在编写代码时，随着逻辑逐渐复杂，我们的代码量也会变大。虽然有函数可以把一部分代码逻辑封装起来，但是所有代码都放到一个文件里，显然也不是个好办法。

所以我们需要将一些代码放到不同文件中，通过文件来区分这些代码的功能。

比如我们有下面这个函数：

```
---函数功能：
-- 生成从 1-max 的 table
-- @输入值：table 的最大值
-- @返回：  table 结果
-- @例子：  local list = getNumberList(10)
function getNumberList(max)
    local t = {}
    for i=1,max do
        table.insert(t,i)
    end
    return t
end
```

我们新建一个文件叫 tools.lua，把这个函数放进去，现在，整个文件如下面这样：

tools.lua

```lua
---函数功能：
-- 生成从 1-max 的 table
-- @输入值：table 的最大值
-- @返回：  table 结果
-- @例子：  local list = getNumberList(10)
local function getNumberList(max)
    local t = {}
    for i=1,max do
        table.insert(t,i)
    end
    return t
end

--手动返回一个 table，包含了上面的函数
return {
    getNumberList = getNumberList,
}
```

现在，我们封装的这个函数就能在其他文件里被调用了，具体代码如下：

```lua
--引用 tools.lua 文件，并加载
local tool = require("tools")

local list = tool.getNumberList(12)
```

当调用了 require 接口后，Lua 虚拟机会自动加载你调用的文件，执行文件的内容，然后返回你文件里 return 的结果。

为了更好地理解这段话，我们可以看下面两个文件，其中 run.lua 是被运行的那个入口文件

test.lua

```
--以便一会儿返回使用的 table
local temp = {}

--把全局变量 a 更改了
a = 1

--local 变量无法被外部调用
--但是可以在文件内被调用
local b = 2

--文件在被 require 的时候，会被执行
--把全局变量 c 更改了
c = a + b

--使函数在 table 里
function temp.addB()
    --文件内部可以调用变量 b
    b = b + 1
    return b
end

--返回 table
return temp
```

run.lua

点我快速测试下面的代码

```
local test = require("test")
print(a)--输出 1
print(b)--输出 nil，因为 b 是 local 变量
print(c)--输出 3

print(test.addB())--输出 3
print(test.addB())--输出 4
```

```
print(test.addB())--输出 5
```

同时，每个文件最多只会被 require 一次，如果有多个 require，只有第一次会执行

十七 AT 命令手册

1 AT 命令概述

1.1 文档目的

本手册详细介绍了合宙Luat LTE模块做支持的 AT 命令集。

1.2 惯例和术语缩写

本手册中模块可以被称为如下术语：

1）ME (移动设备)，它可包括移动终端（MT），终端适配器（TA）

2）MS (移动台)，它包括移动设备（ME）和用户识别模块（SIM）

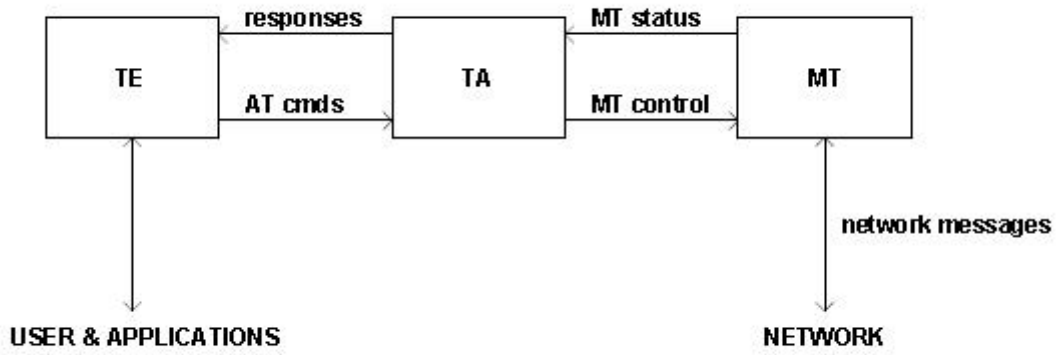
3）TA (终端设配器)

4）DCE (数据通信设备) 或者传真 DCE (传真调制解调器，传真板)

在应用中，控制器通过串口发送 AT 命令来控制 GSM 模块。串口线对端的控制器可以被称为如下术语：

1） TE (终端设备) 或

2） DTE (数据终端设备) 或者简单地说，一个嵌入式应用



其他术语缩写：

AT	ATtention; this two-character abbreviation is always used to start a command line to be sent fromTE to TA
BCD	Binary Coded Decimal
DCE	Data Circuit terminating Equipment
DTE	Data Terminal Equipment
IMEI	International Mobile station Equipment Identity
ICCID	Integrate circuit card identity
IRA	International Reference Alphabet (ITU-T T.50)
ME	Mobile Equipment
MT	Mobile Termination
SIM	Subscriber Identity Module
TA	Terminal Adaptor, e.g. a GSM data card (equal to DCE)
TE	Terminal Equipment, e.g. a computer (equal to DTE)
URC	Unsolicited Reslut Code
NTP	Network Time Protocol
NITZ	Network Identity and Time Zone
MO	Mobile Originated

1.3 AT 命令语法

本手册中所有命令行必须以"AT"或"at"作为开头，以回车（<CR>）作为结尾。响应通常紧随命令之后，它的样式是"<回车><新行><响应内容><回车><新行>"(<CR><LF><响应内容><CR><LF>) 。整本手册里，只有<响应内容>被自始至终介绍，而<回车><新行>被有意省略了。

合宙无线模块提供的AT 命令包含符合GSM07.05、GSM07.07 和ITU-T Recommendation V.25ter 的命令.

所有AT 命令从语法上可以分为三类："基础类"，"S 参数类"以及"扩展类"，分别介绍如下：

1.3.1 基础类 AT 命令

这类AT 命令具有"AT<x><n>"或者"AT&<x><n>"格式，其中"<x>"是命令；"<n>"可以是一个或者多个参数。例如："ATE<n>"，该命令用于为DCE 开关回显功能，即DCE 会依据"<n>"值决定是否把接收的字符回显给DTE。"<n>"是选配参数，若没有赋值，模块将使用默认值。

1.3.2 S 参数类 AT 命令

这类AT 命令格式为"ATS<n>=<m>"，其中"<n>"是S 寄存器的索引；"<m>"是赋予的值。"<m>" 是选配参数，若没有赋值，模块将使用默认值。

1.3.3 扩展类 AT 命令

一般来讲，扩展类命令依据功能可以分为以下几种类型：

命令类型	语法	说明
测试命令	AT+<X>=?	该命令用于查询设置命令或内部程序设置的参数以及其取值范围
查询命令	AT+<X>?	该命令用于返回参数的当前值
设置命令	AT+<X>=<…>	该命令用于设置用户自定义的参数值
执行命令	AT+<X>	该命令用于读取受GSM 模块内部程序控制的不可变参数

扩展类命令语法：

在命令行 TA 返回结果中<>内为必配参数，[]内为选配参数；

在每条命令中，选配参数和必配参数必须按照规定的顺序排列，各参数间必须用逗号隔开；

字符串参数一般放在双引号中；

实际使用中，<>，[]不必输入。

一行内同时输入多条 AT 命令的语法：

数条 AT 命令可以被放在同一命令行中输入。这样则不需要在每条命令的开头输入"AT"或 "at"，仅需要在命令行的开头输入"AT"或"at"即可。请注意：在扩展类的命令后面，应增加一个分号作为定界符，基础类命令以及S参数命令则不用输入分号。

例如：

命令（→）/返回（←）	实例	解释和说明

→	ATE0&W+CREG=2;&W	设置回显关闭并保存，设置网络注册状态URC上报模式为2并保存
←	OK	设置成功
→	AT+CFUN=1;+CIMI;+VER	
←	460060500266902 AirM2M_780E_V1121_LTE_AT OK	

1.4 AT 命令最大响应时间

以下表格中列举出的AT，不管设置命令、查询命令抑或测试命令，最大响应时间都如表格所示，表格没列举出来的AT，不管测试命令、查询命令抑或设置命令，最大响应时间都是9秒。

AT 命令	最大响应时间（单位：秒）
COPS	300
CGACT	108
CGATT	108
CFUN	45
CMGS	90
CMGW	90
CPBW	72
CPIN	180
CSTT	60
CIICR	90
CIPSHUT	90

2 基本命令

2.1 查询制造商名称：AT+CGMI

语法规则：

命令类型	语法	返回

执行命令	AT+CGMI	<manufacturer> OK

参数定义：

参数	定义	取值	对取值的说明
<manufacturer>	生产厂商 ID		取值由模块厂商定义

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CGMI	查询模块的生产厂商的ID
←	+CGMI: "AirM2M" OK	返回查询结果

2.2 查询模块型号：AT+CGMM

语法规则：

命令类型	语法	返回
执行命令	AT+CGMM	<model> OK

参数定义：

参数	定义	取值	对取值的说明
<model>	模块的型号		由厂商而定

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CGMM	查询模块型号
←	+CGMM: "Air780E" OK	返回查询结果

2.3 查询模块版本信息：AT+CGMR

语法规则：

命令类型	语法	返回
执行命令	AT+CGMR	Revision: <revision> OK

测试命令	AT+CGMR=?	OK

参数定义：

参数	定义	取值	对取值的说明
<revision>	软件版本标识		由厂商定义

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CGMR	查询软件版本标识
←	+CGMR: "AirM2M_780E_V1120_LTE_AT" OK	返回查询结果

2.4 查询 IMEI 号：AT+CGSN

语法规则：

命令类型	语法	返回
执行命令	AT+CGSN	<IMEI> OK
测试命令	AT+CGSN=?	OK

参数定义：

参数	定义	取值	对取值的说明
<IMEI>	产品序列号，即国际移动设备识别码 IMEI (International Mobile Equipment Identification)	-	15 位数字组成

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CGSN	查询产品IMEI
←	359759002514931 OK	返回查询结果

2.5 查询 SIM 卡 ICCID 号码：AT+CCID(/ICCID)

语法规则：

命令类型	语法	返回
执行命令	AT+CCID	<iccid>

		OK
	AT+ICCID	+ICCID:<iccid> OK

参数定义：

参数	定义	取值	对取值的说明
<ICCID>	产品 ICCID 号 (Integrated circuit card identity，集成电路卡识别码)		一般由 20 位数字组成

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CCID	查询SIM卡ICCID号
←	89860117831003134201 OK	返回查询结果
→	AT+ICCID	查询SIM卡ICCID号
←	+ICCID: 89860117831003134201 OK	返回查询结果

2.6 查询 IMSI：AT+CIMI

语法规则：

命令类型	语法	返回和说明
执行命令	AT+CIMI	<IMSI> OK

参数定义：

参数	定义	取值	对取值的说明
<IMSI>	国际移动台用户识别码（International Mobile Subscriber Identity）		由 15 位数字组成

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CIMI	查询IMSI号
←	460001841426414	返回查询结果

	OK	

2.7 查询产品信息：ATI

语法规则：

命令类型	语法	返回
执行命令	ATI	<module info> OK

参数定义：

参数	定义	取值	对取值的说明
<module info>	模块相关信息（厂商，版本）	-	由厂商定义

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	ATI	查询模块信息
←	AirM2M_780E_V1120_LTE_AT OK	返回查询结果

2.8 查询模块 FIRMWARE 版本：AT+VER

语法规则：

命令类型	语法	返回
执行命令	AT+VER	<firmware ver> OK
注意事项	返回结果与ATI一致	

参数定义：

参数	定义	取值	对取值的说明
< firmware ver>	模块内部软件版本	-	由厂商定义

举例：

命令（→）/ 返回（←）	实例	解释和说明

→	AT+VER	查询模块信息
←	AirM2M_780E_V1120_LTE_AT OK	返回查询结果

2.9 查询平台硬件版本：AT^HVER

语法规则：

命令类型	语法	返回
执行命令	AT^HVER	^HVER:<hardversion> OK

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT^HVER	查询模块平台硬件信息
←	^HVER: EC618 OK	返回查询结果

2.10 查询各种信息：AT*I

语法规则：

命令类型	语法	返回
设置命令	AT*I	Manufacturer:<manufacturer> Model: <model> Revision: <revision> HWVer: <hwver> Buildtime: <Buildtime> IMEI: <imei> ICCID:<iccid> IMSI:<imsi> OK

参数定义：

参数	Definition	取值	解释

<manufacturer>	+CGMI 命令的返回		
<model>	+CGMM 命令的返回		
<revision>	+CGMR 命令的返回		
<hwver>	硬件版本		
<Buildtime>	版本固件的编译时间		
<imei>	同+CGSN 命令的返回值		
<iccid>	同+ICCID 命令的返回值		
<imsi>	国际移动台用户识别码（International Mobile Subscriber Identity）		由 15 位数字组成

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT*I	
←	Manufacturer: AirM2M Model: Air780E Revision: AirM2M_780E_V1120_LTE_AT HWver: A12 Buildtime: Mar 6 2023 20:04:22 IMEI: 864040060365518 ICCID: 89860621260002571487 IMSI: 460060086257148 OK	

2.11 写 SN 号命令：AT+WISN

AT+WISN 是用来写用户 sn(serial number)到模块里的。

语法规则：

命令类型	语法	返回
设置命令	AT+WISN=<user_sn>	OK
		+CME ERROR: <err>
查询命令	AT+WISN?	<user_sn> OK

参数定义：

参数	定义	取值	对取值的说明
<user_sn>	special SN	-	由用户定义，最长 64 位

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT+WISN="1234567890"	写客户 SN 号
←	OK	
→	AT+WISN?	查询客户 SN 号
←	1234567890 OK	

2.12 重启模块：AT+RESET

重启模块。

语法规则：

命令类型	语法	返回
执行命令	AT+RESET	OK

2.13 重启模块并恢复出厂设置：AT+RSTSET

重启模块并恢复出厂设置。

语法规则：

命令类型	语法	返回
执行命令	AT+RSTSET	OK

2.14 查询开机原因值：AT*EXINFO?

查询开机原因值指令。EC618 系列模块>=1126 软件版本支持。

语法规则：

命令类型	语法	返回
查询命令	AT*EXINFO?	*EXINFO: <Power-on reason> OK

参数定义：

参数	定义	取值	对取值的说明
Power-on reason	开机原因	0	powerkey 或者上电开机
		1	充电或者 AT 指令下载完成后开机

		2	闹钟开机
		3	软件重启
		4	未知原因
		5	RESET 键重启
		6	异常重启
		7	工具控制重启
		8	内部看门狗重启
		9	外部重启
		10	充电开机

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT*EXINFO?	查询开机原因值
←	*EXINFO：0 OK	powerkey 开机

2.15 关闭 EPAT 导出的 PCAP 包的大小限制：AT+CATCHPCAP

关闭 EPAT 导出的 PCAP 包的大小限制指令。EC618 系列模块>=1128 软件版本支持。

注意：关闭 PCAP 包的大小限制，会导致丢失其他 log(除了网络的其他 log)，断电不保存。

语法规则：

命令类型	语法	返回
查询命令	AT+CATCHPCAP?	+CATCHPCAP: <limited> OK
设置命令	AT+CATCHPCAP=<limited>	OK

参数定义：

参数	定义	取值	对取值的说明
limited	是否限制 PCAP 包大小	0	打开限制（默认值为 0）
		1	关闭限制

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT+CATCHPCAP=1	关闭 PCAP 包的大小限制
←	OK	

3 配置命令

3.1 选择 TE 字符集：AT+CSCS

设置命令通知 DCE，DTE 需要使用的字符集，以确保 DCE 和 DTE 在约定的字符集之间准确转换字符串。

语法规则：

命令类型	语法	返回
设置命令	AT+CSCS=<chset>	OK
查询命令	AT+CSCS?	+CSCS: <chset> OK
测试命令	AT+CSCS=?	+CSCS: (<chset>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<chset>	字符集	"GSM"	GSM 7 位默认字符 (3GPP TS 23.038)
		"IRA"	国际参考符号集(ITU-T T.50)
		"UCS2"	16 位通用多字节编码字符集

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CSCS?	
←	+CSCS: "GSM" OK	
→	AT+CSCS=?	查询字符集范围
←	+CSCS:"GSM","IRA","UCS2" OK	

3.2 保存用户当前的配置：AT&W

本文档中大部分的 AT 命令，若希望模块重启后其参数不变，都需要通过 AT&W 保存。该命令将用

户设置的一些 AT 命令参数保存到 NV 中，重开机后相关的配置自动加载并生效。

语法规则：

命令类型	语法	返回和说明
执行命令	AT&W	OK

3.3 设置命令回显模式：ATE

执行命令设置 TA 在命令模式下，是否可以回显从 TE 接收到的字符。

语法规则：

命令类型	语法	返回
执行命令	ATE<value>	OK

参数定义：

参数	定义	取值	对取值的说明
<value>	是否回显	0	回显模式关闭
		1	回显模式打开

3.4 恢复所有参数为出厂配置：AT&F

执行命令把所有参数恢复为出厂配置。

语法规则：

命令类型	语法	返回
执行命令	AT&F	OK

3.5 设置结果码抑制模式：ATQ

此命令用于设置 TA 是否向 TE 发送结果码。并且此设置不会影响响应内的内容。

语法规则：

命令类型	语法	返回和说明
执行命令	ATQ<n>	OK
参数存储模式	设置命令的参数可以通过AT&W命令保存到NVM中，重启不丢失	

参数定义：

参数	定义	取值	对取值的说明
<n>	是否显示结果码	0	向TE发送结果码

		1	结果码被抑制，不发送

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	ATQ1	设置为不返回结果码
←		此时没出现OK这个结果码
→	AT+CREG?	
←	+CREG: 0,1	此时没出现OK这个结果码

3.6 FOTA 空中升级：AT+UPGRADE

FOTA（Firmware Over The Air）是空中升级固件的功能。过程按顺序如下：

1 跟服务器固件版本比较版本号大小；

2 如果服务器固件版本号大，则启动下载固件的过程；

3 下载固件完毕，模块自动重启，烧录新版本；

4 模块再次自动重启，运行新固件

语法规则：

命令类型	语法	返回	说明
设置命令	AT+UPGRADE=<tag>,<value>[,<savetoNV>]	OK	设置<tag>值
	AT+UPGRADE=<tag>	+UPGRADE:<tag>,<value> OK	查询<tag>值
查询命令	AT+UPGRADE?	+UPGRADE: <state> OK	<state>=0 时的返回
		+UPGRADE: <state>,<percent> OK	<state>=1 时的返回
		+UPGRADE: <state>, [<error code>, <http respone>, <fota error cause>] OK	<state>=2 时的返回
执行命令	AT+UPGRADE	OK	手动触发升级

参数定义：

参数	定义	取值	对取值的说明
<tag>		"KEY"	使用"KEY"来设置服务器的 ProductKEY
		"AUTO"	使用"AUTO"设置是否自动升级
		"PERIOD"	使用"PERIOD"设置自动升级的查询周期，单位：秒。当设置的周期到时，模块向服务器查询版本，服务器的模块版本号比自己大时，会发起自动升级。
		"URL"	使用"URL"设置自己的升级服务器的 URL 地址
<savetoNV>	是否把参数保存到 NV	0	保存，缺省值
		1	不保存
<value>	<tag>的取值，不同的<tag>有不同的取值		字符串型，<tag> = "KEY"时，设定此值为合宙 iot 服务器某个产品的 ProductKey 即可在此产品下进行 OTA 升级
		0	<tag> ="AUTO"时的取值，整数型。 0：关闭自动 FOTA（模块不再和服务器通讯） 1：打开自动 FOTA（模块定期向服务器查询） **注：缺省为 1，即自动升级为打开！**
		1	
		60~2^32(4294967296)	<tag> ="PERIOD"时的取值，整数型，单位为秒。缺省值为 86400，即 24 小时
		"http://xxxx.bin"	<tag> = "URL"时的取值，字符串型
<state>		0	未执行 FOTA
		1	正在从服务器下载固件
		2	固件下载结果
<percent>	升级百分比	1~100	当模块处于<state>=1 的状态时，下载固件的百分比
<error code>		-1000	无效的 URL
		-1001	网络错误
		-1002	服务器链接错误
		-1003	无效固件
		-1004	http 应答错误
		-1005	存储固件错误
		-1006	其他错误
<http respone>	http 错误码		详细取值请参考：https://baike.baidu.com/item/HTTP%E7%8A%B6%E6%8
<fota error cause>	FOTA 错误原因	3	无效的设备。找不到设备
		17	无权限
		25	无效的项目
		26	无效的固件
		27	已经是最新版本。通过了前面的检查，但版本号已

			是最新，或没打开升级开关、没指定 imei 等

举例：

命令（→）/ 返回（←）	实例	解释和说明
使用合宙提供的私人服务器进行升级：		
	1 将新版本固件上传服务器端 在通过合宙 ERP 购买模块后，ERP 会自动分配一个升级服务器管理账号，登陆地址为： http://iot.openluat.com/ 登录名是购买者手机号码，登录密码缺省是：888888 登录后**新建项目**，项目名为"**Air7XXE 标准模块**"如下所示： 点击设备列表将会看到已购买模块的 IMEI 列表，自动 FOTA 可以对这些设备进行固件升级。 点击进入**固件列表->创建固件->上传文件**，上传待更新的固件 bin。 固件升级用的 bin 文件在合宙官网发布的 AT 命令固件包内。 以合宙 AirM2M_780E_V1121_LTE_AT 测试固件为例，该升级包解压后会有三个文件，其中 bin 文件需要上传到服务器端。 上传文件完成后，将下图中的两个选项选择：**是**	

完成上述设置以后，模块就能实现自动空中固件升级 FOTA 了。缺省的查询服务器的时间间隔是 24 小时。

注：用户同样可以通过 AT+UPGRADE="AUTO",0 关闭自动更新。

如果客户想再自动升级，有两种方法：

1）输入 AT+UPGRADE 手动触发升级

2）输入 AT+UPGRADE="AUTO",1 重新打开自动升级

2 在服务器端可以查看 FOTA 升级情况

点 固件升级->升级统计后，可以看到每个版本的升级成功数量

3 模块在整个升级过程中的表现

开始下载固件。此时自动上报：

+UPGRADEIND: <percent>

percent: 0-100

	下载完固件后自动重启，开始烧录固件。烧录的时候主动上报： +UPGRADEDL: <percent> percent: 0-100 烧录成功后再次自动重启并主动上报软件已经更新： +UPGRADEVER: <new version> (new version： 新固件版本号，例如： "AirM2M_780E_V1121_LTE_AT ") +UPGRADEVER: <upload state> (upload state：新固件更新状态是否成功上报，1：上报成功，0：上报失败) **注意：从下载固件（打印+UPGRADEIND: <percent>），到模块自动重启后烧录新固件（打印+UPGRADDL: <percent>），再到再次重启后上报升级情况（打印+UPGRADEVER: <new version>），整个过程模块自动完成，禁止用户在这个过程中重启模块或给模块断电！**
使用用户自己的服务器进行升级：	
	用户在自己服务器放好 bin 文件，然后输入： AT+UPGRADE="URL","http://xxxx.bin" OK 同样的，缺省也是自动升级，间隔周期也是 24 小时。模块从服务器下载固件，是用 HTTP 协议。

串口升级：AT+UARTUPGRADE

EC618 系列模块>=1155 版本支持（目前只有标准 AT 和 LPAT 固件支持这个串口指令升级）

语法规则：

命令类型	语法	返回	说明
设置命令	AT+UARTUPGRADE=<cmd>[,<sn>,<nbytes>,<data>,<xor8sum>	响应 OK 如果发生错误，响应： ERROR OR +CME ERROR: <err>	最大响应时间：10S 参数保存模式：不保存
测试命令	AT+UARTUPGRADE=?	响应 +UARTUPGRADE: 列举所支持的 <cmd> OK 如果发生错误，响应： ERROR OR +CME ERROR: <err>	

参数定义：

参数	定义	取值	对取值的说明
<cmd>		0	擦除 FLASH 中 FOTA 下载区
		1	分段下载差分文件，且分段是按顺序依次下载 参数 <sn>,<nbytes>,<data>,<xor8sum>仅在此指令下生效
		2	验证差分文件 校验会返回的状态码 XOTA_PS_PKG_UNFOUND 未写入或已开始写入差分包 XOTA_PS_PKG_UNVERIFIED 差分包写入完成但还未完成校验 XOTA_PS_PKG_VERIFIED 校验差分包成功 XOTA_PS_PKG_VERIFAILED 校验失败 XOTA_PS_PKG_UPGRADED 升级完成/升级结束 XOTA_PS_PKG_UNDEF XOTA_PS_PKG_PARTIAL 其他错误 +UARTUPGRAE ERROR: <err_code> 1 参数错误 2 指令不支持

			3 文件大小不正确 4 sn 分段编号不是以 0 开头或分段编号不连续 5 crc 校验失败 6 没有擦除 FLASH 下载区 7 擦除 Flash 下载区域失败 8 文件写入失败 9 文件读取失败 10 区域中没有找到文件（文件写入失败）
		3	查询差分文件名称
		4	查询差分文件版本号
		5	升级差分文件
<sn>	差分文件分段编号	0- 65535	起始编号为 0，后续分段依次加 1
<nbytes>	分段数据<data>的总字节数	4-1024	为 4 字节的整数倍；如果不是 末尾补 0，直到是 4 的整数倍
<data>	差分文件分段数据	8-2048	以 16 进制可显示字符表示的差分文件分段数据，数据长度为<nbytes>的 2 倍，差分文件通过 FotaToolkit 工具生成
<xor8sum>	差分文件分段数据的 XOR8 运算值（BCC 校验）		以 16 进制可显示字符表示的差分文件分段数据的 XOR8 运算值（BCC 校验），在线计算网站：https://www.lddgo.net/encrypt/bcc

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT+UARTUPGRADE=0	擦除FLASH中FOTA下载区
←	OK	
→	AT+UARTUPGRADE=1,0,1024,ECDF20400000000807B17000013D739771489369C1B8C99236063348853D2C51279344F6641D8134592F8A474FFFF

	FF001D8B19A38FED67D 70F8A14EE85BA47C718 3CD845C4DFB31E3E06E 0001445433631385F415 02E70617200000000000 04250484449000000543 324004E6F724F4B00000 04B0000004B0000004B 0000004B000000630000 004B0000004B0000004 B0000004F0000004B000 0004B0000004B000000 4B0000004B0000004B0 000004B0000004B0000 004B0000004B0000004 B0000004B0000004B00 00004B0000004B00000 04B0000004B0000004B 0000004B0000004B000 0004B0000004B000000 4B0000004B0000004B0 000004B0000004B0000 004B0000004B0000004 B0000004B0000004B00 00004B0000004B00000 04B0000004B0000004B 0000004B0000004B000 0004B0000004B000000 4B0000004B0000004B0 000004B0000004B0000 004B0000004B0000004 B0000004B0000004B00 00004B0000004B00000 04B0000004B0000004B 0000004B0000004B000 0004B0000004B000000 4B0000004B0000004F00 00004253444946463430 010000004B0000000080 0000425A683131415926 5359C831E0EA0000C44 0C0C800000440000008 200030CD00549A9EA7B 00EA901AD206E01F177	

	245385090C831E0EA04 25344494646343001000 0004B000000008000004 25A6831314159265359 C831E0EA0000C440C0C 80000044000000820003 0CD00549A9EA7B00EA9 01AD206E01F17724538 5090C831E0EA0425344 4946463430010000004B 00000000800000425A68 31314159265359C831E0 EA0000C440C0C800000 440000008200030CD00 549A9EA7B00EA901AD2 06E01F177245385090C8 31E0EA04253444946463 430010000004B0000000 0800000425A683131415 9265359C831E0EA0000 C440C0C800000440000 008200030CD00549A9E A7B00EA901AD206E01F 177245385090C831E0E A0425344494646343001 0000004B000000008000 00425A68313141592653 59C831E0EA0000C440C 0C800000440000008200 030CD00549A9EA7B00E A901AD206E01F177245 385090C831E0EA04253 44494646343001000000 63000000800000425A 6831314159265359B288 9DFC0000C5D964FE010 00140000008400002000 040000C200050A60000 2A4A68069B53DD31522 45D20C6E206B3B40335 9BAFB5BDDDE24057E2E E48A70A12165113BF804 25344494646343001000 0004B000000008000004	

	25A6831314159265359 C831E0EA0000C440C0C 8000004400000,c3	
←	OK	
	…	中间省略写入其它分段差分数据的步骤
→	AT+UARTUPGRADE=2	验证差分文件
←	+UARTUPGRADE: 6, XOTA_PS_PKG_VERIFIED OK	通知接收端差分文件验证成功
→	AT+UARTUPGRADE=5	升级差分文件
←	OK	
升级后大概2分钟左右重新，重启后课通过AT*I查看版本信息，确认模块是否升级成功		

3.7 设置指令行终止符：ATS3

设置指令，可设置用于 AT 指令行终止符，该字符能被 TA 识别。

语法规则：

命令类型	语法	返回
设置命令	ATS3=<n>	OK
查询命令	ATS3?	<n> OK

参数定义：

参数	定义	取值	对取值的说明
<n>	指令行终止符的 ASCII 值	13	缺省值：13，对应 ASCII字符为<CR>(回车符) 注：仅支持这个值

3.8 设置命令行编辑字符：ATS5

此命令设置删除命令行先前字符的字符。

语法规则：

命令类型	语法	返回
设置命令	ATS5=<n>	OK
查询命令	ATS5?	<n> OK

参数定义：

参数	定义	取值	对取值的说明
<n>	指令行编辑符的 ASCII 值	8	缺省值：8(对应ASCII字符<BS>后退符)

举例：

命令（→） / 返回（←）	实例	解释和说明
→	ATS5?	查询当前指令行编辑字符
←	8 OK	当前指令行编辑字符为BackSpace 后退符

3.9 设置 CDC 功能模式：AT&C

此命令设置 109（DCD）电路状态与远端接收线上信号检测之间的关系。

语法规则：

命令类型	语法	返回
执行命令	AT&C<value>	OK

参数定义：

参数	定义	取值	对取值的说明
<value>	DCD 电路状态	0	DCD 线始终为ON
		1	DCD线只在数据载波存在时为ON

3.10 设置 DTR 功能模式：AT&D

此命令设置数据模式下，circuit 108/2 (DTR) 从 ON 变为 OFF 状态时， TA 的响应状态。

语法规则：

命令类型	语法	返回
执行命令	AT&D<value>	OK

参数定义：

参数	定义	取值	对取值的说明
<value>	DTR 电路状态	0	TA 忽略DTR 状态
		1	DTR 由ON至OFF：TA在保持当前数据通话的同时，切换至命令模式

		2	DTR 由ON 至OFF：TA 释放数据通话，切换至命令模式。 DTR= OFF 时，为自动应答关闭状态。

3.11 实时时钟：AT+CCLK

语法规则：

命令类型	语法	返回
设置命令	AT+CCLK=<time>	OK
查询命令	AT+CCLK?	+CCLK: <time> OK
测试命令	AT+CCLK=?	OK

参数定义：

参数	定义	取值	对取值的说明
<time>	时间	-	字符串类型(字符串需包含在引号内) ；格式类似"yy/MM/dd,hh:mm:ss±zz"，最后两位字符指示年，月，日，小时，分钟，秒和时区(用当地时间和GMT 时间之间的差别来表示，以1/4 小时格式来表示；范围-47...+48)

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT+CCLK?	查询当前时间
←	+CCLK: "18/08/01,12:12:58+00" OK	返回的查询结果
→	AT+CCLK="18/08/07,13:28:29+32"	将当前时间设置为正确的时间，设置时区可以用"+"号
←	OK	
→	AT+CCLK?	再查询一次当前时间
←	+CCLK: "18/08/07,13:28:31+32" OK	返回的查询结果

3.12 设置 USB 模式：AT+SETUSB

本命令可以设置 USB 口的使用模式。

语法规则：

命令类型	语法	返回

设置命令	AT+SETUSB=<mode>[,<vid>,<pid>]	OK
查询命令	AT+SETUSB?	mode: <mode> Vid: <vid> Pid: <pid> OK
测试命令	AT+SETUSB=?	OK
注意事项	该命令可以关机自动保存	

参数定义：

参数	定义	取值	对取值的说明
<mode>	USB 的使用模式，设置命令的必选参数	1	RNDIS+AT+PPP+DIAG模式，缺省值
		2	ECM+ AT+PPP+DIAG模式
		3	保留
<vid>	Vendor id，设置命令的可选参数		Air780E系列缺省值为0x19d1
<pid>	Product id，设置命令的可选参数		Air780E系列缺省值为0x0001

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+SETUSB=2	设置成ECM网卡
←	OK	
→	AT+SETUSB?	
←	mode: 2 vid: 0x19d1 pid: 0x1 OK	

3.13 设备错误：AT+CMEE

该命令使能或者屏蔽结果码的使用+CME ERROR: <err>，作为和 ME 功能错误相关的指示。

语法规则：

命令类型	语法	返回
设置命令	AT+CMEE=[<n>]	OK
查询命令	AT+CMEE?	+CMEE :<n> OK
测试命令	AT+CMEE=?	+CMEE:(<n>取值列表)

		OK

参数定义：

参数	定义	取值	对取值的说明
<n>	结果码上报模式	0	禁用结果码+ CME ERROR: <err>，使用 ERROR
		1	启用结果码+ CME ERROR: <err>，并使用数字型的<err>取值
		2	启用结果码+ CME ERROR: <err>，并使用冗长方式的<err>取值

3.14 错误码描述：+CME ERROR:<err>

数字型 <err> 取值	冗长方式的<err>取值	解释
常见错误		
0	phone failure	手机故障
1	no connection to phone	未连接到手机
2	phone-adaptor link reserved	预留手机适配器链路
3	operation not allowed	不允许操作
4	operation not supported	不支持操作
5	PH-SIM PIN required	需要PH-SIM卡的PIN
6	PH-FSIM PIN required	需要PH-FSIM的PIN
7	PH-FSIM PUK required	需要PH-FSIM的PUK
10	SIM not inserted	没有插入SIM卡
11	SIM PIN required	需要SIM卡的PIN
12	SIM PUK required	需要SIM卡的PUK
13	SIM failure	SIM卡故障
14	SIM busy	SIM卡遇忙
15	SIM wrong	SIM错误
16	incorrect password	密码无效
17	SIM PIN2 required	需要SIM卡的PIN2
18	SIM PUK2 required	需要SIM卡的PUK2
20	memory full	存储已满
21	invalid index	索引无效
22	not found	未发现
23	memory failure	存储故障
24	text string too long	文本字符串过长
25	invalid characters in text string	文本字符串中的字符无效
26	dial string too long	拨号字符串过长
27	invalid characters in dial string	拨号字符串中的字符无效

30	no network service	无网络业务
31	network timeout	网络超时
32	network not allowed - emergency calls only	网络不允许－只适用于紧急呼叫
40	network personalization PIN required	需要网络个性化PIN
41	network personalization PUK required	需要网络个性化PUK
42	network subset personalization PIN required	需要网络子集个性化PIN
43	network subset personalization PUK required	需要网络子集个性化PUK
44	service provider personalization PIN required	需要服务供应商个性化PIN
45	service provider personalization PUK required	需要服务供应商个性化PUK
46	corporate personalization PIN required	需要公司个性化PIN
47	corporate personalization PUK required	需要公司个性化PUK
48	hidden key required	需要输入隐藏的密码
49	EXE_NOT_SURPORT	
50	EXE_FAIL	（适用于cat1 模块）
50	Invalid Param	无效参数（适用于cat4 模块）
51	NO MEMORY	内存不足（适用于cat1 模块）
52	OPTION NOT SURPORT	选项不支持（适用于cat1 模块）
53	parameters are invalid	无效参数（适用于cat1 模块）
54	EXT_REG_NOT_EXIT	（适用于 cat1 模块）
55	EXT_SMS_NOT_EXIT	（适用于 cat1 模块）
56	EXT_PBK_NOT_EXIT	（适用于 cat1 模块）
57	EXT_FFS_NOT_EXIT	（适用于 cat1 模块）
58	INVALID_COMMAND_LINE	（适用于 cat1 模块）
59	ITF_DIFFERENT	（适用于 cat1 模块）
60	BURN_FLASH_FAIL	（适用于 cat1 模块）
61	TFLASH NOT EXIST	TF 卡不存在（适用于 cat1 模块）
62	FILE NOT EXIST	文件不存在（适用于 cat1 模块）
63	FILE TOO LARGE	文件太大（适用于 cat1 模块）
96	INVALID DATE OR TIME	无效日期或时间（适用于 cat1 模块）
97	DIR CREATE FAIL	创建文件夹失败（适用于 cat1 模块）
98	DIR NOT EXIST	文件夹不存在（适用于 cat1 模块）
99	NOT IMPLEMENTED	不可执行（适用于 cat1 模块）
100	unknown	未知
103	Illegal MS	非法MS
106	Illegal ME	非法ME
107	GPRS services not allowed	不允许GPRS业务
111	PLMN not allowed	不允许PLMN
112	Location area not allowed	不允许位置区
113	Roaming not allowed in this location area	该位置区不允许漫游
132	service option not supported	不支持业务选择

133	requested service option not subscribed	未描述业务选择请求
134	service option temporarily out of order	业务选择暂时无连接
148	unspecified GPRS error	GPRS错误未指明
149	PDP authentication failure	PDP 鉴权失败
150	invalid mobile class	移动类别无效
151	AT command timeout	AT命令超时
181	UNSUPPORTED QCI VALUE	不支持CQI
214	SS_UNKNOWN_SUBSCRIBER	
222	SS_ILLEGAL_SUBSCRIBER	
223	SS_BRERSERV_NOT_PROV	
224	SS_TELESERV_NOT_PROV	
225	SS_ILLEGAL_EQUIPMENT	
226	SS_CALL_BARRED	
229	SS_ILLEGAL_OPERATION	
230	SS_ERROR_STATUS	
231	SS_NOT_AVAILABLE	
232	SS_SUBS_VIOLATION	
233	SS_INCOMPATIBILITY	
234	SS_FACILITY_NOT_SUPPORTED	
240	SS_ABSENT_SUBSCRIBER	
247	SS_SYSTEM_FAILURE	
248	SS_DATA_MISSING	
249	SS_UNEXPECTED_DATA_VALUE	
250	SS_PWD_REGISTRATION_FAILURE	
251	SS_NEGATIVE_PWD_CHECK	
256	SS_NUMOF_PWD_ATTEMPT_VIOL	
264	SIM VERIFY FAIL	（适用于cat1 模块）
265	SIM UNBLOCK FAIL	（适用于cat1 模块）
266	SIM CONDITION NO FULLFILLED	（适用于cat1 模块）
267	SS_POSITION_METHOD_FAILURE	（适用于cat4 模块）
267	SIM UNBLOCK FAIL NO LEFT	（适用于cat1 模块）
268	SIM VERIFY FAIL NO LEFT	（适用于cat1 模块）
269	SIM INVALID PARAMETER	（适用于cat1 模块）
270	SIM UNKNOW COMMAND	（适用于cat1 模块）
271	SIM WRONG CLASS	（适用于cat1 模块）
272	SIM TECHNICAL PROBLEM	（适用于cat1 模块）
273	SIM CHV NEED UNBLOCK	（适用于cat1 模块）
274	SIM NOEF SELECTED	（适用于cat1 模块）
275	SIM FILE UNMATCH COMMAND	（适用于cat1 模块）
276	SIM CONTRADICTION CHV	（适用于cat1 模块）
277	SIM CONTRADICTION INVALIDATION	（适用于cat1 模块）
278	SIM MAXVALUE REACHED	（适用于cat1 模块）

279	SIM PATTERN NOT FOUND	（适用于cat1 模块）
280	SIM FILEID NOT FOUND	（适用于cat1 模块）
281	SIM STK BUSY	（适用于cat1 模块）
282	SIM UNKNOW	（适用于cat1 模块）
283	SIM PROFILE ERROR	（适用于cat1 模块）
284	SS_UNKNOWN_ALPHABET	
285	SS_USSD_BUSY	
323		Cat1模块
339	SS_MAXMPTY_CALLS_EXCEEDED	
340	SS_RESOURCES_NOT_AVAILABLE	
501	WIFI labtool reture error	
502	BT labtool reture error	
503	FM labtool reture error	
504	MRD file already exist	
505	MRD file with same version already exist	
506	MRD file with newer version already exist	
507	MRD authorization failure	
508	(U)SIM PUK blocked	
509	Vendor not supported	
510	NVM path not exist	
511	NVM file comcfg error	
535	PROTOCOL stack busy	
600	BTSAP card not accessible	
601	BTSAP card powered off	
602	BTSAP card removed	
603	BTSAP card powered on	
604	BTSAP data not available	
605	BTSAP not supported	
606	Non-Production mode	
753	missing required cmd parameter	CRSM 缺少参数
754	Invalid SIM command	CRSM 无效命令
755	Invalid file id	CRSM 无效的文件
756	Missing required P1/2/3 parameter	CRSM 缺少P 参数
757	Invalid P1/2/3 parameter	CRSM 无效的P 参数
758	Missing required command data	CRSM 缺少命令数据
759	invalid characters in command data	CRSM 命令行中有无效字符
765	Invalid input value	无效输入值
766	Unsupported mode	不支持的模式
767	Operation failed	操作失败
768	Mux already running	多路复用已经在运行
769	Unable to get control	不能获得控制权
770	SIM network reject	SIM 网络拒绝

771	Call setup in progress	正在建立呼叫
772	SIM powered down	SIM 关闭了
773	SIM file not present	SIM 文件不在
774	RAC refresh net time failure	
791	Param count not enough	
792	Param count beyond	
793	Param value range beyond	
794	Param type not match	
795	Param format invalid	
796	Get a null param	
797	CFUN state is 0 or 4	
810	No Error	
811	Unrecognized Command	
812	Return Value Error	
813	Syntax Error	
814	Unspecified Error	
815	Data Transfer Already	
816	Action Already	
817	Not At Cmd	
818	Multi Cmd too long	
819	Abort Cops	
820	No Call Disc	
821	BT SAP Undefined	
822	BT SAP Not Accessible	
823	BT SAP Card Removed	
824	AT Not Allowed By Customer	
890	GPS_NOT_RUNNING	
891	GPS_IS_RUNNING	
892	GPS_IS_FIXING	
893	GPS_IS_SLEEPING	
894	GPS_NOT_SLEEPING	
900	DIAED_REJECT	
901	PDP_NO_ACTIVE	
902	PDP_ACTIVE	
910	TCP_CONNECTION_REJECT	
911	TCP_CONNECT_OVERTIME	
912	SOCKET_CONNECTION_EXIST	
913	SOCKET_CONNECTION_NOT_EXIST	
914	BUFFER_OVER_SIZE	
915	SENDING_OVERTIME	
916	DNS_EXIST	
917	DNS_PARSE_OVERTIME	
918	DNS_PARSE_ERROR	
980	INPUT_VALUE_ERROR	
981	OTHER_ERROR	

982	ERROR	
983	NOT_ALLOWED	
1000	UPGRADE_INVALID_URL	
1001	UPGRADE_NET_ERROR	
1002	UPGRADE_SERVER_CONNECT_ERROR	
1003	UPGRADE_INVALID_FILE	
1004	UPGRADE_SERVER_RESPONSE_ERROR	
1005	UPGRADE_WRITE_FLASH_ERROR	
1006	UPGRADE_ERROR	
65535	Other Error	

4 设备控制命令

4.1 模块功能模式：AT+CFUN

设置命令选择模块的功能模式。

语法规则：

命令类型	语法	返回
设置命令	AT+CFUN=<fun>[,<rst>]	OK
查询命令	AT+CFUN?	+CFUN: <fun> OK
测试命令	AT+CFUN=?	+CFUN: (<fun>取值列表),(<rst>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<fun>	功能模式	0	最少功能
		1	全部功能
		4	飞行模式。关闭模块的RF电路的接收和发送功能
<rst>	是否需要重启	0	在设置<fun>级别时，不复位ME
		1	在设置<fun>级别时，复位ME

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT+CFUN=?	查询各个参数的取值范围
←	+CFUN: (0,1,4),(0,1) OK	查询结果
→	AT+CFUN?	查询当前的功能模式
←	+CFUN: 1 OK	查询结果
→	AT+CFUN=1,1	被用来主动重启模块，重启后进入全功能模式
←	OK	

4.2 关机：AT+CPOWD

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CPOWD=<n>	<n>=0 ，紧急关机，没有任何返回 <n>=1 ，正常关机，返回：NORMAL POWER DOWN

4.3 输入 PIN 码：AT+CPIN

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CPIN=<pin>[,<newpin>]	OK 说明：如果需要的PIN 是SIM PUK 或者SIM PUK2，则需要第二个pin。<new pin>用来取代 SIM 卡中的原有的pin。
查询命令	AT+CPIN?	+CPIN: <code> OK
测试命令	AT+CPIN=?	OK
URC	+CPIN:<code>	

参数定义：

参数	定义	取值	对取值的说明
<pin>	密码	-	字符串型
<newpin>	新密码	-	字符串型
<code>		READY	ME不再需要提供密码
		SIM PIN	ME等待提供SIM卡的PIN码
		SIM PUK	ME等待提供SIM卡的PUK码
		SIM PIN2	ME等待提供SIM卡的PIN2码
		SIM PUK2	ME等待提供SIM卡的PUK2码
		SIM REMOVED	SIM卡未检出

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT+CPIN?	查询PIN码锁状态
←	+CPIN: READY OK	表示PIN码锁并未开启

→	AT+CLCK="SC",1,"1234"	开启开机PIN码锁，1234是PIN码，SC表示是SIM卡
←	OK	返回OK后，重启模块
←	+CPIN: SIM PIN	重新开机后，模块会自动上报PIN码状态，SIM PIN表示开机PIN码为ON的状态（即开机需要输入PIN码）
→	AT+CPIN="1234"	此时需要输入PIN码
←	+CPIN: READY OK	表示密码正确，PIN码锁解锁
→	AT+CLCK="SC",2	查询当前的开机PIN码是否仍然开启
←	+CLCK: 1 OK	1表示仍然有开机PIN码提示
→	AT+CLCK="SC",0,"1234"	关闭开机PIN码提示
←	OK	返回OK后重新开机
←	+CPIN: READY	重新开机后，模块会自动上报PIN码状态，READY表示开机PIN码：OFF

4.4 设备锁定：AT+CLCK

执行命令锁定、解锁、查询 ME 或网络设备<fac>。一般需要输入密码。

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CLCK=<fac>,<mode>[,<password>[,<class>]]	当<mode>≠2，返回： OK
		当<mode>=2，返回： +CLCK:<status>[,<class1><CR><LF> +CLCK:<status>,<class2>[...]] OK
测试命令	AT+CLCK=?	+CLCK:(<fac>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<fac>	设备	"CS"	Control surface（锁控制接口）
		"PS"	将设备锁定到某个或某些SIM卡上
		"PF"	将设备锁定到第一个插入的SIM卡上
		"SC"	SIM/UICC（锁SIM卡）

		"AO"	禁止所有出局呼叫
		"OI"	禁止所有国际出局呼叫
		"OX"	禁止所有国际出局呼叫，归属国除外
		"AI"	禁止所有入局呼叫
		"IR"	归属国以外漫游时，禁止所有入局呼叫
		"NT"	禁止TA未保存的号码入呼叫
		"NM"	禁止MT未保存的号码入呼叫
		"NS"	禁止 SIM/UICC 未保存的号码入呼叫
		"NA"	禁止在哪个存储空间都未保存的号码呼入
		"AB"	禁止所有服务，仅当 mode=0 有效
		"AG"	禁止所有出局服务，仅当 mode=0 有效
		"AC"	禁止所有入局服务，仅当 mode=0 有效
		"FD"	SIM 卡固定拨号
		"PN"	网络个性化(请参考GSM 02.22)
		"PP"	服务供应商个性化(请参考GSM 02.22)
		"PU"	网络子集个性化(请参考GSM 02.22)
		"PC"	企业个性化(请参考GSM 02.22)
<mode>	工作模式	0	解锁
		1	锁定
		2	查询状态
<status>	激活状态	0	非激活
		1	激活
<classx>	业务类型	1	语音(电话业务)
		2	数据(所有承载业务；当<mode>=2时，如果TA不支持16、32、64、128 的取值，该参数仅表示部分承载业务)
		4	传真(传真业务)
		8	短消息服务
		16	数据电路同步
		32	数据电路异步
		64	专用分组访问
		128	PAD专用接入
<password>	密码		字符型；与ME用户接口使用的设备密码、修改密码命令+CPWD 设置的密码相同

4.5 修改密码：AT+CPWD

语法规则：

命令类型	语法	返回
设置命令	AT+CPWD=<fac>,<oldpwd>,<newpwd>	OK
测试命令	AT+CPWD=?	+CPWD: (<fac>,<pwdlength>)取值列表 OK

参数定义：

参数	定义	取值	对取值的说明
<oldpwd>,<newpwd>	老密码，新密码	-	字符型；与 ME用户接口使用的设备密码、修改密码命令+CPWD设置的密码相同
<pwdlength>	密码长度	-	整数型，设备支持的最大密码长度
<fac>	设备		与AT+CLCK中定义相同

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CLCK="SC",1,"1234"	如果要修改开机PIN码，必须先要打开开机PIN码锁
←	OK	
→	AT+CPWD="SC","1234","8888"	修改开机PIN码为8888
←	OK	此时重启模块
←	+CPIN: SIM PIN	重新开机后，模块会自动上报PIN码状态，SIM PIN表示开机PIN码：ON
→	AT+CPIN="8888"	此时需要输入PIN码
←	+CPIN: READY OK	表示密码正确，PIN码锁解锁

4.6 网络灯闪烁的时间间隔：AT+SLEDS

语法规则：

命令类型	语法	返回
设置命令	AT+SLEDS=<mode>,<timer_on>,<timer_off>	OK
查询命令	AT+SLEDS?	+SLEDS:<mode>, <timer_on>,<timer_off> OK
测试命令	AT+SLEDS=?	+SLEDS: (<mode>取值列表), (<timer_on>取值列表), (<timer_off>取值列表)

		OK

参数定义：

参数	定义	取值	对取值的说明
<mode>	设置何种状态下的闪灯间隔	1	设置没有注册上网络时的网络灯闪烁时间间隔
		2	设置已经注册上网络时的网络灯闪烁时间间隔
		3	设置PPP通讯状态时的网络灯闪烁时间间隔
<timer_on>	灯亮的时间	0或40~65535	单位ms ， 0为常灭
<timer_off>	灯灭的时间	0或40~65535	单位ms ， 0为常亮

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+SLEDS=?	测试命令
←	+SLEDS:(1-3),(40~65535),(40~65535) OK	返回结果
→	AT+SLEDS?	查询命令
←	+SLEDS:1,200,1800 +SLEDS:2,1800,200 +SLEDS:3,125,125 OK	此为缺省配置

4.7 设置 TE-TA 波特率：AT+IPR

语法规则：

命令类型	语法	返回和说明
设置命令	AT+IPR=<rate>	OK
查询命令	AT+IPR?	+IPR: <rate> OK
测试命令	AT+IPR=?	+IPR: (自适应波特率取值列表),(固定波特率取值列表) OK
注意事项	1. 缺省波特率=0，即自适应波特率。 2. 模块开机时需要MCU那边输入若干个"AT"或（注："aT","At"无法训练波特率），即可把模块的波特率训练成跟主控一致，此时会有初始化信息上报，表示训练成功。 3. 训练成功后，AT命令可以识别大写，小写或大小写混写。 4. "AT+IPR=x;&W"，可以设置固定波特率并保存。重启模块后，模块仍然是这个波特率x，如果上位机波	

	特率也是x，此时不用输入任何东西，模块就会自动上报开机初始化信息。 注：x是波特率，例如115200

参数定义：

参数	定义	取值	对取值的说明
<rate>	波特率，单位 bps	0	自适应波特率（自适应范围随模块型号不同而有所不同）
		600	
		1200	
		2400	
		4800	
		9600	
		19200	
		38400	
		57600	
		115200	
		230400	
		460800	
		921600	

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT+IPR=?	查询当前支持的波特率范围
	+IPR:(600,1200,2400,4800,9600,19200,38400,57600,115200,230400),(0,600,1200,2400,4800,9600,19200,38400,57600,115200,230400,460800,921600) OK	

4.8 设置 TE-TA 帧格式：AT+ICF

语法规则：

命令类型	语法	返回和说明
设置命令	AT+ICF=<format>,[<parity>]	OK
查询命令	AT+ICF?	+ICF: < format >[,<parity >] OK
测试命令	AT+ICF=?	+ICF: (<format>取值列表),(<parity>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<format >	帧格式	0	自动识别
		1	8 数据位0 奇偶校验位2 停止位
		2	8 数据位1 奇偶校验位1 停止位
		3	8 数据位0 奇偶校验位1 停止位
		4	7 数据位0 奇偶校验位2 停止位
		5	7 数据位1 奇偶校验位1 停止位
		6	7 数据位0 奇偶校验位1 停止位 注：0 奇偶校验位是指无奇偶校验位，此时<parity>被忽略
<parity>	校验位	0	奇数位（Odd）
		1	偶数位（Even）

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT+ICF=?	查询参数取值范围
←	+ICF: (1-6),(0-1) OK	

4.9 设置指令行换行字符：ATS4

使用该指令的设置命令，可设置用于结果码和信息文本换行的字符。

语法规则：

命令类型	语法	返回
设置命令	ATS4=<n>	OK
查询命令	ATS4?	<n> OK

参数定义：

参数	定义	取值	对取值的说明
<n>	指令行换行符的 ASCII 值	10	缺省值：10(对应ASCII字符<LF>换行符) 注：仅支持这个值

4.10 TE-TA 本地流量控制：AT+IFC

流控简介：

对 模块与 DTE 之间可靠的通信而言，流控是很重要的。假使一次数据或传真呼叫中，发送端的发送速度大于接收端的接收速度，当接收缓存接近满的时候，接收端应该有方法让发送端暂停发送，直到它重新赶上。这就是流控。

一般来说，有两种方法来实现流控。它们是软件流控和硬件流控。在多路复用模式下，推荐采用硬件流控。合宙模块对这两种流控方式都支持。

软件流控：

软件流控发送不同的字符去暂停(XOFF，十进制 19)和恢复(XON，十进制 17)数据流。它适用于仅仅有三根线的串口通信。

合宙LTE模块如果使用软件流控，则需要通过命令：**AT+IFC=1,1** 来配置

因为这个配置不保存，所以若希望重启后继续使用软件流控，需要通过 AT&W 保存到用户配置表中。

注：当模块传送的是二进制/十六进制数据时，不适合软件流控方式，因为TE可能会将二进制数据当成流控字符。

硬件流控：

硬件流控通过 RTS/CTS 信号来实现。当接收缓存快满的时候，模块把 CTS 信号设置为无效，数据传输被暂停。当模块的接收缓存可以接收更多数据时，CTS 信号重新被设置为有效。
要实现硬件流控功能，请确保你的应用串口包含 RTS/CTS 线。

语法规则：

命令类型	语法	返回和说明
设置命令	AT+IFC=<dce_by_dte>,[< dte_by_dce>]	OK
查询命令	AT+IFC?	+IFC: <dce_by_dte>,<dte_by_dce> OK
测试命令	AT+IFC=?	+IFC: (<dce_by_dte>取值列表),<dte_by_dce>取值列表) OK
注意事项	模块缺省无流控。如果需要硬件流量控制，请输入"AT+IFC=2,2;&W"，重启模块就一直生效了。	

参数定义：

参数	定义	取值	对取值的说明
< dce_by_dte>	TE 接收来自 TA 数据时采用的流控方法	0	无流控
		1	软件流控
		2	硬件流控
< dte_by_dce>	TA 接收来自 TE 数据时采用的流控方法	0	无流控
		1	软件流控
		2	硬件流控

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+IFC=?	查询< dce_by_dte>和< dte_by_dce>取值范围
←	+IFC: (0-2),(0-2) OK	查询结果
→	AT+IFC?	查询当前的流量控制方式
←	+IFC: 2,2 OK	硬件流控

4.11 开启和关闭 SIM 卡在位硬件检测：AT+CSDT

该命令的设置命令可以开启 USIM_CD 这个引脚的功能。当 AT+CSDT=1 的时候，使能该引脚。此时配合 SIM 卡外围检测电路就能检测 SIM 卡在位状态，即插卡或掉卡。

注：USIM_CD 脚为 USIM 卡在位检测脚，具体定义请参考相关模型的硬件手册

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CSDT=<mode>,[<level>]	OK
查询命令	AT+CSDT?	+CSDT:<mode>,[<level>] OK
测试命令	AT+CSDT=?	+CSDT: (<mode>取值列表) OK
注意事项	当检测到SIM卡在位的时候，立刻会有一个URC 上报：+CPIN: READY 当检测到SIM卡不在位的时候，立刻会有一个URC 上报：+CPIN: SIM REMOVED 如果需要关机保存，请输入AT+CSDT=<mode>;&W	

	AT+CSDT=1：默认上升沿触发，不插卡是低，插入卡是高电平 AT+CSDT=1,0：可以配置下降沿触发，不插卡是高，插入卡是低电平 AT+CSDT=1,1：可以配置上升沿触发，不插卡是低，插入卡是高电平

参数定义：

参数	定义	取值	对取值的说明
<mode>	SIM 在位检测的开启状态	0	关闭SIM卡在位检测
		1	开启SIM卡在位检测

4.12 检测某个插槽的 SIM 卡是否在位：AT*SIMDETEC

设置命令用来检测某个插槽的 SIM 卡是否在位。

测试命令用来显示目前所支持的 SIM 插槽。

语法规则：

命令类型	语法	返回
设置命令	AT*SIMDETEC=<simslot>	*SIMDETEC: <simslot>,<state> OK
测试命令	AT*SIMDETEC=?	*SIMDETEC: (1,2) OK

参数定义：

参数	定义	取值	对取值的说明
<simslot>	哪个 sim 卡槽	1	主sim卡
		2	备用sim卡(暂不支持)
<state>	SIM卡状态	NOS	SIM卡不存在
		SIM	SIM卡已插入

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT*SIMDETEC=1	
←	*SIMDETEC: 1, NOS	SIM卡不在位

	OK	

4.13 获取 Sim 类型：AT*EUICC

获知 SIM 卡类型。

语法规则：

命令类型	语法	返回
查询命令	AT*EUICC?	*EUICC: <n> OK

参数定义:

参数	定义	取值	对取值的说明
<n>	SIM card type	0	SIM
		1	USIM

4.14 通过 UART 口设置睡眠唤醒：AT+CSCLK

通过串口使模块进入睡眠的先决条件是：不接 USB 口。

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CSCLK=<n>	OK
查询命令	AT+CSCLK?	+CSCLK: <n> OK
测试命令	AT+CSCLK=?	+CSCLK: (list of supported <n>s) OK

参数定义：

参数	定义	取值	对取值的说明
<n>	睡眠设置	0	关闭模块睡眠功能。模块无法进入睡眠状态。缺省值。
		1	睡眠模式1。由模块MAIN_DTR脚控制是否进入睡眠。 当MAIN_DTR拉高（缺省为高，因为有内部上拉），设置AT+CSCLK=1，没有其他中断产生（GPIO、来电、来短信等），模块将自动进入睡眠模式1。在这种模式下，模块仍能接收来自网络的呼叫和短消息。 当模块处于睡眠模式1时，可以通过以下的几种方法唤醒模块。 ● 模块接收到外部中断信号；

			● 模块接收到语音或数据呼叫； ● 模块接收到短消息（SMS）； ● 串口接收到AT命令； ● 拉低AP_WAKEUP_MODULE引脚大概50ms 注意：模块收到语音、数据呼叫或短消息后会有URC上报
		2	睡眠模式2。 设置AT+CSCLK=2后，模块会连续监测串口数据，如果模块的串口上没有数据输入，并且没有其他中断产生（GPIO，来电，来短信，来数据等），缺省5秒后模块会自动进入睡眠模式2（**注：睡眠模式2情况下，AP_WAKEUP_MODULE电平对模块睡眠唤醒功能无影响，这个是跟休眠模式1最主要的区别**）。在这种模式下，模块仍能接收来自网络的呼叫和短消息。 当模块处于睡眠模式2时，可以通过以下的几种方法唤醒模块。 ● 模块接收到外部中断信号； ● 模块接收到语音或数据呼叫； ● 模块接收到短消息（SMS）； ● 串口接收到AT命令。
		3	睡眠模式 3(**超低功耗**)。 设置AT+CSCLK=3后，模块会连续监测串口数据，如果模块的串口上没有数据输入，并且没有其他中断产生（GPIO，来电，来短信，来数据等），缺省5秒后模块会自动进入睡眠模式3（**注：睡眠模式3情况下，AP_WAKEUP_MODULE电平对模块睡眠唤醒功能无影响，这个是跟休眠模式1最主要的区别**）。在这种模式下，模块仍能接收来自网络的呼叫和短消息。 当模块处于睡眠模式3时，可以通过以下的几种方法唤醒模块。 ● 模块接收到外部中断信号； ● 模块接收到语音或数据呼叫； ● 模块接收到短消息（SMS）； ● 串口接收到AT命令。 软件版本>=V1026支持。

举例：

命令（→）/返回（←）	实例	解释和说明
睡眠唤醒应用实例 1		
→	AT+CSCLK=2	设置为睡眠模式2。在这种睡眠模式下，以下情况同时满足时，模块进入睡眠。 ● 模块在AT口无输入 ● 没有URC上报（包括没有来电，没有短信，没有收到服务器发来

		的数据等） ● 无GPIO中断
←	OK	
→	AT+WAKETIM?	查询进入睡眠的时间
←	+WAKETIM:1 OK	查询结果为1秒钟。1秒钟是CSCLK设置睡眠后缺省进入睡眠的时间
→	AT+WAKETIM=8	如果需要修改进入睡眠的时间，可以通过WAKETIM来设置，例如改为8（一般情况下不需要设置）
←	OK	
		模块唤醒方式有以下几种： 1） 串口输入几个AT命令（一个往往唤不醒，需要多输几个） 2） 任意URC上报（包括来电，来短信，收到服务器发的数据等） 3） GPIO中断
→	AT+CSCLK=0	
←	OK	0，设置为不允许模块睡眠
睡眠唤醒应用实例 2		
→	AT+CSCLK=1	设置为睡眠模式1。在这种睡眠模式下，以下情况同时满足时，模块进入睡眠。 ● 模块在AT口无输入 ● 没有URC上报（包括没有来电，没有短信，没有收到服务器发来的数据等） ● 模块AP_WAKEUP_MODULE为高（AP_WAKEUP_MODULE高，是允许模块睡眠；AP_WAKEUP_MODULE低，是唤醒模块） ● 无GPIO中断
←	OK	
→	AT+WAKETIM?	查询进入睡眠的时间
←	+WAKETIM:1 OK	查询结果为1秒钟。1秒钟是CSCLK设置睡眠后缺省进入睡眠的时间
→	AT+WAKETIM=8	如果需要修改进入睡眠的时间，可以通过WAKETIM来设置，例如改为8（一般情况下不需要设置）
←	OK	
		模块唤醒方式有以下几种： ● 串口输入几个AT命令（一两个AT就可以了） ● 任意URC上报（包括来电，来短信，收到服务器发的数据等） ● GPIO中断 ● AP_WAKEUP_MODULE唤醒（AP_WAKEUP_MODULE低，唤醒；AP_WAKEUP_MODULE高，允许睡眠）
→	AT+CSCLK=0	

←	OK	0，设置为不允许模块睡眠
超低功耗应用实例（用于大多数的数传业务场景）		
→	AT+CSCLK=3	设置为睡眠模式3(**超低功耗**)。在这种睡眠模式下，以下情况同时满足时，模块进入睡眠。 ● 模块在AT口无输入 ● 没有URC上报（包括没有来电，没有短信，没有收到服务器发来的数据等） ● 无GPIO中断
←	OK	
→	AT+WAKETIM=1	设置在IDL状态下等待1秒进入休眠状态
←	OK	
→	AT*RTIME=2	设置在数传模式下，等待2秒进入休眠状态
←	OK	
		模块唤醒方式有以下几种： 4) 串口输入几个AT命令（一个往往唤不醒，需要多输几个） 5) 任意URC上报（包括来电，来短信，收到服务器发的数据等） 6) GPIO中断
→	AT+CSCLK=0	
←	OK	0，设置为不允许模块睡眠

4.15 设置在数传模式下，等待多长时间进入休眠状态：AT*RTIME

语法规则：

命令类型	语法	返回和说明
设置命令	AT*RTIME=<wait_time>	OK
查询命令	AT*RTIME?	*RTIME: <wait_time> OK
注意事项	**信号正常情况下，<wait_time>值越短功耗就越低，建议设置为 2。如果实际使用环境信号比较差，数据收发经常出现重传延迟，那这个值就需要改大。**	

参数定义：

参数	定义	取值	对取值的说明
<wait_time>	在数传模式下，等待多长时间进入休眠状态。	0~20	单位:秒，0表示关闭。 缺省值为0。

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT*RTIME=2	**设置等待时间为 2 秒**
←	OK	

4.16 超低功耗指令：AT+POWERMODE

EC618 系列模块>=V1143（LPAT 版本）支持，超低功耗方案参考：airpsm.cn

语法规则：

命令类型	语法	返回和说明
设置命令	AT+POWERMODE=<mode>[,<para>][,<para2>]	OK
查询命令	AT+POWERMODE?	+POWERMODE:<mode>[,<para>] OK
注意事项	1. 响应优先模式和平衡模式，串口波特率设置为 9600，可以保证休眠的时候发送的第一包 AT 不丢失 2. PSM+模式，串口波特率设置为 9600，可以保证必唤醒 3. 需要设置AT+CFGRI=1指令，才能拉低RI管脚 4. AT+CIPSEND=10\r1234567890 (发送数据用左侧，长度后面直接\r加数据的方式发送，因为等待>可能会导致超时模块进入休眠) 5.PSM+模式不支持自动升级，fota升级建议手动升级，此模式下读取的CSQ，CESQ不准确	

参数定义：

参数	定义	取值	对取值的说明
<mode>	响应优先模式（关机会保存）	"PRO"	功能介绍 1.远程唤醒速度快(2秒内) 2.信号好的情况下能达到474uA 的平均待机电流 3.支持UART和GPIO 中断唤醒 唤醒方式 1.链接的服务器，下发数据到模块，模块会被唤醒，并且打印URC，同时拉低RI管脚120ms

			2.通过DTR下拉 或者VBUS上拉可以唤醒模块 3.通过UART 发送数据给模块，可以唤醒模块
	平衡模式（关机会保存）	"STD"	功能介绍 1.远程唤醒速度快(2分钟内) 2.信号好的情况下能达到220uA 的平均待机电流 3.支持UART和GPIO 中断唤醒 唤醒方式 1.链接的服务器，下发数据到模块，模块会被唤醒，并且打印URC，同时拉低RI管脚120ms 2.通过DTR下拉 或者VBUS上拉可以唤醒模块 3.通过UART 发送数据给模块，可以唤醒模块
	PSM+模式（关机会保存）	"PSM+"	功能介绍 1.远程无法唤醒 2.待机功耗2.89uA 3.支持UART和GPIO中断以及定时器唤醒(会拉低RI唤醒或者打开单片机) 唤醒方式 1.如果输入了time 则定期会唤醒单片机，执行响应的操作 2.通过DTR下拉 或者VBUS上拉可以唤醒模块 3.通过UART 发送数据给模块，可以唤醒模块
	退出超低功耗模式（关机会保存）	"CLOSE"	
<para>	PSM+模式下，定时拉低 RI 管脚		单位：s
	"PRO","STD"模式下,是否开启 ipv6 （EC618 系列模块>=V1146（LPAT 版本）支持）	0	关闭（默认关闭）
		1	打开
<para2> （EC618 系列块>=V1148 （LPAT 版本）	PSM+模式下,是否打开 TAU (TAU 配置为 1 待机功耗能到 2.9uA，配置为 0，基站分配 TAU 时间为 54 分钟)	0	打开TAU（默认）
		1	关闭 TAU

支持）			

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+IPR=9600;&W	切换为9600，可以保证休眠的时候发送的第一包AT 不丢失
←	OK	
→	AT+POWERMODE="PRO"	设置为响应优先模式
←	OK	
→	AT+CIPSTART="TCP",112.125.89.8,42951	先进入PRO 模式后，再链接服务器)
←	CONNECT OK	
→	AT^HEARTCONFIG=1,0,300	设置心跳周期
←	OK	
→	AT^HEARTBEATHEX=0,6,414243444546	设置心跳内容
←	OK	

4.17 设置睡眠等待时间：AT+WAKETIM

语法规则：

命令类型	语法	返回和说明
设置命令	AT+WAKETIM=<wait_time>	OK
查询命令	AT+WAKETIM?	+WAKETIM:<wait_time> OK
注意事项	● **请用 WAKETIM 命令设置模块睡眠时间，请用 CSCLK 设置睡眠。**	

参数定义：

参数	定义	取值	对取值的说明
<wait_time>	在 IDLE 状态下（无 AT,无短信，无通话且 AP_WAKEUP_MODULE 为高的情况下）等待多长时间进入睡眠状态	0~100	单位:秒，0表示不等待。 缺省值为1。

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+WAKETIM=8	设置睡眠时间为8秒

←	OK	

4.18 设置 RI 指示功能：AT+CFGRI

设置命令设置 RI 脚（模块中对应 MAIN_RI 脚）在收到 URC 上报的时候是否会有低脉冲指示。如果打开指示功能(<status>=1)，则相应的 URC 到来时，RI 会产生一个 120ms 的低脉冲(缺省设置)。

语法规则：

命令类型	语法	返回
设置命令	AT+CFGRI=<status> [,<h_time>][,<l_time>][,<count>]	OK
查询命令	AT+CFGRI?	+CFGRI:<status> OK
注意事项	● 无论<status>是0还是1，当收到短信的时候，缺省都会产生一个120ms低脉冲； ● 只有设置AT+CFGRI=1后，数据业务（包括TCPIP,HTTP,MQTT,FTP）到来时的URC上报，才会使RI产生低脉冲	

参数定义：

参数	定义	取值	对取值的说明
<status>	RI 指示状态	0	RI指示功能关闭
		1	RI指示功能打开（TCPIP/FTP/HTTP/MQTT及其他URC）
<h_time>	拉高的时间	>0	单位：毫秒
<l_time>	拉低的时间	>0	单位：毫秒
<count>	拉低的次数	>0	单位：毫秒

举例:

命令(→)/返回(←)	实例	解释和说明
→	AT+CFGRI=0	这个是模块的缺省设置。 在此设置下，当收到短信的时候，会产生一个120ms低脉冲；当来电话的时候，会跳变为低电平，直到接通或挂断电话才会变成高电平
←	OK	
→	AT+CFGRI=1	在此设置下： 当收到短信的时候，会产生一个120ms低脉冲；当来电话的时候，会跳变为低电平，直到接通或挂断电话才会变成高电平； 数据业务（包括TCPIP,HTTP,MQTT,FTP）到来时的URC上报，会产生一个120ms低脉冲。

←	OK	
→	AT+CFGRI=1,200,150,3	收到短信或数据业务的URC时，会产生3个150ms的低脉冲，低脉冲之间的拉高时间为200ms；当来电话的时候，会跳变为低电平，直到接通或挂断电话才会变成高电平
←	OK	

4.19 保存 RI 设置功能：AT+CFGRISAVE

设置是否保存 AT+CFGRI 指令设置的值。

语法规则：

命令类型	语法	返回
设置命令	AT+CFGRISAVE=<save>	OK
查询命令	AT+CFGRISAVE?	+CFGRISAVE:<save> OK
注意事项	AT+CFGRI设置对应值后,需执行AT+CFGRISAVE=1才会保持对应值	

参数定义：

参数	定义	取值	对取值的说明
<save>	是否关键保存	0	不保存
		1	保存

4.20 设置唤醒字符串指令：AT^WAKEUPHEX

设置唤醒字符串指令,发送了这个指令，并且打开了 RI,那么只有指定字符串的 URC 上报才能唤醒 RI 脉冲，其他的都会屏蔽掉，想要禁用这个指令就在字符串里面填空格就行，里面的字符串是 acsii 码的 16 进制的字符串，例如 A 就是 41。(EC618 系列模块>=V1131 版本支持)

语法规则：

命令类型	语法	返回
设置命令	AT^WAKEUPHEX=<str>	OK
查询命令	AT^WAKEUPHEX?	^WAKEUPHEX: <str> OK
测试命令	AT^WAKEUPHEX=?	^WAKEUPHEX="str" OK

注意事项	字符串必须为 acsii 码的 16 进制表示，其他直接报错，不支持不可显示字符

参数定义：

参数	定义	取值	对取值的说明
<str>	是否关键保存	acsii 码的 16	字符串必须为acsii码的16进制表示，其他直接报错，不支持不可显示字符

举例:

命令(→)/返回(←)	实例	解释和说明
→	AT+CFGRI=1	打开了RI上报功能
←	OK	
→	AT^WAKEUPHEX="414243"	设置唤醒字符串
←	OK	
→	AT+CIPSTART="TCP","112.125.89.8",35046	连接服务器，当服务器发送 414243 以 HEX 发送的字符串时，会触发 RI 脉冲，其他的 URC 不符合的不会触发。
→	AT^WAKEUPHEX=" "	禁用（想要禁用这个指令就在字符串里面填空格就行）
←	OK	

4.21 读取 ADC：AT+CADC

语法规则：

命令类型	语法	返回
设置命令	AT+CADC=<adc_id>,<mode>	OK
读取命令	AT+CADC=<adc_id>,2	+CADC:<adc_id>,<volt> OK
注意	读取之前需先打开对应通道，默认最大量程1.2V	

参数定义：

参数	定义	取值	对取值的说明
<adc_id>	ADC id，共有两路	0	
		1	
<mode>	ADC 使用模式	0	禁用
		1	启用

		2	查询该路adc的电压(>=V1115版本支持)
<volt>	ADC 电压		单位：毫伏

举例:

命令(→)/返回(←)	实例	解释和说明
→	AT+CADC=0,1	打开ADC0
←	OK	
→	AT+CADC=0,2	读取ADC0值
←	+CADC: 0,1086 OK	

4.22 读取 VBAT 电压：AT+CBC

语法规则：

命令类型	语法	返回
读取命令	AT+CBC	+CBC: <volt> OK
测试命令	AT+CBC=?	OK
设置命令	AT+CBC=<powerOnVol>,<powerOffVol>	OK
读取命令	AT+CBC?	+CBC:<powerOnVol>,<powerOffVol> OK

参数定义：

参数	定义	取值	对取值的说明
<volt>	VBAT 当前电压		单位：毫伏
<powerOnVol>	开机检测的电压值	>=0	单位：毫伏
<powerOffVol>	关机检测的电压值	>=0	单位：毫伏

4.23 打开/关闭网络灯：AT+CNETLIGHT

本命令的设置命令是用来打开或关闭网络状态指示灯（NET_STATUS）。

本命令的查询命令是用来查询网络状态指示灯是否已经打开。

Air780E 系列模块分别用如下管脚信号来指示网络的状态。如下表描述了管脚定义。

网络指示管脚定义

管脚名	类型	序号	电压域	作用
NET_STATUS	DO	16	LDOAON	指示模块的网络运行状态

状态	管脚工作状态	网络状态
NET_STATUS	亮 0.2 秒，灭 1.8 秒	搜网状态
	亮 1.8 秒，灭 0.2 秒	待机
	亮 0.125 秒，灭 0.125 秒	数据传输状态 注意：该状态提示仅限于 PPP 拨号成功或者 AT 指令主动激活 PDP 成功，RNDIS 联网成功

语法规则：

命令类型	语法	返回
设置命令	AT+CNETLIGHT=<n>	OK
读取命令	AT+CNETLIGHT?	+CNETLIGHT:<n> OK

参数定义：

参数	定义	取值	对取值的说明
<n>	网络状态指示灯的开关状态	0	关闭
		1	打开

4.24 SIM 实体卡，虚拟卡切换：AT+ECSIMCFC

EC618 系列模块 LSVS 软件版本支持

语法规则：

命令类型	语法	返回
设置命令	AT+ECSIMCFG="SoftSim",<mode>	OK
查询命令	AT+ECSIMCFG?	+ECSIMCFG: "SimPowerSave",<mode>

		+ECSIMCFG: "SimPowerSave",<mode> +ECSIMCFG: "SimPresenceDetect",<mode> +ECSIMCFG: "SimSlot",<mode> +ECSIMCFG: "SoftSim",<mode> OK
测试命令	AT+ECSIMCFG=?	+ECSIMCFG: (list of supported) OK
注意事项	需要进飞行模式下进行卡切换（进入飞行模式（AT+CFUN=0），切换卡，退出飞行模式 AT+CFUN=1）	

参数定义:

参数	定义	取值	对取值的说明
<mode> (SoftSim)	是否切换为虚拟卡	0	切换为实体卡
		1	切换为虚拟卡

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT+CFUN=0	切换为飞行模式
←	OK	
→	AT+ECSIMCFG="SoftSim",1	切换为虚拟卡
←	OK	
→	AT+CFUN=1	退出飞行模式
←	OK	

4.25 SIM 卡切换：AT+SIMCROSS

模块模块开机后首先会去查询 SIM0 接口上是否有插入 SIM 卡，如果检测到 SIM0 接口上的 SIM 卡，就会读取 SIM0 接口的卡信息去注册网络；

如果 SIM0 接口上没有检测到 SIM 卡，则会再去检测 SIM1 接口上是否有 SIM 卡，如果检测到 SIM1 接口上的 SIM 卡，就会读取 SIM1 接口的卡信息去注册网络；

如果 SIM1 接口上也没有检测到 SIM 卡，则会报错:未插入 SIM 卡；

SIM0 接口和 SIM1 接口如果同时插入了 SIM 卡，默认会使用 SIM0 接口上的 SIM 卡，同时也可以通过 AT+SIMCROSS 这个指令来切换。

语法规则：

命令类型	语法	返回
设置命令	AT+SIMCROSS=<id>	OK
查询命令	AT+SIMCROSS?	+SIMCROSS:<id> OK
测试命令	AT+SIMCROSS=?	+SIMCROSS:(<id>取值范围) OK
注意事项	本命令关机保存，但是需重启生效 EC618 模块需要进飞行模式下进行卡切换（进入飞行模式（AT+CFUN=0），切换卡，退出飞行模式 AT+CFUN=1）	

参数定义:

参数	定义	取值	对取值的说明
<id>	SIM No.	0	SIM 卡 0
		1	SIM 卡 1 或内置贴片卡

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT+SIMCROSS?	
←	+SIMCROSS:0 OK	SIM 卡位置为 0
→	AT+SIMCROSS=1	切换成内置贴片卡或 SIM 卡 1
←	OK	

4.26 SIM 卡自动切换开关：AT*SIMAUTO

可根据 AT+ECSIMCFG? 返回值中的 simSlot 来判断使用哪个卡进行网络通信。

语法规则：

命令类型	语法	返回
设置命令	AT*SIMAUTO=<state>	OK
查询命令	AT*SIMAUTO?	*SIMAUTO: <state> OK
注意事项	本命令关机保存，但是需重启生效	

参数定义:

参数	定义	取值	对取值的说明

<state>	自动切换开关状态	0	关闭自动切换
		1	打开 SIM 自动切换功能，根据设置的通信卡来判断
		2	切换 SIM 卡，通过信号质量来选择合适的通信卡 注意： 此配置每次开机不会自动运行，需要通过 AT + SIMSWT 命令来触发(1.运行 AT + SIMSWT 时，会触发择优选择通信卡操作，并且会保存系统择优选择的通信卡信息，来保证重启开机后会按照保存的通信卡进行网络通信。2.当需要更新选择的通信卡时，只需要再次运行此 AT + SIMSWT 即可。3.当不知道保存的通信卡时，可以通过 AT + SIMSWT? 来查询保存的通信卡。)
注意事项	SIMSWT 命令只有在模式 2 下使用才有效		

4.27 RNDIS/ECM 功能开关：AT+RNDISCALL

EC618 系列模块>=V1140 版本支持

语法规则：

命令类型	语法	返回
设置命令	AT+RNDISCALL=<mode>	OK
读取命令	AT+RNDISCALL?	+RNDISCALL:<mode> OK
测试命令	AT+RNDISCALL=?	+RNDISCALL:(0-disable;1-enable) OK

参数定义：

参数	定义	取值	对取值的说明
<mode>	工作模式	0	关闭RNDIS/ECM网卡，重启生效
		1	打开RNDIS/ECM网卡，重启生效

4.28 重新打开 USB：AT+SYSNV=1,"ureboot",1

语法规则：

命令类型	语法	返回
执行命令	AT+SYSNV=1,"ureboot",1	OK
查询命令	AT+SYSNV=0,"ureboot"	+SYSNV: "ureboot",1 OK

4.29 GPIO 口控制：AT+CGPIO

注：本命令仅适用于合宙 4G CAT1 模块（EC618 系列）>=V1115 版本

语法规则：

命令类型	语法	返回
设置命令	AT+CGPIO=<mode>,<gpio_id>,<set_mode>	+CGPIO: <mode>,<gpio_id>,<set_mode> OK
注意事项	设计上不考虑gpio复用情况，部分gpio复用设置不生效是正常现象，另外输入默认状态原厂限制无法更改，也是正常现象。	

参数定义：

参数	定义	取值	对取值的说明
<mode>	状态	0	输入
		1	输出
<gpio_id>	具体 GPIO 号		参考对应硬件手册
<set_mode>	输出高低电平	0,1	如果是输出的话，就是： 1：输出高电平 0：输出低电平
	输入上下拉状态	0,1,2	如果是输入的话,就是： 0：下拉 1：上拉 2：悬空

4.30 IO 电平控制：AT+IOSET

注：本命令仅适用于合宙 4G CAT1 模块（EC618 系列）>=V1155 版本

语法规则：

命令类型	语法	返回
设置命令	AT+IOSET=<mode>	 OK
查询指令	AT+IOSET?	+IOSET: <mode> OK
测试指令	AT+IOSET=?	+IOSET: <0,1,2> OK

参数定义：

参数	定义	取值	对取值的说明
<mode>	IO 电平控制模式	0	硬件控制
		1	软件切换到1.8V

		2	软件切换到3.3V

5 网络服务相关命令

5.1 查询信号质量：AT+CSQ

语法规则：

命令类型	语法	返回
执行命令	AT+CSQ	+CSQ: <rssi>,<ber> OK
测试命令	AT+CSQ=?	+CSQ: (list of supported <rssi>s),(list of supported<ber>s) OK

参数定义：

参数	定义	取值	对应的接收信号强度（dbm）
<rssi>	接收信号强度指示(received signal strength indication) <rssi>=(接收信号强度 dBm+113)/2	0	小于等于-115dBm
		1	-111dBm
		2~30	-109～-53dBm
		31	大于等于-51dBm
		99	未知或不可测
<ber>	信道误码率(bit error rate，只有通话建立后，才能获知该值)	0~7	GSM 05.08 section 8.2.4所示的RXQUAL值
		99	未知或不可测

举例:

命令（→）/返回（←）	实例	解释和说明
→	AT+CSQ	查询信号强度和质量
←	+CSQ: 15,0 OK	查询结果

5.2 查询信号质量（扩展）：AT+CESQ

执行命令返回接收信号的各个参数。如果当前的服务小区不是一个GERAN小区，<rxlev>和<rxqual >设置为99；如果当前服务小区不是一个UTRA FDD或UTRA TDD小区，<rscp>设置为

255；如果当前服务小区不是一个UTRA FDD小区，<ecno>设置为255；如果当前服务小区不是一个E-UTRA小区，<rsrq>和<rsrp>设置为255。

语法规则：

命令类型	语法	返回
执行命令	AT+CESQ	+CESQ: <rxlev>,<rxqual >,<rscp>,<ecno>,<rsrq>,<rsrp> OK

参数定义:

参数	定义	取值	对取值的说明
<rxlev>	接收信号强度(received signal strength level ; 3GPP TS 45.008 subclause 8. 1.4)；整数型	0	rssi < -110 dBm
		1	-110 dBm ≤ rssi < -109 dBm
		2	-109 dBm ≤ rssi < -108 dBm
		…	…
		61	-50 dBm ≤ rssi < -49 dBm
		62	-49 dBm ≤ rssi < -48 dBm
		63	-48 dBm ≤ rssi
		…	…
		99	未知或不可测
<rxqual >	接收信号质量（请参考 3GPP TS 45.008 subclause 8. 2.4中表格R中XQUAL值）；整数型	0	BER <0.2 % Assumed value = 0.14 %
		1	0.2 %<BER<0.4 %Assumed value = 0.28 %
		2	0.4 %<BER<0.8 % Assumed value = 0.57 %
		3	0.8 %<BER<1.6 % Assumed value = 1.13 %
		4	1.6 %<BER<3.2 % Assumed value = 2.26 %
		5	3.2 %<BER<6.4 % Assumed value = 4.53 %
		6	6.4 %<BER<12.8 %Assumed value = 9.05 %
		7	12.8 %<BER Assumed value = 18.10 %
		99	未知或不可测
<rscp>	received signal code power(请参考 3GPP TS 25.133subclause 9.1 .1.3 和 3GPP TS 25.123subclause 9.1 .1.1.3) ；整数型	0	rscp < -120 dBm
		1	-120 dBm ≤ rscp < -119 dBm
		2	-119 dBm ≤ rscp < -118 dBm
		…	…
		94	-27 dBm ≤ rscp < -26 dBm
		95	-26 dBm ≤ rscp < -25 dBm
		96	-25 dBm ≤ rscp
		255	未知或不可测
<ecno>	ratio of the received energy per PN chip to the total received power spectral density (see	0	Ec/Io < -24 dB
		1	-24 dB ≤ Ec/Io < -23.5 dB
		2	-23.5 dB ≤ Ec/Io < -23 dB

	3GPP TS 25.133)；整数型	…	…
		47	-1 dB ≤ Ec/Io < -0.5 dB
		48	-0.5 dB ≤ Ec/Io < 0 dB
		49	0 dB ≤ Ec/Io
		255	未知或不可测
<rsrq>	reference signal received quality (请参考 3GPP TS 36.133 subclause 9. 1.7) ；整数型	0	rsrq < -19.5 dB
		1	-19.5 dB ≤ rsrq < -19 dB
		2	-19 dB ≤ rsrq < -18.5 dB
		…	…
		32	-4 dB ≤ rsrq < -3.5 dB
		33	-3.5 dB ≤ rsrq < -3 dB
		34	-3 dB ≤ rsrq
		255	未知或不可测
<rsrp>	reference signal received power (请参考 3GPP TS 36.133 subclause 9. 1.4) ；整数型	0	rsrp < -140 dBm
		1	-140 dBm ≤ rsrp < -139 dBm
		2	-139 dBm ≤ rsrp < -138 dBm
		…	…
		95	-46 dBm ≤ rsrp < -45 dBm
		96	-45 dBm ≤ rsrp < -44 dBm
		97	-44 dBm ≤ rsrp
		255	未知或不可测

5.3 打开 CSQ 主动上报：AT*CSQ

上报的URC（CSQ indicators）如下所列：

+CSQ:<rssi>,<ber>

+CESQ:<rxlev>,<ber>,<rscp>,<ecno>,<rsrq>,<rsrp>

***CESQ: <rxlev>,<ber>,<rscp>,<ecno>,<rsrq>,<rsrp>,<sinr>**

语法规则：

命令类型	语法	返回
设置命令	AT*CSQ=<n>	OK
查询命令	AT*CSQ?	*CSQ:<n> OK
测试命令	AT*CSQ=?	*CSQ = (list of supported <n>s)

		OK

参数定义:

参数	定义	取值	对取值的说明
<n>		0	关闭主动上报
		1	打开主动上报
<rssi>,<ber>	请参阅 AT+CSQ		
<rxlev>,<rscp>,<ecno>,<rsrq>,<rsrp>	请参阅 AT+CESQ		
<sinr>	信号与干扰加噪声比（信噪比）		

5.4 查询卡的 MCC 以及 MNC：AT+MNCC

EC618 系列模块>=V1153 版本支持

语法规则：

命令类型	语法	返回
执行命令	AT+MNCC	mcc:<mcc>,mnc:<mnc> OK

参数定义:

参数	定义	取值	对取值的说明
<mcc>	卡的 mcc		
<mnc>	卡的 mnc		

5.5 网络注册信息：AT+CREG

设置命令打开或关闭+CREG的URC上报，URC上报内容如下：

设置<n>=1，当网络注册状态发生变化时，主动上报+CREG: <stat>

设置<n>=2，当网络注册状态或驻网小区发生变化时，主动上报+CREG: <stat>[,<lac>,<ci>[,<act>]]

查询命令返回当前<n>,<stat>的值，以及当<n>=2时<lac>,<ci>的值。

语法规则：

命令类型	语法	返回
设置命令	AT+CREG=[<n>]	OK
查询命令	AT+CREG?	<n>=0(default)或 1: +CREG: <n>,<stat>

		OK <n>=2: +CREG: <n>,<stat>,<lac>,<ci>,<act> OK
测试命令	AT+CREG=?	+CREG: (list of supported <n> values) OK
URC report	+CREG: <stat>	If <n>=1，当网络注册状态发生变化时，上报此URC
	+CREG: <stat>[,<lac>,<ci>[,<act>]]	If <n>=2，当网络注册状态改变或位置区小区发生变化时
	+CEREG: <stat>[,[<tac>],[<ci>],[<AcT>][,<cause_type>,<reject_cause>]]	If <n>=3，当网络注册状态改变或位置区小区发生变化时

参数定义:

参数	定义	取值	对取值的说明
<n>	URC 上报状态	0	禁用网络注册非请求结果码（URC,Unsolicited Result Code）
		1	启用网络注册非请求结果码+CREG: <stat>
		2	启用网络注册和位置信息非请求结果码 +CREG：<stat>[,<lac>,<ci>[,<act>]]
		3	启用网络注册和位置信息非请求结果码 +CEREG: <stat>[,[<tac>],[<ci>],[<AcT>][,<cause_type>,<reject_cause>]]
<stat>	当前网络注册状态	0	未注册；ME 当前没有搜索要注册业务的新运营商
		1	已注册，本地网
		2	未注册，但 ME 正在搜索要注册业务的新运营商
		3	注册被拒绝
		4	未知
		5	已注册，漫游
		6	注册本地网，SMS-only (仅当<AcT>= E-UTRAN时可能)
		7	注册漫游网，SMS-only (仅当<AcT>= E-UTRAN时可能)
		8	仅附着紧急承载业务(不支持)
		9	注册归属地"CSFB not preferred"业务（仅当<AcT>= E-UTRAN时可能）
		10	注册漫游地"CSFB not preferred"业务（仅当<AcT>= E-UTRAN时可能）
		11	仅紧急业务可用
<lac>	位置区	-	16进制数，字符串型
<ci>	小区id	-	16进制数，字符串型
<act>	接入模式	0	GSM

		1	GSM Compact
		2	UTRAN
		3	GSM w/EGPRS
		4	UTRAN w/HSDPA
		5	UTRAN w/HSUPA
		6	UTRAN w/HSDPA and HSUPA
		7	E-UTRAN
		8	UTRAN HSPA+(CAT1 模块) EC-GSM-IoT(CAT4 模块)

举例:

命令(→)/ 返回(←)	AT Sequences	解释
→	AT+CREG=?	查询<n>的取值范围
←	+CREG:(0,1,2,3) OK	查询结果
→	AT+CREG?	<n>缺省是0，此时查询网络的注册状态
←	+CREG: 0,1 OK	查询出来<n>=0，<stat>=1 （已注册，并且注册的是本地网）
←（URC）	+CREG:0	拔掉天线或走入没有信号的区域，此时会有URC上报，表明当前未注册网络
←（URC）	+CREG:1	装上天线或走入有信号的区域，此时此时会有URC上报，表明当前重新注册上了网络
→	AT+CREG=2	设置<n>=2
←	OK	
→	AT+CREG?	查询网络的注册状态
←	+CREG: 2,1,"1863","0183db22",7 OK	查询到<n>=2，<stat>=1，<lac>=1863，<ci>=0183db22，<act>=7
←（URC）	+CREG: 1,"1863","01a2c315",7	移动模块，当小区号发生改变的时候，会有个URC上报上来

5.6 E-UTRAN EPS 网络注册状态：AT+CEREG

设置命令打开或关闭+CEREG的URC上报，URC上报内容如下：

设置<n>=1，当在E-UTRAN网的EPS注册状态发生变化时，主动上报+CEREG: <stat>

设置<n>=2，当在E-UTRAN网的EPS注册状态或驻网小区发生变化时，主动上报+CEREG: <stat>[,<tac>,<ci>,<act>]

设置<n>=3，当在E-UTRAN网的EPS注册状态或驻网小区发生变化时，主动上报 +CEREG: <stat>[,<tac>,<ci>,<act>[,<cause_type>,<reject_cause>]]

语法规则：

命令类型	语法	返回
设置命令	AT+CEREG=<n>	OK
查询命令	AT+CEREG?	+CEREG: <n>,<stat>[,[<tac>],[<ci>],[<AcT>[,<cause_type>,<reject_cause>]]] OK
测试命令	AT+CEREG=?	+CEREG: (list of supported <n>s) OK
URC上报	+CEREG: <stat>	<n>=1，当在E-UTRAN网的EPS注册状态发生变化时
	+CEREG: <stat>[,[<tac>],[<ci>],[<AcT>]]	<n>=2，当在E-UTRAN网的EPS注册状态或驻网小区发生变化时
	+CEREG: <stat>[,[<tac>],[<ci>],[<AcT>][,<cause_type>,<reject_cause>]]	<n>=3，当在E-UTRAN网的EPS注册状态或驻网小区发生变化时
	+CEREG: <stat>[,[<tac>],[<ci>],[<AcT>][,,[,[<Active Time>],[<Periodic-TAU>]]]]	<n>=4，对于请求 PSM 的 UE，使能网络注册状态主动上报、位置信息和网络定时器配置主动上报
	+CEREG:<stat>[,[<tac>],[<ci>],[<AcT>][,[<cause_type>],[<reject_cause>][,[<Active-Time>],[<Periodic-TAU>]]]]	<n>=5，对于请求 PSM 的 UE，使能网络注册状态主动上报、位置信息、注册失败原因和 网络定时器配置主动上报，上报内容为

参数定义:

参数	定义	取值	对取值的说明
<n>	URC 上报状态	0	禁止上报网络注册状态 URC +CEREG
		1	允许主动上报+CEREG: <stat>
		2	允许主动上报+CREG:<stat>[,<lac>,<ci>]
		3	允许主动上报 +CEREG: <stat>[,[<tac>],[<ci>],[<AcT>][,<cause_type>,<reject_cause>]]
		4	+CEREG: <stat>[,[<tac>],[<ci>],[<AcT>][,,[,[<Active Time>],[<Periodic-TAU>]]]]
		5	+CEREG:<stat>[,[<tac>],[<ci>],[<AcT>][,[<cause_type>],[<reject_cause>][,[<Active-Time>],[<Periodic-TAU>]]]]
<stat>	当前网络注册状态	0	未注册；ME 当前没有搜索要注册业务的新运营商
		1	已注册，本地网

		2	未注册，但 ME 正在搜索要注册业务的新运营商
		3	注册被拒绝
		4	未知(超出 E-UTRAN 网覆盖范围)
		5	注册漫游网
		6	注册归属地"SMS only"业务
		7	注册漫游地"SMS only"业务
		8	仅附着紧急承载业务
		9	注册归属地"CSFB not preferred"业务
		10	注册漫游地"CSFB not preferred"业务
		11	仅紧急业务可用
<tac>	跟踪区号	-	字符串型，16进制数
<ci>	小区id	-	字符串型，16进制数
<act>	接入模式	0	GSM
		1	GSM Compact
		2	UTRAN
		3	GSM w/EGPRS
		4	UTRAN w/HSDPA
		5	UTRAN w/HSUPA
		6	UTRAN w/HSDPA and HSUPA
		7	E-UTRAN
		8	UTRAN HSPA+(CAT1 模块) EC-GSM-IoT(CAT4 模块)
<cause_type>	整数型，定义 <reject_cause>类型	0	显示<reject_cause>包括一个 EMM 原因值（请参考 3GPP TS 24.301 Annex A）
		1	显示<reject_cause>值由厂家定义
<reject_cause>	整数型，定义注册失败原因		此值的类型由<cause_type>定义

5.7 设置网络模式：AT^SYSCONFIG

本命令设置系统模式，GSM/WCDMA接入顺序，漫游和业务域特征。

语法规则：

命令类型	语法	返回
设置命令	AT^SYSCONFIG=<mode>,<acqorder>,<roam>,<srvdoman>	OK
查询命令	AT^SYSCONFIG?	^SYSCONFIG:<mode>,<acqorder>,<roam>,<srvdomain> OK

参数定义：

参数	定义	取值	对取值的说明
<mode>	网络模式	2	Automatic selection
		13	GSM ONLY
		14	WCDMA ONLY
		15	TD-SCDMA ONLY
		16	LTE+UTRAN+GSM
<acqorder>	网络接入顺序	0	Automatic
		1	GSM first, then UTRAN
		2	UTRAN first ,then GSM
		3	LTE first,then GSM or UTRAN
<roam>	漫游支持	0	roaming disabled
		1	roaming enabled
		2	No Change
<srvdomain>	域设置	0	CS_ONLY
		1	PS_ONLY
		2	CS_PS
		3	ANY
		4	No Change

5.8 查询 cid 相关的上下文定义：AT+CGCONTRDP

设置命令返回<cid>相关的<bearer_id>, <apn>, <local_addr and subnet_mask>, <gw_addr>, <DNS_prim_addr>, <DNS_sec_addr>, <P-CSCF_prim_addr>, <P-CSCF_sec_addr>, <IM_CN_Signalling_Flag>和<LIPA_indication>。

语法规则：

命令类型	语法	返回
设置命令	AT+CGCONTRDP[=<cid>]	[+CGCONTRDP: <cid>,<bearer_id>,<apn>[,<local_addr>,<subnet_mask>[,<gw_addr>[,<DNS_prim_addr>[,<DNS_sec_addr>[,<P-CSCF_prim_addr>[,<P-CSCF_sec_addr>[,<IM_CN_Signalling_Flag>[,<LIPA_indication>]]]]]]]] [<CR><LF>+CGCONTRDP: <cid>,<bearer_id>,<apn>[,<local_addr>,<subnet_mask>[,<gw_addr>[,<DNS_prim_addr>[,<DNS_sec_addr>[,<P-CSCF_prim_addr>[,<P-CSCF_sec_addr>[,<IM_CN_Signalling_Flag>[,<LIPA_indication>]]]]]]]] […]] OK
测试命令	AT+CGCONTRDP=?	+CGCONTRDP: (list of <cid>s associated with active contexts) OK

参数定义:

参数	定义	取值	对取值的说明
<cid>	定义了一个特定的 PDP 上下文。		整数型
<bearer_id>	指定了一个承载，例如：EPS 网中的 EPS 承载，UMTS/GPRS 中的 NSAPI 承载。		整数型
<apn>	Access Point Name，接入点名称，用来选择 GGSN 或外部分组数据网		字符串型
<local_addr>	模块本地 IP 地址		字符串型
<subnet_mask>	子网掩码		字符串型
<gw_addr>	网关 IP 地址		字符串型
<DNS_prim_addr>	主 DNS 服务器 IP 地址		字符串型
<DNS_sec_addr>	辅 DNS 服务器 IP 地址		字符串型
<P-CSCF_prim_addr>	主 P-CSCF 服务器 IP 地址		字符串型
<P-CSCF_sec_addr>	辅P-CSCF服务器IP地址		字符串型
<IM_CN_Signalling_Flag>	整数型，定义PDP上下文是否仅与IM CN子系统最大值相关	0	否
		1	是
<LIPA_indication>	整数型，显示PDP上下文是否与LIPA PDN相关。本参数无法设置	0	否
		1	是

举例：

命令(→)/返回(←)	实例
→	AT+CGCONTRDP=1
←	+CGCONTRDP: 1,5,"xz01.njm2mapn","10.64.16.223","","10.64.16.223","58.240.57.33","221.6.4.66" OK

5.9 运营商查询和选择：AT+COPS

设置命令用来尝试选择和注册一个 GSM/UMTS 网络运营商。<mode>用来选择是自动注册(<oper>被忽略)，还是手动注册到<oper>（<oper>的取值由<format>定义）。如果手动选择的运营商<oper>不可用，那么也不注册其他的运营商，除非<mode>=4.

当<mode>=2 时，强制退网，并一直保持退网状态直到<mode>设置为 0，1 或 4。

正在执行注网或退网的当中设置此命令将不予执行。

查询命令返回当前的<mode>，当前注册的运营商<oper>以及当前使用的接入技术（Access Technology）。

语法规则：

命令类型	语法	返回
设置命令	AT+COPS=<mode>[,<format>[,<oper>[,<AcT>[,<Domain>]]]]	OK
查询命令	AT+COPS?	+COPS: <mode>[,<format>,<oper>[,< AcT>[, <Domain>]]] OK
测试命令	AT+COPS=?	+COPS: [list of supported (<stat>,long alphanumeric <oper>,short alphanumeric <oper>,numeric <oper>[,< AcT>])s][,,(list of supported <mode>s),(list of supported <format>s)] OK

参数定义:

参数	定义	取值	对取值的说明
<mode>	运营商注册模式	0	自动模式；<oper>被忽略
		1	手动注册运营商(<oper>必须有，<AcT>可以有)
		2	手动退网并一直保持退网状态直到<mode>设置为 0,1,4
		3	仅设置<format> (用于查询命令+COPS?)；不尝试进行注册或注销

			(<oper>字段可忽略)
		4	手动/自动(<oper>字段不可忽略)；如果手动选择失败，将进入自动选择模式(<mode>=0)
<format>	运营商的格式	0	长字符串型<oper> (采用字母数字格式)，最多 16 字符
		1	短字符串型<oper> (采用字母数字格式)，最多 8 字符
		2	数字式字符串型<oper>
<oper>	与<format>相对应的运营商取值	-	字符型；<format>表示该字符串采用字母数字型还是数字型；数字型表示运营商的方法为MCC（3位）+MNC（2位），即移动国家码+移动网络码
<stat>	网络可用状态	0	未知
		1	可用网络
		2	当前网络
		3	禁用网络
<AcT>	Access Technology， 网络类型	0	GSM
		1	GSM Compact
		2	UTRAN
		3	GSM w/EGPRS
		4	UTRAN w/HSDPA
		5	UTRAN w/HSUPA
		6	UTRAN w/HSDPA and HSUPA
		7	E-UTRAN
		8	UTRAN HSPA+
<Domain>	域	0	仅 CS
		1	仅 PS
		2	CS/PS 都有

举例:

命令（→）/返回（←）	实例	解释和说明
→	AT+COPS?	查询网络信息
←	+COPS: 0,2,"46001",7 OK	
→	AT+COPS=3,1	设置 <format>=1 短字符串型(采用字母数字格式)
←	OK	

→	AT+COPS?	查询网络信息
←	+COPS: 0,1,"UNICOM",7 OK	
→	AT+COPS=3,0	设置 <format>=0 长字符串型(采用字母数字格式)
←	OK	
→	AT+COPS?	查询网络信息
←	+COPS: 0,0,"CHN-UNICOM",7 OK	

5.10 自动时区更新：AT+CTZU

查询命令可以查询是否使能通过NITZ来更新模块时间的功能。

缺省为使能。(模块可以通过AT+CTZU=0/1设置禁用/启用)

注： NITZ=Network Identity and Time Zone，可通过运营商网络来提供本地时间。

语法规则：

命令类型	语法	返回
查询命令	AT+CTZU?	+CTZU: <fun> OK
测试命令	AT+CTZU=?	+CTZU: (list of supported <fun>s) OK

参数定义:

参数	定义	取值	对取值的说明
<fun>	时区更新状态	1	允许 NITZ 自动更新

5.11 打开 NITZ 自动上报：AT+CTZR

使用该命令设置是否打开 NITZ URC 自动上报。

该命令不支持设置，仅支持查询。缺省为打开。

语法规则：

命令类型	语法	返回
查询命令	AT+CTZR?	+CTZR: <fun> OK

测试命令	AT+CTZR=?	+CTZR: (list of supported <fun>s) OK

参数定义:

参数	定义	取值	对取值的说明
<fun>	报告状态	0	不允许 NITZ URC 上报
		1	允许 NITZ URC上报

5.12 (URC)NITZ 自动上报：+NITZ:<time>,<ds>

语法规则：

URC
+NITZ:<time>,<ds>

参数定义:

参数	定义	取值	对取值的说明
<time>	时间	yy/mm/dd,hh:mm:ss±tz	tz :时区, -48~+48 举例: +NITZ:11/08/02,09:27:39+32,0
<ds>	夏令时	0	没有调整夏令时
		1	+1 小时(等于 4 个 quarter in <tz>)调整为夏令时
		2	+2 小时(等于 8 个季度<tz>)调整为夏令时

5.13 设置 NITZ 自动上报时区：AT+NITZDISSET

EC618 系列模块>=1140 软件版本支持。

语法规则：

命令类型	语法	返回
查询命令	AT+NITZDISSET?	+NITZDISSET: <zone>,<id>
设置命令	AT+NITZDISSET=<zone>,<id>	OK

参数定义:

参数	定义	取值	对取值的说明
<zone>	东西时区	"e"	东时区
		"w"	西时区

<id>	时区的号	0-12	

举例：

命令(→)/返回(←)	实例	解释和说明
→	AT+NITZDISSET="e",8	表示 URC 显示东八区时间
←	OK	

5.14 查询当前工作频段：AT*BANDIND

设置命令可以打开频段自动上报。

查询命令返回当前的工作频段。

语法规则：

命令类型	语法	返回
设置命令	AT*BANDIND[=<n>]	OK
查询命令	AT*BANDIND?	*BANDIND: <n>[,<band>,<AcT>] OK
测试命令	AT*BANDIND=?	*BANDIND: (0,1) OK
URC	<n>=1而且频段改变时，自动上报 URC: ***BANDIND: <band>, <Act>**	

参数定义：

参数	定义	取值	对取值的说明
<n>	<n>=1而且频段改变时，自动上报 *BANDIND: <band>, <Act>	0	disable
		1	enable
<act>	Access Technology，接入机制	0	GSM
		1	GSM Compact
		2	UTRAN
		3	GSM w/EGPRS
		4	UTRAN w/HSDPA
		5	UTRAN w/HSUPA

		6	UTRAN w/HSPA	
		7	E-UTRAN	
		8	UTRAN HSPA+	
<band>	频段	0	PGSM 900	当<act>=0/1/3
		1	DCS GSM 1800	
		2	PCS GSM 1900	
		3	EGSM 900	
		4	GSM 450	
		5	GSM 480	
		6	GSM 850	
		0	UMTS BAND1	当<act>=2/4/5/6/8
		1	UMTS BAND2	
		2	UMTS BAND3	
		3	UMTS BAND4	
		4	UMTS BAND5	
		5	UMTS BAND6	
		6	UMTS BAND7	
		7	UMTS BAND8	
		8	UMTS BAND9	
		9	UMTS BAND10	
		10	UMTS BAND11	
		11	UMTS BAND12	
		12	UMTS BAND13	
		13	UMTS BAND14	
		14	UMTS BAND15	
		15	UMTS BAND16	
		16	UMTS BAND17	
		17	UMTS BAND18	
		18	UMTS BAND19	
		1	LTE BAND 1	当<act>=7

		2	LTE BAND 2	
		3	LTE BAND 3	
		4	LTE BAND 4	
		5	LTE BAND 5	
		6	LTE BAND 6	
		7	LTE BAND 7	
		8	LTE BAND 8	
		9	LTE BAND 9	
		10	LTE BAND 10	
		11	LTE BAND 11	
		12	LTE BAND 12	
		13	LTE BAND 13	
		14	LTE BAND 14	
		15	LTE BAND 15	
		16	LTE BAND 16	
		17	LTE BAND 17	
		18	LTE BAND 18	
		19	LTE BAND 19	
		20	LTE BAND 20	
		21	LTE BAND 21	
		22	LTE BAND 22	
		23	LTE BAND 23	
		24	LTE BAND 24	
		25	LTE BAND 25	
		26	LTE BAND 26	
		27	LTE BAND 27	
		28	LTE BAND 28	
		29	LTE BAND 29	
		30	LTE BAND 30	
		31	LTE BAND 31	

		32	LTE BAND 32	
		33	LTE BAND 33	
		34	LTE BAND 34	
		35	LTE BAND 35	
		36	LTE BAND 36	
		37	LTE BAND 37	
		38	LTE BAND 38	
		39	LTE BAND 39	
		40	LTE BAND 40	
		41	LTE BAND 41	

5.15 查询接入机制（Access Technology）：AT^CACAP

查询命令返回当前小区的接入机制(Access Technology)。

语法规则：

命令类型	语法	返回
查询命令	AT^CACAP?	+CACAP: <act> OK
测试命令	AT^CACAP=?	+CACAP:(0-7) OK

参数定义：

参数	定义	取值	对取值的说明
<act>		0	GSM
		1	GSM Compact
		2	UTRAN
		3	GSM w/EGPRS
		4	UTRAN w/HSDPA
		5	UTRAN w/HSUPA
		6	UTRAN w/HSPA

		7	E-UTRAN

5.16 查询当前的系统信息：AT^SYSINFO

本命令查询当前的系统信息，例如：服务状态，业务域，漫游状态等。

语法规则：

命令类型	语法	返回
执行命名	AT^SYSINFO	^SYSINFO:<srv_status>,<srv_domain>,<roam_status>,<sys_mode>,<sim_state>,<sys_submode> OK

参数定义：

参数	定义	取值	对取值的说明
<srv_status>	服务状态	0	no service
		1	restricted service
		2	valid service
		3	restricted area service
		4	power service
<srv_domain>	业务域	0	no service
		1	CS only
		2	PS only
		3	CS and PS
<roam_status>	漫游状态	0	no roaming
		1	roaming
<sys_mode>	网络模式	0	no service
		1	reserved
		2	reserved
		3	GSM/GPRS
		4	WCDMA
		5	TD_SCDMA
		17	LTE
<sim_state>	SIM 卡状态	0	sim卡无效

		1	sim卡有效
		255	SIM未插入或PIN码未解锁
<sys_submode>	网络子模式	0	GSM
		1	GSM Compact
		2	UTRAN
		3	GSM w/EGPRS
		4	UTRAN w/HSDPA
		5	UTRAN w/HSUPA
		6	UTRAN w/HSDPA and HSUPA
		7	E-UTRAN

5.17 设置基站定位（LBS）服务器的域名和端口：AT+GSMLOCFG

语法规则：

命令类型	语法	返回
设置命令	AT+GSMLOCFG=<addr>,<port>	OK
测试命令	AT+GSMLOCFG=?	+GSMLOCFG: (addr),(port) OK
查询命令	AT+GSMLOCFG?	+GSMLOCFG: bs.openluat.com:12411 OK

参数定义：

参数	定义	取值	对取值的说明
<addr >	服务器域名		目前只支持公司的特定的适配网站
<port>	端口		目前只支持公司的特定的适配网站

举例:

命令（→）/返回（←）	实例	解释和说明
→	AT+GSMLOCFG="bs.openluat.com",12411	设置服务器域名和端口
←	OK	

→	AT+GSMLOCFG?	查询服务器域名和端口
←	+GSMLOCFG: bs.openluat.com:12411 OK	

5.18 读取基站定位（LBS）信息和时间：AT+CIPGSMLOC

语法规则：

命令类型	语法	返回
设置命令	AT+CIPGSMLOC=<type>,<cid>	**If <type>=1:** +CIPGSMLOC:<locationcode>[,<latitude>,<longitude>,<date>,<time>] OK **If <type>=2:** +CIPGSMLOC: <locationcode>[,<date>,<time>] OK **If error is related to ME functionality:** +CME ERROR: <err>
测试命令	AT+CIPGSMLOC=?	+CIPGSMLOC:(list of supported <type>s),(range of <cid>) OK

参数定义：

参数	定义	取值	对取值的说明
<type >	操作类型	1	查看精度、维度和时间
		2	只查看时间
<cid>	as <cid> defined in +SAPBR	1-3	
<longitude>	当前经度(以度为单位)		经度(小数点后保留 7 位)
<latitude>	当前纬度，以度表示		纬度(小数点后保留 7 位)
<date>	格式为 yy/mm/dd		例如 2023/11/08
<time>	格式为 hh/mm/ss		例如 15:47:26
<locationcode>		0	成功
		1	未找到数据

		6	参数错误
		7	未知错误
		404	未找到
		408	请求超时
		601	网络错误
		602	内存不足
		603	DNS错误
		604	堆栈忙
		65535	其它错误

举例:

命令（→）/返回（←）	实例	解释和说明
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	设置承载类型为GPRS
←	OK	
→	AT+SAPBR=3,1,"APN",""	设置PDP承载之APN参数 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用（此<apn>可以通过AT+CGDCONT?来查询），所以输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN
←	OK	
→	AT+SAPBR=1,1	激活GPRS PDP上下文
←	OK	
→	AT+SAPBR=2,1	查询是否激活
←	+SAPBR: 1,1,010.169.179.213 OK	返回中有IP地址表明激活成功
→	AT+CIPGSMLOC=1,1	查询位置和时间（超时时间30S）
←	+CIPGSMLOC: 0,034.7983328,114.3214505,2023/06/05,14:38:50 OK	
→	AT+CIPGSMLOC=2,1	只查询时间
←	+CIPGSMLOC: 0,2023/06/05,14:38:55 OK	
→	AT+SAPBR=0,1	去激活PDP上下文

←	OK	

5.19 读取 WIFI 定位信息和时间：AT+WIFILOC

语法规则：

命令类型	语法	返回
设置命令	AT+WIFILOC=<type>,<cid>[,<mode>]	**If <type>=1:** +WIFILOC:<locationcode>[,<latitude>,<longitude>,<date>,<time>] OK **If <type>=2:** +WIFILOC: <locationcode>[,<date>,<time>] OK **If error is related to ME functionality:** +CME ERROR: <err>
测试命令	AT+WIFILOC=?	+WIFILOC:(list of supported <type>s),(range of <cid>) OK

参数定义：

参数	定义	取值	对取值的说明
<type >	操作类型	1	查看精度、维度和时间
		2	只查看时间
<cid>	as <cid> defined in +SAPBR	1-3	
<mode>	扫描优先级(默认数据优先) EC618 系列模块>=V1143 版本支持	0	数据优先
		1	wifi 扫描优先
<longitude>	当前经度(以度为单位)		经度(小数点后保留 7 位)
<latitude>	当前纬度，以度表示		纬度(小数点后保留 7 位)
<date>	格式为 yy/mm/dd		例如 2023/11/08
<time>	格式为 hh:mm:ss		例如 15:47:26
<locationcode>		0	成功
		1	未找到数据
		2	权限错

		6	参数错误
		4 或 7	未知错误
		255	wifi成功
		404	未找到
		408	请求超时
		601	网络错误
		602	内存不足
		603	DNS错误
		604	堆栈忙
		65535	其它错误

举例:

命令（→）/返回（←）	实例	解释和说明
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	设置承载类型为GPRS
←	OK	
→	AT+SAPBR=3,1,"APN",""	设置PDP承载之APN参数 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用（此<apn>可以通过AT+CGDCONT?来查询），所以输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN
←	OK	
→	AT+SAPBR=1,1	激活GPRS PDP上下文
←	OK	
→	AT+SAPBR=2,1	查询是否激活
←	+SAPBR: 1,1,010.169.179.213 OK	返回中有IP地址表明激活成功
→	AT+WIFILOC=1,1	查询位置和时间（超时时间40S）
←	+WIFILOC: 0,034.7983328,114.3214505,2023/06/05,14:40:32 OK	
→	AT+WIFILOC=2,1	只查询时间
←	+WIFILOC: 0,2023/06/05,14:40:32	

	OK	
→	AT+SAPBR=0,1	去激活PDP上下文
←	OK	

5.20 获取 WIFI 信息：AT+WIFISCAN

EC618 系列模块>=V1150 版本支持

语法规则：

命令类型	语法	返回	说明
设置命令	AT+WIFISCAN=<wifiPriority>,<maxScanNum>,<channel>	+WIFISCAN: <mac>, <rssi>, <channel> OK	指定条件扫描（阻塞方式查询）
查询命令	AT+WIFISCAN?	+WIFISCAN: <mac>, <rssi>, <channel> OK	阻塞方式查询所有信道
执行命令	AT+WIFISCAN	OK +WIFISCAN: <mac>, <rssi>, <channel>	非阻塞方式查询所有信道，立即返回，查询完毕主动上报

参数定义：

参数	定义	取值	对取值的说明
wifiPriority	数据/wifi扫描优先	0	数据通信优先
		1	wifi 扫描优先
maxScanNum	最大扫描wifi显示数量	1~30	
<channel>	扫描信道	0~255	只支持 2.4G 频段，所以支持扫描的信道有 1~14
<mac>	MAC地址		
<rssi>	接收信号强度指示（Received Signal Strength Indication）		

5.21 显示 WIFISCAN 扫描数量的设置指令：AT+WIFISCANCOUNT

EC618 系列模块>=V1144 版本支持

语法规则：

命令类型	语法	返回	说明
设置命令	AT+WIFISCANCOUNT=<mode>	OK	
查询命令	AT+WIFISCANCOUNT?	+WIFISCANCOUNT: <mode> OK	

参数	定义	取值	对取值的说明
<mode>	是否显示WIFISCAN扫描数量	0	不显示
		1	显示

举例:

命令（→）/返回（←）	实例	解释和说明
→	AT+WIFISCANCOUNT=1	设置显示WIFISCAN扫描数量
←	OK	
→	AT+WIFISCAN	非阻塞方式查询所有信道
←	OK +WIFISCAN COUNT: 10 +WIFISCAN: "22:f2:2c:6f:55:9a",-42,11 +WIFISCAN: "18:f2:2c:6f:55:9a",-42,11 +WIFISCAN: "c8:bf:4c:ce:d4:fe",-53,11 +WIFISCAN: "a6:f9:33:3a:45:e4",-60,1 +WIFISCAN: "c0:cc:42:f0:6b:b0",-84,4 +WIFISCAN: "78:13:e0:d2:75:e6",-84,5 +WIFISCAN: "48:7d:2e:5c:66:a4",-84,13 +WIFISCAN: "e0:38:3f:dc:17:f0",-85,7 +WIFISCAN: "4c:e9:e4:1c:ba:88",-91,1 +WIFISCAN: "34:f7:16:19:6f:a2",-96,11	

5.22 WIFISCAN 扫描否显示 SSID 和时间提前量的设置指令：AT+WIFISCANCONF

EC618 系列模块>=V1155 版本支持

语法规则：

命令类型	语法	返回	说明
设置命令	AT+WIFISCANCONF=<SSID>,<Time Advance>	OK	设置 AT+WIFISCAN 返回的 wifi 参数中是否显示 ssid和时间提前量
查询命令	AT+WIFISCANCONF?	+WIFISCANCONF: SSID: 0 TaValue: 0 OK	默认参数为0，都不显示
注意事项	如果WIFI扫描出来的SSID为空字符串 “”，说明该wifi是被隐藏的。时间提前量一般情况下获取都是-1，要处于连接态的时才会获取到值，例如刚建立tcp连接，或者刚执行过一次基站或者wifi定位。		

参数	定义	取值	对取值的说明
<SSID>	SSID，Wifi名称	0	不显示
		1	显示
<Time Advance>	时间提前量	0	不显示
		1	显示

举例:

命令（→）/返回（←）	实例	解释和说明
→	AT+WIFISCANCONF=1,1	设置显示WIFISCAN扫描显示ssid和时间提前量
←	OK	
→	AT+WIFISCAN	非阻塞方式查询所有信道
	OK +WIFISCAN: "","22:f2:2c:6f:55:9a",-48,11,-1 +WIFISCAN:"TP-LINK_559A","18:f2:2c:6f:55:9a",-48,11,1 +WIFISCAN: "Atmation","c8:bf:4c:ce:d4:fe",-52,11,-1	

	+WIFISCAN: " "ESP"","2a:22:08:09:01:0f",-60,11,-1 +WIFISCAN: "","4c:e9:e4:1c:ba:88",-83,1,-1 +WIFISCAN: "HY2019","4c:e9:e4:1c:ba:8a",-84,1,-1 +WIFISCAN: "xjdz","f4:ee:14:13:5b:7a",-87,1,-1 +WIFISCAN:"ChinaNet-DNXs","e0:38:3f:dc:17:f0",-89,3,-1 +WIFISCAN:"ChinaNet-7sbA","08:6b:d1:69:a6:00",-89,6,-1 +WIFISCAN: "CMCC-p6g9","78:13:e0:d2:75:e6",-94,9,-1	

5.23 流量查询命令：AT^DATAINFO

语法规则：

命令类型	语法	返回
设置命令	AT^DATAINFO=<n>,<period>	OK
查询命令	AT^DATAINFO? (只有打开状态下可查询)	^DATAINFO: <n>,<period> OK
执行命令	AT^DATAINFO (只有打开状态下可用，查询使用流量，单位字节，掉电保存)	^DATAINFO: <上行>,<下行> OK
测试命令	AT^DATAINFO=?	^DATAINFO: (0-1),(0,30-65535) OK

参数定义：

参数	定义	取值	对取值的说明
<n>	流量查询开关 （掉电保存）	0	关闭
		1	打开
<period>	保存流量数据时间间隔	30-65535	单位：秒

5.24 (URC)系统模式： ^MODE

显示系统模式有改变。

语法规则：

URC
^MODE:<SysMainMode>,<SysMode>

参数定义：

参数	取值	解释
<SysMainMode>,<SysMode>	17,17	TD LTE capabilities （4G）
	5/15,8	3G only (3G)
	5/15,7	3G, HSDPA, and HSDPA capabilities (3G)
	5/15,6	3G and HSUPA capabilities (3G)
	5/15,5	3G and HSDPA capabilities (3G)
	3,3	GSM, GPRS, and EGPRS capabilities （2G）
	3,2	GSM and GPRS capabilities (2G)
	3,1	GSM only (2G)
	0,0	无服务

5.25 小区信息查询：AT+CCED

本命令可以查询本小区和最多 6 个临小区的信息。

语法规则：

命令类型	语法	返回
设置命令	AT+CCED=<mode>,<requested dump>	OK
测试命令	AT+CCED=?	+CCED: (list of <mode>s),(list of <requested dump>s) OK

参数定义：

参数	定义	取值	对取值的说明
<mode>	工作模式	0	一次上报
		1	周期性上报
		2	关闭上报
<requested dump>	消息类型	1	主小区，即服务小区信息
		2	邻小区信息 如果是 LTE 邻小区，则上报：

			MCC,MNC,frequency,cellid,rsrp,rsrq,tac,SrxLev,pcid 如果是 GSM 邻小区，则上报： MCC,MNC,lac,cellid,bsic,rxlev
		8	主小区的 RSSI 指示，即 Rxlev（0~31）

举例：

<requested dump>=1，上报主小区或服务小区的信息：

如果当前是 LTE 模式，则上报：

+CCED:LTE current cell:MCC,MNC,imsi,roamingFlag,bandInfo,bandwidth,dlEarfcn,cellid,rsrp,rsrq,tac,SrxLev,pcid

如果当前是 GSM 模式，则上报：

+CCED:GSM current cell info:MCC,MNC,lac,cellid,bsic,rxlev,RxLevSub,Arfcn

举例：

一次性查询当前小区(LTE 模式)：

AT+CCED=0,1

+CCED:LTE current cell:460,00,460045926307603,0,40,n100,39148,140542123,51,29,6334,34,351

OK

一次性查询当前小区(GSM 模式)：

AT+CCED=0,1

+CCED:GSM current cell info:460,00,18be,5045,13,63,63,6

OK

当前小区信息周期性上报（LTE 模式）：

AT+CCED=1,1

+CCED:LTE current cell:460,00,460045926307603,0,40,n100,39148,140542123,61,31,6334,44,351

OK

+CCED:LTE current cell:460,00,460045926307603,0,40,n100,39148,140542123,61,31,6334,44,351
+CCED:LTE current cell:460,00,460045926307603,0,40,n100,39148,140542123,61,31,6334,44,351
+CCED:LTE current cell:460,00,460045926307603,0,40,n100,39148,140542123,61,35,6334,44,351

当前小区信息周期性上报（GSM 模式）：

AT+CCED=1,1

+CCED:GSM current cell info:460,00,18be,5045,13,61,61,6

OK

+CCED:GSM current cell info:460,00,18be,5045,13,63,63,6
+CCED:GSM current cell info:460,00,18be,5045,13,63,63,6
+CCED:GSM current cell info:460,00,18be,5045,13,63,63,6
+CCED:GSM current cell info:460,00,18be,5045,13,63,63,6

<requested dump>=2，上报邻接小区的信息：

如果当前是 LTE 模式，则上报：

+CCED:LTE neighbor cell: MCC,MNC,frequency,cellid,rsrp,rsrq,tac,SrxLev,pcid

如果当前是 GSM 模式，则上报：

+CCED:GSM neighbor cell info: MCC,MNC,lac,cellid,bsic,rxlev

举例：

一次性查询邻接小区(LTE 模式)：

AT+CCED=0,2

+CCED:LTE neighbor cell:460,00,38950,140541985,57,24,6334,36,351
+CCED:LTE neighbor cell:460,00,1300,26224401,48,24,6334,27,37
+CCED:LTE neighbor cell:460,00,1300,26224402,43,15,6334,22,38
+CCED:LTE neighbor cell:460,00,38400,26224397,42,23,6334,21,191
+CCED:LTE neighbor cell:460,00,40936,12793923,34,15,6334,13,191
+CCED:LTE neighbor cell:460,00,3590,26224415,44,9,6334,27,318
+CCED:LTE neighbor cell:460,00,3590,26224416,47,19,6334,30,319

OK

一次性查询邻接小区(GSM 模式)：

AT+CCED=0,2

+CCED:GSM neighbor cell info:460,00,6334,20522,31,75
+CCED:GSM neighbor cell info:460,00,6334,0,21,80
+CCED:GSM neighbor cell info:460,00,6334,20521,30,91

OK

邻接小区信息周期性上报（LTE 模式）：

AT+CCED=1,2

+CCED:LTE neighbor cell:460,00,38950,140541985,57,24,6334,36,351
+CCED:LTE neighbor cell:460,00,1300,26224401,48,24,6334,27,37
+CCED:LTE neighbor cell:460,00,1300,26224402,43,15,6334,22,38
+CCED:LTE neighbor cell:460,00,38400,26224397,42,23,6334,21,191
+CCED:LTE neighbor cell:460,00,40936,12793923,34,15,6334,13,191
+CCED:LTE neighbor cell:460,00,3590,26224415,44,9,6334,27,318

+CCED:LTE neighbor cell:460,00,3590,26224416,47,19,6334,30,319

OK

+CCED:LTE neighbor cell:460,00,38950,140541985,57,24,6334,36,351
+CCED:LTE neighbor cell:460,00,1300,26224401,48,24,6334,27,37
+CCED:LTE neighbor cell:460,00,1300,26224402,43,15,6334,22,38
+CCED:LTE neighbor cell:460,00,38400,26224397,42,23,6334,21,191
+CCED:LTE neighbor cell:460,00,40936,12793923,34,15,6334,13,191
+CCED:LTE neighbor cell:460,00,3590,26224415,44,9,6334,27,318
+CCED:LTE neighbor cell:460,00,3590,26224416,47,19,6334,30,319

邻接小区信息周期性上报（GSM 模式）：

AT+CCED=1,2

+CCED:GSM neighbor cell info:460,00,6334,20522,31,76
+CCED:GSM neighbor cell info:460,00,6334,0,21,80
+CCED:GSM neighbor cell info:460,00,6334,20521,30,94

OK

+CCED:GSM neighbor cell info:460,00,6334,20522,31,85
+CCED:GSM neighbor cell info:460,00,6334,0,21,78
+CCED:GSM neighbor cell info:460,00,6334,20521,30,83

5.26 在工程模式下查询 GSM/UMTS/LTE 信息：AT+EEMGINFO

语法规则：

命令类型	语法	返回
查询命令	AT+EEMGINFO?	+EEMGINFO:<state>,<nw_type> OK +EEMLTESVC:<info> +EEMLTEINTRA:<info> +EEMLTEINTER:<info>

参数定义:

参数	定义	取值	解释
<state>	MT state	0	Idle 状态

		1	Dedicated 状态
		2	PS PTM 状态
		3	有效状态
<nw_type>	network type	0	GSM
		1	UMTS
		2	LTE

5.27 LTE 工程模式下服务小区信息：+EEMLTESVC

语法规则：

+EEMLTESVC: <mcc>,<length of mnc> ,<mnc>,<tac>,<PCI>,<dlEuarfcn>,<ulEuarfcn>,<band>,<dlBandwidth>,<ci>,<transMode>,<rsrp>,<rsrq>,<sinr>,<MainRsrp>,<DiversityRsrp>,<MainRsrq>,<DiversityRsrq>,<rssi>,<cqi>,<currPuschTxPower>,<rankIndex>,<ErrorModeState>,<emmState>,<serviceState>,<IsSingleEmmRejectCause>,<EMMRejectCause>,<MmeGroupId>,<MmeCode>,<mTmsi>

参数定义：

参数	解释
<mcc>	Mobile Country Code
<length of mnc>	length of mnc
<mnc>	Mobile Network Code
<tac>	Tracking area code
<PCI>	Physical Cell Identifier
<dlEuarfcn>	downlink arfcn
<ulEuarfcn>	uplink arfcn
<band>	band
<dlBandwidth>	dlBandwidth
<ci>	cellId
<transMode>	transMode
<rsrp>	rsrp
<rsrq>	rsrq
<sinr>	sinr
<MainRsrp>	Rsrp in main antenna
<DiversityRsrp>	Rsrp in slave antenna
<MainRsrq>	Rsrq in main antenna
<DiversityRsrq>	Rsrq in slave antenna
<rssi>	rssi

<cqi>	cqi
<currPuschTxPower>	current Pusch Tx Power in dBm
<rankIndex>	rankIndex
<ErrorModeState>	ErrorModeState
<emmState>	emmState
<serviceState>	serviceState
<IsSingleEmmRejectCause>	IsSingleEmmRejectCause
<EMMRejectCause>	EMMRejectCause
<MmeGroupId>	MmeGroupId
<MmeCode>	MmeCode
<mTmsi>	mTmsi

5.28 LTE 工程模式下同频信息：+EEMLTEINTRA

展示 LTE 工程模式下同频（intra frequency）信息。

语法规则：

URC
+EEMLTEINTRA: <p1>,<p2>,<p3>,<p4>,<p5>,<p6>,<p7>,<p8>,<p9>

参数定义：

参数	解释
<p1>	index of ENGMODE INTRAFREQ
<p2>	phyCellId
<p3>	euArfcn
<p4>	rsrp
<p5>	rsrq
<p6>	mcc
<p7>	mnc
<p8>	tac
<p9>	cellId

5.29 LTE 工程模式下频间信息：+EEMLTEINTER

展示 LTE 工程模式下频间（inter frequency）信息。

语法规则：

+EEMLTEINTER: <p1>,<p2>,<p3>,<p4>,<p5>,<p6>,<p7>,<p8>,<p9>

参数定义：

参数	解释
<p1>	index of ENGMODE INTERFREQ
<p2>	phyCellId
<p3>	euArfcn
<p4>	rsrp
<p5>	rsrq
<p6>	mcc
<p7>	mnc
<p8>	tac
<p9>	cellId

5.30 (URC)网络服务类型指示

URC	释义
+E_UTRAN Service	表示进入 4G LTE 网络服务
+GSM Service	提示进入 2G GSM 网络服务

6 音频相关命令

注：本章命令仅适用于 EC618 系列模块 LSAT 版本>=V1103 版本支持

6.1 TTS（Text To Speech）功能：AT+CTTS

语法规则：

命令类型	语法	返回
设置命令	AT+CTTS=<mode>[,<text>][,<pcm>]	OK
查询命令	AT+CTTS?	+CTTS: <status> OK
测试命令	AT+CTTS=?	OK

URC	当 TTS 播放完毕，会有以下 URC 上报： +CTTS:0

参数定义：

参数	定义	取值	对取值的说明
<mode>	TTS工作模式	0	停止播放TTS
		1	播放TTS，<text>使用UCS2编码
		2	播放TTS，<text>使用GBK编码（中文）和ASCII编码（数字）
		3	播放TTS，<text>使用uft-8编码（EC618系列模块LSAT版本>=V1103支持）
<text>	TTS文本		中文和数字，最大长度2047字节
<pcm>	TTS模式	0	播放TTS
		1	播放TTS，输出PCM数据
<status>	TTS工作状态	0	TTS非播放状态
		1	TTS播放状态
注意：<pcm>参数，EC618系列模块LSAT版本>=V1111支持。(默认值为0)			

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CTTS=2,"123，欢迎你的到来"	播放数字和中文。 注：目前暂时无法播放英文，<text>中的英文发音不是单词，而是一个一个分离的字母。
←	OK	
←(URC)	+CTTS:0	播放完毕，会有这个URC上报
→	AT+CTTS=0	也可以在播放过程中用这个命令终止播放
←	OK	

6.2 设置 TTS 播放模式：AT+CTTSPARAM

语法规则：

命令类型	语法	返回
设置命令	AT+CTTSPARAM=<volume>,<mode>,<pitch>,<speed>[,<channel>][,<codec>][,<papin>][,<codecstart>][,<pa_open>][,<codec_pa_close>][,<def_pa_out>,<codecend>]	OK
查询命令	AT+CTTSPARAM?	+CTTSPARAM:<volume>,<mode>,<pitch>,<speed>,<channel>,<codec>,<papin>,<codecstart>,<pa_open>,<codec_pa_close>,<def_pa_out>

		OK

参数定义：

参数	定义	取值	对取值的说明
<volume>	TTS播放音量	0-100	缺省值为50
<mode>	TTS播放模式，range：0-3	0	自动读数字，先根据数字规则读数字
		1	自动读取数字，首先根据电报规则读取数字
		2	根据电报规则读取数字
		3	根据数字规则读取数字
<pitch>	TTS播放音高	1-100	缺省值为50
<speed>	TTS播放速度	1-100	缺省值为50
<channel>	TTS播放通道	0	主通道
		1	辅通道
<codec>	解码芯片	0	ES7149
		1	TM8211
		2	内置模拟音频模式
<papin>	具体GPIO号		PA引脚具体GPIO号,参考对应硬件手册
<codecstart>	输入空白声音的时长(ms)	1-2000	默认值为1500
<pa_open>	指令开启后多久打开PA(ms)	1-2000	默认值为150
<codec_pa_close>	播放结束后，关闭pa 和codec 之间间隔的时间(ms)	1-2000	默认值为1
<def_pa_out>	控制pa播放时io的电平高低	0	默认是0，播放是高电平，静音是低电平
		1	设置成1，播放是低电平，静音是高电平
<codecend>	播放完tts后多久关闭pa使能	1-2000	1

注意：<codec>参数，EC618系列模块LSAT版本>=V1103支持。(默认值为0)

<papin>参数，EC618系列模块LSAT版本>=V1121支持。(内置模拟音频模式播放方式需要使用内置模拟音频模式专用板测试)

<codecstart>，<pa_open>，<codec_pa_close>参数，EC618系列模块LSAT版本>=V1131支持。

<def_pa_out>参数，EC618系列模块LSAT版本>=V1149支持。(默认值为0)

<codecend>参数，EC618系列模块LSAT版本>=V1155支持。(默认值为1)

6.3 语音文件播放：AT+CAUDPLAY

EC618 系列模块 AUAT 版本>=V1146 版本支持

语法规则：

命令类型	语法	返回

设置命令	AT+CAUDPLAY=<mode>[,<filename>]	OK
查询命令	AT+CAUDPLAY?	+CAUDPLAY: <mode> OK
测试命令	AT+CAUDPLAY=?	+CAUDPLAY: (<mode>取值范围列表),<filename> OK
URC 上报	1)AT+CAUDPLAY=1,<filename>开始播放，模块自己结束播放后提示+CAUDPLAY: 1,<duration> // <duration>为播放持续时间 2) AT+CAUDPLAY=2 结束播放，提示+CAUDPLAY: 1,<duration> //<duration>为从开始播放到停止播放的持续时间 3) AT+CAUDPLAY=3 暂停播放，OK 4) AT+CAUDPLAY=4 恢复播放，模块自己结束播放后提示+CAUDPLAY: 1,<duration> // <duration>为从开始播放到停止播放的持续时间	

参数定义：

参数	定义	取值	对取值的说明
<mode>	使用模式	1	开始播放(start)
		2	停止播放(stop)
		3	暂停播放(pause)
		4	恢复播放(resume)
<filename>	音频文件名字		AMR,WAV,MP3格式。需要加后缀，支持8K采样音频数据
<duration>	播放持续时间		单位：毫秒

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+FSCREATE="test.mp3"	创建音频文件
←	OK	
→	AT+FSWRITE="test.mp3",0,1268,20	输入音频文件。这里长度1268只是举例，请以实际为准
←	>	**这里输入音频文件内容（二进制，即16进制）**
←	OK	
→	AT+CAUDPLAY=1,"test.mp3"	播放音频文件 注：音频文件名字需要加后缀
←	OK	
←	+CAUDPLAY: 1,2585	播放结束上报

6.4 本地音频播放音量控制：AT+CRSL

本命令影响设备本地音频播放的音量。

语法规则：

命令类型	语法	返回
设置命令	AT+CRSL=<level>	OK
查询命令	AT+CRSL?	+CRSL:<level> OK
测试命令	AT+CRSL=?	+CRSL: (<level>取值范围) OK

参数定义：

参数	定义	取值	对取值的说明
<level>	音频通道	1~15	1 声音最小，15 声音最大。缺省值为 15

7 短消息命令

7.1 PDU 短信编码格式介绍

Octet 1								Octet 2								Octet sequence
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
SCA Length = n								1	TON			NPI				1 ~ 2
SCA																3 ~ (1+n)
RP	UDHI	SRR	VPF		RD	MTI		TP-MR								(2+n)~ (3+n)
DA – length = m								1	TON			NPI				(4+n)~(5+n)
Destination Address																(6+n) ~ (6 + n + (m+1) / 2)
PID								DCS								(7 + n + (m+1) / 2) ~ (8 + n + (m+1) / 2)
VP (Valid Period) (length = s, s= 1 octet or 7 octet according to VPF value)																(8 + n + (m+1) / 2)~ (8 + s + n + (m+1) / 2)
UDL (User Data Length) = x								UD (user data)								(9 + s + n + (m+1) / 2) ~(10 + x + s + n + (m+1) / 2)
UD (user data)																

图表 1 ： MO 短信 PDU 格式

Octet 1								Octet 2								Octet sequence
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
SCA Length = n								1	TON			NPI				1 ~ 2
SCA (Service Center Address)																3 ~ (1+n)
RP	UDHI	SRI			MMS	MTI		OA length = m								(2+n)~ (3+n)
1	TON				NPI			OA (Originating Address)								(4+n)~(5+n)
OA (Originating Address)																(6+n) ~ (5 + n + (m+1) / 2)
PID								DCS								(6 + n + (m+1) / 2) ~ (7 + n + (m+1) / 2)
SCTS (Service Center Time Stamp)																(8 + n + (m+1) / 2)~ (14 + s + n + (m+1) / 2)
UDL (User Data Length) = x								UD (user data)								(15 + x + n + (m+1) / 2) ~(16 + x + n + (m+1) / 2)
UD (user data)																

图表 2 ： MT 短信 PDU 格式

参数定义：

参数	定义	解释和说明
MO	Mobile Originated	模块发送的
MT	Mobile Terminated	模块接收的
SCA Length		短信中心地址的长度
TON	Type of Number	号码类型： 000：未知 001：国际 010：国内 111：留作扩展
NPI	Numbering Plan Identifier	号码鉴别： 0000：未知 0001：ISDN/电话号码 1111：留作扩展
SCA	Short Message Center Address	短信中心地址
MTI	Message Type Identifier	信息类型： Bit Explanation 11 Reserved 10 SMS-STATUS REPORT (SC => MS) 01 SMS-SUBMIT (MS => SC) 00 SMS-DELIVER (SC => MS)
RD	Reject Duplicate	拒绝重复短信
VPF	Validity Period Format	有效期格式
SRR	Status Report Request	状态报告请求，在MO短信中设定
SRI	Status Report Indication	状态报告指示，在MT短信中指示该MT短信是否为状态报告
UDHI	User Data Header Indicator	用户数据头指示
RP	Reply Path	回复路径

7.2 选择短消息服务:AT+CSMS

语法规则：

命令类型	语法	返回
设置命令	AT+CSMS=<service>	+CSMS:<mt>,<mo>,<bm> OK
查询命令	AT+CSMS?	+CSMS:<service>,<mt>,<mo>,<bm> OK
测试命令	AT+CSMS=?	+CSMS:(<service>取值列表)

		OK

参数定义：

参数	定义	取值	对取值的说明
<service>	短消息服务级别	0	GSM03.40和03.41(SMS的AT指令语法与GSM07.05 Phase 2中的4.7.0版本兼容；支持不需要新指令语法的Phase 2+特性(比如：使用Phase 2+新编码方案的消息路由))
		1	GSM03.40 和 03.41(SMS 的 AT 指令语法与 GSM07.05 Phase 2+版本兼容)
		128	pdu 模式操作向前兼容于与 phase2不一致的版本。此时会认为短信 pdu 里不包含短信中心号（暂不支持）
<mt>	SMS-MO（发短信）	0	不支持
		1	支持
<mo>	SMS-MT（收短信）	0	不支持
		1	支持
<bm>	小区广播消息	0	不支持
		1	支持

7.3 短消息优先存储区选择：AT+CPMS

本命令可设置用于读取、存储等操作的存储器，包括<mem1>, <mem2>和<mem3>。

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CPMS=<mem1>[,<mem2>[,<mem3>]]	+CPMS:<used1>,<total1>,<used2>,<total2>,<used3>,<total3> OK
查询命令	AT+CPMS?	+CPMS: <mem1>,<used1>,<total1>,<mem2>,<used2>,<total2>,<mem3>,<used3>,<total3> OK
测试命令	AT+CPMS=?	+CPMS:(<mem1>取值列表),(<mem2>取值列表),(<mem3>取值列表) OK

参数定义：

！

参数	定义	取值	对取值的说明
<mem1>	读取和删除消息时使用的存储器，涉及以下3个AT指令：AT+CMGL AT+CMGR AT+CMGD	"SM"	SM即SIM 卡
		"ME"	ME为模块
<mem2>	写、存储和发送消息时使用的存储器，涉及以下2个AT指令：AT+CMSS 和 AT+CMGW	"SM"	SM即SIM 卡
		"ME"	ME为模块
<mem3>	若没有建立到 TE的路由，则将接收的消息存储在该存储器	"SM"	SM即SIM 卡
		"ME"	ME为模块
<used1><used2><used3>	<mem1，2，3>中当前存储消息的数量	-	整数型
<total1><total2><total3>	<mem1，2，3>中可存储的消息的总量	-	整数型

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CPMS=?	
←	+CPMS: ("SM","ME"),("SM","ME"),("SM","ME") OK	
→	AT+CPMS?	查询当前优选存储区类型
←	+CPMS: "SM",8,50,"SM",8,50,"SM",8,50 OK	注：目前仅支持"SM"存储类型。请使用：AT+CPMS="SM","SM","SM" 设置短信存储类型。

7.4 短消息中心地址：AT+CSCA

该指令适用于 PDU 格式和 TEXT 格式，使用设置指令，可更新级 SMSC(Short Message Service Center)地址。通过该地址，可以发送移动终端 SMS，TEXT 模式下，发送命令和写命令都可使用该项设置；PDU 模式下，发送和设置指令也可使用该项设置，但条件是 PDU 编码后的 SMSC 地址长度等于 0。这里必须说明的是虽然用户可以自行设置短消息的服务中心地址，但不能随心所欲，否则短消息发送不出去，因此在发送短消息之前首先必须搞清楚 SIM 卡所属的短消息服务中心地址。

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CSCA=<sca>[,<tosca>]	OK
查询命令	AT+CSCA?	+CSCA:<sca>,<tosca>

		OK
测试命令	AT+CSCA=?	OK

参数定义：

参数	定义	取值	对取值的说明
<sca>	短信中心地址（Short Message Center Address）	-	字符型；BCD数字(或GSM缺省字母字符)需要转换为字符；格式由<tosca>指定
<tosca>	短信中心地址格式（Type of sca）	-	8 位整数型(缺省值请参考<toda>) 129 ISDN/电话编号方式设计，国家的/国际的未知。 145 ISDN/电话编号方式设计，国际号码。 161 ISDN/电话编号方式设计，国家的号码。 128~255 其他值查阅 GSM 04.08 章节 10.5.4.7

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CSCA="+8613010314500",145	对一个联通的SIM卡设置短信中心号码
←	OK	
→	AT+CSCA?	查询下短信中心号码的设置
←	+CSCA: "+8613010314500",145 OK	已经设置成功

7.5 短消息格式：AT+CMGF

设置指令用于指定短消息的输入和发送的格式，即告诉TA输入输出的消息格式是PDU格式还是TEXT格式。

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CMGF=[<mode>]	OK
查询命令	AT+CMGF?	+CMGF: <mode> OK
测试命令	AT+CMGF=?	+CMGF: (<mode>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<mode>	显示消息发送、列表、读和写指令以及接收到消息时的主动汇报使用的格式	0	PDU模式，默认值
		1	TEXT模式

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT+CMGF?	查询当前模式
←	+CMGF: 0 OK	当前模式是PDU模式

7.6 设置短信 TEXT 模式参数：AT+CSMP

文本模式下(即AT+CMGF=1)，当向网络侧**发送**短消息或将短消息**存放**在存储器中时，使用该设置命令可选择需要的附加参数取值。除此之外，设置命令还可用于设置从SMSC 接收到该短消息时算起的有效期（<vp>的取值范围为0... 255）或定义有效期终止的绝对时间(<vp>为字符串时)。

语法规则：

命令类型	语法	返回
设置命令	AT+CSMP=[<fo>[,<vp>[,<pid>[,<dcs>]]]]	OK
查询命令	AT+CSMP?	+CSMP:<fo>,<vp>,<pid>,<dcs> OK
测试命令	AT+CSMP=?	+CSMP: (<fo>取值列表),(<vp>取值列表), (<pid>取值列表), (<dcs>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<fo>	短信首字节（First Octet）	17,21,33,37,49,53	对<fo>字节的具体描述如下（以 SMS-SUBMIT 为例）： b7 b6 b5 b4 b3 b2 b1 b0 RP UDHI SRR VPF RD MTI MTI：消息类型 b1=0&b0=0 表示SMS-DELIVER b1=0&b0=1 表示SMS-SUBMIT 其他消息类型请参考GSM03.40 VPF：定义短信有效时间的格式 b4=1&b3=0：Relative format，此时<vp>是1个字节的整数型 b4=1&b3=1：Absolute format，此时<vp>是7个字节的整数型 SRR：Status Report Request，设置是否需要短信状态报告 UDHI：User Data Header Indicator，指示User Data单元是否有一个header RP：Reply Path，回复路径

<table>
<tr><td></td><td></td><td></td><td>RD：Reject Duplicate，拒绝重复短信</td></tr>
<tr><td><vp></td><td>短信有效期（Valid Period）</td><td></td><td>取值由<fo>字段的VPF决定：
如果VPF=10(Binary)，则<vp>为相对模式，与短信有效时间的对应关系如下：
<table><tr><th><vp>值</th><th>有效时间</th></tr><tr><td>0-143(00 to 8F)</td><td>(vp + 1) x5分钟</td></tr><tr><td>144-167(90 to A7)</td><td>12小时 + ((vp – 143)×30分钟</td></tr><tr><td>168-196(A8 to C4)</td><td>(vp – 166) x 1 天</td></tr><tr><td>197-255(C5 to FF)</td><td>(vp – 192) x 1 星期</td></tr></table>如果VPF=11(Binary)，则<vp>为绝对模式，是7个字节的字符型，表示短信有效期到期的时间点。</td></tr>
<tr><td><pid></td><td>TP-协议-标识</td><td>0</td><td>整数型，具体请参考GSM03.40</td></tr>
<tr><td><dcs></td><td>短信内容编码方案</td><td></td><td>整数型，具体请参考GSM03.38。一般情况下：
0- 7bit GSM Default
4- 8bit Data
8- UCS2</td></tr>
</table>

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT+CSMP=17,11,0,0	设置TEXT模式短信发送的各参数： <fo>=17（缺省），表示MTI=01（binary）=SMS-SUBMIT，并且VPF=10（binary）= Relative format <vp>=11，表示有效时间为（11+1）x5分钟=1个小时 <dcs>=0，表示编码格式是7bit GSM Default
←	OK	

7.7 控制 TEXT 模式下短信头信息显示：AT+CSDH

设置命令可控制是否在文本模式下的结果码中显示详细的头信息.

语法规则：

命令类型	语法	返回
设置命令	AT+CSDH=[<show>]	OK
查询命令	AT+CSDH?	+CSDH:<show> OK
测试命令	AT+CSDH=?	+CSDH:<show> OK

参数定义：

参数	定义	取值	对取值的说明
<show>	是否显示头信息	0	对于SMS-DELIVER 和 SMS-SUBMIT短信，不在+CMT, +CMGL, +CMGR 结果码中显示+CSCA和+CSMP指令设置的参数(<sca>、<tosca>、 <fo>、<vp>、<pid>、<dcs>)，也不显示<length>、<toda>或<tooa>；对于 +CMGR 指令结果码中的 SMS-COMMAND，不显 示<pid>、<mn>、<da>、<toda>、<length>、<cdata>
		1	在结果码中显示这些取值

举例：

命令（→）/ 返回（←）	实例	解释和说明
→	AT+CSDH?	查询是否显示头信息
←	+CSDH: 0 OK	查询结果为：不显示相关头信息
→	AT+CMGR=8	查询位置为8的短信，内容为"good!"
←	+CMGR: "REC READ","+86131******56",,"12/08/08,10:43:04+32" Good! OK	查询结果不显示 <tooa>,<fo>,<pid>,<dcs>,<sca>, <tosca>,<length>
→	AT+CNMI=2,2,0,0,0	设置短信自动内容直接上报给TE，不缓存（将+CNMI的第二个参数<mt>设置为2即可）
←	OK	
←（URC）	+CMT: "+86131******56",,"12/08/08,11:09:23+32" Report	发送一个内容为Report的短信给模块，上报内容不显示<fo>、<vp>、<pid> 和 <dcs>、<sca>、<tosca>
→	AT+CSDH=1	设置<show>=1，即显示头信息
←	OK	
→	AT+CMGR=8	仍旧查询位置8的短信
←	+CMGR: "REC UNREAD" ,"+86131******56",,"12/08/08,10:43:04+32",145,17,0,0,"+8613800210500",145,5 Good! OK	查询结果显示 <tooa>,<fo>,<pid>,<dcs>,<sca>, <tosca>,<length>
←（URC）	+CMT: "+86131******56",,"12/08/08,11:05:45+32",145,17,0,0,"+8613800210500",145,7	发送一个内容为weather的短信给模块，上报内容显示<fo>、<vp>、<pid> 和 <dcs>、<sca>、<tosca>

	weather	

7.8 新消息指示：AT+CNMI

该指令用于 PDU 格式和 TEXT 格式，当 TE 处于在用状态时(如：DTR 信号处于"ON"状态)，使用设置指令，可设置新消息如何从网络侧发送到 TE。若 TE 处于待用状态（如：DTR 信号处于"OFF"状态），消息接收流程应该按照 GSM 03.38 的规定。

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CNMI=[<mode>[,<mt>[,<bm>[,<ds>[,<bfr>]]]]]	OK
查询命令	AT+CNMI?	+CNMI:<mode>,<mt>,<bm>,<ds>,<bfr> OK
测试命令	AT+CNMI=?	+CNMI: (<mode>取值列表),(<mt>取值列表),(<bm>取 值列表),(<ds>取值列表),(<bfr>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<mode>	TA 给 TE 传送短信的模式	0	缓冲TA中的非请求结果码；若TA结果码缓冲器已满，结果码指示可以缓冲存储在其他存储空间或者把最旧的非请求结果码指示丢弃，替换为新接收到的指示。
		1	当TA-TE间的链路被占用(比如：在线数据模式下)，丢弃结果码指示，并拒绝新接收消息的非请求结果码。否则，直接转发给TE。
		2	当TA-TE间的链路被占用(比如：在线数据模式下)，缓冲TA中的非请求结果码；当链路释放后，把所有结果码发送给TE。否则，直接转发给TE。
		3	在TA 处于数据模式的情况下，使用特定的TA-TE 连接技术将结果码和数据同时传给TE。
<mt>	新短信上报的方式	0	没有SMS-DELIVER的指示发送给TE
		1	若 SMS-DELIVER 存储在 ME/TA，存储位置靠非请求结果码 +CMTI: <mem>,<index> 来提示给TE。
		2	SMS-DELIVER消息(类别2的消息和位于消息等待指示组中的消息(存储消息))直接发送到TE。 使用如下指令的非请求结果码： +CMT：[<alpha>],<length><CR><LF><pdu> (启用PDU模式)或者+CMT：<oa>,[<alpha>],<scts> ,<tooa>[,*<fo>,<pid>,<dcs>,<sca>,<tosca>,<length>*]<CR><LF><data>(启用TEXT模式。斜体部分是否显示由+CSDH设置命令决定)

<table>
<tr><td rowspan="4"></td><td rowspan="4"></td><td></td><td>说明：若AT指令接口作为唯一的显示设备，ME必须支持类别0消息和位于消息等待指示组中的消息的存储(丢弃消息)。</td></tr>
<tr><td>3</td><td>通过使用<mt>=2定义的非请求结果码，类别3的SMS-DELIVER消息直可接发送到TE。其他数据编码方案下的消息显示结果均遵循<mt>=1的定义。</td></tr>
<tr><td colspan="2"><table>
<tr><td><mt></td><td>no class or class 1</td><td>class 0 or messagewaiting indication group (discard)</td><td>class 2 or messagewaiting indication group (store)</td><td>class 3</td></tr>
<tr><td>1</td><td>+CMTI</td><td>[+CMTI1)]</td><td>+CMTI</td><td>+CMTI</td></tr>
<tr><td>2</td><td>+CMT & +CNMA3)</td><td>+CMT [& +CNMA2)]</td><td>+CMTI</td><td>+CMT & +CNMA3)</td></tr>
<tr><td>3</td><td>+CMTI</td><td>[+CMTI1)]</td><td>+CMTI</td><td>+CMT& +CNMA3)</td></tr>
<tr><td colspan="5">1) 除了AT命令没有其他显示手段的时候，显示该结果码；
2) 当+CSMS <service>=1并且ME只有AT命令这一个显示手段时，需要+CNMA确认
3) 当+CSMS <service>=1时需要+CNMA确认
注：如果在一定的时间内没有+CNMA确认，MT将不再发短信提示给TE</td></tr>
</table>SMS-DELIVER 结果码（+CMT, +CMTI）和确认(+CNMA)的关系总结：</td></tr>
<tr><td rowspan="2"><bm></td><td rowspan="2">小区广播短消息的上报方式</td><td>0</td><td>无 CBM 指示发送到 TE</td></tr>
<tr><td>2</td><td>收到的CBM使用如下格式直接发送到TE：
+CBM：<length><CR><LF><pdu>(启用PDU 模式)或者
+CBM： <sn>,<mid>,<dcs>,<page>,<pages><CR><LF><data>(启用TEXT模式)</td></tr>
<tr><td rowspan="2"><ds></td><td rowspan="2">短信状态报告上报方式</td><td>0</td><td>无SMS-STATUS-REPORTS发送到TE。</td></tr>
<tr><td>1</td><td>SMS-STATUS-REPORT消息使用如下格式直接发送到TE：
+CDS：<length><CR><LF><pdu>(启用PDU模式)或者
+CDS：<fo>,<mr>,[<ra>],[<tora>],<scts>, <dt>,<st>(启用TEXT模式)
注：<dt>,<st>请参考+CMGR</td></tr>
<tr><td rowspan="2"><bfr></td><td rowspan="2"></td><td>0</td><td>当<mode>为1～3时，这条指令所定义的TA缓存中的结果码被发送到TE(在发送之前，OK应该被接收到)</td></tr>
<tr><td>1</td><td>当<mode>为1～3时，将清除该指令中定义的TA对非请求结果码的缓冲</td></tr>
</table>

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CNMI=?	查询参数范围
←	+CNMI: (0-3),(0-3),(0-3),(0-1),(0-1) OK	模块查询结果
→	AT+CNMI=2,1	设置CNMI参数

←	OK	
←（URC）	+CMTI: "SM",1	此时收到一个短信，缓存在<mem1>中，只用+CMTI上报新短信位置索引
→	AT+CNMI=1,2	设置<mt>=2，即新短信不缓存，直接上报
←	OK	
→	AT+CSMS=1	必须把+CSMS的第一个参数设置为1，才支持+CNMA命令
←	OK	
→	AT+CMGF?	查询当前短信模式
←	+CMGF: 0 OK	是PDU模式
←（URC）	+CMT: ,24 0891683108200105F0040D91683129634152F60000218080418442230 4F7349B0D	此时收到一个PDU短信，具体分析如下： +CMT: ,24 24-PUD长度，短信中心号码不算在PDU内 08- Length of SCA，短信中心地址长度（按字节数计算，包括91在内） 91- 短信中心地址的TON/NPI 683108200105F0- 短信中心地址，需要两两反转，反转后是8613800210500。 04- First Octet，PDU的首字节 0D-源地址长度（号码长度） 91683129634152F6- 源地址。需要两两反转，反转后是8613923614256 00- PID（Protocol Identifier） 00- DCS（Data Coding Scheme），0表示7BIT GSM DEFAULT 21808041844223- SCTS（SM Center Time Stamp），短信中心时间戳，表示SC收到短信的时间为:12年8月8日14:48:24,+8GMT 04- 用户数据的长度 F7349B0D- 7BIT GSM DEFAULT编码的will
→	AT+CNMA	收到短信消息后，立即发送 AT+CNMA 命令
←	OK	
→	AT+CMGF=1	设置为TEXT模式
←	OK	
	AT+CNMI=1,2	
←（URC）	+CMT: "+86131******56",,"12/08/08,11:05:45+32",145,17,0,0,"+86138002105	收到一个新短信，是TEXT模式

	00",145,7 Will-go	
→	AT+CNMA	收到短信消息后，立即发送AT+CNMA命令
←	OK	

7.9 新短消息确认：AT+CNMA

使用执行命令，可确认是否正确接收新消息(SMS-DELIVER 或SMS-STATUS-REPORT)，该新消息是由 MT 直接发送到 TE 而不缓存。

在同时满足2个条件的情况下，需要通过AT+CNMA给出短信确认：

→通过 AT+CSMS=1，将<service>设为 1；

→通过AT+CNMI命令将<mt>设为2或者将<ds>设置为1；

在满足上述2个条件后，TE如果在收到短信后没有通过AT+CNMA给MT确认，CNMI的参数<mt>和<ds>会被重置为0，MT也不再给TE发短信。

语法规则：

命令类型	语法	返回和说明
执行命令	TEXT模式（AT+CMGF=1）： AT+CNMA	OK
	PDU模式（AT+CMGF=0）： AT+CNMA[=<n>[,<length>[<CR>PDUis given<ctrl-Z/ESC>]]]	OK 说明：<Ctrl+Z>即十六进制的 1A，<ESC>即十六进制的 1B
测试命令	AT+CNMA=?	OK

参数定义：

参数	定义	取值	对取值的说明
<n>	PDU 模式下确认短信的方式	0	该指令与文本模式定义的指令执行类似
		1	发送 RP-ACK(或正确接收到的缓冲结果码)
		2	发送 RP-ERROR(若 PDU 未给定，ME/TA 将发送 GSM 03.40 TP-FCS 取值设定为 "FF"的 SMS-DELIVER-REPORT 消息(非请求错误原因))

举例：

命令（→）/返回（←）	实例	解释和说明
		具体实例请参考+CNMI条目

7.10 发送短信：AT+CMGS

使用设置指令，可将 SMS(SMS-SUBMIT)从 TE 发送到网络侧。发送成功后，消息参考值<mr>将返回给 TE。

语法规则：

命令类型	语法	返回和说明
设置命令	文本模式 （AT+CMGF=1）情况下： AT+CMGS=<da>[,<toda>]<CR> >text to send<CTRL-Z/ESC>	发送成功则返回： +CMGS:<mr> OK 说明：<CTRL-Z>即0x1A是发送，<ESC>即0x1B是终止发送。
		发送失败则返回： +CMS ERROR:<err>
	PDU模式（AT+CMGF=0）情况下： AT+CMGS=<length><CR> >PDU to send<CTRL-Z/ESC>	发送成功，返回： +CMGS:<mr> OK 说明：<Ctrl+Z>即十六进制的1A，<ESC>即十六进制的1B
		发送失败则返回： +CMS ERROR: <err>
测试命令	AT+CMGS=?	OK

参数定义：

参数	定义	取值	对取值的说明
<da>	目的地址（Destination Address）	-	GSM 03.40 TP-Destination-Address 中"地址－取值"字段，字符型；将 BCD 数值(或缺省 GSM 字母格式的字符) 转换为当前选择的 TE 字符集中的字符(请参考 TS 07.07 中的+CSCS 指令)；地址类型由<toda>定义
<toda>	目的地址类型（Type of <da>）	-	整数型的 GSM 04.11 TP-Destination-Address 中的 8 位"类型－地址"字段(当<da>的首字符为+(IRA 43)时， 缺省值为 145；否则缺省值为 129)
<length>	TPDU短信长度	-	整数型。该长度不将短信中心号码的长度计算在内
<mr>	消息参考（MessageReference）	0~255	整数型的 GSM 03.40 TP-Message-Reference

举例：

命令（→）/返回（←）	实例	解释和说明
发送TEXT模式英文短信：		
→	AT+CMGF=1	设置为TEXT模式

←	OK	
→	AT+CSMP?	查询当前的TEXT模式短信参数
←	+CSMP: 17,11,0,0 OK	当前的<dcs>=0 （GSM）
	AT+CSCS?	
	+CSCS: "IRA" OK	
→	AT+CMGS="139****6785"	发英文短信给测试手机。 注：139****6785是发送目的号码，请按照实际填写，不要照抄
←	>	此时会返回>
→	hello,world！<Ctrl+Z>	返回>后，输入短信内容，例如：hello,world！然后输入<Ctrl+Z>即0x1A就可以发送出去了
←	+CMGS: 108 OK	发送成功，<mr>=108
发送TEXT模式中文短信：		
→	AT+CMGF=1	设置为TEXT模式
←	OK	
→	AT+CSMP?	
←	+CSMP: 17,167,0,0 OK	
→	AT+CSMP=17,167,0,8	修改最后一个参数<dcs>=8
←	OK	
→	AT+CMGS="139****6785"	注：139****6785是发送目的号码，请按照实际填写，不要照抄
←	>	此时会返回>
→	611F8C22<Ctrl+Z>	返回>后，输入短信内容611F8C22，是16进制格式。即：0x610x1F0x8C0x22，<Ctrl+Z>发送，<Ctrl+Z>即十六进制的1A
←	+CMGS: 73 OK	发送成功
发送PDU格式的中文短信：		
→	AT+CMGF=0	设置成PDU模式
←	OK	

<table>
<tr><td>→</td><td>AT+CMGS=19
>0011100D91683161450179F90008
0004611F8C22 [Ctrl+Z]</td><td>19-是PDU字符串的长度(短信中心单元不应包括在内)
00- 表示短信中心的长度为0，这种情况下发送短信的时候直接从SIM卡中取SCA。该00是短信中心单元，不计入PDU长度中
11- PDU短信首字节
<table><tr><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>RP</td><td>UDHI</td><td>SRR</td><td colspan="2">VPF</td><td>RD</td><td colspan="2">MTI</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr></table>
10- TP-MR，benchmark
0D- 目标地址（DA）长度
91683161450179F9- 目标地址（收信方号码）
00- Protocol Identifier (PID)
08- Data coding scheme (DCS)，8表示UCS2
00- Valid Period (VP)
04- User Data的长度
611F8C22- User Data，发送的短信内容（本例为中文"感谢"）</td></tr>
<tr><td>←</td><td>+CMGS: 110

OK</td><td>发送成功</td></tr>
</table>

7.11 把消息写入存储器：AT+CMGW

使用设置指令，可将SMS(SMS-DELIVER 或 SMS-SUBMIT)从TE发送到存储器<mem2>，并返回已存储消息的存储位置<index>参数。除非<stat>指定其他参数，否则，该消息的状态将被设置为"存储未发送"。

语法规则：

<table>
<tr><th>命令类型</th><th>语法</th><th>返回和说明</th></tr>
<tr><td rowspan="3">设置命令</td><td rowspan="2">TEXT模式下（AT+CMGF=1）：
AT+CMGW[=<oa/da>[,<tooa/toda>[,<stat>]]]<CR>text is entered<ctrl-Z/ESC></td><td>成功，返回：
+CMGW: <index>

OK
说明：<Ctrl+Z>即十六进制的1A，<ESC>即十六进制的1B</td></tr>
<tr><td>失败则返回：
+CMS ERROR: <err></td></tr>
<tr><td>PDU模式下（AT+CMGF=0）：
AT+CMGW=<length>[,<stat>]<CR> PDU is given<ctrl-Z/ESC></td><td>成功，返回：
+CMGW: <index>

OK</td></tr>
</table>

		说明：<Ctrl+Z>即十六进制的1A，<ESC>即十六进制的1B
		失败则返回： +CMS ERROR: <err>
测试命令	AT+CMGW=?	返回： OK

参数定义：

参数	定义	取值	对取值的说明
<da>	目的地址（Destination Address）	-	GSM 03.40 TP-Destination-Address 中"地址－取值"字 段，字符型；将 BCD 数值(或缺省 GSM 字母格式的字符) 转换为当前选择的 TE 字符集中的字符(请参考 TS 07.07 中的+CSCS 指令)；地址类型由<toda>定义
<toda>	目的地址类型（Type of <da>）	-	整数型的 GSM 04.11 TP-Destination-Address 中的 8 位"类型－地址"字段(当<da>的首字符为+(IRA 43)时， 缺省值为 145；否则缺省值为 129)
<oa>	源 地 址（ OriginatingAddress）	-	GSM 03.40 TP-Originating-Address 中"地址－取值"字 段，字符型；将 BCD 数值(或缺省 GSM 字母格式的字符) 转换为当前选择的 TE 字符集中的字符(请参考 TS 07.07 中的+CSCS 指令)；地址类型由<tooa>定义
<tooa>	源地址类型（Type of <oa>）	-	整数型的 GSM 04.11 TP-Originating-Address 中的 8 位"类型－地址"字段(当<oa>的首字符为+(IRA 43)时， 缺省值为 145；否则缺省值为 129)
<length>	PDU短信长度	-	整数型。该长度不将短信中心号码的长度计算在内
<stat>	短信状态	REC UNREAD	已接收的未读消息(TEXT模式下，即+CMGF=1)
		REC READ	已接收的已读消息(TEXT模式下，即+CMGF=1)
		STO UNSENT	存储未发送消息(TEXT模式下，即+CMGF=1)
		STO SENT	存储已发送消息(TEXT模式下，即+CMGF=1)
		0	已接收的未读消息(PDU模式下，即+CMGF=0)
		1	已接收的已读消息(PDU模式下，即+CMGF=0)
		2	存储未发送消息(PDU模式下，即+CMGF=0)
		3	存储已发送消息(PDU模式下，即+CMGF=0)

举例：

命令（→）/返回（←）	实例	解释和说明
存储一个TEXT短信(英文)：		
→	AT+CMGF=1	设置为TEXT模式
←	OK	
→	AT+CSCS="GSM"	或AT+CSCS="IRA"

←	OK	
→	AT+CSMP=17,167,0,0	
←	OK	
→	AT+CMGW="139****6785"	发送短信给目的号码
←	>	此时会返回>
→	HI!<Ctrl+Z>	返回>后，输入短信内容，然后输入<Ctrl+Z>即0x1A就可以保存短信
←	+CMGW: 6 OK	返回的index=6，表示该短信在存储区的位置是第6个
存储一个TEXT短信(中文)：		
→	AT+CMGF=1	设置为TEXT模式
←	OK	
→	AT+CSCS="GSM"	设置AT命令中字符的编码格式为UCS2
←	OK	
→	AT+CSMP=17,167,0,8	Dcs=8，表示短信存放的格式为UCS2编码
←	OK	
→	AT+CMGW="10086"	目的地址是10086(UCS2编码格式) 内容为"感谢"(UCS2编码格式)
←	>	此时会返回>
→	611F8C22<Ctrl+Z>	返回>后，输入短信内容611F8C22（hex格式），然后输入<Ctrl+Z>即0x1A就可以保存短信
←	+CMGW: 7 OK	该短信存在index=7的位置
存储一个PDU短信：		
→	AT+CMGF=0	设置为PDU格式
←	OK	
→	AT+CMGW=19	
←	>	此时会返回>
→	001110D91683161450179F9000800004611F8C22<Ctrl+Z>	PDU格式短信内容为"感谢"。然后输入<Ctrl+Z>即0x1A就可以保存短信
←	+CMGW: 8 OK	该短信存在index=8的位置

7.12 从存储器发送短信：AT+CMSS

使用设置指令，可将消息存储器<mem2>中，位置取值参数为<index>的消息发送到网络侧(SMS-SUBMIT 或

SMS-COMMAND)。若给定 SMS-SUBMIT 消息的新接收地址参数<da>，应使用该参数，而不能使用已存储消息的参数。发送成功后，参考值<mr> 将返回给 TE。

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CMSS=<index>[,<da>[,<toda>]]	+CMSS:<mr> OK
测试命令	AT+CMSS=?	OK

参数定义：

参数	定义	取值	对取值的说明
<index>	待发送短信在存储器的地址	-	整数型；关联存储器支持的地址编号范围内的取值
<da>	目的地址（Destination Address）	-	GSM 03.40 TP-Destination-Address 中"地址一取值"字 段，字符型；将 BCD 数值(或缺省 GSM 字母格式的字符) 转换为当前选择的 TE 字符集中的字符(请参考 TS 07.07 中的+CSCS 指令)；地址类型由<toda>定义
<toda>	目的地址类型（Type of <da>）	-	整数型的 GSM 04.11 TP-Destination-Address 中的 8 位"类型一地址"字段(当<da>的首字符为+(IRA 43)时， 缺省值为 145；否则缺省值为 129)
<mr>	消息参考（MessageReference）	0~255	整数型的 GSM 03.40 TP-Message-Reference

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CMGF=1	设置为TEXT模式
←	OK	
→	AT+CSCS="GSM"	或AT+CSCS="IRA"
←	OK	
→	AT+CMSS=6	发送+CMGW实例中被+CMGW存储起来的编号为6的短信，接收方的号码仍为+CMGW存入的号码
←	+CMSS: 11 OK	发送成功，<mr>=11
→	AT+CMSS=7,"13192310560"	发送+CMGW实例中被+CMGW存储起来的编号为7的中文短信，并且改变接收方号码为13192310560
←	+CMSS: 12	发送成功

	OK	

7.13 短信链路控制命令：AT+CMMS

SET 命令控制短信中继协议（RP）层的链接的连续性。当使能了该特性，并且网络也支持时，发送连续多条短信时 RP 层的链接将一直存在，而无需在每两条短信之间再进行拆链和建链的过程，从而发送的速度将快很多。

READ 命令返回当前的参数值。

TEST 命令返回命令支持的参数值。

命令类型	语法	返回和说明
设置命令	AT+CMMS= [<n>]	OK
查询命令	AT+CMMS?	+CMMS: <n> OK
测试命令	AT+CMMS=?	+CMMS: (<n>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<n>	状态	0	禁能
		1	使能一次。当前一条发送短信的返回和后一条发送短信的间隔小于1-5秒（具体的取值取决于MS的实现），RP链接保持不断。当超过了1-5秒时，RP链接中断，并且<n>值将自动设置回0，即不再继续使能。
		2	一直使能。当前一条发送短信的返回和后一条发送短信的间隔小于1-5秒（具体的取值取决于MS的实现），RP链接保持不断。当超过了1-5秒时，RP链接中断，并且<n>值仍保持为2，即继续使能

7.14 读短信：AT+CMGR

使用设置指令，可将消息存储器<mem1>中，索引为<index>的消息返回到TE。若该消息处于"已接收未读"状态，则将其状态变为"已接收已读"。

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CMGR=<index>	PDU模式下（AT+CMGF=0），返回：

		+CMGR:<stat>,[<alpha>],<length><CR><LF><pdu> OK
		如果是TEXT模式（AT+CMGF=1）： 对于SMS-DELIVER： +CMGR:<stat>,<oa>,[<alpha>],<scts>[,*<tooa>,<fo>,<pid>,<dcs>,<sca>,<tosca>,<length>*]<CR><LF><data> OK 对于SMS-SUBMIT： +CMGR:<stat>,<da>,[<alpha>][,*<toda>,<fo>,<pid>,<dcs>,[<vp>],<sca>,<tosca>,<length>*]<CR><LF><data> OK 对于SMS-STATUS-REPORT： +CMGR:<stat>,<fo>,<mr>,[<ra>],[<tora>],<scts>,<dt>,<st> OK 对于SMS-COMMAND： +CMGR:<stat>,<fo>,<ct>[,*<pid>,[<mn>],[<da>],[<toda>],<length><CR><LF><cdata>*] OK 注:以上斜体字是否显示由+CSDH的设置决定
测试命令	AT+CMGR=?	OK

参数定义：

参数	定义	取值	对取值的说明
<da>，<oa>			请参考AT+CMGW条目
<toda>,<tooa>			
<length>			
<stat>			
<alpha>	MT 电话簿记录对应<da>或<oa>的显示		字符型
<pid>	Protocol Identification		请参考AT+CSMP条目
<fo>	PDU短信首字节		
<vp>	Valid Period		

<dcs>	Data Coding System		
<scts>	短信中心时间戳（Short Message Center Time Stamp）		时间-字符串型GSM 03.40 TP-Service-Centre-Time-Stamp
<dt>	Discharge time		时间-字符串型GSM 03.40 TP-Discharge-Time，与<st>成对出现
<st>	Status		整数型GSM 03.40 TP-Status 描述上一个已经发送的MO短信的状态
<ct>	Command Type		整数型GSM 03.40 TP-Command-Type，缺省为0
<ra>	接收地址		字符串型的GSM 03.40 TP-Recipient-Address 地址-取值字段
<cdata>	TEXT 模式下SMS-COMMAND的返回		TP-Command-Data(GSM 03.40)
<mr>	消息参考（MessageReference）		TP-Message-Reference(GSM 03.40)，整数型
<mn>	消息序号		TP-Message-Number(GSM 03.40)，整数型

举例：

命令（→）/返回（←）	实例	解释和说明
用TEXT模式读取短信：		
→	AT+CMGF=1	设置为TEXT模式
←	OK	
→	AT+CSCS="GSM"	+CSCS命令决定了读取的短信内容的编码方式
←	OK	
→	AT+CMGR=6	读取index=6的英文短信
←	+CMGR: "REC READ","+86139*******9","12/03/30,20:40:31+32" HI! OK	这个英文短信的内容为"HI"
→	AT+CMGR=1	Air720S系列模块读一个内容有中文的短信
←	+CMGR: "REC READ","002B00380036003100330031003600320033003100300032003600330",,"13/01/06,10:11:47+32" 611F8C2200310032 OK	注： TEXT模式下Air720S系列模块收到带有中文的短信，显示的是短信内容的UCS2码的可见字符形式 例如本例是"感谢12"的UCS2码的可见字符形式 TEXT模式下如果Air720S系列模块收到的是不含中文的短信，则直接显示内容

→	AT+CMGR=2	Air720U系列模块读一个内容有中文的短信
←	+CMGR: "REC READ","002B00380036003100330031003600320033003100300032003600330033",,"13/01/06,10:11:47+32" 感谢34 OK	注： Air720U系列模块一个内容有中文的短信，是用的GB2312编码
用PDU模式读取一个短信：		
→	AT+CMGF=0	设置为PDU模式
←	OK	
→	AT+CMGR=9	读取index=9的短信
←	+CMGR: 0,,24 0891683108200105F0240D91683161450179F900082180904121102304611F8C22 OK	

7.15 列举短消息：AT+CMGL

使用设置指令，可将查询优选消息存储器<mem1>中，状态值为<stat>的消息显示在 TE 中。若该消息处于"已接收未读"状态，则将其状态变为"已接收已读"。

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CMGL=<stat>	如果是PDU模式（AT+CMGF=0），则<stat>取值如下： 0已接收的未读消息 1 已接收的已读消息 2 已存储的未发送短信 3 已存储的已发送短信 4 所有短信 且返回如下： +CMGL:<index>,<stat>,[<alpha>],<length><CR><LF><pdu><CR><LF>+CMGL:<index>,<stat>,[<alpha>],<length><CR><LF><pdu>[...]] OK
		如果是TEXT模式（AT+CMGF=1），则<stat>取值如下： "REC UNREAD" 已接收的未读消息 "REC READ" 已接收的已读消息 "STO UNSENT" 已存储的未发送短信

		"STO SENT" 已存储的已发送短信 "ALL" 所有短信 **注意：对于以上取值，所有字母要大写。双引号可加可不加。** 对于SMS-DELIVER或SMS-SUBMIT，则返回（注：斜体字是否显示由+CSDH的设置决定）： +CMGL:<index>,<stat>,<oa/da>,[<alpha>],[<scts>][,<*tooa/toda*>,<*length*>]<CR><LF><data>[<CR><LF>+CMGL:<index>,<stat>,<da/oa>,[<alpha>],[<scts>][,<*tooa/toda*>,<*length*>]<CR><LF><data>[...]] OK 对于SMS-STATUS-REPORT，则返回： +CMGL:<index>,<stat>,<fo>,<mr>,[<ra>],[<tora>],<scts>,<dt>,<st>[<CR><LF>+CMGL:<index>,<stat>,<fo>,<mr>,[<ra>],[<tora>],<scts>,<dt>,<st>[...]] OK 对于SMS-COMMAND，则返回： +CMGL:<index>,<stat>,<fo>,<ct>[<CR><LF>+CMGL:<index>,<stat>,<fo>,<ct>[...]] OK
测试命令	AT+CMGL=?	+CMGL: (<stat>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
			本词条的所有参数在以前的命令中都有详细叙述，这里不再赘述

举例：

命令（→）/返回（←）	实例	解释和说明
在TEXT模式下列举短信：		
→	AT+CMGF=1	设置为TEXT模式
←	OK	
→	AT+CMGL=?	查询<stat>取值列表
←	+CMGL: "REC UNREAD","REC READ","STO UNSENT","STO SENT","ALL"	

	OK	
→	AT+CMGL="ALL"	查询所有的短信（注意：ALL必须为大写）
	+CMGL: 8,"REC READ","+8613162310263",,"12/08/08,10:43:04 +32" hi +CMGL: 9,"REC READ","+8613162310263",,"12/08/09,14:12:01 +32" aŒ" OK	所有短信 Index=1~7被删掉了而已。
在PDU模式下列举短信：		
→	AT+CMGF=0	设置为PDU模式
←	OK	
→	AT+CMGL=?	
←	+CMGL: (0-4) OK	
→	AT+CMGL=4	查询所有的短信
←	+CMGL: 8,1,,24 0891683108200105F0240D91683161320162F3 000021808001344023 04D7A2930A +CMGL: 9,1,,24 0891683108200105F0240D91683161320162F3 000821809041211023 04611F8C22 OK	显示查询结果

7.16 删除短消息：AT+CMGD

使用设置指令，可删除优选消息存储器<mem1>中，位置号码参数为<index>的消息。

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CMGD=<index>	返回： OK 说明：这个设置命令是删除<mem>1中索引为index的短信

	AT+CMGD=<index>,<delflag>	返回： OK 说明：这个设置命令是删除所有状态为<delflag>的短信。当<delflag>不等于0时，<index>参数被忽略
测试命令	AT+CMGD=?	+CMGD：(<index>取值列表),(<delflag>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<index>	短信位置索引	-	整数型；关联存储器支持的地址编号范围内的取值
<delflag>	删除类型，整数型。当<delflag>=1，2，3，4时，<index>参数将被忽略	0	删除指定位置号码为<index>的短消息
		1	删除优选存储器其中所有已读的短消息，保留未读短消息和已存储的MO短消息（无论是否发送） MO:Mobile Originated
		2	删除优选存储器中所有已读的消息和已发送的MO短消息，保留未读的以及未发送的已存储MO短消息
		3	删除优选存储器中所有已读的短消息，已发送和未发送的已存储MO短消息，保留未读的短消息
		4	删除优选存储器中包括未读在内所有的短消息

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CPMS="SM"	设置存储区为SIM
←	OK	
→	AT+CMGD=1	删除掉index=1的短信
←	OK	
→	AT+CMGD=1,4	所有的短信都被删除掉
←	OK	

7.17 短信业务失败结果码：CMS ERROR：<err>

短消息业务失败结果编码描述一个移动设备或网络的错误。其作用与错误结果编码类似。该编码常在一个指令失败时出现。返回的结果码为：+CMS ERROR: <err>

数字型<err>取值	冗长方式的<err>取值	解释

1	Unassigned(unallocated) number	
3	No route to destination	
6	Channel unacceptable	
8	Operator determined barring	
10	Call barred	
11	Reserved	
16	Normal call clearing	
17	User busy	
18	No user reponding	
19	User alerting,no answer	
21	Short message transfer rejected	
22	Number changed	
25	Pre-emption	
26	Non-selected user clearing	
27	Destionation out of service	
28	Invalid number format (incomplete number)	
29	Facility rejected	
30	Response to STATUS ENQUIRY	
32	Normal,unspecified	
34	No circuit/channel available	
38	Network out of order	
41	Temporary failure	
42	Switching equipment Congestion	
43	Access information discarded	
44	Requested circuit/channel not available	
47	Resources unavailable, unspecified	
49	Quality of service unavailable	
50	Requested facility not subscribed	
55	Requested facility not subscribed	
57	Bearer capability not authorized	
58	Bearer capability not presently available	
63	Service or option not available, unspecified	
65	Bearer service not implemented	
68	ACM equal or greater than ACM maximum	
69	Requested facility not implemented	
70	Only restricted digital information bearer capability is available	
79	Service or option not implemented, unspecified	
81	Invalid transaction identifier value	
87	User not member of CUG	
88	Incompatible destination	
91	Invalid transit network selection	
95	Semantically mandatory information	
96	Invalid mandatory information	
97	Message type non-existent or not implemented	

98	Message type not compatible with protocol state	
99	Information element non-existent or not implemented	
100	Conditional information element error	
101	Message not compatible with protocol	
102	Recovery on timer expiry	
111	Protocol error, unspecified	
127	Interworking, unspecified	
128	Telematic interworking not supported	
129	Short message Type 0 not supported	
130	Cannot replace short message	
143	Unspecified TP-PID error	
144	Data coding scheme (alphabet) not supported	
145	Message class not supported	
159	Unspecified TP-DCS error	
160	Command cannot be acted	
161	Command unsupported	
175	Unspecified TP-Command error	
176	TPDU not supported	
192	SC busy	
193	No SC subscription	
194	SC system failure	
195	Invalid SME address	
196	Destination SME barred	
197	SM Rejected-Duplicate SM	
198	TP-VPF not supported	
199	TP-VP not supported	
208	SIM SMS storage full	
209	No SMS storage capability in SIM	
210	Error in MS	
211	Memory Capacity Exceeded	
212	SIM Application Toolkit Busy	
213	SIM data download error	
224	CP retry exceed	
225	RP trim timeout	
226	SMS connection broken	
255	Unspecified error cause	
300	ME failure	ME 错误
301	SMS service of ME reserved	预留 ME 的 SMS 业务
302	operation not allowed	操作不允
303	operation not supported	操作不支持
304	invalid PDU mode parameter	PDU 模式下无效的参数
305	invalid text mode parameter	TEXT 模式下无效的参数
310	(U)SIM not inserted	SIM 卡未插入
311	(U)SIM PIN required	需要 SIM 卡的 PIN

312	PH-(U)SIM PIN required	需要 PH-SIM 卡的 PIN
313	(U)SIM failure	SIM 卡故障
314	(U)SIM busy	SIM 卡遇忙
315	(U)SIM wrong	SIM 错误
316	(U)SIM PUK required	需要 SIM 卡的 PUK
317	(U)SIM PIN2 required	需要 SIM 卡的 PIN2
318	(U)SIM PUK2 required	需要 SIM 卡的 PUK2
320	memory failure	存错错误
321	invalid memory index	无效的存贮索引
322	memory full	存储满
330	SMSC address unknown	短信中心号码未知
331	no network service	无网络服务
332	network timeout	网络超时
340	no +CNMA acknowledgement expected	无预期的+CNMA 确认
500	unknown error	未知错误
512	USER ABORT	（适用于cat1 模块）
513	UNABLE TO STORE	（适用于cat1 模块）
514	INVALID STATUS	（适用于cat1 模块）
515	INVALID ADDR CHAR	（适用于cat1 模块）
516	INVALID LEN	（适用于cat1 模块）
517	INVALID PDU CHAR	（适用于cat1 模块）
518	INVALID PARA	（适用于cat1 模块）
519	INVALID LEN OR CHAR	（适用于cat1 模块）
520	INVALID TXT CHAR	（适用于cat1 模块）
521	TIMER EXPIRED	（适用于cat1 模块）
528		PDU 中无效（非16 进制）字符
529		PDU 长度不正确
530	SMS SEND FAIL	（适用于cat1 模块）
531		根据制造商不同而变化
532		地址中有无效(非16 进制)字符
533		无效地址
534		PDU 长度（UDL）不正确
536		SCA 长度不正确
537		无效的第一个8 位字节(应为2 或34)
538		无效的命令类型
539		SRR 位未设置
540		SRR 设置
604	unspecified parsing error	未指定的解析错误

8 分组域相关命令

8.1 GPRS 网络注册状态：AT+CGREG

设置指令控制关于GPRS注册状态非请求结果码的显示。

当<n>=1 并且 MT 的 GPRS 注册状态发生改变，即会有+CGREG:<stat>的 URC 上报。

当 <n>=2 并且或 GPRS 注册状态发生变化或注册小区发生改变，会有：+CGREG: <stat>[,<lac>,<ci>,<act>,<rac>]的 URC 上报。

当<n>=3 并且或GPRS注册状态发生变化或注册小区发生改变，会有：+CGREG: <stat>[,<lac>,<ci>,<act>,<rac>[,<cause_type>,<reject_cause>]]的URC上报。

查询指令返回结果码的显示形式<n>和一个可以表明MT网络注册状态的参数<stat>。仅当<n>=2且MT在网络中注册后，才返回位置信息要素<lac>和<ci>。

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CGREG=[<n>]	OK
查询命令	AT+CGREG?	+CGREG:<n>,<stat>[,<lac>,<ci>] OK
测试命令	AT+CGREG=?	+CGREG:(<n>取值列表) OK
URC 上报	+CGREG: <stat>	如果设置<n>=1，当网络注册状态发生改变时，会有这样的URC（unsolicited result code）
	+CGREG: <stat>[<lac>,<ci>,<act>,<rac>]	如果设置<n>=2，当网络注册状态或注册的小区(ci)发生改变时，会有这样的URC上报
	+CGREG: <stat>[,<lac>,<ci>,<act>,<rac>[,<cause_type>,<reject_cause>]]	如果设置<n>=3，当网络注册状态或注册的小区(ci)发生改变时，会有这样的URC上报
	+CEREG: <stat>[,[<tac>],[<ci>],[<AcT>][,,[,[<Active Time>],[<Periodic-TAU>]]]]	如果设置<n>=4，当网络注册状态或注册的小区(ci)发生改变时，会有这样的URC上报
	+CEREG:<stat>[,[<tac>],[<ci>],[<AcT>][,[<cause_type>],[<reject_cause>][,[<Active-Time>],[<Periodic-TAU>]]]]	如果设置<n>=5，当网络注册状态或注册的小区(ci)发生改变时，会有这样的URC上报

参数定义：

参数	定义	取值	对取值的说明

<n>	上报状态	0	禁用网络注册非请求结果码+CGREG:
		1	启用网络注册非请求结果码+CGREG: <stat>
		2	启用网络注册和位置信息非请求结果码 +CGREG:<stat>[,<lac>,<ci>,<act>,<rac>]
		3	启用网络注册和位置信息非请求结果码 +CGREG: <stat>[,<lac>,<ci>,<act>,<rac>[,<cause_type>,<reject_cause>]]
		4	启用网络注册和位置信息非请求结果码 +CEREG: <stat>[,[<tac>],[<ci>],[<AcT>][,,[,[<Active Time>],[<Periodic-TAU>]]]]
		5	启用网络注册和位置信息非请求结果码 +CEREG:<stat>[,[<tac>],[<ci>],[<AcT>][,[<cause_type>],[<reject_cause>][,[<Active-Time>],[<Periodic-TAU>]]]]
<stat>	当前网络注册状态	0	未注册；ME 当前没有搜索注册业务的新运营商
		1	已注册，本地网
		2	未注册，但 ME 正在搜索注册业务的新运营商，但是当前没有可用的公共陆地移动网络（PLMN），一旦PLMN有效，UE将开始GPRS附着。
		3	注册被拒绝。GPRS业务被禁用，即使用户请求，UE也不被允许附着GPRS网络
		4	未知
		5	已注册，漫游
		6	注册归属地"SMS only"业务
		7	注册漫游地"SMS only"业务
		8	仅附着紧急承载业务(See NOTE 2)
		9	注册归属地"CSFB not preferred"业务
		10	注册漫游地"CSFB not preferred"业务
		11	仅紧急业务可用
<lac>	位置区号(Location Area Code)	-	字符型；2 字节十六进制位置区代码(比如：00C3相当于十进制中的195)
<ci>	小区号(Cell Id)	-	字符型；2 字节十六进制小区编号
<rac>	Routing area code		
<cause_type>	定义同+CEREG		
<reject_cause>	定义同+CEREG		

举例：

命令（→）/	实例	解释和说明

返回（←）		
→	AT+CGREG?	查询当前GPRS注册状态
←	+CGREG: 0,1 OK	<n>=0,<stat>=1
→	AT+CGREG=1	设置<n>=1，当模块注册状态发生改变时，会有一个URC上报+CGREG: <stat>
←（URC）	+CGREG: 1	当模块注册状态发生改变
→	AT+CGREG=2	设置<n>=2，当模块注册状态发生改变时，会有一个URC上报+CGREG: <stat>[,<lac>,<ci>,<act>]
←（URC）	+CGREG: 5,"18be","9363",7	当模块注册状态发生改变，或移动模块，使模块所处小区号发生改变的时候，会有个URC上报上来
→	AT+CGREG?	查询注册状态
←	+CGREG: 2,1,"1863","00a2c315",7 OK	

8.2GPRS 附着分离：AT+CGATT

设置命令用于将 MT 附着 GPRS 业务，或将 MT 从 GPRS 业务分离。

查询命令返回当前的 GPRS 附着状态。

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CGATT=<state>	OK
查询命令	AT+CGATT?	+CGATT: <state> OK
测试命令	AT+CGATT=?	+CGATT: (<state>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<n>	GPRS 附着状态	0	分离
		1	附着

举例：

命令（→）/	实例	解释和说明

返回（←）		
→	AT+CGATT?	查看当前GPRS附着状态
←	+CGATT: 1 OK	<state>=1，标明当前GPRS已经附着
→	AT+CGATT=?	查看<state>的取值范围
←	+CGATT: (0-1) OK	查询结果

8.3PDP 上下文定义：AT+CGDCONT

设置命令设置以 cid 为标志的 PDP 上下文参数。

查询命令查询所有的 PDP 上下文定义。

语法规则：

命令类型	语法	返回
设置命令	AT+CGDCONT=<cid>[,<PDP_type>[,<APN>[,<PDP_addr>[,<d_comp>[,<h_comp>]]]]]	OK
查询命令	AT+CGDCONT?	[+CGDCONT:<cid>,<PDP_type>,<APN>,<PDP_addr>,<d_comp>,<h_comp>[,<pd1>[,…[,pdN]]][<CR><LF>+CGDCONT:<cid>,<PDP_type>,<APN>,<PDP_addr>,<d_comp>,<h_comp>[,<pd1>[,…[,pdN]]] […]]] OK
测试命令	AT+CGDCONT=?	+CGDCONT: (<cid>取值列表),<PDP_type>,,,(<d_comp>取值列表),(<h_comp>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<cid>	PDP 上下文标识，用于标识一个 PDP 上下文定义	1~15	整数型。该参数对TE-MT接口而言是本地参数，并且可用于其他PDP上下文相关指令
<PDP_type>	分组数据协议类型；字符型	"IP"	支持"IP"互联网协议 IP(Internet Protocol)(IETF STD5)，默认值
		"IPV6"	Internet Protocol, version 6 (IETF RFC 2460)

		"PPP"	Point to Point Protocol (IETF STD 51)
<APN>	接入点名称（Access Point Name）		字符串型，用于选择GGSN或外部分组数据网络的逻辑名称。若该参数取值为空或省略，则需要请求签约值。
<PDP_address>	分组数据协议地址		字符型，这是IP协议地址，格式为："<n>.<n>.<n>.<n>"其中<n>=0~255；用于标识对于特定PDP上下文，MT分配的地址空间。若该参数为空或等于"0.0.0.0 "，MT将被要求分配一个动态地址。使用+CGPADDR命令可读出此分配地址。
<d_comp>	用于控制PDP 数据压缩。仅适用于SNDCP。	0	关闭PDP数据压缩(默认值)
		1	打开PDP数据头压缩（由厂商定义压缩方式，暂不支持！）
		2	V.42bis
<h_comp>	控制 PDP 头压缩。 数字型参数	0	关闭PDP头压缩(默认值)
		1	RFC1144（仅适用于SNDCP）
		2	RFC2507（暂不支持）
<pd1>… <pdN>	与 <PDP_type> 相关的参数		字符串型

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CGDCONT?	开机注册后查询当前PDP上下文
←	+CGDCONT: 1,"IP","xz01.njm2mapn","10.173.90.29",0,0 OK	缺省已经有一个PDP上下文，这个PDP上下文，用于模块RNDIS网卡功能以及TCPIP/HTTP/MQTT/FTP等数据业务。
→	AT+CGDCONT=?	查询参数取值范围
←	+CGDCONT: (1-15),"IP",,,,,(0),(0,2),(0),(0),(0,1),(0),(0,1),(0) +CGDCONT: (1-15),"IPV6",,,,,(0),(0,2),(0),(0),(0,1),(0),(0),(0) +CGDCONT: (1-15),"IPV4V6",,,,,(0),(0,2),(0),(0),(0,1),(0),(0,1),(0) +CGDCONT: (1-15),"Non-IP",,,,,(0),(0,2),(0),(0),(0,1),(0),(0),(0),(0,1) OK	查询结果

8.4PDP 上下文鉴权参数：AT+CGAUTH

本命令是 AT+CGDCONT 命令的扩展，设置相关 PDP 上下文的鉴权方式。

查询命令查询所有 PDP 上下文鉴权信息。

语法规则：

命令类型	语法	返回
设置命令	AT+CGAUTH=<cid>[,<auth_prot>[,<userid>[,<password>]]]	OK
查询命令	AT+CGAUTH?	[+CGAUTH: <cid>,<auth_prot>,<userid>,<password>] [<CR><LF>+CGAUTH: <cid>,<auth_prot>,<userid>,<password> [. . .]]
测试命令	AT+CGAUTH=?	+CGAUTH: (range of supported <cid>s), (list of supported <auth_prot>s), (range of supported <userid>s), (range of supported <password>s)

参数定义：

参数	定义	取值	对取值的说明
<cid>	PDP 上下文标识，用于标识一个 PDP 上下文定义		整数型。该参数对TE-MT接口而言是本地参数，并且可用于其他PDP上下文相关指令
<auth_prot>	鉴权类型	0	None
		1	PAP
		2	CHAP
<userid>	用户名		
<password>	密码		

8.5显示 PDP 地址：AT+CGPADDR

使用设置命令，返回的是<cid>所标识的 PDP 地址；

使用该执行命令，可返回指定上下文标识<cid>的 PDP 地址列表；

当 PDP 上下文未建立的时候，使用该命令是无法查询和显示 PDP 地址的。

语法规则：

命令类型	语法	返回
设置命令	AT+CGPADDR=<cid>	+CGPADDR:<cid>,<PDP_ipv4_addr>[,<PDP_ipv6_addr>] OK
执行命令	AT+CGPADDR	+CGPADDR:<cid>,<PDP_ipv4_addr>[,<PDP_ipv6_addr>][<CR><LF>+CGPADDR:<cid>,<PDP_ipv4_addr>[,<PDP_ipv6_ad

		dr>][...]] OK
测试命令	AT+CGPADDR=?	+CGPADDR: (<cid>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<cid>	PDP 上下文标识	1~8	整数型。该参数对TE-MT接口而言是本地参数，并且可用于其他PDP上下文相关指令
<PDP_PDP_ipv4_addr>	分组数据协议地址		字符型，这是IP协议地址，格式为： "<n>.<n>.<n>.<n>"其中<n>=0~255；用于标识对于特定PDP上下文，MT分配的地址空间。
<PDP_PDP_ipv6_addr>	分组数据协议地址		字符型，这是IP协议地址，格式为："<n>.<n>.<n>.<n>.<n>.<n>.<n>.<n>.<n>.<n>.<n>.<n>.<n>.<n>.<n>.<n>"其中<n>=0~255；用于标识对于特定PDP上下文，MT分配的地址空间。

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CGACT?	查询当前的PDP上下文激活状态
←	+CGACT: 1,1 OK	<state>=1，表明<cid>=1的PDP上下文处于已经激活状态
→	AT+CGPADDR=1	查询<cid>=1的PDP上下文对应的PDP地址（即IP地址）
←	+CGPADDR: 1,"10.162.229.248" OK	<cid>=1 PDP上下文分配的IP地址(普通物联网络)
←	+CGPADDR: 1,"10.143.27.227","36.14.4.108.140.129.135.84.0.1.0.1.82.198.143.152" OK	<cid>=1 PDP上下文分配的IP地址(支持IPV6的卡)

8.6PDP 上下文激活：AT+CGACT

使用执行指令，可激活或去激活指定的 PDP上下文。该指令成功执行后，MT保持 V.250ter指令状态。若 PDP上下文已处于所请求状态，则该状态保持不变。

当执行该指令的激活形式时，若MT没有附着 GPRS，MT首先进行 GPRS 附着，然后尝试激活指定的上下文。若没指定<cid>，则指令的激活形式激活所有定义的上下文。若没指定<cid>，则指令的失效形式使所有激活的上下文失效。

查询命令返回所有的已定义的PDP上下文的信息。

语法规则：

命令类型	语法	返回
设置命令	AT+CGACT=<state>,<cid>	OK
查询命令	AT+CGACT?	+CGACT:<cid>,<state>[<CR><LF>+CGACT:<cid>,<state>[...]] OK
测试命令	AT+CGACT=?	+CGACT: (<state>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<state>	PDP 上下文激活状态	0	未激活
		1	激活
<cid>	PDP 上下文标识		整数型。该参数对TE-MT接口而言是本地参数，并且可用于其他PDP上下文相关指令。

举例：

命令（→）/返回（←）	实例	解释和说明
		具体实例请参考+CGPADDR条目的实例

8.7控制非请求 GPRS 事件上报：AT+CGEREP

设置命令启用或禁用 URC 提示+CGEV: XXX。当设置为启用时，当数据域终端或网络侧发生某些事件，会有此 URC 上报。

语法规则：

命令类型	语法	返回
设置命令	AT+CGEREP=<mode>[,<bfr>]	OK

查询命令	AT+CGEREP?	+CGEREP:<mode>,<bfr> OK
测试命令	AT+CGEREP=?	+CGEREP:(<mode>取值列表), (<bfr>取值列表) OK
URC 上报	+CGEV: XXX	如果<mode>=1，当MT或网络侧发生某些事情时，会有此URC上报

参数定义：

参数	定义	取值	对取值的说明
<mode>	结果码上报的模式	0	缓冲MT中的非请求结果码；如果MT结果码缓冲器已满，则丢弃最旧的结果码。不转发结果码给TE。
		1	当MT-TE链路处于保留状态时（比如处于在线数据状态），丢弃非请求结果码；否则MT直接将非请求结果码转发给TE。

8.8(URC)分组域事件 URC 上报：+CGEV

当分组域或网络侧有事件上报时，会有此打印主动上报。

语法规则：

URC
+CGEV: XXX

事件上报：

事件	解释
+CGEV: NW PDN DEACT <cid>	网络迫使上下文失效。
+CGEV: ME PDN DEACT <cid> [,< pdnReason >]	MT迫使上下文失效。
+CGEV: ME PDN ACT <cid> [,< pdnReason >]	ME 激活了上下文。
+CGEV: NW MODIFY <cid>,<change_reason>,<event_type>	网络修改了上下文。
+CGEV: ME MODIFY <cid>,<change_reason>,<event_type>	所述 MT 修改了上下文。

参数定义：

参数	取值	对取值的说明
<cid>		PDP 上下文标识

<pdnReason>	0	仅允许 IPV4
	1	仅允许 IPV6
	2	仅允许单个地址承载
	3	仅允许单个地址承载，而激活的第二承载失败
<change_reason>(整型，位对应某种改变)	Bit 1	TFT 改变
	Bit 2	Qos 改变
	Bit 3	WLAN Offload 改变
<event_type>		指示这是一个通知事件还是 TE 必须回复已收到(目前不支持)
<PDN>		上报里有 PDN 表示该激活或去激活的上下文是与 LTE 里 PDN 连接关联或者是与 GMS/UMTS 里的主要 PDP 上下文相关联

举例：

命令（→）/返回（←）	实例	解释和说明
←	+CGEV: ME PDN ACT 1,0	
←	+CGEV: NW PDN DEACT 1	
←	+CGEV: ME DETACH	

8.9通过 cid 查询本地 IP 地址：AT*GETIP

语法规则：

命令类型	语法	返回
设置命令	AT*GETIP=<cid>	+GETIP: <ip>,<gateway> OK

参数定义：

参数	定义	取值	对取值的说明
<cid>	承载场景 id	1	PDP 上下文标识

9 NTP 相关命令

Network Time Protocol（NTP）是用来使计算机时间同步化的一种协议，它可以使计算机对其服务器或时钟源（如石英钟，GPS 等等)做同步化，它可以提供高精准度的时间校正（LAN 上与标准间差小于 1 毫秒，WAN 上几十毫秒），且可介由加密确认的方式来防止恶毒的协议攻

击。时间按 NTP 服务器的等级传播。按照离外部 UTC 源的远近把所有服务器归入不同的 Stratum（层）中。

模块支持 SNTP 协议（Simple Network Time Protocol），并有一套 AT 命令实现网络时间同步功能。

9.1 设置 GPRS 承载场景 ID：AT+CNTPCID

命令类型	语法	返回
设置命令	AT+CNTPCID=<cid>	OK
查询命令	AT+CNTPCID?	+CNTPCID:<cid> OK
测试命令	AT+CNTPCID=?	+CNTPCID:(<cid>取值范围) OK

参数定义：

参数	定义	取值	对取值的说明
<cid>	承载场景 id	1-3	取值同+SAPBR 命令的<cid>

9.2 同步网络时间：AT+CNTP

命令类型	语法	返回
设置命令	AT+CNTP=<NTP server>[,<time zone>]	OK
执行命令	AT+CNTP	OK +CNTP: <code>
查询命令	AT+CNTP?	+CNTP: <NTP server>,<time zone>，<time out> OK
测试命令	AT+CNTP=?	OK
注意事项	网络时间同步成功后，可以用AT+CCLK?命令来查询模块当前时间	

参数定义：

参数	定义	取值	对取值的说明

<NTP server>	NTP 服务器	域名或ip地址	
<time zone>	本地时区	-47~+48	单位：1/4 时区。时区是-12~+12，但是有的国家会用半时区，甚至 1/4 时区，为兼顾这些国家，时区单位为：1/4 时区。 负号表示是西时区，正号表示为东时区
<code>	操作码	1	网络时间同步成功
		61	网络错误
		62	DNS 解析错误
		63	连接错误
		64	服务响应错误
		65	服务响应超时
<time out>	超时时间		单位为秒，若未设置默认 5 秒

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+SAPBR=3,1,"Contype","GPRS"	激活PDP是使用CNTP命令同步时间的前提
←	OK	
→	AT+SAPBR=3,1,"APN",""	设置PDP承载之APN参数 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用（此<apn>可以通过AT+CGDCONT?来查询），所以输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN
←	OK	
→	AT+SAPBR=1,1	激活<cid>=1的PDP
←	OK	
→	AT+CNTPCID=1	设置使用的PDP的<cid>=1
←	OK	
→	AT+CNTP	
←	OK +CNTP:1	
→	AT+CCLK?	
←	+CCLK: "18/05/16,15:49:28+32" OK	

11文件系统读写命令

11.1 创建一个文件：AT+FSCREATE

语法规则：

命令类型	语法	返回
设置命令	AT+FSCREATE=<filename>	OK 或 ERROR
查询命令	AT+FSCREATE=?	OK 或 ERROR

参数定义：

参数	定义	取值	对取值的说明
<filename>	文件名		字符串型，不用加双引号，不超过 64 个字节

11.2 读文件：AT+FSREAD

语法规则：

命令类型	语法	返回
设置命令	AT+FSREAD=<filename>,<mode>,<filesize>,<position>	<data> OK 或 ERROR
查询命令	AT+FSREAD=?	OK 或 ERROR

参数定义：

参数	定义	取值	对取值的说明
<filename>	文件名		字符串型，不用加双引号，不超过 64 个字节
<mode>	读文件的模式	0	从文件开头读文件
		1	从文件的<position>位置开始读文件
<filesize>	要读取的文件的长度		不能大于 10240 个字节。如果文件的实际长度小于该值，则读取的是文件的实际长度。
<position>	读文件的起始位置		该取值应小于文件的大小。当<mode>=0 时，该值无效

<data>	读取的文件数据		

11.3 读文件的时候是否返回头：AT+FSREADHEAD

注：本命令 EC618 系列模块>=V1118 版本支持。

发送 AT+FSREADHEAD=1 后，AT+FSREAD 读文件时会多出头+FSREAD:

语法规则：

命令类型	语法	返回
设置命令	AT+FSREADHEAD=<mode>	OK 或 ERROR
查询命令	AT+FSREADHEAD?	+FSREADHEAD: <mode> OK 或 ERROR

参数定义：

参数	定义	取值	对取值的说明
<mode>	读文件时是否返回头信息	0	不显示+FSREAD：头信息
		1	显示+FSREAD：头信息

11.4 写文件：AT+FSWRITE

语法规则：

命令类型	语法	返回
设置命令	AT+FSWRITE=<filename>,<mode>,<filesize>,<inputtime>	> （出现>后写入数据） OK 或 ERROR 或 TimeOut
查询命令	AT+FSWRITE=?	OK 或 ERROR

注意事项	当输入数据的长度达到<filesize>时，自动写入。 或输入数据的时间超过<inputtime>时，返回TimeOut

参数定义：

参数	定义	取值	对取值的说明
<filename>	文件名		字符串型，不用加双引号，不超过 64 个字节
<mode>	写文件的模式	0	从文件的开头开始写文件
		1	从文件的末尾开始写文件
<filesize>	要写的数据的长度		不能大于 10240 个字节
<inputtime>	输入持续的时间	1~4294967295	单位：秒

11.5 获取盘符：AT+FSDRIVE

语法规则：

命令类型	语法	返回
执行命令	AT+FSDRIVE=<n>	+FSDRIVE:<drive> OK
查询命令	AT+FSDRIVE=?	OK

参数定义：

参数	定义	取值	对取值的说明
<n>	盘符类型	0	本地盘符
<drive>	盘符		字符串型。加不加双引号皆可. 如果是本地存储，则盘符为 C

11.6 显示文件目录列表：AT+FSLS

语法规则：

命令类型	语法	返回
设置命令	AT+FSLS=<filepath>	list of subdirectories/files OK
查询命令	AT+FSLS=?	OK
注意事项	1) 如果返回的结果最后一个字符是\，则表示为目录名，否则为文件名 2) 查询模块本地FLASH根目录使用：AT+FSLS="/"或AT+FSLS="C:/" 3) 查询外接TF卡根目录使用：AT+FSLS="D:/"	

参数定义：

参数	定义	取值	对取值的说明
<filepath>	文件路径		字符串型，不用加双引号，不超过 64 个字节。

11.7 获取可用空间大小：AT+FSMEM

执行命令用来获取文件系统剩余空间大小。

语法规则：

命令类型	语法	返回
执行命令	AT+FSMEM	+FSMEM: <size> OK
查询命令	AT+FSMEM=?	OK 或 ERROR

参数定义：

参数	定义	取值	对取值的说明
<size>	文件系统可用空间大小		单位：字节

11.8 创建目录：AT+FSMKDIR

语法规则：

命令类型	语法	返回
设置命令	AT+FSMKDIR=<dir_name>	OK 或 ERROR
查询命令	AT+FSMKDIR=?	OK 或 ERROR

参数定义：

参数	定义	取值	对取值的说明
<dir_name>	目录名		字符串型，不用加双引号，不超过 64 个字节

11.9 删除目录：AT+FSRMDIR

语法规则：

命令类型	语法	返回
设置命令	AT+FSRMDIR=<dir_name>	OK 或 ERROR
查询命令	AT+FSRMDIR=?	OK 或 ERROR

参数定义：

参数	定义	取值	对取值的说明
<dir_name>	文件名		字符串型，不用加双引号，不超过 64 个字节

11.10 删除文件：AT+FSDEL

语法规则：

命令类型	语法	返回
设置命令	AT+FSDEL=<filename>	OK 或 ERROR
查询命令	AT+FSDEL=?	OK 或 ERROR

参数定义：

参数	定义	取值	对取值的说明
<filename>	文件名		字符串型，不用加双引号，不超过 64 个字节

11.11 获取文件大小：AT+FSFLSIZE

语法规则：

命令类型	语法	返回
设置命令	AT+FSFLSIZE=<filename>	+FSFLSIZE: <size> OK

查询命令	AT+FSFLSIZE=?	OK

参数定义：

参数	定义	取值	对取值的说明
<filename>	文件名		字符串型，不用加双引号，不超过 64 个字节
<size>	文件大小		单位：字节

11.12 使用方法举例

由于这部分的命令具有高度相关性，所以将各个命令的应用例程集中在一起描述。

举例：

命令（→）/ 返回（←）	实例	解释和说明
本地文件系统读写命令使用方法：		
→	AT+FSMKDIR="/testdir"	创建一级目录
←	OK	
→	AT+FSMKDIR="/testdir/testdir"	创建二级目录
←	OK	
→	AT+FSCREATE="/testdir/test.txt"	一级目录下创建文件
←	OK	
→	AT+FSLS="/testdir"	显示一级目录列表
←	testdir\ test.txt OK	
→	AT+FSWRITE="/testdir/test.txt",0,10,10	写文件
→	>1234567890	当出现">"后，输入写入的数据：1234567890。
←	OK	
→	AT+FSREAD="/testdir/test.txt",0,10,10	读取文件数据
←	1234567890 OK	
→	AT+FSMEM	获取文件系统可用空间大小
←	+FSMEM: 1304000 bytes OK	
→	AT+FSFLSIZE="/testdir/test.txt"	获取文件大小
←	+FSFLSIZE: 10 OK	
→	AT+FSDEL="/testdir/test.txt"	删除文件

←	OK	
→	AT+FSRMDIR="/testdir/testdir"	删除目录
←	OK	

12嵌入式 TCPIP 命令

12.1 启动多 IP 连接：AT+CIPMUX

语法规则：

命令类型	语法	返回
设置命令	AT+CIPMUX=<n>	OK
查询命令	AT+CIPMUX?	+CIPMUX: <n> OK
测试命令	AT+CIPMUX=?	+CIPMUX: (0,1) OK
注意事项	● 只在 IP initial 状态，本命令的设置命令才能设置成功	

参数定义：

参数	定义	取值	对取值的说明
<n>	多路连接开关	0	单路连接（缺省值）
		1	多路连接

12.2 启动任务并设置接入点 APN、用户名、密码：AT+CSTT

语法规则：

命令类型	语法	返回
设置命令	AT+CSTT=<apn>[,<username>[,<password>]]	OK
查询命令	AT+CSTT?	+CSTT: <apn>,<user name>,<password> OK
测试命令	AT+CSTT=?	+CSTT: "APN","USER","PWD" OK
注意事项	1） 设置命令和执行命令只有在 IP INITIAL 状态下执行有效。执行设置命令后，状态变为 IP START 2） 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用(此<apn>可以通过AT+CGDCONT?来查询)，所以直接输入AT+CSTT即可，模块会按照自动获取的<apn>来设置CSTT	

	的APN

参数定义：

参数	定义	取值	对取值的说明
<apn>	GPRS 接入点名称	-	字符串参数(双引号可加可不加)，不超过 128 个字符
<user name>	GPRS 用户名	-	字符串参数(双引号可加可不加)，不超过 32 个字符
<password>	GPRS 密码	-	字符串参数(双引号可加可不加)，不超过 32 个字符

12.3 专网卡设置 APN、用户名、密码和鉴权方式：AT+CPNETAPN

语法规则：

命令类型	语法	返回
设置命令	AT+CPNETAPN=<mode>,<apn>,<user>,<pwd>,<authmode>	OK
查询命令	AT+CPNETAPN?	MODE:<mode> APN:<apn> USR:<user> PWD:<password> AUTHMODE:<authmode> OK
测试命令	AT+CPNETAPN=?	+CPNETAPN: setmode=[2,3],"apn","Username","Password",authtype=[0,1,2] OK

参数定义：

参数	定义	取值	对取值的说明
<mode>		2	保存并立即生效
		3	删除保存的参数
<apn>	GPRS 接入点名称		字符串参数(双引号可加可不加)，不超过 128 个字符。 注：此参数不可以为空
<user>	GPRS 用户名		字符串参数(双引号可加可不加)，不超过 32 个字符。如果为空，则表示为""
<pwd>	GPRS 密码		字符串参数(双引号可加可不加)，不超过 32 个字符。如果为空，则表示为""
<authmode>		0	无鉴权
		1	PAP
		2	CHAP

举例：

命令(→)/返回(←)	实例	解释和说明
→	AT+CPNETAPN=2,jscmiot,u9682,iot98765,2	设置专网卡的APN。各个参数请据实填写，不要照抄。设置此命令时模块会自动将AT+AUTOAPN设置为0，不用提前输入AT+AUTOAPN=0
←	OK	设置成功。设置成功后参数会保存到NV，模块会自动重启协议栈(部分卡需要手动重启协议栈)
	重启后缺省的PDP承载(<cid>=1)已经为所设置的APN，并已经处于激活可用状态。这个缺省的PDP承载，用于模块RNDIS网卡功能以及TCPIP/HTTP/MQTT/FTP等数据业务。 AT+CGDCONT? +CGDCONT: 1,"IP","jscmiot","10.67.64.153",,,,,,,,,,, OK 后面数据应用中CSTT或SAPBR设置APN的时候，只需要APN为空或设置<专网apn>即可，不需要再设置<user>和<pwd> 例如： AT+CSTT=jscmiot AT+CSTT AT+SAPBR=3,1,"APN","jscmiot" AT+SAPBR=3,1,"APN",""	

12.4 激活移动场景(或发起 GPRS 或 CSD 无线连接)：AT+CIICR

语法规则：

命令类型	语法	返回和说明
执行命令	AT+CIICR	如果激活成功，返回： OK 如果激活失败，返回： ERROR
测试命令	AT+CIICR=?	返回： OK
注意事项	● AT+CIICR 只有在IP START状态下才能激活移动场景，执行后状态变为IP CONFIG。 ● 在模块接受激活场景操作后，如果移动场景激活成功，状态变为IPGPRSACT，返回OK，否则返回ERROR。	

12.5 查询本地 IP 地址：AT+CIFSR

语法规则：

命令类型	语法	返回和说明
执行命令	AT+CIFSR	<IP address>
测试命令	AT+CIFSR=?	OK
注意事项	只有在移动场景已激活的状态：IP GPRSACT、TCP/UDPCONNECTING、CONNECT OK、IP CLOSE 下才可以通过AT+CIFSR 查询得到本地 IP 地址，否则返回 ERROR	

参数定义：

参数	定义	取值	对取值的说明
<IP address>	IP 地址	-	字符串参数(字符串需要加引号)

12.6 设置 TCP 使用 SSL 功能：AT+CIPSSL

语法规则：

命令类型	语法	返回
设置命令	AT+CIPSSL=<n>	OK
查询命令	AT+CIPSSL?	+CIPSSL: <n> OK
测试命令	AT+CIPSSL=?	+CIPSSL: (0-1) OK
注意事项	在 CIPSTART 之前输入设置命令 AT+CIPSSL=1 来打开 SSL 功能。打开 SSL 功能后，模块会在 TCP 连接建立后自动进行 SSL 验证。 当前仅支持作为 SSL Client 应用。	

参数定义：

参数	定义	取值	对取值的说明
<n>	SSL 功能开关	0	关闭 SSL
		1	打开 SSL

12.7 配置 TCP SSL 参数：AT+SSLCFG

设置命令用来设置 SSL 版本，SSL 加密算法（ciphersuites），安全等级（security level），CA 证书（Certificate Authority Certificate），客户端证书（client certificate）和客户端密钥（client key）。这些参数在 SSL 协议的握手过程中会用到。

语法规则：

命令类型	语法	返回

设置命令	AT+SSLCFG="sslversion",<n>[,<sslversion>]	**如果<sslversion>缺失，则查询<n>对应的SSL版本：** +SSLCFG: "sslversion",<n>,<sslversion> OK **否则，设置<n>对应的SSL版本：** 如果格式和参数正确，返回： OK 如果命令格式或参数错误，返回： ERROR
	AT+SSLCFG="ciphersuite",<n>[,<ciphersuites>]	**如果<ciphersuites>缺失，则查询<n>对应的加密算法：** +SSLCFG: ciphersuite",<n>,<ciphersuites> OK **否则，设置<n>对应的加密算法：** 如果格式和参数正确，返回： OK 如果目录格式或参数错误，返回： ERROR
	AT+SSLCFG="cacert",<n>[,<cacertpath>]	**如果<cacertpath>参数缺失，则查询<n>对应的CA证书路径：** +SSLCFG:"cacert",<n>,<cacertpath> OK **否则，设置<n>对应的CA证书路径：** 如果格式和参数正确，返回： OK 如果目录格式或参数错误，返回： ERROR
	AT+SSLCFG="clientcert",<n>[,<client_cert_path>]	**如果<client_cert_path>参数缺失，则是查询<n>对应的客户端证书路径：** +SSLCFG:"clientcert",<n>,<client_cert_path> OK **否则，设置<n>对应的客户端证书路径：** 如果格式和参数正确，返回： OK

		如果命令格式或参数错误，返回： ERROR
	AT+SSLCFG="clientkey",<n>[,<client_key_path>]	**如果<client_key_path>缺失，则是查询<n>对应的客户端密钥路径：** +SSLCFG:"clientkey",<n>,<client_key_path> OK **否则，设置<n>对应的客户端密钥路径：** 如果格式和参数正确，返回： OK 如果命令格式或参数错误，返回： ERROR
	AT+SSLCFG="seclevel",<n>[,<seclevel>]	**如果<seclevel>参数缺失，则是查询<n>相关的安全等级：** +SSLCFG:"seclevel",<n>,<seclevel> OK **否则，设置<n>对应的安全等级：** 如果格式和参数正确，返回： OK 如果命令格式或参数错误，返回： ERROR
	AT+SSLCFG="hostname",<n>[,<hostname>]	**如果<hostname>参数缺失，则是查询<n>相关的域名：** +SSLCFG:"hostname",<n>,<hostname> OK **否则，设置<n>对应的主机名：** 如果格式和参数正确，返回： OK 如果命令格式或参数错误，返回： ERROR
	AT+SSLCFG="ignorelocaltime",<n>[,<ignoreltime>]	**如果<ignorelocaltime>缺失，则是查询<n>相关的证书过期时间检查这项的设置：** +SSLCFG:"ignorelocaltime",<n>,<ignoreltime> OK

		否则，设置<n>对应的证书过期时间检查参数： 如果格式和参数正确，返回： OK 如果命令格式或参数错误，返回： ERROR
	AT+SSLCFG="negotiatetimeout",<n>[,<negotiate_time>]	**如果<negotiate_time>参数缺失，则是查询<n>对应的最大SSL协商时间：** +SSLCFG:"negotiatetimeout",<n>,<negotiate_time> OK **否则，设置<n>对应的最大SSL写上协商时间：** 如果格式和参数正确，返回： OK 如果命令格式或参数错误，返回： ERROR
	AT+SSLCFG="clientrandom",<n>[,<randbytes>]	**如果<randbytes>缺失，则是查询<n>相关的随机数：** +SSLCFG:"clientrandom",<n>,<randbytes> OK **否则，设置<n>对应的随机数：** 如果格式和参数正确，返回： OK 如果命令格式或参数错误，返回： ERROR
	AT+SSLCFG="premaster",<n>[,<premaster>]	**如果<premaster>缺失，则是查询<n>相关的<premaster>：** +SSLCFG:"premaster",<n>,<premaster> OK **否则，设置<n>对应的premaster：** 如果格式和参数正确，返回： OK 如果命令格式或参数错误，返回： ERROR

	AT+SSLCFG="verifymode",<n>[,<verifymode>]	**如果<verifymode>缺失，则查询<n>相关的证书验证模式，此时返回：** +SSLCFG:"verifymode",<n>,<verifymode> OK **否则，设置证书认证模式是根证书认证还是其他证书认证：** 如果格式和参数正确，返回： OK 如果命令格式或参数错误，返回： ERROR
	AT+SSLCFG="XXXXX",<n>,	擦除相应的参数。"XXXXX"是指："sslversion"，"ciphersuite"，"cacert"等关键字。 **注意：<n>后一定要有逗号，如果<n>后面没有逗号，则只是查询。**
测试命令	AT+SSLCFG=?	OK
注意事项	TCP SSL 的功能示例，请参考本章后面的例子。	

参数定义：

参数	定义	取值	对取值的说明
<n>	SSL 上下文 id	0~5,34,88,153	整数型。 TCP 功能时与 CIPSTART 中的<n>绑定。例如：当多链接中 CIPSTART 中设置链接号为 1，则 SSL 上下文 id 也为 1。 TCP 单链接中 SSL 上下文 id 固定为 0. FTP 功能时<n>=34（十进制） MQTT 功能时<n>=88（十进制） HTTP 功能时<n>=153（十进制）
<sslversion>	SSL 版本	0	SSL3.0
		1	TLS1.0
		2	TLS1.1
		3	TLS1.12
		4	ALL above
<ciphersuites>	SSL ciphersuites	0X0035	TLS_RSA_WITH_AES_256_CBC_SHA
		0X002F	TLS_RSA_WITH_AES_128_CBC_SHA
		0X0005	TLS_RSA_WITH_RC4_128_SHA
		0X0004	TLS_RSA_WITH_RC4_128_MD5
		0X000A	TLS_RSA_WITH_3DES_EDE_CBC_SHA
		0X003D	TLS_RSA_WITH_AES_256_CBC_SHA256

		0XFFFF	ALL above
<cacertpath>	被信任的 CA 证书路径		字符串型
<client_cert_path>	客户端证书路径		字符串型
<client_key_path>	客户端密钥路径		字符串型
<seclevel>	安全等级	0	No authentication
		1	服务器鉴权
		2	服务器鉴权和客户端鉴权（如果服务器要求的话）
<hostname>	主机名		
<ignoreltime>	该参数决定如何对待过期证书	0	关心证书的过期时间
		1	忽略证书的过期时间
<negotiate_time>	SSL 协商阶段的最大时间	10~300	单位：秒
<clientrandom>	随机数，十六进制数组成的字符串，支持 56 和 64 个字节	数字和字母的组合	字符串型，双引号可加可不加，数字的范围：0~9，字母的范围：ABCDEF（大小写不敏感）。例如 56 个字节的：101B12C3141516171F192021222324252627282930313233343536 37
<premaster>	premaster		
<verifymode>	证书验证模式	0	根证书认证
		1	其他证书认证

12.8 配置本地 TCP 端口：AT+CLPORT

语法规则：

命令类型	语法	返回	
设置命令	单链接： AT+CLPORT=<mode>,<port> 多链接： AT+CLPORT=<n>,<mode>,<port>	OK 或 ERROR	
查询命令	AT+CLPORT?	+CLPORT: <TCP port>,<UDP port> OK	单链接 (CIPMUX=0)
		+CLPORT: 0,<TCP port>,<UDP port> +CLPORT: 1,<TCP port>,<UDP port> +CLPORT: 2,<TCP port>,<UDP port> +CLPORT: 3,<TCP port>,<UDP port> +CLPORT: 4,<TCP port>,<UDP port> +CLPORT: 5,<TCP port>,<UDP port>	多链接 (CIPMUX=1)

		OK	
测试命令	AT+CLPORT=?	+CLPORT: ("TCP","UDP"),(0-65535) OK	单链接 (CIPMUX=0)
		+CLPORT: (0-5),("TCP","UDP"),(0-65535) OK	多链接 (CIPMUX=1)

参数定义：

<n>	Link No.	0~5	整数型，表示连接序号
<mode>	连接类型，字符串型（双引号可加可不加）	"TCP"	建立TCP连接
		"UDP"	建立UDP连接
<port>	模块服务端口	1~65535	整数型

12.9 建立 TCP 连接或注册 UDP 端口号：AT+CIPSTART

语法规则：

命令类型	语法	返回和说明
设置命令	单路连接(+CIPMUX=0)时： AT+CIPSTART=<mode>,<sever>,<port>	如果格式正确且处于IP INITIAL 或者 IP STATUS或TCP/UDP CLOSE状态，返回： **OK** 否则返回： **+CME ERROR <err>** 紧接着会有**URC**上报，上报内容如下： 如果连接已经存在，返回： **ALREADY CONNECT** 如果连接成功(非透传)，返回： **CONNECT OK** 如果连接成功(透传)，返回： **CONNECT** 否则返回： **STATE: <sl_state>** **CONNECT FAIL**
	多路连接(+CIPMUX=1)时：	如果格式正确且处于 IP STATUS或IP PROCESSING时，返回： **OK**

	AT+CIPSTART=<n>,<mode>,<server>,<port>	否则返回： **+CME ERROR <err>** 紧接着会有**URC**上报，上报内容如下： 如果连接已经存在，返回： **<n>,ALREADY CONNECT** 如果连接成功，返回： **<n>,CONNECT OK** 否则返回： **<n>,CONNECT FAIL**
测试命令	AT+CIPSTART=?	单路连接(+CIPMUX=0)时返回： +CIPSTART: (<mode>取值列表),(IP address range),(port range) +CIPSTART: (<mode>取值列表),(domain name),(port range) OK
		多路连接(+CIPMUX=1)时返回： +CIPSTART: (<n>取值列表),(<mode>取值列表),(IP addressrange),(port range) +CIPSTART: (<n>取值列表),(<mode>取值列表),(domain name),(portrange) OK
注意事项	● 此命令应用于建立 TCP/UDP 连接； ● 当前状态可用 AT+CIPSTATUS 查询； ● 单路连接时只当前状态为 IP INITIAL 或者 IP STATUS 或 TCP/UDP CLOSE 时可执行，多路连接时当前状态为 IP STATUS 或 IP PROCESSING 时可执行； ● 在当前状态不是上述可执行状态时，需执行 AT+CIPSHUT 后再开始建立连接； ● 多路连接时，设置此命令前，必须先执行 AT+CSTT, AT+CIICR,AT+CIFSR 这三个命令。	

参数定义：

参数	定义	取值	对取值的说明
<n>	Link No.	0~5	整数型，表示连接序号
<mode>	连接类型，字符串型（双引号可加可不加）	"TCP"	建立TCP连接
		"UDP"	建立UDP连接
<server>	远端服务器 IP 地址或域名皆可	最大128个字节	字符串参数（双引号可加可不加）
<port>	远端服务端口	1~65535	整数型

<sl_state>	单连接状态	IP INITIAL	初始化
		IP START	启动任务
		IP CONFIG	配置场景
		IP GPRSACT	场景已激活
		IP STATUS	获得本地 IP 状态
		TCP CONNECTING/UDP CONNECTING/SERVER LISTENING	TCP 连接中/UDP 端口注册中/服务器侦听中
		CONNECT OK	连接建立成功
		TCP CLOSING/UDP CLOSING	正在关闭 TCP 连接，正在注销 UDP 端口
		TCP CLOSED/UDP CLOSED	连接断开 /UDP 端口被注销
		PDP DEACT	场景被释放

12.10 选择 TCPIP 应用模式：AT+CIPMODE

语法规则：

命令类型	语法	返回
设置命令	AT+CIPMODE=<mode>	OK
查询命令	AT+CIPMODE?	+CIPMODE: <mode> OK
测试命令	AT+CIPMODE=?	+CIPMODE: (0-NORMAL MODE,1-TRANSPARENT MODE) OK
注意事项	此命令只有在IP INITIAL状态下才能进行设置 只有TCP单链接才支持透明传输模式	

参数定义：

参数	定义	取值	对取值的说明
<mode>	TCPIP 应用模式	0	非透明传输模式
		1	透明传输模式

12.11 选择非透传数据发送模式：AT+CIPQSEND

语法规则：

命令类型	语法	返回

设置命令	AT+CIPQSEND=<n>	OK
查询命令	AT+CIPQSEND?	+CIPQSEND: <n> OK
测试命令	AT+CIPQSEND=?	+CIPQSEND: (0,1,2) OK

参数定义：

参数	定义	取值	对取值的说明
<n>	非透传数据发送模式	0	缺省值。快发模式0。 当服务器收到数据，模块返回: SEND OK （单链接） 或 <n>, SEND OK （多链接）
		1	快发模式1。当数据发送到模块 ， 终 端 返 回： DATA ACCEPT:<length> （单链接） DATA ACCEPT:<n>,<length> （多链接）
		2	慢发模式 当服务器收到数据，模块返回: SEND OK （单链接） 或 <n>, SEND OK （多链接）

注：

- 慢发模式每发送一笔数据需要服务器那边的确认，而快发则发送到模块就可以了，不需要服务器的确认。推荐使用 0 或 1，即快发模式；
- 这两种快发模式实现机制没有区别，区别只在于发送完数据，模式 0 提示 SEDN OK，模式 1 提示 DATA ACCEPT；
- 建议使用 CIPACK 命令查询每笔数据对端是否已经收到

12.12 设置接收的数据末尾是否自动添加回车换行：AT+CIPRXF

语法规则：

命令类型	语法	返回
设置命令	AT+CIPRXF=<n>	OK
查询命令	AT+CIPRXF?	+CIPRXF:<n> OK

测试命令	AT+CIPRXF=?	+CIPRXF: (<n>的取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<n>	模式	0	每笔收到的数据末尾自动添加一个回车换行（\r\n）
		1	每笔收到的数据末尾不添加回车换行（\r\n）

12.13 配置透明传输模式：AT+CIPCCFG

语法规则：

命令类型	语法	返回
设置命令	AT+CIPCCFG=<NmRetry>,<WaitTm>,<SendSz>,<esc>[,<Rxmode>,<RxSize>,<Rxtimer>,<BufClean>]	OK
查询命令	AT+CIPCCFG?	+CIPCCFG: <NmRetry>,<WaitTm>,<SendSz>,<esc>,<Rxmode>,<RxSize>,<Rxtimer>,<BufClean> OK
测试命令	AT+CIPCCFG=?	+CIPCCFG: (NmRetry:3-8),(WaitTm:2-10),(SendSz:1-1460),(esc:0,1),(Rxmode:0,1),(RxSize:50-1460),(Rxtimer:20-1000),(BufClean:0,1) OK
注意事项	此命令只在单路连接（AT+CIPMUX=0）且AT+CIPMODE=1的情况下可以设置	

参数定义：

参数	定义	取值	对取值的说明
<NmRetry>	一个 IP 包的重传次数	3~8	缺省值为5
<WaitTm>	间隔发送时间	2~10	单位为 100ms，缺省值为 2。
<SendSz>	每次发送数据字节数	1~1460	缺省值为1024
<esc>	是否开启转义序列 **[软件上不支持转义功能，请忽略此参数所代表的意义]**	1	开启转义序列
		0	不开启转义序列，默认值
<Rxmode>	从串口接收数据的时间间隔设置	1	设置时间间隔，间隔为<Rxtimer>
		0	没有时间间隔

<RxSize>	每次接收数据的长度	50-1460	单位为 bytes
<Rxtimer>	从串口接收数据的时间间隔	20-1000	单位为 ms，缺省值为 50ms
<BufClean>	退出透传后是否清空发送缓存区	0	不清空缓冲区 一旦出错自动退出透传模式后，保留缓存数据。下次重连进入透传模式后，会发送之前缓冲区中的数据给服务器
		1	清空缓存区 一旦出错自动退出透传模式后，缓存被清空。下次重连进入透传模式后，不会发送之前缓冲区中被清空的数据给服务器

12.14 发送数据：AT+CIPSEND

语法规则：

命令类型	语法	返回和说明
设置命令	单路连接(AT+CIPMUX=0)： AT+CIPSEND=<length>	本命令用于在单链接模式下发送长度不可变的数据 1) 正常情况下返回 **>** (后面有个空格) >(空格)后输入<length>长度的数据，则数据自动发送出去。 如果是快发模式0或慢发模式，则数据发送成功后，有如下的返回： **SEND OK** 如果是快发模式1，则数据发送成功后，有如下的返回： **DATA ACCEPT:<length>** 如果数据发送失败，则有如下的返回： **SEND FAIL** 2) 如果连接未建立或者连接被断开，返回： **+CME ERROR <err>**
	多路连接(+CIPMUX=1)： AT+CIPSEND=<n>[,<length>]	当<length>省略时，本命令用于多连接模式下发送长度可变的数据。响应"> "， 此时 输入数据，执行CTRL+Z(0x1A)发送，或执行ESC(0x1B)中止操作； 当<length>不省略时，本命令用于多连接模式下发送长度不可变的数据。响应"> "，>后输入<length>长度的数据，则数据自动发送出去。 如果连接未建立或者连接被断开，返回： **+CME ERROR <err>**

		如果是快发模式0或慢发模式，则数据发送成功后，有如下的返回： **<n>,SEND OK** 如果是快发模式1，则数据发送成功后，有如下的返回： **DATA ACCEPT:<n>,<length>** 如果数据发送失败，返回： **<n>,SEND FAIL**
执行命令	AT+CIPSEND	本命令用于在单链接模式下发送长度可变的数据 响应"**>** "， 此时 输入数据，执行CTRL+Z(0x1A)发送，或执行ESC(0x1B)中止操作 如果连接未建立或已被断开，返回： **+CME ERROR <err>** 如果是慢发模式，则数据发送成功后，有如下的返回： **SEND OK** 如果是快发模式，则数据发送成功后，有如下的返回： **DATA ACCEPT:<length>** 如果数据发送失败，返回： **SEND FAIL**
查询命令	AT+CIPSEND?	单路连接(AT+CIPMUX=0)返回： +CIPSEND: <size> OK
		多路连接(AT+CIPMUX=1)返回： +CIPSEND: <n>,<size> OK
测试命令	AT+CIPSEND=?	单路连接(AT+CIPMUX=0)返回： +CIPSEND: <length> OK
		多路连接(AT+CIPMUX=1)返回： +CIPSEND: <0-7>,<length> OK

注意事项	● 数据的最大发送长度由网络决定。 ● 通过 AT+CIPATS 可以在设定的时间内自动发送数据。 ● 只有在连接已建立的状态下才可以发送数据。 ● 一次最多能发送字节数不大于<size>值 ● **不推荐使用慢发模式！** ● 发送命令 AT+CIPSEND 或 AT+CIPSEND=<length>以\r 或\r\n 为结尾。若用户以\r 结尾，如果发送的数据第一个字节恰好是\n，模块会判断发送命令以\r\n 为结尾，导致发送的数据的第一个字符\n 被吞掉。所以，此时建议用户用 CIPSEND 发送数据以\r\n 为结尾。 例如： AT+CIPSEND\r >\n123456 模块会判断为 AT+CIPSEND\r\n，会把数据中第一个字符\n 吃掉。服务器那边收到的数据是 123456。 解决的方法是： AT+CIPSEND\r\n >\n123456 再例如： AT+CIPSEND=7\r >\n123456 模块会判断为 AT+CIPSEND\r\n，会把数据中第一个字符\n 吃掉，导致数据少一个字符，会一直发不出去。解决的方法是： AT+CIPSEND=7\r\n >\n123456

参数定义：

参数	定义	取值	对取值的说明
<n>	Link No. 与+CIPSTRAT 中<n>定义一致	0~5	整数型，表示连接序号
<length>	发送数据长度	-	整数型，它必须小于<size>
<size>	每次发送数据最大值	-	整数型，目前是1460字节

12.15 设置自动发送数据前的定时时间：AT+CIPATS

语法规则：

命令类型	语法	返回
设置命令	AT+CIPATS=<mode>[,<time>]	OK
查询命令	AT+CIPATS?	+CIPATS: <mode>,<time> OK
测试命令	AT+CIPATS=?	+CIPATS: (<mode>取值列表),(<time>取值列表)

		OK

参数定义：

参数	定义	取值	对取值的说明
<mode>	是否设置自动发送时间，整数型	0	未设置自动发送数据（缺省值）
		1	设置自动发送数据
<time>	数据将在多少秒后被发送	1~100	整数型，以秒为单位

12.16 设置发送数据时是否显示‘>’和发送情况提示：AT+CIPSPRT

语法规则：

命令类型	语法	返回
设置命令	AT+CIPSPRT=<send prompt>	OK
查询命令	AT+CIPSPRT?	+CIPSPRT: <send prompt> OK
测试命令	AT+CIPSPRT=?	+CIPSPRT: (<send prompt>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<send prompt>	执行 AT+CIPSEND 后是否显示‘>’和发送情况提示（即‘SEND OK‘或‘DATA ACCEPT‘）。 整数型	0	不显示‘>’，但返回 "SEND OK"或"DATA ACCEPT" **注:**返回 "SEND OK"或"DATA ACCEPT"由AT+CIPQSEND这个命令的设定来决定
		1	显示‘>’，且返回 "SEND OK" 或"DATA ACCEPT" 缺省值 **注:**返回 "SEND OK"或"DATA ACCEPT" 由AT+CIPQSEND这个命令的设定来决定
		2	不显示‘>’，不返回 "SEND OK" 或"DATA ACCEPT"

12.17 查询当前连接状态：AT+CIPSTATUS

语法规则：

命令类型	语法	返回和说明
执行命令	AT+CIPSTATUS	如果是单路连接(AT+CIPMUX=0)，返回： OK STATE: <sl_state>
		如果是多路连接 (AT+CIPMUX=1)，返回： OK STATE:<ml_state> C:<n>,<bearer>, <TCP/UDP>, <IP address>, <port>, <client state>
查询指令	AT+CIPSTATUS=<n> （多路连接时支持）	多路连接 (AT+CIPMUX=1)，返回： <n>,<bearer>, <TCP/UDP>, <IP address>, <port>, <client state>
测试命令	AT+CIPSTATUS=?	返回： OK

参数定义：

参数	定义	取值	对取值的说明
<n>	Link No.	0~5	整数型，表示连接序号 与+CIPSTRAT中<n>定义一致
<bearer>	GPRS 承载方式	0~1	缺省是 0
<IP address>	IP 地址	-	字符串参数(字符串需要加引号)
<port>	端口号	-	整数型
<sl_state>	单连接状态	IP INITIAL	初始化
		IP START	启动任务
		IP CONFIG	配置场景
		IP GPRSACT	场景已激活
		IP STATUS	获得本地 IP 状态
		TCP CONNECTING/UDP CONNECTING/SERVER LISTENING	TCP 连接中/UDP 端口注册中/服务器侦听中
		CONNECT OK	连接建立成功
		TCP CLOSING/UDP CLOSING	正在关闭 TCP 连接，正在注销 UDP 端口
		TCP CLOSED/UDP CLOSED	连接断开 /UDP 端口被注销

		PDP DEACT	场景被释放
<ml_state>	多链接状态	IP INITIAL	初始化
		IP START	启动任务
		IP CONFIG	配置场景
		IP GPRSACT	场景已激活
		IP STATUS	获得本地 IP 状态
		IP PROCESSING	IP 数据阶段
		PDP DEACT	场景被释放
<client state>	客户端状态	INITIAL	初始化
		CONNECTING	正在连接
		CONNECTED	已连接
		REMOTE CLOSING	对端关闭
		CLOSING	正在关闭
		CLOSED	已关闭

12.18 查询已连接数据传输状态：AT+CIPACK

语法规则：

命令类型	语法	返回
设置命令	多路连接(+CIPMUX=1)： AT+CIPACK=<n>	+CIPACK: <txlen>, <acklen>, <nacklen> OK
执行命令	单路连接(AT+CIPMUX=0)： AT+CIPACK	+CIPACK: <txlen>, <acklen>, <nacklen> OK
测试命令	AT+CIPACK=?	OK
注意事项	当链接建立后，查询AT+CIPACK，<txlen>, <acklen>, <nacklen>三个参数的初始值都是0，每发一笔数据，这三个参数都会累积增加。 AT+CIPSHUT后或链接断链后重连，查询AT+CIPACK，三个参数都重置为0	

参数定义：

参数	定义	取值	对取值的说明
<n>	Link No.	0~5	整数型，表示连接序号。与+CIPSTRAT中<n>定义一致
<txlen>	链接<n>建立以来累计已发送的数据字节数	-	整数型
<acklen>	链接<n>建立以来累计服务器已确认收到的数据字节数	-	整数型

<nacklen>	链接<n>建立以来服务器尚未确认收到的数据字节数	-	整数型

12.19 设置为 CSD 或 GPRS 连接模式：AT+CIPCSGP

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CIPCSGP=<mode>,[<apn>,<user>,<pwd>]	OK
查询命令	AT+CIPCSGP?	+CIPCSGP: <mode>, <apn>, <user>, <pwd> OK
测试命令	AT+CIPCSGP=?	+CIPCSGP: 0-CSD,DIAL NUMBER,USER NAME,PASSWORD,RATE(0-3) +CIPCSGP: 1-GPRS,APN,USER NAME,PASSWORD OK

参数定义：

参数	定义	取值	对取值的说明
<mode>	无线连接模式	1	GPRS连接
		2	CSD连接
GPRS下要设置的参数：			
<apn>	GPRS 接入点名称	-	字符串参数(字符串需要加引号)
<username>	GPRS 用户名	-	字符串参数(字符串需要加引号)
<password>	GPRS 密码	-	字符串参数(字符串需要加引号)

12.20 配置域名服务器 DNS：AT+CDNSCFG

语法规则：

命令类型	语法	返回
设置命令	AT+CDNSCFG=<pri_dns>[,<sec_ dns>[,<cid>]]	OK
查询命令	AT+CDNSCFG?	PrimaryDns: <pri_dns> SecondaryDns: <sec_dns> OK
测试命令	AT+CDNSCFG=?	+CDNSCFG: ("Primary DNS"),("Secondary DNS")

		OK

参数定义：

参数	定义	取值	对取值的说明
<pri_dns>	主域名服务器的 IP地址		字符串参数(字符串需要加引号)
<sec_ dns>	备域名服务器的 IP 地址		字符串参数(字符串需要加引号)
<cid>	定义同+SAPBR 中<cid>	1~3	当使用 SAPBR 激活 pdp 承载后，如有需要，再使用带<cid>的命令设置 DNS 服务器

举例：

命令(→) 返回(←)	实例	解释和说明
+CDNSCFG命令查询和修改域名服务器的场景应用1（TCPIP,MQTT应用）：		
→	AT+CREG?	查询当前GPRS注册状态
←	+CREG: 0,1 OK	<n>=0，表示禁用URC上报 <stat>=1，标识已经注册GPRS网络，而且是本地网
→	AT+CSTT	
←	OK	
→	AT+CIICR	
←	OK	
→	AT+CIFSR	
←	10.113.72.66	
→	AT+CDNSCFG?	查询默认DNS服务器
←	PrimaryDns: 211.136.112.50 SecondaryDns: 211.136.150.66 OK	
→	AT+CDNSCFG=ip1,ip2	如果有需要，客户可以修改DNS服务器 ip1和ip2请按照实际的dns服务器地址输入，ip1和ip2可以加双括号，也可以不加
←	OK	
→	AT+CIPSTART=TCP,<server domain>,<port>	连接一个域名地址。所有参数可加双括号，也可不加 本例是TCPIP的应用举例。如果是MQTT应用，此时可依次输入： AT+MCONFIG,AT+MIPSTART,AT+MCONNECT等命令，具体请参考MQTT使用方法举例
←	OK	

	CONNECT OK	
+CDNSCFG查询和修改域名服务器的场景应用2（HTTP,FTP应用）：		
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	
←	OK	
→	AT+SAPBR=3,1,"APN",""	
←	OK	
→	AT+SAPBR=1,1	激活cid=1的PDP上下文
←	OK	
→	AT+SAPBR=2,1	
←	+SAPBR: 1,1,010.169.179.213 OK	
→	AT+CDNSCFG=ip1,ip2,1	
←	OK	
→	AT+CDNSCFG?	查询DNS服务器。这种应用场景，必须要先执行+CDNSCFG设置命令才能查询
←	PrimaryDns: ip1 SecondaryDns: ip2 OK	以上ip1,ip2都是实际的DNS域名服务器的ip地址，可加双括号，也可不加
→	HTTP应用，依次输入：AT+HTTPINIT，AT+HTTPPARA，AT+HTTPACTION等命令，具体请参考：HTTP使用方法举例 FTP应用，依次输入：AT+FTPCID，AT+FTPSERV，AT+FTPUN，AT+FTPPW等命令，具体请参考：FTP使用方法举例	

12.21 域名解析：AT+CDNSGIP

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CDNSGIP=<domain name>	如果命令正确，而且域名解析成功，响应： OK +CDNSGIP: 1, <domain name>,<IPaddress> 如果命令正确，但域名解析失败，响应： OK +CDNSGIP:0,<dns error code>

		如果命令语法错误，响应： ERROR
测试命令	AT+CDNSGIP=?	返回： OK

参数定义：

参数	定义	取值	对取值的说明
<domain name>	Internet 上注册的域名	-	字符串参数(字符串需要加引号)，不超过 128 个字节
<IPaddress>	域名对应的 IP 地址	-	字符串参数(字符串需要加引号)
<dns error code>	DNS 相关的错误码	10	GENERAL ERROR
		11	MAX RETRIES
		12	NO SERVER ADDR
		13	NO MEMORY
		14	INVALID NAME
		15	INVALID RESP
		其他	一些其他的错误代码

12.22 设置单链接接收数据时是否显示发送方的 IP 地址和端口号：AT+CIPSRIP

语法规则：

命令类型	语法	返回
设置命令	AT+CIPSRIP=<mode>	OK
查询命令	AT+CIPSRIP?	+CIPSRIP: <mode> OK
测试命令	AT+CIPSRIP=?	+CIPSRIP: (<mode>取值列表) OK
注意事项	此命令只在单路连接模式下有效(+CIPMUX=0)	

参数定义：

参数	定义	取值	对取值的说明
<mode>	是否显示发送方的 IP 地址和端口号	0	不显示提示（缺省值）
		1	显示提示，格式如下: RECV FROM:<IP ADDRESS>:<PORT>

12.23 设置单链接接收数据是否显示 IP 头：AT+CIPHEAD

语法规则：

命令类型	语法	返回
设置命令	AT+CIPHEAD=<mode>	OK
查询命令	AT+CIPHEAD?	+CIPHEAD: <mode> OK
测试命令	AT+CIPHEAD=?	+CIPHEAD: (<mode>取值列表) OK
注意事项	此命令只有在单路连接的模式下 (+CIPMUX=0)才有效	

参数定义：

参数	定义	取值	对取值的说明
<mode>	接收数据时是否增加 IP 头提示	0	未设置IP头
		1	设置IP头，格式是 "+IPD,data length:"

12.24 设置单链接接收数据是否在 IP 头显示传输协议：AT+CIPSHOWTP

语法规则：

命令类型	语法	返回
设置命令	AT+CIPSHOWTP=<mode>	OK
查询命令	AT+CIPSHOWTP?	+CIPSHOWTP: <mode> OK
测试命令	AT+CIPSHOWTP=?	+CIPSHOWTP: (<mode>取值列表) OK
注意事项	此命令只有在单路连接的模式下 (+CIPMUX=0) 并且 AT+CIPHEAD=1 时才有效	

参数定义：

参数	定义	取值	对取值的说明
<mode>	接收数据时是否在	0	不显示传输协议 （默认值）

	IP 头显示传输协议	1	显示传输协议，格式是"+IPD,<data Size>,<TCP/UDP>:<data>"

12.25 多链接时接收数据：+RECEIVE

语法规则：

命令类型	语法	返回和说明
URC 上报	+RECEIVE,<n>,<length>: Received data	注：Received data 是接收到的数据，和 +RECEIVE,<n>,<length>分开，另起一行

参数定义：

参数	定义	取值	对取值的说明
<n>	Link No. 与+CIPSTRAT 中<n>定义一致	0~5	整数型，表示连接序号
<length>	接收到的数据的长度	-	整数型

12.26 保存 TCPIP 应用上下文：AT+CIPSCONT

该命令的执行命令保存包含相应的TCPIP AT 命令参数，即TCPIP 应用上下文，当系统重启时，参数将自动载入。
查询命令查询当前的 TCPIP 应用上下文的设置。

语法规则：

命令类型	语法	返回和说明
查询命令	AT+CIPSCONT?	+CIPSCONT:<value> +CIPCSGP:<mode> Gprs Config APN:<apn> Gprs Config UserId:<user name> Gprs Config Password:<password> +CIPHEAD:<mode> +CIPSHOWTP:<mode> +CIPSRIP:<mode> +CIPATS:<mode>,<time> +CIPSPRT:<send prompt> +CIPQSEND:<n> +CIPMODE:<mode> +CIPCCFG:<NmRetry>,<WaitTm>,<SendSz>,<esc>,<Rxmode>,<RxSize>,<Rxtimer>

		+CIPMUX:<n> +CIPDPDP:<mode>, <interval>, <timer> +CIPRXGET:<mode> +CIPRDTIMER: 2000,3500 OK
执行命令	AT+CIPSCONT	OK

参数定义：

参数	定义	取值	对取值的说明
<value>	是否保存上下文	0	保存 TCPIP 应用上下文
		1	默认值，表示不保存 TCPIP 应用上下文

12.27 手动获得网络数据：AT+CIPRXGET

语法规则：

命令类型	语法	返回和说明
设置命令	单链接(AT+CIPMUX=0)时： AT+CIPRXGET=<mode>[,<len>] 多链接(AT+CIPMUX=1)时： AT+CIPRXGET=<mode>[,<id>[,<len>]]	打开手动获取数据功能： AT+CIPRXGET=1(或5) OK 设置为1后，第一次上报数据会有+CIPRXGET:1的URC上报，后面再来数据就不报了，直到AT+CIPRXGET=2或3来读取，才会再次上报。 设置为5后，每次来数据皆有上报，不管上次数据有没有读过。
		收到+CIPRXGET:1输入以下命令来手动读取数据： AT+CIPRXGET=2,<len> +CIPRXGET:2,<cnlen>,<rlen> ……数据….. OK
		收到+CIPRXGET:1也可以16进制读取： AT+CIPRXGET=3,<len> +CIPRXGET:3,<cnlen>,<rlen> ……数据….. OK

		查询还有多少数据未读： AT+CIPRXGET=4 +CIPRXGET:4,<rlen> OK
	多链接(AT+CIPMUX=1)时： AT+CIPRXGET=<mode>,<n>[,<len>]	打开手动获取数据功能输入： AT+CIPRXGET=1(或5) OK 后续来数据会有+CIPRXGET:1的URC上报
		收到+CIPRXGET:1输入以下命令来手动读取数据： AT+CIPRXGET=2,<n>,<len> +CIPRXGET:2,<n>,<cnlen>,<rlen> ……数据….. OK
		收到+CIPRXGET:1也可以16进制读取： AT+CIPRXGET=3,<n>,<len> +CIPRXGET:3,<n>,<cnlen>,<rlen> ……数据….. OK
		查询还有多少数据未读： AT+CIPRXGET=4,<n> +CIPRXGET:4,<n>,<rlen> OK
查询命令	AT+CIPRXGET?	+CIPRXGET:<mode> OK
测试命令	AT+CIPRXGET=?	OK
URC 上报	+CIPRXGET:1	单链接，设置AT+CIPRXGET=1或5后，当收到网络数据时，会有如此的URC上报，表示收到数据
	+CIPRXGET:1,<n>	多链接，设置AT+CIPRXGET=1或5后，当收到网络数据时，会有如此的URC上报，表示收到数据

参数定义：

参数	定义	取值	对取值的说明
<mode>	模式	0	关闭手动获取数据功能。缺省值。
		1	开启手动获取数据功能。 设置为1后，第一次上报数据会有+CIPRXGET:1的URC上报，后面再来数据就不报了，直到AT+CIPRXGET=2或3来读取，才会再次上报
		2	读取数据
		3	读取数据（16进制）
		4	还有多少数据未读
		5	开启手动获取数据功能。 设置为5后，每次来数据皆有上报。
<n>	多链接时的 Link id	0-5	定义同+CIPSTART命令的<n>
<len>	将要读取的数据的长度	1-1460	普通字符方式读取时
		1-730	HEX方式读取时
<cnlen>	已经读取的数据		单位：字节
<rlen>	尚未读取的数据		单位：字节

举例：

命令（→）/返回（←）	实例	解释和说明
单链接时如何手动读取网络数据		
→	AT+CIPRXGET=1	开启手动获取网络数据功能
←	OK	
→	AT+CGREG?	查询当前GPRS注册状态
←	+CGREG: 0,1 OK	<n>=0，表示禁用URC上报 <stat>=1，标识已经注册GPRS网络，而且是本地网
→	AT+CIPSTART="TCP","36.9.88.120",6001	连接服务器（单链接）
←	OK	
←（URC）	CONNECT OK	已经连上了
←（URC）	+CIPRXGET: 1	服务器那边发数据过来了
→	AT+CIPRXGET=2,150	读取数据，读150个字符，以普通的字符的方式读取
←	+CIPRXGET: 2,10,0 1234567890 OK	读到了10个数据：1234567890，还有0个数据未读取
→	AT+CIPRXGET=4	查询下还有多少数据未读

←	+CIPRXGET: 4,0 OK	0个未读
←（URC）	+CIPRXGET: 1	服务器那边又发数据过来了
→	AT+CIPRXGET=3,150	读取数据，读150个字符，以HEX的方式读取
←	+CIPRXGET: 3,5,0 48454C4C4F OK	读到了5个数据：HELLO，还有0个数据未读取
多链接时如何手动读取网络数据		
→	AT+CIPRXGET=1	开启手动获取网络数据功能
←	OK	
→	AT+CIPMUX=1	
←	OK	
→	AT+CSTT	模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文（此<apn>可以通过AT+CGDCONT?来查询），所以直接输入AT+CSTT即可，模块会按照自动获取的<apn>来设置CSTT的APN
←	OK	
→	AT+CIICR	
←	OK	
→	AT+CIFSR	
←	OK	
→	AT+CIPSTART=3,"TCP","36.9.88.120",6001	建立多链接
←	OK	
←（URC）	3, CONNECT OK	
←（URC）	+CIPRXGET: 1,3	id=3的链接上，收到服务器发来的数据
→	AT+CIPRXGET=2,3,10	读取链接3的数据，读10个字符，以普通的字符的方式读取
←	+CIPRXGET: 2,3,5,0 AAAAA OK	在链接3上读取了5个字符：AAAAA，还有0个未读
→	AT+CIPRXGET=4,3	查询下链接3还有多少数据未读
←	+CIPRXGET: 4,3,0 OK	0个未读

12.28 关闭 TCP 或 UDP 连接：AT+CIPCLOSE

语法规则：

命令类型	语法	返回和说明
设置命令	单路连接时 AT+CIPCLOSE=<id>	返回： CLOSE OK
	多路连接时 AT+CIPCLOSE=<n>[,<id>]	返回： <n>,CLOSE OK
执行命令	AT+CIPCLOSE	如果关闭成功，返回： CLOSE OK 如果关闭失败，返回： ERROR
测试命令	AT+CIPCLOSE=?	返回： OK
注意事项	● 执行命令只对单链接有效，多链接模式下返回ERROR ● 执行命令AT+CIPCLOSE 只有在TCP/UDP CONNECTING或CONNECT OK状态下才会关闭连接，否则会认为关闭失败返回ERROR ● 单路连接模式下，关闭后的状态为IP CLOSE	

参数定义：

参数	定义	取值	对取值的说明
<id>	关闭模式	0	慢关（缺省值）
		1	快关
<n>	Link No.	0~5	整数型，表示连接序号。与+CIPSTRAT中<n>定义一致

12.29 关闭移动场景：AT+CIPSHUT

语法规则：

命令类型	语法	返回和说明
执行命令	AT+CIPSHUT	如果关闭成功，返回： SHUT OK 如果关闭失败，返回： ERROR

测试命令	AT+CIPSHUT=?	返回： OK
注意事项	● 可以用 AT+CIPSHUT 正常关闭移动场景，关闭后状态为 IPINITIAL。 ● 多路连接时执行 AT+CIPSHUT，所有的 IP 连接都将关闭。 ● 如果上报"+PDP:DEACT"，标识 GPRS 被网络释放，此时仍然需要执行 AT+CIPSHUT 来改变状态。	

12.30 修改 RNDIS 网卡网关 IP 地址：AT+ROUTEIP

EC618 系列模块>=V1140 版本支持

语法规则：

命令类型	语法	返回
设置命令	AT+ROUTEIP=<ip>	OK
查询命令	AT+ROUTEIP?	<ip> OK
测试命令	AT+ROUTEIP=?	OK

参数定义：

参数	定义	取值	对取值的说明
<ip>	当前的 RNDIS 网卡网关 IP 地址		IP 地址，双引号可加可不加，只支持 192.168.X.2

12.31 Ping 回声请求命令:AT+CIPPING

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CIPPING=<IPaddr>[,<retryNum>[,<dataLen>[,<timeout>[,<ttl>]]]]	+CIPPING: <replyId>,<Ip Address>,<replyTime>,<ttl>[<CR><LF> +CIPPING: <replyId>,<Ip Address>,<replyTime>,<ttl> [...]] OK
查询命令	AT+CIPPING?	+CIPPING: <retryNum>,<dataLen>,<timeout>,<ttl> OK
测试命令	AT+CIPPING=?	+CIPPING: (list of supported <retryNum>s),(list of supported <dataLen>s),(list of supported <timeout>s),(list of supported <ttl>s) OK

注意事项	● 发送 PING 命令之前，需激活 GPRS PDP 上下文。 ● 当发送 PING 到时无回应，那么返回的信息则显示<replyTime>=600 并且<ttl>=255。 ● 执行本命令时，如果 GPRS PDP 上下文由于某种原因被去激活了，例如掉网，那么本命令立即终止执行。

参数定义：

参数	定义	取值	对取值的说明
<IPaddr>	PING的目标服务器地址。IP地址或域名都支持	ip address	
		Domain name	
<retryNum>	需要发送的PING的数量	1-100	缺省值:4
		0	可以不停的进行ping，并且最大的ping数量可以达到0xffffffff个 EC618系列模块>=V1120版本支持
<dataLen>	PING请求的长度	0-1024	缺省值:32
<timeout>	The timeout waiting for a single Echo Reply	1-600	单位：100 ms
<ttl>	time to live	1-255	缺省值:64
<replyId>	Echo Reply serial number		
<Ip Address>	IP Address of the remote host		
<replyTime>	time to receive the response		单位：1ms

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+CSTT	
←	OK	
→	AT+CIICR	
←	OK	
→	AT+CIFSR	
←	10.207.9.213	
→	AT+CIPPING="www.baidu.com"	
←	+CIPPING: 1,"36.152.44.96",35,54 +CIPPING: 2,"36.152.44.96",20,54 +CIPPING: 3,"36.152.44.96",20,54 +CIPPING: 4,"36.152.44.96",35,54 OK	
→	AT+CIPPING="www.baidu.com",0	可以不停的进行ping，并且最大的ping数量可以达

		到0xffffffff个 EC618 系列模块>=V1120 版本支持
←	+CIPPING: 1,"112.80.248.76",40,54 +CIPPING: 2,"112.80.248.76",45,54 +CIPPING: 3,"112.80.248.76",35,54 +CIPPING: 4,"112.80.248.76",35,54 +CIPPING: 5,"112.80.248.76",40,54 +CIPPING: 6,"112.80.248.76",35,54 ...	

12.32 设置 TCP 保活（keep-alive）参数:AT+CIPTKA

语法规则：

命令类型	语法	返回和说明
设置命令	AT+CIPTKA=<mode>[,<keepIdle>[,<keepInterval>[,<keepCount>]]]	OK
查询命令	AT+CIPTKA?	+CIPTKA:<mode>,<keepIdle>,<keepInterval>,<keepCount> OK
测试命令	AT+CIPTKA=?	+CIPTKA:(listofsupported<mode>s),(listofsupported<keepIdle>s),(listofsupported<keepInterval>s),(listofsupported<keepCount>s) OK

参数定义：

参数	定义	取值	对取值的说明
<mode>	是否开启TCP保活（keep-alive）	0	关闭
		1	开启
<keepIdle>	在<keepIdle>时间内链接上无任何数据交互，则发送初始保活探针（initial keep-alive probe）	30~7200	单位为秒，缺省值为7200
<keepInterval>	保活探针重传的间隔时间	30~600	单位为秒，缺省值为75
<keepCount>	发送保活探针的最大数量	1~9	单位为次，缺省值为9

12.33 心跳包参数设置:AT^HEARTCONFIG

本命令用于配置心跳的相关参数（是否允许发送心跳包，socket id，心跳间隔）。

语法规则：

命令类型	语法	返回和说明
设置命令	AT^HEARTCONFIG=<option>,<socket_id>,<heartbeat_time>	OK
查询命令	AT^HEARTCONFIG?	^HEARTCONFIG: <enable>,<socket_id>,<heartbeat_time> OK
测试命令	AT^HEARTCONFIG=?	OK
注意事项	目前仅支持一路链接设置心跳包。	

参数定义：

参数	定义	取值	对取值的说明
<option>	是否开启心跳包功能	0	disabled
		1	enabled
<socket_id>	连接id，定义同+CIPSTART中<n>	0~5	如果是单链接，固定为0 如果是多链接，取值为0~5
<heartbeat_time>	心跳间隔时间	5~600	单位为秒，缺省值为120

举例：

命令(→)/返回(←)	实例	解释和说明
→	AT^HEARTCONFIG?	查询当前心跳配置
←	^HEARTCONFIG: 0,0,120 OK	缺省配置为：心跳为关，心跳间隔为 120 秒
→	AT^HEARTCONFIG=1,0,180	开启心跳功能，cid=0，并设置心跳间隔为 180 秒
←	OK	
→	AT^HEARTCONFIG?	再次查询心跳设置
←	^HEARTCONFIG: 1,0,180 OK	心跳包功能已经打开，当跟服务器建立 cid=0 的链接后（可以是单链接，也可以是多链接），会自动发送心跳包，心跳包内容缺省为模块的 IMEI

12.34 设置心跳包内容:AT^HEARTBEAT

本命令设置心跳包的内容。默认状态下将自己的 IMEI 作为心跳包内容。

语法规则：

命令类型	语法	返回
设置命令	AT^HEARTBEAT=<socket_id>,<data>	OK
查询命令	AT^HEARTBEAT?	^HEARTBEAT: <socket_id>,<data> OK
测试命令	AT^HEARTBEAT=?	OK

参数定义：

参数	定义	取值	对取值的说明
<socket_id>	连接id，定义同+CIPSTART中<n>	0~5	如果是单链接，固定为0 如果是多链接，取值为0~5
<data>	心跳包内容		字符串类型，最长为256字节

举例：

命令(→)/返回(←)	实例	解释和说明
→	AT^HEARTBEAT?	查询当前心跳内容
←	^HEARTBEAT: 0,866714044915436 OK	缺省心跳为 IMEI
→	AT^HEARTBEAT=0,TCP-heart	设置新的心跳内容
←	OK	
→	AT^HEARTBEAT?	再次查询
←	^HEARTBEAT: 0,TCP-heart OK	

12.35 设置 HEX 编码格式心跳包内容:AT^HEARTBEATHEX

本命令将心跳包的内容设置为十六进制数据。

语法规则：

命令类型	语法	返回
设置命令	如果是单链接（即AT+CIPMUX=0）	OK

	AT^HEARTBEATHEX=<len>,<data>	>返回后，设置的心跳包内容会自动发送出去
	如果是多链接（即AT+CIPMUX=1） AT^HEARTBEATHEX=<socket_id>,<len>,<data>	
查询命令	如果是单链接（即AT+CIPMUX=0） AT^HEARTBEATHEX?	^HEARTBEATHEX: <len>,<data> OK
	如果是多链接（即AT+CIPMUX=1） AT^HEARTBEATHEX?	^HEARTBEATHEX:<socket_id>,<len>,<data> OK
测试命令	AT^HEARTBEATHEX=?	OK

参数定义：

参数	定义	取值	对取值的说明
<socket_id>	连接id，定义同+CIPSTART中<n>	0~5	如果是单链接，固定为0 如果是多链接，取值为0~5
<len>	数据长度		
<data>	心跳包内容		字符串类型，最长为256字节

举例：

命令(→)/返回(←)	实例	解释和说明
→	AT+CIPMUX=1	设置为多链接
←	OK	
→	AT+CSTT	
←	OK	
→	AT+CIICR	
←	OK	
→	AT+CIFSR	
←	10.134.100.30	
→	AT+CIPSTART=5,"TCP","116.28.164.159",40117	连接一个回环服务器（即你发给它什么，它回什么） 注:这个地址和端口是随意写的，请不要照抄
←	OK 5, CONNECT OK	
→	AT^HEARTCONFIG=1,5,120	打开第5路连接的心跳
←	OK	

→	AT^HEARTBEATHEX=5,6,414243444546	设置新的心跳内容为 ABCDEF
←	OK	
←	+RECEIVE,5,6: ABCDEF	

12.36 查询心跳包发送情况:AT^HEARTINQUIRE

本命令用于查询心跳包发送情况。

语法规则：

命令类型	语法	返回和说明
查询命令	AT^HEARTINQUIRE?	^HEARTINQUIRE: <suctime>,<nextime>,<heartbeat_time> OK
测试命令	AT^HEARTINQUIRE=?	OK

参数定义：

参数	定义	取值	对取值的说明
<suctime>	多少秒前发送成功了一条		单位：秒
<nextime>	还剩余多少秒发下一条		单位：秒
<heartbeat_time>	一共发了多少条了		单位：条

举例：

命令(→)/返回(←)	实例	解释和说明
→	AT^HEARTCONFIG=1,0,120	设置心跳包功能开启
←	OK	
→	AT+CIPSTART="TCP","36.6.*.*",12345	连接一个服务器 服务器地址和端口请根据自己实际情况填写，不要照抄
←	OK CONNECT OK	
→	AT^HEARTINQUIRE?	查询心跳包发送情况
←	^HEARTINQUIRE: 115,5,8 OK	
→	AT^HEARTCONFIG=0	设置心跳包功能关闭

←	OK	
→	AT^HEARTINQUIRE?	查询心跳包发送情况
←	^HEARTINQUIRE: 0,0,0 OK	统计已经清零

12.37 从数据模式或 PPP 在线模式切换至命令模式：+++

+++ 字符序列可使TA 忽略当前AT 接口的数据传输，并切换至命令模式。它允许TA 在保持与远端服务器数据连接的状态下，仍然可输入AT命令。

语法规则：

命令类型	语法	返回
执行命令	+++	OK **说明**：为避免+++ 被错误的识别为数据，需要遵循以下步骤： 1. "+++"输入前 T1 时间（1 秒）内无字符输入。 2.在 0.5 second 内连续输入三个+号，每个+号之间不能有其他字符。 3. "+++"输入后 T1 时间（0.5 秒）内无字符输入。 4.切换至命令模式，否则重新进入步骤 1。

举例：

命令（→）/返回（←）	实例	解释和说明
→	ATD*99#	数据拨号
←	CONNECT	数据拨号成功
→	+++	+++退出数据模式，返回到命令模式
←	OK	出现OK，表明已经返回到命令模式
→	ATH	挂断数据链接，否则下次无法再数据拨号
←	OK	挂断成功

12.38 从命令模式切换至数据模式：ATO

语法规则：

命令类型	语法	返回和说明
执行命令	ATO	成功，则返回： CONNECT

		说明：TA 继续数据传输连接，即从命令模式切换回数据模式
		若无法恢复到数据模式，则返回： NO CARRIER

举例：

命令（→）/返回（←）	实例	解释和说明
→	ATD*99#	数据拨号
←	CONNECT	成功
→	+++	切到AT命令模式
←	OK	成功
→	ATO	再切到数据模式
←	CONNECT	成功

12.39 TCP/UDP 错误码

在 TCP 应用过程中如果发生错误，将会以 TCP ERROR:<err code>形式上报。

TCP 错误码：

错误码<err code>	中文释义	英文释义
0	成功	No error
1	TCPIP线程没有被使用	TCPIP is idle
2	没有可用的 tsapi	No tsapi
3	无效的 tsapi	Invalid tsapi
4	空间不足	No buffer
5	网络错误	Network error
6	远程主机不可达	Remote host is unreachable
7	地址正在使用中	The address is already in use
8	地址无效	The address is not available
9	携带的数据太多或者太少	The supplied buffer is too large or small
10	参数无效	Invalid parameter
11	远程主机拒绝连接	Remote host has rejected the connection
12	超时	Time out
13	连接被终止	An established connection is aborted
14	连接被重置	Remote host has reset the connection
15	socket 连接已经建立	The socket is already connected
16	socket 没有连接	The socket is not connected

17	socket 连接已经被断开	The socket has been shutdown
18	未知错误	Undefined error

在 UDP 应用过程中如果发生错误，将会以 UDP ERROR:<err code>形式上报。

UDP 错误码：

错误码<err code>	中文释义	英文释义
0	成功	No error
1	TCPIP线程没有被使用	TCPIP is idle
2	没有可用的 tsapi	No tsapi
3	无效的 tsapi	Invalid tsapi
4	回调未注册	The callback has not been registered
5	空间不足	No buffer
6	网络错误	Network error
7	远程主机拒绝连接	Remote host has rejected the connection
8	远程主机不可达	Remote host is unreachable
9	地址正在使用中	The address is already in use
10	地址无效	The address is not available
11	携带的数据太多或者太少	The supplied buffer is too large or small
12	参数无效	Invalid parameter
13	TCPIP 线程忙	TCPIP is busy
14	未知错误	Undefined error
15	socket 连接已经建立	The socket is already connected

12.40 状态机

当输入 TCPIP 相关命令以后，模块的状态也会发生相应的迁移。查询状态的命令是 AT+CIPSTATUS 。

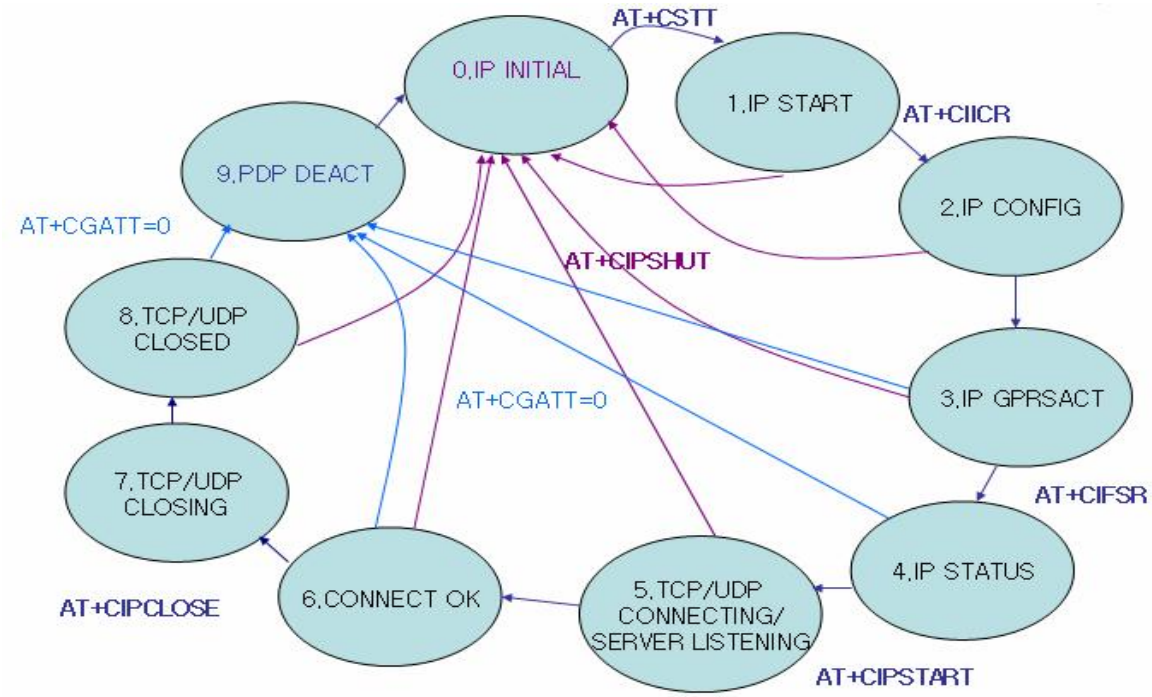


图 1 ： 单链接时的状态机

关于单连接状态机的几点说明：

◆输入 AT+CIICR，会马上进入 IP CONFIG 状态，当返回 OK 后，会进入到 IP GPRSACT 状态；

◆输入 AT+CIPSTART 后，会立马进入 IP/UDP CONNECTING 状态，如果后续模块上报 CONNECT OK 这个 URC，表明连接服务器成功，此时进入 CONNECT OK 状态；

◆输入 AT+CIPCLOSE 后，立马进入 TCP/UDP CLOSING 状态，此时如果模块上报 CLOSE OK，则表明关闭与服务器的连接成功，此时模块进入 TCP/UDP CLOSED 状态；

◆如果模块上报+PDP DEACT 这个 URC，则标志着模块释放 PDP 上下文，并进入了 PDP DEACT 状态；

◆在 IP GPRSACT，IP STATUS，CONNECT OK 以及 TCP/UDP CLOSED 状态下，输入 AT+CGATT=0，则也可以使模块释放上下文，进入 PDP DEACT 状态；

◆模块进入 PDP DEACT 状态，仍需要输入 AT+CIPSHUT，进入 IP INITIAL 状态；

◆模块在各个状态下均可以输入 AT+CIPSHUT，进入 IP INITIAL 状态。

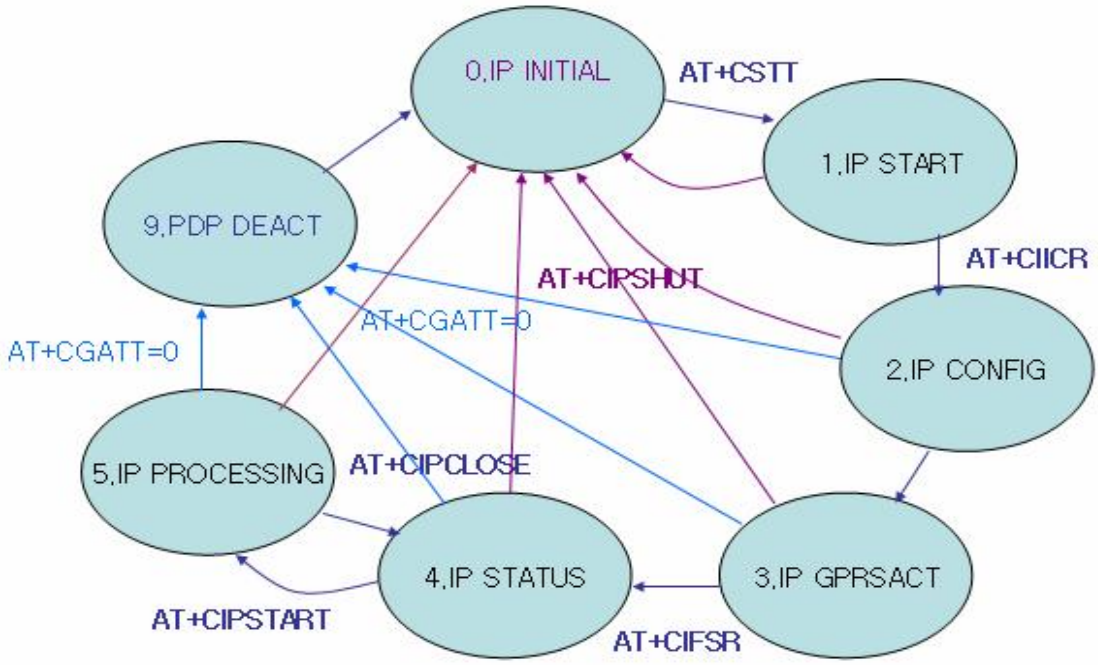
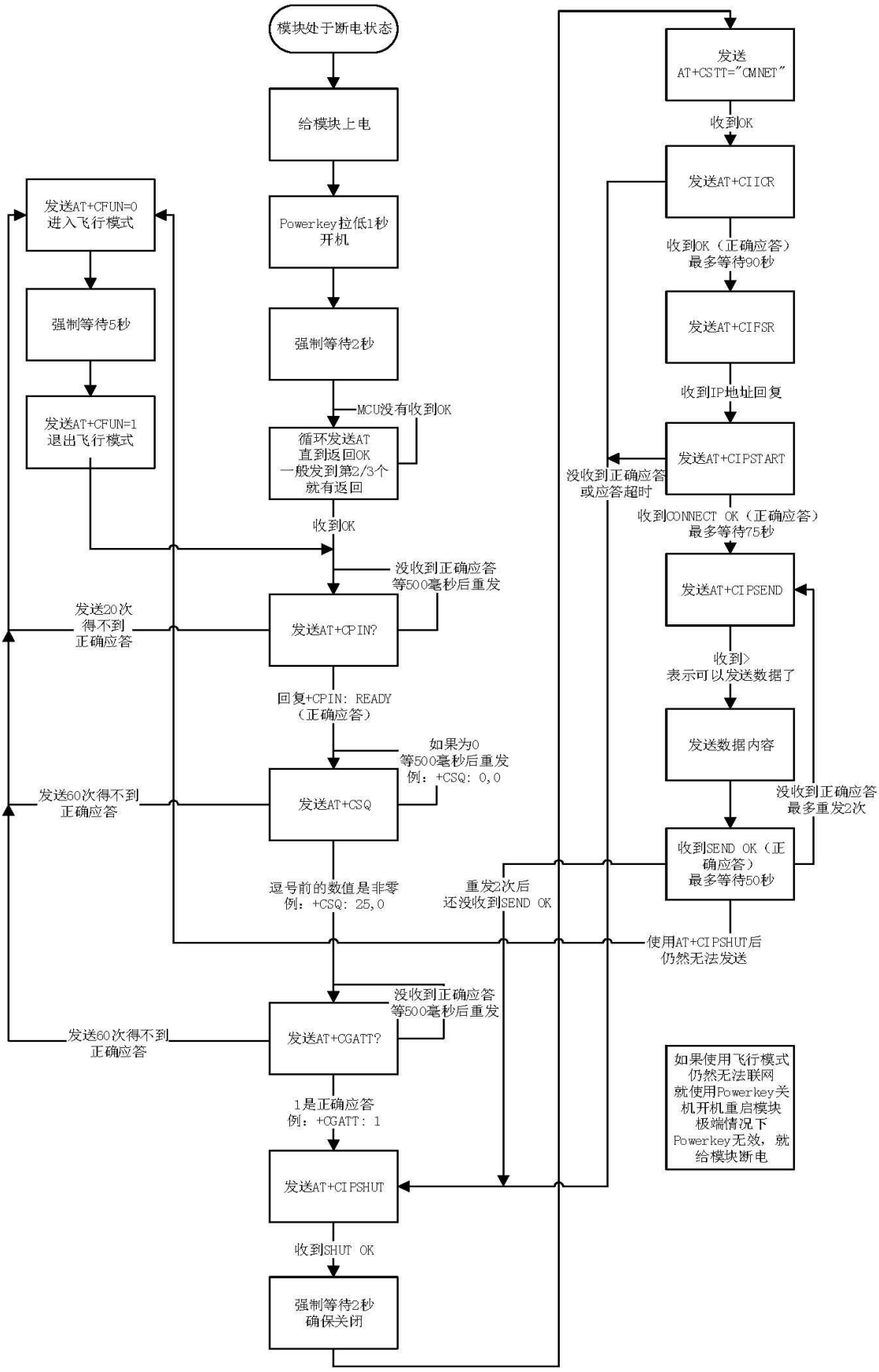


图 2 ： 多链接时的状态机

12.41 模块上电初始化以及 TCPIP 流程

见下图：


模块处于断电状态
给模块上电
Powerkey拉低1秒
开机
强制等待2秒
MCU没有收到OK
循环发送AT
直到返回OK
一般发到第2/3个
就有返回
收到OK
没收到正确应答
等500毫秒后重发
发送AT+CPIN?
发送20次
得不到
正确应答
回复+CPIN: READY
（正确应答）
如果为0
等500毫秒后重发
例：+CSQ: 0,0
发送AT+CSQ
发送60次得不到
正确应答
逗号前的数值是非零
例：+CSQ: 25,0
没收到正确应答
等500毫秒后重发
发送AT+CGATT?
发送60次得不到
正确应答
1是正确应答
例：+CGATT: 1
发送AT+CIPSHUT
收到SHUT OK
强制等待2秒
确保关闭
发送AT+CFUN=0
进入飞行模式
强制等待5秒
发送AT+CFUN=1
退出飞行模式
发送
AT+CSTT="CMNET"
收到OK
发送AT+CIICR
收到OK（正确应答）
最多等待90秒
发送AT+CIFSR
收到IP地址回复
发送AT+CIPSTART
没收到正确应答
或应答超时
收到CONNECT OK（正确应答）
最多等待75秒
发送AT+CIPSEND
收到>
表示可以发送数据了
发送数据内容
没收到正确应答
最多重发2次
收到SEND OK（正
确应答）
最多等待50秒
重发2次后
还没收到SEND OK
使用AT+CIPSHUT后
仍然无法发送
如果使用飞行模式
仍然无法联网
就使用Powerkey关
机开机重启模块
极端情况下
Powerkey无效，就
给模块断电

12.42 使用方法举例

由于这部分的命令具有高度相关性，所以将各个命令的应用例程集中在一起描述。

举例：

命令（→）/返回（←）	实例	解释和说明
TCP非透传应用1：模块做为客户端，单链接，发送数据 （事实上，TCP发送方式有快发和慢发两种，由+CIPQSEND命令来设置发送方式。区别就是：慢发每发送一笔数据需要服务器那边的确认，而快发则发送到模块就可以了，不需要服务器的确认。 慢发可能会出现长时间没响应的情况，AT通道就堵住了，所以建议采用**快发**模式）		
→	AT+CREG?	查询当前GPRS注册状态
←	+CREG: 0,1 OK	<n>=0，表示禁用URC上报 <stat>=1，标识已经注册GPRS网络，而且是本地网
→	AT+CGATT?	查看当前GPRS附着状态
←	+CGATT: 1 OK	<state>=1，标明当前GPRS已经附着
→	AT+CIPMUX=0	设置为单链接模式
←	OK	
→	AT+CIPQSEND=1	设置为快发模式（推荐使用这种模式）
←	OK	
→	AT+CSTT	启动任务，设置APN 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文（此<apn>可以通过AT+CGDCONT?来查询），所以直接输入AT+CSTT即可，模块内部会按照自动获取的<apn>来设置CSTT的APN
←	OK	
→	AT+CIICR	激活移动场景，获取IP地址
←	OK	
→	AT+CIFSR	查询分配的IP地址
←	010.083.172.111	
→	AT+CIPSTATUS	查询下链接状态
←	OK STATE: IP STATUS	

→	AT+CIPSTART="TCP","60.166.18.9",7500 或 AT+CIPSTART="TCP","60.166.18.9","7500"	其中： "TCP"为链接的协议类型 "60.166.18.9" 为对端服务器的IP地址 7500 为对端服务器的TCP端口号 注：CIPSTART设置命令所有的参数，双括号可以用，也可以不用
←	OK	
←(URC)	CONNECT OK	如果链接成功，会有如此URC上报
→	AT+CIPSTATUS	查询下链接状态
←	OK STATE: CONNECT OK	链接建立成功
→	AT+CIPSEND	发送数据（不定长度，手动发送）
→	>1234567890<CTRL-Z>	当出现">"后，输入待发送的数据：0123456789。<CTRL-Z>用来发送数据，发送16进制数0x1A即等同于发送<CTRL-Z>。
←	DATA ACCEPT:10	发送成功而且对端服务器收到数据
→	AT+CIPACK	每发一笔，查询下发送状态，可以知道上笔数据服务器有没有收到
←	+CIPACK: 10,10,0 OK	第一个10，表明已经发送的数据字节数，第二个10表示服务器收到的数据字节数，0表示服务器尚未收到的数据字节数
→	AT+CIPSEND=10	发送数据（确定长度）
→	>1234567890	
←	DATA ACCEPT:10	输入数据达到10个字节，不用发送<CTRL-Z>即十六进制的1A，会自动发送
→	AT+CIPATS=1,10	设置自动发送，自动发送的定时为10S
←	OK	
→	AT+CIPSEND	发送数据
→	>1234567890	
←	DATA ACCEPT:10	10s定时器溢出，不用发送<CTRL-Z>即十六进制的1A，数据会自己发送
→	AT+CIPSEND=100	
→	>123	
←	DATA ACCEPT:3	10s定时器溢出，输入内容不必达到100字节，也不用发送<CTRL-Z>即十六进制的1A，数据会自己发送
→	AT+CIPCLOSE	关闭TCP链接
←	CLOSE OK	关闭成功
→	AT+CIPSTATUS	查询下链接状态

←	OK STATE: TCP CLOSED	TCP链接已经关闭
→	AT+CIPSHUT	关闭移动场景
←	SHUT OK	关闭成功
→	AT+CIFSR	查询当前的模块IP
←	ERROR	IP地址已经没有了
TCP非透传应用2：模块做为客户端，单链接，发送数据，开启SSL功能		
→	AT+CIPMUX=0	设置为单链接模式
←	OK	
→	AT+CIPQSEND=1	发送模式为快发
←	OK	
→	AT+CIPSTATUS	查询链接状态
←	OK STATE: IP INITIAL	
→	AT+CIPSSL=1	打开SSL功能(本例中双方都不需要验证证书)
←	OK	
→	AT+CIPSTART="TCP","60.166.18.9",7500	当模块设置为单链接并且状态为IP INITIAL时，也可以用CIPSTART直接建立连接，不必先输入CSTT CIICR CIFSR 请写实际的服务器地址和端口，不要照抄
←	OK	
←(URC)	CONNECT OK	如果链接成功，会有如此URC上报
→	AT+CIPSEND	发送数据
→	>1234567890<CTRL-Z>	<CTRL-Z>用来发送数据，发送16进制数0x1A即等同于发送<CTRL-Z>。
←	DATA ACCEPT:10	表明模块接收了从TE输入的10个字节的待发数据
	……	+CIPCLOSE,+CIPSHUT，不再赘述
TCP非透传应用3：模块做为客户端，单链接，发送数据，开启SSL功能（双向证书验证）		
→	AT+CIPMUX=0	设置为单链接模式
←	OK	
→	AT+CIPQSEND=1	设置为快发
←	OK	

→	AT+CIPSSL=1	开启SSL功能开关为开
←	OK	
→	AT+FSCREATE="ca.crt"	创建CA 证书文件
←	OK	
→	AT+FSCREATE="client.crt"	创建客户端证书文件
←	OK	
→	AT+FSCREATE="client.key"	创建客户端密钥文件
←	OK	
→	AT+FSWRITE="ca.crt",0,2080,15	文件长度2080只是举例，要根据实际填写。下同。
←	>	这里输入CA证书文件
←	OK	
←	AT+FSWRITE="client.crt",0,128,10	
→	>	这里输入客户端证书文件
←	OK	
→	AT+FSWRITE="client.key",0,188,10	
←	>	这里输入客户端密钥文件
←	OK	
→	AT+SSLCFG="cacert",0,"ca.crt"	设置服务器CA 证书 SSL 上下文id，在单链接的情况下缺省为0
←	OK	
→	AT+SSLCFG="clientcert",0,"client.crt"	设置客户端证书
←	OK	
→	AT+SSLCFG="clientkey",0,"client.key"	设置客户端KEY
←	OK	
→	AT+SSLCFG="seclevel",0,2	设置安全等级
←	OK	
→	AT+SSLCFG="ciphersuite",0,0X0035	设置加密套件
←	OK	
→	AT+SSLCFG="clientrandom",0,101B12C3141516171F19202122232425262728293031323334353637	设置随机数
←	OK	
→	AT+CIPSTART=TCP,tcplab.openluat.com,57513	
←	OK	
←(URC)	CONNECT OK	
→	AT+CIPSEND=10	发送数据（确定长度）
→	>1234567890	
←	DATA ACCEPT:10	输入数据达到10个字节，不用发送<CTRL-Z>

		即十六进制的1A，数据会自动发送
→	AT+CIPSHUT	
←	OK	
→	AT+CIPSSL=0	关闭 SSL 功能
←	OK	
UDP非透传应用1：模块做为客户端，单链接，数据发送		
→	AT+CIPMUX=0	设置为单链接模式
←	OK	
→	AT+CIPQSEND=1	设置为快发模式
←	OK	
→	AT+CSTT	启动任务，设置APN 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文（此<apn>可以通过AT+CGDCONT?来查询），所以直接输入AT+CSTT即可，模块内部会按照自动获取的<apn>来设置CSTT的APN
←	OK	
→	AT+CIICR	激活移动场景，获取IP地址
←	OK	
→	AT+CIFSR	查询分配的IP地址
←	010.083.172.111	
→	AT+CIPSTART="UDP","60.166.18.9",6100	建立UDP链接，其中： "UDP"为链接的协议类型 "60.166.18.9" 为对端服务器的IP地址 6100 为对端服务器的UDP端口号
←	OK	
←(URC)	CONNECT OK	如果链接成功，会有如此URC上报
→	AT+CIPSEND	发送数据
→	>1234567890<CTRL-Z>	<CTRL-Z>即十六进制的1A
←	DATA ACCEPT:10	成功将数据发送到模块
	……	+CIPCLOSE,+CIPSHUT，不再赘述
多连接应用1：模块做为客户端，多链接，发送数据，SSL证书双向认证功能打开（快发） **注：本例是同时链接不同的两个服务器。**		
→	AT+CIPSSL=1	开启SSL功能开关
←	OK	
→	AT+CIPMUX=1	

←	OK	
→	AT+FSCREATE="server1.crt"	创建server1 的CA 证书文件
←	OK	
→	AT+FSCREATE="server2.crt"	创建server2 的CA 证书文件
←	OK	
→	AT+FSCREATE="client.crt"	创建客户端证书文件
←	OK	
→	AT+FSCREATE="client.key"	创建客户端密钥文件
←	OK	
→	AT+FSWRITE="server1.crt",0,2080,15	文件长度2080只是举例，要根据实际填写。下同。
←	>	这里输入服务器1的CA证书文件
←	OK	
→	AT+FSWRITE="server1.crt",0,1962,15	
←	>	这里输入服务器2的CA证书文件
←	OK	
←	AT+FSWRITE="client.crt",0,128,10	
→	>	这里输入客户端证书文件
←	OK	
→	AT+FSWRITE="client.key",0,188,10	
←	>	这里输入客户端密钥文件
←	OK	
→	AT+SSLCFG="cacert",1,"server1.crt"	设置服务器1的CA 证书 多链接的SSL上下文id和多链接的链接id绑定，链接1的SSL上下文id也是1
←	OK	
→	AT+SSLCFG="cacert",2,"server2.crt"	设置服务器2的CA 证书
←	OK	
→	AT+SSLCFG="clientcert",1,"client.crt"	设置SSL上下文1（即链接1）的客户端证书
←	OK	
→	AT+SSLCFG="clientcert",2,"client.crt"	设置SSL上下文2（即链接2）的客户端证书
←	OK	
→	AT+SSLCFG="clientkey", 1,"client.key"	设置SSL上下文1（即链接1）的客户端KEY
←	OK	
→	AT+SSLCFG="clientkey", 2,"client.key"	设置SSL上下文2（即链接2）的客户端KEY

←	OK	
→	AT+SSLCFG="seclevel",1,2	设置SSL上下文1（即链接1）的安全等级：双向认证
←	OK	
→	AT+SSLCFG="seclevel",2,2	设置SSL上下文2（即链接2）的安全等级：双向认证
←	OK	
→	AT+SSLCFG="ciphersuite",1,0X0035	设置SSL上下文1（即链接1）的加密套件
←	OK	
→	AT+SSLCFG="ciphersuite",2,0X0035	设置SSL上下文2（即链接2）的加密套件
←	OK	
→	AT+SSLCFG="clientrandom",1,10B12C3141516171F19202122232425262728293031323334353637D	设置链接1的随机数
←	OK	
→	AT+SSLCFG="clientrandom",2,581B12C3141516171F1920214A23249C26272829303132333435363 2	设置链接2的随机数
←	OK	
→	AT+CIPQSEND=1	设置为快发模式
←	OK	
→	AT+CSTT	设置模块APN 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文(此<apn>可以通过AT+CGDCONT?来查询），所以直接输入AT+CSTT即可，模块内部会按照自动获取的<apn>来设置CSTT的APN
←	OK	
→	AT+CIICR	
←	OK	
→	AT+CIFSR	
←	010.083.172.111	
→	AT+CIPSTART=1,"TCP","60.166.12.210",7500	
←	OK	
←(URC)	1,CONNECT OK	Id=1的 TCP 链接成功建立
→	AT+CIPSTART=2,"TCP","60.166.13.215",6100	
←	OK	
←(URC)	2,CONNECT OK	Id=2的 TCP 链接成功建立
→	AT+CIPSTATUS	查询当前链路状态
←	OK	

	STATE: IP PROCESSING C: 0,,"","","","INITIAL" C: 1,0,"TCP","60.166.12.210","7500","CONNECTED" C: 2,0,"TCP","60.166.12.210","7500","CONNECTED" C: 3,,"","","","INITIAL" C: 4,,"","","","INITIAL" C: 5,,"","","","INITIAL"	
→	AT+CIPSEND=1	在 TCP 1链接上发送数据
→	>1234567890<CTRL-Z>	<CTRL-Z>即十六进制的1A
←	DATA ACCEPT:1,10	
→	AT+CIPSEND=2	在 TCP 2链接上发送数据
→	>1234567890<CTRL-Z>	<CTRL-Z>即十六进制的1A
←	DATA ACCEPT:2,10	
	……	其他发送数据的动作
→	AT+CIPSHUT	如果想关闭所有的连接
←	SHUT OK	
→	AT+CIPMUX=0	如果想关闭多连接模式，则必须在AT+CIPSHUT后才可以
←	OK	
→	AT+CIPSSL=0	关闭SSL功能。
←	OK	

多连接应用2：模块做为客户端，多链接，发送数据，未开启SSL功能

→	AT+CIPMUX=1	设置为多链接模式
←	OK	
→	AT+CIPQSEND=1	设置为快发模式
←	OK	
	……	**仿照TCP非透传应用1的例子，**依次发送CSTT CIICR CIFSR这三个命令
→	AT+CIPSTART=4,"TCP","60.166.12.210",7500	
←	OK	
←(URC)	4, CONNECT OK	Id=6的 TCP 链接成功建立
→	AT+CIPSTART=5,"UDP","60.166.12.210",6100	
←	OK	
←(URC)	5, CONNECT OK	Id=7的 UDP 链接成功建立
→	AT+CIPSEND=4,10	在 TCP 链接上发送数据，定长发送

→	>1234567890	
←	DATA ACCEPT:4,10	
→	AT+CIPSEND=5	在 UDP 链接上发送数据，不定长发送
→	>1234567890<CTRL-Z>	<CTRL-Z>即十六进制的1A
←	DATA ACCEPT:5,10	
→	AT+CIPATS=1,10	设置为自动发送，并且定时器设置为10S，从该命令返回OK时开始计时
←	OK	
→	AT+CIPSEND=4	在id=6的TCP链接上发送数据
→	>TEST Auto fast send	输入待发送的字符串（不必输入<CTRL-Z>即十六进制的1A）
←	DATA ACCEPT:4,19	10S定时器到时，数据自动发送
	……	+CIPCLOSE,+CIPSHUT，不再赘述

单链接非透传时接收数据：

	……	建立一个TCP链接，前面已经详述，在此不再赘述
→	AT+CIPHEAD=1	设置接收数据时，有数据头
←	OK	
←(UCR)	+IPD,4:TEST	模块收到SERVER那边发来的一串字符数据：TEST
→	AT+CIPSHOWTP=1	设置接收数据时，显示数据所使用的协议
←	OK	
←(UCR)	+IPD,4,TCP:TEST	模块收到SERVER那边发来的数据：TEST。所使用的协议也会显示出来。 **如果是UDP链接，则会显示：** **+IPD,4,UDP:TEST**
→	AT+CIPSRIP=1	设置接收数据时显示发送方的IP地址和端口
←	OK	
←(URC)	+RECV FROM: 60.166.12.210:7500 +IPD,4:TEST	接收服务器发来的数据，是一个字符"TEST"，长度为4
→	AT+CIPSHOWTP=0	不在接收数据头中显示协议类型
←	OK	
→	AT+CIPHEAD=0	不显示数据头
←	OK	
→	AT+CIPSRIP=0	不显示数据发送放的IP地址和端口
←	OK	
←(URC)	TEST	此时接收到数据:TEST

多链接非透传时接收数据：		
	……	**仿照TCP非透传应用1的例子，**依次发送 CSTT CIICR CIFSR这三个命令，并建立连接号为0的TCP链接和连接号为1的UDP链接
←(URC)	+RECEIVE,0,7: TEST123	在链接0的TCP链接上收到7个字符的字符串： TEST123
←(URC)	+RECEIVE,1,10: TEST123456	在链接1的UDP链接上收到10个字符的字符串： TEST123456
透明传输应用1：TCP数据传输		
→	AT+CIPMODE=1	设置TCPIP应用为透明传输模式
←	OK	
→	AT+CIPSTART="TCP","60.166.18.9",7500	建立TCP链接，其中： "TCP"为链接的协议类型 "60.166.18.9" 为对端服务器的IP地址 7500 为对端服务器的TCP端口号
←	OK	
←(URC)	CONNECT	如果链接成功，会有如此URC上报
→/←	……	透传开始，在这里输入数据并传送，此时也可以接收服务器传来的数据
→	……+++	如果想返回AT命令模式，则在数据后面输入+++ **注： +++需要满足一定的条件才会被模块认为是escape sequence，否则会被认为是数据:** **1，第一个+之前需要1000ms的间隔** **2，最后一个+之后需要500ms的间隔** **3，三个+之间的间隔不能超过500ms**
←	OK	OK表示已经返回到AT命令模式
→	ATO	ATO用于返回数据模式
←	CONNECT	CONNECT表示进入透传模式
→/←	……	此时又可以开始传输数据
←	TCP ERROR:×× 或 CLOSED	当传输中有协议栈错误发生时，会转入AT命令状态，并上报该错误码
→	AT+CIPSHUT	发生这种情况时，+CIPSHUT关闭连接。 如果没有错误发生，则+++返回AT命令状态，再+CIPSHUT关闭连接
←	SHUT OK	

透明传输应用2：UDP数据传输		
→	AT+CIPMODE=1	设置TCPIP应用为透明传输模式
←	OK	
→	AT+CSTT	设置模块APN 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文（此<apn>可以通过AT+CGDCONT?来查询），所以直接输入AT+CSTT即可，模块内部会按照自动获取的<apn>来设置CSTT的APN
←	OK	
→	AT+CIICR	激活移动场景，获取IP地址
←	OK	
→	AT+CIFSR	查询分配的IP地址
←	010.083.172.111	
→	AT+CIPSTART="UDP","60.166.18.9",6100	建立UDP链接，其中： "UDP"为链接的协议类型 "60.166.18.9" 为对端服务器的IP地址 6100 为对端服务器的UDP端口号
←	OK	
←(URC)	CONNECT	如果链接成功，会有如此URC上报
→/←	……	透传开始，在这里输入数据并传送，此时也可以接收服务器传来的数据
←	UDP ERROR:××	当传输中有协议栈错误发生时，会转入AT命令状态，并上报该错误码
→	AT+CIPSHUT	发生这种情况时，+CIPSHUT关闭连接。 如果没有错误发生，则+++返回AT命令状态，再+CIPSHUT关闭连接
←	SHUT OK	
←	OK	

域名解析：

注：该命令只有在执行完at+cstt、at+ciicr、at+cifsr后才能正常工作，这三条命令的输入方法请参考：TCP非透传应用1的例子

→	AT+CDNSGIP="WWW.SINA.COM.CN"	解析新浪网站的域名
←	OK +CDNSGIP:1,"WWW.SINA.COM.CN","221.179.180.76"	返回IP地址

发送和接收格式总结：

数据发送(多链接模式，以链接号<n>=5 为范例)

	链接协议	快发	慢发
单链接	TCP	AT+CIPSEND >test TCP DATA ACCEPT:8	AT+CIPSEND >test TCP SEND OK
	UDP	AT+CIPSEND >test UDP DATA ACCEPT:8	AT+CIPSEND >test UDP SEND OK
多链接	TCP	AT+CIPSEND=5 >test TCP DATA ACCEPT:5,8	AT+CIPSEND=5 >test TCP 5,SEND OK
	UDP	AT+CIPSEND=5 >test UDP DATA ACCEPT:5,8	AT+CIPSEND=5 >test UDP 5,SEND OK

数据接收(多链接模式以链接号<n>=1 为范例)

	链接协议	AT+CIPHEAD=0	AT+CIPHEAD=1	
			+CIPSHOWTP=0	+CIPSHOWTP=1
单链接	TCP	TEST123	+IPD,7:TEST123	+IPD,7,TCP:TEST123
	UDP	TEST123	+IPD,7:TEST123	+IPD,7,UDP:TEST123
多链接	TCP	+RECEIVE,1,7: TEST123	+RECEIVE,1,7: TEST123	+RECEIVE,1,7: TEST123
	UDP	+RECEIVE,1,7: TEST123	+RECEIVE,1,7: TEST123	+RECEIVE,1,7: TEST123

如果模块出现上下文去激活的主动上报，该如何处理：

←(URC)	+PDP DEACT	PDP去激活，此时需要重新激活一个PDP上下文才能继续使用TCPIP的AT命令
→	AT+CIPSHUT	关闭移动场景
←	OK	
	此时有两种处理方式，见右侧	1） AT+CFUN=0，AT+CFUN=1后，再重新建立连接 2） AT+RESET重启模块后再重新建立连接

13 IP 应用相关命令

13.1 IP 应用设置：AT+SAPBR

语法规则：

命令类型	语法	返回
执行命令	AT+SAPBR=<cmd_type>,<cid>[,<ConParamTag>,<ConParamValue>]	如果<cmd_type> = 2 +SAPBR: <cid>,<Status>,<IP_Addr> OK
		如果<cmd_type> = 4 +SAPBR:<ConParamTag>,<ConParamValue> OK
		其余 OK
测试命令	AT+SAPBR=?	+SAPBR: (0-4),(1-3),"ConParamTag","ConParamValue" OK
URC 上报	+SAPBR <cid>: DEACT	当移动场景去激活时，会有此上报
注意事项	SAPBR设置承载参数APN的时候需要注意以下事项： 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用（此<apn>可以通过AT+CGDCONT?来查询），所以直接输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN	

参数定义：

参数	定义	取值	对取值的说明

<cmd_type>	命令类型	0	关闭承载
		1	打开承载
		2	查询承载状态
		3	设置承载参数
		4	获取承载参数
<cid>	承载上下文标识	1~3	
<Status>	承载的状态	0	正在连接
		1	已经连接
		2	正在关闭
		3	已经关闭
<IP_Addr>	承载IP地址		
<ConParamTag>	承载参数	"CONTYPE"	因特网连接类型。 取值请参考参数<ConParamValue_ConType>
		"APN"	接入点名称，最长支持 50 个字符
		"USER"	用户名称：最长支持 50 个字符
		"PWD"	密码：最长支持 50 个字符
		"PHONENUM"	CSD电话号码
		"RATE"	CSD连接速率。 取值请参考<ConParamValue_Rate>。
<ConParamValue>			
<ConParamValue_ConType>	因特网连接类型	"CSD"	CSD，电路交换数据业务
		"GPRS"	GPRS，通用分组无线业务 注：GPRS只是兼容2G模块指令的输入格式，不会强制切换到GPRS网络上，真实的数据承载网络取决于模块当时注册的网络制式
<ConParamValue_Rate>	CSD连接速率	0	2400
		1	4800
		2	9600
		3	14400

14 HTTP 相关命令

14.1 初始化 HTTP 服务：AT+HTTPINIT

语法规则：

命令类型	语法	返回

执行命令	AT+HTTPINIT	OK
测试命令	AT+HTTPINIT=?	OK
注意事项	1. 在使用HTTP服务前，应该先用AT+HTTPINIT命令来初始化HTTP协议栈 2. 如果使用的是https需要进行证书，加密套件，加密等级等相关的设置请参考SSLCFG指令	

14.2 启用 SSL：AT+HTTPSSL

语法规则：

命令类型	语法	返回
设置命令	AT+HTTPSSL=<n>	OK
读取命令	AT+HTTPSSL?	+ HTTPSSL: <n> OK
测试命令	AT+HTTPSSL=?	+HTTPSSL: (0-1) OK

参数	定义	取值	对取值的说明
<n>	HTTP SSL 功能开关	0	关闭 SSL 功能
		1	开启 SSL 功能

14.3 设置 HTTP 参数值：AT+HTTPPARA

语法规则：

命令类型	语法	返回
设置命令	AT+HTTPPARA=<HTTPParamTag>,<HTTPParamValue>	OK
查询命令	AT+HTTPPARA?	+HTTPPARA: list of <HTTPParamTag>:<HTTPParamValue>) OK
测试命令	AT+HTTPPARA=?	+HTTPPARA: "HTTPParamTag"," HTTPParamValue" OK

参数定义：

参数	定义	取值	对取值的说明
<HTTPParamTag> :HTTP参数，包括：			
"CID"	承载上下文号码(必选参	1~3	

	数)		
"URL"	HTTP 或 HTTPS URL(必选参数) **注：同时支持 HTTPS 和 HTTP URL**	"http://server/path:tcpPort " 或： "https://server/path:tcpPort " 最大长度500个字节	Server: FQDN or IP-address Path: path of file or directory tcpPort: 如果参数省略，将服务连接到 HTTP 默认端口 80。 参考"IETF-RFC 261
"UA"	应用程序必须设置用户代理来识别移动终端。通常操作系统和软件版本信息在设置时都会携带浏览器标识符。	-	默认值为:AM_MODULE
"PROIP"	HTTP 代理服务器的 IP 地址	-	
"PROPORT"	HTTP 代理服务器的 PORT	-	
"REDIR"	作为 HTTP 客户端时用此标志控制重定向机制。如果此标记设置为 1，当服务器发送重定向码(范围 30x)时，客户端自动发送新的 HTTP 请求	-	默认值为 0(无定向)
"BREAK"	HTTP 方法"GET"的参数，整数型	-	获取从断点到结束点的部分数据，注意不是所有的 HTTP 服务器都支持<BREAK>参数。BREAK 最小值是 0。
"BREAKEND"	和"BREAK"一起使用，用于断点续传功能。整数型。	-	如果"BREAKEND"大于"BREAK"，续传的范围从"BREAKEND"到"BREAK"。如果"BREAKEND"小于"BREAK"，续传的范围从"BREAK"到文件结尾。如果"BREAKEND"和"BREAK"均为 0，将不会续传。
"TIMEOUT"	设置 http 请求的超时时间	-	单位是秒，默认 120 秒
"CONTENT"	设置发送的请求实体数据数据类型	-	即 Content-Type
"USER_DEFINED"	用户自定义参数，为了兼容合宙 2G 模块		用户自定义参数的取值。例如：AT+HTTPPARA="USER_DEFINED","Content-type: json-user-define" **注**：如果需要设置多条用户自定义参数，则一条一条地输入。后面输入的不会覆盖以前的。
"USERDATA"	用户自定义参数，作用同"USER_DEFINED"，为了兼容 SIMCOM 模块		用户自定义参数的取值。例如：AT+HTTPPARA="USERDATA","Content-type: json-user-define" **注**:如果想设置多条用户定义参数，则多

			条参数之间可以用\r\n 连接。例如： AT+HTTPPARA="USERDATA","Content-Type:application/json\r\nAPPKEY:FW" 在 MCU 程序中需要将\r\n 写成\\r\\n 值得一提的是有些 PC 串口工具，例如 SSCOM，会将\r 和\n 当做控制字符处理，所以也需要将\r\n 写成\\r\\n 而另外一些工具，例如 XCOM，不会将\r 和\n 当做控制字符处理，所以直接输入\r\n
<HTTPParamValue> ：即<HTTPParamTag>的取值。 注："USER_DEFINED" 和"USERDATA"中内嵌的双引号，用\22表达。			

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+HTTPPARA?	
←	+HTTPPARA: CID: 1 URL: UA: AM_MODULE PROIP: 0.0.0.0 PROPORT: 0 REDIR: 0 BREAK: 0 BREAKEND: 0 TIMEOUT: 120 CONTENT: USERDATA: OK	

14.4 写数据：AT+HTTPDATA

语法规则：

命令类型	语法	返回
设置命令	AT+HTTPDATA=<size>,<time>	DOWNLOAD OK
测试命令	AT+HTTPDATA=?	+HTTPDATA: (<size>取值列表),(<time>取值列表) OK

参数定义：

参数	定义	取值	对取值的说明
<size>	POST 数据的大小	0-3356	AT 版本 最大长度为 3356
		0-130048	LSAT 版本 最大长度为 130048
		0	后面一个参数不判断，相当于把内容清除掉
<time>	输入数据的最长时间	1000-120000	单位：ms
注：强烈建议设置的时间<time>要能够全部输入所有的数据，下载数据的真实大小不能大于<size>			

14.5 HTTP 方式激活：AT+HTTPACTION

语法规则：

命令类型	语法	返回
设置命令	AT+HTTPACTION=<method>	OK 后面紧跟 Unsolicited Result Code: +HTTPACTION: <Method>,<StatusCode>,<DataLen>
		或如果错误与 ME 功能相关，则返回： +CME ERROR: <err> 后面紧跟 Unsolicited Result Code: +HTTPACTION: <Method>,<StatusCode>,<DataLen>
测试命令	AT+HTTPACTION=?	+HTTPACTION: (0-2) OK

参数定义：

参数	定义	取值	对取值的说明
<Method>	HTTP 方法说明	0	GET
		1	POST
		2	HEAD
<DataLen>	得到的数据长度	-	整数型
<StatusCode>	HTTP 状态码，由远端服务器响应， 参考 TTP1.1(RFC2616)	100	继续（Continue）
		101	交换协议(Switching Protocols)
		200	确定(OK)
		201	已创建(Created)

		202	已接受(Accepted)
		203	非权威消息(Non-Authoritative Information)
		204	无内容(No Content)
		205	重置内容(Reset Content)
		206	部分内容(Partial Content)
		300	多重选择(Multiple Choices)
		301	永久删除(Moved Permanently)
		302	找到(Found)
		303	参考其他(See Other)
		304	未修改(Not Modified)
		305	使用代理服务器(Use Proxy)
		307	临时重定向(Temporary Redirect)
		400	错误请求(Bad Request)
		401	未授权(Unauthorized)
		402	付费请求(Payment Required)
		403	禁止(Forbidden)
		404	找不到(Not Found)
		405	方法不被允许(Method not Allowed)
		406	不可接受(Not Acceptable)
		407	要求进行代理身份认证(Proxy AuthenticationRequired)
		408	请求超时 (Request Time-out)
		409	冲突(Conflict)
		410	所请求资源不在服务器上有效，且不知道转发地址(Gone)
		411	需要输入长度(Length Required)
		412	前提条件失败 (Precondition Failed)
		413	请求实体太大(Request Entity Too Large)
		414	请求URI太长(Request-URI Too Large)
		415	媒体类型不支持(Unsupported Media Type)
		416	所请求的范围无法满足(Requested range notsatisfiable)
		417	执行失败(Expectation Failed)
		500	内部服务器错误(Internal Server Error)
		501	未执行 (Not Implemented)
		502	网关错误(Bad Gateway)
		503	服务不可用(Service Unavailable)
		504	网关超时(Gateway Time-out)
		505	HTTP 版本不支持(HTTP Version not supported)
		600	非 HTTP PDU 格式(Not HTTP PDU)
		601	网络错误(Network Error)

		602	内存不足(No memory)
		603	DNS 错误(DNS Error)
		604	栈忙(Stack Busy)
		605	SSL建立通道失败
		606	SSL通讯警告错误

14.6 HTTP 方式激活(扩展)：AT+HTTPEXACTION

注 ：本命令 EC618 系列模块>=V1106 版本支持

语法规则：

命令类型	语法	返回
设置命令	AT+HTTPEXACTION=<method>[,<len>]	OK
		或如果错误与 ME 功能相关，则返回： +CME ERROR: <err>
测试命令	AT+HTTPEXACTION=?	+HTTPEXACTION: (0-1) OK

参数定义：

参数	定义	取值	对取值的说明
<Method>	HTTP 方法说明	0	GET
		1	POST
<len>	POST 的长度	-	整数型

14.7 查询 HTTP 服务响应：AT+HTTPREAD

语法规则：

命令类型	语法	返回
设置命令	AT+HTTPREAD=<start_address>,<byte_size>	+HTTPREAD:<date_len> <data> OK
执行命令	AT+HTTPREAD	+HTTPREAD:<date_len> <data> OK 读取 AT+HTTPACTION=0 或 AT+HTTPDATA 命令的所有响

		应数据。 执行命令用来将 HTTP 服务器的响应输出到 UART 或者输出准备好 POST 到服务器的数据。
测试命令	AT+HTTPREAD=?	+HTTPREAD: (list of supported <start_address>s),(list of supported<byte_size>s) OK

参数定义：

参数	定义	取值	对取值的说明
<date_len>	实际输出数据长度		
<data>	HTTP 服务器对AT+HTTPACTION=0命令的响应数据		
<start_address>	输出数据的起点	0~3356	单位:字节
<byte_size>	输出数据的长度	1~3356	单位:字节

14.8 获取 HTTP 服务响应数据：AT+HTTPGET

语法规则：

命令类型	语法	返回
设置命令	AT+HTTPGET=<path>	+HTTPGET:<date_len> OK

参数定义：

参数	定义	取值	对取值的说明
<date_len>	实际输出数据长度		
<path>	http 下载文件路径		长度不超过 255 的 ASCII 字符串。

14.9 获取 HTTP 服务响应数据(扩展)：AT+HTTPEXGET

注 ：本命令 EC618 系列模块>=V1106 版本支持

语法规则：

命令类型	语法	返回
设置命令	AT+HTTPEXGET[=<len>]	+HTTPGET:<date_len> date OK

参数定义：

参数	定义	取值	对取值的说明
<len>	GET数据长度		
<date_len>	实际输出数据长度		

14.10 推送 HTTP 数据(扩展)：AT+HTTPEXPOST

注 ：本命令 EC618 系列模块>=V1106 版本支持

语法规则：

命令类型	语法	返回
设置命令	AT+HTTPEXPOST=<len>[,<timeout>]	> OK +HTTPEXPOST: len

参数定义：

参数	定义	取值	对取值的说明
<len>	POST数据的大小	不能超过httpexaction里设置的长度	长度和不能超过httpexaction里设置的长度
<timeout>	输入数据的最长时间	1-120000	单位：ms

14.11 下载文件并保存到文件系统中：AT+HTTPGETTOFS

注 ：本命令 EC618 系列模块>=V1148 版本支持

语法规则：

命令类型	语法	返回	
设置命令	AT+HTTPGETTOFS=<loc>,<filename>	OK	成功
		ERROR	失败
查询命令	AT+HTTPGETTOFS?	+HTTPGETTOFS: <status>,<writelen>,<totalLength> OK	
URC 上报	+HTTPGETTOFS: <响应码>,<writelen>	+HTTPGETTOFS: <HTTP响应码>，<writelen>	
测试命令	AT+HTTPGETTOFS=?	OK	
注意事项	如果两次下载都用同一个文件名，上次下载的内容会被覆盖		

参数定义：

参数	定义	取值	对取值的说明

<status>	工作模式	0	不在 HTTPGETTOFS 过程
		1	处于 HTTPGETTOFS 过程
<loc>	文件保存的位置。	0	保存于 ROM，文件夹固定为"/USER/HTTP"
<filename>	文件名		字符串型，最长 64 个字符
<writelen>	当前保存到文件系统多少数据		
<totalLength>	总共保存到文件系统多少数据		
<error>	错误码		HTTP 错误码

14.12 查询 HTTP 头信息：AT+HTTPHEAD

语法规则：

命令类型	语法	返回
执行命令	AT+HTTPHEAD	+HTTPREAD:<date_len> <data> OK
测试命令	AT+HTTPHEAD=?	OK

参数定义：

参数	定义	取值	对取值的说明
<date_len>	头信息的实际长度		
<data>	头信息		

14.13 保存 HTTP 应用上下文：AT+HTTPSCONT

执行命令保存包含 AT 命令参数的 HTTP 应用上下文，当系统重启时，参数将自动载入。
查询命令返回 HTTP 应用上下文。

语法规则：

命令类型	语法	返回
执行命令	AT+HTTPSCONT	+HTTPREAD: (list of supported <start_address>s),(list of supported<byte_size>s) OK
查询命令	AT+HTTPSCONT?	+HTTPSCONT:<mode> CID:<value> URL: <value> UA: <value> PROIP: <value>

		PROPORT: <value> REDIR: <value> BREAK: <value> BREAKEND: <value> USERDATA:<data> OK

参数定义：

参数	定义	取值	对取值的说明
<mode>	HTTP 上下文保存模式	0	保存，值取自 NVRAM
		1	未保存，值取自 RAM

14.14 终止 HTTP 任务：AT+HTTPTERM

语法规则：

命令类型	语法	返回
设置命令	AT+HTTPTERM	OK
测试命令	AT+HTTPTERM=?	OK

14.15 HTTP 错误码：ERROR：<err code>

HTTP 错误码会以 ERROR:<err code>形式上报。

<err code>定义：

取值	英文说明	中文说明
0	Unknown session id	未知的会话 ID
1	File is too short	文件的内容太短
2	DNS is fail	域名解析失败
3	HTTP is busy	HTTP 任务正忙
4	Socket is wrong	套接字失败
5	Connect fail	连接失败
6	File is error	文件错误
7	Connection is closed	连接已关闭
8	Connection is destroyed	连接已销毁
9	HTTP header is not found	HTTP 头不存在
10	HTTP authentication scheme is not supported	HTTP 认证机制不支持
11	PDP active is wrong	PDP 激活失败
12	Param is wrong	参数有误
13	No buffer	缓冲区不足

14	PDP deactive is wrong	PDP 去激活失败

14.16 使用方法举例

由于这部分的命令具有高度相关性，所以将各个命令的应用例程集中在一起描述。

举例：

命令（→）/返回（←）	实例	解释和说明
HTTP GET 命令使用方法：		
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	设置HTTP功能的承载类型
←	OK	
→	AT+SAPBR=3,1,"APN",""	设置pdp承载参数之APN 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用（此<apn>可以通过AT+CGDCONT?来查询），所以输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN
←	OK	
→	AT+SAPBR=1,1	激活该承载的GPRS PDP上下文
←	OK	
→	AT+SAPBR=2,1	查询下承载的状态
←	+SAPBR: 1,1,010.169.179.213 OK	第一个参数1表示cid 第二个参数1表示已经连接 第三个参数表示模块获取的IP地址
→	AT+HTTPINIT	HTTP协议栈初始化
←	OK	
→	AT+HTTPPARA="CID",1	设置HTTP会话参数：CID
←	OK	
→	AT+HTTPPARA="URL","airtest.openluat.com"	设置HTTP会话参数：URL
←	OK	
→	AT+HTTPACTION=0	GET 开始
←	OK	
←	+HTTPACTION: 0,200,285	出现这些URC上报表明GET数据成功，等待READ
→	AT+HTTPREAD	读取从HTTP 服务器GET的数据
←	+HTTPREAD: 285 ……	…..表示HTTP数据

	OK	
→	AT+HTTPTERM	结束HTTP服务
←	OK	
HTTP POST 命令使用方法：		
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	设置HTTP功能的承载类型 SAPBR是为了激活PDP上下文
←	OK	
→	AT+SAPBR=3,1,"APN",""	设置 APN 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用（此<apn>可以通过AT+CGDCONT?来查询），所以输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN
←	OK	
→	AT+SAPBR=1,1	激活该承载的GPRS PDP上下文
←	OK	
→	AT+SAPBR=2,1	查询下承载的状态
←	+SAPBR: 1,1,010.169.179.213 OK	第一个参数1表示cid 第二个参数1表示已经连接 第三个参数表示模块获取的IP地址
→	AT+HTTPINIT	HTTP协议栈初始化
←	OK	
→	AT+HTTPSSL=1	设置使用SSL连接(不需要验证证书)
←	OK	
→	AT+HTTPPARA="CID",1	设置HTTP会话参数：CID
←	OK	
→	AT+HTTPPARA="URL","http://httpbin.org/post"	设置HTTP会话参数：URL
←	OK	
→	AT+HTTPDATA=4,100000	输入 4 个字节，等待时间是 10S,输入时间要设定的足够大以保证数据输入
← → ←	DOWNLOAD 中国 OK	DOWNLOAD 表示已经准备好输入数据 输入中国（中国两个字占用 4 个字节） OK 出现表示输入结束
→	AT+HTTPACTION=1	POST 开始
←	OK	
←	+HTTPACTION: 1,200,389	表示POST成功
→	AT+HTTPREAD	读取内容

←	+HTTPREAD: 389
	{
	"args": {},
	"data":
	"data:application/octet-stream; base64,1tC5
	+g==",
	"files": {},
	"form": {},
	"headers": {
	"Accept": "*/*",
	"Content-Length": "4",
	"Host": "httpbin.org",
	"User-Agent": "AM_MODULE",
	"X-Amzn-Trace-Id":
	"Root=1-64b780a7-55229ee24bfef1ca66010
	740"
	},
	"json": null,
	"origin": "122.97.220.43",
	"url": "http://httpbin.org/post"
	}
	
	OK	

→	AT+HTTPTERM	结束HTTP服务
←	OK	下载HTTP会话从AT+SAPBR=2,1开始，如果此命令查询IP地址还在，则HTTPINIT HTTPPARA依次执行下去
带SSL证书验证功能的HTTPS过程：		
→	AT+FSCREATE="ca.crt"	创建服务器端CA 证书文件
←	OK	
→	AT+FSCREATE="client.crt"	创建客户端证书文件
←	OK	
→	AT+FSCREATE="client.key"	创建客户端密钥文件
←	OK	
→	AT+FSWRITE="ca.crt",0,2080,15	文件长度2080字节只是举例，要根据实际填写。下同。
←	>	这里输入CA证书文件
←	OK	
←	AT+FSWRITE="client.crt",0,128,10	
→	>	这里输入客户端证书文件
←	OK	
→	AT+FSWRITE="client.key",0,188,10	

←	>	这里输入客户端密钥文件
←	OK	
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	
←	OK	
→	AT+SAPBR=3,1,"APN",""	设置PDP承载之APN参数 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用（此<apn>可以通过AT+CGDCONT?来查询），所以输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN
←	OK	
→	AT+SAPBR=1,1	
←	OK	
→	AT+SAPBR=2,1	
←	+SAPBR: 1,1,010.169.179.213 OK	
→	AT+HTTPSSL=1	开启SSL功能开关为开
←	OK	
→	AT+SSLCFG="cacert",153,"ca.crt"	设置服务器CA 证书 SSL 上下文id，在TCP单链接的情况下缺省为0； 在HTTPS链接下为153，下同
←	OK	
→	AT+SSLCFG="clientcert",153,"client.crt"	设置客户端证书
←	OK	
→	AT+SSLCFG="clientkey",153,"client.key"	设置客户端KEY
←	OK	
→	AT+SSLCFG="seclevel",153,2	设置安全等级
←	OK	
→	AT+SSLCFG="ciphersuite",153,0X0035	设置加密套件
←	OK	
→	AT+SSLCFG="clientrandom",153,01B12C31 41516171F19202122232425262728293031 323334353637D	设置随机数
←	OK	
→	AT+HTTPINIT	HTTP协议栈初始化
←	OK	
→	AT+HTTPPARA="CID",1	设置HTTP会话参数：CID
←	OK	

→	AT+HTTPPARA="URL","https://**.***.***"	设置HTTP会话参数：URL 请写具体的网址，而不要照抄
←	OK	
→	AT+HTTPACTION=0	GET 开始
←	OK	
←	+HTTPACTION:0,200,1348 +HTTPACTION:0,200,1348 +HTTPACTION:0,200,1348 … …	出现这些URC上报表明GET数据成功，等待READ
→	AT+HTTPREAD	读取从HTTP 服务器GET的数据
←	+HTTPREAD:1592 …… OK	…..表示HTTP数据
→	AT+HTTPTERM	结束HTTP服务
←	OK	
HTTP HEAD 过程：		
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	
←	OK	
→	AT+SAPBR=3,1,"APN",""	
←	OK	
→	AT+SAPBR=1,1	
←	OK	
→	AT+SAPBR=2,1	
←	+SAPBR: 1,1,010.169.179.213 OK	
→	AT+HTTPINIT	
←	OK	
→	AT+HTTPPARA="CID",1	
←	OK	
→	AT+HTTPPARA="URL","https://**.***.***"	设置HTTP会话参数：URL 请写具体的网址，而不要照抄
←	OK	
→	AT+HTTPACTION=2	AT+HTTPACTION=0也可以
←	OK	
→	AT+HTTPHEAD	
←	+HTTPHEAD: <date_len>	<date_len>是具体的数字，---data---是具体的头信

	---data--- OK	息
HTTP　HTTPEXACTION&HTTPEXGET&HTTPEXPOST 过程： 注 ：本命令 EC618系列模块 >=V1106版本支持		
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	
←	OK	
→	AT+SAPBR=3,1,"APN",""	
←	OK	
→	AT+SAPBR=1,1	
←	OK	
→	AT+SAPBR=2,1	
←	+SAPBR: 1,1,010.169.179.213 OK	
→	AT+HTTPINIT	
←	OK	
→	AT+HTTPPARA="CID",1	
←	OK	
→	AT+HTTPPARA="URL","http://**.***.***"	设置HTTP会话参数：URL 请写具体的网址，而不要照抄
←	OK	
→	AT+HTTPEXACTION=1,10	HTTP POST, 10 字节
←	OK +HTTPEXPOST	
→	AT+HTTPEXPOST=10,10000	等待 10 字节数据，超时时间 10s
← → ←	> 0123456789 OK	>表示已经准备好输入数据 输入 0123456789 OK 出现表示输入结束
←	+HTTPEXPOST: 10	POST 成功
←	+HTTPEXGET	服务器下发 HTTP 响应
→	AT+HTTPEXGET	读取 HTTP 响应
←	+HTTPEXGET: <date_len> ---data--- OK	<date_len>是具体的数字，---data---是具体的数据
←	+HTTPEXACTION: 1,200	HTTP 响应已读完，会话结束
→	AT+HTTPTERM	
←	OK	
HTTP　HTTPGETTOFS 过程：		

注 ：本命令 EC618 系列模块 >=V1148 版本支持		
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	
←	OK	
→	AT+SAPBR=3,1,"APN",""	
←	OK	
→	AT+SAPBR=1,1	
←	OK	
→	AT+SAPBR=2,1	
←	+SAPBR: 1,1,"10.74.31.148" OK	
→	AT+HTTPINIT	
←	OK	
→	AT+HTTPPARA="CID",1	
←	OK	
→	AT+HTTPPARA="URL","http://**.***.***"	设置HTTP会话参数：URL 请写具体的网址，而不要照抄
←	OK	
→	AT+HTTPGETTOFS=0,"test"	//会创建一个路径是/USER/HTTP/test 的文件
←	+HTTPGETTOFS: 200,39968	//下载完成的上报，200 是 http 状态码，39968 是下载的总长度
→	AT+FSLS=/USER/HTTP/	查询保存的文件
←	test OK	

15 FTP 相关命令

15.1 设置 FTP 控制端口：AT+FTPPORT

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPPORT=<value>	OK	正常返回
		ERROR	输入格式有误
查询命令	AT+FTPPORT?	+FTPPORT:<value>	

		OK	
测试命令	AT+FTPPORT=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<value>	FTP 控制端口号	1~65535	缺省值为 21

15.2 设置 FTP 主动或被动模式：AT+FTPMODE

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPMODE=<value>	OK	正常返回
		ERROR	输入格式有误
查询命令	AT+FTPMODE?	+FTPMODE:<value> OK	
测试命令	AT+FTPMODE=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<value>	FTP 主动被动模式	0	主动模式（暂不支持）
		1	被动模式

15.3 设置 FTP 数据传输类型：AT+FTPTYPE

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPTYPE=<value>	OK	正常返回
		ERROR	输入格式有误
查询命令	AT+FTPTYPE?	+FTPTYPE:<value> OK	
测试命令	AT+FTPTYPE=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<value>	FTP 数据传输类型	"A"	FTP ASCII 字符集

		"I"	FTP Binary 字符集

15.4 设置 FTP 输入类型：AT+FTPPUTOPT

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPPUTOPT=<value>	OK	正常返回
		ERROR	输入格式有误
查询命令	AT+FTPPUTOPT?	+FTPPUTOPT:<value> OK	
测试命令	AT+FTPPUTOPT=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<value>	FTP 数据传输类型	"APPE"	添加文件
		"STOU"	存储唯一的文件
		"STOR"	存储文件

15.5 设置 FTP 承载标识：AT+FTPCID

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPCID=<value>	OK	正常返回
		ERROR	输入格式有误
查询命令	AT+FTPCID?	+FTPCID:<value> OK	
测试命令	AT+FTPCID=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<value>	FTP 承载标识	1~3	同+SAPBR 中<cid>定义。缺省值为 1。

15.6 设置 FTP 下载续传：AT+FTPREST

语法规则：

命令类型	语法	返回	说明

设置命令	AT+FTPREST=<value>	OK	正常返回
		ERROR	输入格式有误
查询命令	AT+FTPREST?	+FTPREST:<value> OK	
测试命令	AT+FTPREST=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<value>	要续传的断点		

15.7 设置 FTP 服务器地址：AT+FTPSERV

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPSERV=<value>	OK	正常返回
		ERROR	输入格式有误
查询命令	AT+FTPSERV?	+FTPSERV:<value> OK	
测试命令	AT+FTPSERV=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<value>	FTP 服务器地址， IP 或域名		32-bit 十进制数，用 . 分开，形式如：xxx.xxx.xxx.xxx。 如果是 DNS，则为长度不超过 49 的 ASCII 字符串。

15.8 设置 FTP 用户名称：AT+FTPUN

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPUN=<value>	OK	正常返回
		ERROR	输入格式有误
查询命令	AT+FTPUN?	+FTPUN:<value> OK	
测试命令	AT+FTPUN=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<value>	FTP 用户名		为长度不超过 49 的 ASCII 字符串。

15.9 设置 FTP 密码：AT+FTPPW

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPPW=<pw>	OK	正常返回
		ERROR	输入格式有误
查询命令	AT+FTPPW?	+FTPPW:<pw> OK	
测试命令	AT+FTPPW=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<pw>	FTP 密码		为长度不超过 49 的 ASCII 字符串。

15.10 设置 FTP 下载文件名称：AT+FTPGETNAME

设置服务器端目标文件的名称。

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPGETNAME=<name>	OK	正常返回
		ERROR	输入格式有误
查询命令	AT+FTPGETNAME?	+FTPGETNAME:<name> OK	
测试命令	AT+FTPGETNAME=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<name>	FTP 下载文件名称		为长度不超过 99 的 ASCII 字符串。

15.11 设置 FTP 下载文件路径：AT+FTPGETPATH

本命令是设置目标文件在服务器端的路径。

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPGETPATH=<value>	OK	正常返回
		ERROR	输入格式有误
查询命令	AT+FTPGETPATH?	+FTPGETPATH:<value> OK	
测试命令	AT+FTPGETPATH=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<value>	FTP 下载文件路径		长度不超过 255 的 ASCII 字符串。

15.12 设置 FTP 上传文件名称：AT+FTPPUTNAME

设置将文件上传到服务器后的保存名称。

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPPUTNAME=<value>	OK	正常返回
		ERROR	输入格式有误
查询命令	AT+FTPPUTNAME?	+FTPPUTNAME:<value> OK	
测试命令	AT+FTPPUTNAME=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<value>	FTP 上传文件名称		长度不超过 99 的 ASCII 字符串。

15.13 设置 FTP 上传文件路径：AT+FTPPUTPATH

设置将文件上传到服务器后的保存目录。

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPPUTPATH=<value>	OK	正常返回
		ERROR	输入格式有误
查询命令	AT+FTPPUTPATH?	+FTPPUTPATH:<value> OK	
测试命令	AT+FTPPUTPATH=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<value>	FTP 上传文件路径		长度不超过 99 的 ASCII 字符串。

15.14 远程服务器上创建文件目录：AT+FTPMKD

语法规则：

命令类型	语法	返回	说明
执行命令	AT+FTPMKD	OK +FTPMKD: 1,0	创建成功
		OK +FTPMKD: 1,<error>	创建失败
		+CME ERROR: <err>	如果是命令错误
测试命令	AT+FTPMKD=?	OK	
注意事项	执行命令创建的文件目录由命令AT+FTPGETPATH定义		

参数定义：

参数	定义	取值	对取值的说明
<error>	错误码		与 AT+FTPGET 命令的<error>定义相同

15.15 远程服务器上删除文件目录：AT+FTPRMD

语法规则：

命令类型	语法	返回	说明
执行命令	AT+FTPRMD	OK +FTPRMD: 1,0	删除成功
		OK +FTPRMD: 1,<error>	删除失败

		+CME ERROR: <err>	如果是命令错误
测试命令	AT+FTPRMD=?	OK	
注意事项	执行命令删除的文件目录由命令AT+FTPGETPATH定义		

参数定义：

参数	定义	取值	对取值的说明
<error>	错误码		与 AT+FTPGET 命令的<error>定义相同

15.16 下载文件：AT+FTPGET

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPGET=<mode>[,<reqlength>]	OK	输入AT+FTPGET=1 的返回
		+FTPGET:2,<cnlength> ……..数据……. OK	输入： AT+FTPGET=2, <reqlength>的返回
URC 上报	+FTPGET:1,1	输入AT+FTPGET=1后，有此上报，表示有数据了，第一个参数1表示<mode>为1	
	+FTPGET:1,<error>	输入AT+FTPGET=1后，有此上报，表示FTP下载失败，第一个参数1表示<mode>为1	
	+FTPGET:1,0	表示数据传输结束，第一个参数1表示<mode>为1	
测试命令	AT+FTPGET=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<mode>	工作模式	1	打开 FTP 会话
		2	读入 FTP 下载数据
<reqlength>	请求读入的数据长度	1~1460	
<cnlength>	确认读入的数据长度	1~1460	可能小于<reqlength>。0 表示没有数据可以读入。
<error>	错误码	61	网络错误 net error
		62	DNS 错误 DNS error
		63	连接错误 connect error
		64	超时 timeout
		65	服务器错误 server error
		66	操作禁止 operation not allowed

		70	应答错误 reply error
		71	用户错误 user error
		72	口令错误 password error
		73	类型错误 type error
		74	保持错误 rest error
		75	被动错误 passive error
		76	主动错误 active error
		77	操作错误 operate error
		78	上传错误 upload error
		79	下载错误 download error
		80~84	FTP SSL 连接错误
		85	文件错误
		86	主动退出

15.17 上传文件：AT+FTPPUT

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPPUT=<mode>[,<reqlength>]	OK	输入AT+FTPPUT=1的返回
		+FTPPUT:2,<cnlength> …….. //input data here OK	输入： AT+FTPPUT=2, <reqlength> 的返回
		OK +FTPPUT: 1,0	输入： AT+FTPPUT=2,0 的返回
URC 上报	+FTPPUT:1,1,<maxlength>	输入AT+FTPPUT=1后，有此上报，表示可以上传数据了 此时通过输入AT+FTPPUT=2,<reqlength>上传数据 第一个参数1表示<mode>为1	
	+FTPPUT:1,0	表示数据传输会话结束。第一个参数1表示<mode>为1	
	+FTPPUT:1,<error>	如果<mode>=1，而且FTP会话失败。错误码<error>定义请参考AT+FTPGET 错误码<error>定义	
测试命令	AT+FTPPUT=?	OK	

参数定义：

参数	定义	取值	对取值的说明
<mode>	FTP 上传工作模式	1	打开 FTP 会话
		2	写 FTP 上传数据

<reqlength>	请求上传的数据长度	0-<maxlength>	
<cnlength>	确认可以上传的数据长度		
<maxlength>	单次可以上传的最大的长度，取决于网络状态		

15.18 下载文件(扩展)：AT+FTPEXTGET

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPEXTGET=<mode>	OK	当<mode>=0或1
	AT+FTPEXTGET=<mode>,<filename>	+FTPEXTGET:2,<totalLength> OK	当<mode>=2
	AT+FTPEXTGET=<mode>,<readpos>,<readlen>	+FTPEXTGET:3,<outputLength> …… //这里是输出到串口的数据 OK	当<mode>=3
	AT+FTPEXTGET=<mode>	OK +FTPEXTGET: 4,<outputLength> //outputLength数据长度	当<mode>=4 AT+FTPEXTGET=4,outputLength //读数据
查询命令	AT+FTPEXTGET?	+FTPEXTGET: <status>[,<receivedLength>] OK	
测试命令	AT+FTPEXTGET=?	OK	
URC 上报	+FTPEXTGET:1,0	<mode>=1且FTPEXTGET结束，会有如此上报	
	+FTPEXTGET:1,<error>	<mode>=1且FTPEXTGET出错，会有如此上报。错误码<error>定义请参考AT+FTPGET 错误码<error>定义	
	+FTPEXTGET:2,<totalLength>	<mode>=2	
	+FTPEXTGET:3,<outputLength>	<mode>=3	
	+FTPEXTGET:4,<outputLength>	<mode>=4	
注意事项	1） 当FTPEXTPUT<mode>=1时，不可使用本命令 2） 如果文件大小(<receivedLength>)<300KB，可以使用这个命令；如果文件大小(<receivedLength>)>=300KB，请使用缺省的FTPGET method (AT+FTPEXTGET=0) 3） 本命令的使用方法请参考本章最后一部分：使用方法举例 4） <mode>=4仅适用于合宙 4G CAT1 模块（Air780E /Air600E系列）>=V1106版本支持		

参数定义：

参数	定义	取值	对取值的说明
<mode>	工作模式	0	使用缺省的 FTPGET 方法
		1	使用扩展的 FTPGET 方法
		2	保存下载的数据到文件中
		3	输出下载的数据到串口
		4	流式获取数据
<filename>	文件名		字符串型，最长 64 个字符。 注:只需指定文件名，不要指定路径，因为保存的路径是确定的 C:/USER/FTP
<totalLength>	保存到文件的所有数据长度，用于 mode 2	<302512	单位：字节
<outputLength>	输出到串口的数据长度，用于 mode 3 或 4	<302512	单位：字节
<readpos>	读取文件数据的起始位置，用于 mode 3		0
<readlen>	读取长度，用于 mode 3		单位：字节
<status>	FTPEXTGET 的状态	0	不在 FTPEXTGET 过程中
		1	处于 FTPEXTGET 过程
<receivedLength>	下载的数据的长度		单位：字节
<error>	错误码		与 AT+FTPGET 命令的<error>定义相同

15.19 上传文件(扩展)：AT+FTPEXTPUT

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPEXTPUT=<mode>[,<pos>,<len>,<timeout>]	OK	当<mode>=0或1
		+FTPEXTPUT: <pos>,<len> ..... //这里从串口输入数据 OK	当<mode>=2
测试命令	AT+FTPEXTPUT=?	OK	
注意事项	本命令的使用方法请参考本章最后一部分：使用方法举例		

参数定义：

参数	定义	取值	对取值的说明
<mode>	工作模式	0	使用缺省的 FTPPUT 方法

		1	使用扩展的 FTPPUT 方法
		2	将数据 PUT 到 RAM
<pos>	每次 PUT 的起始位置	0-300K	单位：byte
<len>	每次 PUT 的数据长度	0-300K	单位：byte
<timeout>	串口数据传输的超时时间	1000-1000000	单位：ms

15.20 下载文件并保存到文件系统中：AT+FTPGETTOFS

语法规则：

命令类型	语法	返回	
设置命令	AT+FTPGETTOFS=<loc>,<filename>[,<num>,<time>]	OK	成功
		ERROR	失败
查询命令	AT+FTPGETTOFS?	+FTPGETTOFS: <status>[,<rcvlen>,<writelen>] OK	
URC 上报	+FTPGETTOFS:0,<totalLength>	如果下载成功	
	+FTPGETTOFS:<error>	如果下载失败	
测试命令	AT+FTPGETTOFS=?	OK	
注意事项	重连以后会断点续传 如果两次下载都用同一个文件名，上次下载的内容会被覆盖		

参数定义：

参数	定义	取值	对取值的说明
<status>	工作模式	0	不在 FTPGETTOFS 过程
		1	处于 FTPGETTOFS 过程
<loc>	文件保存的位置。	0	保存于 ROM，文件夹固定为"C:/USER/FTP"
<filename>	文件名		字符串型，最长 64 个字符
<num>	自动重连次数	0-255	缺省值：3
<time>	等多少秒开始自动重连	0-60	单位：秒，缺省值：5
<rcvlen>	当前从 FTP 服务器 GET 多少数据		
<writelen>	当前保存到文件系统多少数据		
<totalLength>	总共保存到文件系统多少数据		
<error>	错误码		与 AT+FTPGET 命令的<error>定义相同

15.21 从文件系统上传文件到服务器：AT+FTPPUTFRMFS

语法规则：

命令类型	语法	返回
执行命令	AT+FTPPUTFRMFS=<filepath>[,<num>,<time>]	OK
查询命令	AT+FTPPUTFRMFS?	+FTPPUTFRMFS: <status>[,<putLength>] OK
测试命令	AT+FTPPUTFRMFS=?	OK
URC 上报	+FTPPUTFRMFS:0,<totalLength>	当上传结束，有此上报
	+FTPPUTFRMFS:<error>	当上传出错，有此上报。 <error>定义请参考AT+FTPGET命令的<error>定义
注意事项	自动重连后从断点开始续传	

参数定义：

参数	定义	取值	对取值的说明
<filepath>	文件名		字符串型，由数字和字母组成，最长 128 字节
<num>	自动重连次数	0-255	缺省值：3
<time>	等多少秒开始自动重连	0-60	单位：秒，缺省值：5
<status>	FTP 上传状态	0	不在上传状态
		1	处于上传状态
<putLength>	当前从文件系统上传了多少数据		单位：字节
<totalLength>	从文件系统共上传了多少数据		单位：字节

15.22 从文件系统加载到 RAM 中并用 FTPPUT 上传：AT+FTPFILEPUT

语法规则：

命令类型	语法	返回
设置命令	AT+FTPFILEPUT=<mode>[,<filename>]	OK
测试命令	AT+FTPFILEPUT=?	OK
注意事项	当FTPEXTPUT命令<mode>=1时，本命令不可用	

参数定义：

参数	定义	取值	对取值的说明
<mode>	工作模式	0	结束从文件系统加载到 RAM

		1	开始从文件系统加载到 RAM 并上传
<filename>	文件名	0-128字节	当 mode 为 1 时必须填入文件名，为 0 不需要

15.23 获取远程服务器上文件目录：AT+FTPLIST

语法规则：

命令类型	语法	返回
执行命令	AT+FTPLIST=<mode>[,<reqlen>]	如果<mode>=1，则命令格式为： AT+FTPLIST=1 返回值为： OK或 +CME ERROR: <err>
		如果<mode>=2，则命令格式为： AT+FTPLIST=2,<reqlen> 返回值为： +FTPLIST：2,<cnflen> ….数据… OK
测试命令	AT+FTPLIST=?	OK
URC 上报	输入AT+FTPLIST=1后， 如果是一个成功的FTP会话，后续会有如下的URC上报： +FTPLIST:1,1 如果是FTP数据传输结束，后续会有如下的URC上报： +FTPLIST:1,0 如果FTP会话失败，后续会有如下的URC上报： +FTPLIST:1,<error>	

参数定义：

参数	定义	取值	对取值的说明
<mode>	工作模式	1	打开 FTP 获取文件目录会话
		2	读取 FTP 获取文件目录的数据
<reqlen>	请求的数据的长度	1~1460	
<cnflen>	实际的数据长度	1~1460	
<error>	同+FTPGET 中<error>定义		

15.24 获取远程服务器上文件大小：AT+FTPSIZE

语法规则：

命令类型	语法	返回	说明
执行命令	AT+FTPSIZE	OK +FTPSIZE: 1,0,<size>	获取文件大小成功
		OK +FTPSIZE: 1,<error>,0	获取文件大小失败
		+CME ERROR: <err>	如果是命令错误
测试命令	AT+FTPSIZE=?	OK	
注意事项	文件由命令AT+FTPGETNAME和AT+FTPGETPATH指定		

参数定义：

参数	定义	取值	对取值的说明
<size>	文件大小		单位：字节
<error>	错误码		与 AT+FTPGET 命令的<error>定义相同

15.25 获取 FTP 状态：AT+FTPSTATE

语法规则：

命令类型	语法	返回
执行命令	AT+FTPSTATE	+FTPSTATE: <state> OK
测试命令	AT+FTPSTATE=?	OK

参数定义：

参数	定义	取值	对取值的说明
<state>	工作状态	0	Idle
		1	FTP 会话进行中，包括 FTPGET、FTPPUT、FTPDELE

15.26 保存 FTP 应用上下文：AT+FTPSCONT

语法规则：

命令类型	语法	返回	说明

查询命令	AT+FTPSCONT?	+FTPSCONT: <value> +FTPSERV: <value> +FTPPORT: <value> +FTPUN: <value> +FTPPW: <value> +FTPCID: <value> +FTPMODE: <value> +FTPTYPE: <value> +FTPPUTOPT: <value> +FTPREST: <value> +FTPGETNAME: <value> +FTPGETPATH: <value> +FTPPUTNAME: <value> +FTPPUTPATH: <value> +FTPTIMEOUT: <value> OK	
执行命令	AT+FTPSCONT	OK	将FTP上下文保存，等模块重启后，将自动载入上下文参数并生效

15.27 删除服务器上指定的文件：AT+FTPDELE

语法规则：

命令类型	语法	返回
执行命令	AT+FTPDELE	OK
测试命令	AT+FTPDELE=?	OK
URC 上报	+FTPDELE:1,0	表示删除成功。第一个参数1表示FTP会话打开
	+FTPDELE:1,<error>	表示FTP文件删除失败。错误码<error>定义请参考AT+FTPGET 错误码<error>定义
注意事项	文件由命令AT+FTPGETNAME和AT+FTPGETPATH指定	

15.28 退出当前 FTP 会话：AT+FTPQUIT

语法规则：

命令类型	语法	返回	说明
设置命令	AT+FTPQUIT	OK	成功
		ERROR	失败
测试命令	AT+FTPQUIT=?	OK	

15.29 使用方法举例

由于这部分的命令具有高度相关性，所以将各个命令的应用例程集中在一起描述。

举例：

命令（→）/返回（←）	实例	解释和说明
FTPGET 命令使用方法：		
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	设置FTP功能的承载类型
←	OK	
→	AT+SAPBR=3,1,"APN",""	设置PDP承载之APN参数 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用。（此<apn>可以通过AT+CGDCONT?来查询），所以输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN
←	OK	
→	AT+SAPBR=1,1	激活该承载的GPRS PDP上下文
←	OK	
→	AT+SAPBR=2,1	查询下承载的状态
←	+SAPBR: 1,1,010.169.179.213 OK	第一个参数1表示cid 第二个参数1表示已经连接 第三个参数表示模块获取的IP地址
→	AT+FTPCID=1	设置cid
←	OK	
→	AT+FTPSERV="36.7.2.150"	设置服务器ip地址，也可以使用域名。本文只是举例，请填写自己的服务器地址，不要照抄。
←	OK	
→	AT+FTPUN="user556"	设置服务器用户名。请填写自己FTP服务器的用户名，不要照抄
←	OK	
→	AT+FTPPW="222333"	输入密码。请填写自己FTP服务器的密码，不要照抄
←	OK	
→	AT+FTPGETPATH="/"	设置下载文件的路径
←	OK	
→	AT+FTPLIST=1	
←	OK	

	+FTPLIST: 1,1	
→	AT+FTPLIST=2,1460	
←	+FTPLIST: 2,1460 drwxr-xr-x 1 ftp ftp 0 Feb 24 2018 .1 -rw-r--r-- 1 ftp ftp 107 Jul 10 17:08 1.txt drwxr-xr-x 1 ftp ftp 0 Apr 27 2018 11 -rw-r--r-- 1 ftp ftp 0 Sep 19 2018 111.txt drwxr-xr-x 1 ftp ftp 0 Dec 19 2018 1221 -rw-r--r-- 1 ftp ftp 7 Apr 25 2019 1222.txt -rw-r--r-- 1 ftp ftp 18380 May 07 2018 12220.txt -rw-r--r-- 1 ftp ftp 0 Apr 26 2019 12222.txt -rw-r--r-- 1 ftp ftp 10 Apr 28 2018 122343.txt -rw-r--r-- 1 ftp ftp 254 Sep 04 2018 123456.txt -rw-r--r-- 1 ftp ftp 601283 Jul 21 2018 2018_7_21.sdl -rw-r--r-- 1 ftp ftp 8 Apr 26 2019 33333.txt -rw-r--r-- 1 ftp ftp 14 Oct 13 2017 66.txt -rw-r--r-- 1 ftp ftp 11525352 Mar 13 2019 Air720_CSDK_demo_flash.blf -rw-r--r-- 1 ftp ftp 65536 Jul 31 2018 app2_flash.bin -rw-r--r-- 1 ftp ftp 0 Sep 29 2017 ccc.txt -rw-r--r-- 1 ftp ftp 65536 Jan 18 2018 demo_ota_flash.bin -rw-r--r-- 1 ftp ftp 65536 Jul 30 2018 demo_timer_flash.bin -rw-r--r-- 1 ftp ftp 131072 Oct 10 2017 demo_timer_flash_org.bin -rw-r--r-- 1 ftp ftp 534110 Feb 24 2018 FM320X 绠€浠.pdf -rw-r--r-- 1 ftp ftp 16384 Mar 26 2019 FOTA_APP_720D.bin -rw-r--r-- 1 ftp ftp 5103616 Mar 26 2019 FOTA_CORE_APP_720D.bin drwxr-xr-x 1 ftp ftp 0 Dec 12 2018 get OK	drwxr-xr-x 第一个字母表示文件类型。 d:文件目录 -：普通文件 后面的每三个字符分成一组来看，即 rwx 、r-x 、r-x。 第一组rwx是root组对文件的权限； 第二个r-x是一般用户（用户组）对文件的权限； 第三个r-x是其他用户对文件的权限 r 是可读，w 是可写，x 是可执行， rwx 意思是可读，可写，可执行； r-x 是可读，可执行，不可写； 同理： r-- 是可读，不可写，不可执行 rw- 是可读，可写，不可执行
→	AT+FTPQUIT	退出FTPLIST会话，才能开始后续的会话。 退出FTPLIST会话的方法有两种，一是等待+FTPLIST:1,0上报，二是主动退出FTPLIST会话
←	OK	这里采用的是主动退出FTPLIST会话

	+FTPLIST: 1,86	
→	AT+FTPGETNAME="1.txt"	设置下载文件的名称
←	OK	
→	AT+FTPGET=1	开始FTP下载会话
←	OK	
←	+FTPGET: 1,1	这个是URC上报，表示有数据了
→	AT+FTPGET=2,20	读入20个数据
←	+FTPGET: 2, 10 ??/ ??/ OK	确定读到的是10个数据
←（URC）	+FTPGET: 1,0	等一段时间会提示FTP下载会话结束
→	AT+FTPREST=6	如果FTP连接或会话意外关闭，还可以**断点续传**。从断掉的点开始下载。
←	OK	
→	AT+FTPGET=1	
←	OK +FTPGET: 1,1	+FTPGET: 1,1表示有数据可读
→	AT+FTPQUIT	也可以在提示+FTPGET: 1,0前主动用AT+FTPQUIT退出当前FTP会话
←	OK +FTPGET: 1,86	
FTPPUT 命令使用方法：		
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	设置FTP功能的承载类型
←	OK	
→	AT+SAPBR=3,1,"APN",""	设置PDP承载之APN参数 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用（此<apn>可以通过AT+CGDCONT?来查询），所以输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN
←	OK	
→	AT+SAPBR=1,1	激活该承载的GPRS PDP上下文
←	OK	
→	AT+SAPBR=2,1	查询下承载的状态
←	+SAPBR: 1,1,010.169.179.213	第一个参数1表示cid

	OK	第二个参数1表示已经连接 第三个参数表示模块获取的IP地址
→	AT+FTPCID=1	
←	OK	
→	AT+FTPSERV="36.7.33.111"	设置服务器ip地址，也可以使用域名。本文只是举例，请填写自己的服务器地址，不要照抄。
←	OK	
→	AT+FTPUN="test112"	设置服务器用户名。请填写自己FTP服务器的用户名，不要照抄
←	OK	
→	AT+FTPPW="777888"	输入密码。请填写自己FTP服务器的密码，不要照抄
←	OK	
→	AT+FTPPUTNAME="1222.txt"	设置上传文件的名称
←	OK	
→	AT+FTPPUTPATH="/11/"	设置上传文件的路径
←	OK	
→	AT+FTPPUT=1	
←	OK	
←	+FTPPUT:1,1,1360	这是URC上报，表示可以上传数据了，单次上传的最大长度为1360
→	AT+FTPPUT=2,7	
←	+FTPPUT: 2,7 //此时输入 7 个字符 OK	输入7个字符后，会返回OK，表示输入结束，已经上传。
(URC)	+FTPPUT: 1,0	等待一段时间不动作，会提示FTP PUT会话结束
→	AT+FTPQUIT	也可以在提示+FTPPUT: 1,0之前用AT+FTPQUIT主动结束会话
←	OK +FTPPUT: 1,86	
FTPEXTGET 命令使用方法：		
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	设置FTP功能的承载类型
←	OK	
→	AT+SAPBR=3,1,"APN",""	设置PDP承载之APN参数 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用。此<apn>可以通过AT+CGDCONT?来查询。

		这里输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN
←	OK	
→	AT+SAPBR=1,1	激活该承载的GPRS PDP上下文
←	OK	
→	AT+SAPBR=2,1	查询下承载的状态
←	+SAPBR: 1,1,010.169.179.213 OK	第一个参数1表示cid 第二个参数1表示已经连接 第三个参数表示模块获取的IP地址
→	AT+FTPCID=1	
←	OK	
→	AT+FTPSERV="36.7.33.111"	设置服务器 IP 地址，也可以使用域名
←	OK	
→	AT+FTPUN="test112"	设置用户名
←	OK	
→	AT+FTPPW="777888"	输入密码
←	OK	
→	AT+FTPGETNAME="TEST2.txt"	设置下载文件的名字（请注意：FTPEXTGET命令只支持小于300K的文件的下载）
←	OK	
→	AT+FTPGETPATH="/11/"	设置下载文件所在的文件夹
←	OK	
→	AT+FTPEXTGET=1	开始FTP下载
←	OK	
→	AT+FTPEXTGET?	查询下载了多少字节
←	+FTPEXTGET: 1,246840 OK	
←（URC）	+FTPEXTGET: 1,0	有这个URC上报，表示文件下载完毕，可以保存或读取了
→	AT+FTPEXTGET=2,hello	保存的文件名为 hello
←	+FTPEXTGET: 2, 296895 OK	
→	AT+FTPEXTGET=3,0,296895	将数据显示在串口
←	+FTPEXTGET: 3, 296895 ……这里是显示在串口的数据…… OK	
→	AT+FTPEXTGET=0	设置 FTPEXTGET 模式关闭

←	OK	
→	AT+FTPEXTGET=4	仅 Air780E 系列模块>=V1106 版本支持
←	OK	
←（URC）	+FTPEXTGET: 4,1460	有数据上报
→	AT+FTPEXTGET=4,1460	读数据
→	+FTPEXTGET: 4,1460 ……这里是显示在串口的数据…… OK	
FTPEXTPUT使用方法：		
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	设置FTP功能的承载类型
←	OK	
→	AT+SAPBR=3,1,"APN",""	设置PDP承载之APN参数 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用。此<apn>可以通过AT+CGDCONT?来查询。 这里输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN
←	OK	
→	AT+SAPBR=1,1	激活该承载的GPRS PDP上下文
←	OK	
→	AT+SAPBR=2,1	查询下承载的状态
←	+SAPBR: 1,1,010.169.179.213 OK	第一个参数1表示cid 第二个参数1表示已经连接 第三个参数表示模块获取的IP地址
→	AT+FTPCID=1	
←	OK	
→	AT+FTPSERV="36.7.33.111"	设置服务器 IP 地址，也可以使用域名
←	OK	
→	AT+FTPUN="test112"	设置用户名
←	OK	
→	AT+FTPPW="777888"	输入密码
←	OK	
→	AT+FTPPUTPATH="/11/"	
←	OK	
→	AT+FTPPUTNAME="TEST8861.txt"	
←	OK	

→	AT+FTPEXTPUT=1	
←	OK	
→	AT+FTPEXTPUT=2,0,5,10000	将文件数据从串口输入到模块 RAM。 第 2 个参数是 offset，第 3 个参数是数据长度， 第 4 个参数是超时时间 10s
←	+FTPEXTPUT: 0,5 //这里输入 5 个字符：12345 OK	
→	AT+FTPEXTPUT=2,5,7,10000	这次 offset 是 5，长度为 7
←	+FTPEXTPUT: 5,7 //这里输入 7 个字符：67890AB OK	
→	AT+FTPPUT=1	将 RAM 中的文件传到 FTP 服务器
←	OK	
←（URC）	+FTPPUT: 1,0	等一段时间会提示FTP上传会话结束。此时查看服务器/11/TEST8861.txt，看到的的内容是：1234567890AB
→	AT+FTPEXTPUT=0	改为普通的FTP PUT模式
←	OK	
FTPGETTOFS命令使用方法：		
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	设置FTP功能的承载类型
←	OK	
→	AT+SAPBR=3,1,"APN",""	设置PDP承载之APN参数 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用。此<apn>可以通过AT+CGDCONT?来查询。 这里输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN
←	OK	
→	AT+SAPBR=1,1	激活该承载的GPRS PDP上下文
←	OK	
→	AT+SAPBR=2,1	查询下承载的状态
←	+SAPBR: 1,1,010.169.179.213 OK	第一个参数1表示cid 第二个参数1表示已经连接 第三个参数表示模块获取的IP地址
→	AT+FTPCID=1	
←	OK	

→	AT+FTPSERV="36.7.33.111"	设置服务器 IP 地址，也可以使用域名
←	OK	
→	AT+FTPUN="test112"	设置用户名
←	OK	
→	AT+FTPPW="777888"	输入密码
←	OK	
→	AT+FTPGETNAME="TEST.mp3"	
←	OK	
→	AT+FTPGETPATH="/11/"	
→	OK	
→	AT+FTPGETTOFS=0,"TEST-1.mp3"	开始下载，并存于本地ROM中。都只能指定文件名，不能指定路径，因为存放的路径是固定的 这种下载方式支持自动断点续传
←	OK	
→	AT+FTPGETTOFS?	查询下载情况
←	+FTPGETTOFS: 1,245900,207715 OK	已下载245900字节,207715字节已被存入文件系统
←(URC)	+FTPGETTOFS: 0,245900	下载成功，总共下载了多少字节。可用文件系统命令对文件进行操作
→	AT+FSLS=C:/USER/FTP/	查询保存的文件
←	hello TEST-1.mp3 OK	
→	AT+CAUDPLAY=1,"C:/USER/FTP/TEST-1.mp3"	播放保存的文件
←	OK	
←(URC)	+CAUDPLAY: 1,600	播放结束
FTPPUTFRMFS命令使用方法：		
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	设置FTP功能的承载类型
←	OK	
→	AT+SAPBR=3,1,"APN",""	设置PDP承载之APN参数 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用。此<apn>可以通过AT+CGDCONT?来查询。 这里输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN

←	OK	
→	AT+SAPBR=1,1	激活该承载的GPRS PDP上下文
←	OK	
→	AT+SAPBR=2,1	查询下承载的状态
←	+SAPBR: 1,1,010.169.179.213 OK	第一个参数1表示cid 第二个参数1表示已经连接 第三个参数表示模块获取的IP地址
→	AT+FTPCID=1	
←	OK	
→	AT+FTPSERV="36.7.33.111"	设置服务器 IP 地址，也可以使用域名
←	OK	
→	AT+FTPUN="test112"	设置用户名
←	OK	
→	AT+FTPPW="777888"	输入密码
←	OK	
→	AT+FTPPUTPATH="/"	
←	OK	
→	AT+FTPPUTNAME="TEST001.txt"	传到服务器的文件以TEST001.txt这个名字存放
←	OK	
→	AT+FTPPUTFRMFS="C:/USER/FTP/TEST-1.txt"	将C:\USER\FTP\TEST-1.txt的文件传到服务器
←	OK +FTPPUTFRMFS: 0,41580	上传结束
FTPFILEPUT命令使用方法：		
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	设置FTP功能的承载类型
←	OK	
→	AT+SAPBR=3,1,"APN",""	设置PDP承载之APN参数 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用。此<apn>可以通过AT+CGDCONT?来查询。 这里输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN
←	OK	
→	AT+SAPBR=1,1	激活该承载的GPRS PDP上下文

←	OK	
→	AT+SAPBR=2,1	查询下承载的状态
←	+SAPBR: 1,1,010.169.179.213 OK	第一个参数1表示cid 第二个参数1表示已经连接 第三个参数表示模块获取的IP地址
→	AT+FTPCID=1	
←	OK	
→	AT+FTPSERV="36.7.33.111"	设置服务器 IP 地址，也可以使用域名
←	OK	
→	AT+FTPUN="test112"	设置用户名
←	OK	
→	AT+FTPPW="777888"	输入密码
←	OK	
→	AT+FTPPUTPATH="/"	
←	OK	
→	AT+FTPPUTNAME="TEST_80.txt"	传到服务器后保存的文件名
←	OK	
→	AT+FTPFILEPUT=1,"C:/USER/FTP/TEST-1.txt"	将C:\USER\FTP\TEST-1.txt加载到内存
←	OK	
→	AT+FTPPUT=1	开始上传
←	OK	
←	+FTPPUT: 1,0	上传结束
→	AT+FTPFILEPUT=0	结束FTPFILEPUT模式
←	OK	
FTP远程创建/删除目录或文件：		
→	AT+SAPBR=3,1,"CONTYPE","GPRS"	设置FTP功能的承载类型
←	OK	
→	AT+SAPBR=3,1,"APN",""	设置PDP承载之APN参数 模块注册网络后会从网络自动获取<apn>并激活一个PDP上下文，用于RNDIS上网使用。此<apn>可以通过AT+CGDCONT?来查询。 这里输入AT+SAPBR=3,<cid>,"APN","" 即可，模块内部会按照自动获取的<apn>来设置APN
←	OK	
→	AT+SAPBR=1,1	激活该承载的GPRS PDP上下文
←	OK	

→	AT+SAPBR=2,1	查询下承载的状态
←	+SAPBR: 1,1,010.169.179.213 OK	第一个参数1表示cid 第二个参数1表示已经连接 第三个参数表示模块获取的IP地址
→	AT+FTPCID=1	设置cid
←	OK	
→	AT+FTPSERV="36.7.87.100"	设置服务器ip地址，也可以使用域名
←	OK	
→	AT+FTPUN="user"	设置用户名
←	OK	
→	AT+FTPPW="123456"	输入密码
←	OK	
→	AT+FTPGETPATH="/test"	设置要创建的文件目录
←	OK	
→	AT+FTPMKD	创建目录
←	OK	
←	+FTPMKD: 1,0	
→	AT+FTPRMD	删除目录
←	OK	
←	+FTPRMD: 1,0	
→	AT+FTPGETPATH="/"	
←	OK	
→	AT+FTPGETNAME="HI.txt"	
←	OK	
→	AT+FTPDELE	删除HI.txt这个文件
←	OK +FTPDELE: 1,0	

16 MQTT 相关命令

16.1 设置 MQTT 相关参数：AT+MCONFIG

语法规则：

命令类型	语法	返回	说明
设置命令	AT+MCONFIG=<clientid>[,<username>,<password>[,<will_qos>,<will_retain>,<will_topic>,<will_message>]]	OK	正常返回
		ERROR	输入格式有误

测试命令	AT+MCONFIG=?	+MCONFIG: <clientid>[,<username>,<password>[,(0-2),(0,1), <will_topic>,<will_message>]] OK

参数定义：

参数	定义	取值	对取值的说明
<clientid>	客户端身份	字符串	最大 256 个字节。可以用""括住，也可以不用"" (不能相同，否则最后一个刚连上服务器就会被踢掉)
<username>	登录服务器的用户名	字符串	最大 256 个字节。可以用""括住，也可以不用""
<password>	登录服务器的密码	字符串	最大 256 个字节。可以用""括住，也可以不用""
<will_qos>	服务质量	0	至多一次
		1	至少一次
		2	确保只有一次
<will_retain>	保留标志	0	如果 will_topic 字段没有设置，该字段也需要设置为 0 如果 will_topic 字段有设置，服务器必须将遗嘱消息当做非保留信息发布
		1	如果 will_topic 字段有设置，服务器必须将遗嘱信息当做保留信息发布
<will_topic>	遗嘱消息的主题	字符串	最大 256 个字节。需用""括住 (遗嘱主题不能相同，否则收不到遗嘱消息)
<will_message>	遗嘱消息内容	字符串	最大 1360 个字节。需用""括住

16.2 建立 TCP 连接：AT+MIPSTART

语法规则：

命令类型	语法	返回	说明
设置命令	普通链接： AT+MIPSTART=<svraddr>,<port> SSL链接： AT+SSLMIPSTART=<svraddr>,<port>	OK	正常返回
		ERROR	输入格式有误
		输入这条设置命令以后，后续会有URC上报。 1） 单链接（AT+CIPMUX=0） 如果链接成功地建立，则上报： CONNECT OK 如果链接已经存在，则上报： ALREADY CONNECT	

		如果链接失败，则上报： STATE:<state> CONNECT FAIL 2） 多链接（AT+CIPMUX=1） 如果链接成功地建立，则上报： 7,CONNECT OK 如果链接已经存在，则上报： ALREADY CONNECT 如果链接失败，则上报： 7,CONNECT FAIL
测试命令	AT+MIPSTART=?	+MIPSTART:"(0,255).(0,255).(0,255).(0,255)",(1-65535) +MIPSTART:"DOMAIN NAME",(1-65535) OK
	AT+SSLMIPSTART=?	+SSLMIPSTART:"(0,255).(0,255).(0,255).(0,255)",(1-65535) +SSLMIPSTART:"DOMAIN NAME",(1-65535) OK
注意事项	**当使用SSL链接进行数据传输时，链接命令为: AT+SSLMIPSTART=<svraddr>,<port>** **其余跟普通链接一样。这点请知悉！**	

参数定义：

参数	定义	取值	对取值的说明
<svraddr>	服务器 IP 地址或 DNS 地址	domain name或 XXX.XXX.XXX.XXX	XXX 取值范围：0~255 可以用""括住，也可以不用""
<port>	server port	1-65535	可以用""括住，也可以不用""

16.3 客户端向服务器请求会话连接：AT+MCONNECT

语法规则：

命令类型	语法	返回	说明
设置命令	AT+MCONNECT=<clean_session>,<keepalive>	OK	设置成功
测试命令	AT+MCONNECT=?	+MCONNECT:(0-1),(1-65535)	测试命令的返回的是

		OK	<clean_session>和 <keepalive>的取值范围
URC	设置命令设置成功，返回OK后，后续会根据连接情况自动上报URC： 如果连接成功则返回：CONNACK OK 如果连接失败则返回：ERROR		

参数定义：

参数	定义	取值	对取值的说明
<clean_session>		0	服务端必须基于当前会话（使用客户端标识符识别） 的状态恢复与客户端的通信。 如果没有与这个客户端标识符关联的会话， 服务端必须创建一个新的会话。在连接断开之后，当连接断开后，客户端和服务端必须保存会话信息 [MQTT-3.1.2-4]。当清理会话标志为 0的会话连接断开之后，服务端必须将之后的 QoS 1 和 QoS 2 级别的消息保存为会话状态的一部分， 如果这些消息匹配断开连接时客户端的任何订阅 [MQTT-3.1.2-5]。服务端也可以保存满足相同条件的 QoS 0 级别的消息。
		1	client 和 server 都会抛弃以前的会话，建立一个新的会话。会话持续时间与网络连接持续时间一样长。与此会话相关的会话状态数据在后序的会话中不被采用。
<keepalive>	保活时间	1-65535	时间单位：秒 设备端在保活时间内至少需要发送一次报文，包括 PING 请求。 如果服务器端在保活时间内未接收到任何报文，会断开连接，设备端需要发起重连。 建议取值在 300s 以上。

16.4 发布消息：AT+MPUB

本命令从客户端到服务器，传送应用消息。

语法规则：

命令类型	语法	返回	说明
设置命令	AT+MPUB=<topic>,<qos>,<retain>,<message>	OK	qos=0
		OK PUBACK	qos=1
		OK PUBREC PUBCOMP	qos=2

		ERROR	失败
测试命令	AT+MPUB=?	+MPUB:<topic>,(0-2),(0-1),<message> OK	

参数定义：

参数	定义	取值	对取值的说明
<qos>	服务质量	0	至多一次
		1	至少一次
		2	确保只有一次
<retain>	保留标志	0	服务器不能存储这个消息也不能移除或替换任何现存的保留消息。
		1	服务器必须存储这个应用消息和它的服务等级质量(QOS)，以便它可以被分发给未来的主题名匹配的订阅者。
<topic>	消息主题	字符串	最大 256 个字节。可以用""括住，也可以不用""
<message>	消息内容	字符串	最大 4100 个字节。字符串类型，须用双引号括住。 **注:消息中内嵌的双引号请用\22 表达；控制字符回车\r(0x0D)请用\0D 表达；控制字符换行\n（0x0A）请用\0A 表达；控制字符反斜杠\（0x5C）请用\5C 表达** **如果是 MCU 发消息，可能需要用\\22，\\0D，\\0A，\\5C 来表达，即\需要转义成**

16.5 发布定长消息：AT+MPUBEX

本命令从客户端到服务器，以定长方式传送应用消息。

注：只支持最大 4100 字节收发。

语法规则：

命令类型	语法	返回	说明
设置命令	AT+MPUBEX=<topic>,<qos>,<retain>,[len]	首先返回一个>\ 1）如果指定了len，当长度大于等于len就退出数据模式发送数据返回OK。 2）如果没指定len，则以1A做为结束符，或者5s超时（两次数据间隔超过5s）发送数据返回OK	qos=0
		首先返回一个>	qos=1

		1）如果指定了len，当长度大于等于len就退出数据模式发送数据返回OK。 2）如果没指定len，则以1A做为结束符，或者5s超时（两次数据间隔超过5s）发送数据返回OK PUBACK	
		首先返回一个> 1）如果指定了len，当长度大于等于len就退出数据模式发送数据返回OK。 2）如果没指定len，则以1A做为结束符，或者5s超时（两次数据间隔超过5s）发送数据返回OK PUBREC PUBCOMP	qos=2
		ERROR	失败。 一般由语法错误或条件不具备引起
测试命令	AT+MPUBEX=?	+MPUBEX:<topic>,(0-2),(0-1),<len> OK	

参数定义：

参数	定义	取值	对取值的说明
<qos>	服务质量	0	至多一次
		1	至少一次
		2	确保只有一次
<retain>	保留标志	0	服务器不会存储这个消息也不能移除或替换任何现存的保留消息。
		1	服务器必须存储这个应用消息和它的服务等级质量(QOS)，以便它可以被分发给未来的主题名匹配的订阅者。
<topic>	消息主题	字符串	最大 256 个字节。可以用""括住，也可以不用""
<len>	数据长度	1~4100	

16.6 订阅主题：AT+MSUB

本命令从客户端到服务器，用于一个或多个订阅主题。

语法规则：

命令类型	语法	返回	说明
设置命令	AT+MSUB=<topic>,<qos>	OK SUBACK	成功
		ERROR	失败
测试命令	AT+MSUB=?	+MSUB:<topic>,(0-2) OK	
URC	当设置命令成功以后，如果收到MQTT消息会有相应的URC上报： 如果已经设置AT+MQTTMSGSET=0，收到MQTT消息会上报： +MSUB：<topic>,<len>,<message> 如果已经设置AT+MQTTMSGSET=1，收到MQTT消息会上报： +MSUB：<store_addr>		

参数定义：

参数	定义	取值	对取值的说明
<topic>	消息主题	字符串	最大 256 个字节。可以用""括住，也可以不用""
<qos>	服务质量	0	至多一次
		1	至少一次
		2	确保只有一次
<len>	接收数据长度		单位：字节
<message>	数据内容	字符串	最大 4100 个字节
<store_addr>		0-3	接收消息时缓存的位置

16.7 取消订阅主题：AT+MUNSUB

本命令从客户端到服务器，用于取消订阅主题。

语法规则：

命令类型	语法	返回	说明
设置命令	AT+MUNSUB=<topic>	OK UNSUBACK	成功

		ERROR	失败
测试命令	AT+MUNSUB=?	+MUNSUB:<topic> OK	成功

参数定义：

参数	定义	取值	对取值的说明
<topic>	消息主题	字符串	最大 256 个字节。可以用""括住，也可以不用""

16.8 打印收到的所有的订阅消息：AT+MQTTMSGGET

语法规则：

命令类型	语法	返回	说明
执行命令	AT+MQTTMSGGET	[+MSUB: <topic>,<len>,<message>] [+MSUB: <topic>,<len>,<message>] [+MSUB: <topic>,<len>,<message>] [+MSUB: <topic>,<len>,<message>] OK	执行命令将会打印收到的保存在cache中的主题订阅消息。执行完以后，<status>将会变成invalid。
查询命令	AT+MQTTMSGGET?	+MQTTMSGGET：0,<status> +MQTTMSGGET：1,<status> +MQTTMSGGET：2,<status> +MQTTMSGGET：3,<status> OK	
测试命令	AT+MQTTMSGGET=?	OK	
注意事项	当 AT+MQTTMSGSET=1，执行命令可以打印订阅消息。一次最多打印**4**条。如果一次上报多于**4**条，则打印最新的**4**条，最老的那条将被覆盖。		

参数定义：

参数	定义	取值	对取值的说明
<topic>	消息主题	字符串	最大 256 个字节。
<len>	接收消息长度		单位：字节
<message>	消息内容	字符串	最大 4100 个字节。
<status>	消息状态	VALID	有效数据，AT+MQTTMSGGET执行模块可以打印这些消息
		INVALID	无效数据

16.9 设置订阅消息的打印模式：AT+MQTTMSGSET

语法规则：

命令类型	语法	返回	说明
设置命令	AT+MQTTMSGSET=<mode>	OK	成功
		ERROR	失败
查询命令	AT+MQTTMSGSET?	+MQTTMSGSET:<mode> OK	
测试命令	AT+MQTTMSGSET=?	+MQTTMSGSET:(0,1) OK	

参数定义：

参数	定义	取值	对取值的说明
<mode>	消息上报模式	0	主动上报到串口。有新订阅消息时，上报的 URC 为： +MSUB：<topic>,<len>,<message>
		1	缓存模式。有新订阅消息时，上报的 URC 为： +MSUB：<store_addr> 然后用 AT+MQTTMSGGET 来读消息

16.10 MQTT 消息编码格式切换：AT+MQTTMODE

本命令设置 MQTT 发布消息的编码格式，是 ASCII 还是 HEX。

语法规则：

命令类型	语法	返回	说明
设置命令	AT+MQTTMODE=<mode>	OK	成功
		ERROR	失败
查询命令	AT+MQTTMODE?	+MQTTMODE: <mode> OK	
测试命令	AT+MQTTMODE=?	+MQTTMODE: (0,1) OK	

参数定义：

参数	定义	取值	对取值的说明
<mode>	MQTT 发布消息的编码格式	0	ASCII格式
		1	HEX 格式

举例：

命令（→）/返回（←）	实例	解释和说明
→	AT+MQTTMODE=1	设置发布MQTT消息的格式为：HEX
←	OK	
→	AT+MPUB="test",0,0,"313233"	比如发布消息内容为123，则发布的格式为十六进制格式的可见模式，实际消息接收者收到的消息内容是123
←	OK	
→	AT+MQTTMODE=0	设置发布MQTT消息的格式为：ASCII
←	OK	
→	AT+MPUB="test",0,0,"123"	发布消息的内容 123
←	OK	

16.11 关闭 TCP 连接：AT+MIPCLOSE

语法规则：

命令类型	语法	返回	说明
设置命令	AT+MIPCLOSE	OK	成功
		ERROR	失败
测试命令	AT+MIPCLOSE=?	OK	返回OK，表示支持此命令

16.12 关闭 MQTT 连接：AT+MDISCONNECT

语法规则：

命令类型	语法	返回	说明
设置命令	AT+MDISCONNECT	OK	成功
		ERROR	失败
测试命令	AT+MDISCONNECT=?	OK	返回OK，表示支持此命令

16.13 查询 MQTT 连接状态：AT+MQTTSTATU

语法规则：

命令类型	语法	返回	说明
执行命令	AT+MQTTSTATU	+MQTTSTATU :<state> OK	成功
		ERROR	失败

测试命令	AT+MQTTSTATU=?	OK	返回OK，表示支持此命令

参数定义：

参数	定义	取值	对取值的说明
<state>	MQTT 连接状态	0	离线
		1	已经登陆认证过，可以PUB数据了
		2	还没认证，需要发送 MCONNECT 命令

16.14 使用方法举例

由于这部分的命令具有高度相关性，所以将各个命令的应用例程集中在一起描述。

举例：

命令（→）/返回（←）	实例	解释和说明
普通流程以及SSL不带证书验证流程：		
→	AT+CGREG?	查询当前GPRS注册状态
←	+CGREG: 0,1 OK	<n>=0，表示禁用URC上报 <stat>=1，标识已经注册GPRS网络，而且是本地网
→	AT+CGATT?	查看当前GPRS附着状态
←	+CGATT: 1 OK	<state>=1，标明当前GPRS已经附着
→	AT+MCONFIG=<clientid>,XXXX,$$$$	注意： XXXX 是用户名 $$$$ 是密码 请开发者写真实的<clientid>、用户名和密码，不要照抄，这三个参数加不加双引号都可以。如果用户名和密码为空，则可以写成： AT+MCONFIG=<clientid>,"",""
←	OK	
→	AT+MIPSTART="ip 或域名","port"	这里，请填上用户自己的mqtt服务器的IP地址或域名地址，以及端口号 **注：** **1）模块开机注册后缺省就有一个激活的PDP承载，查询AT+CGDCONT?可以看到一个IP。此时可以直接用MQTT的AT命令。** **2）当使用SSL链接进行数据传输时（无需证书验证），链接命令格式为:** **AT+SSLMIPSTART=<svraddr>,<port>**

		其余跟普通链接一样。这点请知悉！
←	OK	
←(URC)	CONNECT OK	
→	AT+MCONNECT=1,60	建立mqtt会话
←	OK CONNACK OK	**注：在MIPSTART返回CONNECT OK后才能发MCONNECT命令，而且要立即发，否则会被服务器踢掉。** 收到CONNACK OK后才能发布消息
→	AT+MSUB="mqtt/topic",0	订阅
←	OK SUBACK	
→	AT+MPUB="mqtt/topic",0,0,"SSSSddddd"	发布，消息格式缺省为ASCII格式
←	OK	
→	AT+MQTTMODE=1	设置消息格式为HEX格式
←	OK	
→	AT+MPUB="mqtt/topic",0,0,"313233"	发送消息"123"到主题"mqtt/topic"
←	OK	
←(URC)	+MSUB: 0	上报方式是cache方式，需要用+MQTTMSGGET来读
→	AT+MQTTMSGGET?	
←	+MQTTMSGGET: 0,VALID +MQTTMSGGET: 1,INVALID +MQTTMSGGET: 2,INVALID +MQTTMSGGET: 3,INVALID OK	
→	AT+MQTTMSGGET	
←	+MSUB: mqtt/topic,9 byte,SSSSddddd OK	
→	AT+MQTTMSGSET=0	设置为直接上报消息
←	OK	
→	AT+MPUB="mqtt/topic",0,0,"SSSSddddd"	
←	OK	
←(URC)	+MSUB: "mqtt/topic",9 byte,SSSSddddd	
→	AT+MDISCONNECT	模块先关闭MQTT连接
←	OK	
→	AT+MIPCLOSE	关闭TCP链接
←	OK	

SSL带证书验证流程（单向认证）：		
→	AT+CGATT?	查看当前GPRS附着状态
←	+CGATT: 1 OK	<state>=1，标明当前GPRS已经附着
→	AT+FSCREATE="ca.crt"	创建CA 证书文件
←	OK	
→	AT+FSWRITE="ca.crt",0,1282,15	1282是证书文件长度，15为超时时间，这里是举例说明，请填入实际数据，不要照抄
←	OK	
→	AT+SSLCFG="cacert",88,"ca.crt"	设置CA证书文件名称，ca.crt为CA证书文件名称
←	OK	
→	AT+SSLCFG="seclevel",88,1	设置认证模式为只对服务器认证
←	OK	
→	AT+MCONFIG=<clientid>,XXXX,$$$$	注意： XXXX 是用户名 $$$$ 是密码 请开发者写真实的<clientid>、用户名和密码，不要照抄，这三个参数加不加双引号都可以。如果用户名和密码为空，则可以写成： AT+MCONFIG=<clientid>,"",""
←	OK	
→	AT+SSLMIPSTART=<svraddr>,<port>	<svraddr>,<port>请修改为真实的MQTT服务器和端口
←	OK	
→	AT+MCONNECT=1,60	建立mqtt会话
←	OK CONNACK OK	**注：在MIPSTART返回CONNECT OK后才能发MCONNECT命令，而且要立即发，否则会被服务器踢掉。** 收到CONNACK OK后才能发布消息
		后面的流程跟上面普通流程相同
SSL带证书验证流程（双向认证）：		
→	AT+FSCREATE="ca.crt"	创建CA 证书文件
←	OK	
→	AT+FSWRITE="ca.crt",0,1282,15	1282是证书文件长度，15为超时时间，这里是举例说明，请填入实际数据，不要照抄
	>这里输入 CA 证书文件内容，1282 个字节	
←	OK	
→	AT+FSCREATE="client.crt"	创建客户端证书文件

←	OK	
→	AT+FSWRITE="client.crt",0,1026,15	
	>这里输入客户端证书文件，1026 个字节	
←	OK	
→	AT+FSCREATE="client.key"	创建客户端key文件
←	OK	
→	AT+FSWRITE="client.key",0,1706,15	
	>这里输入客户端 key 文件内容，1706 个字节	
←	OK	
→	AT+SSLCFG="cacert",88,"ca.crt"	设置CA证书文件为ca.crt
←	OK	
→	AT+SSLCFG="clientcert",88,"client.crt"	设置客户端证书文件为client.crt
←	OK	
→	AT+SSLCFG="clientkey",88,"client.key"	设置客户端密钥文件为client.key
←	OK	
→	AT+SSLCFG="seclevel",88,2	设置认证模式为双向认证
←	OK	
→	AT+MCONFIG=<clientid>,XXXX,$$$$	注意： XXXX 是用户名 $$$$ 是密码 请开发者写真实的<clientid>、用户名和密码，不要照抄，这三个参数加不加双引号都可以。如果用户名和密码为空，则可以写成： AT+MCONFIG=<clientid>,"",""
←	OK	
→	AT+SSLMIPSTART=<svraddr>,<port>	<svraddr>,<port>请修改为真实的MQTT服务器和端口
←	OK	
→	AT+MCONNECT=1,360	建立mqtt会话
←	OK CONNACK OK	**注：在MIPSTART返回CONNECT OK后才能发MCONNECT命令，而且要立即发，否则会被服务器踢掉。** 收到CONNACK OK后才能发布消息
		后面的流程跟上面普通流程相同

如果模块出现 TCP 断链的主动上报，该如何处理：

←(URC)	CLOSED	TCP发生断链

→	AT+MQTTSTATU	查询MQTT连接状态
←	+MQTTSTATU :0 OK	0为离线
		后面从MIPSTART开始重新连接

如果模块出现上下文去激活的主动上报，该如何处理：

←(URC)	+PDP DEACT	PDP去激活，此时需要重新激活一个PDP上下文才能继续使用MQTT的AT命令
→	AT+CIPSHUT	关闭移动场景
←	OK	
	此时有三种处理方式，见右侧	3） AT+CGDCONT=5,"IP","<apn>"//<apn>请填写实际APN AT+CGACT=1,5 再从MIPSTART开始 4） AT+CFUN=0，再AT+CFUN=1 5） AT+RESET重启模块

18 GPS 相关的命令

注：本命令仅适用于合宙 4G CAT1 模块（Air780EG 系列）。

18.1GPS 开关：AT+CGNSPWR

语法规则：

命令类型	语法	返回	说明
设置命令	AT+CGNSPWR=<status>	OK	开关正常
		ERROR	开关失败
查询命令	AT+CGNSPWR?	+CGNSPWR: <status> OK	

参数定义：

参数	定义	取值	对取值的说明
<status>	开关状态	0	关闭
		1	打开

18.2 读取 GNSS 信息：AT+CGNSINF

语法规则：

命令类型	语法	返回
执行命令	AT+CGNSINF	+CGNSINF: <GNSS run status>,<Fix status>, <UTC date & Time>, <Latitude>,<Longitude>, <MSL Altitude>,<Speed Over Ground>, <CourseOver Ground>, <Fix Mode>,<Reserved1>,<HDOP>,<PDOP>, <VDOP>,<Reserved2>,<GNSS Satellites in View>, <GNSS Satellites Used>,<GLONASS Satellites Used>,<Reserved3>,<C/N0 max>, <HPA>,<VPA> OK

参数定义：

参数	定义	取值	对取值的说明
<GNSS run status>	GNSS 运行状态	0	GNSS OFF
		1	GNSS ON
<Fix status>	定位状态	0	not fixed position
		1	fixed position
<UTC date & Time>	UTC时间	yyyyMMddhhmmss	yyyy: [1980,——] MM : [1,12] dd: [1,31]

			hh: [0,23] mm: [0,59] ss:[0,60]
<Latitude>	纬度	±dd.dddddd	[-90.000000,90.000000]
<Longitude>	经度	±ddd.dddddd	[-180.000000,180.000000]
<MSL Altitude>	海拔高度		单位：meters
<Speed Over Ground>	对地速度	0~999.99	单位：Km/hour
<Course Over Ground>	对地航向	0~360.00	单位：degress
<Fix Mode>	定位模式	1	1 为未定位
		2	2 为 2D 定位
		3	3 为 3D 定位
<Reserved1>	保留 1		
<HDOP>	水平精度因子	0~99.9	
<PDOP>	位置精度因子	0~99.9	
<VDOP>	垂直精度因子	0~99.9	
<Reserved2>	保留 2		
<GNSS Satellites in View>	可视 GNSS 卫星	0~99	
<GNSS Satellites Used>	使用 GNSS 卫星	0~99	
<GLONASS Satellites Used>	使用 GLONASS 卫星	0~99	
<Reserved3>	保留 3		
<C/N0 max>	最大信噪比	0~55	单位：dBHz
<HPA>		0~9999.9	单位：meters
<VPA>		0~9999.9	单位：meters

18.3 打开 GNSS URC 上报：AT+CGNSURC

语法规则：

命令类型	语法	返回
设置命令	AT+CGNSURC=<period_time> OK	OK +CME ERROR:<err>
查询命令	AT+CGNSURC?	+CGNSURC: <period_time> OK
测试命令	AT+CGNSURC=?	+CGNSURC: (0-255) OK
URC	与AT+CGNSINF的返回格式相同： +CGNSINF: <GNSS run status>,<Fix status>, <UTC date & Time>,<Latitude>,<Longitude>, <MSL Altitude>,<Speed Over Ground>, <Course Over Ground>, <Fix Mode>,<Reserved1>,<HDOP>,<PDOP>, <VDOP>,<Reserved2>,<GNSS Satellites in View>,	

	<GNSS Satellites Used>,<GLONASS Satellites Used>,<Reserved3>,<C/N0 max>,<HPA>,<VPA>

参数定义：

参数	定义	取值	对取值的说明
<period_time>		0	关闭导航数据URC上报
		1~255	开启导航数据URC周期上报并设置周期时间

18.4 将读取到的 GNSS 数据发送到 AT 口：AT+CGNSTST

语法规则：

命令类型	语法	返回
设置命令	AT+CGNSTST=<mode> OK	OK
		+CME ERROR:<err>
测试命令	AT+CGNSTST?	+CGNSTST:<mode> OK
测试命令	AT+CGNSTST=?	+CGNSTST:(0-1) OK

参数定义：

参数	定义	取值	对取值的说明
<mode>	开关	0	switch off
		1	switch on

18.5 给 GNSS 发送控制命令：AT+CGNSCMD

语法规则：

命令类型	语法	返回
设置命令	AT+CGNSCMD=<cmdType>,<cmdString>,<CmdHeadString>] OK	OK
		+CME ERROR:<err>
测试命令	AT+CGNSCMD=?	+CGNSCMD:(0-1),"cmdString" OK

参数定义：

参数	定义	取值	对取值的说明
<cmdType>	命令类型	0	NMEA style command。目前仅支持这种。
		1	HEX style command。
<cmdString>	命令字符串		字符串类型
<CmdHeadString>	应答特征		填入应答的特征字符，用于捕获命令的结果

18.6 设置辅助定位：AT+CGNSAID

秒定位， 由时间辅助定位<time>，EPO 文件辅助<epo>，位置信息辅助<loc> 3 个参数共同作用完成的。这 3 个辅助定位都使能后， AT+CGNSPWR=1 打开 gps 就会跑秒定位的流程。

语法规则：

命令类型	语法	返回
设置命令	AT+CGNSAID=<mode>,<time>,<epo>,<loc>	OK
		+CME ERROR:<err>
测试命令	AT+CGNSAID=?	+CGNSAID: (0-31)(0-1)(0-1)(0-1) OK

参数定义：

参数	定义	取值	对取值的说明
<mode>	参数保留	0-31	本参数暂不起作用
<time>	是否同步 GNSS UTC 时间	0	不使能
		1	使能（建议开启）
<epo>	是否同步 epo（Extended Pridiction Orbit）文件	0	不使能
		1	使能（建议开启）
<loc>	是否使能位置辅助定位	0	不使能
		1	使能（建议开启）

18.7 删除 EPO 文件：AT+CGNSDEL

语法规则：

命令类型	语法	返回
设置命令	AT+CGNSDEL=<mode>	+CGNSDEL: <mode>,0,0 OK
		+CME ERROR:<err>

参数定义：

参数	定义	取值	对取值的说明
<mode>	模式	1	1和2保留 3 删除EPO文件
		2	
		3	

18.8 定义 NMEA 解析 ： AT+CGNSSEQ

语法规则：

命令类型	语法	返回
设置命令	AT+CGNSSEQ=" s t r "	OK
		+CME ERROR:<err>
查询命令	AT+CGNSSEQ?	+CGNSSEQ: %s OK
测试命令	AT+CGNSSEQ=?	+CGNSSEQ: (GGA,GSA,RMC,GSV) OK

参数定义：

参数	定义	取值	对取值的说明
s t r	字符串		字符串类型

18.9 使用方法举例

由于这部分的命令具有高度相关性，所以将各个命令的应用例程集中在一起描述。

举例：

命令（→）/返回（←）	实例	解释和说明
使用举例		
→	AT+CGNSPWR?	查询GPS是否已经打开
←	+CGNSPWR: 0 OK	0是未打开
→	AT+CGNSPWR=1	打开GPS
←	OK	
→	AT+CGNSAID=31,1,1,1	使能位置辅助定位(指令立马返回

		OK，根据信号强度，大概2-10秒左右会定位成功)
←	OK	
→	AT+CGNSINF	查询GNSS信息
←	+CGNSINF: 1,1,20201110032427,31.820789,117.117390,78.500,0.00,130.07,3,,1.79,0.89,4.00,,12,11,,,34,, OK	
→	AT+CGNSURC=1	设置定位信息自动上报，每隔5个fix，就上报一次
←	OK	
←（URC）	+UGNSINF: 1,1,20201110031835,31.820751,117.117314,63.900,0.00,0.00,3,,1.50,0.89,4.00,,13,13,,,39,, +UGNSINF: 1,1,20201112031836,31.820751,117.117314,63.400,0.00,0.00,3,,1.50,0.89,4.00,,13,13,,,38,, +UGNSINF: 1,1,20201110031837,31.820754,117.117314,63.000,0.00,0.00,3,,1.50,0.89,4.00,,13,13,,,38,,……	每隔5个定位，就上报一次
→	AT+CGNSURC=0	关闭定位信息自动上报
←	OK	

十八 LuatOS API

adc - 数模转换

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 adc 的 demo 例子

示例

```
-- 本库可读取硬件 adc 通道，也支持读取 CPU 温度和 VBAT 供电电源(若模块支持的话)

-- 读取 CPU 温度，单位为 0.001 摄氏度，是内部温度，非环境温度
adc.open(adc.CH_CPU)local temp =
adc.get(adc.CH_CPU)adc.close(adc.CH_CPU)

-- 读取 VBAT 供电电压，单位为 mVadc.open(adc.CH_VBAT)local vbat =
adc.get(adc.CH_VBAT)adc.close(adc.CH_VBAT)

-- 物理 ADC 通道请查阅 adc.get 或者 adc.read 的注释
```

常量

```
adc.open(id)

adc.setRange(range)

adc.read(id)

adc.get(id)

adc.close(id)
```

常量

常量	类型	解释
adc.ADC_RANGE_3_6	number	air105 的 ADC 分压电阻开启，范围 0~3.76V
adc.ADC_RANGE_1_8	number	air105 的 ADC 分压电阻关闭，范围 0~1.88V
adc.ADC_RANGE_3_8	number	air780E 开启 ADC0,1 分压电阻，范围 0~3.8V
adc.ADC_RANGE_1_2	number	air780E 关闭 ADC0,1 分压电阻，范围 0~1.2V
adc.CH_CPU	number	CPU 内部温度的通道 id
adc.CH_VBAT	number	VBAT 供电电压的通道 id
adc.T1	number	ADC1 (如存在多个 adc 可利用此常量使用多 ADC 例如 adc.open(ADC1+2) 打开 ADC1 channel 2)
adc.T2	number	ADC2 (如存在多个 adc 可利用此常量使用多 ADC 例如 adc.open(ADC2+3) 打开 ADC2 channel 3)

adc.open(id)

打开 adc 通道

参数

传入值类型	解释
int	通道 id,与具体设备有关,通常从 0 开始

返回值

返回值类型	解释
boolean	打开结果

例子

```
-- 打开 adc 通道 4,并读取 if adc.open(4) then
    log.info("adc", adc.read(4)) -- 返回值有 2 个，原始值和计算值,通常只需要后者
    log.info("adc", adc.get(4))  -- 返回值有 1 个，仅计算值
endadc.close(4) -- 若需要持续读取，则不需要 close，功耗会高一点.
```

---

adc.setRange(range)

设置 ADC 的测量范围，注意这个和具体芯片有关，目前只支持 air105/Air780E 系列

参数

传入值类型	解释
int	range 参数,与具体设备有关,比如 air105 填 adc.ADC_RANGE_1_8 和 adc.ADC_RANGE_3_6
return	nil

返回值

无

例子

```
-- 关闭 air105 内部分压
adc.setRange(adc.ADC_RANGE_1_8)-- 打开 air105 内部分压
adc.setRange(adc.ADC_RANGE_3_6)


-- EC618 系列(Air780E 等)支持 2 种
adc.setRange(adc.ADC_RANGE_1_2) -- 关闭分压
adc.setRange(adc.ADC_RANGE_3_8) -- 启用分压
```

---

adc.read(id)

读取 adc 通道

参数

传入值类型	解释
int	通道 id,与具体设备有关,通常从 0 开始

返回值

返回值类型	解释
int	原始值,一般没用,可以直接抛弃
int	从原始值换算得出的实际值，通常单位是 mV

例子

```
-- 打开 adc 通道 2,并读取
if adc.open(2) then
```

```
-- 这里使用的是 adc.read 会返回 2 个值，推荐走 adc.get 函数,直接取实际值
log.info("adc", adc.read(2))endadc.close(2)
```

---

adc.get(id)

获取 adc 计算值

参数

传入值类型	解释
int	通道 id,与具体设备有关,通常从 0 开始

返回值

返回值类型	解释
int	单位通常是 mV，部分通道会返回温度值,单位千分之一摄氏度. 若读取失败,会返回-1

例子

```
-- 本 API 在 2022.10.01 后编译的固件可用
-- 打开 adc 通道 2,并读取
if adc.open(2) then
    log.info("adc", adc.get(2))endadc.close(2) -- 按需关闭
```

---

adc.close(id)

关闭 adc 通道

参数

传入值类型	解释
int	通道 id,与具体设备有关,通常从 0 开始

返回值

无

例子

```
-- 打开 adc 通道 2,并读取
if adc.open(2) then
    log.info("adc", adc.read(2))endadc.close(2)
```

antbot - 蚂蚁链

已适配 Air780E/Air700E

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 antbot 的 demo 例子

示例

```
-- Copyright (C) 2015-2023 Ant Group Holding Limited
-- 本模块由蚂蚁链提供实现并开放给用户使用
-- 具体用法请查阅 demo,并联系蚂蚁链获取技术支持
antbot.init()
antbot.app_sta_get()
antbot.version_get()
antbot.asset_register(asset_id, asset_type, asset_dataver)
antbot.asset_data_publish(data)
```

```
antbot.device_status_get()
antbot.asset_status_get(asset_id)
antbot.channel_switch(cmd)
antbot.config_set(config)
antbot.config_get()
```

antbot.init()

初始化

参数

无

返回值

返回值类型	解释
int	0:成功 其他值为失败

例子

```
-- 初始化蚂蚁链的底层适配
antbot.init()
```

---

antbot.app_sta_get()

获取客户端状态

参数

无

返回值

返回值类型	解释
int	状态码

例子

无

---

antbot.version_get()

获取 SDK 版本号

参数

无

返回值

返回值类型	解释
string	版本号,如果获取失败返回 nil

例子

无

---

antbot.asset_register(asset_id, asset_type, asset_dataver)

asset 资源注册

参数

传入值类型		解释
string	asset_id	资源 ID
string	asset_type	资源类型
string	asset_dataver	资源数据版本

返回值

返回值类型　　解释

int　　0:成功 其他值为失败

例子

无

---

antbot.asset_data_publish(data)

asset 资源发布

参数

传入值类型　　解释

string　　data 资源数据

返回值

返回值类型　　解释

int　　0:成功 其他值为失败

例子

无

---

antbot.device_status_get()

获取设备状态

参数

无

返回值

返回值类型　解释

int　　设备状态

例子

无

---

antbot.asset_status_get(asset_id)

获取 assset 状态

参数

传入值类型　　解释

string　　asset_id　资源 ID

返回值

返回值类型　解释

int　　资源状态

例子

无

---

antbot.channel_switch(cmd)

切换 channel

参数

传入值类型　　解释

int　　0 - 关闭, 1 - 开启

返回值

返回值类型	解释
int	0:成功 其他值为失败

例子

无

---

antbot.config_set(config)

配置设备

参数

传入值类型	解释
string	config 配置内容

返回值

返回值类型	解释
int	0:成功 其他值为失败

例子

无

---

antbot.config_get()

获取设备配置

参数

无

返回值

返回值类型　解释

string　　配置内容

例子

无

audio - 多媒体-音频

已适配 Air101/Air103 Air601 Air105 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 audio 的 demo 例子

常量

audio.start(id, audio_format, num_channels, sample_rate, bits_per_sample, is_signed)

audio.write(id, data)

audio.stop(id)

audio.pause(id, pause)

audio.on(id, event, func)

audio.play(id, path, errStop)

audio.tts(id, data)

audio.playStop(id)

audio.isEnd(id)

audio.getError(id)

audio.config(id, paPin, onLevel, dacDelay, paDelay, dacPin, dacLevel, dacTimeDelay)

audio.vol(id, value)

audio.micVol(id, value)

audio.setBus(id, bus_type)

audio.debug(on_off)

audio.pm(id,pm_mode)

常量

常量	类型	解释
audio.RESUME	number	PM 模式 工作模式
audio.STANDBY	number	PM 模式 待机模式，PA 断电，codec 待机状态，系统不能进低功耗状态，如果 PA 不可控，codec 进入静音模式
audio.SHUTDOWN	number	PM 模式 关机模式，PA 断电，可配置的 codec 关机状态，不可配置的 codec 断电，系统能进低功耗状态
audio.POWEROFF	number	PM 模式 断电模式，PA 断电，codec 断电，系统能进低功耗状态
audio.PCM	number	PCM 格式，即原始 ADC 数据
audio.MORE_DATA	number	audio.on 回调函数传入参数的值，表示底层播放完一段数据，可以传入更多数据
audio.DONE	number	audio.on 回调函数传入参数的值，表示底层播放完全部数据了
audio.BUS_DAC	number	硬件输出总线，DAC 类型
audio.BUS_I2S	number	硬件输出总线，I2S 类型
audio.BUS_SOFT_DAC	number	硬件输出总线，软件模式 DAC 类型

audio.start(id, audio_format, num_channels, sample_rate, bits_per_sample, is_signed)

启动一个多媒体通道准备播放音频

参数

传入值类型	解释
int	多媒体播放通道号，0 或者 1
int	音频格式

传入值类型	解释
int	声音通道数
int	采样频率
int	采样位数
boolean	是否有符号，默认 true

返回值

返回值类型	解释
boolean	成功 true，失败 false

例子

```
audio.start(0, audio.PCM, 1, 16000, 16)
```

---

audio.write(id, data)

往一个多媒体通道写入音频数据

参数

传入值类型	解释
string	or zbuff 音频数据

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
audio.write(0, "xxxxxx")
```

---

audio.stop(id)

停止指定的多媒体通道

参数

传入值类型	解释
int	audio id,例如 0

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
audio.stop(0)
```

---

audio.pause(id, pause)

暂停/恢复指定的多媒体通道

参数

传入值类型	解释
int	audio id,例如 0
boolean	onoff true 暂停，false 恢复

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
audio.pause(0, true) --暂停通道 0
```

```
audio.pause(0, false) --恢复通道 0
```

---

audio.on(id, event, func)

注册 audio 播放事件回调

参数

传入值类型	解释
int	audio id, audio 0 写 0, audio 1 写 1
function	回调方法，回调时传入参数为 1、int 通道 ID 2、int 消息值，只有 audio.MORE_DATA 和 audio.DONE

返回值

返回值类型	解释
nil	无返回值

例子

```
audio.on(0, function(audio_id, msg)
    log.info("msg", audio_id, msg)end)
```

---

audio.play(id, path, errStop)

播放或者停止播放一个文件，播放完成后，会回调一个 audio.DONE 消息，可以用 pause 来暂停或者恢复，其他 API 不可用。考虑到读 SD 卡速度比较慢而拖累 luavm 进程的速度，所以尽量使用本 API

参数

传入值类型	解释

传入值类型	解释
int	音频通道
string/table	文件名，如果为空，则表示停止播放，如果是 table，则表示连续播放多个文件，主要应用于云喇叭，目前只有 EC618 支持，并且会用到 errStop 参数
boolean	是否在文件解码失败后停止解码，只有在连续播放多个文件时才有用，默认 true，遇到解码错误自动停止

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
audio.play(0, "xxxxxx")                    --开始播放某个文件 audio.play(0)
                                           --停止播放某个文件
```

---

audio.tts(id, data)

TTS 播放或者停止

参数

传入值类型	解释
int	音频通道
string/zbuff	需要播放的内容

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

无

---

audio.playStop(id)

停止播放文件，和 audio.play(id)是一样的作用

参数

传入值类型	解释
int	audio id,例如 0

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
audio.playStop(0)
```

---

audio.isEnd(id)

检查当前文件是否已经播放结束

参数

传入值类型	解释
int	音频通道

返回值

返回值类型	解释

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
audio.isEnd(0)
```

---

audio.getError(id)

获取最近一次播放结果，不是所有平台都支持的，目前只有 EC618 支持

参数

传入值类型	解释
int	音频通道

返回值

返回值类型	解释
boolean	是否全部播放成功，true 成功，false 有文件播放失败
boolean	如果播放失败，是否是用户停止，true 是，false 不是
int	第几个文件失败了，从 1 开始

例子

```
local result, user_stop, file_no = audio.getError(0)
```

---

audio.config(id, paPin, onLevel, dacDelay, paDelay, dacPin, dacLevel, dacTimeDelay)

配置一个音频通道的特性，比如实现自动控制 PA 开关。注意这个不是必须的，一般在调用 play 的时候才需要自动控制，其他情况比如你手动控制播放时，就可以自己控制 PA 开关

参数

传入值类型	解释
int	音频通道
int	PA 控制 IO
int	PA 打开时的电平
int	在 DAC 启动前插入的冗余时间，单位 100ms，一般用于外部 DAC
int	在 DAC 启动后，延迟多长时间打开 PA，单位 1ms
int	外部 dac 电源控制 IO，如果不填，则表示使用平台默认 IO，比如 Air780E 使用 DACEN 脚，air105 则不启用
int	外部 dac 打开时，电源控制 IO 的电平，默认拉高
int	音频播放完毕时，PA 与 DAC 关闭的时间间隔，单位 1ms，默认 0ms

返回值

无

例子

```
audio.config(0, pin.PC0, 1) --PA 控制脚是 PC0，高电平打开，air105 用这个配置就可以用了

audio.config(0, 25, 1, 6, 200)      --PA 控制脚是 GPIO25，高电平打开，Air780E 云喇叭板用这个配置就可以用了
```

---

audio.vol(id, value)

配置一个音频通道的音量调节，直接将原始数据放大或者缩小，不是所有平台都支持，建议尽量用硬件方法去缩放

参数

传入值类型　　解释

int　　音频通道

int　　音量，百分比，1%~1000%，默认 100%，就是不调节

返回值

返回值类型　　解释

int　　当前音量

例子

```
local result = audio.vol(0, 90)    --通道 0 的音量调节到 90%，result 存放了调节后的音量水平，有可能仍然是 100
```

---

audio.micVol(id, value)

配置一个音频通道的 mic 音量调节

参数

传入值类型　　解释

int　　音频通道

int　　mic 音量，百分比，1%~100%，默认 100%，就是不调节

返回值

返回值类型　　解释

int　　当前 mic 音量

例子

```
local result = audio.vol(0, 90)    --通道 0 的音量调节到 90%，result 存放了调节后的音量水平，有可能仍然是 100
```

---

audio.setBus(id, bus_type)

配置一个音频通道的硬件输出总线，只有对应 soc 软硬件平台支持才设置对应类型

参数

传入值类型	解释
int	音频通道,例如 0
int	总线类型，例如 audio.BUS_SOFT_DAC
int	硬件 id，例如 总线类型为 audio.BUS_I2S 时,硬件 id 即为 i2s codec 的 i2c id

返回值

返回值类型	解释
nil	无返回值

例子

```
audio.setBus(0, audio.BUS_SOFT_DAC)          --通道 0 的硬件输出通道设置为软件 DAC
audio.setBus(0, audio.BUS_I2S)       --通道 0 的硬件输出通道设置为 I2S
```

---

audio.debug(on_off)

配置调试信息输出

参数

传入值类型	解释
boolean	true 开 false 关

返回值

无

例子

无

---

audio.pm(id,pm_mode)

audio 休眠控制(一般会自动调用不需要手动执行)

参数

传入值类型	解释
int	音频通道
int	休眠模式

返回值

返回值类型	解释
boolean	true 成功

例子

```
audio.pm(multimedia_id,audio.RESUME)
```

bit64 - 32 位系统上对 64 位数据的基本算术运算和逻辑运算

已适配 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 bit64 的 demo 例子

bit64.to32(data64bit)

bit64.to64(data32bit)
bit64.show(a,type,flag)
bit64.plus(a,b,flag1,flag2)
bit64.minus(a,b,flag1,flag2)
bit64.pide(a,b,flag1,flag2)
bit64.shift(a,b,flag)
bit64.strtoll(data, base)

bit64.to32(data64bit)

64bit 数据转成 32bit 输出

参数

传入值类型	解释
string	9 字节数据

返回值

返回值类型	解释
any	根据 64bit 数据输出 int 或者 number

例子

无

---

bit64.to64(data32bit)

32bit 数据转成 64bit 数据

参数

传入值类型　　解释

int/number　32bit 数据

返回值

返回值类型　　解释

string　　9 字节数据

例子

无

---

bit64.show(a,type,flag)

64bit 数据格式化打印成字符串，用于显示值

参数

传入值类型	解释
string	需要打印的 64bit 数据
int	进制，10=10 进制，16=16 进制，默认 10，只支持 10 或者 16
boolean	整形是否按照无符号方式打印，true 是，false 不是，默认 false，浮点忽略

返回值

返回值类型　　解释

string　　可以打印的值

例子

无

bit64.plus(a,b,flag1,flag2)

64bit 数据加,a+b,a 和 b 中有一个为浮点，则按照浮点运算

参数

传入值类型	解释
string	a
string/int/number	b
boolean	整形运算时是否按照无符号方式，true 是，false 不是，默认 false，浮点运算忽略
boolean	浮点运算结果是否要强制转成整数，true 是，false 不是，默认 false，整形运算忽略

返回值

返回值类型	解释
string	9 字节数据

例子

无

bit64.minus(a,b,flag1,flag2)

64bit 数据减,a-b,a 和 b 中有一个为浮点，则按照浮点运算

参数

传入值类型	解释
string	a

传入值类型	解释
string/int/number	b
boolean	整形运算时是否按照无符号方式，true 是，false 不是，默认 false，浮点运算忽略
boolean	浮点运算结果是否要强制转成整数，true 是，false 不是，默认 false，整形运算忽略

返回值

返回值类型	解释
string	9 字节数据

例子

无

---

bit64.multi(a,b,flag1,flag2)

64bit 数据乘,a*b,a 和 b 中有一个为浮点，则按照浮点运算

参数

传入值类型	解释
string	a
string/int/number	b
boolean	整形运算时是否按照无符号方式，true 是，false 不是，默认 false，浮点运算忽略
boolean	浮点运算结果是否要强制转成整数，true 是，false 不是，默认 false，整形运算忽略

返回值

返回值类型　　解释

string　　9 字节数据

例子

无

---

bit64.pide(a,b,flag1,flag2)

64bit 数据除,a/b,a 和 b 中有一个为浮点，则按照浮点运算

参数

传入值类型	解释
string	a
string/int/number	b
boolean	整形运算时是否按照无符号方式，true 是，false 不是，默认 false，浮点运算忽略
boolean	浮点运算结果是否要强制转成整数，true 是，false 不是，默认 false，整形运算忽略

返回值

返回值类型　　解释

string　　9 字节数据

例子

无

---

bit64.shift(a,b,flag)

64bit 数据位移 a>>b 或者 a<<b

参数

传入值类型	解释
string	a
int	b
boolean	位移方向，true 左移<<，false 右移>>，默认 false

返回值

返回值类型	解释
string	9 字节数据

例子

无

---

bit64.strtoll(data, base)

将字符串转为 LongLong 数据

参数

传入值类型	解释
string	待转换的数据,必须存在
int	转换进制，默认 10，可选 16 或 8

返回值

返回值类型	解释
string	9 字节数据

例子

```
-- 本 API 于 2023.10.27 添加
-- 提醒，如果转换失败，是返回 9 个字节的 0x00
local data = bit64.strtoll("864040064024194", 10)log.info("data", data:toHex())log.info("data", bit64.show(data))
```

camera - 摄像头

已适配 Air105 Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 camera 的 demo 例子

常量

camera.init(InitReg_or_cspi_id, cspi_speed, mode, is_msb, rx_bit, seq_type, is_ddr, only_y, scan_mode, w, h)

camera.on(id, event, func)

camera.start(id)

camera.stop(id)

camera.close(id)

camera.capture(id, save_path, quality)

camera.video(id, w, h, out_path)

camera.startRaw(id, w, h, buff)

camera.getRaw(id)

camera.preview(id, onoff)

常量

常量	类型	解释
camera.AUTO	number	摄像头工作在自动模式
camera.SCAN	number	摄像头工作在扫码模式，只输出 Y 分量

camera.init(InitReg_or_cspi_id, cspi_speed, mode, is_msb, rx_bit, seq_type, is_ddr, only_y, scan_mode, w, h)

初始化摄像头

参数

传入值类型	解释
table/integer	如果是 table,则是 DVP 摄像头的配置见 demo/camera/AIR105,同时忽略后续参数;如果是数字,则是 camera spi 总线序号
int	camera spi 总线速度
int	camera spi 模式,0~3
int	字节的 bit 顺序是否是 msb,0 否 1 是
int	同时接收 bit 数,1,2,4
int	byte 序列,0~1
int	双边沿采样配置,0 不启用,其他值根据实际 SOC 决定
int	只接收 Y 分量，0 不启用，1 启用，扫码必须启用，否则会失败
int	工作模式，camera.AUTO 自动,camera.SCAN 扫码
int	摄像头宽度
int	摄像头高度

返回值

返回值类型	解释
int/false	成功返回 camera_id，失败返回 false

例子

```
camera_id = camera.init(GC032A_InitReg)--屏幕输出 rgb 图像--初始化后需要 start 才开始输出/扫码
camera.start(camera_id)--开始指定的 camera
```

camera.on(id, event, func)

注册摄像头事件回调

参数

传入值类型	解释
int	camera id, camera 0 写 0, camera 1 写 1
string	事件名称
function	回调方法

返回值

返回值类型	解释
nil	无返回值

例子

```
camera.on(0, "scanned", function(id, str)
--id int camera id
--str 多种类型 false 摄像头没有正常工作，true 拍照模式下拍照成功并保存完成， int 原始数据模式下本次返回的数据大小， string 扫码模式下扫码成功后的解码值
    print(id, str)end)
```

---

camera.start(id)

开始指定的 camera

参数

传入值类型	解释
int	camera id,例如 0

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
camera.start(0)
```

---

camera.stop(id)

停止指定的 camera

参数

传入值类型	解释
int	camera id,例如 0

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
camera.stop(0)
```

---

camera.close(id)

关闭指定的 camera，释放相应的 IO 资源

参数

传入值类型	解释
int	camera id,例如 0

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
camera.close(0)
```

---

camera.capture(id, save_path, quality)

camera 拍照

参数

传入值类型	解释
int	camera id,例如 0
string/zbuff/nil	save_path,文件保存路径，空则写在上次路径里，默认是/capture.jpg，如果是 zbuff，则将图片保存在 buff 内不写入文件系统
int	quality, jpeg 压缩质量，1 最差，占用空间小，3 最高，占用空间最大而且费时间，默认 1

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false,真正完成后通过 camera.on 设置的回调函数回调接收到的长度

例子

```
camera.capture(0)
```

camera.video(id, w, h, out_path)

camera 输出视频流到 USB

参数

传入值类型	解释
int	camera id,例如 0
int	宽度
int	高度
int	输出路径，目前只能用虚拟串口 0

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
camera.video(0, 320, 240, uart.VUART_0)
```

camera.startRaw(id, w, h, buff)

启动 camera 输出原始数据到用户的 zbuff 缓存区，输出 1fps 后会停止，并通过 camera.on 设置的回调函数回调接收到的长度，如果需要再次输出，请调用 camera.getRaw

参数

传入值类型	解释
int	camera id,例如 0
int	宽度

传入值类型	解释
int	高度
zbuff	用于存放数据的缓存区，大小必须不小于 w X h X 2 byte

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
camera.startRaw(0, 320, 240, buff)
```

---

camera.getRaw(id)

再次启动 camera 输出原始数据到用户的 zbuff 缓存区，输出 1fps 后会停止，并通过 camera.on 设置的回调函数回调接收到的长度，如果需要再次输出，请继续调用本 API

参数

传入值类型	解释
int	camera id,例如 0

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
camera.getRaw(0)
```

---

camera.preview(id, onoff)

启停 camera 预览功能，直接输出到 LCD 上，只有硬件支持的 SOC 可以运行

参数

传入值类型	解释
int	camera id,例如 0
boolean	true 开启，false 停止

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
camera.preview(1, true)
```

cc - 通话功能

已适配 Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 cc 的 demo 例子

```
cc.lastNum()
cc.dial(sim_id,number)
cc.hangUp(sim_id)
cc.accept(sim_id)
cc.init(multimedia_id)
```

```
cc.record(on_off,upload_zbuff1, upload_zbuff2, download_zbuff1, download_zbuff2)

cc.quality()

cc.on(event, func)
```

cc.lastNum()

获取最后一次通话的号码

参数

无

返回值

返回值类型	解释
string	获取最后一次通话的号码

例子

无

---

cc.dial(sim_id,number)

拨打电话

参数

传入值类型	解释
number	sim_id
number	电话号码

返回值

返回值类型	解释
bool	拨打电话成功与否

例子

无

---

cc.hangUp(sim_id)

挂断电话

参数

传入值类型	解释
number	sim_id

返回值

无

例子

无

---

cc.accept(sim_id)

接听电话

参数

传入值类型	解释
number	sim_id

返回值

返回值类型　　解释

bool　　接听电话成功与否

例子

无

---

cc.init(multimedia_id)

初始化电话功能

参数

传入值类型　　解释

number　multimedia_id　多媒体 id

返回值

返回值类型　解释

bool　　成功与否

例子

无

---

cc.record(on_off,upload_zbuff1, upload_zbuff2, download_zbuff1, download_zbuff2)

录音通话

参数

传入值类型　　解释

传入值类型	解释
boolean	开启关闭通话录音功能，false 或者 nil 关闭，其他开启
zbuff	上行数据保存区 1,zbuff 创建时的空间容量必须是 640 的倍数,下同
zbuff	上行数据保存区 2,和上行数据保存区 1 组成双缓冲区
zbuff	下行数据保存区 1
zbuff	下行数据保存区 2,和下行数据保存区 1 组成双缓冲区

返回值

返回值类型	解释
bool	成功与否，如果处于通话状态，会失败

例子

```
buff1 = zbuff.create(6400,0,zbuff.HEAP_AUTO)buff2 =
zbuff.create(6400,0,zbuff.HEAP_AUTO)buff3 =
zbuff.create(6400,0,zbuff.HEAP_AUTO)buff4 =
zbuff.create(6400,0,zbuff.HEAP_AUTO)cc.on("record",
function(type, buff_point)

 log.info(type, buff_point) -- type==true 是下行数据，false 是上行
数据 buff_point 指示双缓存中返回了哪一个 end)

cc.record(true, buff1, buff2, buff3, buff4)
```

---

cc.quality()

获取当前通话质量

参数

无

返回值

返回值类型	解释

返回值类型	解释
int	1 为低音质(8K)，2 为高音质(16k)，0 没有在通话

例子

无

---

cc.on(event, func)

注册通话回调

参数

传入值类型	解释
string	事件名称 音频录音数据为”record”
function	回调方法

返回值

返回值类型	解释
nil	无返回值

例子

```
cc.on("record", function(type, buff_point)
 log.info(type, buff_point) -- type==true 是下行数据，false 是上行数据 buff_point 指示双缓存中返回了哪一个 end)
```

codec - 多媒体-编解码

已适配 Air101/Air103 Air601 Air105 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 codec 的 demo 例子

常量

codec.create(type, isDecoder, quality)

codec.info(decoder, file_path)

codec.data(decoder, out_buff)

codec.encode(coder, in_buffer, out_buffer, mode)

codec.release(coder)

常量

常量	类型	解释
codec.MP3	number	MP3 格式
codec.WAV	number	WAV 格式
codec.AMR	number	AMR-NB 格式，一般意义上的 AMR
codec.AMR_WB	number	AMR-WB 格式

codec.create(type, isDecoder, quality)

创建编解码用的 codec

参数

传入值类型	解释
int	多媒体类型，目前支持 codec.MP3 codec.AMR
boolean	是否是解码器，true 解码器，false 编码器，默认 true，是解码器
int	编码等级，部分 bsp 有内部编码器，可能需要提前输入编码等级，比如 air780ep 的内部 amr 编码器

返回值

返回值类型	解释

返回值类型	解释
userdata	成功返回一个数据结构,否则返回 nil

例子

```
-- 创建解码器
local decoder = codec.create(codec.MP3)--创建一个 mp3 的 decoder
-- 创建编码器
local encoder = codec.create(codec.AMR, false)--创建一个 amr 的 encoder
-- 创建编码器
local encoder = codec.create(codec.AMR_WB, false, 8)--创建一个 amr-wb 的 encoder，编码等级默认 8
```

---

codec.info(decoder, file_path)

decoder 从文件中解析出音频信息

参数

传入值类型	解释
userdata	解码用的 decoder
string	文件路径

返回值

返回值类型	解释
boolean	是否成功解析
int	音频格式
int	声音通道数
int	采样频率
int	采样位数

返回值类型　　解释

boolean　　是否有符号

例子

```
local result, audio_format, num_channels, sample_rate,
bits_per_sample, is_signed= codec.info(coder, "xxx")
```

---

codec.data(decoder, out_buff)

decoder 从文件中解析出原始音频数据，比如从 MP3 文件里解析出 PCM 数据，这里的文件路径已经在 codec.info 传入，不需要再次传入

参数

传入值类型　　解释

userdata	解码用的 decoder
zbuff	存放输出数据的 zbuff，空间必须不少于 16KB
int	最少解码出多少字节的音频数据,默认 16384

返回值

返回值类型　　解释

boolean　　是否成功解析

例子

```
-- 大内存设备
local buff = zbuff.create(16*1024)
local result = codec.data(coder, buff)-- 小内存设备
local buff = zbuff.create(8*1024)
local result = codec.data(coder, buff, 4096)
```

---

codec.encode(coder, in_buffer, out_buffer, mode)

编码音频数据，由于 flash 和 ram 空间一般比较有限，除了部分 bsp 有内部 amr 编码功能，目前只支持 amr-nb 编码

参数

传入值类型	解释
userdata	codec.create 创建的编解码用的 coder
zbuff	输入的数据,zbuff 形式,从 0 到 used
zbuff	输出的数据,zbuff 形式,自动添加到 buff 的尾部,如果空间大小不足,会自动扩展,但是会额外消耗时间,甚至会失败,所以尽量一开始就给足空间
int	amr_nb 的编码等级 0~7(即 MR475~MR122)值越大消耗的空间越多,音质越高,默认 0 amr_wb 的编码等级 0~8,值越大消耗的空间越多,音质越高,默认 0

返回值

返回值类型	解释
boolean	成功返回 true,失败返回 false

例子

```
codec.encode(amr_coder, inbuf, outbuf, codec.AMR_)
```

---

codec.release(coder)

释放编解码用的 coder

参数

传入值类型	解释

传入值类型	解释
coder	codec.create 创建的编解码用的 coder

返回值

无

例子

```
codec.release(coder)
```

coremark - 跑分

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

coremark.run()

开始跑分

参数

无

返回值

返回值类型	解释
nil	无返回值,结果直接打印在日志中

例子

```
-- 大部分情况下，这个库都不会包含在正式版固件里
-- 若需使用,可以参考 wiki 文档自行编译或使用云编译
-- https://wiki.luatos.com/develop/compile.html
```

```
-- 跑分的 main.lua 应移除硬狗代码，防止重启
-- 若设备支持自动休眠，应关闭休眠功能
-- 若设备支持更多的频率运行，建议设置到最高频率
-- 使用 -O3 比 -O2 -Os 的分数更高，通常情况下
-- 会一直独占线程到执行完毕，然后在控制台输出结果
coremark.run()
-- 跑分图一乐^_^
```

crypto - 加解密和 hash 函数

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 crypto 的 demo 例子

crypto.md5(str)

crypto.hmac_md5(str, key)

crypto.sha1(str)

crypto.hmac_sha1(str, key)

crypto.sha256(str)

crypto.hmac_sha256(str, key)

crypto.sha512(str)

crypto.hmac_sha512(str, key)

crypto.cipher_encrypt(type, padding, str, key, iv)

crypto.cipher_decrypt(type, padding, str, key, iv)

crypto.crc16(method, data, poly, initial, finally, inReversem outReverse)

crypto.crc16_modbus(data, start)

crypto.crc32(data)

crypto.crc8(data, poly, start, revert)

crypto.crc7(data, poly, start)

crypto.trng(len)

crypto.totp(secret,time)

crypto.base64_encode(data)

crypto.base64_decode(data)

crypto.cipher_list()

crypto.cipher_suites()

crypto.md_file(tp, path, hmac)

crypto.md(tp, data, hmac)

crypto.hash_init(tp)

crypto.hash_update(stream, data)

crypto.hash_finish(stream)

crypto.checksum(data, mode)

crypto.md5(str)

计算 md5 值

参数

传入值类型	解释
string	需要计算的字符串

返回值

返回值类型	解释
string	计算得出的 md5 值的 hex 字符串

例子

```
-- 计算字符串"abc"的 md5
log.info("md5", crypto.md5("abc"))
```

---

crypto.hmac_md5(str, key)

计算 hmac_md5 值

参数

传入值类型　　解释

string　　需要计算的字符串

string　　密钥

返回值

返回值类型　　解释

string　　计算得出的 hmac_md5 值的 hex 字符串

例子

```
-- 计算字符串"abc"的 hmac_md5
log.info("hmac_md5", crypto.hmac_md5("abc", "1234567890"))
```

---

crypto.sha1(str)

计算 sha1 值

参数

传入值类型　　解释

string　　需要计算的字符串

返回值

返回值类型　　解释

string　　计算得出的 sha1 值的 hex 字符串

例子

```
-- 计算字符串"abc"的 sha1
log.info("sha1", crypto.sha1("abc"))
```

---

crypto.hmac_sha1(str, key)

计算 hmac_sha1 值

参数

传入值类型	解释
string	需要计算的字符串
string	密钥

返回值

返回值类型	解释
string	计算得出的 hmac_sha1 值的 hex 字符串

例子

```
-- 计算字符串"abc"的 hmac_sha1
log.info("hmac_sha1", crypto.hmac_sha1("abc", "1234567890"))
```

---

crypto.sha256(str)

计算 sha256 值

参数

传入值类型	解释
string	需要计算的字符串

返回值

返回值类型	解释
string	计算得出的 sha256 值的 hex 字符串

例子

```
-- 计算字符串"abc"的 sha256
log.info("sha256", crypto.sha256("abc"))
```

---

crypto.hmac_sha256(str, key)

计算 hmac_sha256 值

参数

传入值类型	解释
string	需要计算的字符串
string	密钥

返回值

返回值类型	解释
string	计算得出的 hmac_sha256 值的 hex 字符串

例子

```
-- 计算字符串"abc"的 hmac_sha256
log.info("hmac_sha256", crypto.hmac_sha256("abc", "1234567890"))
```

---

crypto.sha512(str)

计算 sha512 值

参数

传入值类型	解释
string	需要计算的字符串

返回值

返回值类型	解释
string	计算得出的 sha512 值的 hex 字符串

例子

```
-- 计算字符串"abc"的 sha512
log.info("sha512", crypto.sha512("abc"))
```

---

crypto.hmac_sha512(str, key)

计算 hmac_sha512 值

参数

传入值类型	解释
string	需要计算的字符串
string	密钥

返回值

返回值类型	解释
string	计算得出的 hmac_sha512 值的 hex 字符串

例子

```
-- 计算字符串"abc"的 hmac_sha512
log.info("hmac_sha512", crypto.hmac_sha512("abc", "1234567890"))
```

---

crypto.cipher_encrypt(type, padding, str, key, iv)

对称加密

参数

传入值类型	解释
string	算法名称，例如 AES-128-ECB/AES-128-CBC，可查阅 crypto.cipher_list()
string	对齐方式，支持 PKCS7/ZERO/ONE_AND_ZEROS/ZEROS_AND_LEN/NONE
string	需要加密的数据
string	密钥,需要对应算法的密钥长度
string	IV 值，非 ECB 算法需要

返回值

返回值类型	解释
string	加密后的字符串

例子

```
-- 计算 AES
local data = crypto.cipher_encrypt("AES-128-ECB", "PKCS7", "1234567890123456", "1234567890123456")
local data2 = crypto.cipher_encrypt("AES-128-CBC", "PKCS7", "1234567890123456", "1234567890123456", "1234567890666666")
```

---

crypto.cipher_decrypt(type, padding, str, key, iv)

对称解密

参数

传入值类型	解释
string	算法名称，例如 AES-128-ECB/AES-128-CBC，可查阅

传入值类型	解释
	crypto.cipher_list()
string	对齐方式，支持 PKCS7/ZERO/ONE_AND_ZEROS/ZEROS_AND_LEN/NONE
string	需要解密的数据
string	密钥,需要对应算法的密钥长度
string	IV 值，非 ECB 算法需要

返回值

返回值类型	解释
string	解密后的字符串

例子

```
-- 用 AES 加密,然后用 AES 解密
local data = crypto.cipher_encrypt("AES-128-ECB", "PKCS7", "1234567890123456", "1234567890123456")
local data2 = crypto.cipher_decrypt("AES-128-ECB", "PKCS7", data, "1234567890123456")
-- data 的 hex 为 757CCD0CDC5C90EADBEEECF638DD0000
-- data2 的值为 1234567890123456
```

---

crypto.crc16(method, data, poly, initial, finally, inReversem outReverse)

计算 CRC16

参数

传入值类型	解释

传入值类型	解释
string	CRC16 模式（”IBM”,”MAXIM”,”USB”,”MODBUS”,”CCITT”,”CCITT-FALSE”,”X25”,”XMODEM”,”DNP”,”USER-DEFINED”）
string	字符串
int	poly 值
int	initial 值
int	finally 值
int	输入反转,1 反转,默认 0 不反转
int	输入反转,1 反转,默认 0 不反转

返回值

返回值类型	解释
int	对应的 CRC16 值

例子

```
-- 计算 CRC16
local crc = crypto.crc16("")
```

---

crypto.crc16_modbus(data, start)

直接计算 modbus 的 crc16 值

参数

传入值类型	解释
string	数据
int	初始化值,默认 0xFFFF

返回值

返回值类型　　解释

int　　对应的 CRC16 值

例子

```
-- 计算 CRC16 modbus

local crc = crypto.crc16_modbus(data)-- 2023.11.06 新增初始值设置
crc = crypto.crc16_modbus(data, 0xFFFF)
```

---

crypto.crc32(data)

计算 crc32 值

参数

传入值类型 解释

string　　数据

返回值

返回值类型　　解释

int　　对应的 CRC32 值

例子

```
-- 计算 CRC32

local crc = crypto.crc32(data)
```

---

crypto.crc8(data, poly, start, revert)

计算 crc8 值

参数

传入值类型	解释
string	数据
int	crc 多项式，可选，如果不写，将忽略除了数据外所有参数
int	crc 初始值，可选，默认 0
boolean	是否需要逆序处理，默认否

返回值

返回值类型	解释
int	对应的 CRC8 值

例子

```
-- 计算 CRC8

local crc = crypto.crc8(data)local crc = crypto.crc8(data, 0x31, 0xff, false)
```

---

crypto.crc7(data, poly, start)

计算 crc7 值

参数

传入值类型	解释
string	数据
int	crc 多项式，可选，默认 0xE5
int	crc 初始值，可选，默认 0x00

返回值

返回值类型	解释
int	对应的 CRC7 值

例子

```
-- 计算 CRC7，本 API 于 2023.10.07 新增
local crc = crypto.crc7(data)
local crc = crypto.crc7(data, 0x31, 0xff)
```

---

crypto.trng(len)

生成真随机数

参数

传入值类型	解释
int	数据长度

返回值

返回值类型	解释
string	指定随机数字符串

例子

```
-- 生成 32 位随机数 ir
local r = crypto.trng(4)
local _, ir = pack.unpack(r, "I")
```

---

crypto.totp(secret,time)

计算 TOTP 动态密码的结果

参数

传入值类型	解释

传入值类型	解释
string	网站提供的密钥（就是 BASE32 编码后的结果）
int	可选，时间戳，默认当前时间

返回值

返回值类型	解释
int	计算得出的六位数结果 计算失败返回 nil

例子

```
--使用当前系统时间计算
local otp = crypto.totp("asdfassdfasdfass")
```

---

crypto.base64_encode(data)

将数据进行 base64 编码

参数

传入值类型	解释
string	待编码的数据

返回值

返回值类型	解释
string	编码后的数据

例子

```
-- 本函数与 string.toBase64 是同一个
local data = "123"
local bdata = crypto.base64_encode(data)
log.info("base64", "encode", data, bdata)
```

```
data = crypto.base64_decode(data)
log.info("base64", "decode", data, bdata)
```

---

crypto.base64_decode(data)

将数据进行 base64 解码

参数

传入值类型	解释
string	待解码的数据

返回值

返回值类型	解释
string	解码后的数据

例子

```
-- 本函数与 string.fromBase64 是同一个
local data = "123"
local bdata = crypto.base64_encode(data)
log.info("base64", "encode", data, bdata)
data = crypto.base64_decode(data)
log.info("base64", "decode", data, bdata)
```

---

crypto.cipher_list()

获取当前固件支持的 cipher 列表

参数

无

返回值

返回值类型	解释
table	本固件支持的 cipher 列表,字符串数组

例子

```
-- 本 API 于 2022.07.27 添加
local ciphers = crypto.cipher_list()
if ciphers then
        log.info("crypto", "ciphers list", json.encode(ciphers))
end
```

---

crypto.cipher_suites()

获取当前固件支持的 cipher suites 列表

参数

无

返回值

返回值类型	解释
table	本固件支持的 cipher suites 列表,字符串数组

例子

```
-- 本 API 于 2022.11.16 添加
local suites = crypto.cipher_suites()
if suites then
    log.info("crypto", "ciphers suites", json.encode(suites))
end
```

crypto.md_file(tp, path, hmac)

计算文件的 hash 值(md5/sha1/sha256 及 hmac 形式)

参数

传入值类型	解释
string	hash 类型，大小字母，例如 “MD5” “SHA1” “SHA256”
string	文件路径，例如 /luadb/logo.jpg
string	hmac 值,可选

返回值

返回值类型	解释
string	HEX 过的 hash 值,若失败会无返回值

例子

```
-- 无 hmac 的 hash 值
log.info("md5", crypto.md_file("MD5", "/luadb/logo.jpg"))
log.info("sha1", crypto.md_file("SHA1", "/luadb/logo.jpg"))
log.info("sha256", crypto.md_file("SHA256", "/luadb/logo.jpg"))
-- 带 hmac 的 hash 值
log.info("hmac_md5", crypto.md_file("MD5", "/luadb/logo.jpg",
"123456"))
log.info("hmac_sha1", crypto.md_file("SHA1", "/luadb/logo.jpg",
"123456"))
log.info("hmac_sha256", crypto.md_file("SHA256",
"/luadb/logo.jpg", "123456"))
```

crypto.md(tp, data, hmac)

计算数据的 hash 值(md5/sha1/sha256 及 hmac 形式)

参数

传入值类型	解释
string	hash 类型，大小字母，例如 “MD5” “SHA1” “SHA256”
string	待处理的数据
string	hmac 值,可选

返回值

返回值类型	解释
string	HEX 过的 hash 值,若失败会无返回值

例子

```
-- 无 hmac 的 hash 值
log.info("md5", crypto.md("MD5", "1234567890"))
log.info("sha1", crypto.md("SHA1", "1234567890"))
log.info("sha256", crypto.md("SHA256", "1234567890"))
-- 带 hmac 的 hash 值
log.info("hmac_md5", crypto.md("MD5", "1234567890", "123456"))
log.info("hmac_sha1", crypto.md("SHA1", "1234567890", "123456"))
log.info("hmac_sha256", crypto.md("SHA256", "1234567890",
"123456"))
```

---

crypto.hash_init(tp)

创建流式 hash 用的 stream

参数

传入值类型	解释
string	hash 类型，大写字母，例如 “MD5” “SHA1” “SHA256”
string	hmac 值，可选

返回值

返回值类型	解释
userdata	成功返回一个数据结构,否则返回 nil

例子

```
-- 无 hmac 的 hash stream
local md5_stream = crypto.hash_init("MD5")
local sha1_stream = crypto.hash_init("SHA1")
local sha256_stream = crypto.hash_init("SHA256")
-- 带 hmac 的 hash stream
local md5_stream = crypto.hash_init("MD5", "123456")
local sha1_stream = crypto.hash_init("SHA1", "123456")
local sha256_stream = crypto.hash_init("SHA256", "123456")
```

---

crypto.hash_update(stream, data)

流式 hash 更新数据

参数

传入值类型	解释
userdata	crypto.hash_init()创建的 stream，必选
string	待计算的数据,必选
return	无

返回值

无

例子

```
crypto.hash_update(stream, "OK")
```

---

crypto.hash_finish(stream)

获取流式 hash 校验值并释放创建的 stream

参数

传入值类型	解释
userdata	crypto.hash_init()创建的 stream,必选

返回值

返回值类型	解释
string	成功返回计算得出的流式 hash 值的 hex 字符串，失败无返回

例子

```
local hashResult = crypto.hash_finish(stream)
```

---

crypto.checksum(data, mode)

计算 checksum 校验和

参数

传入值类型	解释
string	待计算的数据,必选
int	模式,累加模式, 0 - 异或, 1 - 累加, 默认为 0

返回值

返回值类型	解释
int	checksum 值,校验和

例子

```
-- 本函数在 2022.12.28 添加
-- 单纯计算 checksum 值
local ck = crypto.checksum("OK")
log.info("checksum", "ok", string.format("%02X", ck))-- 第二个参数 mode 在 2023.5.23 日添加
```

eink - 墨水屏操作库

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 eink 的 demo 例子

常量

eink.init(tp, args,spi_device)

eink.setup(full, spiid, pin_busy, pin_reset, pin_dc, pin_cs)

eink.sleep()

eink.clear(color, force)

eink.setWin(width, height, rotate)

eink.setFont(font)

eink.print(x, y, str, colored)

eink.show(x, y, noClear)

eink.draw(buff, buff2, noclear)

eink.line(x, y, x2, y2, colored)

eink.rect(x, y, x2, y2, colored, fill)

eink.circle(x, y, radius, colored, fill)

eink.qrcode(x, y, str, size)

eink.bat(x, y, bat)

eink.weather_icon(x, y, code)

eink.model(m)

eink.drawXbm(x, y, w, h, data)

eink.setCtx(index)

eink.async(index)

常量

常量	类型	解释
eink.MODEL_1in02d	number	1.02 寸 d
eink.MODEL_1in54	number	1.54 寸
eink.MODEL_1in54_V2	number	1.54 寸_V2
eink.MODEL_1in54b	number	1.54 寸 b
eink.MODEL_1in54b_V2	number	1.54 寸 b_V2
eink.MODEL_1in54_V3	number	1.54 寸_V3
eink.MODEL_1in54c	number	1.54 寸 c
eink.MODEL_1in54r	number	1.54 寸三色屏 152*152
eink.MODEL_2in13	number	2.13 寸
eink.MODEL_2in13bc	number	2.13 寸 bc
eink.MODEL_2in13d	number	2.13 寸 d
eink.MODEL_2in13_V2	number	2.13 寸 V2
eink.MODEL_2in54b_V3	number	2.13 寸 b_V3
eink.MODEL_2in66	number	2.66 寸
eink.MODEL_2in66b	number	2.66 寸 b
eink.MODEL_2in7	number	2.7 寸
eink.MODEL_2in7b	number	2.7 寸 b
eink.MODEL_2in9	number	2.9 寸
eink.MODEL_2in9_V2	number	2.9 寸_V2

常量	类型	解释
eink.MODEL_2in9bc	number	2.9 寸 bc
eink.MODEL_2in9b_V3	number	2.9 寸 b_V3
eink.MODEL_2in9d	number	2.9 寸 d
eink.MODEL_3in52	number	3.52 寸
eink.MODEL_3in7	number	3.7 寸
eink.MODEL_4in2	number	4.2 寸
eink.MODEL_4in2V2	number	4.2 寸 V2(ssd1683)
eink.MODEL_4in2bc	number	4.2 寸 b
eink.MODEL_4in2b_V2	number	4.2 寸 V2
eink.MODEL_5in65f	number	5.65 寸 f
eink.MODEL_5in83	number	5.83 寸
eink.MODEL_5in83bc	number	5.83 寸 bc
eink.MODEL_5in83_V2	number	5.83 寸 V2
eink.MODEL_5in83b_V2	number	5.83 寸 bV2
eink.MODEL_7in5	number	7.5 寸
eink.MODEL_7in5_HD	number	7.5 寸 HD
eink.MODEL_7in5_V2	number	7.5 寸 V2
eink.MODEL_7in5bc	number	7.5 寸 bc
eink.MODEL_7in5b_HD	number	7.5 寸 b_HD
eink.MODEL_7in5b_V2	number	7.5 寸 b_V2
eink.font_opposansm12	font	12 号字体
eink.font_unifont_t_symbols	font	符号字体
eink.font_open_iconic_weather_6x_t	font	天气字体
eink.font_opposansm16	font	16 号字体
eink.font_opposansm18	font	18 号字体

常量	类型	解释
eink.font_opposansm20	font	20 号字体
eink.font_opposansm22	font	22 号字体
eink.font_opposansm24	font	24 号字体
eink.font_opposansm32	font	32 号字体
eink.font_opposansm12_chinese	font	12 号中文字体
eink.font_opposansm14_chinese	font	14 号中文字体
eink.font_opposansm16_chinese	font	16 号中文字体
eink.font_opposansm18_chinese	font	18 号中文字体
eink.font_opposansm20_chinese	font	20 号中文字体
eink.font_opposansm22_chinese	font	22 号中文字体
eink.font_opposansm24_chinese	font	24 号中文字体
eink.font_opposansm32_chinese	font	32 号中文字体

eink.init(tp, args,spi_device)

eink 显示屏初始化

参数

传入值类型	解释
number	eink 类型，当前支持：https://wiki.luatos.com/api/eink.html#id1
	附加参数,与具体设备有关：
	pin_busy（busy）
table	port：spi 端口,例如 0,1,2…如果为 device 方式则为”device”
	pin_dc：eink 数据/命令选择引脚
	pin_rst：eink 复位引脚
userdata	spi 设备,当 port = “device”时有效

返回值

无

例子

```
-- 初始化 spi0 的 eink.MODEL_4in2bc)
-- 注意:eink 初始化之前需要先初始化 spi
spi_eink = spi.deviceSetup(0,20,0,0,8,20000000,spi.MSB,1,1)
log.info("eink.init",eink.init(eink.MODEL_4in2bc,{port =
"device",pin_dc = 17, pin_pwr = 7,pin_rst = 19,direction = 2,w =
160,h = 80,xoffset = 1,yoffset = 26},spi_eink))
```

---

eink.setup(full, spiid, pin_busy, pin_reset, pin_dc, pin_cs)

初始化 eink

参数

传入值类型	解释
int	全屏刷新 0,局部刷新 1,默认是全屏刷新
int	所在的 spi,默认是 0
int	Busy 忙信号管脚
int	Reset 复位管脚
int	DC 数据命令选择管脚
int	CS 使能管脚

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

无

---

eink.sleep()

进入休眠模式，再次使用时需要重新初始化

参数

无

返回值

无

例子

无

---

eink.clear(color, force)

清除绘图缓冲区，默认不会马上刷新到设备

参数

传入值类型		解释
number	color	可选，默认 1。刷屏颜色
bool	force	可选，默认 false。如果为 true 则马上清屏

返回值

返回值类型	解释
nil	无返回值

例子

无

---

eink.setWin(width, height, rotate)

设置窗口

参数

传入值类型		解释
int	width	宽度
int	height	高度
int	rotate	显示方向,0/1/2/3, 相当于旋转 0 度/90 度/180 度/270 度

返回值

返回值类型	解释
nil	无返回值

例子

无

---

eink.getWin()

获取窗口信息

参数

无

返回值

返回值类型	解释

返回值类型	解释
int	width 宽
int	height 高
int	rotate 旋转方向

例子

无

---

eink.setFont(font)

设置字体

参数

传入值类型	解释
userdata	字体

返回值

无

例子

```
-- 设置为字体,对之后的 print 有效
eink.setFont(eink.font_opposansm12_chinese)
```

---

eink.print(x, y, str, colored)

绘制字符串

参数

传入值类型	解释

传入值类型	解释
int	x 坐标
int	y 坐标
string	字符串
int	颜色，可以是 0 或者 1，默认是 0

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 先设置字体，然后写字
-- 可用字体取决于具体的固件，如果没有你想要的大小,可以云编译一份自定义固件
-- font_opposansm8_chinese
-- font_opposansm10_chinese
-- font_opposansm12_chinese
-- font_opposansm14_chinese
-- font_opposansm16_chinese
eink.setFont(eink.font_opposansm12_ch inese)eink.print(10, 20, "LuatOS")
```

---

eink.show(x, y, noClear)

将缓冲区图像输出到屏幕

参数

传入值类型	解释
int	x 输出的 x 坐标,默认 0

传入值类型	解释
int	y 输出的 y 坐标,默认 0
bool	可选，默认 false。如果为 true 则不进行清屏，直接刷上新内容

返回值

返回值类型	解释
nil	无返回值

例子

无

---

eink.draw(buff, buff2, noclear)

直接输出数据到屏幕,支持双色数据

参数

传入值类型	解释
userdata	zbuff 指针
userdata	zbuff 指针
bool	可选，默认 false。如果为 true 则不进行清屏，直接刷上新内容

返回值

返回值类型	解释
nil	无返回值

例子

无

---

eink.line(x, y, x2, y2, colored)

缓冲区绘制线

参数

传入值类型	解释
int	起点 x 坐标
int	起点 y 坐标
int	终点 x 坐标
int	终点 y 坐标

返回值

返回值类型	解释
nil	无返回值

例子

```
eink.line(0, 0, 10, 20, 0)
```

---

eink.rect(x, y, x2, y2, colored, fill)

缓冲区绘制矩形

参数

传入值类型	解释
int	左上顶点 x 坐标
int	左上顶点 y 坐标
int	右下顶点 x 坐标

传入值类型	解释
int	右下顶点 y 坐标
int	默认是 0
int	是否填充,默认是 0,不填充

返回值

返回值类型	解释
nil	无返回值

例子

```
eink.rect(0, 0, 10, 20)eink.rect(0, 0, 10, 20,0, 1) -- Filled
```

---

eink.circle(x, y, radius, colored, fill)

缓冲区绘制圆形

参数

传入值类型	解释
int	圆心 x 坐标
int	圆心 y 坐标
int	半径
int	默认是 0
int	是否填充,默认是 0,不填充

返回值

返回值类型	解释
nil	无返回值

例子

```
eink.circle(0, 0, 10)eink.circle(0, 0, 10, 1, 1) -- Filled
```

---

eink.qrcode(x, y, str, size)

缓冲区绘制 QRCode

参数

传入值类型	解释
int	x 坐标
int	y 坐标
string	二维码的内容
int	显示大小 (注意:二维码生成大小与要显示内容和纠错等级有关,生成版本为 1-40(对应 21x21 - 177x177)的不定大小,如果和设置大小不同会自动在指定的区域中间显示二维码,如二维码未显示请查看日志提示)

返回值

返回值类型	解释
nil	无返回值

例子

无

---

eink.bat(x, y, bat)

缓冲区绘制电池

参数

传入值类型　　解释

int　　x 坐标

int　　y 坐标

int　　电池电压,单位毫伏

返回值

返回值类型　　解释

nil　　无返回值

例子

无

---

eink.weather_icon(x, y, code)

缓冲区绘制天气图标

参数

传入值类型　　解释

int　　x 坐标

int　　y 坐标

int　　天气代号

返回值

返回值类型　　解释

nil　　无返回值

例子

无

---

eink.model(m)

设置墨水屏驱动型号

参数

传入值类型	解释
int	型号名称，例如 eink.model(eink.MODEL_1in54_V2)

返回值

返回值类型	解释
nil	无返回值

例子

无

---

eink.drawXbm(x, y, w, h, data)

绘制位图

参数

传入值类型	解释
int	X 坐标
int	y 坐标
int	位图宽
int	位图高
int	位图数据,每一位代表一个像素

返回值

无

例子

```
-- 取模使用 PCtoLCD2002 软件即可
-- 在(0,0)为左上角,绘制 16x16 "今" 的位图
eink.drawXbm(0, 0, 16,16, string.char(

0x80,0x00,0x80,0x00,0x40,0x01,0x20,0x02,0x10,0x04,0x48,0x08,0x84,0x10,0x83,0x60,

0x00,0x00,0xF8,0x0F,0x00,0x08,0x00,0x04,0x00,0x04,0x00,0x02,0x00,0x01,0x80,0x00))
```

---

eink.setCtx(index)

切换颜色画板，适合多色墨水屏

参数

传入值类型	解释
int	颜色索引，黑色为 0，红色为 1

返回值

无

例子

```
-- 仅适合多色墨水屏，对单色墨水屏(只有黑白)的无效
eink.setCtx(1)-- 切换后，所有 drawXXX 都会操作在指定颜色的画板
```

---

eink.async(index)

异步方式，使用此方式需要先 require(“sysplus”) 之后 eink.clear().wait() eink.show().wait()进行刷屏

参数

传入值类型	解释
int	1 使用异步

返回值

无

例子

```
    eink.async(1)
    spi_eink = spi.deviceSetup(spi_id,pin_cs,0,0,8,20*1000*1000,spi.MSB,1,1)
    eink.init(eink.MODEL_1in54,
            {port = "device",pin_dc = pin_dc, pin_busy = pin_busy,pin_rst = pin_reset},
            spi_eink)
    eink.setWin(200, 200, 0)
    sys.wait(100)
    log.info("e-paper 1.54", "Testing Go")
    eink.print(30, 20, "LuatOS-AIR780E",0x00)
    eink.show().wait()
    log.info("e-paper 1.54", "Testing End")
```

errDump - 错误上报

已适配 Air101/Air103 Air601 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 errDump 的 demo 例子

示例

```
-- 基本用法, 10 分钟上报一次,如果有的话
if errDump then
    errDump.config(true, 600)
end
-- 附开源服务器端: https://gitee.com/openLuat/luatos-devlog
```

errDump.dump(zbuff, type, isDelete)

errDump.record(string)

errDump.config(enable, period, user_flag, custom_id, host, port)

errDump.dump(zbuff, type, isDelete)

手动读取异常日志，主要用于用户将日志发送给自己的服务器而不是 IOT 平台，如果在 errDump.config 配置了周期上传，则不能使用本函数

参数

传入值类型	解释
zbuff	日志信息缓存，如果为 nil 就不会读出，一般当
int	日志类型，目前只有 errDump.TYPE_SYS 和 errDump.TYPE_USR
boolean	是否删除日志

返回值

返回值类型	解释

返回值类型	解释
boolean	true 表示本次读取前并没有写入数据，false 反之，在删除日志前，最好再读一下确保没有新的数据写入了

例子

```
local result = errDump.dump(buff, errDump.TYPE_SYS, false)
--读出系统记录的异常日志
local result = errDump.dump(nil, errDump.TYPE_SYS, true)
--清除系统记录的异常日志
```

---

errDump.record(string)

写入用户的异常日志，注意最大只有 4KB，超过部分新的覆盖旧的，开启自动上传后会上传到合宙 IOT 平台

参数

传入值类型	解释
string	日志

返回值

返回值类型	解释
nil	无返回值

例子

```
errDump.record("socket long time no connect")
--记录下"socket long time no connect"
```

---

errDump.config(enable, period, user_flag, custom_id, host, port)

配置关键日志上传 IOT 平台，这里的日志包括引起 luavm 异常退出的日志和用户通过 record 写入的日志，类似于 air 的 errDump

参数

传入值类型	解释
boolean	是否启用记录功能，false 的话将不会记录任何日志
int	定时上传周期，单位秒，默认 600 秒，这个是自动上传时候后的重试时间时间，在开机后或者有 record 操作后会很快尝试上传到合宙 IOT 平台一次，如果为 0，则不会上传，由用户 dump 后自己上传自己的平台
string	用户的特殊标识，可以为空
string	设备识别号, 4G 设备默认是 imei,其他设备默认是 mcu.unique_id
string	服务器域名,默认 dev_msg1.openluat.com
int	服务器端口,默认

返回值

返回值类型	解释
nil	无返回值

例子

```
errDump.config(true, 3600, "12345678")
--一个小时尝试上次一次，上传时会在 imei 后附加上 12345678
errDump.config(false) --关闭记录功能，不再上传
errDump.config(true, 0) --记录，但是不会主动上传，由用户实现上传功能
-- 2023.09.22 新增 custom_id 参数
errDump.config(true, 3600, nil, "ABC")
```

```
--一个小时尝试上次一次，上传时使用自定义的设备识别号 ABC
-- 2023.12.8 新增 host 和 port 参数
errDump.config(true, 3600, nil, nil, "dev_msg1.openluat.com", 12425)
```

fastlz - FastLZ 压缩

已适配 Air101/Air103 Air601 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

示例

```
-- 与 miniz 库的差异
-- 内存需求量小很多，miniz 库接近 200k，fastlz 只需要 32k+原始数据大小
-- 压缩比，miniz 的压缩比要好于 fastlz
-- 准备好数据 local bigdata = "123jfoiq4hlkfjbnasdilfhuqwo;hfashfp9qw38hrfaios;hfiuoaghfluaeisw"
-- 压缩之 local cdata = fastlz.compress(bigdata)
-- lua 的 字符串相当于有长度的 char[],可存放包括 0x00 的一切数据
if cdata then
    -- 检查压缩前后的数据大小
    log.info("fastlz", "before", #bigdata, "after", #cdata)
    log.info("fastlz", "cdata as hex", cdata:toHex())

    -- 解压，得到原文
    local udata = fastlz.uncompress(cdata)
    log.info("fastlz", "udata", udata)
end
```

fastlz.compress(data, level)

fastlz.uncompress(data, maxout)

fastlz.compress(data, level)

快速压缩

参数

传入值类型	解释
string	待压缩的数据，少于 400 字节的数据不建议压缩，且压缩后的数据不能大于 32k
int	压缩级别,默认 1，可选 1 或者 2, 2 的压缩比更高(有时候)

返回值

返回值类型	解释
string	若压缩成功,返回数据字符串，否则返回 nil

例子

```
-- 注意，压缩过程的内存消耗如下
-- 系统内存，固定 32k
-- lua 内存，原始数据的大小的 1.05 倍,最小占用 1024 字节.
```

---

fastlz.uncompress(data, maxout)

快速解压

参数

传入值类型	解释
string	待解压的数据
int	解压后的最大大小，默认是 4k，可按需调整

返回值

返回值类型	解释
string	若解压成功,返回数据字符串，否则返回 nil

例子

无

fatfs - 读写 fatfs 格式

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 fatfs 的 demo 例子

示例

```
-- 通常只使用 fatfs.mount 挂载 tf/sd 卡,其他操作走 io 库就可以了
fatfs.mount(mode,mount_point, spiid_or_spidevice, spi_cs, spi_speed, power_pin, power_on_delay, auto_format)
fatfs.getfree(mount_point)
fatfs.debug(value)
```

fatfs.mount(mode,mount_point, spiid_or_spidevice, spi_cs, spi_speed, power_pin, power_on_delay, auto_format)

挂载 fatfs

参数

传入值类型	解释
int	fatfs 模式,可选 fatfs.SPI,fatfs.SDIO,fatfs.RAM,fatfs.USB
string	虚拟文件系统的挂载点，默认是 /fatfs
int	传入 spi device 指针,或者 spi 的 id,或者 sdio 的 id
int	片选脚的 GPIO 号, spi 模式有效,若前一个参数传的是 spi device,这个参数就不需要传
int	SPI 最高速度,默认 10M，若前 2 个参数传的是 spi device,这个参数就不需要传
int	TF 卡电源控制脚,TF 卡初始前先拉低复位再拉高,如果没有,或者是内置电源控制方式,这个参数就不需要传
int	TF 卡电源复位过程时间,单位 ms,默认值是 1
bool	挂载失败是否尝试格式化,默认是 true,即自动格式化. 本参数在 2023.8.16 添加

返回值

返回值类型	解释
bool	成功返回 true，否则返回 nil 或者 false
string	失败的原因

例子

```
-- 方法 1，使用 SPI 模式
    local spiId = 2
    local result = spi.setup(
        spiId,--串口 id
        255, -- 不使用默认 CS 脚
        0,--CPHA
        0,--CPOL
```

```
        8,--数据宽度
        400*1000  -- 初始化时使用较低的频率
    )
    local TF_CS = pin.PB3
    gpio.setup(TF_CS, 1)
    --fatfs.debug(1) -- 若挂载失败,可以尝试打开调试信息,查找原因
    -- 提醒，若 TF/SD 模块带电平转换，通常不支持 10M 以上的波特率!!
    fatfs.mount(fatfs.SPI,"SD", spiId, TF_CS, 24000000)
    local data, err = fatfs.getfree("SD")
    if data then
        log.info("fatfs", "getfree", json.encode(data))
    else
        log.info("fatfs", "err", err)
    end
-- 往下的操作，使用 io.open("/sd/xxx", "w+") 等 io 库的 API 就可以了
```

---

fatfs.getfree(mount_point)

获取可用空间信息

参数

传入值类型	解释
string	挂载点，需要跟 fatfs.mount 传入的值一致

返回值

返回值类型	解释
table	若成功会返回 table,否则返回 nil
int	导致失败的底层返回值

例子

```
-- table 包含的内容有
-- total_sectors 总扇区数量
-- free_sectors 空闲扇区数量
-- total_kb 总字节数,单位 kb
-- free_kb 空闲字节数, 单位 kb
-- 注意,当前扇区大小固定在 512 字节

    local data, err = fatfs.getfree("SD")
    if data then
        log.info("fatfs", "getfree", json.encode(data))
    else
        log.info("fatfs", "err", err)
    end
```

---

fatfs.debug(value)

设置调试模式

参数

传入值类型	解释
bool	是否进入调试模式,true 代表进入调试模式,增加调试日志

返回值

返回值类型	解释
nil	无返回值

例子

无

fonts - 字体库

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

fonts.list(tp)

fonts.u8g2_get(name, tp)

fonts.u8g2_load(path, path)

fonts.list(tp)

返回固件支持的字体列表

参数

传入值类型	解释
string	类型，默认 u8g2，还可以是 lvgl

返回值

返回值类型	解释
table	字体列表

例子

```
-- API 新增于 2022-07-12
if fonts.list then
    log.info("fonts", "u8g2", json.encode(fonts.list("u8g2")))
end
```

fonts.u8g2_get(name, tp)

获取字体

参数

传入值类型	解释
string	字体名称，例如 opposansm8_chinese unifont_t_symbols
string	类型，默认 u8g2，还可以是 lvgl

返回值

返回值类型	解释
userdata	若字体存放,返回字体指针，否则返回 nil

例子

```
oppo_8 = fonts.get("opposansm8_chinese", "u8g2")
if oppo_8 then
    u8g2.SetFont(oppo_8)
else
    log.warn("fonts", "no such font opposansm8_chinese")
end-- 若使用云编译的自定义字库，使用方式如下
oppo_8 = fonts.get("oppo_bold_8", "u8g2") -- oppo_bold_8 是云编译界面的字库命名
if oppo_8 then
    u8g2.SetFont(oppo_8)
else
    log.warn("fonts", "no such font opposansm8_chinese")
end
```

---

fonts.u8g2_load(path, path)

从文件加载字体

参数

传入值类型	解释
string	字体路径，例如 /luadb/abc.bin
string	类型，默认 u8g2. 也支持 lvgl

返回值

返回值类型	解释
userdata	若字体存放,返回字体指针，否则返回 nil

例子

```
-- API 新增于 2022-07-11
-- 提醒: 若文件位于/luadb 下，不需要占用内存
-- 若文件处于其他路径，例如 tf/sd 卡，spi flash，会自动加载到内存，消耗 lua vm 的内存空间
-- 加载后请适当引用，不必反复加载同一个字体文件
oppo12 = fonts.load("/luadb/oppo12.bin")
if oppo12 then
    u8g2.SetFont(oppo12)
else
    log.warn("fonts", "no such font file oppo12.bin")
end
```

fota - 底层固件升级

已适配 Air105 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 fota 的 demo 例子

示例

```
-- 如果是从 http 获取升级包，那么看 demo/fota 就可以了
-- 以下是从其他途径获取更新包后，调用本 fota 库的基本逻辑
-- 逐段传入
sys.taskInit(function()
    fota.init()
    while 1 do
        local buf = xxx -- 这里是从其他途径获取的升级包片段
        -- buf 可以是 zbuff 也可以是 string
        -- 每次写入的数据长度最大不应超过 4k
        local result, isDone, cache = fota.run(buf)
        if not result then
            log.info("fota", "出错了")
            break
        end
        if isDone then
            while true do
                local succ,fotaDone  = fota.isDone()
                if not succ then
                    log.info("fota", "出错了")
                    break
                end
                if fotaDone then
                    log.info("fota", "已完成")
```

```
                    break
                end
                sys.wait(100)
            end
            break
        end
        sys.wait(100)
    endend)
-- 使用文件一次性传入
sys.taskInit(function()
    fota.init()
    fota.file("/xxx") -- 传入具体的路径
end)
```

fota.init(storge_location, len, param1)

fota.wait()

fota.run(buff)

fota.file(path)

fota.isDone()

fota.finish(is_ok)

fota.init(storge_location, len, param1)

初始化 fota 流程

参数

传入值类型	解释
int/string	fota 数据存储的起始位置 如果是 int，则是由芯片平台具体判断 如果是 string，则存储在文件系统中

传入值类型	解释
	如果为 nil，则由底层决定存储位置
int	数据存储的最大空间
userdata	param1，如果数据存储在 spiflash 时,为 spi_device

返回值

返回值类型	解释
boolean	成功返回 true，失败返回 false

例子

```
-- 初始化 fota 流程
local result = fota.init(0, 0x00300000, spi_device)     --由于105 的 flash 从 0x01000000 开始，所以 0 就是外部 spiflash
local result = fota.init()  --ec618 使用固定内部地址，所以不需要参数了
```

---

fota.wait()

等待底层 fota 流程准备好

参数

传入值类型	解释
boolean	是否完整走完流程，true 表示正确走完流程了

返回值

返回值类型	解释
boolean	准备好返回 true

例子

```
local isDone = fota.wait()
```

---

fota.run(buff)

写入 fota 数据

参数

传入值类型	解释
zbuff/string	fota 数据，尽量用 zbuff

返回值

返回值类型	解释
boolean	有异常返回 false，无异常返回 true
boolean	接收到最后一块返回 true
int	还未写入的数据量，超过 64K 必须做等待

例子

```
local result, isDone, cache = fota.run(buf) -- 写入 fota 流程
-- 提示：，如果传入的是 zbuff，写入成功后，请自行清空 zbuff 内的数据
```

---

fota.file(path)

从指定文件读取 fota 数据

参数

传入值类型	解释
string	文件路径

返回值

返回值类型	解释
boolean	有异常返回 false，无异常返回 true
boolean	接收到最后一块返回 true
int	还未写入的数据量，超过 64K 必须做等待

例子

```
local result, isDone, cache = fota.file("/xxx.bin") -- 写入 fota 流程-- 本 API 于 2023.03.23 添加
```

---

fota.isDone()

等待底层 fota 流程完成

参数

无

返回值

返回值类型	解释
boolean	有异常返回 false，无异常返回 true
boolean	写入到最后一块返回 true

例子

```
local result, isDone = fota.isDone()
```

---

fota.finish(is_ok)

结束 fota 流程

参数

传入值类型	解释
boolean	是否完整走完流程，true 表示正确走完流程了

返回值

返回值类型	解释
boolean	成功返回 true，失败返回 false

例子

```
-- 结束 fota 流程 local result = fota.finish(true)
```

fs - 文件系统额外操作

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 fs 的 demo 例子

fs.fsstat(path)
fs.fsize(path)

fs.fsstat(path)

获取文件系统信息

参数

传入值类型	解释
string	路径,默认”/”,可选

返回值

返回值类型	解释
boolean	获取成功返回 true,否则返回 false
int	总的 block 数量
int	已使用的 block 数量
int	block 的大小,单位字节
string	文件系统类型,例如 lfs 代表 littlefs

例子

```
-- 打印根分区的信息
log.info("fsstat", fs.fsstat("/"))
```

---

fs.fsize(path)

获取文件大小

参数

传入值类型	解释
string	文件路径

返回值

返回值类型	解释
int	文件大小,若获取失败会返回 0

例子

```
-- 打印 main.luac 的大小
log.info("fsize", fs.fsize("/main.luac"))
```

fskv - kv 数据库,掉电不丢数据

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 fskv 的 demo 例子

示例

```
-- 本库的目标是替代 fdb 库
-- 1. 兼容 fdb 的函数
-- 2. 使用 fdb 的 flash 空间,启用时也会替代 fdb 库
-- 3. 功能上与 EEPROM 是类似的
fskv.init()fskv.set("wendal", 1234)log.info("fskv", "wendal", fskv.get("wendal"))
--[[ fskv 与 fdb 的实现机制导致的差异
                    fskv          fdb
1. value 长度        4096           255
2. key 长度          63             64
3. 空间利用率(对比)   较低           较高
4. 读取速度          恒定           脏数据影响速度,非恒定
5. 写入数据          恒定           脏数据影响速度,非恒定
6. 均衡擦除          自动           自动]]
```

fskv.init()

fskv.set(key, value)

fskv.sett(key, skey, value)

fskv.get(key, skey)

fskv.del(key)

fskv.clear()

fskv.iter()

fskv.next(iter)

fskv.status()

fskv.init()

初始化 kv 数据库

参数

无

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
if fskv.init() then
    log.info("fdb", "kv 数据库初始化成功")
end
-- 关于清空 fdb 库
-- 下载工具是没有提供直接清除 fdb 数据的途径的，但有办法解决
-- 写一个 main.lua，执行 fskv.kvdb_init 后 执行 fskv.clear() 即可全清 fdb 数据.
```

---

fskv.set(key, value)

设置一对 kv 数据

参数

传入值类型

	解释
string	key 的名称,必填,不能空字符串
string	用户数据,必填,不能 nil，支持字符串/数值/table/布尔值，数据长度最大 4095 字节

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
-- 设置数据，字符串,数值,table,布尔值,均可
-- 但不可以是 nil，function，userdata，task
log.info("fdb", fskv.set("wendal", "goodgoodstudy"))
log.info("fdb", fskv.set("upgrade", true))
log.info("fdb", fskv.set("timer", 1))
log.info("fdb", fskv.set("bigd", {name="wendal",age=123}))
```

---

fskv.sett(key, skey, value)

设置 table 内的键值对数据

参数

传入值类型	解释
string	key 的名称,必填,不能空字符串
string	table 的 key 名称，必填，不能是空字符串
string	用户数据,必填,支持字符串/数值/table/布尔值，数据长度最大 4095

传入值类型	解释
字节	

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false/nil

例子

```
-- 本 API 在 2023.7.26 新增,注意与 set 函数区别
-- 设置数据, 字符串,数值,table,布尔值,均可
-- 但不可以是 function, userdata, task
log.info("fdb", fskv.sett("mytable", "wendal", "goodgoodstudy"))
log.info("fdb", fskv.sett("mytable", "upgrade", true))
log.info("fdb", fskv.sett("mytable", "timer", 1))
log.info("fdb", fskv.sett("mytable", "bigd",
{name="wendal",age=123}))
-- 下列语句将打印出 4 个元素的 table
log.info("fdb", fskv.get("mytable"),
json.encode(fskv.get("mytable")))
-- 注意: 如果 key 不存在, 或者原本的值不是 table 类型,将会完全覆盖
-- 例如下列写法,最终获取到的是 table,而非第一行的字符串
log.info("fdb", fskv.set("mykv", "123"))
log.info("fdb", fskv.sett("mykv", "age", "123"))
-- 保存的将是 {age:"123"}

-- 如果设置的数据填 nil, 代表删除对应的 key
log.info("fdb", fskv.sett("mykv", "name", "wendal"))
log.info("fdb", fskv.sett("mykv", "name")) -- 相当于删除
--
```

---

fskv.get(key, skey)

根据 key 获取对应的数据

参数

传入值类型	解释
string	key 的名称,必填,不能空字符串
string	可选的次级 key,仅当原始值为 table 时有效,相当于 fskv.get(key)[skey]

返回值

返回值类型	解释
any	存在则返回数据,否则返回 nil

例子

```lua
if fskv.init() then
    log.info("fdb", fskv.get("wendal"))
end
-- 若需要"默认值", 对应非 bool 布尔值, 可以这样写
local v = fskv.get("wendal") or "123"
```

---

fskv.del(key)

根据 key 删除数据

参数

传入值类型	解释
string	key 的名称,必填,不能空字符串

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

```
log.info("fdb", fskv.del("wendal"))
```

---

fskv.clear()

清空整个 kv 数据库

参数

无

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

```
-- 清空 fskv.clear()
```

---

fskv.iter()

kv 数据库迭代器

参数

无

返回值

返回值类型	解释
userdata	成功返回迭代器指针,否则返回 nil

例子

```
-- 清空
local iter = fskv.iter()
if iter then
    while 1 do
        local k = fskv.next(iter)
        if not k then
            break
        end
        log.info("fdb", k, "value", fskv.kv_get(k))
    end
end
```

---

fskv.next(iter)

kv 迭代器获取下一个 key

参数

传入值类型	解释
userdata	fskv.iter()返回的指针

返回值

返回值类型	解释
string	成功返回字符串 key 值，否则返回 nil

例子

```
-- 清空
local iter = fskv.iter()
if iter then
    while 1 do
        local k = fskv.next(iter)
        if not k then
            break
        end
        log.info("fskv", k, "value", fskv.get(k))
    end
end
```

---

fskv.status()

获取 kv 数据库状态

参数

无

返回值

返回值类型	解释
int	已使用的空间,单位字节
int	总可用空间，单位字节
int	总 kv 键值对数量，单位个

例子

```
local used, total,kv_count = fskv.status()
log.info("fdb", "kv", used,total,kv_count)
```

ftp - ftp 客户端

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 ftp 的 demo 例子

ftp.login(adapter,ip_addr,port,username,password)

ftp.command(cmd)

ftp.pull(local_name,remote_name)

ftp.push(local_name,remote_name)

ftp.close()

ftp.login(adapter,ip_addr,port,username,password)

FTP 客户端

参数

传入值类型	解释
int	适配器序号，只能是 socket.ETH0, socket.STA, socket.AP,如果不填,会选择平台自带的方式,然后是最后一个注册的适配器
string	ip_addr 地址
string	port 端口,默认 21
string	username 用户名
string	password 密码
bool/table	是否为 ssl 加密连接,默认不加密,true 为无证书最简单的加密，table 为有证书的加密

传入值类型	解释
server_cert	服务器 ca 证书数据
client_cert	客户端 ca 证书数据
client_key	客户端私钥加密数据
client_password	客户端私钥口令数据

返回值

返回值类型	解释
bool/string	成功返回 true 失败返回 string

例子

```
ftp_login = ftp.login(nil,"xxx")
```

---

ftp.command(cmd)

FTP 命令

参数

传入值类型	解释
string	cmd 命令 目前支持:NOOP SYST TYPE PWD MKD CWD CDUP RMD DELE LIST

返回值

返回值类型	解释
string	成功返回 true 失败返回 string

例子

```
print(ftp.command("NOOP").wait())
print(ftp.command("SYST").wait())
print(ftp.command("TYPE I").wait())
print(ftp.command("PWD").wait())
print(ftp.command("MKD QWER").wait())
print(ftp.command("CWD /QWER").wait())
print(ftp.command("CDUP").wait())
print(ftp.command("RMD QWER").wait())
print(ftp.command("DELE /1/12222.txt").wait())
```

---

ftp.pull(local_name,remote_name)

FTP 文件下载

参数

传入值类型		解释
string	local_name	本地文件
string	remote_name	服务器文件

返回值

返回值类型	解释
bool/string	成功返回 true 失败返回 string

例子

```
ftp.pull("/1222.txt","/1222.txt").wait()
```

---

ftp.push(local_name,remote_name)

FTP 文件上传

参数

传入值类型		解释
string	local_name	本地文件
string	remote_name	服务器文件

返回值

返回值类型	解释
bool/string	成功返回 true 失败返回 string

例子

```
ftp.push("/1222.txt","/1222.txt").wait()
```

---

ftp.close()

FTP 客户端关闭

参数

无

返回值

返回值类型	解释
bool/string	成功返回 true 失败返回 string

例子

```
ftp.close().wait()
```

gmssl - 国密算法

已适配 Air101/Air103 Air601 Air105 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 gmssl 的 demo 例子

sm.sm2encrypt(pkx,pky,data, mode, mode2)
sm.sm2decrypt(private,data,mode,mode2)
sm.sm3(data)
sm.sm3hmac(data, key)
gmssl.sm4encrypt(mode,padding,originStr,password)
gmssl.sm4decrypt(mode,padding,encodeStr,password)
sm.sm2sign(private,data,id)
sm.sm2verify(pkx, pky, data, id, sig)

sm.sm2encrypt(pkx,pky,data, mode, mode2)

sm2 算法加密

参数

传入值类型	解释
string	公钥 x,必选. HEX 字符串
string	公钥 y,必选. HEX 字符串
string	待计算的数据,必选,最长 32 字节, 非 HEX 字符串
boolean	输出模式,默认 false. false-GMSSL 默认格式 DER, true-网站兼容模式
boolean	标准版本,默认 false. false-C1C3C2 新国际, true-C1C2C3 老国际

返回值

返回值类型	解释
string	加密后的字符串，原样输出,未经 HEX 转换. 若加密失败会返回 nil 或空字符串

例子

```
-- 提示 mode/mode2 参数是 2023.10.17 新增
-- 由于 SM2 在各平台的实现都有差异,用法务必参考 demo
```

---

sm.sm2decrypt(private,data,mode,mode2)

sm2 算法解密

参数

传入值类型	解释
string	私钥,必选,HEX 字符串
string	待计算的数据,必选,原始数据,非 HEX 字符串
boolean	输出模式,默认 false. false-GMSSL 默认格式 DER, true-网站兼容模式
boolean	标准版本,默认 false. false-C1C3C2 新国际, true-C1C2C3 老国际

返回值

返回值类型	解释
string	解密后的字符串,未经 HEX 转换.若解密失败会返回 nil 或空字符串

例子

```
-- 提示 mode/mode2 参数是 2023.10.17 新增
-- 由于 SM2 在各平台的实现都有差异,用法务必参考 demo
```

---

sm.sm3(data)

sm3 算法,算 HASH 值

参数

传入值类型	解释
string	待计算的数据,必选

返回值

返回值类型	解释
string	对应的 hash 值

例子

```
local encodeStr = gmssl.sm3("lqlq666lqlq946")log.info("testsm.sm3update",string.toHex(encodeStr))
```

---

sm.sm3hmac(data, key)

sm3 算法,算 HASH 值,但带 HMAC

参数

传入值类型	解释
string	待计算的数据,必选
string	密钥

返回值

返回值类型	解释
string	对应的 hash 值

例子

```
local encodeStr = gmssl.sm3hmac("lqlq666lqlq946",
"123")log.info("testsm.sm3update",string.toHex(encodeStr))
```

---

gmssl.sm4encrypt(mode,padding,originStr,password)

SM4 加密算法

参数

传入值类型	解释
string	加密模式, CBC 或 ECB
string	填充方式, NONE/ZERO/PKCS5/PKCS7
string	加密的字符串
string	密钥
string	偏移量

返回值

返回值类型	解释
string	加密后的数据

例子

```
local originStr = "SM4 ECB ZeroPadding test"--加密模式：ECB；填充
方式：ZeroPadding；密钥：1234567890123456；密钥长度：128 bit

local encodeStr =
gmssl.sm4encrypt("ECB","ZERO",originStr,"1234567890123456")

print(originStr,"encrypt",string.toHex(encodeStr))

log.info("testsm.decrypt",gmssl.sm4decrypt("ECB","ZERO",encodeS
tr,"1234567890123456"))

originStr = "SM4 ECB Pkcs5Padding test"--加密模式：ECB；填充方式：
Pkcs5Padding；密钥：1234567890123456；密钥长度：128 bit

encodeStr =
gmssl.sm4encrypt("ECB","PKCS5",originStr,"1234567890123456")
```

```
print(originStr,"encrypt",string.toHex(encodeStr))

log.info("testsm.decrypt",gmssl.sm4decrypt("ECB","PKCS5",encode
Str,"1234567890123456"))

originStr = "SM4 CBC Pkcs5Padding test"--加密模式：CBC；填充方式：
Pkcs5Padding；密钥：1234567890123456；密钥长度：256 bit；偏移量：
1234567890666666

encodeStr =
gmssl.sm4encrypt("CBC","PKCS5",originStr,"1234567890123456","12
34567890666666")

print(originStr,"encrypt",string.toHex(encodeStr))

log.info("testsm.decrypt",gmssl.sm4decrypt("CBC","PKCS5",encode
Str,"1234567890123456","1234567890666666"))
```

---

gmssl.sm4decrypt(mode,padding,encodeStr,password)

SM4 解密算法

参数

传入值类型	解释
string	加密模式, CBC 或 ECB
string	填充方式, NONE/ZERO/PKCS5/PKCS7
string	已加密的字符串
string	密钥
string	偏移量

返回值

返回值类型	解释
string	解密的字符串

例子

```
-- 参考 gmssl.sm4encrypt
```

sm.sm2sign(private,data,id)

sm2 算法签名

参数

传入值类型	解释
string	私钥,必选,HEX 字符串
string	待计算的数据,必选,原始数据,非 HEX 字符串
string	id 值,非 HEX 字符串,可选,默认值”1234567812345678”

返回值

返回值类型	解释
string	前面字符串,未经 HEX 转换.若签名失败会返回 nil

例子

```
-- 本 API 于 2023.10.19 新增
-- 具体用法请查阅 demo
```

sm.sm2verify(pkx, pky, data, id, sig)

sm2 算法验签

参数

传入值类型	解释
string	公钥 X,必选,HEX 字符串
string	公钥 Y,必选,HEX 字符串
string	待计算的数据,必选,原始数据,非 HEX 字符串

传入值类型	解释
string	id 值,非 HEX 字符串,可选,默认值”1234567812345678”
string	签名数据,必须 64 字节,非 HEX 字符串

返回值

返回值类型	解释
boolean	验证成功返回 true,否则返回 nil

例子

```
-- 本 API 于 2023.10.19 新增
-- 具体用法请查阅 demo
```

gpio - GPIO 操作

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 gpio 的 demo 例子

小技巧

本库还有视频教程，点此链接查看

常量

gpio.setup(pin, mode, pull, irq, alt)

gpio.caplevel(pin, level,func)

gpio.set(pin, value)

gpio.get(pin)

gpio.close(pin)

gpio.setDefaultPull(val)

gpio.toggle(pin)

gpio.pulse(pin,level,len,delay)

gpio.debounce(pin, ms, mode)

常量

常量	类型	解释
gpio.NONE	number	无效引脚,一般用于告诉底层某功能引脚不指定
gpio.LOW	number	低电平
gpio.HIGH	number	高电平
gpio.PULLUP	number	上拉
gpio.PULLDOWN	number	下拉
gpio.RISING	number	上升沿触发
gpio.FALLING	number	下降沿触发
gpio.BOTH	number	双向触发,部分设备支持
gpio.HIGH_IRQ	number	高电平触发,部分设备支持
gpio.LOW_IRQ	number	低电平触发,部分设备支持

gpio.setup(pin, mode, pull, irq, alt)

设置管脚功能

参数

传入值

类型	解释
int	pin gpio 编号,必须是数值
	mode 输入输出模式：
any	数字 0/1 代表输出模式
	nil 代表输入模式

传入值

类型	解释
int	function 代表中断模式 pull 上拉下列模式，可以是 gpio.PULLUP 或 gpio.PULLDOWN，或者 0.需要根据实际硬件选用 irq 中断触发模式,默认 gpio.BOTH。中断触发模式 上升沿 gpio.RISING 下降沿 gpio.FALLING 上升和下降都触发 gpio.BOTH
int	alt 复用选项，目前只有 EC618 平台需要这个参数，有些 GPIO 可以复用到不同引脚上，可以选择复用选项（0 或者 4）从而复用到对应的引脚上

返回值

返回值类型	解释
any	输出模式返回设置电平的闭包，输入模式和中断模式返回获取电平的闭包

例子

```
-- 设置 gpio17 为输入 gpio.setup(17, nil)
-- 设置 gpio17 为输出,且初始化电平为低,使用硬件默认上下拉配置
gpio.setup(17, 0)
-- 设置 gpio17 为输出,且初始化电平为高,且启用内部上拉
gpio.setup(17, 1, gpio.PULLUP)
-- 设置 gpio27 为中断，默认双向触发
gpio.setup(27, function(val)
    print("IRQ_27",val)
    -- 提醒，val 并不代表触发方向，仅代表中断后某个时间点的电平
```

```
end, gpio.PULLUP)
-- 设置 gpio27 为中断，仅上升沿触发
gpio.setup(27, function(val)
    print("IRQ_27",val)
    -- 提醒，val 并不代表触发方向，仅代表中断后某个时间点的电平
end, gpio.PULLUP, gpio.RISING)
-- alt_func 于 2023.7.2 新增
-- 本功能仅对部分平台有效，且仅用于调整 GPIO 复用,其他复用方式请使用 muc.iomux 函数
-- 以下示例代码，将 I2S_DOUT 复用成 gpio18
-- AIR780E 的 PIN33(模块管脚序号)，对应 paddr 38，默认功能是 I2S_DOUT，复用成 gpio18
-- 方向输出,且初始化电平为低,使用硬件默认上下拉配置
-- Air780E(EC618 系列的 GPIO 复用请查阅 https://air780e.cn 首页硬件资料表格中的 Air780E&Air780EG&Air780EX&Air700E_GPIO_table_20231227.pdf)
-- Air780EP(EC718P 系列的 GPIO 复用请查阅 https://air780ep.cn 首页硬件资料表格中的 Air780E&Air780EG&Air780EX&Air700E_GPIO_table_20231227.pdf)
gpio.setup(18, 0, nil, nil, 4)
-- 提醒:
-- 当管脚为输入模式或中断,才能通过 gpio.get()获取到电平
-- 当管脚为输出模式,才能通过 gpio.set()设置电平
-- 当管脚为输出模式,通过 gpio.get()总会得到 0
-- 中断回调的 val 参数不代表触发方向，仅代表中断后某个时间点的电平
-- 对 Cat.1 模块,通常只有 AONGPIO 才能双向触发，其他 GPIO 只能单向触发
-- 默认设置下,中断是没有防抖时间的,可以通过 gpio.set_debounce(pin, 50)来设置防抖时间
-- pull 参数的额外说明，上拉/下拉配置
-- 对于部分的 BSP 来说，只支持 gpio.PULLUP 或 gpio.PULLDOWN，但有部分 BSP 支持开漏模式
-- 对于支持开漏的 bsp，pull 参数要传 0 才能开启开漏模式，不是传 nil
```

```
-- 例如:
-- EC618 系列(Air780E/Air780EG/Air780EX/Air700E 等)
-- EC718 系列(Air780EP/Air780EPV 等)
-- XT804 系列(Air101/Air103/Air601)
```

---

gpio.caplevel(pin, level,func)

捕获管脚电平持续时长，单位 us

参数

传入值类型	解释
int	pin GPIO 编号,必须是数值
int	level 需要捕获的电平，可以是 高电平 gpio.HIGH，低电平 gpio.LOW，或者直接写数值 1 或 0，即管脚上正常时间处于 level 的反，捕获设定的 level 持续时间
function	func 完成捕获后的回调函数，仅一个参数，参数为捕获到的时间长度 number 型数值，单位 us

返回值

返回值类型	解释
any	返回获取电平的闭包

例子

```
-- 捕获 pin.PA07 为高电平的持续时间
gpio.caplevel(pin.PA07,1,function(val) print(val) end)
```

---

gpio.set(pin, value)

设置管脚电平

参数

传入值类型	解释
int	pin GPIO 编号,必须是数值
int	value 电平，可以是 高电平 gpio.HIGH，低电平 gpio.LOW，或者直接写数值 1 或 0

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 设置 gpio17 为低电平
gpio.set(17, 0)
```

---

gpio.get(pin)

获取管脚电平

参数

传入值类型	解释
int	pin GPIO 编号,必须是数值

返回值

返回值类型	解释
value	电平，高电平 gpio.HIGH，低电平 gpio.LOW，对应数值 1 和 0

例子

```
-- 获取 gpio17 的当前电平
gpio.get(17)
```

---

gpio.close(pin)

关闭管脚功能(高阻输入态),关掉中断

参数

传入值类型	解释
int	pin GPIO 编号,必须是数值

返回值

返回值类型	解释
nil	无返回值,总是执行成功

例子

```
-- 关闭 gpio17
gpio.close(17)
```

---

gpio.setDefaultPull(val)

设置 GPIO 脚的默认上拉/下拉设置, 默认是平台自定义(一般为开漏).

参数

传入值类型	解释
int	val 0 平台自定义,1 上拉, 2 下拉

返回值

返回值类型	解释

返回值类型	解释
boolean	传值正确返回 true,否则返回 false

例子

```
-- 设置 gpio.setup 的 pull 默认值为上拉
gpio.setDefaultPull(1)
```

---

gpio.toggle(pin)

变换 GPIO 脚输出电平,仅输出模式可用

参数

传入值类型	解释
int	管脚的 GPIO 号

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 本 API 于 2022.05.17 添加-- 假设 GPIO16 上有 LED, 每 500ms 切换一次开关
gpio.setup(16, 0)sys.timerLoopStart(function()
    gpio.toggle(16)
end, 500)
```

---

gpio.pulse(pin,level,len,delay)

在同一个 GPIO 输出一组脉冲, 注意, len 的单位是 bit, 高位在前.

参数

传入值类型	解释
int	gpio 号
int/string	数值或者字符串.
int	len 长度 单位是 bit, 高位在前.
int	delay 延迟,当前无固定时间单位

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 通过 PB06 脚输出输出 8 个电平变化.
gpio.pulse(pin.PB06,0xA9, 8, 0)
```

---

gpio.debounce(pin, ms, mode)

防抖设置, 根据硬件 ticks 进行防抖

参数

传入值类型	解释
int	gpio 号, 0~127, 与硬件相关
int	防抖时长,单位毫秒, 最大 65555 ms, 设置为 0 则关闭
int	模式, 0 冷却模式, 1 延时模式. 默认是 0

返回值

返回值类型	解释

返回值类型　解释

nil　　无返回值

例子

```
-- 消抖模式，当前支持 2 种，2022.12.16 开始支持 mode=1
-- 0 触发中断后,马上上报一次，然后冷却 N 个毫秒后,重新接受中断
-- 1 触发中断后,延迟 N 个毫秒,期间没有新中断且电平没有变化,上报一次
-- 开启防抖，模式 0-冷却，中断后马上上报，但 100ms 内只上报一次
gpio.debounce(7, 100)
-- 若芯片支持 pin 库，可用 pin.PA7 代替数字 7
-- 开启防抖，模式 1-延时，中断后等待 100ms,期间若保持该电平了,时间到之后上报一次
-- 对应的,如果输入的是一个 50hz 的方波,那么不会触发任何上报
gpio.debounce(7, 100, 1)
-- 关闭防抖,时间设置为 0 就关闭
gpio.debounce(7, 0)
```

gtfont - 高通字库芯片

已适配 Air101/Air103 Air601 Air105 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

示例

```
-- 已测试字体芯片型号 GT5SLCD1E-1A
-- 如需要支持其他型号,请报 issue
```

gtfont.init(spi_device)

初始化高通字体芯片

参数

传入值类型	解释
userdata	仅支持 spi device 生成的指针数据

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
-- 特别提醒：使用本库的任何代码，都需要 额外 的 高通字体芯片 !!
-- 没有额外芯片是跑不了的!!
gtfont.init(spi_device)
```

hmeta - 硬件元数据

已适配 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 hmeta 的 demo 例子

示例

```
-- 本库开发中
--[[    这个库的作用是展示当前硬件的能力，例如：
1. 有多少 GPIO，各 GPIO 默认模式是什么，是否支持上拉/下拉
2. 有多少 I2C,支持哪些速率
3. 有多少 SPI,支持哪些速率和模式
4. 扩展属性，例如区分 Air780E 和 Air600E
]]
```

hmeta.model()
hmeta.hwver()

hmeta.model()

获取模组名称

参数

无

返回值

返回值类型	解释
string	若能识别到,返回模组类型，否则会是 nil

例子

```
sys.taskInit(function()
    while 1 do
        sys.wait(3000)
        -- hmeta 识别底层模组类型的
        -- 不同的模组可以使用相同的 bsp,但根据封装的不同,根据内部数据仍可识别出具体模块
        log.info("hmeta", hmeta.model())
        log.info("bsp",   rtos.bsp())
    end
end)
```

---

hmeta.hwver()

获取模组的硬件版本号

参数

无

返回值

返回值类型	解释
string	若能识别到,返回模组类型，否则会是 nil

例子

```
sys.taskInit(function()
    while 1 do
        sys.wait(3000)
        -- hmeta 识别底层模组类型的
        -- 不同的模组可以使用相同的 bsp,但根据封装的不同,根据内部数据仍可识别出具体模块
        log.info("hmeta", hmeta.model(), hmeta.hwver())
        log.info("bsp",   rtos.bsp())
    end
end)
```

http - http 客户端

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 http 的 demo 例子

示例

```
-- 使用 http 库,需要引入 sysplus 库，且需要在 task 内使用
require "sys"
require "sysplus"
sys.taskInit(function()
        sys.wait(1000)
        local code,headers,body = http.request("GET",
"http://www.example.com/abc").wait()
        log.info("http", code, body)
end)
```

http.request(method,url,headers,body,opts,ca_file,client_ca, client_key, client_password)

http 客户端

参数

传入值类型	解释
string	请求方法，支持 GET/POST 等合法的 HTTP 方法
string	url 地址，支持 http 和 https，支持域名，支持自定义端口
tabal	请求头 可选 例如 {[“Content-Type”] = “application/x-www-form-urlencoded”}
string/zbuff	body 可选
table	额外配置 可选 包含 timeout:超时时间单位 ms 可选,默认 10 分钟,写 0 即永久等待 dst:下载路径,可选 adapter:选择使用网卡,可选 debug:是否打开 debug 信息,可选,ipv6:是否为 ipv6 默认不是,可选 callback:下载回调函数,参数 content_len:总长度 body_len:以下载长度 userdata 用户传参,可选 userdata:回调自定义传参
string	服务器 ca 证书数据，可选，一般不需要
string	客户端 ca 证书数据，可选，一般不需要，双向 https 认证才需要
string	客户端私钥加密数据，可选，一般不需要，双向 https 认证才需要

传入值类型	解释
string	客户端私钥口令数据，可选，一般不需要，双向 https 认证才需要

返回值

返回值类型	解释
int	code，服务器反馈的值>=100，最常见的是 200.如果是底层错误,例如连接失败，返回值小于 0
tabal	headers 当 code>100 时，代表服务器返回的头部数据
string/int	body 服务器响应的内容字符串,如果是下载模式，则返回文件大小

例子

```
--[[
code 报错信息列表:
-1 HTTP_ERROR_STATE 错误的状态，一般是底层异常,请报 issue
-2 HTTP_ERROR_HEADER 错误的响应头部，通常是服务器问题
-3 HTTP_ERROR_BODY 错误的响应体,通常是服务器问题
-4 HTTP_ERROR_CONNECT 连接服务器失败，未联网,地址错误,域名错误
-5 HTTP_ERROR_CLOSE 提前断开了连接，网络或服务器问题
-6 HTTP_ERROR_RX 接收数据报错，网络问题
-7 HTTP_ERROR_DOWNLOAD 下载文件过程报错，网络问题或下载路径问题
-8 HTTP_ERROR_TIMEOUT 超时，包括连接超时,读取数据超时
-9 HTTP_ERROR_FOTA fota 功能报错,通常是更新包不合法
]]
-- GET 请求
local code, headers, body =
http.request("GET","http://site0.cn/api/httptest/simple/time").
wait()
log.info("http.get", code, headers, body)-- POST 请求
```

```
local code, headers, body =
http.request("POST","http://httpbin.com/post", {},
"abc=123").wait()

log.info("http.post", code, headers, body)

-- GET 请求,但下载到文件 local code, headers, body =
http.request("GET","http://httpbin.com/", {}, "",
{dst="/data.bin"}).wait()

log.info("http.get", code, headers, body)

-- 自定义超时时间, 5000ms

http.request("GET","http://httpbin.com/", nil, nil,
{timeout=5000}).wait()
```

httpsrv - http 服务端

已适配 Air101/Air103 Air601 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 httpsrv 的 demo 例子

httpsrv.start(port, func)

httpsrv.stop(port)

httpsrv.start(port, func)

启动并监听一个 http 端口

参数

传入值类型	解释
int	端口号
function	回调函数

返回值

返回值类型	解释
bool	成功返回 true，否则返回 false

例子

```
-- 监听 80 端口
httpsrv.start(80, function(client, method, uri, headers, body)
    -- method 是字符串，例如 GET POST PUT DELETE
    -- uri 也是字符串 例如 / /api/abc
    -- headers table 类型
    -- body 字符串
    log.info("httpsrv", method, uri, json.encode(headers), body)
    if uri == "/led/1" then
        LEDA(1)
        return 200, {}, "ok"
    elseif uri == "/led/0" then
        LEDA(0)
        return 200, {}, "ok"
    end
    -- 返回值的约定 code, headers, body
    -- 若没有返回值，则默认 404, {} ,""
    return 404, {}, "Not Found" .. uri
    end)
    -- 关于静态文件
    -- 情况 1: / , 映射为 /index.html
    -- 情况 2: /abc.html , 先查找 /abc.html，不存在的话查找
/abc.html.gz
    -- 若 gz 存在，会自动以压缩文件进行响应，绝大部分浏览器支持.
    -- 当前默认查找 /luadb/xxx 下的文件,暂不可配置
```

httpsrv.stop(port)

停止 http 服务

参数

传入值类型　解释

int　　端口号

返回值

返回值类型　　解释

nil　　当前无返回值

例子

无

i2c - I2C 操作

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 i2c 的 demo 例子

常量

i2c.exist(id)

i2c.setup(id, speed, pullup)

i2c.createSoft(scl,sda,delay)

i2c.send(id, addr, data,stop)

i2c.recv(id, addr, len)

i2c.writeReg(id, addr, reg, data,stop)

i2c.readReg(id, addr, reg, len)

i2c.close(id)

i2c.readDHT12(id)

i2c.readSHT30(id,addr)

i2c.transfer(id, addr, txBuff, rxBuff, rxLen)

i2c.xfer(id, addr, txBuff, rxBuff, rxLen, transfer_done_topic, timeout)

i2c.scan(id,speed)

常量

常量	类型	解释
i2c.FAST	number	高速
i2c.SLOW	number	低速

i2c.exist(id)

i2c 编号是否存在

参数

传入值类型	解释
int	设备 id，例如 i2c1 的 id 为 1, i2c2 的 id 为 2

返回值

返回值类型	解释
bool	存在就返回 true,否则返回 false

例子

```
-- 检查 i2c1 是否存在
if i2c.exist(1) then
    log.info("存在 i2c1")
end
```

---

i2c.setup(id, speed, pullup)

i2c 初始化

参数

传入值类型	解释
int	设备 id，例如 i2c1 的 id 为 1, i2c2 的 id 为 2
int	I2C 速度，例如 i2c.FAST
bool	是否软件上拉，默认不开启，需要硬件支持

返回值

返回值类型	解释
int	成功就返回 1,否则返回 0

例子

```
-- 初始化 i2c1
i2c.setup(1, i2c.FAST) -- id 正确就一定成功
-- 如需判断 i2c id 是否合法，请使用 i2c.exist 函数
```

---

i2c.createSoft(scl,sda,delay)

新建一个软件 i2c 对象

参数

传入值类型	解释
int	i2c SCL 引脚编号(GPIO 编号)
int	i2c SDA 引脚编号(GPIO 编号)

传入值类型	解释
int	每个操作的延时，单位 us，默认 5

返回值

返回值类型	解释
软件 I2C 对象 可当作 i2c 的 id 使用	

例子

```
-- 注意！这个接口是软件模拟 i2c，速度可能会比硬件的慢
-- 不需要调用 i2c.close 接口
-- 初始化软件 i2c
local softI2C = i2c.createSoft(1, 2, 5)
i2c.send(softI2C, 0x5C, string.char(0x0F, 0x2F))-- 注意，第 3 个参数是 2023.06.19 添加的 delay
-- 通过调整 delay 参数的值，可增加或降低 I2C 的速度
```

---

i2c.send(id, addr, data,stop)

i2c 发送数据

参数

传入值类型	解释
int	设备 id，例如 i2c1 的 id 为 1, i2c2 的 id 为 2
int	I2C 子设备的地址, 7 位地址
integer/string/table	待发送的数据,自适应参数类型
integer	可选参数 是否发送停止位 1 发送 0 不发送 默认发送(105 不支持)

返回值

返回值类型	解释
true/false	发送是否成功

例子

```
-- 往 i2c0 发送 1 个字节的数据
i2c.send(0, 0x68, 0x75)-- 往 i2c1 发送 2 个字节的数据
i2c.send(1, 0x5C, string.char(0x0F, 0x2F))
i2c.send(1, 0x5C, {0x0F, 0x2F})
```

---

i2c.recv(id, addr, len)

i2c 接收数据

参数

传入值类型	解释
int	设备 id, 例如 i2c1 的 id 为 1, i2c2 的 id 为 2
int	I2C 子设备的地址, 7 位地址
int	接收数据的长度

返回值

返回值类型	解释
string	收到的数据

例子

```
-- 从 i2c1 读取 2 个字节的数据
local data = i2c.recv(1, 0x5C, 2)
```

---

i2c.writeReg(id, addr, reg, data,stop)

i2c 写寄存器数据

参数

传入值类型	解释
int	设备 id，例如 i2c1 的 id 为 1, i2c2 的 id 为 2
int	I2C 子设备的地址, 7 位地址
int	寄存器地址
string	待发送的数据
integer	可选参数 是否发送停止位 1 发送 0 不发送 默认发送(105 不支持)

返回值

返回值类型	解释
true/false	发送是否成功

例子

```
-- 从 i2c1 的地址为 0x5C 的设备的寄存器 0x01 写入 2 个字节的数据
i2c.writeReg(1, 0x5C, 0x01, string.char(0x00, 0xF2))
```

---

i2c.readReg(id, addr, reg, len)

i2c 读寄存器数据

参数

传入值类型	解释
int	设备 id，例如 i2c1 的 id 为 1, i2c2 的 id 为 2
int	I2C 子设备的地址, 7 位地址
int	寄存器地址

传入值类型	解释
int	待接收的数据长度
integer	可选参数 是否发送停止位 1 发送 0 不发送 默认发送(105 不支持)

返回值

返回值类型	解释
string	收到的数据

例子

```
-- 从 i2c1 的地址为 0x5C 的设备的寄存器 0x01 读出 2 个字节的数据
i2c.readReg(1, 0x5C, 0x01, 2)
```

---

i2c.close(id)

关闭 i2c 设备

参数

传入值类型	解释
int	设备 id，例如 i2c1 的 id 为 1, i2c2 的 id 为 2

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 关闭 i2c1
i2c.close(1)
```

---

i2c.readDHT12(id)

从 i2c 总线读取 DHT12 的温湿度数据

参数

传入值类型	解释
int	设备 id，例如 i2c1 的 id 为 1, i2c2 的 id 为 2
int	DHT12 的设备地址,默认 0x5C

返回值

返回值类型	解释
boolean	读取成功返回 true,否则返回 false
int	湿度值,单位 0.1%，例如 591 代表 59.1%
int	温度值,单位 0.1 摄氏度，例如 292 代表 29.2 摄氏度

例子

```
-- 从 i2c0 读取 DHT12
i2c.setup(0)
local re, H, T = i2c.readDHT12(0)
if re then
    log.info("dht12", H, T)
end
```

---

i2c.readSHT30(id,addr)

从 i2c 总线读取 DHT30 的温湿度数据(由”好奇星”贡献)

参数

传入值类型	解释

传入值类型	解释
int	设备 id，例如 i2c1 的 id 为 1, i2c2 的 id 为 2
int	设备 addr,SHT30 的设备地址,默认 0x44 bit7

返回值

返回值类型	解释
boolean	读取成功返回 true,否则返回 false
int	湿度值,单位 0.1%，例如 591 代表 59.1%
int	温度值,单位 0.1 摄氏度，例如 292 代表 29.2 摄氏度

例子

```
-- 从 i2c0 读取 SHT30
i2c.setup(0)
local re, H, T = i2c.readSHT30(0)
if re then
    log.info("sht30", H, T)
end
```

---

i2c.transfer(id, addr, txBuff, rxBuff, rxLen)

i2c 通用传输，包括发送 N 字节，发送 N 字节+接收 N 字节，接收 N 字节三种功能,在发送转接收过程中发送 reStart 信号,解决类似 mlx90614 必须带 restart 信号，但是又不能用 i2c.send 来控制的，比如 air105

参数

传入值类型	解释
int	设备 id，例如 i2c1 的 id 为 1, i2c2 的 id 为 2
int	I2C 子设备的地址, 7 位地址

传入值类型	解释
integer/string/zbuff	待发送的数据,自适应参数类型，如果为 nil，则不发送数据
zbuff	待接收数据的 zbuff 如果不用 zbuff，则接收数据将在 return 返回
int	需要接收的数据长度，如果为 0 或 nil，则不接收数据

返回值

返回值类型		解释
boolean	true/false	发送是否成功
string	or nil	如果参数 5 是 interger，则返回接收到的数据

例子

```
local result, _ = i2c.transfer(0, 0x11, txbuff, rxbuff, 1)

local result, _ = i2c.transfer(0, 0x11, txbuff, nil, 0) --只发送 txbuff 里的数据，不接收数据，典型应用就是写寄存器了，这里寄存器地址和值都放在了 txbuff 里

local result, _ = i2c.transfer(0, 0x11, "\x01\x02\x03", nil, 1) --发送 0x01， 0x02，0x03，不接收数据，如果是 eeprom，就是往 0x01 的地址写 02 和 03，或者往 0x0102 的地址写 03，看具体芯片了

local result, rxdata = i2c.transfer(0, 0x11, "\x01\x02", nil, 1) --发送 0x01， 0x02，然后接收 1 个字节，典型应用就是 eeprom

local result, rxdata = i2c.transfer(0, 0x11, 0x00, nil, 1) --发送 0x00，然后接收 1 个字节，典型应用各种传感器
```

---

i2c.xfer(id, addr, txBuff, rxBuff, rxLen, transfer_done_topic, timeout)

i2c 非阻塞通用传输，类似 transfer，但是不会等到 I2C 传输完成才返回，调用本函数会立刻返回，I2C 传输完成后，通过消息回调

参数

传入值

类型	解释
int	设备 id，例如 i2c1 的 id 为 1, i2c2 的 id 为 2
int	I2C 子设备的地址, 7 位地址
zbuff	待发送的数据，由于用的非阻塞模型，为保证动态数据的有效性，只能使用 zbuff，发送的数据从 zbuff.addr 开始，长度为 zbuff.used
zbuff	待接收数据的 zbuff，如果为 nil，则忽略后面参数， 不接收数据。接收的数据会放在 zbuff.addr 开始的位置，会覆盖掉之前的数据，注意 zhuff 的预留空间要足够
int	需要接收的数据长度，如果为 0 或 nil，则不接收数据
string	传输完成后回调的消息
int	超时时间，如果填 nil，则为 100ms

返回值

返回值类型	解释
boolean	true/false 本次传输是否正确启动，true，启动，false，有错误无法启动。传输完成会发布消息 transfer_done_topic 和 boolean 型结果

例子

```
local result = i2c.xfer(0, 0x11, txbuff, rxbuff, 1, "I2CDONE")
if result then
        result, i2c_id, succ, error_code = sys.waitUntil("I2CDONE")
end
if not result or not succ then
        log.info("i2c fail, error code", error_code)
else
        log.info("i2c ok")
end
```

i2c.scan(id,speed)

扫描 i2c 设备

参数

传入值类型	解释
int	设备 id，例如 i2c1 的 id 为 1, i2c2 的 id 为 2
int	速度，可选 i2c.SLOW i2c.FAST i2c.PLUS i2c.HSMODE 默认为 i2c.SLOW,如探测不到则修改此项

返回值

返回值类型	解释
nil	当前无返回值

例子

```
-- 本函数于 2023.07.04 添加
-- 这个函数的主要目标是为了在开发期扫描 i2c 设备
-- 有些 BSP 在指定 addr 无响应时会输出日志,导致输出会被打乱
i2c.scan()
```

i2s - 数字音频

已适配 Air101/Air103 Air601 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 i2s 的 demo 例子

示例

```
-- 这个库属于底层适配库，具体用法请查阅示例
-- demo/multimedia
-- demo/tts
-- demo/record
```

常量

i2s.setup(id, mode, sample, bitw, channel, format, framebit)

i2s.send(id, data, len)

i2s.recv(id, buffer, len)

i2s.close(id)

i2s.on(id, func)

i2s.txStat(id)

i2s.getPara(id)

常量

常量	类型	解释
i2s.MODE_I2S	number	I2S 标准，比如 ES7149
i2s.MODE_LSB	number	LSB 格式
i2s.MODE_MSB	number	MSB 格式，比如 TM8211
i2s.MONO_L	number	左声道
i2s.MONO_R	number	右声道
i2s.STEREO	number	立体声

i2s.setup(id, mode, sample, bitw, channel, format, framebit)

初始化 i2s

参数

传入值类型	解释
int	i2s 通道号,与具体设备有关
int	模式, 0 主机 1 从机
int	采样率,默认 44100. 可选
int	数据位数,默认 16, 可以是 8 的倍数
int	声道, 0 左声道, 1 右声道, 2 立体声. 可选
int	格式, 可选 MODE_I2S, MODE_LSB, MODE_MSB
int	1 个声道的 BCLK 数量, 可选 16 和 32

返回值

返回值类型	解释
boolean	成功与否
int	底层返回值

例子

```
-- 以默认参数初始化 i2s
i2s.setup(0)-- 以详细参数初始化 i2s，示例为默认值
i2s.setup(0, 0, 44100, 16, 0, 0, 16)
```

---

i2s.send(id, data, len)

发送 i2s 数据

参数

传入值类型	解释
int	通道 id
string	数据, 可以是字符串或 zbuff

传入值类型	解释
int	数据长度,单位字节，字符串默认为字符串全长, zbuff 默认为指针位置

返回值

返回值类型	解释
boolean	成功与否
int	底层返回值,供调试用

例子

```
local f = io.open("/luadb/abc.wav")
while 1 do
    local data = f:read(4096)
    if not data or #data == 0 then
        break
    end
    i2s.send(0, data)
    sys.wait(100)
end
```

---

i2s.recv(id, buffer, len)

接收 i2s 数据，注意在数据在回调时已经存放在 zbuff 里，目前只有 air780e 系列支持

参数

传入值类型	解释
int	通道 id

传入值类型	解释
zbuff	数据缓存区
int	单次返回的数据长度,单位字节,必须与传入的 zbuff 的大小一致

返回值

返回值类型	解释
boolean	成功与否

例子

```
local buffer = zbuff.create(3200)
local succ = i2s.recv(0, buffer, 3200);
```

---

i2s.close(id)

关闭 i2s

参数

传入值类型	解释
int	通道 id

返回值

返回值类型	解释
nil	无返回值

例子

```
i2s.close(0)
```

---

i2s.on(id, func)

注册 I2S 事件回调

参数

传入值类型	解释
int	i2s id, i2s0 写 0, i2s1 写 1
function	回调方法

返回值

返回值类型	解释
nil	无返回值

例子

```
i2s.on(0, function(id, buff)
        if buff then
                log.info("i2s get data in zbuff")
        else
                log.info("i2s tx one block done")
        end
end)
```

---

i2s.txStat(id)

获取 i2s 的发送缓冲区状态

参数

传入值类型	解释
int	i2s id, i2s0 写 0, i2s1 写 1

传入值类型	解释
return	底层缓冲区的总大小
return	底层缓冲区的剩余待发送数据

返回值

无

例子

```
-- 读取发送缓冲区的状态，从而判断是否需要继续传入音频数据
local max, remain = i2s.txStat(0)
log.info("i2s 发送缓冲区状态", max, remain)
```

---

i2s.getPara(id)

获取 I2S 参数，参数具体参考 setup

参数

传入值类型	解释
int	通道 id

返回值

返回值类型	解释
boolean	是否在工作状态 true 是
int	模式, 0 主机 1 从机
int	采样率
int	数据位数
int	声道
int	格式

返回值类型	解释
int	1 个声道的 BCLK 数量

例子

无

iconv - iconv 操作

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

小技巧

本库有专属 demo，点此链接查看 iconv 的 demo 例子

```
iconv.open(tocode, fromcode)
iconv:iconv(inbuf)
iconv.open(tocode, fromcode)
cd:iconv(inbuf)
iconv.close(cd)
```

iconv.open(tocode, fromcode)

打开相应字符编码转换函数

参数

传入值类型	解释
string	释义：目标编码格式 取值：gb2312/ucs2/ucs2be/utf8
string	释义：源编码格式 取值：gb2312/ucs2/ucs2be/utf8

返回值

返回值类型	解释
userdata	编码转换函数的转换句柄,若不存在会返回 nil

例子

```
--unicode 大端编码 转化为 utf8 编码
local iconv = iconv.open("utf8", "ucs2be")
```

---

iconv:iconv(inbuf)

字符编码转换

参数

传入值类型	解释
string	释义：待转换字符串

返回值

返回值类型	解释
number	释义：返回编码转换后的结果 取值：0 成功,-1 失败

例子

```
--unicode 大端编码 转化为 utf8 编码
function ucs2beToUtf8(ucs2s)
    local iconv = iconv.open("utf8", "ucs2be")
    return iconv:iconv(ucs2s)
end
```

---

iconv.open(tocode, fromcode)

打开相应字符编码转换函数

参数

传入值类型	解释
string	tocode$目标编码格式$gb2312/ucs2/ucs2be/utf8
string	fromcode$源编码格式$gb2312/ucs2/ucs2be/utf8
return	tablecd编码转换函数的转换句柄$

返回值

无

例子

```
--unicode 大端编码 转化为 utf8 编码
local cd = iconv.open("utf8", "ucs2be")
```

---

cd:iconv(inbuf)

字符编码转换

参数

传入值类型	解释
string	inbuf$输入字符串$例如:ucs2s
return	number$result$返回编码转换后的结果$0 成功,-1 失败

返回值

无

例子

```
--unicode 大端编码 转化为 utf8 编码
function ucs2beToUtf8(ucs2s)
    local cd = iconv.open("utf8", "ucs2be")
    return cd:iconv(ucs2s)
end
```

---

iconv.close(cd)

关闭字符编码转换

参数

传入值类型	解释
string	cd$iconv.open 返回的句柄$
return	

返回值

无

例子

```
--关闭字符编码转换
local cd = iconv.open("utf8", "ucs2be")
iconv.close(cd)
```

io - io 操作(扩展)

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 io 的 demo 例子

io.exists(path)

io.fileSize(path)

io.readFile(path, mode, offset, len)

io.writeFile(path, data)

io.fill(buff, offset, len)

io.mkfs(path)

io.mkdir(path)

io.rmdir(path)

io.lsdir(path, len, offset)

io.lsmount()

示例

```
-- io 模块是 lua 原生模块,LuatOS 增加了一些 API
-- 请配合 os 模块一起使用
-- 只读模式, 打开文件
local fd = io.open("/xxx.txt", "rb")-- 读写默认,打开文件
local fd = io.open("/xxx.txt", "wb")-- 写入文件,且截断为 0 字节
local fd = io.open("/xxx.txt", "wb+")-- 追加模式
local fd = io.open("/xxx.txt", "a")
-- 若文件打开成功, fd 不为 nil,否则就是失败了
-- 注意, 刷机时所添加的文件, 均在 /luadb 目录下, 只读
if fd then
  -- 读取指定字节数,如果数据不足,就只返回实际长度的数据
  local data = fd:read(12)
  -- 按行读取
  local line = fd:read("*l")
  -- 全部读取
```

```
  local line = fd:read("*a")


  -- 数据写入，仅 w 或 a 模式可调用
  -- 数据需要是字符串，lua 的字符串是带长度的,可以包含任何二进制数据
  fd:write("xxxx")
  -- 以下是写入 0x12，0x13
  fd:write(string.char(0x12, 0x13))


--[[file:seek ([whence [, offset]])
设置及获取基于文件开头处计算出的位置。 设置的位置
由 offset 和 whence 字符串 whence 指定的基点决定。基点可以是：
“set”：基点为 0 （文件开头）；
“cur”：基点为当前位置；
“end”：基点为文件尾；
当 seek 成功时，返回最终从文件开头计算起的文件的位置。 当 seek 失败时，
返回 nil 加上一个错误描述字符串。
whence 的默认值是 "cur"， offset 默认为 0 。 因此，调
用 file:seek() 可以返回文件当前位置，并不改变它； 调
用 file:seek("set") 将位置设为文件开头（并返回 0）； 调
用 file:seek("end") 将位置设到文件末尾，并返回文件大小。]]
  -- 基点为 0 （文件开头）,将位置设为文件开头（并返回 0）
  fd:seek("set")
  -- 基点为当前位置,返回文件当前位置
  fd:seek("cur")
  -- 基点为文件尾,将位置设到文件末尾，并返回文件大小
  fd:seek("end")


  -- 执行完操作后,一定要关掉文件
  fd:close()
end
```

io.exists(path)

判断文件是否存在

参数

传入值类型	解释
string	文件路径

返回值

返回值类型	解释
bool	存在返回 true,否则返回 false

例子

```
log.info("io", "file exists", io.exists("/boottime"))
```

---

io.fileSize(path)

获取文件大小

参数

传入值类型	解释
string	文件路径

返回值

返回值类型	解释
int	文件数据,若文件不存在会返回 nil

例子

```
local fsize = io.fileSize("/bootime")
if fsize and fsize > 1024 then
```

```
  log.info("io", "file size", fsize)
end
```

---

io.readFile(path, mode, offset, len)

读取整个文件,请注意内存消耗

参数

传入值类型	解释
string	文件路径
string	读取模式, 默认 “rb”
int	起始位置,默认 0
int	读取长度,默认整个文件

返回值

返回值类型	解释
string	文件数据,若文件不存在会返回 nil

例子

```
local data = io.readFile("/bootime")-- 注意: offset 和 len 参数是 2023.6.6 添加的
-- 读取 abc.txt, 先跳过 128 字节, 然后读取 512 字节数据
local data = io.readFile("/abc.txt", "rb", 128, 512)
```

---

io.writeFile(path, data)

将数据写入文件

参数

传入值类型	解释
string	文件路径
string	数据

返回值

返回值类型	解释
boolean	成功返回 true，否则返回 false

例子

```
io.writeFile("/bootime", "1")
```

---

io.fill(buff, offset, len)

读取文件并填充到 zbuff 内,但不移动指针位置

参数

传入值类型	解释
userdata	zbuff 实体
int	写入的位置,默认是 0
int	写入的长度,默认是 zbuff 的 len 减去 offset

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 false
int	返回实际读取到的长度，如果小于 0 也说明是读取失败了

例子

```
local buff = zbuff.create(1024)
```

```
local f = io.open("/sd/test.txt")
if f then
  f:fill(buff)
end
```

---

io.mkfs(path)

格式化文件系统,需指定挂载点

参数

传入值类型	解释
string	挂载点

返回值

返回值类型	解释
bool	成功与否
int	底层返回值

例子

```
local ret, errio = io.mkfs("/sd")
log.info("fs", "mkfs", ret, errio)
```

---

io.mkdir(path)

创建文件夹

参数

传入值类型	解释
string	需要建立的目录路径

返回值

返回值类型	解释
bool	成功与否
int	底层返回值

例子

```
local ret, errio = io.mkdir("/data/")
log.info("fs", "mkdir", ret, errio)
```

---

io.rmdir(path)

删除文件夹

参数

传入值类型	解释
string	需要移除的目录路径

返回值

返回值类型	解释
bool	成功与否
int	底层返回值

例子

```
local ret, errio = io.rmdir("/data/")
log.info("fs", "rmdir", ret, errio)
```

---

io.lsdir(path, len, offset)

列出目录下的文件

参数

传入值类型	解释
string	需要枚举的目录路径
int	最大长度，默认 10，最高 50
int	偏移量，默认 0，当目录文件很多时分页查询用

返回值

返回值类型	解释
bool	成功与否
int	底层返回值

例子

```
local ret, data = io.lsdir("/data/", 10, 0)
if ret then
  log.info("fs", "lsdir", json.encode(data))
else
  log.info("fs", "lsdir", "fail", ret, data)
end
```

---

io.lsmount()

列出所有挂载点

参数

无

返回值

返回值类型	解释
table	挂载点列表

例子

```
local data = io.lsmount()
log.info("fs", "lsmount", json.encode(data))
```

ioqueue - io 序列操作

已适配 Air105 Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 ioqueue 的 demo 例子

ioqueue.init(hwtimer_id,cmd_cnt,repeat_cnt)

ioqueue.setdelay(hwtimer_id,time_us,time_tick,continue)

ioqueue.delay(hwtimer_id)

ioqueue.setgpio(hwtimer_id,pin,is_input,pull_mode,init_level)

ioqueue.input(hwtimer_id,pin)

ioqueue.output(hwtimer_id,pin,level)

ioqueue.setcap(hwtimer_id,pin,pull_mode,irq_mode,max_tick)

ioqueue.capture(hwtimer_id)

ioqueue.capend(hwtimer_id,pin)

ioqueue.get(hwtimer_id, input_buff, capture_buff)

ioqueue.start(hwtimer_id)

ioqueue.stop(hwtimer_id)

ioqueue.release(hwtimer_id)

ioqueue.clear(hwtimer_id)

ioqueue.done(hwtimer_id)

ioqueue.exti(pin,pull_mode,irq_mode,onoff)

ioqueue.init(hwtimer_id,cmd_cnt,repeat_cnt)

初始化一个 io 操作队列

参数

传入值类型	解释
int	硬件定时器 id，默认用 0，根据实际 MCU 确定，air105 为 0~5，与 pwm 共用，同一个通道号不能同时为 pwm 和 ioqueue
int	一个完整周期需要的命令，可以比实际的多
int	重复次数,默认是 1，如果写 0 则表示无限次数循环
return	无

返回值

无

例子

```
ioqueue.init(0,10,5) --以 timer0 为时钟源初始化一个 io 操作队列，有 10 个有效命令，循环 5 次
```

---

ioqueue.setdelay(hwtimer_id,time_us,time_tick,continue)

对 io 操作队列增加延时命令

参数

传入值类型	解释
int	硬件定时器 id

传入值类型	解释
int	延时时间,0~65535us
int	延时微调时间,0~255tick,总的延时时间是 time_us * 1us_tick + time_tick
boolean	是否连续是连续延时，默认否，如果是，定时器在时间到后不会停止而是重新计时，

返回值

无

例子

无

---

ioqueue.delay(hwtimer_id)

对 io 操作队列增加一次重复延时，在前面必须有 setdelay 且是连续延时

参数

传入值类型	解释
int	硬件定时器 id
return	无

返回值

无

例子

```
ioqueue.setdelay(0,9,15,true) --延时 9us+15 个 tick,在之后遇到 delay 命令时，会延时 9us+15 个 tick
```

```
ioqueue.delay(0)
```

---

ioqueue.setgpio(hwtimer_id,pin,is_input,pull_mode,init_level)

对 io 操作队列增加设置 gpio 命令

参数

传入值类型	解释
int	硬件定时器 id
int	pin
boolean	是否是输入
int	上下拉模式,只能是 0,gpio.PULLUP,gpio.PULLDOWN
int	初始输出电平
return	无

返回值

无

例子

```
ioqueue.setgpio(0,pin.PB01,true,gpio.PULLUP,0) --PB01 设置成上拉输入

ioqueue.setgpio(0,pin.PB01,false,0,1)--PB01 设置成默认上下拉输出高电平
```

---

ioqueue.input(hwtimer_id,pin)

对 io 操作队列增加读取 gpio 命令

参数

传入值类型	解释
int	硬件定时器 id
int	pin
return	无

返回值

无

例子

```
ioqueue.input(0,pin.PB01)
```

---

ioqueue.output(hwtimer_id,pin,level)

对 io 操作队列增加输出 GPIO 命令

参数

传入值类型	解释
int	硬件定时器 id
int	pin
int	输出电平
return	无

返回值

无

例子

```
ioqueue.output(0,pin.PB01,0)
```

---

ioqueue.setcap(hwtimer_id,pin,pull_mode,irq_mode,max_tick)

对 io 操作队列增加设置捕获某个 IO 命令

参数

传入值类型	解释
int	硬件定时器 id
int	pin
int	上下拉模式,只能是 0,gpio.PULLUP,gpio.PULLDOWN
int	中断模式,只能是 gpio.BOTH,gpio.RISING,gpio.FALLING
int	定时器最大计时时间 考虑到 lua 是 int 类型，最小 0x10000，最大值为 0x7fffffff，默认为最大值
return	无

返回值

无

例子

```
ioqueue.setcap(0,pin.PB01,gpio.PULLUP,gpio.FALLING,48000000)
```

---

ioqueue.capture(hwtimer_id)

对 io 操作队列增加捕获一次 IO 状态命令

参数

传入值类型	解释
int	硬件定时器 id
return	无

返回值

无

例子

```
ioqueue.capture(0)
```

---

ioqueue.capend(hwtimer_id,pin)

对 io 操作队列增加结束捕获某个 IO 命令

参数

传入值类型	解释
int	硬件定时器 id
int	pin
return	无

返回值

无

例子

```
ioqueue.capend(0,pin.PB01)
```

---

ioqueue.get(hwtimer_id, input_buff, capture_buff)

-

获取 io 操作队列中输入和捕获的数据

-

参数

传入值类型	解释
int	硬件定时器 id
zbuff	存放 IO 输入数据的 buff，按照 1byte pin + 1byte level 形式存放数据
zbuff	存放 IO 捕获数据的 buff，按照 1byte pin + 1byte level + 4byte tick 形式存放数据

返回值

返回值类型	解释
int	返回多少组 IO 输入数据
int	返回多少组 IO 捕获数据

例子

```
local input_cnt, capture_cnt = ioqueue.get(0, input_buff, capture_buff)
```

---

ioqueue.start(hwtimer_id)

启动 io 操作队列

参数

传入值类型	解释
int	硬件定时器 id
return	无

返回值

无

例子

```
ioqueue.start(0)
```

---

ioqueue.stop(hwtimer_id)

停止 io 操作队列，可以通过 start 从头开始

参数

传入值类型	解释
int	硬件定时器 id

返回值

返回值类型	解释
int	返回已经循环的次数，如果是 0，表示一次循环都没有完成
int	返回单次循环中已经执行的 cmd 次数，如果是 0，可能是一次循环刚刚结束

例子

```
ioqueue.stop(0)
```

---

ioqueue.release(hwtimer_id)

释放 io 操作队列的资源，下次使用必须重新 init

参数

传入值类型	解释
int	硬件定时器 id

传入值类型	解释
return	无

返回值

无

例子

```
ioqueue.clear(0)
```

---

ioqueue.clear(hwtimer_id)

清空 io 操作队列

参数

传入值类型	解释
int	硬件定时器 id
return	无

返回值

无

例子

```
ioqueue.clear(0)
```

---

ioqueue.done(hwtimer_id)

检测 io 操作队列是否已经执行完成

参数

传入值类型	解释
int	硬件定时器 id

返回值

返回值类型	解释
boolean	队列是否执行完成，

例子

```
local result = ioqueue.done(0)
```

---

ioqueue.exti(pin,pull_mode,irq_mode,onoff)

启动/停止一个带系统 tick 返回的外部中断

参数

传入值类型	解释
int	pin
int	上下拉模式,只能是 0,gpio.PULLUP,gpio.PULLDOWN
int	中断模式,只能是 gpio.BOTH,gpio.RISING,gpio.FALLING
boolean	开关，默认是 false 关
return	无

返回值

无

例子

```
ioqueue.exti(pin.PB01, gpio.PULLUP, gpio.BOTH,
true)ioqueue.exti(pin.PB01)
```

iotauth - IoT 鉴权库，用于生成各种云平台的参数

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

小技巧

本库有专属 demo，点此链接查看 iotauth 的 demo 例子

iotauth.aliyun(product_key, device_name,device_secret,method,cur_timestamp)
iotauth.onenet(produt_id, device_name,key,method,cur_timestamp,version)
iotauth.iotda(device_id,device_secret,cur_timestamp)
iotauth.qcloud(product_id, device_name,device_secret,method,cur_timestamp,sdk_appid)
iotauth.tuya(device_id,device_secret,cur_timestamp)
iotauth.baidu(iot_core_id, device_key,device_secret,method,cur_timestamp)

iotauth.aliyun(product_key, device_name,device_secret,method,cur_timestamp)

阿里云物联网平台三元组生成

参数

传入值类型	解释
string	product_key
string	device_name
string	device_secret
string	method 加密方式,”hmacmd5” “hmacsha1” “hmacsha256” 可选,默认”hmacmd5”
number	cur_timestamp 可选 默认为 32472115200(2999-01-01 0:0:0)
bool	istls 是否 TLS 直连 true:TLS 直连 false:TCP 直连模式 默认 TCP 直连模式

返回值

返回值类型	解释
string	mqtt 三元组 client_id
string	mqtt 三元组 user_name
string	mqtt 三元组 password

例子

```
local client_id,user_name,password = iotauth.aliyun("123456789","abcdefg","Y877Bgo8X5owd3lcB5wWDjryNPoB")
print(client_id,user_name,password)
```

---

iotauth.onenet(produt_id, device_name,key,method,cur_timestamp,version)

中国移动物联网平台三元组生成

参数

传入值类型	解释
string	produt_id 产品 id
string	device_name 设备名称
string	key 设备密钥或者项目的 acess_key
string	method 加密方式,”md5” “sha1” “sha256” 可选,默认”md5”
number	时间戳, 不用填
string	version 可选 默认”2018-10-31”
string	当 key 是 access_key 时, 填 “products/” .. product_id . 本参数于 2024.1.29 新增

返回值

返回值类型	解释
string	mqtt 三元组 client_id
string	mqtt 三元组 user_name
string	mqtt 三元组 password

例子

```
-- OneNet 平台官网: https://open.iot.10086.cn/

-- OneNet 有多种版本，注意区分，一般来说 produt_id 纯数字就是老版本，否则就是新版本

-- 新版 OneNET 平台，产品 id 是英文字母字符串

-- 对应 demo/onenet/studio

local produt_id = "Ck2AF9QD2K"

local device_name = "test"

local device_key = "KuF3NT/jUBJ62LNBB/A8XZA9CqS3Cu79B/ABmfA1UCw="

local client_id,user_name,password = iotauth.onenet(produt_id, device_name, device_key)

log.info("onenet.new", client_id,user_name,password)

-- 旧版 OneNET 平台，产品 id 是数字字符串. 2024.1.29 新增

-- 对应 demo/onenet/old_mqtt

local produt_id = "12342334"

local device_name = "test"

local access_key = "adfasdfadsfadsf="

local client_id,user_name,password = iotauth.onenet(produt_id, device_name, access_key, nil, nil, nil, "products/" .. produt_id)

log.info("onenet.old", client_id,user_name,password)
```

---

iotauth.iotda(device_id,device_secret,cur_timestamp)

华为物联网平台三元组生成

参数

传入值类型	解释
string	device_id
string	device_secret
number	cur_timestamp 可选 如不填则不校验时间戳

返回值

返回值类型	解释
string	mqtt 三元组 client_id
string	mqtt 三元组 user_name
string	mqtt 三元组 password

例子

```
local client_id,user_name,password = iotauth.iotda("6203cc94c7fb24029b110408_88888888","123456789")print(client_id,user_name,password)
```

---

iotauth.qcloud(product_id, device_name,device_secret,method,cur_timestamp,sdk_appid)

腾讯联网平台三元组生成

参数

传入值类型	解释
string	产品 id,创建项目后可以查看到,类似于 LD8S5J1L07
string	设备名称,例如设备的 imei 号
string	设备密钥,创建设备后,查看设备详情可得到
string	method 加密方式,”sha1” “sha256” 可选,默认”sha256”
number	cur_timestamp 可选 默认为 32472115200(2999-01-01 0:0:0)
string	sdk_appid 可选 默认为”12010126”

返回值

返回值类型	解释
string	mqtt 三元组 client_id
string	mqtt 三元组 user_name
string	mqtt 三元组 password

例子

```
local client_id,user_name,password = iotauth.qcloud("LD8S5J1L07","test","acyv3QDJrRa0fW5UE58KnQ==")
print(client_id,user_name,password)
```

---

iotauth.tuya(device_id,device_secret,cur_timestamp)

涂鸦联网平台三元组生成

参数

传入值类型	解释
string	device_id
string	device_secret
number	cur_timestamp 可选 默认 7258089600(2200-01-01 0:0:0)

返回值

返回值类型	解释
string	mqtt 三元组 client_id
string	mqtt 三元组 user_name
string	mqtt 三元组 password

例子

```
local client_id,user_name,password =
iotauth.tuya("6c95875d0f5ba69607nzfl","fb803786602df760")

print(client_id,user_name,password)
```

---

iotauth.baidu(iot_core_id, device_key,device_secret,method,cur_timestamp)

百度物联网平台三元组生成

参数

传入值类型	解释
string	iot_core_id
string	device_key
string	device_secret
string	method 加密方式,”MD5” “SHA256” 可选,默认”MD5”
number	cur_timestamp 可选 如不填则不校验时间戳

返回值

返回值类型	解释
string	mqtt 三元组 client_id
string	mqtt 三元组 user_name
string	mqtt 三元组 password

例子

```
local client_id,user_name,password =
iotauth.baidu("abcd123","mydevice","ImSeCrEt0I1M2jkl")

print(client_id,user_name,password)
```

ir - 红外遥控

已适配 Air101/Air103 Air601 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 ir 的 demo 例子

ir.sendNEC(pin, addr, cmd, repeat, disablePWM)

发送 NEC 数据

参数

传入值类型	解释
int	使用的 GPIO 引脚编号
int	用户码（大于 0xff 则采用 Extended NEC 模式）
int	数据码
int	可选，引导码发送次数（110ms 一次），默认 0 次
bool	可选，是否禁止直接发送 pwm 波，默认 false

返回值

无

例子

```
--直接发 ir.sendNEC(0, 0x11, 0x22)
--外接了 38K 的 PWM 载波，只控制电平
ir.sendNEC(0, 0x11, 0x22,0,true)
```

json - json 生成和解析库

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 json 的 demo 例子

json.encode(obj,t)

json.decode(str)

json.encode(obj,t)

将对象序列化为 json 字符串

参数

传入值类型	解释
obj	需要序列化的对象
string	浮点数精度和模式,这项不存在的时候,为默认值”7f”,数字只支持”0~14”,模式只支持”f/g”

返回值

返回值类型	解释
string	序列化后的 json 字符串，失败的话返回 nil
string	序列化失败的报错信息

例子

```
json.encode(obj)-->浮点数用%.7g 的方式转换为字符串
json.encode(obj,"12f")-->浮点数用%.12f 的方式转换为字符串
-- 2024.1.9 起，默认使用 7f 格式化浮点数
-- 之前的版本使用 7g 格式化浮动数
```

json.decode(str)

将字符串反序列化为对象

参数

传入值类型	解释
string	需要反序列化的 json 字符串

返回值

返回值类型	解释
obj	反序列化后的对象(通常是 table)，失败的话返回 nil
result	成功返回 1,否则返回 0
err	反序列化失败的报错信息

例子

```
json.decode("[1,2,3,4,5,6]")
```

keyboard - 键盘矩阵

已适配 Air105

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 keyboard 的 demo 例子

keyboard.init(port, conf, map, debounce)

初始化键盘矩阵

参数

传入值

类型	解释
int	预留，当前填 0
int	启用的 keyboard 管脚掩码，例如使用 keyboard0~9，则掩码为 0x1FF，若使用 0~3 则 0xF
int	keyboard 管脚方向映射，其中输入为 0,输出为 1，按位设置．例如 keyboard0~3 作为输入, keyboard4~7 为输入，则 0xF0
int	消抖配置,预留,可以不填

返回值

无

例子

```
-- 做一个 4*4 键盘矩阵，使用 keyboard0~7，其中 0~3 做输入，4~7 做输出
-- 使用 keyboard0~7，对应 conf 为 0xFF
-- 其中 0~3 做输入，4~7 做输出，对应 map 为 0xF0
keyboard.init(0, 0xFF, 0xF0)
-- 做一个 2*3 键盘矩阵，使用 keyboard0~4，其中 0~1 做输入，2~4 做输出
-- 使用 keyboard0~4，二进制为 11111， 对应 conf 的十六进制表达为 0x1F
-- 其中 0~1 做输入，2~4 做输出，二进制为 11100 对应 map 为 0x14
-- keyboard.init(0, 0xFF, 0x14)
sys.subscribe("KB_INC", function(port, data, state)
    -- port 当前固定为 0，可以无视
    -- data，需要配合 init 的 map 进行解析
    -- state，1 为按下，0 为 释放
    -- TODO 详细介绍
end)
```

lcd - lcd 驱动模块

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 lcd 的 demo 例子

常量

lcd.init(tp, args, spi_dev, init_in_service)

lcd.close()

lcd.on()

lcd.off()

lcd.sleep()

lcd.wakeup()

lcd.invon()

lcd.invoff()

lcd.cmd(cmd)

lcd.data(data)

lcd.setColor(back,fore)

lcd.draw(x1, y1, x2, y2,color)

lcd.clear(color)

lcd.fill(x1, y1, x2, y2,color)

lcd.drawPoint(x0,y0,color)

lcd.drawLine(x0,y0,x1,y1,color)

lcd.drawRectangle(x0,y0,x1,y1,color)

lcd.drawCircle(x0,y0,r,color)

lcd.drawQrcode(x, y, str, size)

lcd.setFont(font, indentation)

lcd.drawStr(x,y,str,fg_color)

lcd.drawGtfontGb2312(str,size,x,y)

lcd.drawGtfontGb2312Gray(str,size,gray,x,y)

lcd.drawGtfontUtf8(str,size,x,y)

lcd.drawGtfontUtf8Gray(str,size,gray,x,y)

lcd.getSize()

lcd.drawXbm(x, y, w, h, data)

lcd.showImage(x, y, file)

lcd.flush()

lcd.setupBuff(conf, onheap)

lcd.autoFlush(enable)

lcd.rgb565(r, g, b, swap)

常量

常量	类型	解释
lcd.font_opposansm12	font	12 号字体
lcd.font_unifont_t_symbols	font	符号字体
lcd.font_open_iconic_weather_6x_t	font	天气字体
lcd.font_opposansm16	font	16 号字体
lcd.font_opposansm18	font	18 号字体
lcd.font_opposansm20	font	20 号字体
lcd.font_opposansm22	font	22 号字体
lcd.font_opposansm24	font	24 号字体
lcd.font_opposansm32	font	32 号字体
lcd.font_opposansm12_chinese	font	12 号中文字体
lcd.font_opposansm14_chinese	font	14 号中文字体
lcd.font_opposansm16_chinese	font	16 号中文字体
lcd.font_opposansm18_chinese	font	18 号中文字体
lcd.font_opposansm20_chinese	font	20 号中文字体
lcd.font_opposansm22_chinese	font	22 号中文字体
lcd.font_opposansm24_chinese	font	24 号中文字体
lcd.font_opposansm32_chinese	font	32 号中文字体
lcd.HWID_0	硬件 lcd 驱动 id0	(根据芯片支持选择)

常量	类型	解释
lcd.WIRE_3_BIT_9_INTERFACE_I	三线 spi	9bit 模式 I
lcd.WIRE_4_BIT_8_INTERFACE_I	四线 spi	8bit 模式 I
lcd.WIRE_3_BIT_9_INTERFACE_II	三线 spi	9bit 模式 II
lcd.WIRE_4_BIT_8_INTERFACE_II	四线 spi	8bit 模式 II

lcd.init(tp, args, spi_dev, init_in_service)

lcd 显示屏初始化

参数

传入值类型	解释
string	lcd 类型，当前支持： st7796 st7789 st7735 st7735v st7735s gc9a01 gc9106l gc9306x ili9486 custom
table	附加参数,与具体设备有关：

传入值类型	解释
	pin_pwr（背光）为可选项,可不设置
	port：spi 端口,例如 0,1,2…如果为 device 方式则为”device”
	pin_dc：lcd 数据/命令选择引脚
	pin_rst：lcd 复位引脚
	pin_pwr：lcd 背光引脚 可选项,可不设置
	direction：lcd 屏幕方向 0:0° 1:180° 2:270° 3:90°
	w：lcd 水平分辨率
	h：lcd 竖直分辨率
	xoffset：x 偏移(不同屏幕 ic 不同屏幕方向会有差异)
	yoffset：y 偏移(不同屏幕 ic 不同屏幕方向会有差异)
	direction0：0°方向命令，(不同屏幕 ic 会有差异)
	direction90：90°方向命令，(不同屏幕 ic 会有差异)
	direction180：180°方向命令，(不同屏幕 ic 会有差异)
	direction270：270°方向命令，(不同屏幕 ic 会有差异)
	sleepcmd：睡眠命令，默认 0X10
	wakecmd：唤醒命令，默认 0X11
	interface_mode lcd 模式，默认 lcd.WIRE_4_BIT_8_INTERFACE_I
userdata	spi 设备,当 port = “device”时有效
boolean	允许初始化在 lcd service 里运行，默认是 false

返回值

无

例子

```
-- 初始化 spi0 的 st7735s 注意:lcd 初始化之前需要先初始化 spi
spi_lcd = spi.deviceSetup(0,20,0,0,8,2000000,spi.MSB,1,1)
log.info("lcd.init",lcd.init("st7735s",{port = "device",pin_dc = 17, pin_pwr = 7,pin_rst = 19,direction = 2,w = 160,h = 80,xoffset = 1,yoffset = 26},spi_lcd))
```

---

lcd.close()

关闭 lcd 显示屏

参数

无

返回值

无

例子

```
-- 关闭 lcd
lcd.close()
```

---

lcd.on()

开启 lcd 显示屏背光

参数

无

返回值

无

例子

```
-- 开启 lcd 显示屏背光
lcd.on()
```

---

lcd.off()

关闭 lcd 显示屏背光

参数

无

返回值

无

例子

```
-- 关闭 lcd 显示屏背光
lcd.off()
```

---

lcd.sleep()

lcd 睡眠

参数

无

返回值

无

例子

```
-- lcd 睡眠
lcd.sleep()
```

---

lcd.wakeup()

lcd 唤醒

参数

无

返回值

无

例子

```
-- lcd 唤醒
lcd.wakeup()
```

---

lcd.invon()

lcd 反显

参数

无

返回值

无

例子

```
-- lcd 反显
lcd.invon()
```

---

lcd.invoff()

lcd 反显关闭

参数

无

返回值

无

例子

```
-- lcd 反显关闭
lcd.invoff()
```

---

lcd.cmd(cmd)

lcd 命令

参数

传入值类型 解释

int	cmd

返回值

无

例子

```
-- lcd 命令
lcd.cmd(0x21)
```

---

lcd.data(data)

lcd 数据

参数

传入值类型	解释
int	data

返回值

无

例子

```
-- lcd 数据
lcd.data(0x21)
```

---

lcd.setColor(back,fore)

lcd 颜色设置

参数

传入值类型	解释
int	背景色
int	前景色

返回值

无

例子

```
-- lcd 颜色设置
lcd.setColor(0xFFFF,0x0000)
```

---

lcd.draw(x1, y1, x2, y2,color)

lcd 颜色填充

参数

传入值类型	解释
int	左上边缘的 X 位置.
int	左上边缘的 Y 位置.
int	右下边缘的 X 位置.
int	右下边缘的 Y 位置.
string	字符串或 zbuff 对象

返回值

无

例子

```
-- lcd 颜色填充
local buff = zbuff.create({201,1,16},0x001F)
lcd.draw(20,30,220,30,buff)
```

---

lcd.clear(color)

lcd 清屏

参数

传入值类型	解释
int	屏幕颜色 可选参数,默认背景色

返回值

无

例子

```
-- lcd 清屏
lcd.clear()
```

---

lcd.fill(x1, y1, x2, y2,color)

lcd 颜色填充

参数

传入值类型	解释
int	左上边缘的 X 位置.
int	左上边缘的 Y 位置.
int	右下边缘的 X 位置,不含
int	右下边缘的 Y 位置,不含
int	绘画颜色 可选参数,默认背景色

返回值

无

例子

```
-- lcd 颜色填充
```

```
lcd.fill(20,30,220,30,0x0000)
```

---

lcd.drawPoint(x0,y0,color)

画一个点.

参数

传入值类型	解释
int	点的 X 位置.
int	点的 Y 位置.
int	绘画颜色 可选参数,默认前景色

返回值

无

例子

```
lcd.drawPoint(20,30,0x001F)
```

---

lcd.drawLine(x0,y0,x1,y1,color)

在两点之间画一条线.

参数

传入值类型	解释
int	第一个点的 X 位置.
int	第一个点的 Y 位置.
int	第二个点的 X 位置.
int	第二个点的 Y 位置.

传入值类型	解释
int	绘画颜色 可选参数,默认前景色

返回值

无

例子

```
lcd.drawLine(20,30,220,30,0x001F)
```

---

lcd.drawRectangle(x0,y0,x1,y1,color)

从 x / y 位置（左上边缘）开始绘制一个框

参数

传入值类型	解释
int	左上边缘的 X 位置.
int	左上边缘的 Y 位置.
int	右下边缘的 X 位置.
int	右下边缘的 Y 位置.
int	绘画颜色 可选参数,默认前景色

返回值

无

例子

```
lcd.drawRectangle(20,40,220,80,0x001F)
```

---

lcd.drawCircle(x0,y0,r,color)

从 x / y 位置（圆心）开始绘制一个圆

参数

传入值类型	解释
int	圆心的 X 位置.
int	圆心的 Y 位置.
int	半径.
int	绘画颜色 可选参数,默认前景色

返回值

无

例子

```
lcd.drawCircle(120,120,20,0x001F)
```

---

lcd.drawQrcode(x, y, str, size)

缓冲区绘制 QRCode

参数

传入值类型	解释
int	x 坐标
int	y 坐标
string	二维码的内容
int	显示大小 (注意:二维码生成大小与要显示内容和纠错等级有关,生成版本为 1-40(对应 21x21 - 177x177)的不定大小,如果和设置大小不同会自

传入值类型	解释
	动在指定的区域中间显示二维码,如二维码未显示请查看日志提示)

返回值

返回值类型	解释
nil	无返回值

例子

无

---

lcd.setFont(font, indentation)

设置字体

参数

传入值类型		解释
int	font	lcd.font_XXX 请查阅常量表
int	indentation,	等宽字体 ascii 右侧缩进 0~127 个 pixel，等宽字体的 ascii 字符可能在右侧有大片空白，用户可以选择删除部分。留空或者超过 127 则直接删除右半边，非等宽字体无效

返回值

无

例子

```
-- 设置为字体,对之后的 drawStr 有效,调用 lcd.drawStr 前一定要先设置
```

```
-- 若提示 "only font pointer is allow" ，则代表当前固件不含对应字体，可使用云编译服务免费定制-- 云编译文档: https://wiki.luatos.com/develop/compile/Cloud_compilation.html

-- lcd 库的默认字体均以 lcd.font_ 开头
lcd.setFont(lcd.font_opposansm12)lcd.drawStr(40,10,"drawStr")

sys.wait(2000)

lcd.setFont(lcd.font_opposansm12_chinese) -- 具体取值可参考 api 文档的常量表

lcd.drawStr(40,40,"drawStr 测试")
```

---

lcd.drawStr(x,y,str,fg_color)

显示字符串

参数

传入值类型

	解释
int	x 横坐标
int	y 竖坐标 注意:此(x,y)为左下起始坐标
string	str 文件内容
int	fg_color str 颜色 注意:此参数可选，如不填写则使用之前设置的颜色，绘制只会绘制字体部分，背景需要自己清除

返回值

无

例子

```
-- 显示之前先设置为中文字体,对之后的 drawStr 有效
lcd.setFont(lcd.font_opposansm12)

lcd.drawStr(40,10,"drawStr")

sys.wait(2000)
```

```
lcd.setFont(lcd.font_opposansm16_chinese)
lcd.drawStr(40,40,"drawStr 测试")
```

---

lcd.drawGtfontGb2312(str,size,x,y)

使用 gtfont 显示 gb2312 字符串

参数

传入值类型	解释
string	str 显示字符串
int	size 字体大小 (支持 16-192 号大小字体)
int	x 横坐标
int	y 竖坐标

返回值

无

例子

```
-- 注意，gtfont 是额外的字体芯片硬件，需要外挂在 SPI 总线才能调用本函数的
lcd.drawGtfontGb2312("啊啊啊",32,0,0)
```

---

lcd.drawGtfontGb2312Gray(str,size,gray,x,y)

使用 gtfont 灰度显示 gb2312 字符串

参数

传入值类型	解释
string	str 显示字符串

传入值类型	解释
int	size 字体大小 (支持 16-192 号大小字体)
int	gray 灰度[1 阶/2 阶/3 阶/4 阶]
int	x 横坐标
int	y 竖坐标

返回值

无

例子

```
-- 注意，gtfont 是额外的字体芯片硬件，需要外挂在 SPI 总线才能调用本函数的
lcd.drawGtfontGb2312Gray("啊啊啊",32,4,0,40)
```

---

lcd.drawGtfontUtf8(str,size,x,y)

使用 gtfont 显示 UTF8 字符串

参数

传入值类型	解释
string	str 显示字符串
int	size 字体大小 (支持 16-192 号大小字体)
int	x 横坐标
int	y 竖坐标

返回值

无

例子

```
lcd.drawGtfontUtf8("啊啊啊",32,0,0)
```

---

lcd.drawGtfontUtf8Gray(str,size,gray,x,y)

使用 gtfont 灰度显示 UTF8 字符串

参数

传入值类型	解释
string	str 显示字符串
int	size 字体大小 (支持 16-192 号大小字体)
int	gray 灰度[1 阶/2 阶/3 阶/4 阶]
int	x 横坐标
int	y 竖坐标

返回值

无

例子

```
lcd.drawGtfontUtf8Gray("啊啊啊",32,4,0,40)
```

---

lcd.getSize()

获取屏幕尺寸

参数

无

返回值

返回值类型	解释

返回值类型	解释
int	宽，如果未初始化会返回 0
int	高，如果未初始化会返回 0

例子

```
log.info("lcd", "size", lcd.getSize())
```

---

lcd.drawXbm(x, y, w, h, data)

绘制位图

参数

传入值类型	解释
int	X 坐标
int	y 坐标
int	位图宽
int	位图高
int	位图数据,每一位代表一个像素

返回值

无

例子

```
-- 取模使用 PCtoLCD2002 软件即可 阴码 逐行 逆向
-- 在(0,0)为左上角,绘制 16x16 "今" 的位图
lcd.drawXbm(0, 0, 16,16, string.char(

0x80,0x00,0x80,0x00,0x40,0x01,0x20,0x02,0x10,0x04,0x48,0x08,0x8
4,0x10,0x83,0x60,
```

```
0x00,0x00,0xF8,0x0F,0x00,0x08,0x00,0x04,0x00,0x04,0x00,0x02,0x0
0,0x01,0x80,0x00))
```

---

lcd.showImage(x, y, file)

显示图片,当前只支持 jpg,jpeg

参数

传入值类型	解释
int	x 坐标
int	y 坐标
string	文件路径

返回值

无

例子

```
lcd.showImage(0,0,"/luadb/logo.jpg")
```

---

lcd.flush()

主动刷新数据到界面，仅设置 buff 且禁用自动属性后使用

参数

无

返回值

返回值类型	解释
bool	成功返回 true，否则返回 nil/false

例子

无

---

lcd.setupBuff(conf, onheap)

设置显示缓冲区，所需内存大小为 2×宽×高 字节. 请衡量内存需求与业务所需的刷新频次.

参数

传入值类型	解释
userdata	conf 指针，不需要传
bool	true 使用 heap 内存, false 使用 vm 内存，默认使用 vm 内存，不需要主动传

返回值

返回值类型	解释
bool	是否成功

例子

```
-- 初始化 lcd 的 buff 缓冲区，可理解为 FrameBuffer 区域.
lcd.setupBuff()
```

---

lcd.autoFlush(enable)

设置自动刷新，需配合 lcd.setupBuff 使用

参数

传入值类型	解释
bool	是否自动刷新,默认为 true

返回值

无

例子

```
-- 设置 buff 并禁用自动更新
lcd.setupBuff()
lcd.autoFlush(false)-- 禁止自动更新后，需要使用 lcd.flush() 主动刷新数据到屏幕
```

---

lcd.rgb565(r, g, b, swap)

RGB565 颜色生成

参数

传入值类型	解释
int	红色, 0x00 ~ 0xFF
int	绿色, 0x00 ~ 0xFF
int	蓝色, 0x00 ~ 0xFF
bool	是否翻转, true 翻转, false 不翻转. 默认翻转

返回值

返回值类型	解释
int	颜色值

例子

```
-- 本 API 支持多种模式，参数数量分别是 1，2，3，4-- 1. 单参数形式，24bit RGB 值，swap = true，推荐
local red =   lcd.rgb565(0xFF0000)
local green = lcd.rgb565(0x00FF00)
local blue =  lcd.rgb565(0x0000FF)
-- 2. 两参数形式，24bit RGB 值，增加 swap 的设置
local red =   lcd.rgb565(0xFF0000, true)
local green = lcd.rgb565(0x00FF00, true)
local blue =  lcd.rgb565(0x0000FF, true)
-- 3. 三参数形式，红/绿/蓝，各 8bit
local red = lcd.rgb565(0xFF, 0x00, 0x00)
local green = lcd.rgb565(0x00, 0xFF, 0x00)
local blue = lcd.rgb565(0x00, 0x00, 0xFF)
-- 4. 四参数形式，红/绿/蓝，各 8bit，增加 swap 的设置
local red = lcd.rgb565(0xFF, 0x00, 0x00, true)
local green = lcd.rgb565(0x00, 0xFF, 0x00, true)
local blue = lcd.rgb565(0x00, 0x00, 0xFF, true)
```

libgnss - NMEA 数据处理

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 libgnss 的 demo 例子

示例

```
-- 提醒：本库输出的坐标,均为 WGS84 坐标系
-- 如需要在国内地图使用，要转换成对应地图的坐标系，例如 GCJ02 BD09
```

```
-- 相关链接: https://lbsyun.baidu.com/index.php?title=coordinate
-- 相关链接: https://www.openluat.com/GPS-Offset.html
-- 提醒: GPS 功能, GNSS 功能, NMEA 解析功能,均为当前库的子功能
-- 本库的主要功能就是解析 NMEA 协议, 支持内置 GNSS 也支持外置 GNSS
-- 以下是使用本 libgnss 的示例代码
-- 方案 1, 经 lua 层进行数据中转
uart.setup(2, 115200)
uart.on(2, "recv", function(id, len)
    while 1 do
        local data = uart.read(id, 1024)
        if data and #data > 1 then
            libgnss.parse(data)
        else
            break
        end
    end
end)
-- 方案 2, 适合 2022.12.26 之后编译固件,效率更高一些
uart.setup(2, 115200)libgnss.bind(2)
-- 可选调试模式
-- libgnss.debug(true)
sys.subscribe("GNSS_STATE", function(event, ticks)
    -- event 取值有
    -- FIXED 定位成功
    -- LOSE  定位丢失
    -- ticks 是事件发生的时间,一般可以忽略
    log.info("gnss", "state", event, ticks)
end)
```

libgnss.parse(str)
libgnss.isFix()
libgnss.getIntLocation(speed_type)
libgnss.getRmc(data_mode)
libgnss.getGsv()
libgnss.getGsa(data_mode)
libgnss.getVtg(data_mode)
libgnss.getZda()
libgnss.debug(mode)
libgnss.getGga(data_mode)
libgnss.getGll(data_mode)
libgnss.clear()
libgnss.bind(id, next_id)
libgnss.locStr(mode)
libgnss.rtcAuto(enable)
libgnss.on(tp, fn)
libgnss.getTxt()
libgnss.casic_aid(dt, loc)

libgnss.parse(str)

处理 nmea 数据

参数

传入值类型	解释
string	原始 nmea 数据

返回值

无

例子

```
-- 解析 nmea 数据
libgnss.parse(indata)
```

```
log.info("nmea", json.encode(libgnss.getRmc(), "11g"))
```

---

libgnss.isFix()

当前是否已经定位成功

参数

无

返回值

返回值类型	解释
boolean	定位成功与否

例子

```
log.info("nmea", "isFix", libgnss.isFix())
```

---

libgnss.getIntLocation(speed_type)

获取位置信息

参数

传入值类型	解释
int	速度单位,默认是 m/h

返回值

返回值类型	解释
int	lat 数据，格式为 ddddddddd
int	lng 数据，格式为 ddddddddd
int	speed 数据，单位米. 于 2023.9.26 修正

例子

```
-- 建议用 libgnss.getRmc(1)
log.info("nmea", "loc", libgnss.getIntLocation())
-- 2023.12.11 新增 speed_type 参数
--[[
速度单位可选值 0 - m/h 米/小时，默认值，整型 1 - m/s 米/秒，浮点数 2 - km/h 千米/小时，浮点数 3 - kn/h 英里/小时，浮点数
]]
-- 默认 米/小时
log.info("nmea", "loc", libgnss.getIntLocation())-- 米/秒
log.info("nmea", "loc", libgnss.getIntLocation(1))-- 千米/小时
log.info("nmea", "loc", libgnss.getIntLocation(2))-- 英里/小时
log.info("nmea", "loc", libgnss.getIntLocation(3))
```

---

libgnss.getRmc(data_mode)

获取原始 RMC 位置信息

参数

传入值类型	解释
int	坐标类数据的格式, 0-DDMM.MMM 格式, 1-DDDDDDD 格式, 2-DD.DDDDD 格式

返回值

返回值类型	解释
table	原始 rmc 数据

例子

```
-- 解析 nmea
```

```
log.info("nmea", "rmc", json.encode(libgnss.getRmc(2)))-- 实例输出
--[[
{
    "course":0,
    "valid":true,   // true 定位成功,false 定位丢失
    "lat":23.4067,  // 纬度，正数为北纬，负数为南纬
    "lng":113.231,  // 经度，正数为东经，负数为西经
    "variation":0,  // 地面航向，单位为度，从北向起顺时针
    "speed":0       // 地面速度，单位为"节"
    "year":2023,    // 年份
    "month":1,      // 月份，1-12
    "day":5,        // 月份天，1-31
    "hour":7,       // 小时,0-23
    "min":23,       // 分钟,0-59
    "sec":20,       // 秒,0-59
}
]]
```

---

libgnss.getGsv()

获取原始 GSV 信息

参数

无

返回值

返回值类型	解释
table	原始 GSV 数据

例子

```
-- 解析 nmea
log.info("nmea", "gsv", json.encode(libgnss.getGsv()))
--[[实例输出
{
"total_sats":24,      // 总可见卫星数量
 "sats":[
        {
                "snr":27,     // 信噪比
                "azimuth":278, // 方向角
                "elevation":59, // 仰角
                "tp":0,        // 0 - GPS/SASS/QSZZ, 1 - BD
                "nr":4         // 卫星编号
        },
        // 这里忽略了 22 个卫星的信息
        {
                "snr":0,
                "azimuth":107,
                "elevation":19,
        "tp":1,
        "nr":31
        }
        ]
}
]]
```

---

libgnss.getGsa(data_mode)

获取原始 GSA 信息

参数

传入值类型	解释
int	坐标类数据的格式, 0-DDMM.MMM 格式, 1-DDDDDDD 格式, 2-DD.DDDDD 格式

返回值

返回值类型	解释
table	原始 GSA 数据

例子

```
-- 获取
log.info("nmea", "gsa", json.encode(libgnss.getGsa(), "11g"))
-- 示例数据
--[[
{
    "sats":[ // 正在使用的卫星编号
        9,
        6,
        16,
        16,
        26,
        21,
        27,
        27,
        4,
        36,
        3,
        7,
```

```
        8,
        194
],
"vdop":0.03083333, // 垂直精度因子，0.00 - 99.99，不定位时值为 99.99
"pdop":0.0455,     // 水平精度因子，0.00 - 99.99，不定位时值为 99.99
"fix_type":3,      // 定位模式，1-未定位，2-2D 定位，3-3D 定位
"hdop":0.0335      // 位置精度因子，0.00 - 99.99，不定位时值为 99.99
}
]]
```

---

libgnss.getVtg(data_mode)

获取 VTA 速度信息

参数

传入值类型	解释
int	可选, 3-原始字符串，不传或者传其他值，则返回浮点值

返回值

返回值类型	解释
table	原始 VTA 数据

例子

```
-- 解析 nmea
log.info("nmea", "vtg", json.encode(libgnss.getVtg()))
-- 示例
--[[
        {
```

```
        "speed_knots":0,        // 速度，英里/小时
        "true_track_degrees":0,  // 真北方向角
        "magnetic_track_degrees":0, // 磁北方向角

        "speed_kph":0           // 速度，千米/小时

        }
]]
```

---

libgnss.getZda()

获取原始 ZDA 时间和日期信息

参数

无

返回值

返回值类型	解释
table	原始 zda 数据

例子

```
log.info("nmea", "zda", json.encode(libgnss.getZda()))
-- 实例输出
--[[
{
"minute_offset":0,   // 本地时区的分钟，一般固定输出 0
"hour_offset":0,     // 本地时区的小时，一般固定输出 0
"year":2023         // UTC 年，四位数字

"month":1,          // UTC 月，两位，01 ~ 12

 "day":5,            // UTC 日，两位数字，01 ~ 31

"hour":7,           // 小时

"min":50,           // 分
```

```
"sec":14,           // 秒
}
]]
```

---

libgnss.debug(mode)

设置调试模式

参数

传入值类型	解释
bool	true 开启调试,false 关闭调试,默认为 false

返回值

无

例子

```
-- 开启调试，会输出 GNSS 原始数据到日志中
libgnss.debug(true)-- 关闭调试
libgnss.debug(false)
```

---

libgnss.getGga(data_mode)

获取 GGA 数据

参数

传入值类型	解释
int	坐标类数据的格式, 0-DDMM.MMM 格式, 1-DDDDDDD 格式, 2-DD.DDDDD 格式, 3-原始字符串

返回值

返回值类型	解释
table	GGA 数据，若如不存在会返回 nil

例子

```
local gga = libgnss.getGga(2)if gga then
    log.info("GGA", json.encode(gga, "11g"))
End
--实例输出
--[[
{
"dgps_age":0,              // 差分校正时延，单位为秒
"fix_quality":1,  // 定位状态标识 0 - 无效,1 - 单点定位,2 - 差分定位
"satellites_tracked":14,  // 参与定位的卫星数量
"altitude":0.255,          // 海平面分离度，或者成为海拔，单位是米，
"hdop":0.0335,    // 水平精度因子，0.00 - 99.99，不定位时值为 99.99
"longitude":113.231,       // 经度，正数为东经，负数为西经
"latitude":23.4067,        // 纬度，正数为北纬，负数为南纬
"height":0                // 椭球高，固定输出 1 位小数
}
]]
```

---

libgnss.getGll(data_mode)

获取 GLL 数据

参数

传入值类型	解释
int	坐标类数据的格式, 0-DDMM.MMM 格式, 1-DDDDDDD 格式,

传入值类型	解释
	2-DD.DDDDD 格式

返回值

返回值类型	解释
table	GLL 数据，若如不存在会返回 nil

例子

```
local gll = libgnss.getGll(2)
if gll then
    log.info("GLL", json.encode(gll, "11g"))
End
-- 实例数据
--[[
{
"status":"A",          // 定位状态，A 有效，B 无效
"mode":"A",            // 定位模式，V 无效，A 单点解，D 差分解
"sec":20,              // 秒，UTC 时间为准
"min":23,              // 分钟，UTC 时间为准
"hour":7,              // 小时，UTC 时间为准
"longitude":113.231, // 经度，正数为东经，负数为西经
"latitude":23.4067,  // 纬度，正数为北纬，负数为南纬
"us":0                 // 微妙数，通常为 0
}
]]
```

---

libgnss.clear()

清除历史定位数据

参数

无

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 该操作会清除所有定位数据
```

---

libgnss.bind(id, next_id)

绑定 uart 端口进行 GNSS 数据读取

参数

传入值类型	解释
int	uart 端口号
int	转发到 uart 的 id，例如虚拟 uart.VUART_0

返回值

无

例子

```
-- 配置串口信息，通常为 115200 8N1
uart.setup(2, 115200)-- 绑定 uart，马上开始解析 GNSS 数据
libgnss.bind(2)-- 无需再调用 uart.on 然后调用 libgnss.parse
-- 开发期可打开调试日志
libgnss.debug(true)
-- 2023-01-02 之后编译的固件有效
```

```
-- 从 uart2 读取并解析，同时转发到虚拟串口 0
libgnss.bind(2, uart.VUART_0)
```

---

libgnss.locStr(mode)

获取位置字符串

参数

传入值类型	解释
int	字符串模式. 0- Air780EG 所需的格式
return	指定模式的字符串

返回值

无

例子

```
-- 仅推荐在定位成功后调用
```

---

libgnss.rtcAuto(enable)

定位成功后自动设置 RTC

参数

传入值类型	解释
bool	开启与否，默认是 false 关闭

返回值

无

例子

```
-- 开启自动设置 RTC
libgnss.rtcAuto(true)
```

---

libgnss.on(tp, fn)

底层事件回调

参数

传入值类型	解释
string	事件类型,当前支持”raw”

返回值

无

例子

```
-- 本函数一般用于调试，用于获取底层实际收到的数据
libgnss.on("raw", function(data)
    log.info("GNSS", data)
end)
```

---

libgnss.getTxt()

获取非标的 GPTXT 数据

参数

传入值类型	解释
return	GPTXT 所携带的字符串

返回值

无

例子

```
-- 本函数于 2023.6.6 添加
log.info("gnss", "txt", libgnss.getTxt())
-- 测试语句
libgnss.parse("$GPTXT,01,01,01,ANTENNA SHORT*63\r\n")
log.info("GNSS", libgnss.getTxt())
libgnss.parse("$GPTXT,01,01,01,ANTENNA OPEN*25\r\n")
log.info("GNSS", libgnss.getTxt())
libgnss.parse("$GPTXT,01,01,01,ANTENNA OK*35\r\n")
log.info("GNSS", libgnss.getTxt())
```

---

libgnss.casic_aid(dt, loc)

合成 Air530Z 所需要的辅助定位数据

参数

传入值类型	解释
table	时间信息
table	经纬度及海拔

返回值

返回值类型	解释
string	辅助定位数据

例子

```
-- 本函数适合 CASIC 系列 GNSS 模块的辅助定位信息的合成
-- 本函数 2023.11.14 新增
```

```
-- 首先是时间信息,注意是 UTC 时间
-- 时间来源很多，一般建议 socket.sntp()时间同步后的系统时间
local dt = os.date("!*t")
-- 然后是辅助定位坐标
-- 来源有很多方式:
-- 1. 从历史定位数据得到，例如之前定位成功后保存到本地文件系统了
-- 2. 通过基站定位或者 wifi 定位获取到
-- 3. 通过 IP 定位获取到大概坐标
-- 坐标系是 WGS84，但鉴于是辅助定位,精度不是关键因素
local lla = {
    lat = 23.12,
    lng = 114.12
    }
local aid = libgnss.casic_aid(dt, lla)
```

log - 日志库

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

常量

log.setLevel(level)

log.style(val)

log.getLevel()

log.debug(tag, val, val2, val3, ...)

log.info(tag, val, val2, val3, ...)

log.warn(tag, val, val2, val3, ...)

log.error(tag, val, val2, val3, ...)

常量

常量	类型	解释
log.LOG_SILENT	number	无日志模式
log.LOG_DEBUG	number	debug 日志模式
log.LOG_INFO	number	info 日志模式
log.LOG_WARN	number	warning 日志模式
log.LOG_ERROR	number	error 日志模式

log.setLevel(level)

设置日志级别

参数

传入值类型	解释
string	level 日志级别,可用字符串或数值，字符串为(SILENT,DEBUG,INFO,WARN,ERROR,FATAL)，数值为(0,1,2,3,4,5)

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 设置日志级别为 INFO
log.setLevel("INFO")
```

---

log.style(val)

设置日志风格

参数

传入值类型	解释
int	日志风格,默认为 0，不传就是获取当前值

返回值

返回值类型	解释
int	当前的日志风格

例子

```
-- 以 log.info("ABC", "DEF", 123) 为例，假设该代码位于 main.lua 的 12 行
-- 默认日志 0
-- I/user.ABC DEF 123
-- 调试风格 1，添加额外的调试信息
-- I/main.lua:12 ABC DEF 123
-- 调试风格 2，添加额外的调试信息，位置有所区别
-- I/user.ABC main.lua:12 DEF 123
log.style(0) -- 默认风格 0
log.style(1) -- 调试风格 1
log.style(2) -- 调试风格 2
```

---

log.getLevel()

获取日志级别

参数

无

返回值

返回值类型	解释
int	日志级别对应 0,1,2,3,4,5

例子

```
-- 得到日志级别 log.getLevel()
```

---

log.debug(tag, val, val2, val3, …)

输出日志,级别 debug

参数

传入值类型	解释
string	tag 日志标识,必须是字符串
…	需打印的参数

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 日志输出 D/onenet connect ok
log.debug("onenet", "connect ok")
```

---

log.info(tag, val, val2, val3, …)

输出日志,级别 info

参数

传入值类型	解释

传入值类型		解释
string	tag	日志标识,必须是字符串
…		需打印的参数

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 日志输出 I/onenet connect ok
log.info("onenet", "connect ok")
```

---

log.warn(tag, val, val2, val3, …)

输出日志,级别 warn

参数

传入值类型		解释
string	tag	日志标识,必须是字符串
…		需打印的参数

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 日志输出 W/onenet connect ok
log.warn("onenet", "connect ok")
```

log.error(tag, val, val2, val3, …)

输出日志,级别 error

参数

传入值类型		解释
string	tag	日志标识,必须是字符串
…		需打印的参数

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 日志输出 E/onenet connect ok
log.error("onenet", "connect ok")
```

lora - lora 驱动模块

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 lora 的 demo 例子

常量

lora.init(ic, loraconfig,spiconfig)

lora.set_channel(freq)

lora.set_txconfig(ic, txconfig)

lora.set_rxconfig(ic, set_rxconfig)

lora.send(data)

lora.recv(timeout)

lora.mode(mode)

常量

常量	类型	解释
lora.SLEEP	number	SLEEP 模式
lora.STANDBY	number	STANDBY 模式

lora.init(ic, loraconfig,spiconfig)

lora 初始化

参数

传入值类型	解释
string	lora 型号，当前支持： llcc68 sx1268
table	lora 配置参数,与具体设备有关

返回值

无

例子

```
lora.init("llcc68",
    {
        id = 0,             -- SPI id
        cs = pin.PB04,      -- SPI 片选的 GPIO 号,如果没有 pin 库,填 GPIO 数字编号就行
```

```
        res = pin.PB00,    -- 复位脚连接的 GPIO 号,如果没有 pin 库,填 GPIO 数字编号就行
        busy = pin.PB01,   -- 忙检测脚的 GPIO 号
        dio1 = pin.PB06,   -- 数据输入中断脚
        lora_init = true   -- 是否发送初始化命令. 如果是唤醒后直接读取,就传 false
    }
    )
```

---

lora.set_channel(freq)

设置频道频率

参数

传入值类型 解释

number 频率

返回值

无

例子

```
lora.set_channel(433000000)
```

---

lora.set_txconfig(ic, txconfig)

lora 配置发送参数

参数

传入值类型 解释

传入值类型	解释
string	lora 型号，当前支持： llcc68 sx1268
table	lora 发送配置参数,与具体设备有关

返回值

无

例子

```
lora.set_txconfig("llcc68",
    {
        mode=1,
        power=22,
        fdev=0,
        bandwidth=0,
        datarate=9,
        coderate=4,
        preambleLen=8,
        fixLen=false,
        crcOn=true,
        freqHopOn=0,
        hopPeriod=0,
        iqInverted=false,
        timeout=3000
    })
```

lora.set_rxconfig(ic, set_rxconfig)

lora 配置接收参数

参数

传入值类型	解释
string	lora 型号，当前支持： llcc68 sx1268
table	lora 接收配置参数,与具体设备有关

返回值

无

例子

```
lora.set_rxconfig("llcc68",
    {
        mode=1,
        bandwidth=0,
        datarate=9,
        coderate=4,
        bandwidthAfc=0,
        preambleLen=8,
        symbTimeout=0,
        fixLen=false,
        payloadLen=0,
        crcOn=true,
        freqHopOn=0,
        hopPeriod=0,
```

```
    iqInverted=false,
    rxContinuous=false
})
```

---

lora.send(data)

发数据

参数

传入值类型	解释
string	写入的数据

返回值

无

例子

```
lora.send("PING")
```

---

lora.recv(timeout)

开启收数据

参数

传入值类型	解释
number	超时时间，默认 1000 单位 ms

返回值

无

例子

```
sys.subscribe("LORA_RX_DONE", function(data, size)
    log.info("LORA_RX_DONE: ", data, size)
    lora.send("PING")
    end)-- 老版本没有 recv，可以改成 lora.recive
    lora.recv(1000)
```

---

lora.mode(mode)

设置进入模式(休眠，正常等)

参数

传入值类型	解释
number	模式 正常模式:lora.STANDBY 休眠模式:lora.SLEEP 默认为正常模式

返回值

无

例子

```
lora.mode(lora.STANDBY)
```

lora2 - lora2 驱动模块(支持多挂)

已适配 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 lora2 的 demo 例子

常量

lora2.init(ic, loraconfig,spiconfig)

lora_device:set_channel(freq)

lora_device:set_txconfig(txconfig)

lora_device:set_rxconfig(set_rxconfig)

lora_device:send(data)

lora_device:recv(timeout)

lora_device:mode(mode)

lora_device:on(cb)

常量

常量	类型	解释
lora2.SLEEP	number	SLEEP 模式
lora2.STANDBY	number	STANDBY 模式

lora2.init(ic, loraconfig,spiconfig)

lora 初始化

参数

传入值类型	解释
string	lora 型号，当前支持： llcc68 sx1268
table	lora 配置参数,与具体设备有关

返回值

返回值类型	解释
userdata	若成功会返回 lora 对象,否则返回 nil

例子

```
spi_lora =
spi.deviceSetup(spi_id,pin_cs,0,0,8,10*1000*1000,spi.MSB,1,0)

lora_device = lora2.init("llcc68",{res = pin_reset,busy =
pin_busy,dio1 = pin_dio1},spi_lora)
```

---

lora_device:set_channel(freq)

设置频道频率

参数

传入值类型 解释

number 频率

返回值

无

例子

```
lora_device:set_channel(433000000)
```

---

lora_device:set_txconfig(txconfig)

lora 配置发送参数

参数

传入值类型 解释

table lora 发送配置参数,与具体设备有关

返回值

无

例子

```
lora_device:set_txconfig(
    {
        mode=1,
        power=22,
        fdev=0,
        bandwidth=0,
        datarate=9,
        coderate=4,
        preambleLen=8,
        fixLen=false,
        crcOn=true,
        freqHopOn=0,
        hopPeriod=0,
        iqInverted=false,
        timeout=3000
    })
```

---

lora_device:set_rxconfig(set_rxconfig)

lora 配置接收参数

参数

传入值类型	解释
table	lora 接收配置参数,与具体设备有关

返回值

无

例子

```
lora_device:set_rxconfig(
    {
        mode=1,
        bandwidth=0,
        datarate=9,
        coderate=4,
        bandwidthAfc=0,
        preambleLen=8,
        symbTimeout=0,
        fixLen=false,
        payloadLen=0,
        crcOn=true,
        freqHopOn=0,
        hopPeriod=0,
        iqInverted=false,
        rxContinuous=false
    })
```

---

lora_device:send(data)

发数据

参数

传入值类型	解释
string	写入的数据

返回值

无

例子

```
lora_device:send("PING")
```

---

lora_device:recv(timeout)

开启收数据

参数

传入值类型	解释
number	超时时间，默认 1000 单位 ms

返回值

无

例子

```
sys.subscribe("LORA_RX_DONE", function(data, size)
    log.info("LORA_RX_DONE: ", data, size)
    lora_device:send("PING")
    end)
    lora_device:recv(1000)
```

---

lora_device:mode(mode)

设置进入模式(休眠，正常等)

参数

传入值类型	解释
number	模式 正常模式:lora.STANDBY 休眠模式:lora.SLEEP 默认为正常模式

返回值

无

例子

```
lora_device:mode(lora.STANDBY)
```

---

lora_device:on(cb)

注册 lora 回调

参数

传入值类型	解释
function	cb lora 回调,参数包括 lora_device, event, data, size

返回值

返回值类型	解释
nil	无返回值

例子

```
lora_device:on(function(lora_device, event, data, size)
    log.info("lora", "event", event, lora_device, data, size)
    if event == "tx_done" then
        lora_device:recv(1000)
    elseif event == "rx_done" then
        lora_device:send("PING")
    elseif event == "tx_timeout" then

    elseif event == "rx_timeout" then
        lora_device:recv(1000)
    elseif event == "rx_error" then
```

```
    end

    end)--[[event 可能出现的值有    tx_done          -- 发送完成
rx_done          -- 接收完成    tx_timeout       -- 发送超时
rx_timeout       -- 接收超时    rx_error         -- 接收错误]]
```

lvgl - LVGL 图像库

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

小技巧

本库有专属 demo，点此链接查看 lvgl 的 demo 例子

lvgl.draw_mask_radius_param_t()

lvgl.draw_mask_radius_param_t_free(radius)

lvgl.draw_mask_line_param_t()

lvgl.draw_mask_line_param_t_free(line)

lvgl.draw_mask_fade_param_t()

lvgl.draw_mask_fade_param_t_free(fade)

lvgl.font_get(name)

lvgl.font_load(path/spi_device,size,bpp,thickness,cache_size,sty_zh,sty_en)

lvgl.font_free(font)

lvgl.obj_set_event_cb(obj, func)

lvgl.obj_set_signal_cb(obj, func)

lvgl.anim_set_exec_cb(anim, func)

lvgl.anim_set_ready_cb(anim, func)

lvgl.anim_path_set_cb(path, func)

lvgl.event_send(obj, ent)

lvgl.scr_act()

lvgl.layer_top()

lvgl.scr_load(scr)

lvgl.scr_load_anim(scr)

lvgl.theme_set_act(name)

lvgl.sleep(enable)

lvgl.init(w, h, lcd, buff_size, buff_mode)

lvgl.anim_create()

lvgl.anim_free(anim)

lvgl.anim_path_t()

lvgl.anim_path_t_free(anim_path_t)

lvgl.anim_set_path_str(anim, tp)

lvgl.qrcode_create(parent, size, dark_color, light_color)

lvgl.qrcode_update(qrcode, cnt)

lvgl.qrcode_delete(qrcode)

lvgl.style_t()

lvgl.style_create()

lvgl.style_list_create()

lvgl.style_list_t()

lvgl.style_delete(style)

lvgl.style_list_delete(style)

lvgl.demo_benchmark()

lvgl.demo_keypad_encoder()

lvgl.demo_music()

lvgl.demo_printer()

lvgl.demo_stress()

lvgl.demo_widgets()

lvgl.indev_drv_register(tp, dtp)

lvgl.indev_point_emulator_update(x, y, state)

lvgl.indev_kb_update(key)

lvgl.gif_create(parent, path)

lvgl.gif_restart(gif)

lvgl.draw_mask_radius_param_t()

创建一个 lv_draw_mask_radius_param_t

参数

无

返回值

返回值类型	解释
userdata	lv_draw_mask_radius_param_t 指针

例子

```
local radius = lvgl.draw_mask_radius_param_t()
```

---

lvgl.draw_mask_radius_param_t_free(radius)

释放一个 lv_draw_mask_radius_param_t

参数

无

返回值

无

例子

```
local lvgl.draw_mask_radius_param_t_free(radius)
```

---

lvgl.draw_mask_line_param_t()

创建一个 lv_draw_mask_line_param_t

参数

无

返回值

返回值类型	解释
userdata	lv_draw_mask_line_param_t 指针

例子

```
local line = lvgl.draw_mask_line_param_t()
```

---

lvgl.draw_mask_line_param_t_free(line)

释放一个 lv_draw_mask_line_param_t

参数

无

返回值

无

例子

```
local lvgl.draw_mask_line_param_t_free(line)
```

---

lvgl.draw_mask_fade_param_t()

创建一个 lv_draw_mask_fade_param_t

参数

无

返回值

返回值类型	解释
userdata	lv_draw_mask_fade_param_t 指针

例子

```
local fade = lvgl.draw_mask_fade_param_t()
```

---

lvgl.draw_mask_fade_param_t_free(fade)

释放一个 lv_draw_mask_fade_param_t

参数

无

返回值

无

例子

```
local lvgl.draw_mask_fade_param_t_free(fade)
```

---

lvgl.font_get(name)

获取内置字体

参数

传入值类型	解释
string	字体名称+字号，例如 opposans_m_10

返回值

返回值类型	解释
userdata	字体指针

例子

```
local font = lvgl.font_get("opposans_m_12")
```

---

lvgl.font_load(path/spi_device,size,bpp,thickness,cache_size,sty_zh,sty_en)

从文件系统加载字体

参数

传入值类型	解释
string/userdata	字体路径/spi_device (spi_device 为使用外置高通矢量字库芯片)
number	size 可选,字号 16-192 默认 16(使用高通矢量字库)
number	bpp 可选 深度 默认 4(使用高通矢量字库)
number	thickness 可选 粗细值 默认 size * bpp(使用高通矢量字库)
number	cache_size 可选 默认 0(使用高通矢量字库)
number	sty_zh 可选 选择字体 默认 1(使用高通矢量字库)
number	sty_en 可选 选择字体 默认 3(使用高通矢量字库)

返回值

返回值类型	解释
userdata	字体指针

例子

```
local font = lvgl.font_load("/font_32.bin")
--local font = lvgl.font_load(spi_device,16)(高通矢量字库)
```

---

lvgl.font_free(font)

释放字体,慎用!!!仅通过 font_load 加载的字体允许卸载,通过 font_get 获取的字体不允许卸载

参数

传入值类型　解释

string　　字体路径

返回值

返回值类型　解释

userdata　　字体指针

例子

```
local font = lvgl.font_load("/font_32.bin")-- N N N N 操作
-- 确定字体不被使用,不被引用,且内存紧张需要释放
lvgl.font_free(font)
```

---

lvgl.obj_set_event_cb(obj, func)

设置组件的事件回调

参数

传入值类型	解释
userdata	lvgl 组件指针
func	lua 函数，参数有 2 个 (obj, event)，其中 obj 是当前对象, event 是事件类型，为整型

返回值

返回值类型　解释

nil　　无返回值

例子

无

lvgl.obj_set_signal_cb(obj, func)

设置组件的信号回调

参数

传入值类型	解释
userdata	lvgl 组件指针
func	lua 函数，参数有 2 个（obj, signal），其中 obj 是当前对象, signal 是信号类型，为整型

返回值

返回值类型	解释
nil	无返回值

例子

无

lvgl.anim_set_exec_cb(anim, func)

设置动画回调

参数

传入值类型	解释
userdata	动画指针
userdata	lvgl 组件指针

传入值类型	解释
func	lua 函数，参数有 2 个 (obj, value)，其中 obj 是当前对象, signal 是信号类型，为整型

返回值

返回值类型	解释
nil	无返回值

例子

无

---

lvgl.anim_set_ready_cb(anim, func)

设置动画回调

参数

传入值类型	解释
userdata	动画指针
userdata	lvgl 组件指针
func	lua 函数，参数有 1 个 (anim)，其中 anim 是当前对象

返回值

返回值类型	解释
nil	无返回值

例子

无

---

lvgl.anim_path_set_cb(path, func)

设置动画回调

参数

传入值类型	解释
userdata	动画指针
userdata	lvgl 组件指针
func	lua 函数，参数有 1 个 (path)，其中 path 是当前对象

返回值

返回值类型	解释
nil	无返回值

例子

无

---

lvgl.event_send(obj, ent)

发送事件给组件

参数

传入值类型	解释
userdata	组件指针
int	事件 id，例如 lvgl.EVENT_PRESSED

返回值

返回值类型	解释
bool	成功返回 true，对象已被删除的话返回 false 或者 nil
int	底层返回值,如果 obj 为 nil 就返回 nil

例子

无

---

lvgl.scr_act()

获取当前活跃的 screen 对象

参数

无

返回值

返回值类型	解释
指针	screen 指针

例子

```
local scr = lvgl.scr_act()
```

---

lvgl.layer_top()

获取 layer_top

参数

无

返回值

返回值类型　　解释

指针　　layer 指针

例子

无

---

lvgl.layer_sys()

获取 layer_sys

参数

无

返回值

返回值类型　　解释

指针　　layer 指针

例子

无

---

lvgl.scr_load(scr)

载入指定的 screen

参数

传入值类型　　解释

userdata　　screen 指针

返回值

无

例子

```
local scr = lvgl.obj_create(nil, nil)
local btn = lvgl.btn_create(scr)
lvgl.obj_align(btn, lvgl.scr_act(), lvgl.ALIGN_CENTER, 0, 0)
local label = lvgl.label_create(btn)
lvgl.label_set_text(label, "LuatOS!")
lvgl.scr_load(scr)
```

---

lvgl.scr_load_anim(scr)

载入指定的 screen 并使用指定的转场动画

参数

传入值类型	解释
userdata	screen 指针

返回值

无

例子

```
local scr = lvgl.obj_create(nil, nil)
local btn = lvgl.btn_create(scr)
lvgl.obj_align(btn, lvgl.scr_act(), lvgl.ALIGN_CENTER, 0, 0)
local label = lvgl.label_create(btn)
lvgl.label_set_text(label, "LuatOS!")

local scr2 = lvgl.obj_create(nil,nil)
```

```
    local btn2 = lvgl.btn_create(scr2)
    lvgl.obj_align(btn, scr2, lvgl.ALIGN_CENTER, 0, 20)
    local label2 = lvgl.label_create(btn2)
    lvgl.label_set_text(label2, "Btn2")
    lvgl.scr_load(scr)
    --sys.wait(1000);

lvgl.scr_load_anim(scr2,lvgl.SCR_LOAD_ANIM_OVER_LEFT,100,100,false)
```

原函数：`lv_scr_load_anim(lv_obj_t * new_scr, lv_scr_load_anim_t anim_type, uint32_t time, uint32_t delay, bool auto_del)`

---

lvgl.theme_set_act(name)

设置主题

参数

传入

值类型	解释
string	主题名称,可选值有 default/mono/empty/material_light/material_dark/material_no_transition/material_no_focus

返回值

返回值类型	解释
bool	成功返回 true,否则返回 nil

例子

```
-- 黑白主题
```

```
lvgl.theme_set_act("mono")
-- 空白主题
lvgl.theme_set_act("empty")
```

---

lvgl.sleep(enable)

LVGL 休眠控制，暂停/恢复刷新定时器，目前只有 105 和 EC618 可以用

参数

传入值类型	解释
boolean	true 暂停 false 恢复

返回值

无

例子

```
lvgl.sleep(true)                --暂停刷新，系统可以休眠
lvgl.sleep(false)               --恢复刷新，系统不休眠
```

---

lvgl.init(w, h, lcd, buff_size, buff_mode)

初始化 LVGL

参数

传入值类型	解释
int	屏幕宽,可选,默认从 lcd 取
int	屏幕高,可选,默认从 lcd 取
userdata	lcd 指针,可选,lcd 初始化后有默认值,预留的多屏入口
int	缓冲区大小,默认宽*10, 不含色深.

传入值类型	解释
int	缓冲模式,默认 0，单 buff 模式，可选 1,双 buff 模式

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

无

---

lvgl.anim_create()

创建并初始化一个 anim

参数

无

返回值

返回值类型	解释
userdata	anim 指针

例子

```
local anim = lvgl.anim_create()
```

---

lvgl.anim_free(anim)

释放一个 anim

参数

无

返回值

无

例子

```
local lvgl.anim_free(anim)
```

---

lvgl.anim_path_t()

创建一个 lv_anim_path_t

参数

无

返回值

返回值类型	解释
userdata	lv_anim_path_t 指针

例子

```
local anim_path_t = lvgl.anim_path_t()
```

---

lvgl.anim_path_t_free(anim_path_t)

释放一个 lv_anim_path_t

参数

无

返回值

无

例子

```
local lvgl.anim_path_t_free(anim_path_t)
```

---

lvgl.anim_set_path_str(anim, tp)

设置动画路径方式

参数

传入值类型	解释
userdata	动画指针
string	类型，支持 linear/ease_in/ease_out/ease_in_out/overshoot/bounce/step

返回值

返回值类型	解释
nil	无返回值

例子

无

---

lvgl.qrcode_create(parent, size, dark_color, light_color)

创建 qrcode 组件

参数

传入值类型	解释

传入值类型	解释
userdata	父组件
int	长度,因为 qrcode 是正方形
int	二维码中数据点的颜色, RGB 颜色, 默认 0x3333ff
int	二维码中背景点的颜色, RGB 颜色, 默认 0xeeeeff

返回值

返回值类型	解释
userdata	qrcode 组件

例子

```
-- 创建并显示 qrcode
local qrcode = lvgl.qrcode_create(scr, 100)
lvgl.qrcode_update(qrcode, "https://luatos.com")
lvgl.obj_align(qrcode, lvgl.scr_act(), lvgl.ALIGN_CENTER, -100, -100)
```

---

lvgl.qrcode_update(qrcode, cnt)

设置 qrcode 组件的二维码内容,配合 qrcode_create 使用

参数

传入值类型	解释
userdata	qrcode 组件,由 qrcode_create 创建
string	二维码的内容数据

返回值

返回值类型	解释

返回值类型	解释
bool	更新成功返回 true,否则返回 false. 通常只有数据太长无法容纳才会返回 false

例子

无

---

lvgl.qrcode_delete(qrcode)

删除 qrcode 组件

参数

传入值类型	解释
userdata	qrcode 组件,由 qrcode_create 创建

返回值

返回值类型	解释
nil	无返回值

例子

无

---

lvgl.style_t()

创建一个 style

参数

无

返回值

返回值类型	解释
userdata	style 指针

例子

```
local style = lvgl.style_t()
lvgl.style_init(style)
```

---

lvgl.style_create()

创建一个 style 并初始化

参数

无

返回值

返回值类型	解释
userdata	style 指针

例子

```
local style = lvgl.style_create()
```

---

lvgl.style_list_create()

创建一个 style_list

参数

无

返回值

返回值类型　解释

userdata　style 指针

例子

```
local style_list = lvgl.style_list_create()
```

---

lvgl.style_list_t()

创建一个 style_list

参数

无

返回值

返回值类型　解释

userdata　style 指针

例子

```
local style = lvgl.style_list_t()
```

---

lvgl.style_delete(style)

删除 style,慎用,通常不会执行删除操作

参数

传入值类型　解释

传入值类型　解释

userdata　style 指针

返回值

无

例子

```
local style = lvgl.style_create()
-- ...
-- ...
-- lvgl.style_delete(style)
```

---

lvgl.style_list_delete(style)

删除 style_list,慎用,通常不会执行删除操作

参数

传入值类型　解释

userdata　style 指针

返回值

无

例子

```
local style_list = lvgl.style_list_create()
-- ...
-- ...
-- lvgl.style_list_delete(style_list)
```

---

lvgl.demo_benchmark()

lvgl benchmark demo

参数

无

返回值

返回值类型	解释
nil	无返回值

例子

```
lvgl.init()
lvgl.demo_benchmark()
```

---

lvgl.demo_keypad_encoder()

lvgl keypad_encoder demo

参数

无

返回值

返回值类型	解释
nil	无返回值

例子

```
lvgl.init()
lvgl.demo_keypad_encoder()
```

---

lvgl.demo_music()

lvgl music demo

参数

无

返回值

返回值类型	解释
nil	无返回值

例子

```
lvgl.init()
lvgl.demo_music()
```

---

lvgl.demo_printer()

lvgl printer demo

参数

无

返回值

返回值类型	解释
nil	无返回值

例子

```
lvgl.init()
lvgl.demo_printer()
```

---

lvgl.demo_stress()

lvgl stress demo

参数

无

返回值

返回值类型	解释
nil	无返回值

例子

```
lvgl.init()
lvgl.demo_stress()
```

---

lvgl.demo_widgets()

lvgl widgets demo

参数

无

返回值

返回值类型	解释
nil	无返回值

例子

```
lvgl.init()
lvgl.demo_widgets()
```

---

lvgl.indev_drv_register(tp, dtp)

注册输入设备驱动

参数

传入值类型	解释
string	设备类型，当前支持”pointer”,指针类/触摸类均可，”keyboard”,键盘类型
string	设备型号，当前支持”emulator”,模拟器类型

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

```
lvgl.indev_drv_register("pointer", "emulator")
```

---

lvgl.indev_point_emulator_update(x, y, state)

更新模拟输入设备的坐标数据

参数

传入值类型	解释
int	x 坐标,以左上角为 0,右下角为最大值
int	y 坐标,以左上角为 0,右下角为最大值
int	状态, 0 为 释放, 1 为按下

返回值

返回值类型　解释

nil	无返回值

例子

```
-- 模拟在屏幕上的点击,通过 timeout 模拟长按和短按
sys.taskInit(function(x, y, timeout)
    lvgl.indev_point_emulator_update(x, y, 1)
    sys.wait(timeout)
    lvgl.indev_point_emulator_update(x, y, 0)
end, 240, 120, 50)
```

---

lvgl.indev_kb_update(key)

更新键盘输入设备的按键值

参数

传入值类型　　解释

int	按键值，默认为 0，按键抬起

返回值

返回值类型　解释

nil	无返回值

例子

无

---

lvgl.gif_create(parent, path)

创建 gif 组件

参数

传入值类型	解释
userdata	父组件,可以是 nil,但通常不会是 nil
string	文件路径

返回值

返回值类型	解释
userdata	组件指针,若失败会返回 nil,建议检查

例子

```
local gif = lvgl.gif_create(scr, "S/emtry.gif")
if gif then
    log.info("gif", "create ok")
end
```

---

lvgl.gif_restart(gif)

重新播放 gif 组件

参数

传入值类型	解释
userdata	gif 组件支持，由 gif_create 方法返回

返回值

返回值类型	解释
nil	无返回值

例子

```
local gif = lvgl.gif_create(scr, "S/emtry.gif")
if gif then
    log.info("gif", "create ok")
end
```

max30102 - 心率模块(MAX30102)

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

max30102.init(i2c_id,int)

max30102.get()

max30102.shutdown()

max30102.init(i2c_id,int)

初始化 MAX30102 传感器

参数

传入值类型	解释
int	传感器所在的 i2c 总线 id,默认为 0
int	int 引脚

返回值

返回值类型	解释
bool	成功返回 true，否则返回 nil 或者 false

例子

```
if max30102.init(0,pin.PC05) then
    log.info("max30102", "init ok")
```

```
else
    log.info("max30102", "init fail")
end
```

---

max30102.get()

获取心率血氧(大概需要 10s 时间测量)

参数

无

返回值

返回值类型	解释
bool	成功返回 true，否则返回 nil 或者 false
number	心率
number	血氧

例子

无

---

max30102.shutdown()

关闭 max30102

参数

无

返回值

无

例子

无

mcu - 封装 mcu 一些特殊操作

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

常量
mcu.setClk(mhz)
mcu.getClk()
mcu.unique_id()
mcu.ticks()
mcu.hz()
mcu.reg32(address, value, mask)
mcu.x32(value)
mcu.tick64()
mcu.dtick64(tick1, tick2, check_value)
mcu.setXTAL(source_main, source_32k, delay)
mcu.hardfault(mode)
mcu.iomux(type, channel, value)
mcu.altfun(type, sn, pad_index, alt_fun, is_input)

常量

常量	类型	解释
mcu.UART	number	外设类型-串口
mcu.I2C	number	外设类型-I2C
mcu.SPI	number	外设类型-SPI
mcu.PWM	number	外设类型-PWM
mcu.GPIO	number	外设类型-GPIO

常量	类型	解释
mcu.I2S	number	外设类型-I2S
mcu.LCD	number	外设类型-LCD
mcu.CAM	number	外设类型-CAM

mcu.setClk(mhz)

设置主频,单位 MHZ

参数

传入值类型	解释
int	主频,根据设备的不同有不同的有效值,请查阅手册

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

```
-- 注意：并非所有模块都支持主频调整,请查阅手册

-- Air101/Air103/Air601 支持设置成 2/40/80/160/240. 特别提醒，设置到 2M 后，如果要休眠，必须先设置到 80M

-- ESP32 系列支持设置成 40/80/160/240 ，需要 2024.1.1 之后的固件

-- Air780 系列，Air105，不支持设置主频-- Air780 系列，进入休眠模式时自动降频到 24M

-- 设置到 80MHZ

mcu.setClk(80)

sys.wait(1000)

-- 设置到 240MHZ

mcu.setClk(240)

sys.wait(1000)
```

```
-- 设置到 2MHZ
mcu.setClk(2)
sys.wait(1000)
```

---

mcu.getClk()

获取主频,单位 MHZ

参数

无

返回值

返回值类型	解释
int	若失败返回-1,否则返回主频数值,若等于 0,可能处于 32k 晶振的省电模式

例子

```
local mhz = mcu.getClk()
print("Boom", mhz)
```

---

mcu.unique_id()

获取设备唯一 id. 注意,可能包含不可见字符,如需查看建议 toHex()后打印

参数

无

返回值

返回值类型	解释
string	设备唯一 id.若不支持，会返回空字符串.

例子

```
local unique_id = mcu.unique_id()
print("unique_id", unique_id)
```

---

mcu.ticks()

获取启动后的 tick 数,注意会出现溢出会出现负数

参数

无

返回值

返回值类型	解释
int	当前 tick 值

例子

```
local tick = mcu.ticks()
print("ticks", tick)
```

---

mcu.hz()

获取每秒的 tick 数量

参数

无

返回值

返回值类型	解释
int	每秒的 tick 数量,通常为 1000

例子

```
local hz = mcu.hz()
print("mcu.hz", hz)
```

---

mcu.reg32(address, value, mask)

读写 mcu 的 32bit 寄存器或者 ram,谨慎使用写功能,请熟悉 mcu 的寄存器使用方法后再使用

参数

传入值类型	解释
int	寄存器或者 ram 地址
int	写入的值,如果没有,则直接返回当前值
int	位掩码,可以对特定几个位置的 bit 做修改, 默认 0xffffffff,修改全部 32bit

返回值

返回值类型	解释
int	返回当前寄存的值

例子

```
local value = mcu.reg32(0x2009FFFC, 0x01, 0x01) --对 0x2009FFFC 地址上的值,修改 bit0 为 1
```

---

mcu.x32(value)

转换 10 进制数为 16 进制字符串输出

参数

传入值类型	解释
int	需要转换的值

返回值

返回值类型	解释
string	16 进制字符串

例子

```
local value = mcu.x32(0x2009FFFC) --输出"0x2009fffc"
```

---

mcu.tick64()

获取启动后的高精度 tick，如果支持 bit64 库，可以直接输出转换好的 bit64 结构

参数

传入值类型	解释
boolean	是否输出 bit64 结构,true 是,其他都是 false,留空也是 false,用于兼容旧的 demo

返回值

返回值类型	解释
string	当前 tick 值,8 个字节的 uint64,如果支持 64bit 库,同时要求输出 64bit

返回值类型	解释
	结构的话,会输出 9 字节的 string
int	1us 有几个 tick,0 表示未知

例子

```
local tick_str, tick_per = mcu.tick64()
print("ticks", tick_str, tick_per)
```

---

mcu.dtick64(tick1, tick2, check_value)

计算 2 个 64bit tick 的差值

参数

传入值类型	解释
string	64bit 的 string
string	64bit 的 string
int	参考值,可选项,如果为 0,则返回结果中第一个项目为 true

返回值

返回值类型	解释
boolean	与参考值比较,如果大于等于为 true,反之为 false
int	差值 tick1 - tick2,如果超过了 0x7fffffff,结果可能是错的

例子

```
local result, diff_tick = mcu.dtick64(tick1, tick2)
print("ticks", result, diff_tick)
```

---

mcu.setXTAL(source_main, source_32k, delay)

选择时钟源,当前仅 air105 支持

参数

传入值类型	解释
boolean	高速时钟是否使用外部时钟源,如果为空则不改变
boolean	低速 32K 是否使用外部时钟源,如果为空则不改变
int	PLL 稳定时间,在切换高速时钟的时候,根据硬件环境,需要 delay 一段时间等待 PLL 稳定,默认是 1200,建议不小于 1024

返回值

无

例子

```
mcu.setXTAL(true, true, 1248)       --高速时钟使用外部时钟,低速 32K 使用外部晶振, delay1248
```

---

mcu.hardfault(mode)

mcu 死机时处理模式，目前只有 EC618 平台适用

参数

传入值类型	解释
int	处理模式，0 死机停机，1 死机后重启，2 死机后尽量将错误信息提交给外部工具后重启

返回值

无

例子

```
mcu.hardfault(0)      --死机后停机，一般用于调试状态
mcu.hardfault(1)      --死机后重启，一般用于正式产品
mcu.hardfault(2)      --死机后尽量将错误信息提交给外部工具后重启，一般用于压力测试或者正式产品
```

---

mcu.iomux(type, channel, value)

在外设打开前，将外设 IO 复用到非默认配置上，目前只支持 Air780E 的部分外设复用到其他配置，这是一个临时接口，如果后续有更合适的 api，本接口将不再更新

参数

传入值类型	解释
int	外设类型，目前只有 mcu.UART,mcu.I2C
int	总线序号，0~N，
int	新的配置，这个需要根据具体平台决定

返回值

无

例子

```
mcu.iomux(mcu.UART, 2, 1)
-- Air780E 的 UART2 复用到 gpio12 和 gpio13(Air780EG 默认是这个复用，不要动)
mcu.iomux(mcu.UART, 2, 2)    -- Air780E 的 UART2 复用到 gpio6 和 gpio7
mcu.iomux(mcu.I2C, 0, 1)     -- Air780E 的 I2C0 复用到 gpio12 和 gpio13
```

```
mcu.iomux(mcu.I2C, 0, 2)     -- Air780E 的 I2C0 复用到 gpio16 和 gpio17
mcu.iomux(mcu.I2C, 1, 1)     -- Air780E 的 I2C1 复用到 gpio4 和 gpio5
```

---

mcu.altfun(type, sn, pad_index, alt_fun, is_input)

IO 外设功能复用选择，注意普通 MCU 是以 GPIO 号为序号，但是专用 SOC，比如 CAT1 的，是以 PAD 号为序号。本函数不是所有平台适用

参数

传入值类型	解释
int	外设类型，目前有 mcu.UART,mcu.I2C,mcu.SPI,mcu.PWM,mcu.GPIO,mcu.I2S,mcu.LCD,mcu.CAM，具体需要看平台
int	总线序号，0~N，如果是 mcu.GPIO，则是 GPIO 号。具体看平台的 IOMUX 复用表
int	pad 号，如果留空不写，则表示清除配置，使用平台的默认配置。具体看平台的 IOMUX 复用表
int	复用功能序号，0~N。具体看平台的 IOMUX 复用表
boolean	是否是输入功能，true 是，留空是 false

返回值

无

例子

```
mcu.altfun(mcu.GPIO, 46, 32, 1, 0)mcu.altfun(mcu.GPIO, 46)
```

miniz - 简易 zlib 压缩

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

示例

```lua
-- 准备好数据
local bigdata = "123jfoiq4hlkfjbnasdilfhuqwo;hfashfp9qw38hrfaios;hfiuoaghfluaeisw"
-- 压缩之, 压缩得到的数据是 zlib 兼容的,其他语言可通过 zlib 相关的库进行解压
local cdata = miniz.compress(bigdata)
-- lua 的 字符串相当于有长度的 char[],可存放包括 0x00 的一切数据
if cdata then
    -- 检查压缩前后的数据大小
    log.info("miniz", "before", #bigdata, "after", #cdata)
    log.info("miniz", "cdata as hex", cdata:toHex())

    -- 解压, 得到原文
    local udata = miniz.uncompress(cdata)
    log.info("miniz", "udata", udata)
end
```

常量

miniz.compress(data, flags)

miniz.uncompress(data, flags)

常量

常量	类型	解释
miniz.WRITE_ZLIB_HEADER	int	压缩参数,是否写入 zlib 头部数据,compress 函数的默认值
miniz.COMPUTE_ADLER32	int	压缩/解压参数,是否计算/校验 adler-32
miniz.GREEDY_PARSING_FLAG	int	压缩参数,是否快速 greedy 处理,默认使用较慢的处理模式
miniz.NONDETERMINISTIC_PARSING_FLAG	int	压缩参数,是否快速初始化压缩器
miniz.RLE_MATCHES	int	压缩参数, 仅扫描 RLE
miniz.FILTER_MATCHES	int	压缩参数,过滤少于 5 次的字符
miniz.FORCE_ALL_STATIC_BLOCKS	int	压缩参数,是否禁用优化过的 Huffman 表
miniz.FORCE_ALL_RAW_BLOCKS	int	压缩参数,是否只要 raw 块
miniz.PARSE_ZLIB_HEADER	int	解压参数,是否处理 zlib 头部,uncompress 函数的默认值
miniz.HAS_MORE_INPUT	int	解压参数,是否还有更多数据,仅流式解压可用,暂不支持
miniz.USING_NON_WRAPPING_OUTPUT_BUF	int	解压参数,解压区间是否够全部数据,,仅流式解压可用,暂不支持
miniz.COMPUTE_ADLER32	int	解压参数,是否强制校验 adler-32

miniz.compress(data, flags)

快速压缩,需要 165kb 的系统内存和 32kb 的 LuaVM 内存

参数

传入值类型	解释
string	待压缩的数据，少于 400 字节的数据不建议压缩，且压缩后的数据不能大于 32k.
flags	压缩参数,默认是 miniz.WRITE_ZLIB_HEADER，即写入 zlib 头部

返回值

返回值类型	解释
string	若压缩成功,返回数据字符串，否则返回 nil

例子

```
local bigdata = "123jfoiq4hlkfjbnasdilfhuqwo;hfashfp9qw38hrfaios;hfiuoaghfluaeisw"
local cdata = miniz.compress(bigdata)
if cdata then
    log.info("miniz", "before", #bigdata, "after", #cdata)
    log.info("miniz", "cdata as hex", cdata:toHex())
end
```

---

miniz.uncompress(data, flags)

快速解压,需要 32kb 的 LuaVM 内存

参数

传入值类型	解释
string	待解压的数据，解压后的数据不能大于 32k
flags	解压参数,默认是 miniz.PARSE_ZLIB_HEADER，即解析 zlib 头部

返回值

返回值类型	解释
string	若解压成功,返回数据字符串，否则返回 nil

例子

```
local bigdata = "123jfoiq4hlkfjbnasdilfhuqwo;hfashfp9qw38hrfaios;hfiuoaghfluaeisw"
local cdata = miniz.compress(bigdata)
if cdata then
    log.info("miniz", "before", #bigdata, "after", #cdata)
    log.info("miniz", "cdata as hex", cdata:toHex())

    local udata = miniz.uncompress(cdata)
    log.info("miniz", "udata", udata)
end
```

mlx90640 - 红外测温(MLX90640)

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

常量

mlx90640.init(i2c_id,refresh_rate) (注意:2023.5.15 之后使用此接口,用户需要自行初始化 i2c 接口)

mlx90640.feed()

mlx90640.raw_data()

mlx90640.raw_point(index)

mlx90640.ta_temp()

mlx90640.max_temp()

mlx90640.min_temp()

mlx90640.average_temp()

mlx90640.get_vdd()

mlx90640.draw2lcd(x, y, fold)

常量

常量	类型	解释
mlx90640.FPS1HZ	number	FPS1HZ
mlx90640.FPS2HZ	number	FPS2HZ
mlx90640.FPS4HZ	number	FPS4HZ
mlx90640.FPS8HZ	number	FPS8HZ
mlx90640.FPS16HZ	number	FPS16HZ
mlx90640.FPS32HZ	number	FPS32HZ
mlx90640.FPS64HZ	number	FPS64HZ

mlx90640.init(i2c_id,refresh_rate) (注意:2023.5.15 之后使用此接口,用户需要自行初始化 i2c 接口)

初始化 MLX90640 传感器

参数

传入值类型	解释
int	传感器所在的 i2c 总线 id 或者软 i2c 对象,默认为 0
int	传感器的测量速率,默认为 4Hz

返回值

返回值类型	解释
bool	成功返回 true，否则返回 nil 或者 false

例子

```
i2c.setup(i2cid,i2c_speed)
if mlx90640.init(0,mlx90640.FPS4HZ) then
    log.info("mlx90640", "init ok")
    sys.wait(500) -- 稍等片刻
    while 1 do
        mlx90640.feed() -- 取一帧数据
        mlx90640.draw2lcd(0, 0 ,1)-- 需提前把 lcd 初始化好
        sys.wait(250) -- 默认是 4HZ
    end
else
    log.info("mlx90640", "init fail")
end
```

---

mlx90640.feed()

取一帧数据

参数

无

返回值

无

例子

无

---

mlx90640.raw_data()

获取底层裸数据,浮点数矩阵

参数

无

返回值

返回值类型	解释
table	浮点数数据,768 个像素对应的温度值

例子

无

---

mlx90640.raw_point(index)

获取单一点数据

参数

传入值类型	解释
int	索引值(0-767)

返回值

返回值类型	解释
number	单点温度值

例子

无

---

mlx90640.ta_temp()

获取外壳温度

参数

无

返回值

返回值类型	解释
number	外壳温度

例子

无

---

mlx90640.max_temp()

获取最高温度

参数

无

返回值

返回值类型	解释
number	最高温度
number	最高温度位置

例子

无

---

mlx90640.min_temp()

获取最低温度

参数

无

返回值

返回值类型	解释
number	最低温度
number	最低温度位置

例子

无

---

mlx90640.average_temp()

获取平均温度

参数

无

返回值

返回值类型	解释
number	平均温度

例子

无

---

mlx90640.get_vdd()

获取 vdd

参数

无

返回值

返回值类型 解释

number vdd

例子

无

---

mlx90640.draw2lcd(x, y, fold)

绘制到 lcd

参数

传入值类型	解释
int	左上角 x 坐标
int	左上角 y 坐标
int	放大倍数,必须为 2 的指数倍(1,2,4,8,16…)默认为 1

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

无

mobile - 蜂窝网络

已适配 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 mobile 的 demo 例子

示例

```
-- 简单演示
log.info("imei", mobile.imei())
log.info("imsi", mobile.imsi())
local sn = mobile.sn()
if sn then
    log.info("sn",   sn:toHex())
End
log.info("muid", mobile.muid())
log.info("iccid", mobile.iccid())
log.info("csq", mobile.csq())
log.info("rssi", mobile.rssi())
log.info("rsrq", mobile.rsrq())
log.info("rsrp", mobile.rsrp())
log.info("snr", mobile.snr())
log.info("simid", mobile.simid())
```

常量

mobile.imei(index)

mobile.imsi(index)

mobile.sn()

mobile.muid()

mobile.iccid(id)

mobile.number(id)

mobile.simid(id)

mobile.simPin(id,operation,pin1,pin2)

mobile.rtime(time, auto_reset_stack)

mobile.setAuto(check_sim_period, get_cell_period, search_cell_time, auto_reset_stack, network_check_period)

mobile.apn(index, cid, new_apn_name, user_name, password, ip_type, protocol)

mobile.ipv6(onff)

mobile.csq()

mobile.rssi()

mobile.rsrp()

mobile.rsrq()

mobile.snr()

mobile.eci()

mobile.tac()

mobile.enbid()

mobile.flymode(index, enable)

mobile.status()

mobile.getCellInfo()

mobile.reqCellInfo(timeout)

mobile.reset()

mobile.dataTraffic(clearUplink, clearDownlink)

mobile.config(item, value)

mobile.getBand(band, is_default)

mobile.setBand(band, num)

mobile.nstOnOff(onoff, uart_id)

mobile.nstInput(data)

常量

常量	类型	解释
mobile.UNREGISTER	number	未注册
mobile.REGISTERED	number	已注册
mobile.SEARCH	number	正在搜索中
mobile.DENIED	number	注册被拒绝
mobile.UNKNOW	number	未知
mobile.REGISTERED_ROAMING	number	已注册,漫游
mobile.SMS_ONLY_REGISTERED	number	已注册,仅 SMS
mobile.SMS_ONLY_REGISTERED_ROAMING	number	已注册,漫游,仅 SMS
mobile.EMERGENCY_REGISTERED	number	已注册,紧急服务
mobile.CSFB_NOT_PREFERRED_REGISTERED	number	已注册,非主要服务
mobile.CSFB_NOT_PREFERRED_REGISTERED_ROAMING	number	已注册,非主要服务,漫游
mobile.CONF_RESELTOWEAKNCELL	number	小区重选信号差值门限,需要飞行模式设置
mobile.CONF_STATICCONFIG	number	网络静态模式优化,需要飞行模式设置
mobile.CONF_QUALITYFIRST	number	网络切换以信号质量优先,需要飞

常量	类型	解释
		行模式设置,0 不开,1 开启,2 开启并加速切换,功耗会增加
mobile.CONF_USERDRXCYCLE	number	LTE 跳 paging,需要飞行模式设置,谨慎使用,0 是不设置,1~7 增大或减小 DrxCycle 周期倍数,1:1/8 倍 2:1/4 倍 3:1/2 倍 4:2 倍 5:4 倍 6:8 倍 7:16 倍,8~12 配置固定的 DrxCycle 周期,仅当该周期大于网络分配的 DrxCycle 周期时该配置才会生效,8:320ms 9:640ms 10:1280ms 11:2560ms 12:5120ms
mobile.CONF_T3324MAXVALUE	number	PSM 模式中的 T3324 时间,单位 S
mobile.CONF_PSM_MODE	number	PSM 模式开关,0 关,1 开
mobile.CONF_CE_MODE	number	attach 模式，0 为

常量	类型	解释
		EPS ONLY 2 为混合，遇到 IMSI detach 脱网问题，设置为 0，注意设置为 EPS ONLY 时会取消短信功能
mobile.CONF_SIM_WC_MODE	number	SIM 写入次数的配置和读取
mobile.PIN_VERIFY	number	验证 PIN 码操作
mobile.PIN_CHANGE	number	更换 PIN 码操作
mobile.PIN_ENABLE	number	使能 PIN 码验证
mobile.PIN_DISABLE	number	关闭 PIN 码验证
mobile.PIN_UNBLOCK	number	解锁 PIN 码

mobile.imei(index)

获取 IMEI

参数

传入值类型	解释
int	编号,默认 0. 在支持双卡的模块上才会出现 0 或 1 的情况

返回值

返回值类型	解释
string	当前的 IMEI 值,若失败返回 nil

例子

无

---

mobile.imsi(index)

获取 IMSI

参数

传入值类型	解释
int	编号,默认 0. 在支持双卡的模块上才会出现 0 或 1 的情况

返回值

返回值类型	解释
string	当前的 IMSI 值,若失败返回 nil

例子

无

---

mobile.sn()

获取 SN

参数

无

返回值

返回值类型	解释
string	当前的 SN 值,若失败返回 nil. 注意, SN 可能包含不可见字符

例子

```
-- 注意，出厂未必有写 SN
```

```
-- 一般用途的唯一 id，可以用 mobile.imei()代替
-- 如需要真正的唯一 ID，使用 mcu.unique_id()
```

---

mobile.muid()

获取 MUID

参数

无

返回值

返回值类型	解释
string	当前的 MUID 值,若失败返回 nil

例子

无

---

mobile.iccid(id)

获取或设置 ICCID

参数

传入值类型	解释
int	SIM 卡的编号，例如 0, 1，默认 0

返回值

返回值类型	解释
string	ICCID 值,若失败返回 nil

例子

无

---

mobile.number(id)

获取手机卡号，注意，只有写入了手机号才能读出，因此有可能读出来是空的

参数

传入值类型	解释
int	SIM 卡的编号，例如 0, 1，默认 0

返回值

返回值类型	解释
string	number 值,若失败返回 nil

例子

无

---

mobile.simid(id)

获取当前 SIM 卡槽,或者切换卡槽

参数

传入值类型	解释
int	SIM 卡的编号，例如 0, 1，如果支持双卡，比如 EC618，可以填 2 来自适应，但是会占用掉 4 个 IO(gpio4/5/6/23)。如果不填就直接读取当前卡槽

传入值类型

传入值类型	解释
boolean	是否优先用 SIM0，只有 SIM 卡编号写 2 自适应才有用！！！。true 优先用 SIM0，false 则由具体平台决定，支持双卡双待 SIM0 优先，不支持的是上一次检测到的优先，默认是 false，必须在开机就配置，否则就无效了

返回值

返回值类型	解释
int	当前 sim 卡槽编号,若失败返回-1

例子

```
mobile.simid(0) -- 固定使用 SIM0

mobile.simid(1) -- 固件使用 SIM1

mobile.simid(2) -- 自动识别 SIM0, SIM1, 优先级看具体平台
mobile.simid(2, true)  -- 自动识别 SIM0, SIM1, 且 SIM0 优先

-- 提醒, 自动识别是会增加时间的
```

---

mobile.simPin(id,operation,pin1,pin2)

检测当前 SIM 卡是否准备好，对 SIM 卡的 PIN 码做相关操作

参数

传入值类型	解释
int	SIM 卡的编号，例如 0, 1，支持双卡双待的才需要选择
int	PIN 码操作类型，只能是 mobile.PIN_XXXX，不操作就留空
string	更换 pin 时操作的 pin 码，或者验证操作的 pin 码，或者解锁 pin 码时的 PUK，4~8 字节

传入值类型 | **解释**

string | 更换 pin 码操作时的新的 pin 码，解锁 pin 码时的新 PIN，4~8 字节

返回值

返回值类型 | **解释**

boolean | 当无 PIN 操作时，返回 SIM 卡是否准备好，有 PIN 操作时，返回是否成功

例子

```
local cpin_is_ready = mobile.simPin()
-- 当前 sim 卡是否准备好，一般返回 false 就是没卡
local succ = mobile.simPin(0, mobile.PIN_VERIFY, "1234")
-- 输入 pin 码验证
```

---

mobile.rtime(time, auto_reset_stack)

设置 RRC 自动释放时间间隔，当开启时后，遇到极弱信号+频繁数据操作可能会引起网络严重故障，因此需要额外设置自动重启协议栈

参数

传入值类型 | **解释**

int | RRC 自动释放时间，等同于 Air724 的 AT+RTIME，单位秒，写 0 或者不写则是停用，不要超过 20 秒，没有意义

boolean | 网络遇到严重故障时尝试自动恢复，和飞行模式/SIM 卡切换冲突，true 开启，false 关闭，留空时，如果设置了时间则自动开启。原厂优化过协议栈后不需要了。本参数废弃

返回值

返回值类型	解释
nil	无返回值

例子

无

---

mobile.setAuto(check_sim_period, get_cell_period, search_cell_time, auto_reset_stack, network_check_period)

设置一些辅助周期性或者自动功能，目前支持 SIM 卡暂时脱离后恢复，周期性获取小区信息，网络遇到严重故障时尝试自动恢复

参数

传入值类型	解释
int	SIM 卡自动恢复时间，单位毫秒，建议 5000~10000，和飞行模式/SIM 卡切换冲突，不能再同一时间使用，必须错开执行。写 0 或者不写则是关闭功能
int	周期性获取小区信息的时间间隔，单位毫秒。获取小区信息会增加部分功耗。写 0 或者不写则是关闭功能
int	每次搜索小区时最大搜索时间，单位秒。不要超过 8 秒
boolean	网络遇到严重故障时尝试自动恢复，和飞行模式/SIM 卡切换冲突，true 开启，false 关闭，开始状态是 false，留空则不做改变
int	设置定时检测网络是否正常并且在检测到长时间无网时通过重启协议栈来恢复，无网恢复时长，单位 ms，建议 60000 以上，为网络搜索网络保留足够的时间，留空则不做更改

返回值

返回值类型	解释
nil	无返回值

例子

无

---

mobile.apn(index, cid, new_apn_name, user_name, password, ip_type, protocol)

获取或设置 APN，设置 APN 必须在入网前就设置好，比如在 SIM 卡识别完成前就设置好

参数

传入值类型	解释
int	编号,默认 0. 在支持双卡的模块上才会出现 0 或 1 的情况
int	cid，默认 0，如果要用非默认 APN 来激活，必须>0
string	新的 APN,不填就是获取 APN，填了就是设置 APN，是否支持设置取决于底层实现
string	新的 APN 的 username,如果 APN 不是空,那必须填写,如果没有留个空字符串””。如果 APN 是空的，那可以 nil
string	新的 APN 的 password,如果 APN 不是空,那必须填写,如果没有留个空字符串””。如果 APN 是空的，那可以 nil
int	激活 APN 时的 IP TYPE,1=IPV4 2=IPV6 3=IPV4V6,默认是 1
int	激活 APN 时,如果需要 username 和 password,就要写鉴权协议类型,1~3,默认 3,代表 1 和 2 都尝试一下。不需要鉴权的写 0
boolean	是否删除 APN,true 是,其他都否,只有参数 3 新的 APN 不是 string 的时候才有效果

返回值

返回值类型	解释
string	获取到的默认 APN 值,失败返回 nil

例子

```
mobile.apn(0,1,"cmiot","","",nil,0)
-- 移动公网卡设置 APN 为 cmiot,一般不用设置
mobile.apn(0,1,"name","user","password",nil,3)
-- 专网卡设置的 demo，name，user，password 联系卡商获取
```

---

mobile.ipv6(onff)

是否默认开启 IPV6 功能，必须在 LTE 网络连接前就设置好

参数

传入值类型	解释
boolean	开关 true 开启 false 关闭

返回值

返回值类型	解释
boolean	true 当前是开启的，false 当前是关闭的

例子

```
-- 注意，开启 ipv6 后，开机联网会慢 2~3 秒
```

---

mobile.csq()

获取 csq

参数

无

返回值

返回值类型	解释
int	当前 CSQ 值，若失败返回 0. 范围 0 - 31，越大越好

例子

```
-- 注意，4G 模块的 CSQ 值仅供参考，rsrp/rsrq 才是真正的信号强度指标
```

---

mobile.rssi()

获取 rssi

参数

无

返回值

返回值类型	解释
int	当前 rssi 值,若失败返回 0. 范围 0 到 -114，越小越好

例子

无

---

mobile.rsrp()

获取 rsrp,参考信号接收功率

参数

无

返回值

返回值类型	解释
int	当前 rsrp 值,若失败返回 0. 取值范围: -44 ~ -140 ， 值越大越好

例子

无

---

mobile.rsrq()

获取 rsrq,参考信号发送功率

参数

无

返回值

返回值类型	解释
int	当前 rsrq 值,若失败返回 0. 取值范围: -3 ~ -19.5 ， 值越大越好

例子

无

---

mobile.snr()

获取 snr,信噪比

参数

无

返回值

返回值类型	解释
int	当前 snq 值,若失败返回 0.范围 0 - 30, 越大越好

例子

无

---

mobile.eci()

获取当前服务小区的 ECI(E-UTRAN Cell Identifier)

参数

无

返回值

返回值类型	解释
int	当前 eci 值,若失败返回-1

例子

无

---

mobile.tac()

获取当前服务小区的 TAC 或者 LAC

参数

无

返回值

返回值类型	解释
int	当前 eci 值,若失败返回-1. 如果尚未注册到网络,会返回 0

例子

```
-- 本 API 于 2023.7.9 新增
```

---

mobile.enbid()

获取当前服务小区的 eNBID(eNodeB Identifier)

参数

无

返回值

返回值类型	解释
int	当前 enbid 值,若失败返回-1

例子

无

---

mobile.flymode(index, enable)

进出飞行模式

参数

传入值类型	解释
int	编号,默认 0. 在支持双卡的模块上才会出现 0 或 1 的情况
bool	是否设置为飞行模式,true 为设置, false 为退出,可选

返回值

返回值类型	解释
bool	原飞行模式的状态

例子

无

---

mobile.status()

获取网络状态

参数

无

返回值

返回值类型	解释
int	当前网络状态,0:网络未注册;1:网络已注册;2:正在搜网中;3:网络注册被拒绝

例子

无

---

mobile.getCellInfo()

获取机制信息

参数

无

返回值

返回值类型	解释
table	包含基站数据的数组

例子

```
-- 注意：从2023.06.20开始，需要主动请求一次reqCellInfo才会有基站数据.
--示例输出--[[
[
{"rsrq":-10,"rssi":-55,"cid":124045360,"mnc":17,"pci":115,"earfcn":1850,"snr":15,"rsrp":-85,"mcc":1120,"tdd":0},
{"pci":388,"rsrq":-11,"mnc":17,"earfcn":2452,"snr":5,"rsrp":-67,"mcc":1120,"cid":124045331},
{"pci":100,"rsrq":-9,"mnc":17,"earfcn":75,"snr":17,"rsrp":-109,"mcc":1120,"cid":227096712}
]
]]
mobile.reqCellInfo(60)
-- 订阅
sys.subscribe("CELL_INFO_UPDATE", function()
    log.info("cell", json.encode(mobile.getCellInfo()))
end)
-- 定期轮训式
sys.taskInit(function()
    sys.wait(3000)
```

```
    while 1 do
        mobile.reqCellInfo(15)
        sys.waitUntil("CELL_INFO_UPDATE", 15000)
        log.info("cell", json.encode(mobile.getCellInfo()))
    end
end)
```

---

mobile.reqCellInfo(timeout)

发起基站信息查询,含临近小区

参数

传入值类型	解释
int	超时时长,单位秒,默认 15. 最少 5, 最高 60

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 参考 mobile.getCellInfo 函数
```

---

mobile.reset()

重启协议栈

参数

无

返回值

无

例子

```
-- 重启 LTE 协议栈
mobile.reset()
```

---

mobile.dataTraffic(clearUplink, clearDownlink)

数据量流量处理

参数

传入值类型	解释
boolean	清空上行流量累计值，true 清空，其他忽略
boolean	清空下行流量累计值，true 清空，其他忽略

返回值

返回值类型	解释
int	上行流量 GB
int	上行流量 B
int	下行流量 GB
int	下行流量 B

例子

```
-- 获取上下行流量累计值
-- 上行流量值 Byte = uplinkGB * 1024 * 1024 * 1024 + uplinkB
-- 下行流量值 Byte = downlinkGB * 1024 * 1024 * 1024 + downlinkB
local uplinkGB, uplinkB, downlinkGB, downlinkB =
mobile.dataTraffic()
-- 清空上下行流量累计值
```

```
mobile.dataTraffic(true, true)
```

---

mobile.config(item, value)

网络特殊配置，针对不同平台有不同的配置，谨慎使用，目前只有 EC618

参数

传入值类型	解释
int	配置项目，看 mobile.CONF_XXX
int	配置值

返回值

返回值类型	解释
boolean	是否成功

例子

```
-- EC618 配置小区重选信号差值门限，不能大于 15dbm，必须在飞行模式下才能用 mobile.flymode(0,true)
mobile.config(mobile.CONF_RESELTOWEAKNCELL, 15)
mobile.config(mobile.CONF_STATICCONFIG, 1) --开启网络静态优化
mobile.flymode(0,false)
-- EC618 设置 SIM 写入次数的统计
-- 关闭统计
mobile.config(mobile.CONF_SIM_WC_MODE, 0)
-- 开启统计，默认也是开启的.
mobile.config(mobile.CONF_SIM_WC_MODE, 1)
-- 读取统计值,异步，需要通过系统消息 SIM_IND 获取
sys.subscribe("SIM_IND", function(stats, value)
    log.info("SIM_IND", stats)
    if stats == "SIM_WC" then
```

```
        log.info("sim", "write counter", value)
    end
end)
mobile.config(mobile.CONF_SIM_WC_MODE, 2)-- 清空统计值
mobile.config(mobile.CONF_SIM_WC_MODE, 3)
```

---

mobile.getBand(band, is_default)

获取当前使用/支持的 band

参数

传入值类型	解释
zbuff	输出 band
boolean	true 默认支持，false 当前支持的，默认是 false，当前是预留功能，不要写 true

返回值

返回值类型	解释
boolean	成功返回 true，失败放回 false

例子

```
local buff = zbuff.create(40)
mobile.getBand(buff) --输出当前使用的 band，band 号放在 buff 内，buff[0]，buff[1]，buff[2] .. buff[buff:used() - 1]
```

---

mobile.setBand(band, num)

设置使用的 band

参数

传入值类型	解释
zbuff	输入使用的 band
int	band 数量

返回值

返回值类型	解释
boolean	成功返回 true，失败放回 false

例子

```
local buff = zbuff.create(40)
buff[0] = 3
buff[1] = 5
buff[2] = 8
buff[3] = 40
mobile.setBand(buff, 4) --设置使用的 band 一共 4 个，为 3,5,8,40
```

---

mobile.nstOnOff(onoff, uart_id)

RF 测试开关和配置

参数

传入值类型	解释
boolean	true 开启测试模式，false 关闭
int	串口号

返回值

返回值类型	解释
nil	无返回值

例子

```
mobile.nstOnOff(true, uart.VUART_0)          --打开测试模式，并且用虚拟串口发送结果
mobile.nstOnOff(false) --关闭测试模式
```

---

mobile.nstInput(data)

RF 测试数据输入

参数

传入值类型	解释
string or zbuff	用户从串口获取的数据，注意，当获取完所有数据后，需要再传一个 nil 来作为传输结束

返回值

返回值类型	解释
nil	无返回值

例子

```
mobile.nstInput(uart_data)
mobile.nstInput(nil)
```

mqtt - mqtt 客户端

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 mqtt 的 demo 例子

示例

```
-- 具体用法请查看 demo

-- 本库支持的 mqtt 3.1.1，其他版本例如 3.1 或 5 是不支持的

-- 只支持纯 MQTT/MQTTS 通信，不支持 mqtt over websocket

-- 几个大前提:

-- 本库是基于 TCP 链接的，支持加密 TCP 和非加密 TCP

-- 任何通信失败都将断开连接，如果开启了自动重连，那么间隔 N 秒后开始自动重连

-- 上行数据均为一次性的，没有缓存机制，更没有上行的重试/重发机制

-- 如何获知发送成功: 触发 mqttc:on 中 event == "sent" 的事件

-- 关于 publish 时 QOS 值的说明，特制模块上行到云端/服务器端的行为:

-- QOS0，压入底层 TCP 发送堆栈,视为成功-- QOS1，收到服务器回应 PUBACK，视为成功-- QOS2，收到服务器响应 PUBREC,立即上行 PUBCOMP 压入 TCP 发送队列,视为成功

-- 重要的事情说 3 次: 没有重发机制，没有重发机制，没有重发机制

-- 1. MQTT 协议中规定了重发机制，但那是云端/服务器端才会实现的机制，模块端是没有的

-- 2. 上行失败，唯一的可能性就是 TCP 链接出问题了，而 TCP 链接出问题的解决办法就是重连

-- 3. 模块端不会保存任何上行数据，重连后也无法实现重发

-- 那业务需要确定上行是否成功，如何解决:

-- 首先推荐使用 QOS1，然后监听/判断 sent 事件,并选取一个超时时间，就能满足 99.9%的需求

-- 使用 QOS2,反而存在 PUBCOMP 上行失败导致服务器端不广播数据的理论可能
-- demo 里有演示等待 sent 事件的代码，类似于 sys.waitUntil("mqtt_sent", 3000) 搜 mqtt_sent 关键字
```

常量

mqttc:subscribe(topic, qos)

mqttc:unsubscribe(topic)

mqttc:debug(onoff)

mqtt.create(adapter,host,port,ssl,isipv6)

mqttc:auth(client_id,username,password,cleanSession)

mqttc:keepalive(time)

mqttc:on(cb)

mqttc:connect()

mqttc:disconnect()

mqttc:autoreconn(reconnect, reconnect_time)

mqttc:publish(topic, data, qos, retain)

mqttc:close()

mqttc:ready()

mqttc:state()

mqttc:will(topic, payload, qos, retain)

常量

常量	类型	解释
mqtt.STATE_DISCONNECT	number	mqtt 断开
mqtt.STATE_SCONNECT	number	mqtt socket 连接中
mqtt.STATE_MQTT	number	mqtt socket 已连接 mqtt 连接中
mqtt.STATE_READY	number	mqtt mqtt 已连接

mqttc:subscribe(topic, qos)

订阅主题

参数

传入值类型	解释
string/table	主题
int	topic 为 string 时生效 0/1/2 默认 0

返回值

返回值类型	解释

返回值类型	解释
int	消息 id,当 qos 为 1/2 时有效，若底层返回失败,会返回 nil

例子

```
-- 订阅单个 topic，且 qos=0
mqttc:subscribe("/luatos/123456", 0)
-- 订阅单个 topic，且 qos=1
mqttc:subscribe("/luatos/12345678", 1)
-- 订阅多个 topic，且使用不同的 qos
mqttc:subscribe({["/luatos/1234567"]=1,["/luatos/12345678"]=2})
```

---

mqttc:unsubscribe(topic)

取消订阅主题

参数

传入值类型	解释
string/table	主题

返回值

无

例子

```
mqttc:unsubscribe("/luatos/123456")
mqttc:unsubscribe({"/luatos/1234567","/luatos/12345678"})
```

---

mqttc:debug(onoff)

配置是否打开 debug 信息

参数

传入值类型	解释
boolean	是否打开 debug 开关

返回值

返回值类型	解释
nil	无返回值

例子

无

---

mqtt.create(adapter,host,port,ssl,isipv6)

mqtt 客户端创建

参数

传入值类型	解释
int	适配器序号，只能是 socket.ETH0, socket.STA, socket.AP,如果不填,会选择平台自带的方式,然后是最后一个注册的适配器
string	服务器地址,可以是域名，也可以是 ip
int	端口号
bool/table	是否为 ssl 加密连接,默认不加密,true 为无证书最简单的加密，table 为有证书的加密 server_cert 服务器 ca 证书数据 client_cert 客户端证书数据

传入值类型	解释
	client_key 客户端私钥加密数据
	client_password 客户端私钥口令数据
	verify 是否强制校验 0 不校验/1 可选校验/2 强制校验 默认 2
bool	是否为 ipv6 默认不是

返回值

返回值类型	解释
userdata	若成功会返回 mqtt 客户端实例,否则返回 nil

例子

```
-- 普通 TCP 链接
mqttc = mqtt.create(nil,"120.55.137.106", 1884)
-- 加密 TCP 链接,不验证服务器证书
mqttc = mqtt.create(nil,"120.55.137.106", 8883, true)
-- 加密 TCPTCP 链接,单服务器证书验证
mqttc = mqtt.create(nil,"120.55.137.106", 8883,
{server_cert=io.readFile("/luadb/ca.crt")})
-- 加密 TCPTCP 链接,单服务器证书验证, 但可选认证
mqttc = mqtt.create(nil,"120.55.137.106", 8883,
{server_cert=io.readFile("/luadb/ca.crt"), verify=1})
-- 加密 TCPTCP 链接,双向证书验证
mqttc = mqtt.create(nil,"120.55.137.106", 8883, {

        server_cert=io.readFile("/luadb/ca.crt"),

        client_cert=io.readFile("/luadb/client.pem"),

                                client_key="123456",
```

```
client_password="123456",
})
```

---

mqttc:auth(client_id,username,password,cleanSession)

mqtt 三元组配置及 cleanSession

参数

传入值类型	解释
string	设备识别 id,对于同一个 mqtt 服务器来说，通常要求唯一,相同 client_id 会互相踢下线
string	账号 可选
string	密码 可选
bool	清除 session,默认 true,可选

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 无账号密码登录,仅 clientId
mqttc:auth("123456789")
-- 带账号密码登录
mqttc:auth("123456789","username","password")
-- 额外配置 cleanSession,不清除
mqttc:auth("123456789","username","password", false)
-- 无 clientId 模式，服务器随机生成 id，cleanSession 不可配置
mqttc:auth()
```

---

mqttc:keepalive(time)

mqtt 心跳设置

参数

传入值类型	解释
int	可选 单位 s 默认 240s. 最先 15,最高 600

返回值

返回值类型	解释
nil	无返回值

例子

```
mqttc:keepalive(30)
```

---

mqttc:on(cb)

注册 mqtt 回调

参数

传入值类型	解释
function	cb mqtt 回调,参数包括 mqtt_client, event, data, payload

返回值

返回值类型	解释
nil	无返回值

例子

```
mqttc:on(function(mqtt_client, event, data, payload, metas)
        -- 用户自定义代码
        log.info("mqtt", "event", event, mqtt_client, data, payload)
end)
--[[
event 可能出现的值有  conack
-- 服务器鉴权完成,mqtt 连接已经建立，可以订阅和发布数据了,没有附加数据  recv
-- 接收到数据,由服务器下发，data 为 topic 值(string)，payload 为业务数据(string).metas 是元数据(table)，一般不处理.
-- metas 包含以下内容
-- qos 取值范围 0,1,2
-- retain 取值范围 0,1
-- dup 取值范围 0,1  sent
-- 发送完成，qos0 会马上通知，qos1/qos2 会在服务器应答会回调，data 为消息 id  disconnect
-- 服务器断开连接,网络问题或服务器踢了客户端,例如 clientId 重复,超时未上报业务数据
]]
```

---

mqttc:connect()

连接服务器

参数

无

返回值

返回值类型	解释

返回值类型	解释
boolean	发起成功返回 true，否则返回 false

例子

```
-- 开始建立连接
mqttc:connect()
-- 本函数仅代表发起成功，后续仍需根据 ready 函数判断 mqtt 是否连接正常
```

---

mqttc:disconnect()

断开服务器连接(不会释放资源)

参数

无

返回值

返回值类型	解释
boolean	发起成功返回 true，否则返回 false

例子

```
-- 断开连接
mqttc:disconnect()
```

---

mqttc:autoreconn(reconnect, reconnect_time)

自动重连

参数

传入值类型	解释

传入值类型	解释
bool	是否自动重连
int	自动重连周期 单位 ms 默认 3000ms

返回值

无

例子

```
mqttc:autoreconn(true)
```

---

mqttc:publish(topic, data, qos, retain)

发布消息

参数

传入值类型	解释
string	主题,必填
string	消息,必填,但长度可以是 0
int	消息级别 0/1 默认 0
int	是否存档, 0/1,默认 0

返回值

返回值类型	解释
int	消息 id，当 qos 为 1 或 2 时会有效值. 若底层返回是否，会返回 nil

例子

```
mqttc:publish("/luatos/123456", "123")
```

---

mqttc:close()

mqtt 客户端关闭(关闭后资源释放无法再使用)

参数

无

返回值

无

例子

```
mqttc:close()
```

---

mqttc:ready()

mqtt 客户端是否就绪

参数

无

返回值

返回值类型	解释
bool	客户端是否就绪

例子

```
local error = mqttc:ready()
```

---

mqttc:state()

mqtt 客户端状态

参数

无

返回值

返回值类型	解释
number	客户端状态

例子

```
local state = mqttc:state()
-- 已知状态:
-- 0: MQTT_STATE_DISCONNECT
-- 1: MQTT_STATE_CONNECTING
-- 2: MQTT_STATE_CONNECTED
-- 3: MQTT_STATE_READY
-- 4: MQTT_STATE_ERROR
```

---

mqttc:will(topic, payload, qos, retain)

设置遗嘱消息

参数

传入值类型	解释
string	遗嘱消息的 topic
string	遗嘱消息的 payload
string	遗嘱消息的 qos，默认 0，可以不填
string	遗嘱消息的 retain，默认 0，可以不填

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

```
-- 要在 connect 之前调用
mqttc:will("/xxx/xxx", "xxxxxx")
```

nes - nes 模拟器

已适配 Air101/Air103 Air601 Air105 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

nes.init(file_path)

nes.key(key,val)

nes.init(file_path)

nes 模拟器初始化

参数

传入值类型		解释
string	file_path	文件路径

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

```
nes.init("/luadb/super_mario.nes")
```

nes.key(key,val)

nes 模拟器初始化

参数

传入值类型		解释
number	key	按键
number	val	状态 1 按下 0 抬起

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

```
nes.init("/luadb/super_mario.nes")
```

os - os 操作

适配状态未知

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

os.remove(path)

os.rename(old_path, new_path)

os.clock()

os.date(fmt, time)

os.time(mytime)

os.difftime(timeA, timeB)

os.remove(path)

移除文件

参数

传入值类型	解释
string	待移除的文件完整路径

返回值

返回值类型	解释
bool	成功返回 true,其他情况返回 nil
string	失败时返回原因字符串

例子

```
-- 删除根目录下的某个文件
os.remove("/1.txt")
-- 注意，线刷时的文件，一般在 /luadb 目录，这个目录下的文件是只读的
-- 也就是无法执行 os.remove("/luadb/xxx.bin")
```

---

os.rename(old_path, new_path)

文件重命名

参数

传入值类型	解释
string	源文件完整路径
string	目标文件完整路径

返回值

返回值类型	解释
bool	成功返回 true,其他情况返回 nil
string	失败时返回原因字符串

例子

```
-- 注意，只有在相同文件系统下的文件可以重命名
-- 例如:
os.rename("/1.txt", "/2.txt")-- 不同文件系统，或者源文件系统是只读的，则无法执行
--os.rename("/luadb/1.txt", "/luadb/2.txt")
--os.rename("/luadb/1.txt", "/2.txt")
```

---

os.clock()

返回程序使用的按秒计 CPU 时间的近似值

参数

传入值类型	解释
return	时间戳

返回值

无

例子

```
-- 不推荐使用本 API
-- 如需要获取 时间戳，请使用 os.time()
-- 如需获取系统运行时长，请使用 mcu.ticks()
```

os.date(fmt, time)

日期函数

参数

传入值类型	解释
string	格式化字符串,可以是 nil
table	日期时间的 table

返回值

返回值类型	解释
table/string	根据 fmt 的不同,返回值不同

例子

```
-- 值得注意的几点:

-- 1. 若需要 UTC 时间, fmt 的第一个字符写"!"

-- 2. fmt 的格式化遵循 C 函数 strftime, 可以查阅 https://developer.aliyun.com/article/320480

-- 获取本地时间字符串

log.info("本地时间字符串", os.date())

-- 获取 UTC 时间字符串

log.info("UTC 时间字符串", os.date("!%c"))

-- 格式化本地时间字符串

log.info("本地时间字符串", os.date("%Y-%m-%d %H:%M:%S"))

-- 格式化 UTC 时间字符串 log.info("UTC 时间字符串", os.date("!%Y-%m-%d %H:%M:%S"))

-- 格式化时间字符串 log.info("自定义时间的字符串", os.date("!%Y-%m-%d %H:%M:%S", os.time({year=2000, mon=1, day=1, hour=0, min=0, sec=0})))
```

```
-- 获取本地时间的 table
log.info("本地时间字符串", json.encode(os.date("*t")))
-- 获取 UTC 时间的 table
log.info("UTC 时间字符串",  json.encode(os.date("!*t")))
```

---

os.time(mytime)

时间戳函数

参数

传入值类型	解释
table	日期时间的 table
return	时间戳

返回值

无

例子

```
-- 注意注意，这个函数返回的是 UTC 时间戳
-- 时间戳，但 lua 下的精度只能到秒
log.info("UTC 时间戳", os.time())
log.info("自定义时间戳", os.time({year=2000, mon=1, day=1, hour=0, min=0, sec=0}))
```

---

os.difftime(timeA, timeB)

时间差值

参数

传入值类型	解释
int	时间 A,数值类型
int	时间 B,数值类型

返回值

返回值类型	解释
int	时间差值

例子

无

---

otp - OTP 操作库

已适配 Air101/Air103 Air601 Air105 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 otp 的 demo 例子

```
otp.read(zone, offset, len)
otp.write(zone, data, offset)
otp.erase(zone)
otp.lock(zone)
```

otp.read(zone, offset, len)

读取指定 OTP 区域读取数据

参数

传入值类型	解释
int	区域，通常为 0/1/2/3，与具体硬件相关
int	偏移量
int	读取长度，单位字节，必须是 4 的倍数，不能超过 4096 字节

返回值

返回值类型	解释
string	成功返回字符串，否则返回 nil

例子

```
local otpdata = otp.read(0, 0, 64)
if otpdata then
    log.info("otp", otpdata:toHex())
end
```

---

otp.write(zone, data, offset)

往指定 OTP 区域写入数据

参数

传入值类型	解释
int	区域，通常为 0/1/2/3，与具体硬件相关
string	数据，长度必须是 4 个倍数
int	偏移量

返回值

返回值类型	解释
booL	成功返回 true,否则返回 false

例子

无

---

otp.erase(zone)

擦除指定 OTP 区域

参数

传入值类型	解释
int	区域，通常为 0/1/2/3，与具体硬件相关

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

无

---

otp.lock(zone)

锁定 OTP 区域. 特别注意!!一旦加锁即无法解锁,OTP 变成只读!!!

参数

无

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

无

pack - 打包和解包格式串

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库还有视频教程，点此链接查看

示例

```
--[[
'<' 设为小端编码
'>' 设为大端编码
'=' 大小端遵循本地设置
'z' 空字符串,0 字节
'a' size_t 字符串,前 4 字节表达长度,然后接着是 N 字节的数据
'A' 指定长度字符串, 例如 A8, 代表 8 字节的数据
'f' float, 4 字节
'd' double , 8 字节
'n' Lua number , 32bit 固件 4 字节, 64bit 固件 8 字节
'c' char , 1 字节
```

```
'b' byte = unsigned char  , 1 字节
'h' short  , 2 字节
'H' unsigned short  , 2 字节
'i' int  , 4 字节
'I' unsigned int , 4 字节
'l' long , 8 字节, 仅 64bit 固件能正确获取
'L' unsigned long , 8 字节, 仅 64bit 固件能正确获取
]]
-- 详细用法请查看 demo
```

pack.unpack(string, format, init)

pack.pack(format, val1, val2, val3, valn)

pack.unpack(string, format, init)

解包字符串

参数

传入值类型	解释
string	需解包的字符串
string	格式化符号
int	默认值为 1，标记解包开始的位置

返回值

返回值类型	解释
int	字符串标记的位置
any	第一个解包的值，根据 format 值,可能有 N 个返回值

例子

```
local _,a = pack.unpack(x,">h") --解包成 short (2 字节)
```

pack.pack(format, val1, val2, val3, valn)

打包字符串的值

参数

传入值类型	解释
string	format 格式化符号
any	第一个需打包的值
any	第二个需打包的值
any	第二个需打包的值
any	第 n 个需打包的值

返回值

返回值类型	解释
string	一个包含所有格式化变量的字符串

例子

```
local data = pack.pack('<h', crypto.crc16("MODBUS", val))
log.info("data", data, data:toHex())
```

pm - 电源管理

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 pm 的 demo 例子

示例

```
--[[休眠模式简介
-- IDLE 正常运行模式
-- LIGHT 轻睡眠模式:
        CPU 暂停
        RAM 保持供电
        定时器/网络事件/IO 中断均可自动唤醒
        唤醒后程序继续运行
        GPIO 保持电平
-- DEEP 深睡眠模式
        CPU 暂停
        核心 RAM 掉电，保留 RAM 维持供电
        普通 GPIO 掉电,外设驱动掉电
        AON_GPIO 保持休眠前的电平
        dtimer 定时器可唤醒
        wakeup 脚可唤醒
        唤醒后程序从头运行,休眠前的运行时数据全丢
-- HIB 休眠模式
        CPU 暂停
        RAM 掉电，保留 RAM 也掉电
        普通 GPIO 掉电,外设驱动掉电
        AON_GPIO 保持休眠前的电平
        dtimer 定时器可唤醒
        wakeup 脚可唤醒
        唤醒后程序从头运行,休眠前的运行时数据全丢
对部分模块,例如 Air780E，DEEP/HIB 对用户代码没有区别
```

除 pm.shutdown()外，RTC 总是运行的，除非掉电]]

-- 定时器唤醒，请使用 pm.dtimerStart()

-- wakeup 唤醒

-- 如 Air101/Air103，有独立的 wakeup 脚，不需要配置,可直接控制唤醒

-- 如 Air780E 系列，有多个 wakeup 可用，通过 gpio.setup(32)配置虚拟 GPIO 进行唤醒配置

pm.request(pm.IDLE) -- 通过切换不同的值请求进入不同的休眠模式

-- 对应 Air780E 系列，执行后并不一定马上进入休眠模式，如无后续数据传输需求,可先进入飞行模式,然后快速休眠

常量

pm.request(mode)

pm.dtimerStart(id, timeout)

pm.dtimerStop(id)

pm.dtimerCheck(id)

pm.dtimerWkId()

pm.lastReson()

pm.force(mode)

pm.check()

pm.shutdown()

pm.reboot()

pm.power(id, onoff)

pm.ioVol(id, val)

pm.wakeupPin(pin,level)

常量

常量	类型	解释
pm.NONE	number	不休眠模式
pm.IDLE	number	IDLE 模式
pm.LIGHT	number	LIGHT 模式

常量	类型	解释
pm.DEEP	number	DEEP 模式
pm.HIB	number	HIB 模式
pm.USB	number	USB 电源
pm.GPS	number	GPS 电源
pm.GPS_ANT	number	GPS 的天线电源，有源天线才需要
pm.CAMERA	number	camera 电源，CAM_VCC 输出
pm.DAC_EN	number	Air780E 和 Air600E 的 DAC_EN(新版硬件手册的 LDO_CTL，同一个 PIN，命名变更)，注意 audio 的默认配置会自动使用这个脚来控制 CODEC 的使能
pm.LDO_CTL	number	Air780E 和 Air600E 的 LDO_CTL(老版硬件手册的 DAC_EN，同一个 PIN，命名变更)，Air780EP 的 LDO_CTL，注意 audio 的默认配置会自动使用这个脚来控制 CODEC 的使能
pm.PWK_MODE	number	是否开启 ec618 的 powerkey 滤波模式，true 开，注意滤波模式下 reset 变成直接关机
pm.WORK_MODE	number	ec618 的节能模式，0~3，0 完全关闭，1 性能优先，2 平衡，3 极致功耗
pm.IOVL	number	所有 GPIO 高电平电压控制,当前仅 ec618 系列可用

pm.request(mode)

请求进入指定的休眠模式

参数

传入值类型	解释
int	休眠模式,例如 pm.IDLE/LIGHT/DEEP/HIB.

返回值

返回值类型　　解释

boolean　　处理结果,即使返回成功,也不一定会进入，也不会马上进入

例子

```
-- 请求进入休眠模式
--[[
IDLE    正常运行,就是无休眠
LIGHT   轻休眠，CPU 停止，RAM 保持，外设保持，可中断唤醒．部分型号支持从休眠处继续运行
DEEP    深休眠，CPU 停止，RAM 掉电，仅特殊引脚保持的休眠前的电平，大部分管脚不能唤醒设备．
HIB     彻底休眠，CPU 停止，RAM 掉电，仅复位/特殊唤醒管脚可唤醒设备．
]]
pm.request(pm.HIB)
```

---

pm.dtimerStart(id, timeout)

启动底层定时器,在休眠模式下依然生效．只触发一次

参数

传入值类型　　解释

int　　定时器 id,通常是 0-3

int　　定时时长,单位毫秒

返回值

返回值类型　　解释

boolean　　处理结果

例子

```
-- 添加底层定时器
pm.dtimerStart(0, 300 * 1000) -- 5 分钟后唤醒

-- Air780E/Air780EP 系列
-- id = 0 或者 id = 1 是，最大休眠时长是 2.5 小时
-- id = 2 或者 id = 3 是，最大休眠时长是 3 个月
```

---

pm.dtimerStop(id)

关闭底层定时器

参数

传入值类型	解释
int	定时器 id

返回值

无

例子

```
-- 关闭底层定时器
pm.dtimerStop(0) -- 关闭 id=0 的底层定时器
```

---

pm.dtimerCheck(id)

检查底层定时器是不是在运行

参数

传入值类型	解释
int	定时器 id

返回值

返回值类型	解释
boolean	处理结果,true 还在运行，false 不在运行
number	如果运行,运行剩余时间,单位毫秒(需 bsp 支持)

例子

```
-- 检查底层定时器是不是在运行
pm.dtimerCheck(0) -- 检查 id=0 的底层定时器
```

---

pm.dtimerWkId()

检查定时唤醒是哪一个定时器，如果不是定时唤醒的，返回-1

参数

无

返回值

返回值类型	解释
int	处理结果 >=0 是本次定时唤醒的定时器 ID，其他错误，说明不是定时唤醒的

例子

```
local timer_id = pm.dtimerWkId()
```

---

pm.lastReson()

开机原因,用于判断是从休眠模块开机,还是电源/复位开机

参数

无

返回值

返回值类型	解释
int	0-上电/复位开机, 1-RTC 开机, 2-WakeupIn/Pad/IO 开机, 3-Wakeup/RTC 开机
int	0-普通开机(上电/复位),3-深睡眠开机,4-休眠开机
int	复位开机详细原因：0-powerkey 或者上电开机 1-充电或者 AT 指令下载完成后开机 2-闹钟开机 3-软件重启 4-未知原因 5-RESET 键 6-异常重启 7-工具控制重启 8-内部看门狗重启 9-外部重启 10-充电开机

例子

```
-- 是哪种方式开机呢
log.info("pm", "last power reson", pm.lastReson())
```

---

pm.force(mode)

强制进入指定的休眠模式，忽略某些外设的影响，比如 USB

参数

传入值类型	解释
int	休眠模式

返回值

返回值类型	解释
boolean	处理结果,若返回成功,大概率会马上进入该休眠模式

例子

```
-- 请求进入休眠模式
pm.force(pm.HIB)
-- 对应 EC618 系列(Air780E/Air700E 等)，该操作会关闭 USB 通信
-- 唤醒后如需开启 USB，请打开 USB 电压
--pm.power(pm.USB, true)
```

---

pm.check()

检查休眠状态

参数

无

返回值

返回值类型	解释
boolean	处理结果,如果能顺利进入休眠,返回 true,否则返回 false
int	底层返回值,0 代表能进入最底层休眠,其他值代表最低可休眠级别

例子

```
-- 请求进入休眠模式,然后检查是否能真的休眠
pm.request(pm.HIB)
if pm.check() then
    log.info("pm", "it is ok to hib")
else
    -- 对应 EC618 系列(Air780E/Air700E 等)，该操作会关闭 USB 通信
    pm.force(pm.HIB) -- 强制休眠
    -- 唤醒后如需开启 USB，请打开 USB 电压
```

```
    --sys.wait(100)
    --pm.power(pm.USB, true)
end
```

---

pm.shutdown()

关机

参数

无

返回值

返回值类型　解释

nil　　无返回值

例子

```
-- 当前仅 EC618 系列(Air780E/Air600E/Air700E/Air780EG 支持)
-- 需要 2022-12-22 之后编译的固件
pm.shutdown()
```

---

pm.reboot()

重启

参数

无

返回值

返回值类型　解释

返回值类型　解释

nil	无返回值

例子

无

---

pm.power(id, onoff)

开启内部的电源控制，注意不是所有的平台都支持，可能部分平台支持部分选项，看硬件

参数

传入值类型	解释
int	电源控制 id,pm.USB pm.GPS 之类
boolean	or int 开关 true/1 开，false/0 关，默认关，部分选项支持数值

返回值

返回值类型	解释
boolean	处理结果 true 成功，false 失败

例子

```
-- 关闭 USB 电源，反之开启就是传 true
pm.power(pm.USB, false)
-- Air780EG,为内置的 GPS 芯片上电. 注意，Air780EG 的 GPS 和 GPS_ANT 是一起控制的,所以合并了.
pm.power(pm.GPS, true)
-- EC618 系列开启 pwrkey 开机防抖
-- 注意：开启后，复位键就变成关机了!!! pwrkey 要长按 2 秒才能开机
-- pm.power(pm.PWK_MODE, true)
```

```
-- EC618 系列 PSM+低功耗设置
-- ec618 的节能模式，0~3，0 完全关闭，1 性能优先，2 平衡，3 极致功耗
-- 详情访问: https://airpsm.cn
-- pm.power(pm.WORK_MODE, 1)
```

---

pm.ioVol(id, val)

IO 高电平电压控制,当前仅 EC618 系列可用

参数

传入值类型	解释
int	电平 id,目前只有 pm.IOVOL_ALL_GPIO
int	电平值,单位毫伏

返回值

返回值类型	解释
boolean	处理结果 true 成功，false 失败

例子

```
-- EC618 系列设置 IO 电平，范围 1650 ~ 2000，2650~3400 ，单位毫伏，步进 50mv
-- 例如 Air780E/Air600E/Air700E/Air780EG
-- 注意，这里的设置优先级会高于硬件 IOSEL 脚的配置
-- 但开机时依然先使用硬件配置,直至调用本 API 进行配置，所以 io 电平会变化
-- pm.ioVol(pm.IOVOL_ALL_GPIO, 3300)
-- 所有 GPIO 高电平输出 3.3V
-- pm.ioVol(pm.IOVOL_ALL_GPIO, 1800)
-- 所有 GPIO 高电平输出 1.8V
```

pm.wakeupPin(pin,level)

配置唤醒引脚 (当前仅仅 esp 系列可用)

参数

传入值类型	解释
int/table	gpio 引脚
int	唤醒电压 可选,默认低电平唤醒

返回值

返回值类型	解释
boolean	处理结果

例子

```
pm.wakeupPin(8,0)
```

protobuf - ProtoBuffs 编解码

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 protobuf 的 demo 例子

示例

```
-- 加载 pb 文件，这个是从 pbtxt 转换得到的
-- 转换命令: protoc.exe -operson.pb
```

```
--cpp_out=cpp person.pbtxt

-- protoc.exe 下载地址:
https://github.com/protocolbuffers/protobuf/releases

protobuf.load(io.readFile("/luadb/person.pb"))

local tb = {

    name = "wendal",

    id = 123,

    email = "abc@qq.com"

    }-- 用 protobuf 编码数据

    local data = protobuf.encode("Person", tb)

    if data then

    -- 打印数据长度. 编码后的数据含不可见字符, toHex 是方便显示

    log.info("protobuf", #data, (data:toHex()))

    end
```

protobuf.load(pbdata)

protobuf.clear()

protobuf.encode(tpname, data)

protobuf.decode(tpname, data)

protobuf.load(pbdata)

加载 pb 二进制定义数据

参数

传入值类型	解释
string	通过 protoc.exe 程序转换得到的数据,通常从文件读取得到

返回值

返回值类型	解释
bool	成功与否

返回值类型	解释
int	读取了多长，调试用

例子

```
-- 注意，同一个文件只需要加载一次，除非调用过 protobuf.clear()
protobuf.load(io.readFile("/luadb/person.pb"))
```

---

protobuf.clear()

清除已加载的二进制定义数据

参数

无

返回值

返回值类型	解释
nil	无返回值，总是成功

例子

```
-- 清除所有已加载的定义数据
protobuf.clear()
```

---

protobuf.encode(tpname, data)

编码 protobuffs 数据包

参数

传入值类型	解释
string	数据类型名称，定义在 pb 文件中，由 protobuf.load 加载

传入值类型	解释
table	待编码数据，必须是 table，内容符合 pb 文件里的定义

返回值

返回值类型	解释
string	编码后的数据,若失败会返回 nil

例子

```
-- 数据编码
local tb = {
    name = "wendal"}
    local pbdata = protobuf.encode("Person", tb)
    if pbdata then
    -- 注意，编码后的数据通常带不可见字符
    log.info("protobuf", #pbdata, pbdata:toHex())
    end
```

---

protobuf.decode(tpname, data)

解码 protobuffs 数据包

参数

传入值类型	解释
string	数据类型名称，定义在 pb 文件中，由 protobuf.load 加载
string	待编码数据

返回值

返回值类型	解释

返回值类型	解释
table	解码后的数据

例子

```
-- 数据编码 local tb = {
    name = "wendal"}
    local pbdata = protobuf.encode("Person", tb)
    if pbdata then
    -- 注意，编码后的数据通常带不可见字符
    log.info("protobuf", #pbdata, pbdata:toHex())
    end
```

pwm - PWM 模块

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 pwm 的 demo 例子

pwm.open(channel, period, pulse, pnum, precision)

pwm.close(channel)

pwm.capture(channel)

pwm.open(channel, period, pulse, pnum, precision)

开启指定的 PWM 通道

参数

传入值类型	解释
int	PWM 通道
int	频率, 1-1000000hz
int	占空比 0-分频精度
int	输出周期 0 为持续输出, 1 为单次输出，其他为指定脉冲数输出
int	分频精度, 100/256/1000，默认为 100，若设备不支持会有日志提示

返回值

返回值类型	解释
boolean	处理结果,成功返回 true,失败返回 false

例子

```
-- 打开 PWM5，频率 1kHz，占空比 50%
pwm.open(5, 1000, 50)-- 打开 PWM5，频率 10kHz，分频为 31/256
pwm.open(5, 10000, 31, 0, 256)
```

---

pwm.close(channel)

关闭指定的 PWM 通道

参数

传入值类型	解释
int	PWM 通道

返回值

返回值类型	解释
nil	无处理结果

例子

```
-- 关闭 PWM5
pwm.close(5)
```

---

pwm.capture(channel)

PWM 捕获

参数

传入值类型	解释
int	PWM 通道
int	捕获频率

返回值

返回值类型	解释
boolean	处理结果,成功返回 true,失败返回 false

例子

```
-- PWM0 捕获
while 1 do
    pwm.capture(0,1000)
    local ret,channel,pulse,pwmH,pwmL  =
sys.waitUntil("PWM_CAPTURE", 2000)
    if ret then

log.info("PWM_CAPTURE","channel"..channel,"pulse"..pulse,"pwmH".
.pwmH,"pwmL"..pwmL)
    end
end
```

repl - “读取-求值-输出”循环

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

示例

```
--[[
本功能支持的模块及对应的端口
模块/芯片          端口      波特率及其他参数
Air101/Air103    UART0   921600  8 None 1
Air105           UART0   1500000 8 None 1
ESP32C3          UART0   921600  8 None 1
-- 注意，简约版(无 CH343)不支持
ESP32C2          UART0   921600  8 None 1
ESP32S2          UART0   921600  8 None 1
Air780E          虚拟串口 任意              -- 暂不支持从物理 UART 调用
使用方法:
1. 非 Air780E 系列可以使用任意串口工具，打开对应的串口，记得勾选"回车换行"
2. Air780E 请配合 LuaTools 使用，菜单里有 "简易串口工具" 可发送，记得勾选"回车换行"
3. 发送 lua 语句，并以回车换行结束
语句支持情况:
1. 单行 lua 语句，以回车换行结束即可 2. 多行语句，用以下格式包裹起来发送，例如
<<EOF
for k,v in pairs(_G) do
print(k, v)
end
EOF
注意事项:
```

1. 可通过 repl.enable(false)语句禁用 REPL
2. 使用 uart.setup/uart.close 指定 UART 端口后， REPL 自动失效
3. 单行语句一般支持到 510 字节,更长的语句请使用"多行语句"的方式使用
4. 若需要定义全局变量，请使用 _G.xxx = yyy 形式

若有任何疑问，请到 chat.openluat.com 发帖反馈

]]

repl.enable(re)

repl.push(data)

repl.enable(re)

启用或禁用 REPL 功能

参数

传入值类型	解释
bool	启用与否,默认是启用
return	之前的设置状态

返回值

无

例子

```
-- 若固件支持 REPL,即编译时启用了 REPL,是默认启用 REPL 功能的
-- 本函数是提供关闭 REPL 的途径
repl.enable(false)
```

---

repl.push(data)

主动推送待处理的数据到底层

参数

传入值类型	解释
string	待处理的数据,通常从串口来

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 虚拟串口的设备才需要这个函数
```

rsa - RSA 加密解密

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 rsa 的 demo 例子

示例

```
-- 请在电脑上生成私钥和公钥，当前最高支持 4096bit，一般来说 2048bit 就够用了
-- openssl genrsa -out privkey.pem 2048
-- openssl rsa -in privkey.pem -pubout -out public.pem
-- privkey.pem 是私钥，public.pem 是公钥
-- 私钥用于 加密 和 签名，通常保密，放在服务器端
-- 公钥用于 解密 和 验签，一般可公开,放在设备端
-- 为了演示 API 使用，这里把私钥也放在设备上
```

```
local res = rsa.encrypt((io.readFile("/luadb/public.pem")), "abc")

-- 打印结果

log.info("rsa", "encrypt", res and #res or 0, res and res:toHex() or "")

-- 下面是解密，通常不会在设备端进行，这里主要是演示用法，会很慢

if res then

    -- 读取私钥，然后解码数据

    local dst = rsa.decrypt((io.readFile("/luadb/privkey.pem")), res, "")

    log.info("rsa", "decrypt", dst and #dst or 0, dst and dst:toHex() or "")

end

-- 演示签名和验签

local hash = crypto.sha1("1234567890"):fromHex()-- 签名通常很慢，通常是服务器做

local sig = rsa.sign((io.readFile("/luadb/privkey.pem")), rsa.MD_SHA1, hash, "")log.info("rsa", "sign", sig and #sig or 0, sig and sig:toHex() or "")

if sig then

-- 验签是很快的

    local ret = rsa.verify((io.readFile("/luadb/public.pem")), rsa.MD_SHA1, hash, sig)

    log.info("rsa", "verify", ret)

end
```

rsa.encrypt(key, data)

rsa.decrypt(key, data, pwd)

rsa.verify(key, md, hash, sig)

rsa.sign(key, md, hash, pwd)

rsa.encrypt(key, data)

RSA 加密

参数

传入值类型	解释
string	公钥数据,仅支持 PEM 格式
string	待加密数据，不能超过公钥位数的一半，例如 2048bit 的公钥，只能加密 128 字节的数据

返回值

返回值类型	解释
string	加密成功后的数据,若失败会返回 nil

例子

```
-- 下面代码中的 "abc" 是待加密数据
local res = rsa.encrypt((io.readFile("/luadb/public.pem")), "abc")
-- 打印结果
log.info("rsa", "encrypt", res and #res or 0, res and res:toHex() or "")
```

---

rsa.decrypt(key, data, pwd)

RSA 解密

参数

传入值类型	解释
string	私钥数据,仅支持 PEM 格式
string	待解密数据
string	私钥的密码,可选

返回值

返回值类型	解释
string	解密成功后的数据,若失败会返回 nil

例子

```
-- 注意，解密通常很慢，建议在服务器端进行
-- res 是待解密的数据
local dst = rsa.decrypt((io.readFile("/luadb/privkey.pem")), res, "")
log.info("rsa", "decrypt", dst and #dst or 0, dst and dst:toHex() or "")
```

---

rsa.verify(key, md, hash, sig)

RSA 验签

参数

传入值类型	解释
string	公钥数据,仅支持 PEM 格式
int	签名模式，例如 rsa.MD_SHA1 , rsa.MD_SHA256
string	hash 数据，如果是 HEX 字符串,记得 fromHex 转二进制数据
string	sig 数据，如果是 HEX 字符串,记得 fromHex 转二进制数据

返回值

返回值类型	解释
bool	有效返回 true,否则为 false,出错返回 nil

例子

```
local ret = rsa.verify((io.readFile("/luadb/public.pem")), rsa.MD_SHA1, hash, sig)

log.info("rsa", "verify", ret)
```

---

rsa.sign(key, md, hash, pwd)

RSA 签名

参数

传入值类型	解释
string	私钥数据,仅支持 PEM 格式
int	签名模式, 例如 rsa.MD_SHA1 , rsa.MD_SHA256
string	hash 数据, 如果是 HEX 字符串,记得 fromHex 转二进制数据
string	私钥密码, 可选

返回值

返回值类型	解释
string	成功返回 sig 数据, 否则返回 nil

例子

```
local sig = rsa.sign((io.readFile("/luadb/privkey.pem")), rsa.MD_SHA1, hash, "")

log.info("rsa", "sign", sig and #sig or 0, sig and sig:toHex() or "")
```

rtc - 实时时钟

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 rtc 的 demo 例子

rtc.set(tab)

rtc.get()

rtc.timerStart(id, tab)

rtc.timerStop(id)

rtc.setBaseYear(Base_year)

rtc.timezone(tz)

rtc.set(tab)

设置时钟

参数

传入值类型		解释
table	or int	时钟参数,见示例

返回值

返回值类型	解释
bool	成功返回 true,否则返回 nil 或 false

例子

```
rtc.set({year=2021,mon=8,day=31,hour=17,min=8,sec=43})--目前只有Air101/Air103/Air105/EC618 系列支持时间戳方式
rtc.set(1652230554)
```

---

rtc.get()

获取时钟

参数

无

返回值

返回值类型	解释
table	时钟参数,见示例

例子

```
local t = rtc.get()--
{year=2021,mon=8,day=31,hour=17,min=8,sec=43}
log.info("rtc", json.encode(t))
```

---

rtc.timerStart(id, tab)

设置 RTC 唤醒时间

参数

传入值类型	解释
int	时钟 id,通常只支持 0
table	时钟参数,见示例

返回值

返回值类型	解释
bool	成功返回 true,否则返回 nil 或 false

例子

```
-- 使用前建议先 rtc.set 设置为正确的时间
rtc.timerStart(0, {year=2021,mon=9,day=1,hour=17,min=8,sec=43})
```

---

rtc.timerStop(id)

取消 RTC 唤醒时间

参数

传入值类型	解释
int	时钟 id,通常只支持 0

返回值

返回值类型	解释
bool	成功返回 true,否则返回 nil 或 false

例子

```
rtc.timerStop(0)
```

---

rtc.setBaseYear(Base_year)

设置 RTC 基准年,不推荐

参数

传入值类型	解释
int	基准年 Base_year,通常 1900

返回值

无

例子

```
rtc.setBaseYear(1900)
```

---

rtc.timezone(tz)

读取或设置时区

参数

传入值类型	解释
int	时区值,注意单位是 1/4 时区.例如东八区是 32,而非 8. 可以不传
return	当前/设置后的时区值

返回值

无

例子

```
-- 设置为东 8 区 rtc.timezone(32)
-- 设置为东 3 区 rtc.timezone(12)
-- 设置为西 4 区 rtc.timezone(-16)
-- 注意：无论设置时区是多少，rtc.get/set 总是 UTC 时间
-- 时区影响的是 os.date/os.time 函数
-- 只有部分模块支持设置时区，且默认值一般为 32，即东八区
```

rtos - RTOS 底层操作库

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

rtos.receive(timeout)

rtos.timer_start(id,timeout,_repeat)

rtos.timer_stop(id)

rtos.reboot()

rtos.buildDate()

rtos.bsp()

rtos.version()

rtos.standy(timeout)

rtos.meminfo(type)

rtos.firmware()

rtos.setPaths(pathA, pathB, pathC, pathD)

rtos.nop()

rtos.autoCollectMem(period, warning_level, force_level)

rtos.receive(timeout)

接受并处理底层消息队列.

参数

传入值类型	解释
int	超时时长,通常是-1,永久等待

返回值

返回值类型	解释
msgid	如果是定时器消息,会返回定时器消息 id 及附加信息，其他消息由底层决定,不向 lua 层进行任何保证.

例子

无

---

rtos.timer_start(id,timeout,_repeat)

启动一个定时器

参数

传入值类型	解释
int	定时器 id
int	超时时长,单位毫秒
int	重复次数,默认是 0

返回值

返回值类型	解释
id	如果是定时器消息,会返回定时器消息 id 及附加信息，其他消息由底层决定,不向 lua 层进行任何保证.

例子

```
-- 用户代码请使用 sys.timerStart
-- 启动一个 3 秒的循环定时器
rtos.timer_start(10000, 3000, -1)
```

---

rtos.timer_stop(id)

关闭并释放一个定时器

参数

传入值类型	解释
int	定时器 id

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 用户代码请使用 sys.timerStop
rtos.timer_stop(id)
```

---

rtos.reboot()

设备重启

参数

无

返回值

返回值类型	解释
nil	无返回值

例子

无

---

rtos.buildDate()

获取固件编译日期

参数

无

返回值

返回值类型	解释
string	固件编译日期

例子

```
-- 获取编译日期
local d = rtos.buildDate()
```

---

rtos.bsp()

获取硬件 bsp 型号

参数

无

返回值

返回值类型	解释
string	硬件 bsp 型号

例子

```
-- 获取硬件 bsp 型号
local bsp = rtos.bsp()
```

---

rtos.version()

获取固件版本号

参数

无

返回值

返回值类型	解释
string	固件版本号,例如”V0001”

例子

```
-- 读取版本号
local luatos_version = rtos.version()
```

---

rtos.standy(timeout)

进入待机模式，仅部分设备可用，本 API 已废弃，推荐使用 pm 库

参数

传入值类型	解释
int	休眠时长,单位毫秒

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 进入待机模式
rtos.standby(5000)
```

---

rtos.meminfo(type)

获取内存信息

参数

传入值类型	解释
type	“sys”系统内存, “lua”虚拟机内存,”psram”psram 内存，默认为 lua 虚拟机内存

返回值

返回值类型	解释
int	总内存大小,单位字节
int	当前已使用的内存大小,单位字节
int	历史最高已使用的内存大小,单位字节

例子

```
-- 打印内存占用
log.info("mem.lua", rtos.meminfo())
log.info("mem.sys", rtos.meminfo("sys"))
```

---

rtos.firmware()

返回底层描述信息,格式为 LuatOS_$VERSION_$BSP,可用于 OTA 升级判断底层信息

参数

无

返回值

返回值类型	解释
string	底层描述信息

例子

```
-- 打印底层描述信息
log.info("firmware", rtos.firmware())
```

---

rtos.setPaths(pathA, pathB, pathC, pathD)

设置自定义 lua 脚本搜索路径,优先级高于内置路径

参数

传入值类型	解释
string	路径 A，例如“/sdcard/%s.luac”,若不传值,将默认为””,另外,最大长度不能超过 23 字节
string	路径 B，例如“/sdcard/%s.lua”
string	路径 C，例如“/lfs2/%s.luac”
string	路径 D，例如“/lfs2/%s.lua”

返回值

无

例子

```
-- 挂载 sd 卡或者 spiflash 后
rtos.setPaths("/sdcard/user/%s.luac", "/sdcard/user/%s.lua")
require("sd_user_main") -- 将搜索并加载 /sdcard/user/sd_user_main.luac 和 /sdcard/user/sd_user_main.lua
```

---

rtos.nop()

空函数,什么都不做

参数

无

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 这个函数单纯就是 lua -> c -> lua 走一遍
-- 没有参数,没有返回值,没有逻辑处理
-- 在绝大多数情况下,不会遇到这个函数的调用
-- 它通常只会出现在性能测试的代码里，因为它什么都不干.
rtos.nop()
```

---

rtos.autoCollectMem(period, warning_level, force_level)

内存自动收集配置，是 lua 本身收集机制的一种补充，不是必要的，而且只在 luavm 空闲时触发

参数

无

返回值

无

例子

无

sensor - 传感器操作库

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 sensor 的 demo 例子

示例

```
-- 请查阅 demo/dht11 demo/ds18b20
```

```
sensor.ds18b20(pin, check_crc)
sensor.w1_reset(pin)
sensor.w1_connect(pin)
sensor.w1_write(pin, data1,data2)
sensor.hx711(pin_date,pin_clk)
sensor.cs1237(pin_date,pin_clk)
sensor.ws2812b(pin,data,T0H,T0L,T1H,T1L)
sensor.ws2812b_pwm(pin,data)
sensor.ws2812b_spi(pin,data)
sensor.dht1x(pin)
sensor.sc12a(sda,scl)
```

sensor.ds18b20(pin, check_crc)

获取 DS18B20 的温度数据

参数

传入值类型	解释
int	gpio 端口号
boolean	是否校验 crc 值,默认为 true. 不校验 crc 值能提高读取成功的概率,但可能会读取到错误的值

返回值

返回值类型	解释
int	温度数据,单位 0.1 摄氏度，读取失败时返回错误码
boolean	成功返回 true,否则返回 false

例子

```
while 1 do
    sys.wait(5000)
    local val,result = sensor.ds18b20(17, true) -- GPIO17 且校验 CRC 值
    -- val 301 == 30.1 摄氏度
    -- result true 读取成功
    log.info("ds18b20", val, result)
end
```

---

sensor.w1_reset(pin)

单总线协议,复位设备

参数

传入值类型	解释
int	gpio 端口号

返回值

返回值类型	解释
nil	无返回

例子

无

---

sensor.w1_connect(pin)

单总线协议,连接设备

参数

传入值类型　　解释

int　　gpio 端口号

返回值

返回值类型　　解释

boolean　　成功返回 true,失败返回 false

例子

无

---

sensor.w1_write(pin, data1,data2)

单总线协议,往总线写入数据

参数

传入值类型　　解释

int　　gpio 端口号

int　　第一个数据

int　　第二个数据，可以写 N 个数据

返回值

返回值类型　　解释

nil　　无返回值

例子

无

---

sensor.w1_read(pin, len)

单总线协议,从总线读取数据

参数

传入值类型	解释
int	gpio 端口号
int	读取的长度

返回值

返回值类型	解释
int	按读取的长度返回 N 个整数

例子

无

---

sensor.hx711(pin_date,pin_clk)

获取 Hx711 的压力传感数据

参数

传入值类型	解释
int	数据的 gpio 端口号
int	时钟的 gpio 端口号

返回值

返回值类型	解释
int	hx711 读到的数据

例子

```
--  如果设备不存在会卡在读取接口
sys.taskInit(
    function()
        sys.wait(1000)
        local maopi = sensor.hx711(0,7)
        while true do
            sys.wait(2000)
            a = sensor.hx711(0,7) - maopi
            if a > 0 then
                log.info("tag", a / 4.6)
            end
        end
    end)
```

---

sensor.cs1237(pin_date,pin_clk)

获取 cs1237 传感数据

参数

传入值类型	解释
int	数据的 gpio 端口号
int	时钟的 gpio 端口号

返回值

返回值类型	解释
int	cs1237 读到的数据

例子

```
--  如果设备不存在会卡在读取接口
sys.taskInit(
    function()
        sys.wait(1000)
        local cs1237_data = sensor.cs1237(0,7)
        while true do
            sys.wait(2000)
            cs1237_data = sensor.cs1237(0,7) - maopi
            log.info("cs1237_data:", cs1237_data)--得到原始数据
        end
    end)
```

---

sensor.ws2812b(pin,data,T0H,T0L,T1H,T1L)

设置 ws2812b 输出(gpio 驱动方式)

参数

传入值类型	解释
int	ws2812b 的 gpio 端口号
string/zbuff	待发送的数据（如果为 zbuff 数据，则会无视指针位置始终从 0 偏移开始）
int	T0H 时间，表示延时多少个 nop，每个型号不一样，自己调
int	T0L 时间，表示延时多少个 nop
int	T1H 时间，表示延时多少个 nop
int	T1L 时间，表示延时多少个 nop

返回值

无

例子

```
local buff = zbuff.create({8,8,24})
buff:drawLine(1,2,5,6,0x00ffff)
sensor.ws2812b(7,buff,300,700,700,700)
```

---

sensor.ws2812b_pwm(pin,data)

设置 ws2812b 输出(pwm 驱动方式,需要 pwm 能输出 800k 频率，否则无法使用此方法)

参数

传入值类型	解释
int	pwm 端口号
string/zbuff	待发送的数据（如果为 zbuff 数据，则会无视指针位置始终从 0 偏移开始）

返回值

无

例子

```
local buff = zbuff.create({8,8,24})
buff:setFrameBuffer(8,8,24,0x0000ff)
sensor.ws2812b_pwm(7,buff)
```

---

sensor.ws2812b_spi(pin,data)

设置 ws2812b 输出(spi 驱动方式,需要 spi 能输出 5M 频率，否则无法使用此方法)

参数

传入值类型	解释
int	spi 端口号
string/zbuff	待发送的数据（如果为 zbuff 数据，则会无视指针位置始终从 0 偏移开始）

返回值

无

例子

```
local buff = zbuff.create({8,8,24})
buff:setFrameBuffer(8,8,24,0x0000ff)
sensor.ws2812b_spi(2,buff)
```

---

sensor.dht1x(pin)

获取 DHT11/DHT12 的温湿度数据

参数

传入值类型	解释
int	gpio 端口号
boolean	是否校验 crc 值,默认为 true. 不校验 crc 值能提高读取成功的概率,但可能会读取到错误的值

返回值

返回值类型	解释
int	湿度数据,单位 0.01%，读取失败时返回错误值
int	温度数据,单位 0.01 摄氏度，读取失败时返回错误值

返回值类型	解释
boolean	成功返回 true,否则返回 false

例子

```
while 1 do
    sys.wait(1000)
    local h,t,r = sensor.dht1x(17, true) -- GPIO17 且校验 CRC 值
    log.info("dht11", h/100,t/100,r)--90.1 23.22
end
```

---

sensor.sc12a(sda,scl)

获取 sc12a 被触摸的通道数据

参数

传入值类型	解释
int	数据的 gpio 端口号
int	时钟的 gpio 端口号

返回值

返回值类型	解释
int	读取成功返回整形数据，读取失败时返回错误值

例子

```
while true do
  local temp1=sensor.sc12a(4,7)
  if bit.rshift(bit.band( temp1, 0x8000), 15 )==0x01 then
    log.info("被按下的有通道 0")
  end
```

```
if bit.rshift(bit.band( temp1, 0x4000), 14 )==0x01 then
  log.info("被按下的有通道 1")
end
if bit.rshift(bit.band( temp1, 0x2000), 13 )==0x01 then
  log.info("被按下的有通道 2")
end
if bit.rshift(bit.band( temp1, 0x1000), 12 )==0x01 then
  log.info("被按下的有通道 3")
end
if bit.rshift(bit.band( temp1, 0x800), 11 )==0x01 then
  log.info("被按下的有通道 4")
end
if bit.rshift(bit.band( temp1, 0x400), 10 )==0x01 then
  log.info("被按下的有通道 5")
end
if bit.rshift(bit.band( temp1, 0x200), 9 )==0x01 then
  log.info("被按下的有通道 6")
end
if bit.rshift(bit.band( temp1, 0x100), 8 )==0x01 then
  log.info("被按下的有通道 7")
end
if bit.rshift(bit.band( temp1, 0x80), 7 )==0x01 then
  log.info("被按下的有通道 8")
end
if bit.rshift(bit.band( temp1, 0x40), 6 )==0x01 then
  log.info("被按下的有通道 9")
end
if bit.rshift(bit.band( temp1, 0x20), 5 )==0x01 then
  log.info("被按下的有通道 10")
```

```
  end
  if bit.rshift(bit.band( temp1, 0x10), 4 )==0x01 then
    log.info("被按下的有通道 11")
  end
  sys.wait(200)
end
```

sfd - SPI FLASH 操作库

已适配 Air105 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

sfd.init(type, spi_id, spi_cs)

sfd.status(drv)

sfd.read(drv, offset, len)

sfd.write(drv, offset, data)

sfd.erase(drv, offset)

sfd.id(drv)

sfd.init(type, spi_id, spi_cs)

初始化 spi flash

参数

传入值类型	解释
string	类型，可以是”spi”，也可以是”zbuff”，或者”onchip”
int	SPI 总线的 id，或者 zbuff 实例
int	SPI FLASH 的片选脚对应的 GPIO，当类型是 spi 时才需要传

返回值

返回值类型	解释
userdata	成功返回一个数据结构,否则返回 nil

例子

```
local drv = sfd.init("spi", 0, 17)if drv then
    log.info("sfd", "chip id", sfd.id(drv):toHex())
end-- 2023.01.15 之后的固件支持 onchip 类型，支持直接读写片上 flash 的一小块区域,一般是 64k
-- 这块区域通常是 fdb/fskv 库所在的区域，所以不要混着用
local onchip = sfd.init("onchip")
local data = sfd.read(onchip, 0x100, 256)
sfd.erase(onchip, 0x100)
sfd.write(onchip, 0x100, data or "Hi")
```

---

sfd.status(drv)

检查 spi flash 状态

参数

传入值类型	解释
userdata	sfd.init 返回的数据结构

返回值

返回值类型	解释
int	状态值, 0 未初始化成功,1 初始化成功且空闲,2 正忙

例子

```
local drv = sfd.init("spi", 0, 17)
if drv then
```

```
    log.info("sfd", "status", sfd.status(drv))
end
```

---

sfd.read(drv, offset, len)

读取数据

参数

传入值类型	解释
userdata	sfd.init 返回的数据结构
int	起始偏移量
int	读取长度,当前限制在 256 以内

返回值

返回值类型	解释
string	数据

例子

```
local drv = sfd.init("spi", 0, 17)
if drv then
    log.info("sfd", "read", sfd.read(drv, 0x100, 256))
end
```

---

sfd.write(drv, offset, data)

写入数据

参数

传入值类型	解释

传入值类型	解释
userdata	sfd.init 返回的数据结构
int	起始偏移量
string	需要写入的数据,当前支持 256 字节及以下

返回值

返回值类型	解释
boolean	成功返回 true,失败返回 false

例子

```
local drv = sfd.init("spi", 0, 17)
if drv then
    log.info("sfd", "write", sfd.write(drv, 0x100, "hi,luatos"))
end
```

---

sfd.erase(drv, offset)

擦除数据

参数

传入值类型	解释
userdata	sfd.init 返回的数据结构
int	起始偏移量

返回值

返回值类型	解释
boolean	成功返回 true,失败返回 false

例子

```
local drv = sfd.init("spi", 0, 17)
if drv then
    log.info("sfd", "write", sfd.erase(drv, 0x100))
end
```

---

sfd.id(drv)

芯片唯一 id

参数

传入值类型	解释
userdata	sfd.init 返回的数据结构

返回值

返回值类型	解释
string	8 字节(64bit)的芯片 id

例子

```
local drv = sfd.init("spi", 0, 17)
if drv then
    log.info("sfd", "chip id", sfd.id(drv))
end
```

sfud - SPI FLASH sfud 软件包

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 sfud 的 demo 例子

sfud.init(spi_id, spi_cs, spi_bandrate)/sfud.init(spi_device)

sfud.getDeviceNum()

sfud.getDevice(index)

sfud.getDeviceTable()

sfud.chipErase(flash)

sfud.erase(flash,add,size)

sfud.read(flash, addr, size)

sfud.write(flash, addr,data)

sfud.eraseWrite(flash, addr,data)

sfud.getInfo(flash)

sfud.mount(flash, mount_point, offset, maxsize)

sfud.init(spi_id, spi_cs, spi_bandrate)/sfud.init(spi_device)

初始化 sfud

参数

传入值类型	解释
int	spi_id SPI 的 ID/userdata spi_device
int	spi_cs SPI 的片选
int	spi_bandrate SPI 的频率

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

```
--spi
log.info("sfud.init",sfud.init(0,20,20 * 1000 * 1000))
```

```
--spi_device
local spi_device =
spi.deviceSetup(0,17,0,0,8,2000000,spi.MSB,1,0)
log.info("sfud.init",sfud.init(spi_device))
```

---

sfud.getDeviceNum()

获取 flash 设备信息表中的设备总数

参数

无

返回值

返回值类型	解释
int	返回设备总数

例子

```
log.info("sfud.getDeviceNum",sfud.getDeviceNum())
```

---

sfud.getDevice(index)

通过 flash 信息表中的索引获取 flash 设备

参数

传入值类型	解释
int	index flash 信息表中的索引

返回值

返回值类型	解释

返回值类型	解释
userdata	成功返回一个数据结构,否则返回 nil

例子

```
local sfud_device = sfud.getDevice(1)
```

---

sfud.getDeviceTable()

获取 flash 设备信息表

参数

无

返回值

返回值类型	解释
userdata	成功返回一个数据结构,否则返回 nil

例子

```
local sfud_device = sfud.getDeviceTable()
```

---

sfud.chipErase(flash)

擦除 Flash 全部数据

参数

传入值类型	解释
userdata	flash Flash 设备对象 sfud.get_device_table()返回的数据结构

返回值

返回值类型	解释
int	成功返回 0

例子

```
sfud.chipErase(flash)
```

---

sfud.erase(flash,add,size)

擦除 Flash 指定地址指定大小，按照 flash block 大小进行擦除

参数

传入值类型		解释
userdata	flash Flash	设备对象 sfud.get_device_table()返回的数据结构
number	add	擦除地址
number	size	擦除大小

返回值

返回值类型	解释
int	成功返回 0

例子

```
sfud.erase(flash,add,size)
```

---

sfud.read(flash, addr, size)

读取 Flash 数据

参数

传入值类型	解释

传入值类型		解释
userdata	flash Flash	设备对象 sfud.get_device_table()返回的数据结构
int	addr	起始地址
int	size	从起始地址开始读取数据的总大小

返回值

返回值类型		解释
string	data	读取到的数据

例子

```
log.info("sfud.read",sfud.read(sfud_device,1024,4))
```

---

sfud.write(flash, addr,data)

向 Flash 写数据

参数

传入值类型		解释
userdata	flash Flash	设备对象 sfud.get_device_table()返回的数据结构
int	addr	起始地址
string	data	待写入的数据

返回值

返回值类型	解释
int	成功返回 0

例子

```
log.info("sfud.write",sfud.write(sfud_device,1024,"sfud"))
```

---

sfud.eraseWrite(flash, addr,data)

先擦除再往 Flash 写数据

参数

传入值类型		解释
userdata	flash Flash 设备对象	sfud.get_device_table()返回的数据结构
int	addr 起始地址	
string	data 待写入的数据	

返回值

返回值类型	解释
int	成功返回 0

例子

```
log.info("sfud.eraseWrite",sfud.eraseWrite(sfud_device,1024,"sfud"))
```

---

sfud.getInfo(flash)

获取 Flash 容量和 page 大小

参数

传入值类型		解释
userdata	flash Flash 设备对象	sfud.get_device_table()返回的数据结构

返回值

返回值类型	解释

返回值类型	解释
int	Flash 容量
int	page 页大小

例子

```
log.info("sfud.getInfo",sfud.getInfo(sfud_device))
```

---

sfud.mount(flash, mount_point, offset, maxsize)

挂载 sfud lfs 文件系统

参数

传入值类型	解释
userdata	flash Flash 设备对象 sfud.get_device_table()返回的数据结构
string	mount_point 挂载目录名
int	起始偏移量,默认 0
int	总大小, 默认是整个 flash

返回值

返回值类型	解释
bool	成功返回 true

例子

```
log.info("sfud.mount",sfud.mount(sfud_device,"/sfud"))
log.info("fsstat", fs.fsstat("/"))
log.info("fsstat", fs.fsstat("/sfud"))
```

sms - 短信

已适配 Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 sms 的 demo 例子

示例

```
-- 注意，Air780E/Air600E/Air780EG/Air780EG 均不支持电信卡的短信!!
-- 意思是，当上述模块搭配电信 SIM 卡，无法从模块发出短信，也无法在模块接收短信
-- 如果是联通卡或者移动卡，均可收取短信，但实名制的卡才能发送短信

sms.send(phone, msg, auto_phone_fix)
sms.setNewSmsCb(func)
sms.autoLong(mode)
sms.clearLong()
```

sms.send(phone, msg, auto_phone_fix)

发送短信

参数

传入值类型	解释
string	电话号码,必填
string	短信内容,必填
bool	是否自动处理电话号号码的格式,默认是按短信内容和号码格式进行自动判断，设置为 false 可禁用

返回值

返回值类型 **解释**

bool 成功返回 true,否则返回 false 或 nil

例子

无

---

sms.setNewSmsCb(func)

设置新 SMS 的回调函数

参数

传入值类型 **解释**

function 回调函数, 3 个参数, num, txt, metas

返回值

返回值类型 **解释**

nil 传入是函数就能成功,无返回值

例子

```
sms.setNewSmsCb(function(num, txt, metas)
    -- num 手机号码
    -- txt 文本内容
    -- metas 短信的元数据,例如发送的时间,长短信编号
    -- 注意, 长短信会自动合并成一条 txt
    log.info("sms", num, txt, metas and json.encode(metas) or "")
    end)
```

---

sms.autoLong(mode)

设置长短信的自动合并功能

参数

传入值类型	解释
bool	是否自动合并,true 为自动合并,为默认值

返回值

返回值类型	解释
bool	设置后的值

例子

```
-- 禁用长短信的自动合并，一般不需要禁用
sms.autoLong(false)
```

---

sms.clearLong()

清除长短信缓存

参数

无

返回值

返回值类型	解释
int	清理掉的片段数量

例子

```
sms.clearLong()
```

socket - 网络接口

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

小技巧

本库有专属 demo，点此链接查看 socket 的 demo 例子

常量

socket.sntp(sntp_server)

socket.ntptm()

socket.localIP(adapter)

socket.create(adapter, cb)

socket.debug(ctrl, onoff)

socket.config(ctrl, local_port, is_udp, is_tls, keep_idle, keep_interval, keep_cnt, server_cert, client_cert, client_key, client_password)

socket.linkup(ctrl)

socket.connect(ctrl, ip, remote_port, need_ipv6_dns)

socket.discon(ctrl)

socket.close(ctrl)

socket.tx(ctrl, data, ip, port, flag)

socket.rx(ctrl, buff, flag, limit)

socket.wait(ctrl)

socket.listen(ctrl)

socket.accept(ctrl)

socket.release(ctrl)

socket.setDNS(adapter_index, dns_index, ip)

socket.sslLog(log_level)

socket.adapter(index)

socket.remoteIP(ctrl)

常量

常量	类型	解释
socket.ETH0	number	带硬件协议栈的 ETH0，值为 5

常量	类型	解释
socket.LWIP_ETH	number	使用 LWIP 协议栈的以太网卡，值为 4
socket.LWIP_STA	number	使用 LWIP 协议栈的 WIFI STA，值为 2
socket.LWIP_AP	number	使用 LWIP 协议栈的 WIFI AP，值为 3
socket.LWIP_GP	number	使用 LWIP 协议栈的移动蜂窝模块，值为 1
socket.USB	number	使用 LWIP 协议栈的 USB 网卡，值为 6
socket.LINK	number	LINK 事件
socket.ON_LINE	number	ON_LINE 事件
socket.EVENT	number	EVENT 事件
socket.TX_OK	number	TX_OK 事件
socket.CLOSED	number	CLOSED 事件

socket.sntp(sntp_server)

本接口仅支持 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

sntp 时间同步

参数

传入值类型		解释
string/table		sntp 服务器地址 选填
int		适配器序号， 只能是 socket.ETH0（外置以太网），socket.LWIP_ETH（内置以太网），socket.LWIP_STA（内置 WIFI 的 STA），socket.LWIP_AP（内置 WIFI 的 AP），socket.LWIP_GP（内置蜂窝网络的 GPRS），socket.USB（外置 USB 网卡），如果不填，优先选择 soc 平台自带能上外网的适配器，若仍然没有，选择最后一个注册的适配器

返回值

无

例子

```
socket.sntp()

--socket.sntp("ntp.aliyun.com")

--自定义 sntp 服务器地址
--socket.sntp({"ntp.aliyun.com","ntp1.aliyun.com","ntp2.aliyun.
com"})

--sntp 自定义服务器地址

--socket.sntp(nil, socket.ETH0)

--sntp 自定义适配器序号

sys.subscribe("NTP_UPDATE", function()

    log.info("sntp", "time", os.date())

    end)

    sys.subscribe("NTP_ERROR", function()

    log.info("socket", "sntp error")

    socket.sntp()

    end)
```

---

socket.ntptm()

网络对时后的时间戳(ms 级别)

参数

无

返回值

返回值类型	解释
table	包含时间信息的数据

例子

```
-- 本 API 于 2023.11.15 新增

-- 注意，本函数在执行 socket.sntp()且获取到 NTP 时间后才有效
```

```
-- 而且是 2 次 sntp 之后才是比较准确的值
-- 网络波动越小，该时间戳越稳定
local tm = socket.ntptm()
-- 对应的 table 包含多个数据，均为整数值
-- 标准数据
-- tsec 当前秒数,从 1900.1.1 0:0:0 开始算，UTC 时间
-- tms  当前毫秒数
-- vaild 是否有效，true 或者 nil
-- 调试数据，调试用,一般用户不用管
-- ndelay 网络延时平均值,单位毫秒
-- ssec 系统启动时刻与 1900.1.1 0:0:0 的秒数偏移量
-- sms 系统启动时刻与 1900.1.1 0:0:0 的毫秒偏移量
-- lsec 本地秒数计数器,基于 mcu.tick64()
-- lms 本地毫秒数计数器,基于 mcu.tick64()
log.info("tm 数据", json.encode(tm))
log.info("时间戳", string.format("%u.%03d", tm.tsec, tm.tms))
```

---

socket.localIP(adapter)

获取本地 ip

参数

传入

值类型	**解释**
int	适配器序号， 只能是 socket.ETH0（外置以太网），socket.LWIP_ETH（内置以太网），socket.LWIP_STA（内置 WIFI 的 STA），socket.LWIP_AP（内置 WIFI 的 AP），socket.LWIP_GP（内置蜂窝网络的 GPRS），socket.USB（外置 USB 网卡），如果不填，优先选择 soc 平台自带能上外网的适配器，若

传入

值类型	解释
	仍然没有，选择最后一个注册的适配器

返回值

返回值类型	解释
string	通常是内网 ip，也可能是外网 ip，取决于运营商的分配
string	网络掩码
string	网关 IP

例子

```
sys.taskInit(function()
    while 1 do
        sys.wait(3000)
        log.info("socket", "ip", socket.localIP())
            -- 输出示例
            -- 62.39.244.10      255.255.255.255      0.0.0.0
    end
end)
```

---

socket.create(adapter, cb)

在某个适配的网卡上申请一个 socket_ctrl

参数

传入值类型	解释
int	适配器序号， 只能是 socket.ETH0（外置以太网），socket.LWIP_ETH（内

传入值

类型	解释
	置以太网），socket.LWIP_STA（内置 WIFI 的 STA），socket.LWIP_AP（内置 WIFI 的 AP），socket.LWIP_GP（内置蜂窝网络的 GPRS），socket.USB（外置 USB 网卡），如果不填，优先选择 soc 平台自带能上外网的适配器，若仍然没有，选择最后一个注册的适配器
string	or function string 为消息通知的 taskName，function 则为回调函数，如果固件没有内置 sys_wait，则必须是 function

返回值

无

例子

无

---

socket.debug(ctrl, onoff)

配置是否打开 debug 信息

参数

传入值类型	解释
user_data	socket.create 得到的 ctrl
boolean	true 打开 debug 开关

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 打开调试信息,默认是关闭状态
socket.debug(ctrl, true)
```

---

socket.config(ctrl, local_port, is_udp, is_tls, keep_idle, keep_interval, keep_cnt, server_cert, client_cert, client_key, client_password)

配置 network 一些信息，

参数

传入值类型	解释
user_data	socket.create 得到的 ctrl
int	本地端口号，小端格式，如果不写，则自动分配一个，如果用户填了端口号则需要小于 60000, 默认不写
boolean	是否是 UDP，默认 false
boolean	是否是加密传输，默认 false
int	tcp keep live 模式下的 idle 时间（秒），如果留空则表示不启用，如果是不支持标准 posix 接口的网卡（比如 W5500），则为心跳间隔
int	tcp keep live 模式下的探测间隔时间（秒）
int	tcp keep live 模式下的探测次数
string	TCP 模式下的服务器 ca 证书数据，UDP 模式下的 PSK，不需要加密传输写 nil，后续参数也全部 nil
string	TCP 模式下的客户端 ca 证书数据，UDP 模式下的 PSK-ID，TCP 模式下如果不需要验证客户端证书时，忽略，一般不需要验证客户端证书
string	TCP 模式下的客户端私钥加密数据
string	TCP 模式下的客户端私钥口令数据

返回值

返回值类型	解释
boolean	成功返回 true，失败返回 false

例子

```
--最普通的 TCP 传输

socket.config(ctrl)--最普通的加密 TCP 传输，证书都不用验证的那种
socket.config(ctrl, nil, nil ,true)
```

---

socket.linkup(ctrl)

等待网卡 linkup

参数

传入值类型	解释
user_data	socket.create 得到的 ctrl

返回值

返回值类型	解释
boolean	true 没有异常发生，false 失败了，如果 false 则不需要看下一个返回值了
boolean	true 已经 linkup，false 没有 linkup，之后需要接收 socket.LINK 消息

例子

```
-- 判断一下是否已经联网

local succ, result = socket.linkup(ctrl)
```

---

socket.connect(ctrl, ip, remote_port, need_ipv6_dns)

作为客户端连接服务器

参数

传入值类型	解释
user_data	socket.create 得到的 ctrl
string	or int ip 或者域名，如果是 IPV4，可以是大端格式的 int 值
int	服务器端口号，小端格式
boolean	域名解析是否要 IPV6，true 要，false 不要，默认 false 不要，只有支持 IPV6 的协议栈才有效果

返回值

返回值类型	解释
boolean	true 没有异常发生，false 失败了，如果 false 则不需要看下一个返回值了，如果有异常，后续要 close
boolean	true 已经 connect，false 没有 connect，之后需要接收 socket.ON_LINE 消息

例子

```
local succ, result = socket.connect(ctrl, "netlab.luatos.com", 40123)
--[[
常见的连接失败的 code 值，会在日志中显示-1 底层内存不足-3 超时-8 端口已经被占用-11 链接未建立-13 模块主动断开连接-14 服务器主动断开连接
]]
```

socket.discon(ctrl)

作为客户端断开连接

参数

传入值类型	解释
user_data	socket.create 得到的 ctrl

返回值

返回值类型	解释
boolean	true 没有异常发生，false 失败了，如果 false 则不需要看下一个返回值了
boolean	true 已经断开，false 没有断开，之后需要接收 socket.CLOSED 消息

例子

```
local succ, result = socket.discon(ctrl)
```

---

socket.close(ctrl)

强制关闭 socket

参数

传入值类型	解释
user_data	socket.create 得到的 ctrl

返回值

返回值类型	解释
nil	无返回值

例子

无

---

socket.tx(ctrl, data, ip, port, flag)

发送数据给对端，UDP 单次发送不要超过 1460 字节，否则很容易失败

参数

传入值类型	解释
user_data	socket.create 得到的 ctrl
string	or user_data zbuff 要发送的数据
string	or int 对端 IP，如果是 TCP 应用则忽略，如果是 UDP，如果留空则用 connect 时候的参数，如果是 IPV4，可以是大端格式的 int 值
int	对端端口号，小端格式，如果是 TCP 应用则忽略，如果是 UDP，如果留空则用 connect 时候的参数
int	发送参数，目前预留，不起作用

返回值

返回值类型	解释
boolean	true 没有异常发生，false 失败了，如果 false 则不需要看下一个返回值了，如果 false，后续要 close
boolean	true 缓冲区满了，false 没有满，如果 true，则需要等待一段时间或者等到 socket.TX_OK 消息后再尝试发送，同时忽略下一个返回值
boolean	true 已经收到应答，false 没有收到应答，之后需要接收 socket.TX_OK 消息， 也可以忽略继续发送，直到 full==true

例子

```
local succ, full, result = socket.tx(ctrl, "123456",
"xxx.xxx.xxx.xxx", xxxx)
```

---

socket.rx(ctrl, buff, flag, limit)

接收对端发出的数据，注意数据已经缓存在底层，使用本函数只是提取出来，UDP 模式下一次只会取出一个数据包

参数

传入值类型	解释
user_data	socket.create 得到的 ctrl
user_data	zbuff 存放接收的数据，如果缓冲区不够大会自动扩容
int	接收参数，目前预留，不起作用
int	接收数据长度限制，如果指定了，则只取前 N 个字节. 2024.1.5 新增

返回值

返回值类型	解释
boolean	true 没有异常发生，false 失败了，如果 false 则不需要看下一个返回值了，如果 false，后续要 close
int	本次接收到数据长度
string	对端 IP，只有 UDP 模式下才有意义，TCP 模式返回 nil，注意返回的格式，如果是 IPV4，1byte 0x00 + 4byte 地址 如果是 IPV6，1byte 0x01 + 16byte 地址
int	对端 port，只有 UDP 模式下才有意义，TCP 模式返回 0

例子

```
-- 从 socket 中读取数据，ctrl 是 socket.create 返回的，请查阅 demo/socket

local buff = zbuff.create(2048)

local succ, data_len, remote_ip, remote_port = socket.rx(ctrl, buff)

-- 限制读取长度，2024.1.5 新增

-- 注意

-- 如果是 UDP 数据，如果 limit 小于 UDP 数据包长度，只会取前 limit 个字节，剩余数据会丢弃

-- 如果是 TCP 数据，如果有剩余数据，不会丢弃，可继续读取.

-- 有新的数据到来才会有新的 EVENT 数据，未读取完成的数据不会触发新 EVENT 事件

local succ, data_len, remote_ip, remote_port = socket.rx(ctrl, buff, 1500)

-- 读取缓冲区大小，2024.1.5 新增，注意,老版本固件不传 buff 参数会报错的

-- 对于 TCP 数据，这里返回的是待读取的数据的总长度

-- 对于 UDP 数据，这里返回的是单个 UDP 数据包的长度

local succ, data_len = socket.rx(ctrl)

if succ then

        log.info("待收取数据长度", data_len)

end
```

---

socket.wait(ctrl)

等待新的 socket 消息，在连接成功和发送数据成功后，使用一次将 network 状态转换到接收新数据

参数

传入值类型	解释
user_data	socket.create 得到的 ctrl

返回值

返回值类型	解释
boolean	true 没有异常发生，false 失败了，如果 false 则不需要看下一个返回值了，如果 false，后续要 close
boolean	true 有新的数据需要接收，false 没有数据，之后需要接收 socket.EVENT 消息

例子

```
local succ, result = socket.wait(ctrl)
```

---

socket.listen(ctrl)

作为服务端开始监听

参数

传入值类型	解释
user_data	socket.create 得到的 ctrl

返回值

返回值类型	解释
boolean	true 没有异常发生，false 失败了，如果 false 则不需要看下一个返回值了，如果 false，后续要 close
boolean	true 已经 connect，false 没有 connect，之后需要接收 socket.ON_LINE 消息

例子

```
local succ, result = socket.listen(ctrl)
```

socket.accept(ctrl)

作为服务端接收到一个新的客户端，注意，如果是类似 W5500 的硬件协议栈不支持 1 对多，则不需要第二个参数

参数

传入值类型	解释
user_data	socket.create 得到的 ctrl，这里是服务器端
string	or function or nil string 为消息通知的 taskName，function 则为回调函数，和 socket.create 参数一致

返回值

返回值类型	解释
boolean	true 没有异常发生，false 失败了，如果 false 则不需要看下一个返回值了，如果 false，后续要 close
user_data	or nil 如果支持 1 对多，则会返回新的 ctrl，自动 create，如果不支持则返回 nil

例子

```
local succ, new_netc = socket.listen(ctrl, cb)
```

socket.release(ctrl)

主动释放掉 network_ctrl

参数

无

返回值

无

例子

无

---

socket.setDNS(adapter_index, dns_index, ip)

设置 DNS 服务器

参数

传入值类型	解释
int	适配器序号， 只能是 socket.ETH0，socket.STA，socket.AP，如果不填，会选择最后一个注册的适配器
int	dns 服务器序号，从 1 开始
string	or int dns，如果是 IPV4，可以是大端格式的 int 值

返回值

返回值类型	解释
boolean	成功返回 true，失败返回 false

例子

```
-- 设置默认网络适配器的 DNS 配置
socket.setDNS(nil, 1, "114.114.114.114")
-- 设置制定网络适配器的 DNS 配置
```

```
socket.setDNS(socket.ETH0, 1, "114.114.114.114")
```

---

socket.sslLog(log_level)

设置 SSL 的 log 登记

参数

传入值类型　解释

int mbedtls　log 等级

返回值

返回值类型　解释

nil　　无返回值

例子

```
--[[
SSL/TLS log 级别说明 0 不打印 1 只打印错误和警 2 大部分 info3 及 3 以上详细的 debug
过多的信息可能会造成内存碎片化
]]
-- 打印大部分 info 日志 socket.sslLog(2)
```

---

socket.adapter(index)

查看网卡适配器的联网状态

参数

传入

值类型	解释
int	需要查看的适配器序号，可以留空会查看全部网卡，直到遇到 IP READY 的，如果指定网卡，只能是 socket.ETH0（外置以太网），socket.LWIP_ETH（内置以太网），socket.LWIP_STA（内置 WIFI 的 STA），socket.LWIP_AP（内置 WIFI 的 AP），socket.LWIP_GP（内置蜂窝网络的 GPRS），socket.USB（外置 USB 网卡）

返回值

返回值类型	解释
boolean	被查看的适配器是否 IP READY,true 表示已经准备好可以联网了,false 暂时不可以联网
int	最后一个被查看的适配器序号

例子

```
-- 查看全部网卡，直到找到一个是 IP READY 的
local isReady,index = socket.adapter()
--如果 isReady 为 true,则 index 为 IP READY 的网卡适配器序号
--查看外置以太网（比如 W5500）是否 IP READY
local isReady,default = socket.adapter(socket.ETH0)
```

---

socket.remoteIP(ctrl)

获取对端 ip

参数

传入值类型	解释

传入值类型	解释
user_data	socket.create 得到的 ctrl

返回值

返回值类型	解释
string	IP1，如果为 nil，则表示没有获取到 IP 地址
string	IP2，如果为 nil，则表示没有 IP2
string	IP3，如果为 nil，则表示没有 IP3
string	IP4，如果为 nil，则表示没有 IP4

例子

```
-- 注意：，必须在接收到 socket.ON_LINE 消息之后才可能获取到，最多返回 4 个 IP。

-- socket.connect 里如果 remote_port 设置成 0，则当 DNS 完成时就返回 socket.ON_LINE 消息

local ip1,ip2,ip3,ip4 = socket.remoteIP(ctrl)
```

spi - spi 操作库

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 spi 的 demo 例子

小技巧

本库还有视频教程，点此链接查看

常量

spi.setup(id, cs, CPHA, CPOL, dataw, bandrate, bitdict, ms, mode)

spi.createSoft(cs, mosi, miso, clk, CPHA, CPOL, dataw, bitdict, ms, mode)

spi.close(id)

spi.transfer(id, send_data, send_len, recv_len)

spi.recv(id, size)

spi.send(id, data[, len])

spi.deviceSetup(id, cs, CPHA, CPOL, dataw, bandrate, bitdict, ms, mode)

spi_device:close()

spi_device:transfer(send_data[, len])

spi_device:send(data[, len])

spi_device:recv(size)

spi.xfer(id, txbuff, rxbuff, rx_len, transfer_done_topic)

常量

常量	类型	解释
spi.MSB	number	大端模式
spi.LSB	number	小端模式
spi.master	number	主机模式
spi.slave	number	从机模式
spi.full	number	全双工
spi.half	number	半双工
spi.SPI_0	number	SPI0
spi.SPI_1	number	SPI1
spi.SPI_2	number	SPI2
spi.SPI_3	number	SPI3
spi.SPI_4	number	SPI4
spi.HSPI_0	number	高速 SPI0，目前 105 专用

spi.setup(id, cs, CPHA, CPOL, dataw, bandrate, bitdict, ms, mode)

设置并启用 SPI

参数

传入值类型	解释
int	SPI 号,例如 0
int	CS 片选脚,在 w600 不可用请填 nil
int	CPHA 默认 0,可选 0/1
int	CPOL 默认 0,可选 0/1
int	数据宽度,默认 8bit
int	波特率,默认 2M=2000000
int	大小端，默认 spi.MSB，可选 spi.LSB
int	主从设置，默认主 1，可选从机 0. 通常只支持主机模式
int	工作模式，全双工 1，半双工 0，默认全双工

返回值

返回值类型	解释
int	成功返回 0,否则返回其他值

例子

```
-- 初始化 spi
spi.setup(0,20,0,0,8,2000000,spi.MSB,1,1)
```

---

spi.createSoft(cs, mosi, miso, clk, CPHA, CPOL, dataw, bitdict, ms, mode)

设置并启用软件 SPI

参数

传入值类型	解释

传入值类型	解释
int	cs 引脚编号，传入 nil 意为 Lua 控制 cs 脚
int	mosi 引脚编号
int	miso 引脚编号
int	clk 引脚编号
int	默认 0，可选 0/1
int	默认 0，可选 0/1
int	数据宽度，默认 8bit
int	大小端，默认 spi.MSB，可选 spi.LSB
int	主从设置，默认主 1，可选从机 0. 通常只支持主机模式
int	工作模式，全双工 1，半双工 0，默认半双工

返回值

返回值类型	解释
软件 SPI 对象	可当作 SPI 的 id 使用

例子

```
-- 初始化软件 spi
local softSpiDevice = spi.createSoft(0, 1, 2, 3, 0, 0, 8, spi.MSB, 1, 1)
local result = spi.send(softSpiDevice, string.char(0x9f))
```

---

spi.close(id)

关闭指定的 SPI

参数

传入值类型	解释

传入值类型　　解释

int　　SPI 号,例如 0

返回值

返回值类型　　解释

int　　成功返回 0,否则返回其他值

例子

```
-- 初始化 spi
spi.close(0)
```

---

spi.transfer(id, send_data, send_len, recv_len)

传输 SPI 数据

参数

传入值类型　　解释

int　　SPI 号(例如 0)或软件 SPI 对象

string/zbuff　　待发送的数据，如果为 zbuff 数据，则会从对象所处的指针处开始读

int　　可选。待发送数据的长度，默认为 data 长度

int　　可选。读取数据的长度，默认为 1

返回值

返回值类型　　解释

string　　读取成功返回字符串,否则返回 nil

例子

```
-- 初始化 spi
```

```
spi.setup(0,nil,0,0,8,2000000,spi.MSB,1,1)

local recv = spi.transfer(0, "123")--发送 123,并读取数据

local buff = zbuff.create(1024, 0x33) --创建一个初值全为 0x33 的内存区域

local recv = spi.transfer(0, buff)--把 zbuff 数据从指针开始，全发出去,并读取数据
```

---

spi.recv(id, size)

接收指定长度的 SPI 数据

参数

传入值类型	解释
int	SPI 号,例如 0
int	数据长度

返回值

返回值类型	解释
string	读取成功返回字符串,否则返回 nil

例子

```
-- 初始化 spi
spi.setup(0,nil,0,0,8,2000000,spi.MSB,1,1)
local recv = spi.recv(0, 4)--接收 4 字节数据
```

---

spi.send(id, data[, len])

发送 SPI 数据

参数

传入值类型	解释
int	SPI 号,例如 0
string/zbuff	待发送的数据，如果为 zbuff 数据，则会从对象所处的指针处开始读
int	可选。待发送数据的长度，默认为 data 长度

返回值

返回值类型	解释
int	发送结果

例子

```
-- 初始化 spi
spi.setup(0,nil,0,0,8,2000000,spi.MSB,1,1)
local result = spi.send(0, "123")--发送 123
local buff = zbuff.create(1024, 0x33) --创建一个初值全为 0x33 的内存区域
local result = spi.send(0, buff)--把 zbuff 数据从指针开始，全发出去
```

---

spi.deviceSetup(id, cs, CPHA, CPOL, dataw, bandrate, bitdict, ms, mode)

设置并启用 SPI(对象方式)

参数

传入值类型	解释
int	SPI 号,例如 0
int	CS 片选脚,在 w600 不可用请填 nil
int	CPHA 默认 0,可选 0/1
int	CPOL 默认 0,可选 0/1

传入值类型	解释
int	数据宽度,默认 8bit
int	波特率,默认 20M=20000000
int	大小端，默认 spi.MSB，可选 spi.LSB
int	主从设置，默认主 1，可选从机 0. 通常只支持主机模式
int	工作模式，全双工 1，半双工 0，默认全双工

返回值

返回值类型	解释
userdata	spi_device

例子

```
-- 初始化 spi
local spi_device =
spi.deviceSetup(0,17,0,0,8,2000000,spi.MSB,1,1)
```

---

spi_device:close()

关闭指定的 SPI(对象方式)

参数

传入值类型	解释
userdata	spi_device

返回值

返回值类型	解释
int	成功返回 0,否则返回其他值

例子

```
-- 初始化 spi
spi_device.close()
```

---

spi_device:transfer(send_data[, len])

传输 SPI 数据(对象方式)

参数

传入值类型	解释
userdata	spi_device
string/zbuff	待发送的数据，如果为 zbuff 数据，则会从对象所处的指针处开始读
int	可选。待发送数据的长度，默认为 data 长度
int	可选。读取数据的长度，默认为 1

返回值

返回值类型	解释
string	读取成功返回字符串,否则返回 nil

例子

```
-- 初始化 spi
local spi_device =
spi.device_setup(0,17,0,0,8,2000000,spi.MSB,1,1)

local recv = spi_device:transfer("123")--发送 123,并读取数据

local buff = zbuff.create(1024, 0x33) --创建一个初值全为 0x33 的内存区域

local recv = spi_device:transfer(buff)--把 zbuff 数据从指针开始，全发出去,并读取数据
```

---

spi_device:send(data[, len])

发送 SPI 数据(对象方式)

参数

传入值类型	解释
userdata	spi_device
string/zbuff	待发送的数据，如果为 zbuff 数据，则会从对象所处的指针处开始读

返回值

返回值类型	解释
int	发送结果

例子

```
-- 初始化 spi
local spi_device =
spi.device_setup(0,17,0,0,8,2000000,spi.MSB,1,1)
local result = spi_device:send("123")--发送 123
local buff = zbuff.create(1024, 0x33) --创建一个初值全为 0x33 的内存区域
local result = spi_device:send(buff)--把 zbuff 数据从指针开始，全发出去
```

---

spi_device:recv(size)

接收指定长度的 SPI 数据(对象方式)

参数

传入值类型	解释
userdata	spi_device

传入值类型　解释

int	数据长度

返回值

返回值类型　　解释

string	读取成功返回字符串,否则返回 nil

例子

```
-- 初始化 spi
local spi_device =
spi.device_setup(0,17,0,0,8,2000000,spi.MSB,1,1)
local recv = spi_device:recv(4)--接收 4 字节数据
```

---

spi.xfer(id, txbuff, rxbuff, rx_len, transfer_done_topic)

非阻塞方式硬件 SPI 传输 SPI 数据，目的为了提高核心利用率。API 直接返回是否启动传输，传输完成后通过 topic 回调，本 API 适合硬件 SPI 传输大量数据传输，外设功能（LCD SPI，W5500 SPI 之类的）占据的 SPI 和软件 SPI 不能用，少量数据传输建议使用传统阻塞型 API

参数

传入值类型	解释
userdata	or int spi_device 或者 spi_id，注意，如果是 spi_device，需要手动在传输完成后拉高 cs!!!!!!
zbuff	待发送的数据，如果为 nil，则只接收数据，由于用的非阻塞模型，为保证动态数据的有效性，只能使用 zbuff，发送的数据从 zbuff.addr
zbuff	接收数据，如果为 nil，则只发送数据，由于用的非阻塞模型，为保证动态数据的有效性，只能使用 zbuff，接收的数据从 zbuff.addr 开始存

传入值类型

	解释
	储
int	传输数据长度，特别说明 如果为半双工，先发后收，比如 spi flash 操作这种，则长度=发送字节+接收字节，注意上面发送和接收 buff 都要留足够的数据，后续接收数据处理需要跳过发送数据长度字节
string	传输完成后回调的 topic

返回值

返回值类型

	解释
boolean	true/false 本次传输是否正确启动，true，启动，false，有错误无法启动。传输完成会发布消息 transfer_done_topic 和 boolean 型结果

例子

```
local result = spi.xfer(spi.SPI_0, txbuff, rxbuff, 1024, "SPIDONE")
if result then
        result, spi_id, succ, error_code = sys.waitUntil("SPIDONE")
end
if not result or not succ then
        log.info("spi fail, error code", error_code)
 else
        log.info("spi ok")
end
```

sqlite3.open(path)

sqlite3.exec(db, sql)

sqlite3.close(db)

sqlite3 - sqlite3 数据库操作

已适配 Air101/Air103 Air601 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 sqlite3 的 demo 例子

示例

```
-- 注意，本库仍处于开发阶段，大部分 BSP 尚不支持本库-- 本移植基于 sqlite3 3.44.0
sys.taskInit(function()
    sys.wait(1000)
    local db = sqlite3.open("/ram/test.db")
    log.info("sqlite3", db)
    if db then
        sqlite3.exec(db, "CREATE TABLE devs(ID INT PRIMARY KEY NOT NULL, name CHAR(50));")
        sqlite3.exec(db, "insert into devs values(1, \"ABC\");")
        sqlite3.exec(db, "insert into devs values(2, \"DEF\");")
        sqlite3.exec(db, "insert into devs values(3, \"HIJ\");")
        local ret, data = sqlite3.exec(db, "select * from devs;")
        log.info("查询结果", ret, data)
        if ret then
            for k, v in pairs(data) do
                log.info("数据", json.encode(v))
            end
        end
```

```
        sqlite3.close(db)
    end
end)
```

sqlite3.open(path)

打开数据库

参数

传入值类型	解释
string	数据库文件路径,必须填写,不存在就会自动新建

返回值

返回值类型	解释
userdata	数据库指针,是否就返回 nil

例子

```
local db = sqlite3.open("/test.db")
if db then
   -- 数据库操作 xxxx

    -- 用完必须关掉
    sqlite3.close(db)
end
```

---

sqlite3.exec(db, sql)

执行 SQL 语句

参数

传入值类型	解释

传入值类型	解释
userdata	通过 sqlite3.open 获取到的数据库指针
string	SQL 字符串,必须填写

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 nil
table	成功返回查询结果(若有),否则返回报错的字符串

例子

无

---

sqlite3.close(db)

关闭数据库

参数

传入值类型	解释
userdata	通过 sqlite3.open 获取到的数据库指针

返回值

返回值类型	解释
boolean	成功返回 true,否则返回 nil

例子

无

statem - SM 状态机

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 statem 的 demo 例子

statem.create(count, repeat)

statem.create(count, repeat)

创建一个新的状态机.

参数

传入值类型	解释
int	指令条数,默认 32 条
int	重复执行的次数,0 代表不重复，正整数代表具体重复执行的次数. 暂不支持永续执行

返回值

返回值类型	解释
some	若成功,返回状态机指针,否则返回 nil

例子

```
gpio.setup(7, 0, gpio.PULLUP)
gpio.setup(12, 0, gpio.PULLUP)
gpio.setup(13, 0, gpio.PULLUP)
```

```
gpio.setup(14, 0, gpio.PULLUP)
local sm =  statem.create()
            :gpio_set(7, 0) -- gpio 设置为低电平
            :usleep(10)     -- 休眠 10us
            :gpio_set(7, 1) -- gpio 设置为高电平
            :usleep(40)     -- 休眠 40us
            :gpio_set(12, 1) -- gpio 设置为高电平
            :gpio_set(13, 1) -- gpio 设置为高电平
            :gpio_set(14, 1) -- gpio 设置为高电平
            :usleep(40)      -- 休眠 40us
            :gpio_set(7, 0) -- gpio 设置为低电平
            :finish()
-- 执行之,后续会支持后台执行 sm:exec()
```

string - 字符串操作函数

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 string 的 demo 例子

string.toHex(str, separator)

string.fromHex(hex)

string.split(str, delimiter, keepEmtry)

string.toValue(str)

string.urlEncode(“123 abc”)

string.toBase64(str)

string.fromBase64(str)

string.toBase32(str)

string.fromBase32(str)

string.startsWith(str, prefix)

string.endsWith(str, suffix)

string.trim(str, ltrim, rtrim)

string.toHex(str, separator)

将字符串转成 HEX

参数

传入值类型	解释
string	需要转换的字符串
string	分隔符，默认为””

返回值

返回值类型	解释
string	HEX 字符串
number	HEX 字符串的长度

例子

```
string.toHex("\1\2\3") --> "010203" 6
string.toHex("123abc") --> "313233616263" 12
string.toHex("123abc", " ") --> "31 32 33 61 62 63 " 12
```

---

string.fromHex(hex)

将 HEX 转成字符串

参数

传入值类型	解释
string	hex,16 进制组成的串

返回值

返回值类型 解释

string	字符串

例子

```
string.fromHex("010203")       -->  "\1\2\3"
string.fromHex("313233616263") -->  "123abc"
```

---

string.split(str, delimiter, keepEmtry)

按照指定分隔符分割字符串

参数

传入值类型	解释
string	输入字符串
string	分隔符,可选,默认 “,”
bool	是否保留空白片段,默认为 false,不保留. 2023.4.11 之后的固件可用

返回值

返回值类型	解释
table	分割后的字符串表

例子

```
local tmp = string.split("123,233333,122")
log.info("tmp", json.encode(tmp))
local tmp = ("123,456,789"):split(',') --> {'123','456','789'}
log.info("tmp", json.encode(tmp))
-- 保留空片段, 2023.4.11 之后的固件可用
```

```
local str = "/tmp//def/1234/"
local tmp = str:split("/", true)
log.info("str.split", #tmp, json.encode(tmp))
```

---

string.toValue(str)

返回字符串 tonumber 的转义字符串(用来支持超过 31 位整数的转换)

参数

传入值类型	解释
string	输入字符串

返回值

返回值类型	解释
string	转换后的二进制字符串
number	转换了多少个字符

例子

```
string.toValue("123456") --> "\1\2\3\4\5\6"  6
string.toValue("123abc") --> "\1\2\3\a\b\c"  6
```

---

string.urlEncode(“123 abc”)

将字符串进行 url 编码转换

参数

传入值类型	解释
string	需要转换的字符串

返回值

无

例子

无

---

string.toBase64(str)

将字符串进行 base64 编码

参数

传入值类型	解释
string	需要转换的字符串

返回值

返回值类型	解释
string	解码后的字符串,如果解码失败会返回空字符串

例子

无

---

string.fromBase64(str)

将字符串进行 base64 解码

参数

传入值类型	解释
string	需要转换的字符串

返回值

返回值类型	解释
string	解码后的字符串,如果解码失败会返回空字符串

例子

无

---

string.toBase32(str)

将字符串进行 base32 编码

参数

传入值类型	解释
string	需要转换的字符串

返回值

返回值类型	解释
string	解码后的字符串,如果解码失败会返回 0 长度字符串

例子

无

---

string.fromBase32(str)

将字符串进行 base32 解码

参数

传入值类型	解释
string	需要转换的字符串

返回值

返回值类型	解释
string	解码后的字符串,如果解码失败会返回 0 长度字符串

例子

无

---

string.startsWith(str, prefix)

判断字符串前缀

参数

传入值类型	解释
string	需要检查的字符串
string	前缀字符串

返回值

返回值类型	解释
bool	真为 true，假为 false

例子

```
local str = "abc"
log.info("str", str:startsWith("a"))
log.info("str", str:startsWith("b"))
```

---

string.endsWith(str, suffix)

判断字符串后缀

参数

传入值类型	解释
string	需要检查的字符串
string	后缀字符串

返回值

返回值类型	解释
bool	真为 true，假为 false

例子

```
local str = "abc"
log.info("str", str:endsWith("c"))
log.info("str", str:endsWith("b"))
```

---

string.trim(str, ltrim, rtrim)

裁剪字符串,去除头尾的空格

参数

传入值类型	解释
string	需要处理的字符串
bool	清理前缀,默认为 true
bool	清理后缀,默认为 true

返回值

返回值类型	解释
string	清理后的字符串

例子

```
local str = "\r\nabc\r\n"

log.info("str", string.trim(str)) -- 打印 "abc"

log.info("str", str:trim())       -- 打印 "abc"

log.info("str", #string.trim(str, false, true)) -- 仅裁剪后缀,所以长度是 5
```

sys - sys 库

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库还有视频教程，点此链接查看

sys.wait(timeout)
sys.waitUntil(topic, timeout)
sys.taskInit(func, arg1, arg2, argN)
sys.timerStart(func, timeout, arg1, arg2, argN)
sys.timerLoopStart(func, timeout, arg1, arg2, argN)
sys.timerStop(id)
sys.timerStopAll(fnc)
sys.publish(topic, arg1, agr2, argN)
sys.subscribe(topic, func)
sys.unsubscribe(topic, func)
sys.run()

sys.wait(timeout)

Task 协程等待指定时长

参数

传入值类型 **解释**

int 等待时长,单位毫秒,必须大于 0,否则无效

返回值

返回值类型 **解释**

any 通常为 nil,除非主动被唤醒(通常不会)

例子

```
sys.taskInit(function()
    while 1 do
        sys.wait(500)
    endend)
```

---

sys.waitUntil(topic, timeout)

Task 协程等待指定时长或者特定的 topic

参数

传入值类型 **解释**

string 事件 topic

int 等待时长,单位毫秒,必须大于 0,否则无效

返回值

返回值类型 **解释**

boolean 如果是超时,返回 false,否则返回 true

any 对应 topic 的内容

例子

```
sys.taskInit(function()
```

```
    // do something
    local result, data = sys.waitUntil("NET_READY", 30000)
    // do something else
end)
```

---

sys.taskInit(func, arg1, arg2, argN)

创建一个 Task 协程

参数

传入值类型	解释
function	待执行的函数,可以是匿名函数，也可以是 local 或全局函数
any	需要传递的参数 1,可选
any	需要传递的参数 2,可选
any	需要传递的参数 N,可选

返回值

返回值类型	解释
task	协程对象

例子

```
sys.taskInit(function(a, b, c)
    log.info("task", a, b, c) -- 打印 task A B C
end, "A", "B", "N")
```

---

sys.timerStart(func, timeout, arg1, arg2, argN)

创建一个定时器.非 Task,函数里不能直接 sys.waitXXX

参数

传入值类型	解释
function	待执行的函数,可以是匿名函数，也可以是 local 或全局函数
int	延时时长,单位毫秒
any	需要传递的参数 1,可选
any	需要传递的参数 2,可选
any	需要传递的参数 N,可选

返回值

返回值类型	解释
int	定时器 id

例子

```
sys.timerStart(function(a, b, c)
    log.info("task", a, b, c) -- 1000 毫秒后才会执行，打印 task A B C
    end, 1000, "A", "B", "N")
```

---

sys.timerLoopStart(func, timeout, arg1, arg2, argN)

创建一个循环定时器.非 Task,函数里不能直接 sys.waitXXX

参数

传入值类型	解释
function	待执行的函数,可以是匿名函数，也可以是 local 或全局函数
int	延时时长,单位毫秒
any	需要传递的参数 1,可选
any	需要传递的参数 2,可选
any	需要传递的参数 N,可选

返回值

返回值类型　解释

int　　定时器 id

例子

```
sys.timerLoopStart(function(a, b, c)
    log.info("task", a, b, c) -- 1000 毫秒后才会执行, 打印 task A B C
    end, 1000, "A", "B", "N")
```

---

sys.timerStop(id)

关闭一个定时器.

参数

传入值类型　解释

int　　定时器 id

返回值

返回值类型　解释

nil　　无返回值

例子

```
local tcount = 0local tid tid = sys.timerLoopStart(function(a, b, c)
    log.info("task", a, b, c) -- 1000 毫秒后才会执行, 打印 task A B C
    if tcount > 10 then
        sys.timerStop(tid)
    end
```

```
    tcount = tcount + 1
end, 1000, "A", "B", "N")
```

---

sys.timerStopAll(fnc)

关闭同一回调函数的所有定时器.

参数

传入值类型	解释
function	fnc 回调的函数

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 关闭回调函数为 publicTimerCbFnc 的所有定时器
local function publicTimerCbFnc(tag)
       log.info("publicTimerCbFnc",tag)
end
sys.timerStart(publicTimerCbFnc,8000,"first")
sys.timerStart(publicTimerCbFnc,8000,"second")
sys.timerStart(publicTimerCbFnc,8000,"third")
sys.timerStopAll(publicTimerCbFnc)
```

---

sys.publish(topic, arg1, agr2, argN)

往特定 topic 通道发布一个消息

参数

传入值类型	解释
string	topic 的值
any	附带的参数 1
any	附带的参数 2
any	附带的参数 N

返回值

返回值类型	解释
nil	无返回值

例子

```
sys.publish("BT_READY", false)
```

---

sys.subscribe(topic, func)

订阅一个 topic 通道

参数

传入值类型	解释
string	topic 的值
function	回调函数，注意，不能直接使用 sys.waitXXX

返回值

返回值类型	解释
nil	无返回值

例子

```
local function bt_cb(state)
```

```
    log.info("bt", state)
end
    sys.subscribe("BT_READY", bt_cb)
    sys.subscribe("BT_READY", function(state)
        log.info("sys", "Got BT_READY", state)
    end)
```

---

sys.unsubscribe(topic, func)

取消订阅 topic 通道

参数

传入值类型	解释
string	topic 的值
function	回调函数，注意，不能直接使用 sys.waitXXX

返回值

返回值类型	解释
nil	无返回值

例子

```
local function bt_cb(state)
    log.info("bt", state)
End
sys.unsubscribe("BT_READY", bt_cb)
```

---

sys.run()

sys 库主循环方法,仅允许在 main.lua 的末尾调用

参数

无

返回值

返回值类型	解释
nil	无返回值. 这个方法几乎不可能返回.

例子

```
-- 总是 main.lua 的结尾一句,将来也许会简化掉
sys.run()
-- 之后的代码不会被执行
```

sysplus - sys 库的强力补充

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

示例

```
-- 本库是 sys 库的补充, 添加了如下内容:
-- 1. cwait 机制
-- 2. 任务消息机制, 即 sub/pub 的增强版本
-- 在 socket,libnet,http 库等场景需要用到, 所以也需要 require
```

```
sysplus.waitMsg(taskName, target, timeout)
sysplus.sendMsg(taskName, target, arg2, arg3, arg4)
sysplus.taskInitEx(fun, taskName, cbFun, ...)
sysplus.taskDel(taskName)
sysplus.cleanMsg(taskName)
```

sysplus.waitMsg(taskName, target, timeout)

等待接收一个目标消息

参数

传入值类型	解释
string	任务名称，用于唤醒任务的 id
string	目标消息，如果为 nil，则表示接收到任意消息都会退出
int	超时时间，如果为 nil，则表示无超时，永远等待

返回值

返回值类型	解释
table	成功返回 table 型的 msg，超时返回 false

例子

```lua
-- 等待任务
sysplus.waitMsg('a', 'b', 1000)
-- 注意，本函数会自动注册成全局函数 sys_wait
```

---

sysplus.sendMsg(taskName, target, arg2, arg3, arg4)

向目标任务发送一个消息

参数

传入值类型	解释
string	任务名称，用于唤醒任务的 id
any	消息中的参数 1，同时也是 waitMsg 里的 target
any	消息中的参数 2
any	消息中的参数 3

传入值类型	解释
any	消息中的参数 4

返回值

返回值类型	解释
bool	成功返回 true，否则返回 false

例子

```
-- 向任务 a,目标 b 发送消息
sysplus.sendMsg('a', 'b')
-- 注意，本函数会自动注册成全局函数 sys_send
```

---

sysplus.taskInitEx(fun, taskName, cbFun, …)

创建一个任务线程,在模块最末行调用该函数并注册模块中的任务函数,main.lua 导入该模块即可

参数

传入值类型	解释
function	任务函数名,用于 resume 唤醒时调用
string	任务名称,用于唤醒任务的 id
function	接收到非目标消息时的回调函数
any	… 任务函数 fun 的可变参数

返回值

返回值类型	解释
number	返回该任务的线程号

例子

```
sysplus.taskInitEx(task1,'a',callback)
```

---

sysplus.taskDel(taskName)

删除由 taskInitEx 创建的任务线程

参数

传入值类型	解释
string	任务名称,用于唤醒任务的 id

返回值

无

例子

```
sysplus.taskDel('a')
```

---

sysplus.cleanMsg(taskName)

清除指定 task 的消息队列

参数

传入值类型	解释
string	任务名称

返回值

无

例子

```
sysplus.cleanMsg('a')
```

u8g2 - u8g2 图形处理库

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 u8g2 的 demo 例子

常量

u8g2.begin(conf)

u8g2.close()

u8g2.ClearBuffer()

u8g2.SendBuffer()

u8g2.DrawUTF8(str, x, y)

u8g2.DrawButtonUTF8(str, x, y, flags, w, h, v, str)

u8g2.SetFontMode(mode)

u8g2.SetFont(font, indentation)

u8g2.GetDisplayHeight()

u8g2.GetDisplayWidth()

u8g2.SetDrawColor(c)

u8g2.DrawPixel(x,y)

u8g2.DrawLine(x0,y0,x1,y1)

u8g2.DrawCircle(x0,y0,rad,opt)

u8g2.DrawDisc(x0,y0,rad,opt)

u8g2.DrawEllipse(x0,y0,rx,ry,opt)

u8g2.DrawFilledEllipse(x0,y0,rx,ry,opt)

u8g2.DrawBox(x,y,w,h)

u8g2.DrawFrame(x,y,w,h)

u8g2.DrawRBox(x,y,w,h,r)

u8g2.DrawRFrame(x,y,w,h,r)

u8g2.DrawRBox(x,y,w,h,r)

u8g2.DrawRFrame(x,y,w,h,r)

u8g2.DrawGlyph(x,y,encoding)

u8g2.DrawTriangle(x0,y0,x1,y1,x2,y2)

u8g2.SetBitmapMode(mode)

u8g2.DrawXBM(x, y, w, h, data)

u8g2.DrawDrcode(x, y, str, size)

u8g2.SetContrast(contrast)

u8g2.drawGtfontGb2312(str,size,x,y)

u8g2.drawGtfontUtf8(str,size,x,y)

u8g2.CopyBuffer(buff)

u8g2.SetPowerSave(is_enable)

常量

常量	类型	解释
u8g2.DRAW_UPPER_RIGHT	number	上右
u8g2.DRAW_UPPER_LEFT	number	上左
u8g2.DRAW_LOWER_LEFT	number	下左
u8g2.DRAW_LOWER_RIGHT	number	下右
u8g2.DRAW_ALL	number	全部
u8g2.BTN_BW0	number	文本周围没有边框
u8g2.BTN_BW1	number	文本周围的边框，1 像素边框宽度
u8g2.BTN_BW2	number	文本周围的边框，2 像素边框宽度
u8g2.BTN_BW3	number	文本周围的边框，3 像素边框宽度
u8g2.BTN_SHADOW0	number	启用阴影，与框架无间隙
u8g2.BTN_SHADOW1	number	启用阴影，到帧的 1 像素间隙
u8g2.BTN_SHADOW2	number	启用阴影，到帧的 2 像素间隙
u8g2.BTN_INV	number	反转文本
u8g2.BTN_HCENTER	number	将文本置于框架内的中心，并将参考位置更改为文本的中心
u8g2.BTN_XFRAME	number	在按钮周围绘制第二个 1 像素框

u8g2.begin(conf)

u8g2 显示屏初始化

参数

传入值类型	解释
table	conf 配置信息,详细配置看下面的例子

返回值

返回值类型	解释
int	正常初始化 1,已经初始化过 2,内存不够 3,初始化失败返回 4

例子

```
-- conf 配置项说明

-- ic 字符串,主控芯片类型, 支持 custom(自定义命令) ssd1306(默认) ssd1309 ssd1322 sh1106 sh1107 sh1108 st7567 uc1701 ssd1306_128x32

-- direction 数值,方向,可选 0 90 180 270 默认 0

-- mode 字符串,模式,可选 i2c_sw:软件 i2c i2c_hw:硬件 i2c,spi_hw_4pin:硬件 spi

-- i2c_id: 数值,硬件 i2c 时有效

-- i2c_scl: 数值,软件 i2c 时时钟线的 GPIO 编号

-- i2c_sda: 数值软件 i2c 时数据线的 GPIO 编号

-- spi_id、spi_res、spi_dc、spi_cs: 数值,硬件 spi 的 SPI 编号,复位 GPIO 编号,DC 线的 GPIO 编号, CS 线的 GPIO 编号

-- x_offset: 数值,X 轴偏移量,默认按驱动走, 2023.11.10 新增的配置项

-- 初始化硬件 i2c 的 ssd1306

u8g2.begin({ic = "ssd1306",direction = 0,mode="i2c_hw",i2c_id=0})
-- direction 可选 0 90 180 270

-- 初始化硬件 spi 的 ssd1306

u8g2.begin({ic = "ssd1306",direction = 0,mode="spi_hw_4pin",spi_id=0,spi_res=pin.PB03,spi_dc=pin.PB01,spi_cs=pin.PB04}) -- direction 可选 0 90 180 270
```

```
-- 初始化软件 i2c 的 ssd1306
u8g2.begin({ic = "ssd1306",direction = 0,mode="i2c_sw", i2c_scl=1, i2c_sda=4})
-- 通过 PA1 SCL / PA4 SDA 模拟
```

---

u8g2.close()

关闭显示屏

参数

无

返回值

无

例子

```
-- 关闭 disp,再次使用 disp 相关 API 的话,需要重新初始化
u8g2.close()
```

---

u8g2.ClearBuffer()

清屏，清除内存帧缓冲区中的所有像素

参数

无

返回值

无

例子

```
-- 清屏
u8g2.ClearBuffer()
```

---

u8g2.SendBuffer()

将数据更新到屏幕，将存储器帧缓冲区的内容发送到显示器

参数

无

返回值

无

例子

```
-- 把显示数据更新到屏幕
u8g2.SendBuffer()
```

---

u8g2.DrawUTF8(str, x, y)

在显示屏上画一段文字，在显示屏上画一段文字,要调用 u8g2.SendBuffer()才会更新到屏 注意:设置坐标为显示字符串左下角坐标

参数

传入值类型	解释
string	文件内容
int	横坐标
int	竖坐标

返回值

无

例子

```
u8g2.DrawUTF8("wifi is ready", 10, 20)
```

---

u8g2.DrawButtonUTF8(str, x, y, flags, w, h, v, str)

在提供的文本周围画一个框。这与 DrawUTF8 类似，但为文本添加了一些装饰。要调用 u8g2.SendBuffer()才会更新到屏幕 注意:设置坐标为显示字符串左下角坐标

参数

传入值类型	解释
string	文件内容
int	横坐标
int	竖坐标
int	标志 多个标志可以与“或”运算符一起使用。
int	文本的最小宽度。如果为 0（或低于文本宽度），则文本宽度将用于框架。
int	文本前后的额外空间。
int	文本上下的额外空间。

返回值

无

例子

```
u8g2.DrawButtonUTF8("str", 10, 20,u8g2.BTN_BW2,0,2,2)
```

---

u8g2.SetFontMode(mode)

设置字体模式

参数

传入值类型	解释
int	mode 字体模式，启用（1）或禁用（0）透明模式

返回值

无

例子

```
u8g2.SetFontMode(1)
```

---

u8g2.SetFont(font, indentation)

设置字体

参数

传入值类型	解释
userdata	font, u8g2.font_opposansm8 为纯英文 8 号字体,还有 font_opposansm10 font_opposansm12 font_opposansm16 font_opposansm18 font_opposansm20 font_opposansm22 font_opposansm24 font_opposansm32 可选 u8g2.font_opposansm12_chinese 为 12x12 全中文,还有 font_opposansm16_chinese font_opposansm24_chinese font_opposansm32_chinese 可选, u8g2.font_unifont_t_symbols 为符号.
int	indentation，等宽字体 ascii 右侧缩进 0~127 个 pixel，等宽字体的 ascii 字符可能在右侧有大片空白，用户可以选择删除部分。留空或者超过

传入值类型	解释
	127 则直接删除右半边，非等宽字体无效

返回值

无

例子

```
-- 设置为中文字体,对之后的 drawStr 有效
u8g2.SetFont(u8g2.font_opposansm12)
```

---

u8g2.GetDisplayHeight()

获取显示屏高度

参数

无

返回值

返回值类型	解释
int	显示屏高度

例子

```
u8g2.GetDisplayHeight()
```

---

u8g2.GetDisplayWidth()

获取显示屏宽度

参数

无

返回值

返回值类型　　解释

int　　显示屏宽度

例子

```
u8g2.GetDisplayWidth()
```

---

u8g2.SetDrawColor(c)

为所有绘图功能分配绘图颜色。

参数

传入值类型　　解释

int　　c 为颜色值 0 没有色 1 有色 2 与底色 xor

返回值

无

例子

```
u8g2.SetDrawColor(0)
```

---

u8g2.DrawPixel(x,y)

画一个点.

参数

传入值类型　解释

传入值类型	解释
int	X 位置.
int	Y 位置.

返回值

无

例子

```
u8g2.DrawPixel(20, 5)
```

---

u8g2.DrawLine(x0,y0,x1,y1)

在两点之间画一条线.

参数

传入值类型	解释
int	第一个点的 X 位置.
int	第一个点的 Y 位置.
int	第二个点的 X 位置.
int	第二个点的 Y 位置.

返回值

无

例子

```
u8g2.DrawLine(20, 5, 5, 32)
```

---

u8g2.DrawCircle(x0,y0,rad,opt)

在 x,y 位置画一个半径为 rad 的空心圆.

参数

传入

值类型	**解释**
int	圆心位置
int	圆心位置
int	圆半径.
int	选择圆的部分或全部. 默认全画 可选 u8g2.DRAW_UPPER_RIGHT u8g2.DRAW_UPPER_LEFT u8g2.DRAW_LOWER_LEFT u8g2.DRAW_LOWER_RIGHT u8g2.DRAW_ALL

返回值

无

例子

```
u8g2.DrawCircle(60,30,8,u8g2.DRAW_ALL)
```

---

u8g2.DrawDisc(x0,y0,rad,opt)

在 x,y 位置画一个半径为 rad 的实心圆.

参数

传入

值类型	**解释**
int	圆心位置

传入

值类型	**解释**
int	圆心位置
int	圆半径.
int	选择圆的部分或全部. 默认全画 可选 u8g2.DRAW_UPPER_RIGHT u8g2.DRAW_UPPER_LEFT u8g2.DRAW_LOWER_LEFT u8g2.DRAW_LOWER_RIGHT u8g2.DRAW_ALL

返回值

无

例子

```
u8g2.DrawDisc(60,30,8,u8g2.DRAW_ALL)
```

---

u8g2.DrawEllipse(x0,y0,rx,ry,opt)

在 x,y 位置画一个半径为 rad 的空心椭圆.

参数

传入

值类型	**解释**
int	圆心位置
int	圆心位置
int	椭圆大小
int	椭圆大小
int	选择圆的部分或全部. 默认全画 可选 u8g2.DRAW_UPPER_RIGHT u8g2.DRAW_UPPER_LEFT u8g2.DRAW_LOWER_LEFT

传入

值类型	**解释**
	u8g2.DRAW_LOWER_RIGHT u8g2.DRAW_ALL

返回值

无

例子

```
u8g2.DrawEllipse(60,30,8,u8g2.DRAW_ALL)
```

---

u8g2.DrawFilledEllipse(x0,y0,rx,ry,opt)

在 x,y 位置画一个半径为 rad 的实心椭圆.

参数

传入

值类型	**解释**
int	圆心位置
int	圆心位置
int	椭圆大小
int	椭圆大小
int	选择圆的部分或全部. 默认全画 可选 u8g2.DRAW_UPPER_RIGHT u8g2.DRAW_UPPER_LEFT u8g2.DRAW_LOWER_LEFT u8g2.DRAW_LOWER_RIGHT u8g2.DRAW_ALL

返回值

无

例子

```
u8g2.DrawFilledEllipse(60,30,8,15)
```

---

u8g2.DrawBox(x,y,w,h)

从 x / y 位置（左上边缘）开始绘制一个框（填充的框）.

参数

传入值类型	解释
int	左上边缘的 X 位置
int	左上边缘的 Y 位置
int	盒子的宽度
int	盒子的高度

返回值

无

例子

```
u8g2.DrawBox(3,7,25,15)
```

---

u8g2.DrawFrame(x,y,w,h)

从 x / y 位置（左上边缘）开始绘制一个框（空框）.

参数

传入值类型	解释
int	左上边缘的 X 位置
int	左上边缘的 Y 位置

传入值类型	解释
int	盒子的宽度
int	盒子的高度

返回值

无

例子

```
u8g2.DrawFrame(3,7,25,15)
```

---

u8g2.DrawRBox(x,y,w,h,r)

绘制一个从 x / y 位置（左上边缘）开始具有圆形边缘的填充框/框架.

参数

传入值类型	解释
int	左上边缘的 X 位置
int	左上边缘的 Y 位置
int	盒子的宽度
int	盒子的高度
int	四个边缘的半径

返回值

无

例子

```
u8g2.DrawRBox(3,7,25,15)
```

---

u8g2.DrawRFrame(x,y,w,h,r)

绘制一个从 x / y 位置（左上边缘）开始具有圆形边缘的空框/框架.

参数

传入值类型	解释
int	左上边缘的 X 位置
int	左上边缘的 Y 位置
int	盒子的宽度
int	盒子的高度
int	四个边缘的半径

返回值

无

例子

```
u8g2.DrawRFrame(3,7,25,15)
```

---

u8g2.DrawGlyph(x,y,encoding)

绘制一个图形字符。字符放置在指定的像素位置 x 和 y.

参数

传入值类型	解释
int	字符在显示屏上的位置
int	字符在显示屏上的位置
int	字符的 Unicode 值

返回值

无

例子

```
u8g2.SetFont(u8g2_font_unifont_t_symbols)
u8g2.DrawGlyph(5, 20, 0x2603)      -- dec 9731/hex 2603 Snowman
```

---

u8g2.DrawTriangle(x0,y0,x1,y1,x2,y2)

绘制一个三角形（实心多边形）.

参数

传入值类型	解释
int	点 0X 位置
int	点 0Y 位置
int	点 1X 位置
int	点 1Y 位置
int	点 2X 位置
int	点 2Y 位置

返回值

无

例子

```
u8g2.DrawTriangle(20,5, 27,50, 5,32)
```

---

u8g2.SetBitmapMode(mode)

定义位图函数是否将写入背景色

参数

传入值类型	解释
int	mode 字体模式，启用（1）或禁用（0）透明模式

返回值

无

例子

```
u8g2.SetBitmapMode(1)
```

---

u8g2.DrawXBM(x, y, w, h, data)

绘制位图

参数

传入值类型	解释
int	X 坐标
int	y 坐标
int	位图宽
int	位图高
int	位图数据,每一位代表一个像素

返回值

无

例子

```
-- 取模使用 PCtoLCD2002 软件即可
-- 在(0,0)为左上角,绘制 16x16 "今" 的位图
```

```
u8g2.DrawXBM(0, 0, 16,16, string.char(

0x80,0x00,0x80,0x00,0x40,0x01,0x20,0x02,0x10,0x04,0x48,0x08,0x8
4,0x10,0x83,0x60,

0x00,0x00,0xF8,0x0F,0x00,0x08,0x00,0x04,0x00,0x04,0x00,0x02,0x0
0,0x01,0x80,0x00))
```

---

u8g2.DrawDrcode(x, y, str, size)

缓冲区绘制 QRCode

参数

传入值类型	解释
int	x 坐标
int	y 坐标
string	二维码的内容
int	显示大小 (注意:二维码生成大小与要显示内容和纠错等级有关,生成版本为 1-40(对应 21x21 - 177x177)的不定大小,如果和设置大小不同会自动在指定的区域中间显示二维码,如二维码未显示请查看日志提示)

返回值

返回值类型	解释
nil	无返回值

例子

无

---

u8g2.SetContrast(contrast)

调整屏幕对比度

参数

传入值类型	解释
int	对比度（取值范围与 IC 有关，如 st7567 为 0-63）

返回值

返回值类型	解释
nil	无返回值

例子

无

---

u8g2.drawGtfontGb2312(str,size,x,y)

使用 gtfont 显示 gb2312 字符串

参数

传入值类型	解释
string	str 显示字符串
int	size 字体大小 (支持 16-192 号大小字体)
int	x 横坐标
int	y 竖坐标

返回值

无

例子

```
u8g2.drawGtfontGb2312("啊啊啊",32,0,0)
```

---

u8g2.drawGtfontUtf8(str,size,x,y)

使用 gtfont 显示 UTF8 字符串

参数

传入值类型	解释
string	str 显示字符串
int	size 字体大小 (支持 16-192 号大小字体)
int	x 横坐标
int	y 竖坐标

返回值

无

例子

```
u8g2.drawGtfontUtf8("啊啊啊",32,0,0)
```

---

u8g2.CopyBuffer(buff)

获取底层图像缓冲区

参数

传入值类型	解释
userdata	zbuff 实例,空间要大于等于底层 buff 的大小

返回值

返回值类型	解释
int	成功返回 buff 大小,否则返回 nil

例子

```
-- 本函数在 u8g2 初始化之后才能调用
-- 获取大小
local sz = u8g2.CopyBuffer()
-- 拷贝底层 buff
local buff = zbuff.create(sz)
u8g2.CopyBuffer(buff)
```

---

u8g2.SetPowerSave(is_enable)

设置省电模式

参数

传入值类型	解释
boolean	启用与否. true 启用, false 禁用

返回值

返回值类型	解释
nil	无返回值

例子

```
-- 本 API 于 2023.11.02 新增
-- 开启省电
u8g2.SetPowerSave(true)
-- 关闭省电
u8g2.SetPowerSave(false)
```

uart - 串口操作库

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 uart 的 demo 例子

小技巧

本库还有视频教程，点此链接查看

常量

uart.setup(id, baud_rate, data_bits, stop_bits, partiy, bit_order, buff_size, rs485_gpio, rs485_level, rs485_delay)

uart.write(id, data)

uart.read(id, len)

uart.close(id)

uart.on(id, event, func)

uart.wait485(id)

uart.exist(id)

uart.rx(id, buff)

uart.rxSize(id)

uart.rxClear(id)

uart.tx(id, buff, start, len)

uart.createSoft(tx_pin, tx_hwtimer_id, rx_pin, rx_hwtimer_id, adjust_period)

uart.list(max)

常量

常量	类型	解释

常量	类型	解释
uart.Odd	number	奇校验,大小写兼容性
uart.Even	number	偶校验,大小写兼容性
uart.None	number	无校验,大小写兼容性
uart.ODD	number	奇校验
uart.EVEN	number	偶校验
uart.NONE	number	无校验
uart.LSB	number	小端模式
uart.MSB	number	大端模式
uart.VUART_0	number	虚拟串口 0

uart.setup(id, baud_rate, data_bits, stop_bits, partiy, bit_order, buff_size, rs485_gpio, rs485_level, rs485_delay)

配置串口参数

参数

传入值

类型	解释
int	串口 id, uart0 写 0, uart1 写 1，如此类推，最大值取决于设备
int	波特率，默认 115200，可选择波特率表:{2000000,921600,460800,230400,115200,57600,38400,19200,9600,4800,2400}
int	数据位，默认为 8，可选 7/8
int	停止位，默认为 1，根据实际情况，可以有 0.5/1/1.5/2 等
int	校验位，可选 uart.None/uart.Even/uart.Odd

传入值

类型	解释
int	大小端，默认小端 uart.LSB，可选 uart.MSB
int	缓冲区大小，默认值 1024
int	485 模式下的转换 GPIO，默认值 0xffffffff
int	485 模式下的 rx 方向 GPIO 的电平，默认值 0
int	485 模式下 tx 向 rx 转换的延迟时间，默认值 12bit 的时间，单位 us，9600 波特率填 20000

返回值

返回值类型	解释
int	成功返回 0,失败返回其他值

例子

```
-- 最常用 115200 8N1
uart.setup(1, 115200, 8, 1, uart.NONE)
-- 可以简写为 uart.setup(1)
-- 485 自动切换，选取 GPIO10 作为收发转换脚
uart.setup(1, 115200, 8, 1, uart.NONE, uart.LSB, 1024, 10, 0, 2000)
```

---

uart.write(id, data)

写串口

参数

传入值类型	解释
int	串口 id，uart0 写 0，uart1 写 1
string/zbuff	待写入的数据，如果是 zbuff 会从指针起始位置开始读

传入值类型	解释
int	可选，要发送的数据长度，默认全发

返回值

返回值类型	解释
int	成功的数据长度

例子

```
-- 写入可见字符串
uart.write(1, "rdy\r\n")
-- 写入十六进制的数据串
uart.write(1, string.char(0x55,0xAA,0x4B,0x03,0x86))
```

---

uart.read(id, len)

读串口

参数

传入值类型	解释
int	串口 id, uart0 写 0, uart1 写 1
int	读取长度
file/zbuff	可选：文件句柄或 zbuff 对象

返回值

返回值类型	解释
string	读取到的数据 / 传入 zbuff 时，返回读到的长度，并把 zbuff 指针后移

例子

```
uart.read(1, 16)
```

---

uart.close(id)

关闭串口

参数

传入值类型	解释
int	串口 id, uart0 写 0, uart1 写 1

返回值

返回值类型	解释
nil	无返回值

例子

```
uart.close(1)
```

---

uart.on(id, event, func)

注册串口事件回调

参数

传入值类型	解释
int	串口 id, uart0 写 0, uart1 写 1
string	事件名称
function	回调方法

返回值

返回值类型　解释

nil　　无返回值

例子

```
uart.on(1, "receive", function(id, len)
    local data = uart.read(id, len)
    log.info("uart", id, len, data)
end)
```

---

uart.wait485(id)

等待 485 模式下 TX 完成，mcu 不支持串口发送移位寄存器空或者类似中断时才需要，在 sent 事件回调后使用

参数

传入值类型　　解释

int　　串口 id, uart0 写 0, uart1 写 1

返回值

返回值类型	解释
int	等待了多少次循环才等到 tx 完成，用于粗劣的观察 delay 时间是否足够，返回不为 0 说明还需要放大 delay

例子

无

---

uart.exist(id)

检查串口号是否存在

参数

传入值类型	解释
int	串口 id, uart0 写 0, uart1 写 1, 如此类推

返回值

返回值类型	解释
bool	存在返回 true

例子

无

---

uart.rx(id, buff)

buff 形式读串口，一次读出全部数据存入 buff 中，如果 buff 空间不够会自动扩展，目前 air105,air780e 支持这个操作

参数

传入值类型	解释
int	串口 id, uart0 写 0, uart1 写 1
zbuff	zbuff 对象

返回值

返回值类型	解释
int	返回读到的长度，并把 zbuff 指针后移

例子

```
uart.rx(1, buff)
```

---

uart.rxSize(id)

读串口 Rx 缓存中剩余数据量，目前 air105,air780e 支持这个操作

参数

传入值类型	解释
int	串口 id, uart0 写 0, uart1 写 1

返回值

返回值类型	解释
int	返回读到的长度

例子

```
local size = uart.rxSize(1)
```

---

uart.rxClear(id)

清除串口 Rx 缓存中剩余数据量，目前 air105,air780e 支持这个操作

参数

传入值类型	解释
int	串口 id, uart0 写 0, uart1 写 1

返回值

无

例子

```
uart.rxClear(1)
```

---

uart.tx(id, buff, start, len)

buff 形式写串口,等同于 c 语言 uart_tx(uart_id, &buff[start], len);

参数

传入值类型	解释
int	串口 id, uart0 写 0, uart1 写 1
zbuff	待写入的数据，如果是 zbuff 会从指针起始位置开始读
int	可选，要发送的数据起始位置，默认为 0
int	可选，要发送的数据长度，默认为 zbuff 内有效数据，最大值不超过 zbuff 的最大空间

返回值

返回值类型	解释
int	成功的数据长度

例子

```
uart.tx(1, buf)
```

---

uart.createSoft(tx_pin, tx_hwtimer_id, rx_pin, rx_hwtimer_id, adjust_period)

设置软件 uart 的硬件配置，只有支持硬件定时器的 SOC 才能使用，目前只能设置一个，波特率根据平台的软硬件配置有不同的极限，建议 9600，接收缓存不超过 65535，不支持 MSB，支持 485 自动控制。后续仍要 setup 操作

参数

传入值类型 **解释**

int	发送引脚编号
int	发送用的硬件定时器 ID
int	接收引脚编号
int	接收用的硬件定时器 ID
int	发送时序调整，单位是定时器时钟周期，默认是 0，需要根据示波器或者逻辑分析仪进行微调
int	接收时序调整，单位是定时器时钟周期，默认是 0，需要根据示波器或者逻辑分析仪进行微调

返回值

返回值类型 **解释**

int	软件 uart 的 id，如果失败则返回 nil

例子

```
-- 初始化软件 uart
local uart_id = uart.createSoft(21, 0, 1, 2) --air780e 建议用定时器 0 和 2，tx_pin 最好用 AGPIO，防止休眠时误触发对端 RX
```

---

uart.list(max)

获取可用串口号列表，当前仅限 win32

参数

传入值类型 **解释**

int	可选，默认 256，最多获取多少个串口

返回值

返回值类型	解释
table	获取到的可用串口号列表

例子

无

w5500 - w5500 以太网驱动

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 w5500 的 demo 例子

```
w5500.init(spiid, speed, cs_pin, irq_pin, rst_pin, link_pin)
w5500.config(ip, submask, gateway, mac, RTR, RCR, speed)
w5500.bind(id)
w5500.getMac()
```

w5500.init(spiid, speed, cs_pin, irq_pin, rst_pin, link_pin)

初始化 w5500

参数

传入值类型	解释
int	spi 通道号，例如 0, 1, 5，按设备实际情况选
int	spi 速度，可以设置到对应 SPI 的最高速度
int	cs pin，片选脚，对应 W5500 的 SCS

传入值类型	解释
int	irq pin，中断脚，对应 W5500 的 INT
int	reset pin，复位脚，对应 W5500 的 RST
int	link 状态 pin，可以留空不使用，默认不使用

返回值

无

例子

```
w5500.init(spi.SPI_0, 24000000, pin.PB13, pin.PC08, pin.PC09)
```

---

w5500.config(ip, submask, gateway, mac, RTR, RCR, speed)

w5500 配置网络信息

参数

传入值类型	解释
string	静态 ip 地址，如果需要用 DHCP 获取，请写 nil
string	子网掩码，如果使用动态 ip，则忽略
string	网关，如果使用动态 ip，则忽略
string	MAC，写 nil 则通过 MCU 唯一码自动生成，如果要写，长度必须是 6byte
int	重试间隔时间，默认 2000，单位 100us，不懂的不要改
int	最大重试次数，默认 8，不懂的不要改
int	速度类型，目前只有 0 硬件配置，1 自适应，默认为 0

返回值

无

例子

```
w5500.config("192.168.1.2", "255.255.255.0", "192.168.1.1",
string.fromHex("102a3b4c5d6e"))
```

---

w5500.bind(id)

将 w5500 注册进通用网络接口

参数

传入值类型	解释
int	通用网络通道号

返回值

无

例子

```
-- 若使用的版本不带 socket 库，改成 network.ETH0
w5500.bind(socket.ETH0)
```

---

w5500.getMac()

获取 w5500 当前的 MAC，必须在 init 之后用，如果 config 中设置了自己的 MAC，需要延迟一点时间再读

参数

无

返回值

返回值类型　　解释

string　　当前的 MAC

例子

```
local mac = w5500.getMac()log.info("w5500 mac", mac:toHex())
```

wdt.init(timeout)

wdt.setTimeout(timeout)

wdt.feed()

wdt.close()

wdt - watchdog 操作库

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 wdt 的 demo 例子

wdt.init(timeout)

初始化 watchdog 并马上启用.大部分设备的 watchdog 一旦启用就无法关闭.

参数

传入值类型　　解释

int　　超时时长,单位为毫秒

返回值

返回值类型　　解释

返回值类型	解释
bool	成功返回 true,否则返回 false(例如底层不支持)

例子

```
wdt.init(9000)sys.timerLoopStart(wdt.feed, 3000)
```

---

wdt.setTimeout(timeout)

部分设备支持重新设置 watchdog 超时时长

参数

传入值类型	解释
int	超时时长,单位为毫秒

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false(例如底层不支持)

例子

```
wdt.init(10000)
sys.timerLoopStart(wdt.feed, 3000)
sys.wait(5000)
sys.setTimeout(5000)
```

---

wdt.feed()

喂狗,使得超时计时复位,重新计时

参数

无

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false(例如底层不支持)

例子

```
wdt.init(10000)-- 定时喂狗,或者根据业务按需喂狗
sys.timerLoopStart(wdt.feed, 3000)
```

---

wdt.close()

关闭 watchdog,通常不被支持

参数

无

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false(例如底层不支持)

例子

```
wdt.init(10000)
sys.wait(9000)
wdt.close()
```

websocket - websocket 客户端

已适配 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 websocket 的 demo 例子

示例

```
local wsc = nil
if websocket then
       wsc = websocket.create(nil,
"ws://echo.airtun.air32.cn/ws/echo")
    wsc:autoreconn(true, 3000) -- 自动重连机制
    wsc:on(function(wsc, event, data)
        log.info("wsc", event, data)
        if event == "conack" then
            wsc:send((json.encode({action="echo",
device_id=device_id})))
            sys.publish("wsc_conack")
        end
    end)
    wsc:connect()
    --sys.waitUntil("websocket_conack", 15000)
    while true do
        sys.wait(45000)
        if wsc:ready() then
              wsc:send((json.encode({action="echo",
msg=os.date()})))
        end
    end
    wsc:close()
    wsc = nil
end
```

wsc:debug(onoff)

websocket.create(adapter, url)

wsc:on(cb)

wsc:connect()

wsc:autoreconn(reconnect, reconnect_time)

wsc:send(data, fin, opt)

wsc:close()

wsc:ready()

wsc:headers(headers)

wsc:debug(onoff)

配置是否打开 debug 信息

参数

传入值类型	解释
boolean	是否打开 debug 开关

返回值

返回值类型	解释
nil	无返回值

例子

无

---

websocket.create(adapter, url)

websocket 客户端创建

参数

传入值

类型	解释
int	适配器序号，只能是 socket.ETH0, socket.STA, socket.AP,如果不填,会选择平台自带的方式,然后是最后一个注册的适配器
string	连接字符串,参考 usage

返回值

返回值类型	解释
userdata	若成功会返回 websocket 客户端实例,否则返回 nil

例子

```
-- 普通 TCP 链接
wsc = websocket.create(nil,"ws://air32.cn/abc")
-- 加密 TCP 链接
wsc = websocket.create(nil,"wss://air32.cn/abc")
```

---

wsc:on(cb)

注册 websocket 回调

参数

传入值类型	解释
function	cb websocket 回调,参数包括 websocket_client, event, data, payload

返回值

返回值类型	解释
nil	无返回值

例子

```
wsc:on(function(websocket_client, event, data, payload)
        -- 打印各种事件
        log.info("websocket", "event", event, data, payload)end)
--[[
event 的值有:
        conack 连接服务器成功,已经收到
        websocket 协议头部信息,通信已建立
        recv    收到服务器下发的信息,
        data, payload 不为 nil
        sent    send 函数发送的消息,服务器在 TCP 协议层已确认收到
        disconnect 服务器连接已断开
其中 sent/disconnect 事件在 2023.04.01 新增
]]
```

---

wsc:connect()

连接服务器

参数

无

返回值

返回值类型	解释
boolean	发起成功返回 true，否则返回 false

例子

```
-- 开始建立连接
wsc:connect()
```

```
-- 本函数仅代表发起成功，后续仍需根据 ready 函数判断 websocket 是否连接正常
```

---

wsc:autoreconn(reconnect, reconnect_time)

自动重连

参数

传入值类型	解释
bool	是否自动重连
int	自动重连周期 单位 ms 默认 3000ms

返回值

无

例子

```
wsc:autoreconn(true)
```

---

wsc:send(data, fin, opt)

发布消息

参数

传入值类型	解释
string	待发送的数据,必填
int	是否为最后一帧,默认 1,即马上设置为最后一帧，也就是单帧发送
int	操作码，默认为字符串帧 0，可选 1

返回值

返回值类型	解释
bool	成功返回 true,否则为 false 或者 nil

例子

```
-- 简单发送数据
wsc:send("123")
-- 分段发送数据，最后要用 1(即 FIN 帧结束)
wsc:send("123", 0)wsc:send("456", 0)wsc:send("789", 1)
```

---

wsc:close()

websocket 客户端关闭(关闭后资源释放无法再使用)

参数

无

返回值

无

例子

```
wsc:close()
```

---

wsc:ready()

websocket 客户端是否就绪

参数

无

返回值

返回值类型　　解释

bool　　客户端是否就绪

例子

```
local stat = wsc:ready()
```

---

wsc:headers(headers)

设置额外的 headers

参数

传入值类型　　解释

table/string　可以是 table,也可以是字符串

返回值

返回值类型　　解释

bool　　客户端是否就绪

例子

```
-- table 形式
wsc:headers({
        Auth="Basic ABCDEFGG"})
-- 字符串形式
wsc:headers("Auth: Basic ABCDERG\r\n")
```

wlan - wifi 操作

已适配 Air101/Air103 Air601 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 wlan 的 demo 例子

示例

```
--[[
提醒:
对于仅支持 wifiscan 的模块, 仅 init/scan/scanResult 函数是可用的
例如: Air780E/Air600E/Air780EG 等仅支持 wifiscan
]]
```

常量

wlan.init()

wlan.setMode(mode)

wlan.ready()

wlan.connect(ssid, password, auto_reconnect)

wlan.disconnect()

wlan.scan()

wlan.scanResult()

wlan.smartconfig(mode)

wlan.getMac(tp, hexstr)

wlan.setMac(tp, mac)

wlan.getIP()

wlan.createAP(ssid, passwd, gateway, netmask, channel, opts)

wlan.stopAP()

wlan.getInfo()

wlan.powerSave(mode)

wlan.hostname(new_name)

wlan.staIp(dhcp_enable, ip, netmask, gateway)

常量

常量	类型	解释
wlan.ESPTOUCH	esptouch 配网，V1	
wlan.AIRKISS	Airkiss 配网，	微信常用
wlan.ESPTOUCH_V2	esptouch 配网，V2，未测试	

wlan.init()

初始化

参数

无

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

无

---

wlan.setMode(mode)

设置 wifi 模式

参数

传入值类型	解释
int	wifi 模式

返回值

返回值类型　　解释

bool　　成功返回 true,否则返回 false

例子

```
-- 设置为 AP 模式，广播 ssid，接收 wifi 客户端的链接
wlan.setMode(wlan.AP)

-- 设置为 STATION 模式，也是初始化后的默认模式
wlan.setMode(wlan.STATION)

-- 混合模式，做 AP 又做 STATION

wlan.setMode(wlan.APSTA)
```

---

wlan.ready()

作为 STATION 时,是否已经连接上 AP,且获取 IP 成功

参数

无

返回值

返回值类型　　解释

bool　　已经连接成功返回 true,否则返回 false

例子

无

---

wlan.connect(ssid, password, auto_reconnect)

作为 STATION 时,连接到指定 AP

参数

传入值类型	解释
string	AP 的 ssid
string	AP 的 password,可选
int	0 关闭自动重连,1 开启自动重连.当前强制开启自动重连

返回值

返回值类型	解释
bool	发起连接成功返回 true,否则返回 false.注意,不代表连接 AP 成功!!

例子

```
-- 普通模式,带密码
wlan.connect("myap", "12345678")
-- 普通模式,不带密码
wlan.connect("myap")
-- 特殊模式, 重用之前的 ssid 和密码,本次直接连接
-- 注意, 前提是本次上电后已经传过 ssid 和或 password,否则必失败
wlan.connect()
```

---

wlan.disconnect()

作为 STATION 时,断开 AP

参数

无

返回值

无

例子

无

---

wlan.scan()

扫描 wifi 频段

参数

无

返回值

无

例子

```
-- 注意，wlan.scan()是异步 API,启动扫描后会马上返回
-- wifi 扫描成功后，会有 WLAN_SCAN_DONE 消息，读取即可
sys.subscribe("WLAN_SCAN_DONE", function ()
    local results = wlan.scanResult()
    log.info("scan", "results", #results)
    for k,v in pairs(results) do
        log.info("scan", v["ssid"], v["rssi"], (v["bssid"]:toHex()))
    end
end)
-- 下面演示的是初始化 wifi 后定时扫描,请按实际业务需求修改
sys.taskInit(function()
    sys.wait(1000)
    wlan.init()
    while 1 do
        wlan.scan()
        sys.wait(15000)
```

```
    end
end)
```

---

wlan.scanResult()

获取 wifi 扫描结果

参数

无

返回值

返回值类型	解释
table	扫描结果

例子

```
-- 用法请查阅 wlan.scan() 函数
```

---

wlan.smartconfig(mode)

配网

参数

传入值类型	解释
int	配网模式，默认为 esptouch，若传 0 则主动停止配网

返回值

返回值类型	解释
bool	启动成功或停止成功，返回 true，否则返回 false

例子

```
wlan.smartconfig()
local ret, ssid, passwd = sys.waitUntil("SC_RESULT", 180*1000)
-- 最多等 3 分钟
log.info("sc", ret, ssid, passwd)
-- 详细用法请查看 demo
```

---

wlan.getMac(tp, hexstr)

获取 mac

参数

传入值类型	解释
int	设置何种 mac 地址,对 ESP32 系列来说,只能设置 STA 的地址,即 0,默认值也是 0
bool	是否转 HEX 字符，默认是 true,即输出 hex 字符串

返回值

返回值类型	解释
string	MAC 地址,十六进制字符串形式“AABBCCDDEEFF”或原始数据

例子

无

---

wlan.setMac(tp, mac)

设置 mac

参数

传入值类型	解释
int	设置何种 mac 地址,对 ESP32 系列来说,只能设置 STA 的地址,即 0
string	待设置的 MAC 地址,长度 6 字节

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

```
-- 设置 MAC 地址，2023-03-01 之后编译的固件可用
local mac = string.fromHex("F01122334455")
wlan.setMac(0, mac)
-- 部分模块支持恢复默认 MAC，例如 esp32 系列
-- 在 2023-11-01 之后编译的固件可用
local mac = string.fromHex("000000000000")
wlan.setMac(0, mac)
```

---

wlan.getIP()

获取 ip,仅 STATION 或 APSTA 模式下有意义

参数

无

返回值

返回值类型	解释
string	ip 地址,当前仅返回 ipv4 地址,例如 “192.168.1.25”

例子

无

---

wlan.createAP(ssid, passwd, gateway, netmask, channel, opts)

启动 AP

参数

传入值类型	解释
string	AP 的 SSID,必填
string	AP 的密码,可选
string	AP 的网关地址，默认 192.168.4.1
string	AP 的网关掩码，默认 255.255.255.0
int	AP 建立的通道，默认 6
table	AP 的配置选项，可选

返回值

返回值类型	解释
bool	成功创建返回 true,否则返回 false

例子

```
-- 注意，调用本 AP 时,若 wifi 模式为 STATION,会自动切换成 APSTA
wlan.createAP("uiot", "12345678")
-- 设置网关 IP,掩码，通道，2023.7.13 新增，BSP 未必支持
-- wlan.createAP("uiot", "12345678", "192.168.4.1",
"255.255.255.0", 6)
-- opts 更多配置项，2024.3.5 新增
--[[
{
```

```
        hidden = false, -- 是否隐藏 SSID, 默认 false,不隐藏
        max_conn = 4 -- 最大客户端数量, 默认 4
}
]]
```

---

wlan.stopAP()

关闭 AP 功能

参数

无

返回值

返回值类型	解释
bool	成功创建返回 true,否则返回 false

例子

```
wlan.stopAP()
```

---

wlan.getInfo()

获取信息,如 AP 的 bssid,信号强度

参数

无

返回值

返回值类型	解释
table	详情,键值对形式

例子

```
log.info("wlan", "info", json.encode(wlan.getInfo()))
--[[
典型输出
{
        "bssid" : "xxxxxx",
        "rssi"  : -89,
        "gw" : "192.168.1.1"
}
]]
```

---

wlan.powerSave(mode)

读取或设置省电模式

参数

传入值类型	解释
int	省电模式,可选，传入就是设置，例如 wlan.PS_NONE

返回值

返回值类型	解释
int	当前省电模式/设置后的省电模式

例子

```
-- 请查阅常量表  PS_NONE/PS_MIN_MODEM/PS_MAX_MODEM
log.info("wlan", "PS", wlan.powerSave(wlan.PS_NONE))
-- 本 API 于 2023.03.31 新增
```

---

wlan.hostname(new_name)

读取或设置 Hostname

参数

传入值类型	解释
string	新的 hostname,可选，传入就是设置

返回值

返回值类型	解释
string	当前的 hostname 或者设置后的 hostname

例子

```
-- 本 API 于 2023.07.23 新增
-- 本函数应该在 wlan.init 之前设置好，最晚应早于 wlan.connect
-- hostname 的默认值是 "LUATOS_" + 设备的 MAC 值
-- 例如: LUATOS_0022EECC2399
wlan.hostname("我的 wifi 物联网设备")
```

---

wlan.staIp(dhcp_enable, ip, netmask, gateway)

设置 Station 模式下的 IP 获取模式

参数

传入值类型	解释
bool	是否启用 DHCP,默认是 true
string	本机 IP 地址,例如 192.168.2.200，禁用 DHCP 时必填
string	本机 IP 掩码,例如 255.255.255.0，禁用 DHCP 时必填
string	本机 IP 网关,例如 192.168.2.1，禁用 DHCP 时必填

返回值

返回值类型	解释
bool	成功返回 true,否则返回 false

例子

```
-- 本 API 于 2023.10.06 新增
-- 本函数需要在 wlan.init 之后才允许调用
-- 启用 DHCP, 默认也是启用 DHCP,这里是演示 API 使用
wlan.staIp(true)
-- 禁用 DHCP,自行设置 IP/掩码/网关
wlan.staIp(false, "192.168.2.200", "255.255.255.0",
"192.168.2.1")
```

ws2812.create(mode, count, id)

ws2812.set(leds,index, R, G, B)

ws2812.send(leds)

ws2812.args(leds, arg0, arg1, arg2, arg3, arg4)

ws2812 - 幻彩灯带 RGB 控制器(WS2812 系列)

已适配 Air101/Air103 Air601 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

示例

```
-- 本库尚在开发阶段
```

ws2812.create(mode, count, id)

创建上下文

参数

传入值类型	解释
int	传输模式，例如 ws2812.GPIO, ws2812.PWM, ws2812.SPI
int	RGB 灯总个数
int	主参数 id，对于不同的模式有不同的值

返回值

返回值类型	解释
userdata	创建成功返回上下文,否则返回 nil

例子

```
-- GPIO 模式，64 个灯，使用 GPIO9
local leds = ws2812.create(ws2812.GPIO, 64, 9)
-- SPI 模式，32 个灯，使用 SPI1
local leds = ws2812.create(ws2812.SPI, 32, 1)
-- PWM 模式，16 个灯，使用 PWM4
local leds = ws2812.create(ws2812.PWM, 16, 4)
-- HW 模式，64 个灯，使用硬件专用实现,具体 id 需要对照手册
local leds = ws2812.create(ws2812.RMT, 64, 2)
-- 注意：并非所有模块都支持以上所有模式
-- 而且，固件需要开启对应的 GPIO/SPI/PWM 功能才能使用对应的模式
```

---

ws2812.set(leds,index, R, G, B)

设置灯的颜色

参数

传入值类型	解释

传入值类型	解释
userdata	通过 ws2812.create 获取到的上下文
int	灯的编号,从 0 开始
int	RGB 值中的 R 值
int	RGB 值中的 G 值
int	RGB 值中的 B 值

返回值

返回值类型	解释
boolean	设置成功返回 true,否则返回 nil

例子

```
-- RGB 逐个颜色传递
ws2812.set(leds, 5, 0xFF, 0xAA, 0x11)
-- 也支持一个参数传完，与前一条等价
ws2812.set(leds, 5, 0xFFAA11)
```

---

ws2812.send(leds)

发送数据到设备

参数

传入值类型	解释
userdata	通过 ws2812.create 获取到的上下文

返回值

返回值类型	解释
boolean	设置成功返回 true,否则返回 nil

例子

```
-- 没有更多参数，发就完事了
ws2812.send(leds)
```

---

ws2812.args(leds, arg0, arg1, arg2, arg3, arg4)

配置额外参数

参数

传入值类型	解释
userdata	通过 ws2812.create 获取到的上下文
int	额外参数 0
int	额外参数 1
int	额外参数 2
int	额外参数 3
int	额外参数 4

返回值

返回值类型	解释
boolean	设置成功返回 true,否则返回 nil

例子

```
-- 本函数与具体模式有关
--GPIO 模式可调整 T0H T0L, T1H T1L 的具体延时
ws2812.send(leds, t0h, t0l, t1h, t1l)
```

xxtea - xxtea 加密解密

已适配 Air101/Air103 Air601 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

示例

```lua
-- 本库是对 https://github.com/xxtea/xxtea-c 的封装，再次感谢 xxtea 的作者

local text = "Hello World!"

local key = "07946"

local encrypt_data = xxtea.encrypt(text, key)

log.info("testCrypto.xxteaTest","xxtea_encrypt:"..encrypt_data)
local decrypt_data = xxtea.decrypt(encrypt_data, key)

log.info("testCrypto.xxteaTest","decrypt_data:"..decrypt_data)
```

xxtea.encrypt(data, key)

xxtea.decrypt(data, key)

xxtea.encrypt(data, key)

加密

参数

传入值类型		解释
string	data	待加密的数据
string	key	加密用的密钥

返回值

返回值类型	解释

返回值类型 **解释**

string 加密后的数据，失败返回 nil

例子

无

---

xxtea.decrypt(data, key)

解密

参数

传入值类型 **解释**

string data 待解密的数据

string key 解密用的密钥

返回值

返回值类型 **解释**

string 解密后的数据，失败返回 nil

例子

无

ymodem - ymodem 协议

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

示例

```
-- 本库的用途是接收数据，若需要发送文件，建议用 xmodem 库
local handler = ymodem.create("/")
uart.setup(1, 115200)
uart.on(1, "receive", function(id, len)
	while 1 do
		local data = uart.read(id, 512)
		if not data or #data == 0 then
			break
		end
		ymodem.receive(handler, data)
	end
end)
```

ymodem.create(dir_path,file_path)

ymodem.receive(handler, data)

ymodem.reset(handler)

ymodem.release(handler)

ymodem.create(dir_path,file_path)

创建一个 ymodem 处理句柄

参数

传入值类型	解释
string	保存的文件夹路径，默认是”/”
string	强制保存的绝对文件路径，默认是空，如果设置了，就会直接保存在该文件中

返回值

返回值类型	解释
boolean	成功 true，失败 false

例子

```
local handler = ymodem.create("/")
```

---

ymodem.receive(handler, data)

ymodem 接收文件数据并保存

参数

传入值类型	解释
userdata	ymodem 处理句柄
zbuff/string	输入的数据

返回值

返回值类型	解释
boolean	成功 true，失败 false
int	ack 值，需要通过串口/网络等途径返回发送方
int	flag 值，需要通过串口/网络等途径返回发送方，如果有 ack 值则不发送 flag
boolean,	一个文件接收完成 true，传输中 false
boolean,	整个传输完成 true 否则 false

例子

```
-- 注意，数据来源不限，通常是 uart.read 得到 data
no_error,ack,flag,file_done,all_done = ymodem.receive(handler, data)
```

---

ymodem.reset(handler)

重置 ymodem 处理过程

参数

传入值类型　　解释

userdata　　ymodem 处理句柄

返回值

无

例子

```
-- 恢复到初始状态，一般用于接收出错后重置，从而进行下一次接收
ymodem.reset(handler)
```

---

ymodem.release(handler)

释放 ymodem 处理句柄

参数

传入值类型　解释

userdata　　handler

返回值

无

例子

```
ymodem.release(handler)
```

zbuff - c 内存数据操作库

已适配 Air101/Air103 Air601 Air105 ESP32C3 ESP32S3 Air780E/Air700E Air780EP

备注

本页文档由这个文件自动生成。如有错误，请提交 issue 或帮忙修改后 pr，谢谢！

小技巧

本库有专属 demo，点此链接查看 zbuff 的 demo 例子

小技巧

本库还有视频教程，点此链接查看

常量

zbuff.create(length,data,type)

buff:write(para,...)

buff:read(length)

buff:clear(num)

buff:seek(base,offset)

buff:pack(format,val1, val2,...)

buff:unpack(format)

buff:read 类型()

buff:write 类型()

buff:toStr(offset,length)

buff:len()

buff:setFrameBuffer(width,height,bit,color)

buff:pixel(x,y,color)

buff:drawLine(x1,y1,x2,y2,color)

buff:drawRect(x1,y1,x2,y2,color,fill)

buff:drawCircle(x,y,r,color,fill)

buff[n]

buff:free()

buff:resize(n)

buff:copy(start, para,...)

buff:used()

buff:del(offset,length)

buff:query(offset,length,isbigend,issigned,isfloat)

buff:set(start, num, len)

buff:isEqual(start, buff2, start2, len)

常量

常量	类型	解释
zbuff.SEEK_SET	number	以头为基点
zbuff.SEEK_CUR	number	以当前位置为基点
zbuff.SEEK_END	number	以末尾为基点
zbuff.HEAP_AUTO	number	自动申请(如存在 psram 则在 psram 进行申请,如不存在或失败则在 sram 进行申请)
zbuff.HEAP_SRAM	number	在 sram 申请
zbuff.HEAP_PSRAM	number	在 psram 申请

zbuff.create(length,data,type)

创建 zbuff

参数

传入值类型	解释
int	字节数
any	可选参数，number 时为填充数据，string 时为填充字符串
number	可选参数，内存类型，可选：zbuff.HEAP_SRAM(内部 sram,默认) zbuff.HEAP_PSRAM(外部 psram) zbuff.HEAP_AUTO(自动申请,如存在 psram 则在 psram 进行申请,如不存在或失败则在 sram 进行申请) 注意:此项与硬件支持有关

返回值

返回值类型	解释
object	zbuff 对象，如果创建失败会返回 nil

例子

```
-- 创建 zbuff

local buff = zbuff.create(1024) -- 空白的

local buff = zbuff.create(1024, 0x33) --创建一个初值全为 0x33 的内存区域

local buff = zbuff.create(1024, "123321456654") -- 创建，并填充一个已有字符串的内容


-- 创建 framebuff 用的 zbuff

-- zbuff.create({width,height,bit},data,type)

-- table 宽度、高度、色位深度

@int 可选参数，填充数据

@number 可选参数，内存类型，可选：zbuff.HEAP_SRAM(内部 sram,默认) zbuff.HEAP_PSRAM(外部 psram) zbuff.HEAP_AUTO(自动申请,如存在 psram 则在 psram 进行申请,如不存在或失败则在 sram 进行申请) 注意:此项与硬件支持有关

@return object zbuff 对象，如果创建失败会返回 nil

@usage

-- 创建 zbuff

local buff = zbuff.create({128,160,16})--创建一个 128*160 的 framebuff

local buff = zbuff.create({128,160,16},0xf800)--创建一个 128*160 的 framebuff，初始状态红色
```

buff:write(para,…)

zbuff 写数据（从当前指针位置开始；执行后指针会向后移动）

参数

传入值类型	解释
any	写入 buff 的数据，string 时为一个参数，number 时可为多个参数

返回值

返回值类型	解释
number	数据成功写入的长度

例子

```
-- 类 file 的读写操作

local len = buff:write("123") -- 写入数据，指针相应地往后移动，返回写入的数据长度

local len = buff:write(0x1a,0x30,0x31,0x32,0x00,0x01)  -- 按数值写入多个字节数据
```

---

buff:read(length)

zbuff 读数据（从当前指针位置开始；执行后指针会向后移动）

参数

传入值类型	解释
int	读取 buff 中的字节数

返回值

返回值类型	解释
string	读取结果

例子

```
-- 类 file 的读写操作
local str = buff:read(3)
```

---

buff:clear(num)

zbuff 清空数据（与当前指针位置无关；执行后指针位置不变）

参数

传入值类型	解释
int	可选，默认为 0。要设置为的值，不会改变 buff 指针位置

返回值

无

例子

```
-- 全部初始化为 0
buff:clear(0)
```

---

buff:seek(base,offset)

zbuff 设置光标位置（可能与当前指针位置有关；执行后指针会被设置到指定位置）

参数

传入值类型	解释
int	偏移长度
int	where，基点，默认 zbuff.SEEK_SET。zbuff.SEEK_SET：基点为 0 （文件开

传入值

类型	解释
	头），zbuff.SEEK_CUR: 基点为当前位置，zbuff.SEEK_END: 基点为文件尾

返回值

返回值类型	解释
int	设置光标后从 buff 开头计算起的光标的位置

例子

```
buff:seek(0) -- 把光标设置到指定位置
buff:seek(5,zbuff.SEEK_CUR)
buff:seek(-3,zbuff.SEEK_END)
```

---

buff:pack(format,val1, val2,…)

将一系列数据按照格式字符转化，并写入（从当前指针位置开始；执行后指针会向后移动）

参数

传入值类型	解释
string	后面数据的格式（符号含义见下面的例子）
val	传入的数据，可以为多个数据

返回值

返回值类型	解释
int	成功写入的数据长度

例子

```
buff:pack(">IIHA", 0x1234, 0x4567, 0x12,"abcdefg") -- 按格式写入几个数据
-- A string
-- f float
-- d double
-- n Lua number
-- c char
-- b byte / unsignen char
-- h short
-- H unsigned short
-- i int
-- I unsigned int
-- l long
-- L unsigned long
-- < 小端
-- > 大端
-- = 默认大小端
```

---

buff:unpack(format)

将一系列数据按照格式字符读取出来（从当前指针位置开始；执行后指针会向后移动）

参数

传入值类型	解释
string	数据的格式（符号含义见上面 pack 接口的例子）

返回值

返回值类型	解释

返回值类型	解释
int	成功读取的数据字节长度
any	按格式读出来的数据

例子

```
local cnt,a,b,c,s = buff:unpack(">IIHA10") -- 按格式读取几个数据
--如果全部成功读取，cnt 就是 4+4+2+10=20
```

---

buff:read 类型()

读取一个指定类型的数据（从当前指针位置开始；执行后指针会向后移动）

参数

传入值类型	解释
注释	读取类型可为：I8、U8、I16、U16、I32、U32、I64、U64、F32、F64

返回值

返回值类型	解释
number	读取的数据，如果越界则为 nil

例子

```
local data = buff:readI8()
local data = buff:readU32()
```

---

buff:write 类型()

写入一个指定类型的数据（从当前指针位置开始；执行后指针会向后移动）

参数

传入值类型	解释
number	待写入的数据
注释	写入类型可为：I8、U8、I16、U16、I32、U32、I64、U64、F32、F64

返回值

返回值类型	解释
number	成功写入的长度

例子

```
local len = buff:writeI8(10)
local len = buff:writeU32(1024)
```

---

buff:toStr(offset,length)

按起始位置和长度取出数据（与当前指针位置无关；执行后指针位置不变）

参数

传入值类型	解释
int	数据的起始位置（起始位置为 0）,默认值也是 0
int	数据的长度,默认是全部数据

返回值

返回值类型	解释
string	读出来的数据

例子

```
local s = buff:toStr(0,5)--读取开头的五个字节数据

local s = buff:toStr() -- 取出整个 zbuff 的数据

local s = buff:toStr(0, buff:used()) -- 取出已使用的部分，与 buff:query()一样
```

---

buff:len()

获取 zbuff 对象的长度（与当前指针位置无关；执行后指针位置不变）

参数

无

返回值

返回值类型	解释
int	zbuff 对象的长度

例子

```
len = buff:len()len = #buff
```

---

buff:setFrameBuffer(width,height,bit,color)

设置 buff 对象的 FrameBuffer 属性（与当前指针位置无关；执行后指针位置不变）

参数

传入值类型	解释
int	FrameBuffer 的宽度
int	FrameBuffer 的高度
int	FrameBuffer 的色位深度
int	FrameBuffer 的初始颜色

返回值

返回值类型	解释
bool	设置成功会返回 true

例子

```
result = buff:setFrameBuffer(320,240,16,0xffff)
```

---

buff:pixel(x,y,color)

设置或获取 FrameBuffer 某个像素点的颜色（与当前指针位置无关；执行后指针位置不变）

参数

传入值类型	解释
int	与最左边的距离，范围是 0~宽度-1
int	与最上边的距离，范围是 0~高度-1
int	颜色，如果留空则表示获取该位置的颜色

返回值

返回值类型	解释
any	设置颜色时，设置成功会返回 true；读取颜色时，返回颜色的值，读取失败返回 nil

例子

```
rerult = buff:pixel(0,3,0)color = buff:pixel(0,3)
```

---

buff:drawLine(x1,y1,x2,y2,color)

画一条线（与当前指针位置无关；执行后指针位置不变）

参数

传入值类型	解释
int	起始坐标点与最左边的距离，范围是 0~宽度-1
int	起始坐标点与最上边的距离，范围是 0~高度-1
int	结束坐标点与最左边的距离，范围是 0~宽度-1
int	结束坐标点与最上边的距离，范围是 0~高度-1
int	可选，颜色，默认为 0

返回值

返回值类型	解释
bool	画成功会返回 true

例子

```
rerult = buff:drawLine(0,0,2,3,0xffff)
```

---

buff:drawRect(x1,y1,x2,y2,color,fill)

画一个矩形（与当前指针位置无关；执行后指针位置不变）

参数

传入值类型	解释
int	起始坐标点与最左边的距离，范围是 0~宽度-1
int	起始坐标点与最上边的距离，范围是 0~高度-1
int	结束坐标点与最左边的距离，范围是 0~宽度-1
int	结束坐标点与最上边的距离，范围是 0~高度-1

传入值类型	解释
int	可选，颜色，默认为 0
bool	可选，是否在内部填充，默认 nil

返回值

返回值类型	解释
bool	画成功会返回 true

例子

```
rerult = buff:drawRect(0,0,2,3,0xffff)
```

---

buff:drawCircle(x,y,r,color,fill)

画一个圆形（与当前指针位置无关；执行后指针位置不变）

参数

传入值类型	解释
int	**圆心**与最左边的距离，范围是 0~宽度-1
int	**圆心**与最上边的距离，范围是 0~高度-1
int	圆的半径
int	可选，圆的颜色，默认为 0
bool	可选，是否在内部填充，默认 nil

返回值

返回值类型	解释
bool	画成功会返回 true

例子

```
rerult = buff:drawCircle(15,5,3,0xC)
rerult = buff:drawCircle(15,5,3,0xC,true)
```

---

buff[n]

以下标形式进行数据读写（与当前指针位置无关；执行后指针位置不变）

参数

传入值类型	解释
int	第几个数据，以 0 开始的下标（C 标准）

返回值

返回值类型	解释
number	该位置的数据

例子

```
buff[0] = 0xc8
local data = buff[0]
```

---

buff:free()

释放 zbuff 所申请内存 注意：gc 时会自动释放 zbuff 以及 zbuff 所申请内存，所以通常无需调用此函数，调用前请确认您已清楚此函数用处！调用此函数并不会释放掉 zbuff，仅会释放掉 zbuff 所申请的内存，zbuff 需等 gc 时自动释放！！！

参数

无

返回值

无

例子

```
buff:free()
```

---

buff:resize(n)

调整 zbuff 实际分配空间的大小，类似于 realloc 的效果，new = realloc(old, n)，可以扩大或者缩小（如果缩小后 len 小于了 used，那么 used=新 len）

参数

传入值类型	解释
int	新空间大小

返回值

无

例子

```
buff:resize(20)
```

---

buff:copy(start, para,…)

zbuff 动态写数据，类似于 memcpy 效果，当原有空间不足时动态扩大空间

参数

传入值类型	解释
int	写入 buff 的起始位置，如果不为数字，则为 buff 的 used，如果小于 0，则从 used 往前数，-1 = used - 1

传入值类型	解释
any	写入 buff 的数据，string 或 zbuff 者时为一个参数，number 时可为多个参数

返回值

返回值类型	解释
number	数据成功写入的长度

例子

```
local len = buff:copy(nil, "123") -- 类似于 memcpy(&buff[used], "123", 3) used+= 3 从 buff 开始写入数据,指针相应地往后移动

local len = buff:copy(0, "123") -- 类似于 memcpy(&buff[0], "123", 3) if (used < 3) used = 3 从位置 0 写入数据,指针有可能会移动

local len = buff:copy(2, 0x1a,0x30,0x31,0x32,0x00,0x01)  -- 类似于 memcpy(&buff[2], [0x1a,0x30,0x31,0x32,0x00,0x01], 6) if (used < (2+6)) used = (2+6)从位置 2 开始，按数值写入多个字节数据

local len = buff:copy(9, buff2)  -- 类似于 memcpy(&buff[9], &buff2[0], buff2 的 used) if (used < (9+buff2 的 used)) used = (9+buff2 的 used) 从位置 9 开始，合并入 buff2 里 0~used 的内容

local len = buff:copy(5, buff2, 10, 1024)  -- 类似于 memcpy(&buff[5], &buff2[10], 1024) if (used < (5+1024)) used = (5+1024)
```

---

buff:used()

获取 zbuff 里最后一个数据位置指针到首地址的偏移量，来表示 zbuff 内已有有效数据量大小，注意这个不同于分配的空间大小，由于 seek()会改变最后一个数据位置指针，因此也会影响到 used()返回值。

参数

无

返回值

返回值类型	解释
int	有效数据量大小

例子

```
buff:used()
```

---

buff:del(offset,length)

删除 zbuff 0~used 范围内的一段数据，注意只是改变了 used 的值，并不是真的在 ram 里去清除掉数据

参数

传入值类型	解释
int	起始位置 start，默认 0，如果<0 则从 used 往前数，比如 -1 那么 start= used - 1
int	长度 del_len，默认为 used，如果 start + del_len 数值大于 used，会强制调整 del_len = used - start

返回值

无

例子

```
buff:del(1,4) --从位置 1 开始删除 4 个字节数据
buff:del(-1,4) --从位置 used-1 开始删除 4 个字节数据，但是这肯定会超过 used，所以 del_len 会调整为 1，实际上就是删掉了最后一个字节
```

---

buff:query(offset,length,isbigend,issigned,isfloat)

按起始位置和长度 0~used 范围内取出数据，如果是 1,2,4,8 字节，根据后续参数转换成浮点或者整形

参数

传入值类型	解释
int	数据的起始位置（起始位置为 0）
int	数据的长度
boolean	是否是大端格式，如果为 nil，则不会转换，直接字节流输出
boolean	是否是有符号的，默认为 false
boolean	是否是浮点型，默认为 false

返回值

返回值类型	解释
string	读出来的数据

例子

```
local s = buff:query(0,5)--读取开头的五个字节数据
```

---

buff:set(start, num, len)

zbuff 的类似于 memset 操作，类似于 memset(&buff[start], num, len)，当然有 ram 越界保护，会对 len 有一定的限制

参数

传入值类型	解释
int	可选，开始位置，默认为 0,
int	可选，默认为 0。要设置为的值

传入值类型	解释
int	可选，长度，默认为全部空间，如果超出范围了，会自动截断

返回值

无

例子

```
-- 全部初始化为 0
buff:set() --等同于 memset(buff, 0, sizeof(buff))
buff:set(8) --等同于 memset(&buff[8], 0, sizeof(buff) - 8)
buff:set(0, 0x55) --等同于 memset(buff, 0x55, sizeof(buff))
buff:set(4, 0xaa, 12) --等用于 memset(&buff[4], 0xaa, 12)
```

---

buff:isEqual(start, buff2, start2, len)

zbuff 的类似于 memcmp 操作，类似于 memcmp(&buff[start], &buff2[start2], len)

参数

传入值类型	解释
int	可选，开始位置，默认为 0,
zbuff	比较的对象
int	可选，比较的对象的开始位置，默认为 0
int	比较长度

返回值

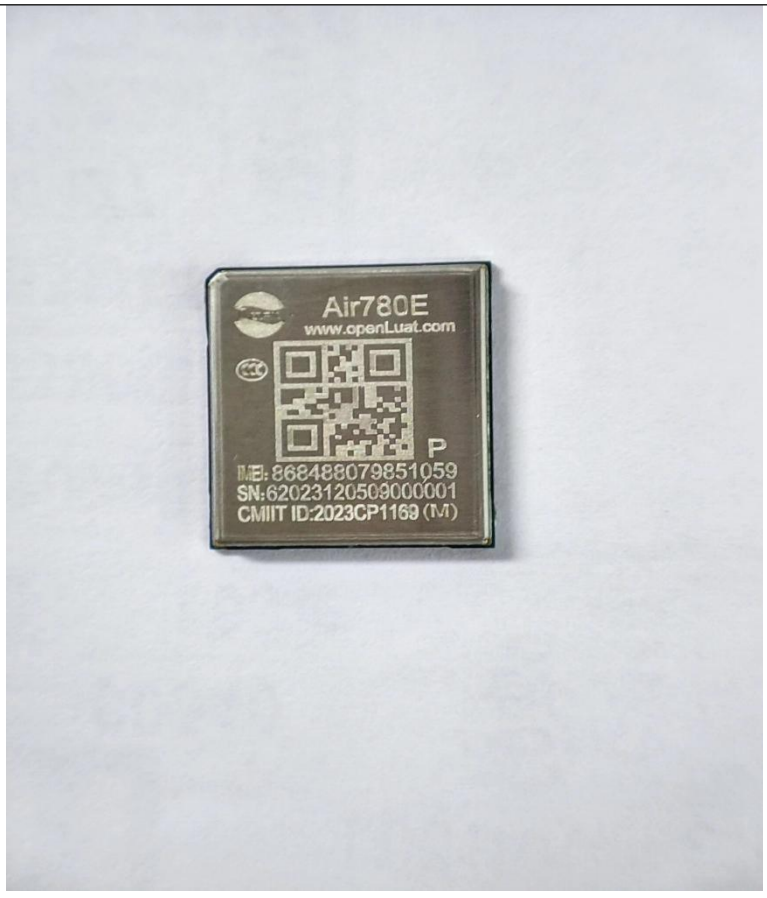
返回值类型	解释
boolean	true 相等，false 不相等
int	相等返回 0，不相等返回第一个不相等位置的序号

例子

```
local result, offset = buff:isEqual(1, buff2, 2, 10) --等同于 memcmp(&buff[1], &buff2[2], 10)
```

十九 高低温射频传导测试报告

测试总览

产品名称	合宙通信模组
单位	上海合宙通信科技有限公司
地址	上海城置广场 611
待测设备型号	Air780EP
设备图片	
测试环境	温度（℃）：-40～85 相对湿度（%RH）： 45～62 大气压力（kPa）： 100.2～102.3

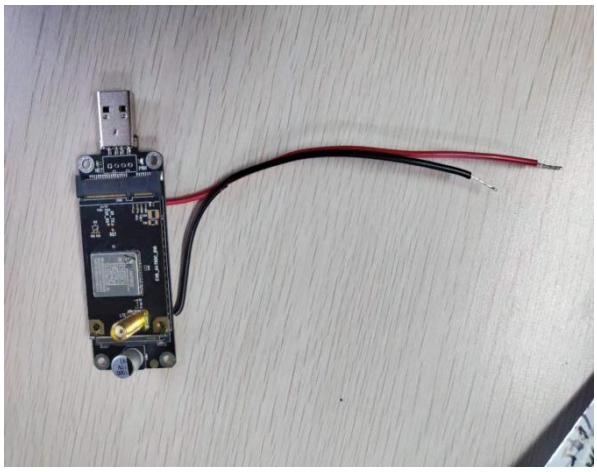
测试设备	综测仪器：CMW500 程控电源：安捷伦 66319D 高低温箱：庆声
测试软件	CMWrun
测试工程师	何祖秒
完成日期	2024/3/11
检验标准（或者依据）	3GPP 36.521
检验结论	测试样品经检验，4G 传导射频所检项目符合 3GPP 36.521 标准

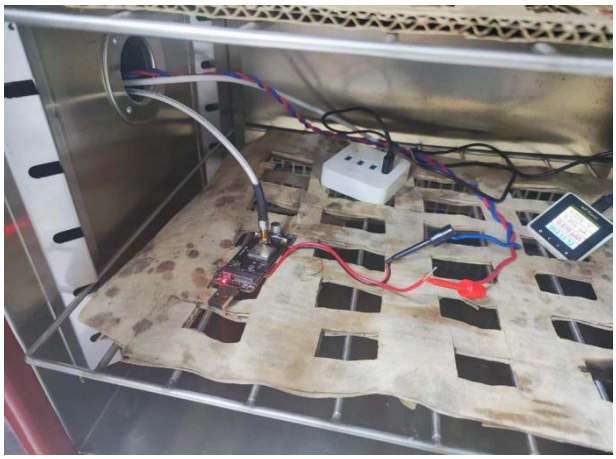
测试设备

测试设备	型号	图片
综测仪	CW500	
测控电源	安捷伦 66319D	

高低温箱	庆声	

测试环境





运行
停止
照明

运行

定值运行
温度
℃
85.00
SV:85.00
MV:5.1
湿度
%
---.-
SV:OFF
运行时间:01:56:17

测试项目

测试条件： 带宽 10M Hz			
	测试项目	**测试标准**	**测试结果**
1	UE Maximum Output Power	＞19.3dBm~＜24.7dBm	Pass
2	Minimum Output Power	＜-39dBm	Pass
3	Frequency Error	≤ 0.1 PPM	Pass
4	Error Vector Magnitude (EVM)	＜12.50%dBm	Pass
5	Adjacent Channel Leakage power Ratio	＞32.2dBc	Pass
6	Reference sensitivity level	@10M＜-96.3dBm	Pass

测试数据

注：极限温度下不保证所有射频指标都能通过 3GPP 测试标准；此报告仅展示 MaxOutPower 及 Sensitivity 测试数据，其它内容请查看附件。

最大输出功率@频段 B1

高温 85 度

6.2.2 Maximum Output Power	Limit Low	Limit High	Measured	Unit	Status
6.2.2 Maximum Output Power @ Band1					
UE Maximum OutputPower:@ULCH: 18050, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.87	dBm	Passed

UE Maximum OutputPower:@ULCH: 18050, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.76	dBm	Passed
UE Maximum OutputPower:@ULCH: 18300, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.29	dBm	Passed
UE Maximum OutputPower:@ULCH: 18300, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.16	dBm	Passed
UE Maximum OutputPower:@ULCH: 18550, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	23.38	dBm	Passed
UE Maximum OutputPower:@ULCH: 18550, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:HIGH)	19.30	24.70	23.21	dBm	Passed

低温-40 度

6.2.2 Maximum Output Power	Limit Low	Limit High	Measured	Unit	Status
6.2.2 Maximum Output Power @ Band1					
UE Maximum OutputPower:@ULCH: 18050, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.54	dBm	Passed

UE Maximum OutputPower:@ULCH: 18050, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.59	dBm	Passed
UE Maximum OutputPower:@ULCH: 18300, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.94	dBm	Passed
UE Maximum OutputPower:@ULCH: 18300, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.94	dBm	Passed
UE Maximum OutputPower:@ULCH: 18550, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	24.52	dBm	Passed
UE Maximum OutputPower:@ULCH: 18550, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:HIGH)	19.30	24.70	24.38	dBm	Passed

射频灵敏度@射频 B1

高温 85 度

7.3 Reference Sensitivity Level	Limit Low	Limit High	Measured	Unit	Status
7.3 Reference Sensitivity Level @ Band1@NS_01					
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 18050, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	96.02	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 18300, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	99.53	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 18550, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	99.96	%	Passed

低温-40 度

7.3 Reference Sensitivity Level	Limit Low	Limit High	Measured	Unit	Status
7.3 Reference Sensitivity Level @ Band1@NS_01					
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 18050, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	99.87	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 18300, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	96.58	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 18550, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed

最大输出功率@频段 B3

高温 85 度

6.2.2 Maximum Output Power	Limit Low	Limit High	Measured	Unit	Status
6.2.2 Maximum Output Power @ Band3					
UE Maximum OutputPower:@ULCH: 19250, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	17.80	24.70	23.15	dBm	Passed
UE Maximum OutputPower:@ULCH: 19250, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	23.25	dBm	Passed
UE Maximum OutputPower:@ULCH: 19250, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	17.80	24.70	23.05	dBm	Passed
UE Maximum OutputPower:@ULCH: 19575, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	22.86	dBm	Passed
UE Maximum OutputPower:@ULCH: 19575, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	23.39	dBm	Passed
UE Maximum OutputPower:@ULCH: 19575, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	22.70	dBm	Passed
UE Maximum OutputPower:@ULCH: 19900, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.27	dBm	Passed
UE Maximum OutputPower:@ULCH: 19900, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	17.80	24.70	22.14	dBm	Passed
UE Maximum OutputPower:@ULCH: 19900, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.06	dBm	Passed

低温-40 度

6.2.2 Maximum Output Power	Limit Low	Limit High	Measured	Unit	Status
6.2.2 Maximum Output Power @ Band3					
UE Maximum OutputPower:@ULCH: 19250, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	17.80	24.70	23.76	dBm	Passed
UE Maximum OutputPower:@ULCH: 19250, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	23.84	dBm	Passed
UE Maximum OutputPower:@ULCH: 19250, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	17.80	24.70	23.43	dBm	Passed
UE Maximum OutputPower:@ULCH: 19575, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.98	dBm	Passed
UE Maximum OutputPower:@ULCH: 19575, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	23.82	dBm	Passed
UE Maximum OutputPower:@ULCH: 19575, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.73	dBm	Passed
UE Maximum OutputPower:@ULCH: 19900, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.93	dBm	Passed
UE Maximum OutputPower:@ULCH: 19900, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	17.80	24.70	24.36	dBm	Passed
UE Maximum OutputPower:@ULCH: 19900, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.80	dBm	Passed

射频灵敏度@射频 B3

高温 85 度

LTE_3GPP_10MHz n/a #868488079851059 03/11/2024 14:13:33

7.3 Reference Sensitivity Level @ Band3@NS_01					
Throughput: DL P_ref -93.3 dBm; 2664 Frames:@ULCH: 19250, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	99.96	%	Passed
Throughput: DL P_ref -93.3 dBm; 2664 Frames:@ULCH: 19575, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -93.3 dBm; 2664 Frames:@ULCH: 19900, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed

低温-40 度

LTE_3GPP_10MHz n/a #868488079851059 03/11/2024 18:12:03

7.3 Reference Sensitivity Level @ Band3@NS_01					
Throughput: DL P_ref -93.3 dBm; 2664 Frames:@ULCH: 19250, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	99.83	%	Passed
Throughput: DL P_ref -93.3 dBm; 2664 Frames:@ULCH: 19575, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -93.3 dBm; 2664 Frames:@ULCH: 19900, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed

最大输出功率@频段 B5

高温 85 度

6.2.2 Maximum Output Power	Limit Low	Limit High	Measured	Unit	Status
6.2.2 Maximum Output Power @ Band5					
UE Maximum OutputPower:@ULCH: 20450, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	22.50	dBm	Passed
UE Maximum OutputPower:@ULCH: 20450, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	22.55	dBm	Passed
UE Maximum OutputPower:@ULCH: 20525, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.18	dBm	Passed
UE Maximum OutputPower:@ULCH: 20525, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	22.92	dBm	Passed
UE Maximum OutputPower:@ULCH: 20600, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	22.60	dBm	Passed
UE Maximum OutputPower:@ULCH: 20600, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:HIGH)	19.30	24.70	22.44	dBm	Passed

射频灵敏度@射频 B5

高温 85 度

7.3 Reference Sensitivity Level	Limit Low	Limit High	Measured	Unit	Status
7.3 Reference Sensitivity Level @ Band5@NS_01					
Throughput: DL P_ref -94.3 dBm; 2664 Frames:@ULCH: 20450, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 25;	95.00	---	99.91	%	Passed
Throughput: DL P_ref -94.3 dBm; 2664 Frames:@ULCH: 20525, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 25;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -94.3 dBm; 2664 Frames:@ULCH: 20600, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 25;	95.00	---	100.00	%	Passed

低温-40 度

7.3 Reference Sensitivity Level	Limit Low	Limit High	Measured	Unit	Status
7.3 Reference Sensitivity Level @ Band5@NS_01					
Throughput: DL P_ref -94.3 dBm; 2664 Frames:@ULCH: 20450, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 25;	95.00	---	99.87	%	Passed
Throughput: DL P_ref -94.3 dBm; 2664 Frames:@ULCH: 20525, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 25;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -94.3 dBm; 2664 Frames:@ULCH: 20600, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 25;	95.00	---	100.00	%	Passed

最大输出功率@频段 B8

高温 85 度

6.2.2 Maximum Output Power	Limit Low	Limit High	Measured	Unit	Status
6.2.2 Maximum Output Power @ Band8					
UE Maximum OutputPower:@ULCH: 21500, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	17.80	24.70	22.73	dBm	Passed

UE Maximum OutputPower:@ULCH: 21500, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	23.01	dBm	Passed
UE Maximum OutputPower:@ULCH: 21500, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	17.80	24.70	22.44	dBm	Passed
UE Maximum OutputPower:@ULCH: 21625, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	22.60	dBm	Passed
UE Maximum OutputPower:@ULCH: 21625, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	23.02	dBm	Passed
UE Maximum OutputPower:@ULCH: 21625, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	22.41	dBm	Passed
UE Maximum OutputPower:@ULCH: 21750, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	22.80	dBm	Passed
UE Maximum OutputPower:@ULCH: 21750, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	17.80	24.70	22.63	dBm	Passed
UE Maximum OutputPower:@ULCH: 21750, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	22.48	dBm	Passed

射频灵敏度@射频 B8

高温 85 度

7.3 Reference Sensitivity Level	Limit Low	Limit High	Measured	Unit	Status
7.3 Reference Sensitivity Level @ Band8@NS_01					
Throughput: DL P_ref -93.3 dBm; 2664 Frames:@ULCH: 21500, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 25;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -93.3 dBm; 2664 Frames:@ULCH: 21625, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 25;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -93.3 dBm; 2664 Frames:@ULCH: 21750, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 25;	95.00	---	100.00	%	Passed

低温-40 度

7.3 Reference Sensitivity Level	Limit Low	Limit High	Measured	Unit	Status
7.3 Reference Sensitivity Level @ Band8@NS_01					
Throughput: DL P_ref -93.3 dBm; 2664 Frames:@ULCH: 21500, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 25;	95.00	---	99.83	%	Passed
Throughput: DL P_ref -93.3 dBm; 2664 Frames:@ULCH: 21625, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 25;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -93.3 dBm; 2664 Frames:@ULCH: 21750, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 25;	95.00	---	100.00	%	Passed

最大输出功率@频段 B34

高温 85 度

6.2.2 Maximum Output Power	Limit Low	Limit High	Measured	Unit	Status
6.2.2 Maximum Output Power @ Band34					
UE Maximum OutputPower:@ULCH: 36250, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.70	dBm	Passed
UE Maximum OutputPower:@ULCH: 36250, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.75	dBm	Passed
UE Maximum OutputPower:@ULCH: 36275, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.70	dBm	Passed
UE Maximum OutputPower:@ULCH: 36275, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.61	dBm	Passed
UE Maximum OutputPower:@ULCH: 36300, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	23.69	dBm	Passed
UE Maximum OutputPower:@ULCH: 36300, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:HIGH)	19.30	24.70	23.57	dBm	Passed

射频灵敏度@射频 B34

高温 85 度

LTE_3GPP_10MHz n/a #868488079851059 03/11/2024 14:13:33

7.3 Reference Sensitivity Level @ Band34@NS_01					
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 36250, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 36275, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 36300, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	99.91	%	Passed

低温-40 度

LTE_3GPP_10MHz n/a #868488079851059 03/11/2024 18:12:03

7.3 Reference Sensitivity Level @ Band34@NS_01					
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 36250, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 36275, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 36300, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed

最大输出功率@频段 B38

高温 85 度

6.2.2 Maximum Output Power	Limit Low	Limit High	Measured	Unit	Status
6.2.2 Maximum Output Power @ Band38					
UE Maximum OutputPower:@ULCH: 37800, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.34	dBm	Passed
UE Maximum OutputPower:@ULCH: 37800, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.29	dBm	Passed
UE Maximum OutputPower:@ULCH: 38000, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.36	dBm	Passed
UE Maximum OutputPower:@ULCH: 38000, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.20	dBm	Passed
UE Maximum OutputPower:@ULCH: 38200, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	23.16	dBm	Passed
UE Maximum OutputPower:@ULCH: 38200, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:HIGH)	19.30	24.70	23.12	dBm	Passed

射频灵敏度@射频 B38

高温 85 度

7.3 Reference Sensitivity Level	Limit Low	Limit High	Measured	Unit	Status
7.3 Reference Sensitivity Level @ Band38@NS_01					
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 37800, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 38000, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 38200, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed

低温-40 度

7.3 Reference Sensitivity Level	Limit Low	Limit High	Measured	Unit	Status
7.3 Reference Sensitivity Level @ Band38@NS_01					
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 37800, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	99.59	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 38000, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	97.03	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 38200, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed

最大输出功率@频段 B39

高温 85 度

6.2.2 Maximum Output Power	Limit Low	Limit High	Measured	Unit	Status
6.2.2 Maximum Output Power @ Band39					
UE Maximum OutputPower:@ULCH: 38300, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.14	dBm	Passed

UE Maximum OutputPower:@ULCH: 38300, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.16	dBm	Passed
UE Maximum OutputPower:@ULCH: 38450, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.36	dBm	Passed
UE Maximum OutputPower:@ULCH: 38450, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.37	dBm	Passed
UE Maximum OutputPower:@ULCH: 38600, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	23.23	dBm	Passed
UE Maximum OutputPower:@ULCH: 38600, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:HIGH)	19.30	24.70	23.23	dBm	Passed

射频灵敏度@射频 B39

高温 85 度

7.3 Reference Sensitivity Level	Limit Low	Limit High	Measured	Unit	Status
7.3 Reference Sensitivity Level @ Band39@NS_01					
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 38300, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 38450, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 38600, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	99.91	%	Passed

低温-40 度

7.3 Reference Sensitivity Level	Limit Low	Limit High	Measured	Unit	Status
7.3 Reference Sensitivity Level @ Band39@NS_01					
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 38300, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	99.83	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 38450, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 38600, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed

最大输出功率@频段 B40

高温 85 度

6.2.2 Maximum Output Power	Limit Low	Limit High	Measured	Unit	Status
6.2.2 Maximum Output Power @ Band40					
UE Maximum OutputPower:@ULCH: 38700, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.11	dBm	Passed
UE Maximum OutputPower:@ULCH: 38700, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.07	dBm	Passed
UE Maximum OutputPower:@ULCH: 39150, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.23	dBm	Passed
UE Maximum OutputPower:@ULCH: 39150, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.21	dBm	Passed
UE Maximum OutputPower:@ULCH: 39600, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	23.48	dBm	Passed
UE Maximum OutputPower:@ULCH: 39600, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:HIGH)	19.30	24.70	23.43	dBm	Passed

射频灵敏度@射频 B40

高温 85 度

7.3 Reference Sensitivity Level	Limit Low	Limit High	Measured	Unit	Status
7.3 Reference Sensitivity Level @ Band40@NS_01					
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 38700, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed

Page 74 of 84

LTE_3GPP_10MHz n/a #868488079851059 03/11/2024 14:13:33

Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 39150, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 39600, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	99.73	%	Passed

低温-40 度

7.3 Reference Sensitivity Level	Limit Low	Limit High	Measured	Unit	Status
7.3 Reference Sensitivity Level @ Band40@NS_01					
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 38700, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed

Page 74 of 84

LTE_3GPP_10MHz n/a #868488079851059 03/11/2024 18:12:03

Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 39150, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -96.3 dBm; 2664 Frames:@ULCH: 39600, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed

最大输出功率@频段 B41

高温 85 度

6.2.2 Maximum Output Power	Limit Low	Limit High	Measured	Unit	Status
6.2.2 Maximum Output Power @ Band41					
UE Maximum OutputPower:@ULCH: 40290, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.13	dBm	Passed
UE Maximum OutputPower:@ULCH: 40290, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	23.00	dBm	Passed
UE Maximum OutputPower:@ULCH: 40290, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.03	dBm	Passed
UE Maximum OutputPower:@ULCH: 40620, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	23.18	dBm	Passed
UE Maximum OutputPower:@ULCH: 40620, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	23.00	dBm	Passed
UE Maximum OutputPower:@ULCH: 40620, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	23.09	dBm	Passed
UE Maximum OutputPower:@ULCH: 41190, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:LOW)	19.30	24.70	22.91	dBm	Passed
UE Maximum OutputPower:@ULCH: 41190, BW: 10 MHz; UL_MOD_RB: QPSK, 1 (RB_Pos:HIGH)	19.30	24.70	22.62	dBm	Passed
UE Maximum OutputPower:@ULCH: 41190, BW: 10 MHz; UL_MOD_RB: QPSK, 12 (RB_Pos:LOW)	19.30	24.70	22.77	dBm	Passed

射频灵敏度@射频 B41

高温 85 度

7.3 Reference Sensitivity Level	Limit Low	Limit High	Measured	Unit	Status
7.3 Reference Sensitivity Level @ Band41@NS_01					
Throughput: DL P_ref -94.3 dBm; 2664 Frames:@ULCH: 40290, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	96.08	%	Passed
Throughput: DL P_ref -94.3 dBm; 2664 Frames:@ULCH: 40620, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -94.3 dBm; 2664 Frames:@ULCH: 41190, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	99.91	%	Passed

低温-40 度

7.3 Reference Sensitivity Level	Limit Low	Limit High	Measured	Unit	Status
7.3 Reference Sensitivity Level @ Band41@NS_01					
Throughput: DL P_ref -94.3 dBm; 2664 Frames:@ULCH: 40290, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed
Throughput: DL P_ref -94.3 dBm; 2664 Frames:@ULCH: 40620, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	99.57	%	Passed
Throughput: DL P_ref -94.3 dBm; 2664 Frames:@ULCH: 41190, BW: 10 MHz; DL_MOD_RB: QPSK, 50; UL_MOD_RB: QPSK, 50;	95.00	---	100.00	%	Passed

总览(高温 85 度)

Summary:

Test Start Time:	03/11/2024 14:13:33
Test End Time:	03/11/2024 14:29:43

Place:_____ Date: _____ Person responsible: _____

LTE_3GPP_10MHz n/a #868488079851059 03/11/2024 14:13:33

Total Test Time:	00:16:10
Weighted Test Time:	00:16:10
Tests Passed:	1498
Tests Failed:	0
Number of Tests:	1498
Errors:	0
Warnings:	0

总览(低温-40 度)

Summary:

Test Start Time:	03/11/2024 18:12:03
Test End Time:	03/11/2024 18:28:18

Place:_____ Date: _____ Person responsible: _____

LTE_3GPP_10MHz n/a #868488079851059 03/11/2024 18:12:03

Total Test Time:	00:16:15
Weighted Test Time:	00:16:15
Tests Passed:	1464
Tests Failed:	34
Number of Tests:	1498
Errors:	0
Warnings:	0

二十 合宙创业故事：我的物联网大学（已完结）

本篇创业故事可通过点击右方链接在线阅读：我的物联网大学

也可直接阅读本章节以下内容：

1 迷茫的前三年

壹 业务的困境

有数据统计，中国企业的死亡率，当年死亡率为 1.6%，第二年死亡率为 6.3%，第三年死亡率为 9.5%；之后的 3-7 年，是企业死亡的高发期，有 60% 的企业会在 5 年内死掉。

合宙是 2014 年 5 月成立的。

到了 2017 年，是合宙的第 4 年，恰好处在了死亡概率最高的最困难期。

2014-2017 年这三年，合宙的产品规划，经历了无线固话模块、户户通定位模块、儿童手表品牌、GPS tracker 整机、物联网产品方案定制这几个阶段。

除了无线固话模块，和户户通定位模块能够勉强维持之外，其他业务都是亏损。

这三年里面，合宙的团队对于客户需求，几乎是有求必应，但是几乎都伴随着客户逐渐式微，导致业务无法持续。

关注客户需求没有错，但是仅仅关注客户需求，不去洞察客户需求，在中国的市场，是没有任何的前景可言的。

进入 2016 下半年，无线固话业务和户户通定位模块业务，也进入了尾声，几乎没有订单了。

这也意味着，合宙几乎没有什么像样的主营业务了。

作为合宙的创始人和 CEO，每个难以入眠的夜晚，我盯着窗棂的微光，无数次问自己同一个问题：

接下来怎么办？

问题就象落入无底的黑暗深渊，跌宕着回声，每一声都狠狠撞击困惑的心……直到，真的看到世界的微光：Tomorrow is another day.

贰 团队信任危机

福无双至，祸不单行。
这简直就是世界运行的客观规律。

一般来说，在企业困难时期，团队往往能够团结一心，一致前进。而一旦赚钱，就会出现分赃不均的猜疑，企业出现信任危机。

但是，合宙的信任危机，在还没盈利的时候，提前出现了。

合宙创立的股权结构，是 CEO 70%，其他三位高管陆相成、李炜镪、刘乐安各 10%。后来，又加入了吴永。

陆相成、李炜镪、刘乐安，都是龙尚的老员工，跟我在一起合作超过 10 年；吴永是艾特维的创始员工之一，跟我在一起也合作超过了 7 年。

在合宙运营三年的时候，销售额还是能做到每年几千万的。而合宙的团队成员，只有 30 人不到。

很自然的，各位创始人认为合宙是盈利的。

可是，由于战略上的失误，虽然有盈利的产品，却因更多的投入，导致毛利消耗殆尽，几乎每年都没有盈利过。

而糟糕的计划管理和供应链管理（也可以说是没有管理），也使得硬件的呆滞料库存进一步吃掉了毛利。

但是，合宙一贯以来的运营模式，都是 CEO 一手遮天，其他人按照 CEO 的指令执行。

CEO 之外的所有人，得不到相对充分的信息量，一直以为合宙的运行起码不会太糟糕。

这就产生了麻烦。

2017 年的某一天，快言快语的李炜镪质问我：“我们的财务状况到底是怎么样的？你是不是在中饱私囊？”

会议室如同瞬间注入制冷剂，其他创始人只顾各自低着头，若有所思，无人言语。

突如其来的沉默，反倒让现场的气氛有如乌云暗涌，沉闷压抑到几乎无法呼吸，我深切感受到沉默背后同样的疑问……

那一刻我满腹委屈又无从说起，只觉得胸口堵得慌：
过去几年所有的压力自己扛着，现在收获的却是满满的质疑？

事业的不顺利，伙伴的不信任，双重的挫败在同一时间重击了我，心里有个人对着我大喊：你呀，怎么这么失败？？！！！

片刻之后，我勉强着故作镇定地表态，会尽快梳理前三年的运营过程，给大家一个明确的交代。然后，草草地结束了这次会议。

众人离去，我独自颓然，思绪万千……

花了两天时间，我慢慢平静下来，逐渐意识到：作为CEO，不能受控于自己的情绪。心里那些所谓的委屈与纠结，又算得了什么？这些都不过是一时感受，而格局才真正关乎未来。

作为一个成熟的创业者，应该跳出自己编织的郁闷牢笼。趁着这个机会梳理过往的经营，可能会更重要。

接下来的几天，我尝试着站在合伙人的立场考虑问题，心情打开了一片亮光，虽谈不上豁然开朗，但也对自己重燃希望。

通过梳理过往三年的经营过程，详细解释为什么我们的业务一直不赚钱；
向其他合伙人坦诚自己经营能力的不足，希望其他几位创始人能够谅解耽误的三年时光。
同时，从现在开始，采用集体决策的管理方式，做到责任共担、分工明确、透明经营。
经过几番从来没有过的直击心灵的会议，我们的创始人团队初步达成了充分的谅解和共识，**无论前路多么艰难，创始人团队表示都会艰苦与共，共同承担。**
今后如果成功了的话，也将会通过清晰的股权和激励制度，透明化的分享收益。
如果失败了，也应该享受这个拼搏的过程。

今天回过头来看，信任危机在这个时间节点的产生，对合宙来说是非常幸运的；
如果这个时候不产生质疑，再遗留到后续的业务阶段的话，成本将会指数级提升，甚至可能会造成团队崩溃的结果。
我很幸运，拥有一个同心同德的创始团队。

叁 关于公司的对话

有个好朋友名叫罗成，开了个小公司，有五六个员工，小有利润。

有一天，他突然问我：“老秦，我的小股东曾斌，股份是我送给他的。他现在想出钱，这是好事还是坏事？”

我一愣：“何出此言？怎么可能是坏事？”

罗成面露难色的说：“他出了钱，我岂不是要公开账目和利润明细了？”

我再次愣住：“难道你还能不公开？”

小伙笑得很勉强：“我想把利润大头留给自己，给他们少分点钱就好了吧？毕竟，这公司是我的啊。”

“你的意思是，曾斌不出钱，你给他干股，这样你少分钱给他，你也没有心理负担对吧？”

罗成说：“对啊，我就是这个意思。”

还好，我刚刚参加了一个培训，这个培训是合宙合作的律师事务所主持的关于公司法的培训。

于是，我当仁不让的充当了一次临时老师，把我们律师的话透传了一下：

“公司制度可能是人类最伟大的发明之一。

公司就是一个法人个体。

股东的角色，管理人的角色，只是在承担职责。

公司的命运，跟个人的命运，是不同的，是相互独立的。

而中国人往往很自然的把公司跟个人的命运绑定在一起，这是不合理的，也是违反公司法的初衷的。”

罗成说：“你的律所是哪个啊，能不能我也合作一下？”

合宙的创始人信任危机解决了，但是公司的业务还没方向。

形势依然严峻。

我每天还是问自己相同的问题：接下来怎么办？赚钱的业务在哪里？

可以说每天都是食不甘味，夜不能寐。

某天，一位同事关切地问我：“秦总，你这脸色，没事儿吧~”

我淡然回了句没事儿啊，直到去洗手间才猛然发觉——镜中惊现一张“包公”脸。

不同的是，电视剧中的包公眼神明厉睿智，额间月芽威严。而我，面色黑青，眼神与额头都写满不知所终的小问号……

2 Luat 的出世

壹　启动火种

有一位软件行业的大神，名字叫做许小刚。

小刚是一位憨厚的年轻的码农，嵌入式、后端、前端，无所不能，是一个很牛的物联网全栈工程师，也是一家物联网软件公司的创始人兼 CEO。

有次跟我、老陆、小强闲聊的时候，小刚憨憨的笑着说：“你们应该开源你们的那一套 Lua 开发应用的宝贝，并且给他取个名字。”

一直以来，合宙的应用开发工程师，喜欢用 Lua 语言来做定制产品的应用逻辑开发。

无论是儿童手表，还是 GPS Tracker，还是给客户做其他的物联网产品定制，一直使用的都是 Lua 语言，**以追求开发和维护的最低人员成本，以及需求迭代的高效率。**

“开放出来有什么好处？”老陆很警觉的问。

“至于有什么收益，我也不清楚。反正开放出来总是没错的，只有好处，没什么坏处的。”小刚依旧憨笑着说，并且笑的越来越灿烂了。

“不过，至少你们得给他先起个名字，才能开源。”这次小刚说话的口吻，好像是教导三个 6 岁的孩子。

我们三个带着满肚子的疑惑，回到了公司。

反正也没什么产品方向，先听小刚的吧，起码这人够憨厚，不会害人的。

但是起名这事情，对一群技术男而言，着实有点为难，几番讨论也没有合意的。

终于一次头脑风暴，小强灵光一闪：“我们用的是 Lua 语言，AT 指令是模块的象征，就叫做 Luat 吧”。

所有人一致同意。

工程师的认同，只需要一个简单的逻辑。

就这样，合宙启动了 Luat 开放平台，以及依托于 Luat 平台的 GPRS 通信模块业务。

说起来，合宙团队从 2006 在龙尚开始就涉足通信模块业务，只不过以前做的都是无线固话垂直行业的模块，类似于目前广和通从事的笔电的上网模块，深耕一个行业，不做通用行业的覆盖，所以在物联网通信行业一直默默无闻。

因此，启动 GPRS 通信模块，相当于是合宙团队承认了自己的不足，回归到模块的老本行。

虽然是模块老本行， 但是却进入了通用模块这么一个全新的市场。**这个市场，是合宙团队完全陌生的。**

标准模块市场在 2016 年的两个主流玩家是 SIMCom 和移远，这两个厂商的目标是物联网市场全覆盖。其他的厂家是局部覆盖，同时也往全行业覆盖去努力。

合宙作为新进入的玩家，几乎是没有任何的客户积累的，比起刘姥姥进入大观园，还差了两截。

最起码，刘姥姥还能看到大观园的具体形象。

而合宙，连客户群是谁都不知道，更别提注重客户体验了。

直到现在，合宙的很多同学依然记得两件事：

第一个事情，第一款 Luat 模块的外观：

这个模块的外观，跟合宙当时的认知特别匹配：**不懂产品设计，不懂市场需求，不知道客户需要什么。**

好在，迈出了第一步。在线上开始销售这款模块，勇敢地展示了自己。

虽然浅薄，虽然丑陋，

但是，又有谁的青春不傻逼呢？

第二个事情，Luat 模块的第一个客户：

第一个 Luat 模块订单，江西省一个县城的客户给的。而这个客户的公司，员工很可能当时只有他一个。

客户收到货的时候，已经是春节前夕的腊月二十六。客户很兴奋地在技术群说，**终于发现了一个能用脚本开发应用的模块，太棒了！**

接下来的几天，客户一直非常活跃的跟合宙的工程师聊怎么写代码。

合宙的团队也很兴奋。

从腊月二十六，中间除夕夜也不间断，一直到春节期间，都在 QQ 群里面跟客户聊天，真是相见恨晚。

在海聊的的这一个星期里面， 客户完成了产品的软件开发。

是的，使用的就是 Luat 的脚本开发。

这个客户的公司目前有多少员工，我们依然不清楚；

我们只知道跟合宙合作一年后，他的座驾，由奇瑞换成了宝马。

而且，在随后的两年内，涌现了很多厂商抄袭他的产品，从而使合宙的模块莫名其妙的垄断了这个细分行业。

这客户是 94 年的年轻小伙，典型的小镇青年。

直到今天， 这位小镇青年企业家，仍然是 Luat 的忠实用户，以及合宙最好的朋友。

贰 通过 Luat 认识市场

春节过后，不断的有人购买合宙的模块，几乎每天都能发出几单样品。

因为那个时候，GPRS 模块这个品类，合宙是唯一的行业卖家。合宙 Luat 的 GPRS 模块，几乎独占了淘宝的全部流量。

这个情况， 大概持续了半年。

半年之后，淘宝逐渐有其他的卖家开始介入，但是也大多是模块厂家的代理商，对合宙几乎没有构成太大威胁。

对合宙的发展造成威胁的，都是合宙自己。

购买了样品的用户，经过两三个月的测试，有一部分开始能够成百上千的采购模块；

也有客户，开始上门谈价，咨询几十万片模块，能够给到什么样的价格支持；

更多的客户，以为合宙就是一家淘宝店。因为不放心，登门拜访，想查看是否真的是一家正规公司。

无论如何，合宙模块的销量，慢慢的、稳定的，开始在爬坡了。

由最初的样品销售，收款几十块上百块，逐渐发展到月销售额过百万，甚至几百万了。

每天查看 QQ 群增加的企业客户名称，回复群里客户提出的技术问题。看到客户批量提货之后心中暗生喜悦，也不由思考：

Luat 的用户体验其实并不见得很好，问题也挺多，为什么会有这么多用户很支持呢？

合宙团队对这个问题，还是有一点清晰的认知。

并不是 Luat 撞大运，恰巧有了这么多客户，而是通信模块这个产业，实在是有接近 20 年没什么技术上的进步，而 Luat 踩对了这个技术进步的节奏，迎合了市场的潜在需求。

为什么这么说呢？

接下来我们将详细讲解通信产业的技术发展。

叁 神奇又辉煌的 AT 指令集

很久以来，通信模块的开发界面是 AT 指令集。

客户设备的主控 MCU，按照约定的格式，发送 AT 指令给通信模块。模块执行通信服务，并应答结果，完成通信功能。

要实现这么个过程，主控 MCU 需要实现一个软件状态机，来维护多个 AT 指令的异步通信状态，这对于 MCU 的编程能力是有一定要求的，并且 CPU 的价格也往往不菲。

但是直到今天，AT 仍然是各个模组厂家的主流使用方式之一。

为什么会是这个局面呢？

先从 AT 指令的来源说起。

在 1960 年代，互联网刚刚兴起的时候，网虫们通过计算机接入 Modem（一般是电话线）上网，计算机跟 Modem 的接口是 RS-232 串口。

这个时候，计算机很贵，Modem 也很贵，上网费也很贵。贵到什么程度呢？一般美国人家里用不起，只能公款消费才能上得起网。

中国呢？

中国要等到 90 年代才能上网，现在还是 60 年代。

因为什么都贵， 所以 RS-232 的支持都很完善，大多数是 9 针方式的通信。支持 9 针的串口，数据线跟信号线定义的很完善。通信之前，会先用信号线做好握手，然后再传输数据，要断线的时候，也是信号线做好断开的协商，然后再断开数据。

一切都是那么的和谐。

到了美国的 70 年代中期，个人电脑开始流行，这种电脑对于成本的要求很极致，像极了当前中国的硬件设备价格战。

为了降低成本，个人电脑就把串口做精简。接头虽然是 9 针，但是电脑主板系统实际上只支持 5 针，甚至 3 针，信号线被认为没啥用，做了大范围的删减。

这时候，有两个美国小伙进入了历史舞台。

这两个都是 20 多岁的年轻人：

一个叫做丹尼斯•贺氏（Dennis Hayes），另一个叫做代尔•赫斯林顿（Dale Heatherington）。

两个人是好基友，也都是互联网发烧友，经常在一起吃饭，一块谈人生，也一块做点新鲜玩意。

1977 年，他俩先开发出来一款用于个人电脑的 Modem，不是很畅销，原因就是上面说的，硬件接口不给力。

俩人花了接近一年时间思考解决方案，终于在 1978 年，他俩在吃饭的时候，**想出了只用数据接口完成信令握手的方法，这就是 AT 指令的方法。**

使用这种方式，两人推出了一款新型的调制解调器，叫做 SmartModem。

因为很巧妙的只用数据线就完成了所有信号线的功能，所以可以完美适配所有的低端 PC 电脑。

这个叫做贺氏的小伙，很喜欢中国文化，给自己起了个中文名字，叫做贺丹毅，老婆也娶的是中国宁波的姑娘。

在 1978 年提出的这套 AT 指令，就已经通过著名的三个加号（+++）方式，从数据通信模式切换到命令控制模式。这种 AT 指令的通信方式，还能允许多台个人电脑共用一个 Modem 上网，大大促进了低成本硬件的通信功能。

这个 SmartModem 很成功，于是这俩人注册了个公司，叫做贺氏（Hayes）公司，在 Modem 行业大火特火，应该是赚了不少钱。

几乎所有 90 年代上网的老网民，都用过 Hayes 的 Modem。

Hayes 猫拨号时发出的滋滋啦啦又叮铃叮铃的声音，承载着人们对于互联网另外一头未知世界的渴望。

虽然 Hayes 的 Modem 在业内取得了巨大的成功，但此后因为商业策略和运作模式问题，经营每况愈下。在 1999 年，Hayes 公司破产关闭了，后来也失去这两个人的消息。

事了拂衣去，深藏功与名。也许美国贺小伙是归隐到宁波过小日子去了。毕竟，1999 年，他才 50 岁。

但是，他对通信行业的影响，实在是太大了。

由于 Hayes 品牌的 SmartModem 太成功，其他的 Modem 厂家纷纷模仿 AT 指令集，甚至试图兼容 Hayes 的指令集，来分一杯羹。

后来的 GSM 通信时代，通信模组也是把自己作为一个 Modem 来对待，只是由电话线的通信承载，升级为无线的通信承载。

为了不改变用户习惯， 各个无线通信 Modem 厂家，也就是模块厂家，诸如 wavecom、西门子、Motorola，都采用了兼容 Hayes 的 AT 指令集，并针对无线通信做了一些指令迭代。

这种通信的方式，有个前提，就是通信 Modem 的价格很贵， AT 指令很方便提供给低成本的服务需求设备来使用 Modem 的服务。

从 90 年代到 2015 年之前，GSM 模组的价格从 3000 多元，一路下降到百元左右。但主控 MCU 的价格，下降更快。到 2015 年的时候，十几块的 MCU 随处都能买到。

低价的 MCU，搭配高贵的通信模块，MCU 通过 AT 去请求模块的服务。

通信模块是标准品，软件几乎不做更改；

MCU 软件是定制品，软件围绕通信模块的特性来定制。

MCU 的成本在整个系统的成本比重太低，所以省掉 MCU 的意愿并不强烈。

所以，AT 指令曾经是很完美的使用方式，在无线模块领域也流行了 20 多年。

肆 OpenCPU 需求的兴起

时光来到 2015 到 2020 这几年，通信模块的价格直线下滑。

GPRS 模组从百元降价到了十几元，而 4G 模块的价格，从 2015 年的 300 多元，到 2020 年 4G Cat1 模块也几乎跌破了 40 元。

而 2015 到 2020 的这几年，MCU 相对平稳。十元的 MCU，可能至多下降 20%到七八块钱，不太可能直接降低到一块钱。

这就给设备商带来了强烈的成本预期，模块价格的降低以及 MCU 的平价，使得设备的价格大幅度降低。突破一个门槛之后，物联网设备的需求迎来爆发。

一个小几十元甚至十几元的通信模块，还要搭配一个七八元的 MCU。

而通信模块的运算和存储资源，是远远大于 MCU 的。**既然这样，为什么不去掉 MCU，所有的任务交给通信模块，进一步降低设备成本呢？**

事实上，这些年模块公司都在尝试省掉 MCU 这个命题。省掉 MCU，设备的主板的成本就显著降低了，在出货量可观的情况下，就有可能大幅度提升利润啊。

Wavecom 在 2010 年之前就推出了 OpenAT 的解决方案，SIMCom、移远都有自己的 OpenCPU 的软件开发包。

但是，很多年过去了，AT 指令的开发方式，仍然是这些模块公司的主要开发方式。使用 OpenCPU 量产的客户寥寥无几，为什么？

因为 C 语言框架的 OpenCPU 开发，在客户进行调试 debug 的时候，需要模块厂家提供大量的人力进行技术支持；而使用模块的公司多如牛毛，全国少说也有三五十万家，任何一家模块公司都无法承担如此高成本的 OpenCPU 技术支持。

所以，AT 指令虽然低效，但其技术服务成本也很低，没有其他的架构能撼动 AT 指令的地位！

这个时候，Luat 悄然出现了。

Luat 采用脚本开发应用，开放了源码，用户的学习成本极低，甚至是 Web 后端和前端工程师都能参与 Luat 的嵌入式开发。

用户可用无痛的方式，大大方方的拿掉 MCU 及其外围电路，仅仅使用低价的通信模块。通过 Luat 开发业务逻辑，完全接管设备的功能控制，做到成本的极简。

当某些设备商采用这种极致低成本方案的时候，高成本的方案，就会逐渐被淘汰出这个市场。

尝到甜头的 Luat 客户，心情肯定无比是舒畅的。

伍 成长的烦恼

Luat 客户陆续带来的各种反馈，就像初春的风，凉爽又不失温暖，猛烈又不失温柔。

这样的感受，充满了整个 2017 年。

原来，物联网模块的销售，也不是那么的困难。
原来，站在风口上，猪都能飞起来，这话是真的。

祸兮福所倚，福兮祸所伏。
随着销售量的一点点增加，合宙的诸多问题，迅速暴露了出来。

当一个产品品类越来越得到市场认同， 企图占有市场主流份额的时候，市场对这个产品的要求会指数级增加，要想真正占有主流份额，必须能够跨越高标准高要求这道坎。

那么， 市场对 Luat 提出了哪些要求呢？

随着客户数量的增加以及销售额的增长，合宙团队无知的弱点，越来越暴露无遗。

问题一：没有供应链管理体系，运营水平就像原始人

因为对销售计划完全没有概念，也就没有进行过任何的排产；最初的前面半年，销售出去的开发板，几乎都是当时担任运营老大的老陆一个器件一个器件，手工焊接出来的；在客户需求指数级上升的时候，合宙的交货能力是无法跟上的。合宙需要向客户证明，合宙有大规模高品质供货的能力。

问题二：模块的固件不稳定

有些客户用 Air200 可以稳定出货，没有遇到什么问题。

但是这个模块并没有经过全面的测试，也没有经过客户的大批量验证， 因此，总会有一些客户的使用场景遇到不稳定的问题。

固件版本的不稳定，一直伴随了 Air200 的整个生命周期，虽然品质在不断提升，但是直到停产，问题也没有完全解决。

直到合宙推出下一款 GPRS 模块——Air202， 固件稳定性的问题， 才得到了彻底的解决。

问题三：团队内部对于 Luat 开放的理解不一致

在 Luat 开放的最初 2 年，合宙团队并没有进行认真的内部沟通，也没有去认真探讨 Luat 开放系统的市场价值在哪里。

因此，合宙的软件研发团队在最初的 2 年，始终没有弄明白，Luat 有什么价值。

所幸的是，即使不知道价值所在，合宙 Luat 的软件研发团队， 一直在兢兢业业的开发、除 bug，使得软件的稳定性日渐提升。

但是，这种对于目标的不理解， 也必然会造成客户服务跟研发的脱节。

问题四：支撑文档不完善

合宙的工程师团队，一直以来，都是只做开发，不写文档。

这是因为，合宙的管理层和工程师，并不理解写好文档的意义在哪里。

文档的不完善，始终被 Luat 社区的网友不断诟病，但是长期没有得到很好的改善。

问题五：技术支持的无法持续

随着 Luat 用户的增加， QQ 群里面的人，从几十人，到几百人，到几千人。

几百人的时候，合宙的技术支持工程师，从早上 8 点多，到凌晨一两点钟，持续不断回答问题， 建立了良好的社区口碑。

但是人数过千之后，再有耐心的工程师也会崩溃。

然而，找不到更好的解决办法。

就是在这种相当低效的运营水平之下，合宙度过了 2017 年。

这一年，合宙的销售额仅有 2400 万元， 是 2016 年的三分之二，原因就是合宙砍掉了其他业务，仅仅保留了 GPRS 模块业务。

这 2400 万元销售额，几乎都是 GPRS 通用模块的销售额， 是合宙进入通用模块赛道的第一份成绩单。

在这一年中， 除了 2400 万元的惨淡销售额之外，还收获了以下成绩：

建立了超过 3000 人的社区：

这个社区，几乎都是 Luat 的用户，大多数是物联网企业的工程师。

他们被 Luat 的简洁开发所吸引，加入到 Luat 开发的大家庭；同时，他们也忍受了 Luat 文档的不完善，固件质量的不稳定，但是绝大多数坚持留了下来。

最初的这 3000 人， 是合宙 Luat 最亲密的朋友。

合宙团队逐渐意识到产品品质的重要性：

最初半年发出去的货，经常会出现品质故障，不是固件不稳定，就是模块的 IMEI 贴错了，或者是工厂没有校准就出货了。

产品的品质控制，成为最急需解决的问题。

不提升品质，就无法持续发展。

带着诸多的问题以及满满的收获，合宙团队怀着复杂的心情，进入 Luat 第二年。

2018 年，又会有哪些挑战？

3 在问题中突围，还是灭亡？

壹 4G 的突围，成为主流模块品牌

合宙团队带着诸多未解决的问题以及基数越来越庞大的用户群，像学步未稳的孩子，欣喜又跌撞着迎向 2018 年。

2018 年，合宙的 Air202 模块崭露头角，不仅颜值有了提升，超高的性价比更是让行业惊叹，此后势不可挡地成为 GPRS 通信模块的一个主流型号。

Air202 的客户群里面，有 70%以上的比例，是采用 Luat 开发产品应用， 只有不到 30%的客户采用传统的 AT 指令进行产品开发。

这已经是对通信模块行业的一个颠覆。

通信模块的功能，不仅仅是通信，还要承担更多的 MCU 运算和控制功能。

但是，有一个巨大的问题，始终存在——直到 2018 年初，合宙都还没有 4G 产品线。

由此，合宙始终不能算作一个主流的通信模块厂商。

在市场的千呼万唤之下，2018 年初合宙正式启动了 Luat 4G 模块的开发，并在 2018 年 9 月份，初步达到量产状态，准备对外发布。

新的问题来了：怎么定价？

当时的 4G 模块形势，高通的 9x07 主芯片占据了 95%的市场份额，价格在 120-150 元 RMB 之间。

合宙的模块是 ASR1802 方案，这个芯片是 ASR 所收购的 Marvell 的产品。几乎没有物联网的客户知道 ASR 是一家什么公司，也不知道 Marvell 是一家什么公司。

中美贸易战刚刚启动，而在物联网行业，**国产 4G 主芯片的份额几乎是 0。**这个时候强力推进国产的 ASR 方案，无疑具备非常高的社会价值，就算合宙不盈利，能把国产芯片份额打开一个缺口，也是值得的。

合宙此时已经拥有超过 5000 个客户的物联网企业客户群，具备一定的推广基础。

尽管如此，要让市场接受从来没出现过的合宙 4G 模块，难度堪比登天啊。

怎么打开局面呢？

对于具备海量用户基础的产品来说，价格无疑是能引起传播的有力武器。

然而价格是把双刃剑，利润减少的同时，无疑会触碰到行业纷繁复杂的利益线，从而使竞争更加激烈。甚至，可能让合宙的故事从此画上句号……

这剑，不是悬在头上，而是直向我的心摩挲而来。

关键时刻，合宙团队针对国内外形势反复探讨，一致认为唯有破釜沉舟，打破价格的桎梏。纵然不能成就合宙，也可为国产 4G 模块的商业化应用略尽绵薄之力。

“风萧萧兮易水寒，壮士一去兮不复还！”——拧成绳的合宙人，做好了慷慨就义的准备。4G 模块，突围则成，不成则亡。

从科技京城 26 楼的落地窗北望，越过蜿蜒的苏州河，那座巨大的摩天轮在夜色中有如魔幻的时钟。我知道，新的一天就要来了。

2018 年 09 月 11 日，合宙正式推出了支持中移动 4G 和 2G 频段的模块 Air720D，线上公开含税价仅为—— 69 元。

如同在封冻的海面投入炸弹，居高不下的 4G 模块价格崩离解析，行业沸腾了！！！

1802 模块的产品介绍发布在合宙的微信公众号，这个公号仅有不足 5000 个粉丝，文章也并没有提到价格是 69 元，只是在合宙的员工转发的时候，注明价格是 69 元。

结果文章阅读量超过了 2.6 万次，被转发分享几千次，口口相传更是不计其数。

合宙和 ASR 都一夜成名。

又经过半年多的技术服务，各种填坑，战事纷纷，合宙终于成为物联网市场的主流 4G 模块供应商。

而 ASR 也变得家喻户晓，其他的模块同行，包括移远在内，为了不被市场淘汰，也纷纷跟进推出 ASR 方案的 4G 模块。

由此，加剧了整个模块的市场竞争，而 4G 模块也进入相对快速的降价通道。

很幸运的，市场接受了合宙。

合宙的故事并没有完结，而是开启了全新的时代……

通过这一战，大家可以看到模块这个业务的技术壁垒不高，企业的规模优势，并不能带来模块产品的技术壁垒。

模块产品只是主芯片的一种封装形式，虽然模块厂家能协助主芯片厂家填上很多网络问题的坑，但是最终问题的解决，还是需要依赖主芯片厂家。

模块公司的出路在哪里呢？

我们的团队越来越认识到，构筑壁垒一定不能只是靠模块业务本身，还需要基于模块的销售，开发更加抽象的商务模式，给用户更高的满意度。

贰　过于分散的客户群，是健康的吗？

我在团队内部分享过一个文章，题目叫做《农村包围城市的战略》，大意是模块业务，只有中小客户群才能保证利润率，大 B 客户的议价能力更强，从而会极大的削弱毛利率水平。

从最近 2 年模块上市公司的财报，也基本上能看出来，毛利率是在持续走低的。

为什么会这样呢？

随着时间的发展，主芯片公司的技术能力，每一年都在得到增强。芯片集成度越来越高，软件做的越来越完善。

但是模块公司的技术能力，却无法增强，因为主芯片并不会开放核心设计给到模块公司。

因此，模块公司越来越趋向于是主芯片的分销商。

模块公司把主芯片做成模块的封装，从而更便利的把主芯片卖给客户。

因此，　模块业务的本质，就是主芯片分销。

无论模块公司承认与否，这都是个客观事实。

所以，模块的毛利率，会越来越趋近于分销的毛利率。

那为什么广和通的毛利率还相对不错呢？

因为广和通的大部分利润来自于笔记本电脑上网所用的 4G 网卡，虽说这个网卡长得像是模块，但是他实实在在的是一个网卡，是一个成品，不是模块。

广和通通过产品的技术和商业壁垒，保证了 PC 网卡这么个产品的较高毛利率。

但是，这跟模块业务实在是没什么必然的联系。

对于大 B 客户来说，模组公司更多的是提供了主芯片到模块这个过程中的加工测试环节的价值，所以只能拿到 PCBA 代工的毛利率，芯片到模组的加工的品质，决定了大 B 客户的留存率。

对于小 B 客户来说，模组公司提供的是一个抽象化的标准品，功能也类似于一个芯片，毛利率多高取决于需求量，因此模组公司的毛利率相对还比较良性。

因此，合宙的众多小客户，无疑是非常健康的。

后来，随着合宙自动化生产的成熟，相继有行业头部客户跟合宙合作，也印证了模组是主芯片分销商的这个本质的判断。

客户群增长的足够多，模块公司的订单量才能更有保证。

做模块的企业，核心就是用户量。

能服务好更多的客户，就是模块公司的核心价值。

叁 扩展流量卡业务

在模块业务不断增长的同时，总有客户询问：合宙有流量卡销售吗？

既然那么多客户有需求，为什么不把蛋糕做大呢？

然而这次团队明显有分歧，争论起来：

“好不容易砍掉了不盈利的产品线，现在又来搞不相干的产品？”

“这样又让合宙进入一个不熟悉的领域，成本太高了吧？”

“合宙的客户群是需要流量卡的，满足客户需求，是盈利的前提呀！”

“无论怎么样，总是要先满足客户需求吧”

……

几番争论之后终于达成共识，这个结果就是合宙成立了一家子公司——**上海展霖**。

合宙控股，聘请了具备流量卡运营和销售经验的伙伴作为股东，利用合宙的客户群资源，进行流量卡的运营和销售。

事实证明，这个决策不但正确，更对合宙的模组业务形成了有力的补充。

此后，合宙的流量卡业务在 2 年之内就达到了年销售额 1000 万以上。

尽管如此，合宙在整个模块市场份额依然小的可怜。

但是，合宙的经营目标，真的只是模块的市场份额吗？

我问自己，

却仿佛在穿越一条浓雾弥漫的大河，看不清彼岸。。。

肆　解决问题的核心是预防问题

前进、踩坑、成长，合宙从 2017 年一路走来历经许多问题，但是最重要的还是品质问题。

为什么呢？

因为品质问题就是成本问题。

品质有问题，不仅仅是产生售后的成本，更大的成本是影响到后续的订单。订单没有了，一切做的工作都是无用功。

模块的品质问题， 可分为几类：

第一个：设计不完善造成的品质不良

模块电路设计的缺陷，主芯片的设计缺陷，元器件选型的失误，都是这种情况。

合宙的 Cat.4 模块，曾有过 flash 选型不严谨，造成了有部分客户的模块变砖，合宙因此做了很多赔偿的惨痛经历。

这部分不良，合宙吃亏一次之后，经过元器件选型和电路设计的严格评审，已经得到了杜绝。

如今合宙 Cat.1 模块公开展示出来的各方面性能全面超越友商，已经很清晰的证明了这一点。

第二个：软件稳定性的不良

因为对于 2G、4G、NB 这一类广域网的模块，各个地方的网络都有不同的配置，信号强度也有复杂的变化，如果软件的兼容适应性不好的话，很容易造成设备的联网不良。

这个问题，是需要模块厂家和主芯片厂家联合不断迭代，经过多年的积累，形成的品质决定的。

合宙团队因为有 15 年的模块研发销售经验，所以这个问题，虽然在 Air200 出现过，但是后续的产品中，问题并不突出。

合宙在 2018 年开始逐渐加强测试能力后，固件的稳定性每年都在提升。

并且合宙的客户群非常广泛，大量的客户经过不同场景的测试，已经对合宙固件进行了千锤百炼的磨砺。

根据目前的市场反馈来看，合宙的模块固件，在行业内是属于头部的品质的。

第三个：加工管控环节的不良

加工管控环节造成的品质问题， 几乎都是管理问题。

从 BOM 管控，加工环节的流程管控，到 QC 检验的管控，再到发货的管控，都有比较成熟的方法来进行预防。

接下来，我将花费一些篇幅讲述，合宙是怎样预防品质问题的。

伍 逼上梁山，自研自动化运营系统

既然品质问题是最重要的问题， 彻底提升模块的品质，成为了最重要而且最紧急的问题。

怎么解决呢？合宙团队产生了如下争论：

“合宙没有品质部，应该马上成立品质部，要有专人去盯着品质问题。”

“合宙的加工厂实力太弱，应该更换实力更强的加工厂。”

“合宙的加工厂的工人管理水平不行，合宙要派专人去工厂跟线管理。”

“合宙应该研发生产信息化系统。”

“合宙应该研发自动测试系统，减少人力使用，从而提升品质。”

……

工程师文化深入骨髓的合宙团队，从来没有想过去寻求外界的帮助，更多的是弓下腰来，去思考彻底解决品质问题的路径。

经过一轮又一轮的讨论，研发自动化生产系统的方案占了上风。

2017 年末，合宙正式启动了自动化测试系统的开发，此后开启了不断演进的过程。伴随自动化测试系统的，还有信息管理系统。

因为对 ERP 本身了解的不够深入，纯粹是觉得 ERP 很牛逼，所以合宙自己开发的信息化系统，称之为**合宙 ERP**。

合宙 ERP 演变到了今天，包括了如下功能：

模块查询（开放给客户的功能，用 IMEI 和箱号，可以查询模块的生产过程和每个节点的时间）；

项目跟踪；

研发产品和 BOM 管理；

软件版本发布和跟踪；

生产管理， 包括工单管理，生产资料管理，以及很重要的测试实时进度和数据的查看；

客户管理和销售管理工具；

发货管理、跟踪下单、合同盖章、开发票、付款、发货单、收货确认等全环节的跟踪；

在线跟踪开发票系统等。

品质不是管理出来的，品质一定是预防出来的。
要想完美提升品质，必须从理论上预防缺陷的发生。

合宙团队选择自己研发自动化生产系统和信息系统，还不知道最终是对还是错，这几年一直持续迭代这个系统，花了巨大的时间成本。

虽然这是一条最难的道路，也是最慢的路，但是效果是可持续的，合宙管理效率在不断提升，客户体验也在不断提升。

在迭代合宙 ERP 的这几年时间里，不断产生如下的对话：

“这个系统真难用，怎么不买一个更专业的啊？”
“专业的人做专业的事，我们一个做模块的公司却在做 ERP，不务正业啊！”
“合宙 ERP 的产品定义太不完善了！”
“只有自己才知道要什么样的 ERP 系统，合宙 ERP 就是好用！”
“合宙其他的 WEB 系统也要采用 ERP 账号统一登陆！”
……
“报告领导：DTU 管理系统、SIM 卡管理系统、问答系统、工单系统、短信分发系统，都已经用 ERP 账号管理！”
历经各种嫌弃各种修改，信息化的工具基本就绪，运营环节的同步效率不是问题了，但是加工品质，还是一个短板。

不过，这个难不倒合宙的天才工程师们。

合宙工程师设计了一个规范，每次生产都用 24 个模块，4 行，6 列，做成一个拼板。

针对 24 拼板，设计镭雕，烧录，写号，校准，综测，功能检测的各个工序。全部工序完成后，再进行自动化包装。

同时，要求每个工序，都不允许人工干预，全部通过机器来进行，每个节点都上报合宙 ERP 系统（其实已经充当了 MES 功能）。

这样的过程，不仅仅不会产生漏网之鱼，整个过程还是可回溯的。

现如今，经过将近三年的持续攻关，合宙的自动化生产系统结合信息化系统， 模块的生产已经不是瓶颈。

业绩的瓶颈，是订单量不够。

怎么才能拿到足够的市场份额，让工厂的生产负荷合理化，又成了一个艰巨的挑战。

陆 问题重新定义

回顾梳理整个 2018 年的过程，合宙团队主要干三件事情：

1. 持续支持客户使用 Luat 方式开发应用，并丰富文档和教程；
2. 开发和发布 ASR1802 的 4G 模块。
3. 持续开发通信模块的自动化测试系统。

这一年，销售额上涨到了 6300 万元， 4G 模块虽打开了局面，但销售额极其有限，主要是靠 2G 模块客户的快速增长。

在第二章中所提到 2017 年的五个问题，有些得到了一些改善，有些问题却更加突出。

供应链的管理体系，依然没有完备，仍然靠团队成员的自觉性和加班来应付市场需求；有些同事，经常加班到凌晨一两点钟（现在这些艰苦奉献的同事已经被发展为合宙的股东）。

模块固件的不稳定，确实得到了本质性的改善，Air202 模块的固件，得到了市场的广泛认可。

团队内部对于 Luat 的认识有了改变。团队的人分为两派，一部分认为 Luat 的开发效率很高，适合客户快速迭代应用；另一部分人依然认为， 用 C 的 SDK 去支持客户开发， 跟 Luat 的区别也并不大。

支持文档不完善的问题，依然故我，没有本质性的改善。虽然建立了 wiki 网站和 ask 系统，但是内容依然杂乱无章，始终被用户群诟病。

技术支持团队在 2018 年依然没有搭建成功，靠着研发成员的自觉性，以及个人参差不齐的能动性，在支撑着用户的开发，导致团队内部的任务分配矛盾，以及用户得不到及时的技术支持的矛盾越来越突出。

在 2018 年底， 合宙团队逐渐意识到，所有运营的问题迟迟得不到改善，也许并不是运营本身的问题， 应该是在一个我们意识不到的地方出现了问题。

正是这年年底， 我在知乎发表了一篇文章：《一个傻逼 CEO 的自我反省》，在这篇文章里面， 根据客户导向的思维，提出了一些管理的疑惑，但是并没有清晰的认识到组织架构和组织效率的重要性。

问题被重新定义了：合宙的问题，不是执行的问题，而是团队成员认知不一致，从而导致源动力不一致的问题。

带着问题，没有解决方案，进入 2019 年。
2019 年，是产生变化最大的一年。

都有哪些变化呢?
且待后文慢慢分解。

4 沉默不会爆发，沉默一定会灭亡！

今天写这个文章的时候，不停的回顾 2019 年初的各种细节，仿佛在回忆上个世纪的场景。

其实也就过了才一年多。

在回顾的过程中，我脑子里面不断闪现一个特别具体的声音，这是一个只有河南人才能理解的声调和语气，用河南东南部方言朗诵鲁迅的名句: **沉默(mai)啊，沉默(mai)！不在沉默(mai)中爆发，就在沉默(mai)中灭亡！**
这里面，mai 读音第二声。

我一边默念，一边在脑子里面修改：
沉默(mai)不会爆发，沉默(mai)一定会灭亡！

这个方言的发音特别掷地有声，如果能有个河南老教师来演示一下就更妥了。

在 2019 年初的时候，合宙团队的运营诸事不太顺，仿佛每个人都非常努力，但是又总是差了一口气，产品品质不完美，客户服务惹抱怨。

不仅仅是生态社区的用户抱怨，线下大客户也抱怨。

甚至，部门之间，也抱怨。

场景一

A 客户深夜 12 点咨询技术问题，技术支持的同事不辞劳苦的在回复技术问题……

这样持续了一周时间。

有一天，销售突然跟金艺抱怨：A 客户的问题为什么没人理会了？客户在跟我抱怨了啊！

金艺说：不是一直在进行技术支持吗？怎么会没人理会呢？

销售说：客户一直在等待我们解决一个低概率出现的问题，但是我们的人一直在等待客户抓 trace。就这样等待了 3 天，一直没有任何讨论了。

客户认为，合宙工程师就应该解决问题；合宙工程师认为，没有 trace 我怎么分析问题啊？

场景二

软件部发布了一个新的固件版本 V021，解决了 5 个软件 bug，其中有一个是客户反馈的重要 bug。

发布之后，测试部马上就开始测试，经过一周时间不休不眠的奋战，提交了测试报告，结论是仍然存在三个 C 类 bug，但是算是测试通过，可以量产。

这事儿就算完成了。

过了一个月，软件部发现，量产出货的版本仍然是 V020 版本，V021 并没有导入量产。

因此，有一个客户下单拿到的货，仍然是 V020，没有解决问题，投诉到销售这里。

销售又投诉研发部，研发部也很委屈：**我们接到客户反馈，很快就发布版本解决问题了，为什么现在客户拿不到新版本的货呢？**

场景三

有一个同事，小 H（不说真名了，免得被友商挖走），几乎每天都要工作到深夜 2 点钟。

但是，小 H 日常和其他部门同事沟通，十次有五六次都得不到及时的回应。

小 H 跟同事私聊的时候，别人不回复；
小 H 在群里@别人的时候，有一部分回应，一部分也不回应。

两年过去，这个脾气很温柔的同事，终于变成了一个急躁的小 H。

只要有人问他事情， 十次有七八次，会被小 H 怼。

小 H 终于变成了合宙人见人怕的魔头。

但是，这不妨碍他是合宙的业绩标兵，因为只有小 H 能梳理清楚合宙的所有运营细节。

有一天，有个客户着急发货，但是合宙做不到，销售快急哭了，六神无主的样子特别可怜。

我问销售：你找小 H 协调一下不就可以了吗？

销售说：我不敢找啊，老秦你能不能帮我说句话，客户需求十万火急啊~

我就问小 H：这个客户的发货能通融一下赶紧发吗？

小 H 说：没问题啊，可是销售压根没跟我提交需求啊~

销售说：我怕被怼，不敢提交啊……

一年多之后的今天，小 H 还在合宙，但是已经重新恢复温柔本性了。

如果你想知道是怎么发生的，可以持续关注本文。

场景四

在过去的两年里，有 2-3 个同事，直接找过我要求加工资。

每次找我加工资，我都会很认真的去思考，到底应该加多少。

突然有一天，我在想，合宙的员工，为什么大多数都不找我谈加工资的事情？

是因为他们不想加工资吗？肯定不是的，合宙的工资水平，在行业内肯定不算很高的。

只是因为他们不想开口，他们更加愿意公司按照制度合理的给他们把待遇提升上去。

所以，才有了合宙每个员工的可变薪酬的改革。

如果你想知道合宙可变薪酬的细节，也请持续关注本文。

以上的几个场景，只是我能回忆出来的众多抱怨的少数几个。

团队成员的不满意，客户的不满意，跟合宙团队拼命付出的现状，形成了巨大的反差。

我没有办法要求合宙的成员更加多的付出，又要提升客户和团队内部的满意度，又不能无限制的扩大团队，因为这样会提升管理成本和运营成本。

怎么解决这个问题呢？

接下来的几章慢慢分解，也希望你能留言给出你的解决方案。

5 个人英雄主义和组织效率的矛盾

跟很多人一样，我也喜欢看武侠小说，尤其是金庸的小说。射雕三部曲印象最深，每一部都看了好几遍。

为什么武侠小说很让人着迷呢？

可能是因为武侠小说， 构建的是一个虚幻的世界。在这个世界里面，人际关系的世界，被简化了。简化到由一个英雄的个体，就可以解决大多数的问题和矛盾。

这跟打游戏，看童话故事、科幻小说一样，给复杂的人性描述了一个虚幻的出口，让人们能够得到一种虚幻的非常爽的感觉。

在这样的场景里面，人们倾向于用一种超现实的力量，比如修炼的极其高超的武功，来解决一个主要的命题。

但是，像郭靖这样的武功修为集大成者，也只能通过稳定团队信心，延缓蒙古大军的推进速度。

郭靖并没有丝毫的办法，阻碍宋朝被蒙古战败。

而现实中的企业发展的问题，靠个人的高超业务能力，一般都无法奏效。

为什么张无忌无所不能，但是最终建立功业的却是朱元璋？

金庸并没有给出答案，还是看一下真实的历史，更有指导意义。

翻一下各种历史，会发现：

每一个朝代更替，建立功业的关键因素是军队的运营。

军队的运营包括几个维度：人员招募（获客）、技能训练和纪律训练（留存）、信心和信念管理（转化）、临场指挥打胜仗（变现）。

运营一个成功的军队，持续多年能够胜多败少，尤其是能够赢下来关键战役，建立朝代的事业，就差不多进入成功节奏了。

而要保证运营成功一个军队，往往还需要一个完整的自洽的理论体系，这个理论体系，能够确保军队的管理有方法论可以作为管理的依据。**这就是组织的作用。**

所以一个成功持久的朝代，文官的政治地位往往大于武官。

回顾我党从成立，到建立新中国的过程，党组织的作用绝对是高屋建瓴的。正因为党组织的理论体系非常完整，执行的面面俱到，才有了一个战斗力爆棚的人民解放军，保证了新中国的顺利诞生。

在当今的商业社会，竞争是越来越充分的。

仅仅是靠单一的优势，无论是产品优势竞争、价格战的竞争，还是技术领先优势的竞争，都越来越难以保证胜出。

通过阅读各种各样的管理书籍 ，学习各种各样的管理理论，国外经典的营销管理的发展，**也会发现一个问题：**

随着商业社会的复杂度的提升，需要实施的商业理论也越来越复杂。**如果一个组织的效率不够高，步调不够一致的话，越来越难以在一个成熟的商业社会里面立足。**

在通信领域发展的这二十多年，华为面临的对手有很多：北电网络、 朗讯、诺基亚、西门子、爱立信、摩托罗拉，都是通信发展史上如雷贯耳的名字。

这些公司的技术积累和专利积累，都远远超过华为。为什么在商业发展的过程中，都被华为甩在身后呢？

华为的组织动员和执行能力， 远远超过上述的这些企业。

> 我在华为和摩托罗拉都工作过，华为团队的执行力就像是一条河流，把人推着走，推动的力量是源源不断的。作为个体永远不用担心没有事情做，各种各样的 review 和评审，不断地把你往既定的目标去牵引。

而在摩托罗拉，只要我会写逻辑清晰的邮件，不是很难的就可以把自己的责任抛给其他的同事。

美国人限制高科技产品供货给华为，其实并没有搞清楚华为竞争力的本质。

美国人更应该去学习华为的组织管理，技术上限制华为并不会根本上遏制华为的发展。

一家比较成功的企业，员工大多数非常忙碌。

这些员工经常会抱怨，我忙碌的好多是无用功啊，跟业务关系都不是很大，太虚了。我们公司是不是有大企业病啊。

就像阿里的员工，我听说绝大多数人的周报会写 2000 字。

而合宙的同事，不仅不用写 2000 字的周报，甚至大多数时间，连周报都不用写。

我一度沾沾自喜，这种不用务虚的管理方式，运作效率是真的高。但是，这种管理方式，带来了上一章所提的各种执行的问题。

看武侠小说，跟企业的运营，真的是完全两个极端的追求。

武侠小说，追求个人英雄主义的极致爽感；
企业运营，追求组织的整体效率和行动一致性。

所以要想解决问题，最好还是从组织管理上入手。看看现有条件下，怎么用相对小的代价，通过改变组织，找到解决问题的方向。

6 2019 年的市场形势

在 2013~2016 的这几年，4G 蜂窝模组还是稀缺品，主芯片主要是高通的 9x15 为主。

龙尚凭借提前研发，投入重兵在高通平台，一直占据 4G 模组的份额第一名，每个月出货在 40K pcs 左右。

SIMCom 紧随其后，出货量和龙尚差异很小。

一个 4G 模组，一般都在人民币 300 元以上。

移远因为延续了 2G 经验，3G 和 4G 也押宝联发科方案。但是联发科对于 IoT 产业链几乎没有投入，导致 3G 和 4G 产品线一直没有起色。

2016 年初，高通推出低成本 4G modem 芯片 9x07 系列。移远抓住市场切换的机会，推出 9x07 方案的模块 EC20，因为自己没有任何包袱，所以移远果断率先降价，同时投入大量人力做好客户的技术支持，很快就把 4G 市场份额做到了第一名。

反观龙尚和 SIMCom，因为固守高通 9x15 时代的优势，没有看清形势，不敢用 9x07 革自己的命，导致错失 4G 崛起的市场良机。

在移远崛起的这几年，合宙还在凭借 Luat 的开源应用方式，刚刚进入标准蜂窝模组的市场。

据说(数据不确定)，移远有超过 200 家代理商。

这些代理商，通过跟客户产生各种化学反应，做到了非常全面的市场覆盖。

2018 年下半年，4G 模组的价格，普遍在 130 到 150 元，代理商的毛利还是相当不错的。

有些超大客户，价格极其敏感的，移远也可能自己服务，保证一定的毛利。

通过这种非常灵活的渠道推进，移远以 EC20 作为明星产品，迅速做到了相对的市场垄断地位。

进入 2019 年，因为合宙在 2018 年底抛出的 69 元 4G 模块，在蜂窝模组行业，引起了悄然的行业变化。

合宙抛出的公开低价模式，对行业的影响，更多的是心理预期变了。

无论客户用不用合宙的模块，客户都知道有这么一个模组价格。

我们销售拜访客户说的最多的一句话就是:

你不用我的模块没关系，你把我作为备胎，心里记着就可以了。

合宙这种坦诚的态度，并不是每一个客户都认同，但是很多很多的客户，对合宙产生了信任，希望能跟合宙合作，进入物联网的大门。

这个市场反应不让人意外。

但是令人意外的是，信任合宙的客户，非常非常多，多到可以用蜂拥而至四个字形容。

这就带来了问题。

要知道，在 2019 年初，合宙的组织还非常小，行业理解也不够深入，组织成员大多数还只会单打独斗，团队协同意识非常弱，所以就迅速出现了前几章出现的问题。

但是令人不那么紧张的是，通过跟友商各位朋友的四处访谈，我们发现：我们面临的问题，**在整个模组行业，各个模组公司都遇到了类似的问题！**

所以，我认为，模组厂家之间，其实并没有太多的直接竞争。

把客户服务好，解决客户的问题，让客户信任，比单纯的价格优势更能够收获客户的心。

这也是为什么，我们不愿意过多关注模组竞争对手的套路和底牌，只把全部精力放在客户身上。**如果我们的份额上不去，一定是合宙自身服务能力出了问题，而不是竞争对手出了什么牌。**

7 跟一个女采购见光死的故事

一个女采购，计划导入我们的 Air202 模块，开发报警器产品，大概一年有个 10 万片的模块需求。

这个女采购声音还比较好听，微信聊天的时候，追货也非常积极。

为方便表述，下文称这个女采购为高欢欢（化名，如有雷同，切勿对号入座）。

自认为合宙公司发展规模还小，所以我决定亲自跟进这个客户，不转给负责销售的老陆，也不算过分。我当然不会告诉你，我没有把客户给老陆，还有个原因——**我，对这个高欢欢有好感。**

虽说老陆跟进的话，订单十有八九就稳当了，但是我仍然没有把客户给老陆。

就这样，客户从导入到调试、测试，再到小批量、大批量出货，还挺顺利。

因为客户距离上海比较远，我也一直没有机会当面拜访客户表达感谢，但是我心里一直想当面拜访。

终于在客户量产后的三个月之后，我找到一个机会，到这个城市出差，顺道去这个客户公司去拜访，顺道看望一下高欢欢。

到了这家公司之后，一切都按照期望的在发生。除了这个高欢欢，我也没有见到其他人，只跟高欢欢一个人在聊。

然而，结局不尽如人意。

聊了俩小时之后，我相当失落的离开了她的公司。

这个高欢欢的相貌也没有让我失望，还是很端正的。两个小时的聊天，没有发生尬聊，也没有发生冷场，聊得内容也还算充实。

可是，话题并没有按照我期望的轨道在走。

我内心去很希望产生一些思想火花的，比如诗词啊，比如感情看法啊之类的。

但是，见面以后，两个小时，除了有几句砍价之外，其他的话题都围绕我们的产品、他们的产品，展开了丰富但是不热烈的讨论。

两个小时一直聊天，仿佛很投机的样子，但是，这都不是我想要的聊天内容。

我想要的是希望有一点两人私人感情哪怕一点点碰撞，也代表我没白来啊。

临走之时，我对小高公司的产品产生了一定的了解，小高也对合宙的产品规划有了一定的了解。

但是，我们两人的私人关系，绝对没有升温，这是让我极度郁闷的。
回到上海之后，我把这个客户，郑重其事的交给了老陆。

我跟小高没有产生私人关系的升华，那么我跟这个客户，就没有办法产生化学反应。这是我做客户比较失败的地方。

我终于认识到，我不是做销售的这块料。
我们团队的分工，应该让更合适的人，做他更适合的事。

我们团队像我这样，
对人性的把握比较粗糙的，有很多；
像老陆一样，
对人性把握很细腻的，也有不少；

我们团队技术能力很出类拔萃的，
能数的出来一堆；
技术能力普通，
但是责任心出众的，也有一打；

我们团队能喝酒，怎么也喝不醉的，
有那么几个；
像我一样喝一点点白酒就痛苦不堪，
一瓶啤酒肚子就要爆炸的，占据了大多数；

我们团队逻辑思维清晰，
把我驳的没话说的，有几个年轻人；
嘴笨吵架不行，只会默默付出的，
也有一大半。
这样一个团队，每个人都有不同的特质。

目标一致，共同奉献。

怎么让这个人群特质差异化很大的团队， 更加提升效率，更好的解决问题呢？

我不禁陷入了沉思。

这个沉思，伴随了 2019 年的整个上半年。

8 怎么样才能让团队更专业？

在越战的早期，美国海军飞行员对阵越军空军的时候，不分伯仲。

1968 年的前 5 个月，美国海军击落 9 架越南的飞机，自己却损失 10 架战机！更夸张的是，美国飞行员发射了 50 枚空对空导弹，居然没有一个打中目标。

这不由得让美国海军开始思考：**怎么样才能够提高飞行员的能力，让他们能够在一定的时间内就具备一流飞行员的水准呢？**

不仅仅是思考，美国海军首脑们立马就行动起来了。

首先，挑选最优秀的飞行员来做教官，让他们模拟越南的战斗机战术，组成红军；

然后，选拔次优秀的飞行员作为学生，组成蓝军。红军和蓝军每天对战。

因为红军的水平更高，经验更丰富，所以常常把蓝军逼到失败的边缘。

蓝军的飞行员在紧急、危险或者各种复杂的状况下，做出的战斗决策有时候好，有时候差劲。战斗结束后，教官和学生马上开始沟通和复盘，总结刚刚过去的作战的详细细节应该怎么处理。

蓝军的教官们，根据作战的细节，分享他们的思考判断过程，帮助和促使学员寻找改进的办法。第二天，继续对战。经过这样持续，一日反复一日的训练，持续了半年时间。

效果怎么样呢？

1970 年，这批美国海军飞行员再次投入战场，他们实现了击落 12.5 架的敌机，自己只损失 1 架飞机的惊人战绩！

这一巨大的成绩背后，遵循了四个原则：

1. 明确的训练目标：蓝军的目标就是达到跟红军同样优秀；

2. 不断跳出舒适区，反复跟比自己更优秀的选手作战；

3. 及时的反馈和改进；

4. 周而复始的高强度训练，专注并且持之以恒。

对于合宙的蜂窝模组业务来说，分为以下几个业务单元：

1 硬件研发

模块封装要小，性能要稳定，温度范围要宽，抗干扰性能要强。

2 固件研发

固件版本要具备高强度的稳定性，还不能频繁更新。

3 文档开发

提供翔实有条理的文档和教程给到客户。

4 技术支持

客户有问必答，有 BUG 及时解决。

5 供应链

能否按时交付低成本的模组给客户。

6 品质控制

能否给出低成本并且高品质的模组给到客户。

7 销售推进

销售团队要能够让客户信任，成交。

上面包含模块业务的三个要素：品质、价格、服务，三者都必须做到足够的专业，才能赢得客户的心。

但是，说着容易，做起来很难。
理想很丰满，现实往往很骨感。

业务涉及的七个维度，任何一项要做到专业度在行业的前列，都需要一个团队的超强的能力和足够的付出。

然而，一个团队的人力的能力总是参差不齐的，怎么能做到超强的专业度，以及足够的能力胜任团队的要求呢？

方案就是：针对团队用合适的方法做训练。
合适的方法就是上面所说的——遵循四个原则的方法。

没有人生来就是天才，也没有人生来就具备熟练的技能。

只要学习方法得当，训练方法得当，任何团队都能像美国海军飞行员一样，快速提升作战技能。

这种训练方法，像一盏明灯，瞬间照亮了我的胸腔。这正是我梦寐以求，苦苦思索的团队能力提升法门。

接下来，怎么利用这四个原则，针对不同的产品线，不同的岗位，不同的人员特质进行提升？

首先，需要一个合适的教练。

而这个教练，很可能是我自己。

那么问题来了，是不是我自己得先提升到合适的认知水平，才能针对团队进行专业性的训练？

怎么提升我自己的认知水平和专业水平？

这，显然又一次陷入了僵局。

9 不能远程升级的 4G 模块都是耍流氓

X 电（化名）曾经是共享充电宝的明星企业。

2017 年 10 月，X 电以 2 亿的估值，获得了天使轮融资 3000 万，开始迅速铺货抢占市场。

在每一个餐桌上放一个单机版本的充电宝，用户扫码付款即可充电。

第一批货，做了 10 万台，投资金额 1800 万元。

> 每一台每小时充电租金 1 元，平均每天 3 次充电。
>
> 这样的话，10 万台充电宝，每天就可以产生 30 万元的流水，每月将会产生流水 900 万元。跟店家分成三七开，每月的收入将会在 630 万元，3 个月就可以收回前期铺货的成本。

X 电的经营团队，具有互联网公司的基因，也具备美团饿了么团队的地推基因。所以铺货的进度很顺利，只用了半年的筹备期，就把建团队、做产品、搭建运营系统这些硬骨头一一啃下了。

2018 年 4 月，10 万台 X 电共享充电宝顺利进入了全国各地的各个门店。

后台开始看到了租金流水。

X 电团队开始筹划用这 10 万台的运营数据，进行 A 轮融资，扩大市场规模。

设备大规模上线运营的第三天，后台运营团队突然发现，有 300 多台设备不在线了。

第四天，又有 500 多台设备不在线了。
第五天，700 多台设备失去联系。

运营团队召集通信模块供应商紧急到现场分析问题，抓日志，Debug，很快定位了问题所在。

原来，为了争取时间，X 电的工程师把后台通信协议的代码，采取 OpenCPU 的方式，写在了 4G 模块内部。

这段代码有个 Bug，在运行一段时间后，会产生内存溢出，导致充电宝上报的报文一直阻塞，必须人工去把充电宝重新开关机才能恢复工作。
因为这个 Bug 在充电宝运行几天之后才能发现，所以测试部门没有覆盖到。

X 电的工程师很快修复了这个 Bug，发布了新版本的 4G 模块固件，并要求运营团队马上升级这个固件。

X 电运营团队非常愤怒：全国各地 10 万台设备，你让我们怎么升级？你们研发部门难道没有在 4G 模块预留远程升级功能吗？

这回轮到研发部傻眼了：为了赶工期，我们只是开发了充电 MCU 的固件远程升级，并没有开发 4G 模块的远程升级功能。

在争论的过程中，每天又有更大批量的充电宝跟服务器失去联系，从每天的几百台，迅速扩大到了每天 3000 多台、5000 多台、8000 多台……

不到两周时间，超过 80%的充电宝，都失去了联络。

尽管有一部分店家采用人工开关机的方式，恢复了充电宝的工作。但是恢复之后，过几天充电宝又无法工作了，店家也就不厌其烦的放弃了……

就这样，10 万台充电宝全军覆没，运营数据也成了泡影。3000 万融资花光之后，第二轮融资失败，X 电退出了历史舞台。

直到现在，共享充电宝市场风起云涌，但是依然再也看不到 X 电的身影。

如果当时 4G 模块有预留远程升级功能， X 电目前的应收，很可能就是一年几十亿，估值也早已超过 50 亿。

因为没有预留远程升级功能，一个潜在的独角兽企业，就这么折戟沉沙，灰飞烟灭。

据调查，几乎所有共享充电宝企业，第一批的铺货，都遇到过通信模块的软件问题。

通过远程升级的功能，把通信模块的固件，以及跟模块搭配的 MCU 的固件，进行无痛的更新，才能使得初创的充电宝企业，顺利成长为一个个独角兽。

在发布合宙 4G 模块之前的几个月，我们也调查了友商的 4G 模块量产后都会遇到哪些问题。

得到的反馈，最多的是遇到软件 Bug。

并且，客户用 4G 模块做成联网设备批量发出去之后，总是会遇到各种稀奇古怪的问题，其中 99%都是软件问题。

所以，合宙的 4G 模块，可以允许有未知的软件 Bug 存在，但是固件的远程升级功能，是必须要经过不断的各种测试——**要确保在任何场景下，只要能通信，就必须要能够保证成功更新固件。**

合宙的 4G 模块，是在 2018 年 9 月发布的，真正小批量出货，是在 2018 年 12 月。

刚把模块发出去给到客户之后的几个月，我们最担心的就是模块是不是会变砖。

如果是因为硬件问题变砖，只能更换模块，赔款。

如果是软件问题，通过远程更新固件解决掉，几乎是没有成本的。

为了确保固件的升级成功，合宙的第一代 4G 模块， 甚至不敢用差分升级，而是**采用全固件升级的方式，确保升级的架构简单可控，从理论上杜绝升级失败的可能性。**

事实上，从 2018 年底开始出货 4G 模块之后，各种故障源源不断的发生。为了解决这些层出不穷的问题，在半年内固件升级了至少五六次。

客户的设备即使遇到了故障，因为能够第一时间进行远程升级，绝大多数都能够安然无恙的使用升级之后的固件运行。

在 2020 年初发布的 Cat.1 模块的策略上，合宙的模块继续提供了远程升级的服务。

为了节省空间，Cat.1 模块引入了差分包的升级方式，升级速度更快，占用模块的空间更小。

制作差分包的工具，也免费公开在合宙的 doc 社区。到今天为止，已有 1000 多个客户在使用这个功能。

购买 4G 通信模块，请第一时间确认，是否带有远程升级功能。

没有远程升级的 4G 模块，都是在耍流氓。

10 Cat.1 模块价格究竟可以多低？

最近一年来，我被问的最多的问题，就是 Cat.1 模块价格。

自从 Cat.1 开始被大众所知，因为低速率和低成本特性，可以覆盖 2G 退网之后的几乎所有应用场景——Cat.1 迅速成了所有物联网设备老板们的梦中情人。

老板们每天念想的都是：得花多少钱才能把 Cat.1 娶回家？

因为合宙有过一次颠覆 4G 模块价格的经历，所以老板们也很自然的对合宙寄予了厚望。

对于这些朋友的价格咨询，我一般的答复是：
第一、Cat.1 模块价格，在明年会持续走低；
第二、具体价格，我真的不知道能到多少。

然而，我逐渐发现，促成老板们成交的，往往并不是价格。

因为模块这个器件，实在是太重要了。

如果选择了一款不靠谱的模块，出现了联网问题，设备几乎就是报废，这个损失足以让企业进入很难翻身的灭顶之灾。

合宙的 Cat.1 模块，在客户使用多种模块进行同样条件对比测试的时候，往往表现更加优异。

在设备大批量铺出去的时候，实际的联网稳定性，超过了其他的模块品牌；

客户用合宙 Cat.1 模块，在跟某远的 Cat.1 模块进行对比测试的时候，合宙 Cat.1 模块每一项指标都能表现更好，这让人非常惊讶。

为什么会这样呢？
其实也不难理解。

合宙为了给客户爸爸们提供一款尽量完美的 Cat.1 模块，放弃了很多东西。

合宙最近三年的多次产品评审，NB 模块的立项始终无法通过，为什么？

——为了保证有限的资源全部投入 Cat.1；

安卓平台的智能模块的市场也很大，合宙也想做，但是多次的产品立项评审，还是无法被通过？为什么？

——为了不能分散 Cat.1 模块的各项资源投入；

5G 发展的如火如荼，各个同行纷纷发布 5G 模块，可是合宙的 5G 模块至今一点动静都没有，为什么？

——因为合宙在闷头深耕 Cat.1 模块，肯定无法分心去搞 5G 模块。

迄今为止，我还没有发现哪个模块企业，像合宙这样痴迷 Cat.1 模块，把全公司的所有资源，毫无保留的全部 All in 到 Cat.1 模块的研发、测试、生产工艺改进、Cat.1 的功能潜力挖掘，以及 Cat.1 模块的隐患预防。

2020 年 3 月，合宙最先发布并量产 24×24mm 小尺寸的 Cat.1 模块；

2020 年 8 月，合宙又发布了 19.6×21.8mm 颠覆自己的更小尺寸的 Cat.1 模块；

截止到今天，合宙 Cat.1 模块销量已经超过了 150 万片，远远领先于所有同行。

合宙 Cat.1 模块服务的客户群，涵盖了几乎所有行业：

POS 机、云喇叭、GPS Tracker、净水器、共享充电宝、充电桩、学生卡、安防摄像头、安防报警器、共享单车、公网对讲机、LED 控制……

这一年来，合宙的研发团队和技术支持团队，人数扩充了一倍以上，并且还在持续扩充的过程中；

合宙的 Cat.1 开发文档，迭代的也越来越丰富；

合宙 Cat.1 模块，提供了三种开发方式：
AT 指令、Luat 的 OpenCPU 开发、C-SDK 的 OpenCPU 开发，满足客户的不同需求及开发喜好。

当然，为了降低成本，合宙也采取了多种措施。

这种降成本，始终秉承一个原则：不影响模块的品质和性能。

模块适用的温度范围，大多是-40℃到+85℃。有客户做过实测，合宙的 Air724 模块可以坚持到 110℃还能工作；但是我们的手册，只是承诺-40℃到+85℃；

模块的射频灵敏度，每一个版本的迭代，都会做最严格的测试，保证发射功率、灵敏度、线性度，都远远超过认证要求。

所以，合宙 Cat.1 模块的认证进度，总是出奇的顺利。

合宙的每一片出厂的 Cat.1 模块，都能够让客户查询测试过程；都具备远程差分升级的能力；都有远程诊断的能力；都能有基站定位的能力；

而这些能力，都没有额外收费。

上面的所有的努力，给用户爸爸们带来的降成本收益，远远大于模块价格降低带来的收益。

尽管如此，合宙 Cat.1 模块的价格，在市场上也还是出了名的价格不高的。

因为合宙的企业文化，就是客户想要什么，合宙就要尽最大能力的满足。

合宙的销售额从 2018 年开始，每一年都比上年翻一倍，肯定是有赖客户爸爸们的喜爱和支持的。

所以，我给客户爸爸们的回答，可以很有自信的说：

合宙的模块价格，肯定是有竞争力的。

如果综合考虑品质风险、研发成本、售后成本（远程升级、远程诊断）、技术服务（线上+线下多种服务），使用合宙 Cat.1 模块的成本，将会具有无可比拟的颠覆性优势。

我还没有告诉你，合宙在四川的自动化贴片测试加工厂，已经稳定运行了超过一年半。

贴片的设备和生产 iPhone 11 的设备同样型号；自动化测试系统搭配全套信息系统，都是合宙自研，不需要采购价格昂贵的通用自动化设备。

这也是让友商羡慕并不断追赶的硬性条件之一。

Cat.1 模块的价格，能做到多低？

这个我无法做承诺，我只能承诺客户爸爸们，合宙 Cat.1 模块，持续提供更靠谱的服务，让客户爸爸们的设备，能够永远在线。

这是合宙能提供的最低价格。

11 男人的郁闷怎么解决？

王强（化名）是我们的一个老客户的 CEO，经营物联网共享设备，跟我们团队合作超过 2 年了。

我跟他比较谈得来，有什么心事，王强还是很愿意告诉我的。

王强经常在下班后，开车回到家，在车里面发一会儿呆。

少则 20 分钟，多则个把小时。
发呆完毕之后，再上楼回家。

回家之后，会装作心情很不错的样子，跟老婆说一下自己一天的情况。

第二天早起，再做出慈父的样子，送孩子上幼儿园，然后再去上班。

然后，晚上下班，继续车里面发呆一会儿，然后再回家。

我问他：这种状况，持续多久了？

他说，说不清多久了，可能至少有 1 年了。

我问他：在车里面想什么呢？

他说，也不知道想些什么，只是觉得，这是一天里面唯一能够真正独处的机会，也是一天里面唯一没有太大压力的时间。

我问他：你回到家能有什么压力呢？

他说：以前也很纳闷，回到家能有什么压力呢？老婆对我很好，孩子也很喜欢爸爸。但是后来我发现，回到家之后，总觉得自己对不起家里面，回到家总有一种愧疚感。

我问：具体是什么样的愧疚感呢？

王强说：

> 陪老婆的时间太少；跟孩子一起玩的时间太少；跟老婆能说的话也越来越少，就仿佛有了外遇似的~
>
> **其实外遇可能就是工作，除了工作，对家里的事情知道的是越来越少了。**
>
> 我老婆虽然嘴上不说，但是从她的表情和动作，以及语气，我能感受到她的不满在日益增加……

我问：那你多陪陪家里面不就好了？

王强说：那么，老秦你能做到吗？

我说：我可能跟你一样， 也有一样的愧疚。只不过我没有驾照，所以没办法躲在车里面。

双方沉默了半分钟，我说：我的办法是，我找机会出差，躲在酒店的房间里面发呆。这个时候的压力也没那么大。

王强说：市场竞争的压力越来越大，跟我 2 年前刚刚创业的情况相比，要使出 3 倍的时间和精力，才能拿到正常的回报。

我说：是啊，现在团队的运营成本越来越高，毛利率却每年都降；虽然出货量越来越大，但是利润并没有显著的增加，反而是运营费用一年比一年的高。

王强：你说说看，这么大的压力，我们民营企业的老板，怎么能心情畅快呢？

我问：你跟你公司团队的同事，沟通也会有类似的问题吗？

王强：好像是有的，也是一样的。我们公司人少，才 15 个人，但是每个人都是我亲自管理和沟通的。

我说：你这样事无巨细，岂不是很累啊？

王强：

> 如果只是辛苦的话，我也就不纠结了。**问题是，辛苦也得不到理解啊！**
>
> 我们团队的同事总是认为，老板肯定赚大钱了，我们打工的，做好本职工作就行了，不需要参与管理，也不需要注意沟通方式，反正干好活就行。

我说：这是不是让你觉得很委屈，感觉这么辛苦的投入很不值得？

王强：是的。事实上，这几年公司压根没赚到钱，我自己的收入每个月才不到 1 万块钱。这也是我每次回家都感到愧疚的原因之一。

我说：我特别能理解你这种感受。**你是不是觉得，没有时间陪家人，还没给家人很好的物质生活，让你对未来充满恐惧？**

王强：是的。我现在都在考虑是不是要把公司关掉，重新出去找工作打工去了。

我说：你当初为什么要创业，而不是继续打一份工？

王强：2 年前，我打工的月收入就已经超过 3 万了。**但是总觉得生活没有激情，为了改变这个生活状态，就选择了创业。**没想到现在越来越郁闷。

我说：你现在最想对你的太太说什么？

王强：我最想对我老婆说：小莉，我每次面对你，都很自责。再有 1 年，我们公司就可以走上轨道，会有利润了。感谢你这两年的不离不弃和对我的理解。**可是我总觉得这么说很空洞。**

我说：你可以说说看，没准就心里好多点呢？

王强：算了，不说这个了。对了，你们最近新出的支持 LVGL 的 Cat.1 模块，对我们帮助很大。我们 CTO 说，最起码节省了他 4 个月的开发时间。

我说：你做了一款支持图形界面的共享足疗机吗？

王强：是的，这款很受老客户欢迎。并且用 Cat.1 直接支持图形界面之后，整个设备成本比传统的方案节省了 50 多块钱。

我说：稳定性测试下来咋样？

王强：测试了两周，看上去不错，没出现什么问题。

我说：希望你大卖，一个月能出货超过 3000 台，你就发达了。

王强：对啊，一台赚 200 块，还能收到客户的服务费。客户赚到钱很爽，我也能有利润了。

我说：

我觉得，你还是要把这个即将到来的喜悦，跟你老婆，还有你的团队，都充分分享一下，可能更好一些。

说不定，他们能给你更好的建议，让你少走弯路呢。

王强：听上去好像是这样。我试试吧。不过我还是很担心这款产品能否成功，不太敢跟他们认真讨论这事儿。

我说：你担心万一失败了，他们会看扁你对吧？

王强：还是会有这个担心的。我特别在意别人对我的看法。

我说：你还是先试试看吧，把你的担心原原本本告诉你的老婆，听听她怎么说。

王强：我试试看吧。

关于王强的下一步的思想动态，后续如果有发展，我再更新给大家。

希望他能有好运。

如果你对他有建议，请在评论区留言。
如果你对我有建议，也请在评论区留言。

12 王强的逆袭翻身

上一章写的关于王强的郁闷事迹， 在后半段我没有抵制住诱惑，加了广告。这段广告的对话是凭空杜撰的，为这事儿我后悔了好几天。

所以我保证，本篇肯定没有广告。

这一篇，我交代一下王强的后续进展。

王强的公司有 15 个人，除了行政助理之外，包括王强在内，几乎都是技术工程师。

这个团队 90%的时间，都在讨论客户的产品需求，以及用什么方式实现需求才能更加节省成本。

同时，为了防止客户乱压价，王强团队还去琢磨怎么提升技术含量，底层的开发，都自己做。

这样就可以突出技术的特点，能够提升跟客户的粘性。

每次团队突破一个技术难题，王强都会兴奋一两天，这个时候，他会对自己说，这么辛苦很值得；

当修改了电路设计降低了成本，并且通过软件实现降成本之前的同样功能时，王强会高兴一星期，整个世界都有被他的团队征服的感觉。

这样的过程持续了三年半。

世界并没有被王强的团队征服；
三年多的辛苦，也没有带来公平的回报。

虽然王强的产品，经过长时间的努力，能符合客户的需求，成本也降到了客户满意的程度。

但是，客户自己却歇业了。

这样的客户，王强团队在三年半的时间里，遇到了 5 个；

客户没有歇业的，订单也是稀稀拉拉，利润还不够付办公室租金。

终于，勉强熬到 2019 年，王强的公司没办法正常发工资了。

王强所有的积蓄，都花完了，还欠了几百万的债务。

在一个雨夜，王强团队 13 个人（之前辞职了 2 个）聚到一起吃散伙饭，大家都喝醉了……

平日里不苟言笑，很爱面子的王强，哭了，哭的稀里哗啦！他对着团队的所有人，不甘又无奈地说：

我对不起你们，我们现在除了散伙，没有其他的选择了。

我们公司甚至没钱发放遣散费，真的是弹尽粮绝了。

我欠你们的工资， 我想办法后面还上，但是，你们现在别跟着我了，都去找工作吧，你们的能力都很强，都可以找到更好的工作。

这 12 个同事，也都哭了。

哭完之后，各自去找工作，散伙了。

团队散伙之后，王强消沉了两周，每天除了抽烟就是抽烟，两周下来瘦了 6 斤。

但是，为了还钱， 还是得出来挣钱。

做什么好呢？
思来想去，王强也找不到新的方向。

索性就先从自己最看不上，但又最容易上手的贸易做起吧。

因为用过蜂窝模块，就先代理蜂窝模块吧。

蜂窝模块的适用范围广，不需要什么研发投入和供应链投入，自己一个人只要找到客户，说清楚产品的特色，客户大多数也就用了。

而在 1 个月之前，模块代理这个业务，王强是无论如何也看不上的，他认为这是一个没有任何技术含量的生意。

> 王强反思自己之前三年的经营模式，15 人的团队，12 人都是研发工程师，而且都是老工程师。
>
> 每个人一年的支出都在 30 万以上，产品做的是还可以，但是一年投入 500 万都不够，过早的就把子弹全都消耗殆尽。

最终，他开始单枪匹马做模块代理，就这样混了近 2 年。

前几天，王强过来找我喝茶聊天。

我问：“最近生意咋样？”

王强一边脱夹克衫，一边不紧不慢回道：
“马马虎虎吧，上个月销售额差不多能突破 1000 万了。”

我问：“怎么可能啊？就你一个人，怎么做 1000 万？？”

王强一边往口袋里面摸香烟，一边说：“我也觉得不可思议，可是现在就是这样。”

我问：“都做啥业务啊？”

王强打火机打着了火，还没点，就说：“还是做模块代理，刚开始上道。”

点上烟，他接着说：“还是做贸易好些，投入小，包袱轻。”

我问：“代理的利润不是越来越低吗？你怎么赚钱？”

王强嘴角微微翘起来，然后又忍不住笑的牙齿露出来一点，说：**“确实是挺低。但是我有办法，那就是控制成本。”**

喷了个烟圈儿，王强接着说：“我现在办公室才 120 平米，租在张江的一个园区的孵化器里面，拿一点补贴，房租很便宜了。”

我问：“你还是得需要几个员工的吧？”

王强脸上暗淡了一下，沉默了一会儿，掐了烟蒂，眼睛看着茶杯，缓缓的说：“从前就是因为带团队太失败，所以现在对招人有阴影了。”

他又端起茶，抿了一口，眼睛望着窗外。

窗外是园区的花园，并不是开花的季节， 只有绿树、灌木，和青青的绿草。草坪中间，有露天的咖啡厅，三三两两的人们坐在咖啡椅上，很闲散的交谈着。

王强接着说：“我现在招了两个小姑娘，给我做商务助理，工资一个月不超过 6000 块，但是可以根据销售业绩和毛利来提成。”

我问：“这样的话，你的成本岂不是很低？”

王强有点不太好意思的看着我，低声说：**“老秦，我现在一个月的运营费用，不算我的工资在内，开销只有 4 万块不到。”**

看着我惊讶的眼神，王强稍微抬高了一点声音说：

“老秦，代理商毛利是挺低的，3 个点，但是我 1000 万销售，也有 30 万的毛利。扣掉费用，一个月赚 20 多万还是轻松的。”

我有点不敢相信的问他：“你是个技术男啊大哥，怎么现在变得这么像个资本家？”

王强听到这句话，上身马上后仰到沙发靠背上，嘴巴笑的合不拢，眼睛突然放出来亮光一样：“我就知道你会这么说我！我现在是真的想明白了，小公司的小老板，千万不要拿乔布斯们、任正非们做榜样。”

他重新点了一支烟，深吸了一口，吐出来，慢慢的，仿佛是很用力的说：“小老板要想活下来，就要认真的去活！**认真活的意思就是，算好自己的投入和产出，不要让自己的现金流断掉。**”

我问：“可是你做销售并不擅长啊，怎么能卖出这么多销售额？”

王强沉吟了几秒钟，抬起头说：

其实销售吧，也不是那么玄乎。我每次到客户那里，都会从我的失败经验里面，去想，我怎么能帮到我的客户，能不那么大的投入，就能把产品做出来。

端起杯子，又抿了一口，他说：“我因为懂点技术，又做过失败的老板，所以经常能懂得我的客户老板的心情，他们喜欢跟我说真话，我也喜欢告诉他们真话。这样一来二去，客户都还挺信任我的。”

我问：“可是客户用模块总是需要技术服务的，你没有技术工程师怎么服务他们？”

王强说：“我都把技术服务推给了模块原厂，自己不做服务。做技术服务太费钱，我不干。”

我说：“你从技术老板，到奸商的跨越，还真是挺快的，佩服佩服！”

王强往后靠了一下，有点如释重负的吐了口气，说：“我现在其实还挺享受奸商这个说法的。这代表我真的进步了。”

又喝了一杯茶，王强说：“我得去给客户送货去了，走了走了，下次再来聊！”

13 生意还是事业？

上一章写了王强逆袭成功，每个月赚 30 万的故事。

文章发出去没几天，王强就又来找我了。

进门的时候，我看他的脸色并不是很好，脸上如同蒙上了一层灰蒙蒙的水雾，沮丧的样子像是刚刚丢了 10 万块。

刚坐下，没等我把水灌满茶壶，王强就说：

“老秦，能不能借点钱？”

我说：“来这里第一句话就借钱，显得不是很有文化。为啥借钱？”

没等我话音落完，王强就急着说：“借的也不多，3 万就够了。”

我说：“我没问你借多少，我是问，你为啥借钱啊？你生意不是挺赚钱的吗？”

王强拿出烟，上下摸索的找打火机，也不说话。

我一边烧水，拿茶叶，一边看着他摸索打火机。

终于找到了，点上烟，深吸了一口，又吐出来，呛得我直咳嗽。

终于，他说：“赚钱是赚钱了，不过到发工资交房租的时候，还是没钱。”

又吸了一口烟，他说：“主要是客户都需要放账，我一般毛利 4-5 个点，放账的时候，扣掉资金成本一个月千分之六来计算，我应该会有 3%的毛利。”

我说：“你的意思是，钱都放账给客户了，你的资金总是不够用对吧？”

王强掐掉第一根烟，脱掉外套，说：“对啊，**我这模块卖的越多，我是越缺钱啊。**”

我说：“那你打算怎么办？这么着也不是办法啊？”

王强接着点燃第二支烟，说道：“我打算接下来收缩销售额，一个月做 500 万就得了。”

我问：“一个月 500 万销售，3 个点的利润，你只赚 15 万毛利，纯利不是变少了吗？”

王强沉默了几秒钟，把刚抽了一口的烟，有点不舍的在烟灰缸掐灭，说道：“抽烟太多了真不行，不抽了。”

然后喝了一口茶，接着说：“我打算把 1000 万的生意，缩减到 500 万。砍掉什么生意呢？**砍掉一半放账的生意，留下款到发货的生意。**”

我说：“听起来这个主意可以啊，虽然利润砍半了，但是现金流能好多了。”

王强盯着我的眼睛，看了半晌，说：“你觉得我这个主意怎么样？我有点拿不定主意。”

我说：“我支持你这么做，这样可能你会轻松很多。”

“不过我想到另外一个问题。”我接着说：“你接下来的发展思路是什么？”

王强看了我一眼，眼光转到窗外的绿地，绿地旁边的小路上，有个小孩走过，一边走一边踢着石子儿。“这也是我一直在纠结的地方~不放账吧，生意无法扩大。放账吧，资金周转不过来，还是做不大。很苦恼……”

我说：“这是不是意味着，**代理这个生意，资金规模才是最核心的竞争力，人的作用是第二位的？**”

王强握紧茶杯，眼睛目不转睛的对着茶杯，说：“其实你说的很对，确实是这样的。**我现在意识到代理这个生意，不算是个很好的生意。**”

我说：“那什么才是好生意呢？”

王强把茶杯放在鼻子下面，使劲的吸气，仿佛这个茶杯是他的香烟一样。吸了好几鼻子，才说：“不用到处求人，也能躺在家里面数钱的生意，是好生意。”

我说：“这只是结果，但是怎么才能达到这个结果呢？”

王强说：“还是要找个能值得长期投入的生意才行，代理这生意，可能过几年就没了。”

我说：“为什么这么说？”

王强放下茶杯，眼睛看着我倒茶的手，缓缓的说：“因为代理是个中间环节，长远来说，中间环节的价值并不大。”

我说：“很多大佬都说过，好的生意，需要有三个特征：行业空间要大，要避开竞争，还需要有好的商业模式。”

王强说：“我看这个代理生意，行业空间是挺大的，但是竞争激烈。但是商业模式是啥意思？”

我说：“商业模式指的是，你投入的资源，主要是投入你的时间和你的钱，**通过某种杠杆手段，能得到指数级收益。**”

王强的下巴微微往下收缩，眼睛微微闭了一会儿，仿佛有点累的样子，然后稍微有点用力的说：

> “代理的生意，投入的钱跟收益，不可能是指数级的收益，是线性收益。甚至规模越大，收益越小。”

我说：“所以对于代理来说，资金成本是关键，你如果有很便宜的资金，做代理也挺好的，资金杠杆可以让你很赚钱。”

王强不等我话音落完，马上说：“说得对啊，可是我没找到便宜的钱。”

我说：“因为你找不到便宜的资金，所以这个生意，你其实做不大。 不过即使做不大，你这个模块代理业务，也确实还是个不错的生意，因为即使没有很多资金，你做 500 万销售额，赚 15 万，并不难的。”

王强沉默了约莫 10 秒钟，然后缓缓的，加重了一点声调说：

> “你说的没错，模块代理生意不难做的原因，**是因为卖模块很容易跟客户的老板能直接交流，因为模块的品质好坏对客户来说太关键了。**”

我说：“模块如果不好用，可能会导致客户的设备都挂掉，客户的损失巨大。”

王强很快的说：“对啊，正因为这样，我跟客户的老板都很容易沟通。”

我说：“跟老板直接沟通，会不会增加做成生意的难度？”

王强的嘴角开始上翘，然后忍不住张开嘴巴笑起来，看着我的眼神也开始带了一点悲天悯人的感觉：

> “不会的。**恰恰相反，老板可能是地球上最可怜的动物。**老板除了给他的公司赚钱之外，可能都不会有其他任何的诉求了。”

我略微有点尴尬，想绕开这个话题：“但是除了老板之外的员工，想的跟老板可能会有点不一样对吧？”

王强收敛了笑，很快的说：“对啊，如果不直接跟老板沟通，而是直接跟其他人沟通的话，要照顾到非常多的细节，**任何一个细节失误，都可能造成失败。**”

我说：“为什么跟老板沟通不用注意太多细节呢？”

王强看着我的眼睛，直白说道：

> “因为老板只在意是不是能帮他赚钱，只要能帮到老板，其他的细节就不那么重要了。
>
> 所以我越来越发现，直接跟老板谈生意，要比跟其他人要容易的多。”

我说：“你这么说，还真是非常有道理！我以后再也不直接跟供应商谈生意了。”

王强又笑起来，嘴巴咧的更大了，尽管刷牙很勤快，但是因为抽烟造成的不那么漂亮的牙齿也露了出来：“老秦你真聪明！以后你就让其他人谈供应商，你在幕后监控就可以了。”

我说：“我们还是回到正题吧，说回到你的生意，看样子你这个生意， 只要每月销售额不超过 500 万，就还算是好生意。”

王强又开始严肃了，马上说：“看来就是如此，我这个模块代理，在没有便宜资金的情况下， 不能做规模化扩张。也许，我做的只能是个生意，不能像你一样，是在做事业。”

我说：“生意跟事业的区别在于，生意能赚钱，但是无法规模化，而事业是可以规模化的。”

王强站起来，伸了个懒腰，又做了三个扩胸的动作，然后说：“事业不仅可以规模化，还必须要能持久，只能干两三年的生意，只是生意，不是事业。”

我说：“嗯，是的。除了要能规模化，能持久经营之外，有好的商业模式能够获得高回报，也是好的事业不可缺的因素。”

王强穿上外套，喝了最后一杯茶，摆摆手说：“你说的都对，我还是要为了我的生意要账去了。走了走了，下次再来聊！”

14 物联网生意经

写这个物联网大学系列的过程中，激发了我多次思考，但是一直没有特别明确的主线。

于是，最近两三个月，我主动出击， 拜访同行的各位大咖。虚心请教，甚至刨根问底，解决了心中的诸多疑问。

跟各位大咖的请教过程中，我深深的感受到：

所有的业务不顺利，以及团队发展的不顺利，都来自于团队带头人是不是能够心态足够开放，放弃自己的执念。这个执念，主要是包括自己的面子和思维的懒惰。

为了更好的分享， 我准备结束这个物联网大学的系列，今后改为单个文章的形式来发布。

最后，我仿照刘禹锡的陋室铭写了一篇《物联网生意经》送给你。感谢你这段时间不厌其烦的阅读我的陋作，希望你能够摆脱搬砖的烦恼，**真正把控自己的内心，游刃有余的把生意做好**

量不在大，有利则行。价不在高，有货则灵。团队虽小，主线分明。会议上争吵，团建入酒精。畅谈有大咖，茶话有文青。可以谈赛道，定使命。无搬砖之烦恼，无抢货之劳形。阿里三板斧，华为逆势行。老子云:“道在心中”。

二十一 合宙创业故事：合宙的第一个 10 年

本篇创业故事可通过点击右方链接在线阅读：合宙的第一个 10 年（连载中，每周更新）

也可直接阅读本章节以下内容：

1 客户破解了我的孤独症

合宙是 2014 年开张的， 今年是 2024 年， 正好 10 年了。

合宙的发展很慢， 发展 10 年了，都还没上市。

这 10 年来收获的最宝贵的资产， 是长期一起奋斗的团队成员， 以及跟合宙一起持续提升品质的合作伙伴。

合宙这 10 年， 有很多故事， 我想把这些故事， 认真的讲给大家听。

合宙初创的几年，连续决策失误，经营陷入泥潭，但是很快通过差异化的产品经营，逐渐走向正轨，这其中的故事，参见《我的物联网大学》；

我今天的故事， 从 2019 年讲起。

在 2019 年，合宙面临一个艰难的选择，让我，以及我的团队，陷入一种情绪， 这是夹杂了期待，兴奋，恐惧，犹豫，患得患失的情绪。

这种复杂的情绪， 持续了接近一年时间。

究竟是什么事情呢?

主要是产品路线的选择问题。

2019 年，物联网模组的主流认识还是 NB-IoT 技术。

NB-IoT 是一种窄带的物联网技术，对这个技术， 华为投入了极大的资源， 推动了 NB 技术纳入了 3GPP 全球通信标准。

华为研发量产了基站，大批量在中国部署。

并且，收购了一家英国芯片公司，专注做 NB-IoT 的芯片开发。

2017 年 6 月，工信部发布了《关于全面推进移动物联网（NB-IoT）建设发展的通知》，于是，从这个时候，一直到 2019 年， 几乎所有的媒体，都在很热心的宣传 NB-IoT 技术。

国家的扶持基金，也开始对采用 NB-IoT 的设备，进行从资金的补贴，同时还对 NB-IoT 给到政策的大力支持。

总而言之，NB-IoT 技术制式，就是时代的宠儿，集万千宠爱于一身的物联网弄潮儿。

合宙在这个时候， 主力产品线是 2G 模组和 4G Cat4 模组， WiFi 模组也同时在经营。

2G 面临几年后的退网， Cat4 的市场也已经非常成熟， 合宙再进一步的机会很小，WiFi 模组也面临极度卷的状态，经营的空间也很小。

这个时候，合宙顺理成章投入 NB 技术的开发，不是很好吗？

但是， 合宙在这个时候， 更加青睐的是一个叫做 Cat.1 的技术制式，也就是 4G Cat1。

这个时候， Cat1 还没有任何一家的芯片量产， 落后了 NB 产业差不多 2 年时间。

第一款 Cat1 芯片，仅仅是有望在 2019 年第四季度有工程样品，2020 年第一季度有望量产。

这个时候， 是尽快的赶上热潮， 搭上 NB-IoT 的快车，实现业绩的快速增长；

还是，忍住 NB 的诱惑，坚持自己的观点，继续固执己见，仅仅上马 Cat1 项目？

有人问，合宙为什么不是两个都要呢？

毕竟小孩子才做选择，成年人是全都要呀。

要知道，合宙只是一家小公司，财力，人力，都不算强大。

如果 2G， NB，Cat4， Cat1 全部上马的话， 要大幅度扩充人力，对管理水平是很大的挑战，运营的成本也会大幅度增加，经营风险一定会急剧加大。

站在理性的角度， 合宙应该做减法， 聚焦资源，把用户最需要的产品做到最精，给到社会更大价值的同时，合宙也能够健康发展。

这时候， 从我的内心来讲， 我是看好 Cat1 的前景，我不看好 NB。

但是跟我采取相同看法的人，实在是太少了，我甚至找不到可以商量的人。

为了对抗这种孤独感， 我陆续写了几篇文章，给自己壮胆。

我从各个角度分析 NB 的商业不可持续， 并且论证 Cat1 技术的商业合理性。

这几篇文章发出去之后， 也没啥反响，没什么人理会。

更悲催的是， 在一个 IoT 高峰论坛上，主办方邀请我做了一小时的发言，讲一下物联网的技术发展趋势。

当时的现场听众有几百个，线上直播的观众过万。

我感到机会难得， 做了很详细的准备。

在演讲台上， 我把 NB 的技术弊端详细讲了一遍，把 NB 的商业不可持续性，也做了详细的说明；

同时，我对 Cat1 做了大力推荐，明确声明 Cat1 才是未来物联网发展的正道。

现场的观众反响热烈， 很多人跟我有了强烈的共鸣。

然而， 高峰论坛之后的媒体发布稿， 把所有嘉宾的发言都做了如实报导，唯独只有把我的发言全部全部删除了，甚至连我的名字都没出现。

我终于明白过来， 我支持 Cat1， 抑制 NB 的观点， 是不被社会的主流观点支持的。

那么这就是一个很严重的问题： 就凭我们合宙这个百人小公司， 去跟全社会的态度抗衡， 不是妥妥的找死吗？

怎么办？

我陷入了深深的纠结之中。

我们公司当前的人员储备，绝对是不允许 NB， Cat1, 2G， WiFi， Cat4 一起做的，这么多产品线一起做， 肯定做不好。

如果给出品质不够完美的产品，就会害了客户，这不是我们想要的。

在最关键的时刻，我们团队的同事，义无反顾的支持了我。

经过几个月艰难的讨论，2019 年 10 月， 合宙团队做出了最终决定：

All In Cat1。

我们放弃 Cat4 模组， 放弃 NB 模组，放弃 WiFi 模组，同时， 2G 也不再做新项目的决定。

这样，我们可以轻装上阵， 专心把 Cat1 做好， 让客户能够放心用 Cat1 去做产品。

做出这个决定的时候， 我的内心是极度忐忑的。

虽然我很看好 Cat1 技术，但是我也非常非常担心 Cat1 的市场需求不按照我预想的方向发展；

我也很担心 NB 市场爆发，让合宙错失发展机会。

带着这样忐忑不安的心情， 每天的心都像揪在了嗓子眼，每天都感觉度日如年。

尽管如此，我们团队的产品研发动作， 一点都没有落下，每天都在加班加点的赶进度。

2020 年 1 月，疫情爆发，这是一个让每个企业陷入困境的灾难性事件。

合宙同事不为灾难影响，工程师们把调试设备带回家，回到家继续热火朝天的开发；

通过微信群沟通项目进展， 通过视频会议来沟通细节；

在疫情好转，重新回到公司正常上班的时候， 合宙的 Cat1 模组产品基本上就量产了。

2020 年 5 月，合宙顺利发布了第一款 Cat1 产品： Air724UG。

这款产品上市是团队花费了很长时间定义的，封装很小，品质非常稳定，功能也完

善，上市几个月，很快就卖爆了。

甚至到了 2024 年的今天， Air724 依然是一款非常流行的产品型号，每个月都有几十万片的销量。

这样的发展， 终于让我松了一口气。

2020 年， 随着 Cat1 模组的量产，以及大批量发货，Cat1 的优点很快就获得了市场认可： 品质稳定，网络覆盖好，用户设备的实施成本低。

而 NB 呢？

也如期爆发了。 尤其是在燃气表抄表，水表抄表的场景， NB 很快就占据了主导地位。

这个时候，再复盘 2019 年的决策， 如果合宙 NB 和 Cat1 两个全都做的话，可能就两个都做不好。

现在只做 Cat1， 放弃 NB， 保障 Cat1 的产品品质和服务品质，对客户，对合宙，都是双赢的。

假如合宙不做 Cat1， 只做 NB， 也有可能会有业绩， 但是 NB 的各种缺陷， 会使得公司的业绩很难长期持续。

就这样，发展到了 2020 年的第四季度，合宙 Cat1 模组的出货量超过了 300 万片。

根据中国电信后台的统计， 2020 年， 合宙的 Cat1 设备在网的数量， 是绝对的第一名。

我们团队的情绪也非常高涨， 都觉得合宙走对了产品路线， 抓住了时代的红利。

团队的每一个人，都期待在 2021 年大干一场， 继续高歌猛进， 拿到更多的市场份额， 然后一鼓作气，上市成功。

然而， 意外往往就是在你没有任何准备的时候， 悄然发生的。

就在合宙团队热情满满，干劲十足的时候， 意外， 发生了。

2 每个人的世界都完全不同

有位大哥，叫做 Jack，但是不姓马。

他是半导体行业的资深大佬。

在 2020 年 9 月的一天， 他很认真的告诉我说：“鹏哥，马上芯片就要大缺货了，很严重很严重的缺货， 你要提早做准备啊。”

我听了心里一惊， 就问他：“为什么会大缺货？”

他告诉我了几个线索， 以及他的推论。 这里不详细复述了，如果你想知道， 加我微信我私信告诉你。

我虽然很认同他的判断， 也很想提前多备一些货。

但是， 在 2020 年的环境， 像我们这种靠商业生存的公司，每年习惯性的动作，都是在持续生产出货，会定期跟上游元器件和芯片厂家谈价格， 期望从上游芯片的降价过程中，得到持续提升的利润，以及更大的市场份额。

尤其是在年底， 已经是谈价季， 商谈第二年的价格策略了。

期待芯片进一步降价，是一种习惯性的贪念。

这个贪念， 使得我虽然内心很认同即将缺货的逻辑，但是在行动上，我没有战胜自己的贪念，也就没有向团队发起提前备货的提议。

这个决定，曾经让我后悔的肠子都青了，有时候都恨不得自己能够跳进黄浦江去。

无论我是不是做缺货的应对，缺货就是真真切切的来临了。

从 2021 年 1 月开始， 各种芯片开始大面积的缺货，而且是断崖式的缺，几乎不给缓冲期。

合宙 Cat1 的先发优势积累的势能，这个时候变成了沉重的经营负担。

2021 年第一季度， 每个月正常的需求，在 150 万片左右。

但是由于供货不足引发的市场恐慌， 导致我们受到了订单挤兑。

客户们担心供货不足， 就拼命的提前下单。 我们每个月实际接到的订单， 超过 400 万片。

这种情况，是我从业 20 年来，从来没有遇到过的。

我真的是被打蒙了。

在这种巨大的不确定性的状态下，我的情绪， 也非常不确定。

这段时间，最怕的就是客户下订单，最担心的就是客户付款给合宙。

收了钱， 却不知道什么时候能够发货。

这种感觉，就像是每天都掉入无底深渊，不知道何时才能够超生。

订单需求每个月超过 400 万片，但是实际拿到的货，只能够生产需求量的八分之一。

支撑我们每天还在奋斗的，就剩下了心里的那个幻想： 这么严重的缺货，应该过几个月就缓解了吧？

然而跟销售团队的兄弟姐妹们相比，我的郁闷根本不算个啥。

他们在第一线，每天看到的都客户的绝望。

这个绝望情绪，很快的就插入到了每一个销售同事的心脏。

销售们的绝望， 又传递到了其他同事。

从而产生了一种特别魔幻的场景：

订单极多， 但是所有人都觉得合宙面临巨大的危机。

所有人都知道，失信于客户，就是自绝于江湖，未来的路将会异常难走。

事态越来越恶化， 合宙拿到的货， 能生产出来的产品， 甚至都无法满足一个量大的客户的订单需求了。

已经很少有客户讨论产品的性能，也没有客户在意售后了，价格也不在乎了。

只要有货，怎么样都行。

一切都像极了末日来临前的疯狂。

怎么办?

就在这种面临客户和销售同事们的巨量的绝望炸弹轰炸，我自己也快要爆炸的的时候， 一个老朋友来找我聊天，他叫做 Edward， 是一个企业陪跑咨询师。

他看我每天的状态不太对， 跟我聊了十几分钟， 其中有句话让我幡然醒悟:

“在危机时刻，你最看重什么，就去做什么。考虑过多的因素，只会让你陷入噪音的干扰。”

听到这句话， 我就像被电击中了一样。

对啊， 每个人的诉求都是不同的， 1 万个人，有 1 万个不同的诉求。

在这众多的不同诉求里面，找到对社会总损失最小的选项， 对合宙的伤害不就是最小的了？ 这不就是我们团队一直追求的吗?

于是， 我们马上开会， 定下了一个供货策略:

全面保证小客户供货，越小的客户， 供货越是要保障，确保小客户都能够拿到一部分货。

为什么做这个决策呢?

因为越是体量大的客户， 供应链能力越强， 如果合宙的供货不足， 大客户们可以迅速导入其他的供应商， 弥补合宙的供货不足的缺口;

而小客户，研发能力和供应链能力都弱一些，如果合宙供货不足， 这些小客户很可能就只能停止业务，就有灾难性的后果。

所以合宙在 2021 年上半年开始， 全面向小客户倾斜供货。

因为这个决策， 合宙丢失了绝大多数的优质大客户， 却让众多的小公司客户渡过了难关。

正如我们预测的，大客户们很快就导入了合宙竞争对手的供应商，安然读过了供应链危机。 甚至有些大客户，一口气导入了七八家供应商，展现了超强的经营能力。

回想起来，如果不是 Edward 的指点迷津， 让我们团队能迅速的做决断， 很可能我这辈子都无法面对自己的良心。

有了这样的结果， 销售同事们虽然仍要承受丢失客户的痛苦， 但是总算是靴子落地， 死也死了个明白，比每天吵架的状态还是好了不少。

但是，另一个问题却又出现了：

由于缺货太严重，客户们的恐慌，造成了一种很魔幻的错觉：

我们的产品是不愁卖的， 所以技术团队和运营团队都出现了不同程度的懈怠。

也不怪同事们， 几个月的缺货的痛苦，让同事们沉浸在产品脱销的恐慌和莫名其妙的得意，二者相夹杂的情绪交替，整个团队几乎失去了产品的改进目标，

也不是所有同事都没有目标。

我发现有个别同事，在研发一个叫做低功耗分析仪的产品， 没错， 是仪表。

一个研发生产销售模组的公司， 居然内部出现了仪表的项目；

我作为这家公司的 CEO，却对这个项目一无所知，真是怪事一件接着一件。

这个项目，一开始就不是走的正常路子。

合宙有 30 多个技术交流 QQ 群，有好几万个开发者的发烧友在群里面，每天聊得很嗨。

我们的同事， 是应群友的要求， 开始琢磨这个低功耗仪表。

当时市面上， 确实没有一款能够精准的测量物联网设备的超低功耗，并且用 PC 端软件精准的展示低功耗的曲线和分析结果。

面对微安的功耗， 工程师们，只能用电流设备粗测一个平均电流值，但是没办法精确知道是在哪个时间点功耗高，哪个时间点功耗低，因此很难进行精确的低功耗性能改进。

当这几个同事把设计的方案和思路，发到 QQ 群的时候，很快受到了群友的追捧。

关注者迅速超过了 5000 人。

在发布第一个内测版本的时候， 尽管问题很多， 但是开发者们如获至宝，立刻抢购一空，还在 B 站发布了众多的评测视频，热度有点瞠目结舌。

然而， 更多的同事，却不赞成这个仪表项目，他们的看法也有道理：

合宙好不容易聚焦了 Cat1， All in 了 Cat1 的赛道，这个时候分散资源做仪表产品线，这么一个合宙几乎没有积累的产品线， 会让合宙的竞争力下滑。

所以他们认为，研发低功耗分析仪的同事不务正业。

而我， 也是持反对低功耗仪表产品的同事之一。

这个低功耗仪表项目，一度面临被叫停的境地。

好在，这个项目最终没有被叫停。

原因也很出格：

同事很聪明， 采用了敏捷开发的方式， 每一次迭代， 都发布到了 QQ 群，每一次发布，都有好几千个用户，很快的给到测试结果，协助下一次迭代。

这个时候停掉项目， 无疑是失信于这几千个用户，会触犯众怒。

无论是我，还是其他同事，面对这几千个用户的热情，都无法那么绝情的关闭项目。

于是，低功耗分析仪，就这么保留下来了。

从 2021 年上半年开始， 开始研发低功耗分析仪，一直到 2024 年初，才开始真正发布量产。

低功耗分析仪， 在接近 3 年的时间，没有直接带来利润的收益，一直在迭代，迭代，持续迭代了几十个硬件版本，数百个软件版本。

这也是让其他同事们无法接受的主要原因之一：

投入地久天长，回报遥遥无期。

然而，随着低功耗仪表的内测， 却有了另一个意外的收获。

我们的 Cat1 模组，功耗比绝大多数的竞品的功耗都要更低。

我开始并不知道这个信息， 一次出差，跟客户聊天，客户告诉了我，说你们的功耗比竞品的同类型号低了 50%， 我感到非常纳闷，我们内部从来没有做过跟竞品的功耗对比测试，没什么功耗会这么低呢？

回到公司仔细调查才发现， 这都是低功耗仪表内测的功劳。

在做仪表内测的时候，我们的工程师同事，同样也是内测用户。

在用自己的仪表， 测试自己的 Cat1 模组的时候， 顺带就把功耗给优化了。

要知道， 在过去的很多年，蜂窝模组，包括 Cat1 的功耗， 都是不被重视的，我们天然就认为 4G 这种蜂窝模组是功耗大户。

这个时候， 也仅仅有很少的客户，在意功耗的高低。

合宙的任何一个 KPI 目标里面，都没有优化功耗的一项。

但是在这段因为缺货，失去了产品改进目标的日子里， 有些同事，自发的在优化功耗，因为他们知道，虽然环境很魔幻， 但是依然有一小部分客户， 是非常非常在意功耗的。

优化的任何一点点功耗，是能够让这些少数派客户心情愉悦的。

这就是同事们的目标。

在失去了目标的时候， 让客户们的心情更高兴， 就是合宙同事的目标。

这些同事，在合宙内部，也是少数派。

但是无论如何，从这个时候开始，合宙团队有人开始越来越关注功耗的每一个细节，开始了漫长的低功耗优化的过程。

而在这个低功耗小插曲的背景， 整个 2021 年，整个行业在意的，依然是能否有货。

在接下来几年的不断复盘过程中， 对于没有提前备货的错误， 我逐渐开始释然：

既然是有贪念， 那就是我的认知，还无法达到克服这个贪念的程度。

只有我能很彻底贪念，排除贪念带来的困扰，才能真正的做出实事求是的决策。

认知以外的钱，确实是赚不到的，也无需过多的后悔。

而缺货带来的痛苦，让我也在思考，我们到底能为这个社会带来什么？

而又一个高人的出现， 给我指点了迷津， 让我茅塞顿开。

我义无反顾的加入了他的组织。

加入这个组织半年来，对我的帮助，甚至超过了我从业 10 年的总和。

具体哪个组织，由于篇幅所限，下回细讲。

如果你等不及想知道，可以扫下面的码加我微信，我私信告诉你。

3 不再说愿景使命价值观

在缺货的日子里， 我一直在思考：

合宙的愿景，使命，价值观是什么？

为了找到答案， 我拟定了不少于 10 个版本， 跟团队也讨论了至少一年多。

我一度以为， 一家公司没有愿景使命价值观的话， 是无法长久走下去的。

但是讨论来讨论去，每个版本的愿景使命价值观，都不像是我和我的同事心里的版本。

我们有个安徽的客户，公司叫做 UB，老板叫做小麻，95 年生，很年轻，技术很厉害。

这家公司在 2018 年创办， 刚刚创办就跟合宙有合作。

小麻的公司只有 8 个人，研发销售旅游景点的景区自行车。

因为疫情，UB 公司在 2020 年颗粒无收，到了 2020 年底，账上的钱只够花 2 个月了。

小麻跟团队的人商量了下， 决定再坚持一年，如果一年后还没起色，公司就解散了。

在这一年的时间，所有人都每个月拿 2000 元，保持最低消费水准。

幸运的是， 2021 年 3 月， 小麻接到了一个救命的订单，做 6000 个共享洗手液设备，每个能赚 300 元， 顺利交付的话，能赚 180 万， 可以让自己的公司暂时渡过危机。

这个设备是要安装在存量的公共场所， 给流动人口做日常消毒用的。

所以，大多数的安装地点不容易取电，所以只能靠电池供电。

为了方便对洗手液的管理， 需要实时监控洗手液的液位，所以 4G 需要实时在线，定期汇报洗手液的状态。

并且， 由于运维成本比较高， 要求设备半年才能换一次电池。

所以，这个项目对功耗要求极高，项目方找了好几家公司，都搞不定续航问题。

小麻仗着自己技术过硬， 更多的是把这个项目看做了救命稻草。

小麻亲自上阵，研发了 3 个月，到了 2021 年 6 月，各项功能都顺利完成，但就是功耗指标达不到，无论怎么优化， 当前的性能，都是用 1 个月就没电了。

眼看订单要黄了，小麻找到我们，希望能帮忙优化功耗。

我们收到需求后， 也知道了小麻团队的危机生存状态。

我们投入了水平最高的工程师，帮助小麻团队优化功耗。

但是花费了 2 个月时间，熬夜调试代码，周末也测试，

到了 2021 年 8 月， 依然只是把功耗降低了 20%，距离项目方的要求相距甚远。

项目方又等了小麻 2 个月， 我们团队跟小麻又做了 2 个月的努力，然而完全没有进展。

这个项目终于还是黄掉了。

而小麻团队，再也没有接到第二个救命的项目。

他的公司， 在 2022 年的春节前， 疫情发生 2 年后，关闭了。

关闭公司的那段时间， 他没有联系我， 我也没有联系他。

到现在也没联系。

我无法知晓他是什么心情，

我也不清楚，他目前在做什么。

对于小麻，我有深深的无力感。

如果是放在 2024 年 3 月的今天， 当时的那个性能， 我们可能一天时间就给他搞定了。

但是在当时， 我们的能力真的是达不到。

我们这么一家成立不到 10 年的小公司， 当客户遇到难题，我们无法帮客户解决，眼睁睁看着客户死掉， 再探讨愿景使命价值观有什么意义呢？

从小麻公司关闭， 一直到今天， 我再也没有讨论过愿景使命价值观。

有人问：合宙的愿景是什么？

我说还不知道；

使命是什么？

也还不清楚。

价值观是什么？

实事求是。

我们的能力还远远不够， 也不可能解决所有客户的问题。

是否可以设定一个能力圈，我们这个百人团队，在能力圈范围内， 是否能把问题解决掉呢？

如果真要设定一个能力圈， 我愿意设定为低功耗能力圈。

我们要把功耗分析仪做到最实用，让客户们能用分析仪很精准的调试电流，把很精细的电流曲线， 展现客户的产品手册里面， 他的硬件产品的竞争力可能就会上升一截；

我们要把 4G 模组的功耗降低一个数量级， 以后再持续降低， 真正要让 4G 模组能够像蓝牙一样省电。

把这两件小事情做好， 在这个小能力圈范围内，做到让客户能轻松赚钱， 难道不比愿景使命价值观好一万倍吗？

所以， 在我们的能力圈无法进一步扩大之前，我坚决不再讨论合宙的愿景使命价值观了！

再细想一下， 愿景使命价值观就是个工具， 对齐团队的工具， 让团队能够有个基准的远景目标和行为准则。

当团队规模不够大的时候， 首要做的，应该是构建能力边界， 在能力边界之内， 给到客户最好的体验， 同时让自己的团队有高于市场平均水准的回报。

所以， 团队规模超过 300 人之前， 都不宜提出愿景使命价值观，而更应该聚焦构建自己的能力圈。

至于为什么是 300 人， 我也不知道，纯属于拍脑袋。

300 人还是 500 人并不重要了，就是这个意思，你肯定懂。

2021 年 5 月， 我偶然看到一个号称是科学创业的组织， 名字叫做一堂， 在招新，只需要付款 9.8 元，即可加入。

我怀着花了 9.8 元的爽感， 加入了一堂的为期一周的试听营。

在这一周里面， 我听了一次直播课， 听了 4 次录播课， 学习了一个关键认知：

所有的商业都是基于假设的。

而提出关键假设， 再验证关键假设， 是创业团队首要应该做的基本动作。

这句话， 突然让我有一种醍醐灌顶的感觉。

我一下子对过去十几年的所有的犯过的错， 有了一个交代。

原来我过去十几年，就没做假设， 任何的假设， 我都错误的当成了结果，从而省掉了 验证过程， 直接就习惯性的 all in 了。

虽然 all in Cat1 是没问题的，但是在 all in Cat1 之前， 我还 all in 了好几个， 都失败了。

这种 all in 的做法， 在一堂的 CEO Truman 看来， 就是最典型的创业错误， 最

不科学的创业动作。

一堂这个传授科学创业的公司，不仅教课， 而且强制学员写作业。

这一周里面， 我连续写了十几份作业，超过了 2 万字， 把我的失败经历做了简单的复盘， 很快就超过了 50 分。

这个拿到 50 分学分的速度，到今天还排在 1 万多个一堂学员的第三名榜单。

但是很快的， 因为一个我从来没有发现的原因， 我的学习热情又极速下降了，几乎是下降到了冰点。

4 高质量的复盘是成功之父

我是 2021 年 5 月加入一堂的。

在第一周疯狂写作业之后，很快就写不下去了。

因为 一堂的作业， 总是让我复盘自己的项目经历，自己的创业经历。

但是，只要我一复盘， 我就发现， 心里非常的痛苦。

要么不写， 只要一写，全都是在往自己的心里扎刀子一样。

对于以前的自己， 本来只是觉得， 有点不太对，应该改正的。

但是， 真要是把以前的项目经历写出来， 感受立马不一样了，那感觉是自己是傻缺的不能再傻缺了。

这种感觉， 真实太难受太难受了。

就好像是有点活不下去的那种感觉。

所以，我基本上就写上一两百字， 就草草结束提交了，结果满分的 6 分作业， 往往只能得到 2 分。

2 分作业，真是个不咋行的反馈，让我瞬间对作业失去了信心。

就这样，长达半年的时间， 基本上没怎么写作业。

不写作业的话， 学习的效果， 几乎等于没有。

就这样，我继续开我的会，继续跟同事做产品，继续拜访客户，就是不写作业了。

直到。。。。。

上海封城了。

在家里面呆了接近 3 个月时间，不能出差，不能面对面开会。

偏偏在这个时候， 一堂的直播课程，还上的很频繁， 正好适合我这种在家无聊又闲不住的人。

于是，很奇怪的事情发生了， 虽然在家里面呆了 3 个月， 但是我好像比去公司上班还要忙。

一堂的 CEO， Truman， 上直播课，经常会到夜里 1 点多钟， 而我上完课，还继续写作业， 有时候写到凌晨 4 点钟。

解封的那天，我查了下，自己写的作业， 已经超过了 20 万字了。

人的感受， 就像一层很薄的纸， 一旦捅破了，后续就畅通无阻了。

自从我第一份作业做了一次过去的失败项目复盘，得了个 5 分的优秀作业之后， 就一发不可收拾， 把自己创业 8 年来，经历过的所有的错误，都尽可能原原本本的做了复盘。

20 多万字作业， 至少有 15 万字，都是在做过去 8 年的错误复盘。

这是一种非常奇怪的感受。

半年前，对于复盘还是如坐针毡， 生不如死；

封在了家里面， 突然就对自己的深刻复盘， 有点甘之若饴， 享受其中了。

虽然在《我的物联网大学》里面， 我已经做了一些失败项目的分享，但是跟在一堂的作业复盘比起来，我的物联网大学的内容， 就像隔靴搔痒， 蜻蜓点水，非常不够真诚，总是想给自己留面子。

可能是我的作业确实是写的比较情真意切，也可能是我的作业里面的失败案例失败是有点多不胜数，总之， Truman 注意到了我， 从作业里面看到了我， 让同事跟我联系， 对我的作业进行了整理， 还让我讲了一堂案例课， 认识到了一堂的上千位同学，真的是让我占尽了便宜。

但是，一直到写了 20 万字作业， 我还不太清楚， 我这就是在做复盘， 我只是以为我在写作业，拿高分，虽然这个高分没一分钱的奖励，还得花钱付学费。

一个成都的客户的，名字叫做老练， 来上海出差， 来我们办公室聊天。

老练说：“我们做业务也有六七年了，但是总感觉是在被惯性推着走。 老秦你说说，我们怎么判断，现在做的业务， 方向是正确的还是错误的？”

我听了之后， 心里咯噔一下， 完了，我也不知道啊， 只好说：“老练，你觉得呢？”

老练说：“我觉得吧， 好像是得赚钱，不赚钱的事情，不能做了，没啥好处”。

我说：“很对啊，还有呢？”

老练说：“可是有些产品，赚钱几个月， 就不赚钱了，很快陷入同质化，价格战了，节奏跟不上啊”。

我脑子里面飞速转动， 还是没什么头绪，于是我对老练说：

“我的困惑跟你一样的。 合宙做过很多产品， 大多数都无法持续，原因有好几个。”

老练说：“比如说呢？”

我说：“我们做的儿童手表品牌， 还买了愤怒的小鸟的 IP 授权，但是没什么卵用，项目还是死了，亏了好多钱。 这个项目无法持续的原因， 主要是我的能力不足， 对营销完全没有认知， 对品牌的经营没有认知，而且我也没有花足够的时间去学习品牌营销，

不死才奇怪呢。”

老练说：“你做这个项目之前， 我就知道你这个项目会死，不奇怪的。”

我尴尬的勉强笑了下，继续说：“我们做的户户通定位模块项目， 主要是因为市场需求不持续死掉的。”

老练说："不对， 户户通定位模块， 及时有需求， 你也会死，因为没有利润呀，你的定价权几乎没有。"

我更尴尬了， 只好强作镇定的说： “老练你太牛了，一语说破梦中人啊！”

老练说：“得了吧， 你肯定知道原因， 只不过你还没像我一样花时间总结。”

我跟老练聊了一下午加上一个晚上， 把我们两家公司的所有失败项目， 以及现在还在做的相对成功的项目， 都做了详细的梳理， 讨论， 然后， 得出了个结论：

产品的方向是否正确， 还是没办法提前判断。

但是，经过这次跟老练的深入沟通， 我内心一直有个种子， 时不时的拿出来问自己， 产品的方向， 真的就没有一个判断方向是否正确的准则吗？

直到半年前， 有一位一堂同学， 给我看了一个讲模型思维的视频， 这个视频的 UP 主非常年轻，一看就是 90 后， 他引用了吉姆柯林斯的《从优秀到卓越》的思考框架， 提出了主营业务的三维度模型：

1， 能不能有做到最好；

2， 能不能持续盈利；

3， 团队是不是持续有热情。

我感觉有一点不完整， 做了下修改：

对于第一个 10 年的初创企业来说， 产品跟业务是不是正确， 应该看是否同时满足三个标准：

1， 是否对社会有正向的价值；

2， 是否持续盈利；

3， 团队是否可以保持足够的热情，为了直到最后而不停止。

其中第三条， 团队是否有热情，也要满足一个二八原则：

做 80%熟悉的事情， 同时冲击 20%陌生的区域继续提升。

满足二八原则的话， 团队的热情，是可以持续的。

合宙成立 10 年以来， 早期失败的业务，

儿童手表的品牌业务， 是因为我和合宙团队的能力差距都太大，导致无法保持持续的热情从而无法持续；

定位器的整机业务， 由于无法持续盈利，而被迫终止；

户户通的模组业务， 也是由于无法持续盈利而终止。

而当前合宙正在主力进行的 4G 模组业务， 按照这三个维度进行匹配：

1， 对社会有价值， 帮助到了几千家企业， 实现了产品的高效运营，客户盈利了，这是合宙对社会呈现的价值；

2， 是否持续盈利， 这个毋庸置疑， 4G 模组是有利润的产品；

3， 团队是否有足够的热情： 模组业务一直合宙团队的老业务， 同时又增加了低功耗的挑战， 团队做起来非常带劲。

对合宙的功耗分析仪业务， 也按照这三个维度做匹配：

1， 对社会有价值： 用了合宙的低功耗仪表， 能够让企业的硬件产品卖的更贵，增加社会总利润；

2， 是否持续盈利： 合宙的仪表的毛利率是健康的；

3， 团队是否有足够的热情： 仪表业务在合宙做了 3 年， 合宙的仪表团队感受到了自己的巨大成就感，未来持续 10 年继续进步没什么问题。

经过这次复盘， 合宙的低功耗 4G 模组业务， 合宙的功耗分析仪业务， 持续发展 10 年， 是非常笃定了。

那么问题来了， 这两个业务， 都是很难的业务， 怎么能保证合宙的所有同事，都能高效的协作，把产品性能做到最极致， 给社会真的呈现极大的价值呢？

我还是没有特别明确的方案，非常期待你给我建议！

5 逐字稿是很笨很笨的工具

到了 2022 年，合宙成立 8 年了。

这 8 年的时间，我在公司里面，听到的最多的话是：

“我没有得到通知啊”

"没人告诉我这件事啊"

“我不知道这个情况啊”

然后。。。。。， 工作的聊天，就这样聊死了。

未来对付这种情况， 我演化出了超强的记忆力， 对合宙的所有项目如数家珍， 产品的细节了如指掌，几乎所有会议我都参加。

我就像个救火队员一样，合宙的所有信息， 我都是最熟悉的。

我一度也很自豪， 原来我的能力这么强啊。

我们开会的时候也很多， 会议持续的时间，经常是超过 3 小时， 有时候周末闭门开会 2 个整天。

疫情刚开始的时候， 我趁着酒店便宜， 每个月拉同事们在酒店封闭开会 2 天。

开会的内容呢， 超过 80% 的时间， 都是在汇报信息， 挨个汇报，大家在下面听。

信息同步的差不多了，会议也快到时间了。

然后就是延长时间， 讨论方案，讨论决策。

就这样，一天的时间，就被会议给占用了。

这样的工作方式， 协作方式， 持续了 8 年。

直到有一天， 我在上一堂的课程的时候， 发现 Truman 讲课用的是逐字稿，Truman 说的内容， 全都在逐字稿上。

他讲课 2 个多小时， 稿子有 2 万多字。

难道 Truman 不是念稿子吗？

还真不是。

Truman 对着逐字稿讲课， 基本的课程内容， 可以念稿子。

但是正因为有了逐字稿， Truman 可以做随意的即兴发挥， 把自己的精力放在讲课的情绪表达， 对面的学生，就能够感受到 Truman 的游刃有余的自信， 以及散发出来的浓厚的讲科学的情感。

看样子，逐字稿真是个好东西啊， 我得用起来。

我在合宙的团队里面，呼吁使用逐字稿， 但是没什么响应。

我自己写逐字稿， 发给所有人。

我还在飞书的公告栏，给每个人创办了个人专区， 但是只有我一个人写。

就这样，我坚持写了三个月。

写自己的思考，写读书笔记， 写会议前的准备， 写市场预判， 写客户沟通情况。

就一直写。

在开会前，我也把开会要同步的信息， 写成逐字稿， 让参加会议的人先看一遍， 再开会。

三个月之后， 我发现老陆也开始写逐字稿了。

再之后， 小龙也开始写逐字稿了。

再过两个月， 我发现团队的一半的人都开始写逐字稿了。

还有些同事问我： 我打字慢，写逐字稿有困难，怎么办？

我回答说：“练打字呗，还能怎么办？”

接下来， 我们达成了一个开会制度：

如果没有合格的逐字稿提前同步背景信息的话， 会议不允许召开。

这个制度，让一些同事很为难。

但是架不住大多数同事，都觉得有了逐字稿的话，开会的时间最起码节省了三分之二，都愿意拥护这个逐字稿，少开会早下班，谁不想要啊。

于是，开会之前，把这个把逐字稿作为前置任务，很快就得到了实施。

再过半年，我们突然发现，有了逐字稿， 80%的会议都不用召开了。

反正之前的会议，都是同步信息，既然逐字稿同步信息了， 确实不用开会了啊。

不开会， 就没那么忙了。

针不戳。

说了这么多逐字稿的好处，

团队里面推行逐字稿，有什么坏处吗？

坏处也很明显。

最明显的坏处是：

招聘变难了。

以前招聘，都是问一遍做过啥，都会些什么， 描述一下工作经验，通过对细节的反复询问， 基本上就能入职了。

现在可好， 同事们都习惯逐字稿了， 面试新人通过之后， 就给他布置一个逐字稿的任务， 先写个几百字上千字交过来。

基本上 10 个里面， 能有一个交稿的就不错了。

所以，合宙现在招人非常非常的困难，

就写稿子这一条， 就拦住了想入职的新人了。

不过现在我也知道了，逐字稿并不是什么新鲜事。

在新东方的教研体系里面， 很早很早就启用逐字稿了。

新东方老师们的妙语连珠，包袱不断，都是提前经过逐字稿千锤百炼过的，不然你以为都是天才吗？ 没那么容易的；

董宇辉在镜头前深情的故事，没有幕后逐字稿的加持， 怎么可能发挥的如此丝滑？

成功的演讲， 只是靠着 PPT， 是绝对要卡壳的。

只有事先准备足够详细的逐字稿， 才能够在演讲台上游刃有余。

要知道，演讲的法则， 只有 7% 是靠语言， 有 38%是靠声音的感染力，还有 55%是靠肢体动作的表达。

但是，如果没有逐字稿的话， 这 7%的语言发挥失常，卡住了， 整个演讲的效果，可就是要演砸了。

2018 年， 我在云栖大会的演讲， 由于不懂得准备逐字稿， 而是写了个 PPT，就上去讲。

结果面对台下几百人，讲到一半，突然大脑一片空白， 足足愣住了 10 秒钟左右，才恢复正常。

这尴尬的一幕，我简直一辈子都忘不了！

成功的演讲者的逐字稿， 不仅仅有详细的文字，逐字逐句的文字， 这个文字是口语化的，就是原封不动的照着讲的逐字稿；

并且， 逐字稿里面，还要把停顿节奏， 重音，轻音，疑问，写上去；

更甚者，还要把表情写上去；

更极致的， 还会把肢体动作，注视观众的时间，互动的细节，都会写上去。

所有的成功，都不是偶然的，都是大量的细节堆砌起来的。

所以，董宇辉的成功， 一定不仅仅是他的才华， 更多的是靠他的勤奋。

学而思的老师们， 讲课倍儿溜， 也都是经过逐字稿练过 N 多遍的；

在美团内部，也是早就用逐字稿实施了好多好多年了。

美团内部不用 PPT， 只用逐字稿， 是王兴一直以来强调的基本功之一。

逐字稿写起来是真麻烦， 没有打字的基本功， 确实做不了逐字稿。

但是挺过了逐字稿的艰难， 剩下来的几十年时间，就是享受逐字稿的便利了。

比如，免掉了 90% 的会议，就是逐字稿的最大优点。

通过逐字稿， 合宙团队慢慢的让我卸任了救火队员的岗位。

我像是开启了我的另一种人生， 突然有点不习惯了。

这种不习惯， 会给我带来另一种犯错的形式吗？

6 成功有秘诀吗？还真有

本章的故事主角，姓侯， 也是理工男， 今年 38 岁， 在经营无锡的一家软件公司，主营的是软件外包业务，有十多个员工。

侯总的客户群， 大多数是软件实力不强的中小公司，这些客户没有软件团队，就把想做的 WEB 管理系统，小程序，APP 的开发工作外包给他做。

最近几年，做软件外包有个最大的问题，就是收到首期款之后，软件做完了，客户迟迟不验收，导致尾款收不到。

在等待尾款的过程中，客户因为经营不善，或者是项目不顺利，客户放弃这个项目了。

这个比例，甚至能超过 60%。

这就导致侯总的公司，几乎是在依赖首期款支撑公司的运营，但是首期款往往只有合同金额占比的 30%，甚至更低。

所以，侯总的软件外包业务，一直都做的很艰难， 艰难到啥程度呢？

有时候发工资都得延期。

侯总自己， 也只拿很少的工资。
侯总跟银行也贷了几百万， 只能贷一年，每年都要还款续贷。一旦银行不给续，资金链就很可能会断掉。
所以，每次到贷款还款的时候，侯总都是心惊胆战的。

如果侯总能够接到大企业的订单， 比如华为，比如中移动，情况会好非常多。
但是侯总的公司规模太小，实力不够，入不了大企业的法眼。

所以，只能接到中小公司的订单， 如此循环往复。

这看上去挺无奈的。

侯总很羡慕合宙有一个持续的硬件业务， 经常来我们公司探讨交流。

侯总希望他的公司，是不是也可以借助合宙的硬件积累， 转型到软硬结合的业务形式。

我也感觉侯总的这个想法非常好。

最近几年，硬件业务比较卷，合宙的硬件毛利也一直提升不上去。

所以我也更希望，合宙在销售硬件的同时，给客户提供软件硬件一体化的系统，给客户的价值就更大了，客户付费也就更多了，合宙的毛利率会有一个比较大的提升。

所以，最近两年，我跟换侯总的交流挺多的，侯总经常会来到上海跟我们团队交流，也做了挺多尝试。

侯总尝试的第一个项目， 是一个定制项目。

简单来说，为一个大集团公司定制一个室内的智能家居产品，在装修工地的现场， 对手机 WIFI 的信号进行探测，从而知道有多少人在现场施工，停留了多长时间， 达到较低成本的对全国各地的装修施工的现场进行数据采集的目的。

最终客户的集团公司，实力很强，但是对通信设备是完全没有认知的。
而合宙跟侯总的团队， 都没有投入，也不可能投入专门的人力去跟客户反复沟通需求，制定产品定义。
因为合宙跟侯总的团队成员， 都有自己的项目在进行中， 都有自己的考核压力，没办法分出大块的时间去调研新的场景需求，更不可能去做完善的新场景的产品定义。

这个项目， 还没启动，就无疾而终了。

侯总尝试的第二个项目， 是智能工厂的管理系统。
侯总经过粗略的调查，发现大多数工厂的管理，都是靠人工的，效率低，容易出现差错，如果在工厂的关键节点上，搭配数据采集的硬件，把数据上传到云端， 并且针对这些数据给出工厂管理的指令，可以把一个传统的中小规模的工厂， 改造成为一个智能化的现代化工厂。

在这个过程中， 侯总也能得到不错的回报。

想法很不错。

侯总出差到每一个有合作意愿的工厂，实地考察，现场交流，跟工厂老板谈判。
在现场沟通了 20 多个工厂之后， 侯总发现每家工厂的设备都不同，管理团队的方法也不同，如果对工程进行智能化改造的话，每一个工程都是一个定制项目， 费用高达数百万。

几乎没有哪个工厂， 能够负担这个费用。

所以，第二次尝试， 也失败了。

侯总没有气馁， 又做了第三次尝试。

这次，侯总瞄准的是，共享充电宝。

侯总经过调查发现， 经过疫情的冲击， 传统的充电宝大品牌， 都受到了不同程度的冲击， 再进一步扩展业务的动力非常不足。

这个时候， 如果他来组织团队， 购买品质不错的共享充电宝的产品， 去发展商铺门店用户，机会很大。

在亲自带领 3 个人，拜访了几百家门店之后，侯总发现， 即使商铺门店愿意合作， 购买了他的充电宝， 侯总也很难赚到钱。 充电宝现在已经是渠道的话语权极大的一个市场状态了， 用户支付的重复费用， 侯总甚至分不到 20%。

侯总尝试说服门店买断充电宝硬件， 侯总只抽取服务费， 发现门店愿意接受的硬件价格也太低， 几乎是亏本销售，即使做进去，后期收取的服务费， 也无法覆盖运营成本。

所以，第三次尝试也失败了。

在这三次尝试的过程中， 我配合着侯总，给他提供硬件方案的咨询服务，也给他介绍靠谱的硬件供应商， 有时候还去跟他一起到现场去沟通。

这个过程，差不多持续了一年多。

在这一年多的时间里面，侯总很忙。

他的电话不断，从早到晚的回复微信。
几乎每周都有两三次应酬， 体重增加了 30 斤！

第三次失败后， 侯总终于意识到了一个比较重要的问题：

现在中国的商业形态， 在几乎所有的业务范围内，几乎都不是 1 和 9 的关系了。

任何的业务， 都只有最多 10%的玩家是良性运营，另外 90% 的玩家，都是交学费亏钱运营的。

要想在一个业务上良性发展， 需要在开始的三年内， 学习，适应，掌握 know how，也就是掌握窍门。
具备了预判这个行业的动态发展的能力， 才有可能真正成为这个赛道的玩家，做到心中有数，游刃有余。

侯总对我说：“像我这样不到半年时间就换一个赛道的，基本上就是在学习常识，等学到了一点常识之后，发现不可能有短期回报，我就撤退，去换一个赛道。 但是换了之后，发现还是重复一样的过程。“

我跟他说：“我以前做儿童手表品牌，也是如此，做了 2 年，发现学费交过头了，被迫终止业务了。”

侯总说：“经过这一年多的不务正业的摸索， 我现在还是觉得，方向是最重要的，方向正确，长期干，成功的概率会更高。”

我想了会，跟他说：“方向正确这点固然很重要很重要， 但是吧，方向走对的人，也很多啊，但是也还是成功率不高吧？”

侯总说：“确实不高， 要求短期回报的话，确实成功率不高，就像我， 半年就要求有回报，这几本不可能。所以不能要求短期回报，得要长期回报，三年不行，就五年，五年不行，就十年，肯定会有回报。”

我一拍大腿：“侯总，你好像这一年多的蹉跎没有白费呀！”

侯总有点郁闷的叹了一口气：“其实我还发现一个问题， 即使是方向正确， 不要短期回报，可能也还不行。”

“为啥？”

侯总说：“方向正确了， 如果不够勤奋， 可能长期也没啥结果。 我看到的相对长期成功的人，都是工作狂，勤奋的很， 起码都比我勤奋。 可能像老秦你这样的工作狂， 成功率更高点。”

我哑然失笑：“你这么说我，我可真高兴！ 我确实是不工作，就浑身难受。”

侯总说：“方向正确的时候， 可能也能让人情不自禁的愿意当工作狂的。”

我这个时候， 突然灵光一闪：

“侯同学你说的会对啊， 方向正确的时候， 要想往前走，要多很多事情啊， 第一个，要做逐字稿，把思路整理清楚，让内部的所有人都明白， 第二个，要频繁发朋友圈，发公众号， 发成本低的广告，让外面的人都看明白； 第三个， 对外说的内容，还得好理解，得反复的讨论什么内容好理解，什么内容不好理解； 第四个， 遇到了有人要合作，得赶紧加班加点赶进度，第五个。。。。 ”

侯总赶紧打断我：“得得得。。。。 赶紧停下，老秦， 你把我说懵逼了，慢点慢点！”

侯总停了一下， 在白板上写下了三行字：

1， 方向正确；

2， 像傻子一样勤奋；

3， 不期待短期回报。

然后回头对外说：“老秦，是不是这样？”

我笑了：“侯总你真牛逼！ 你提炼了一个成功的三要素模型！”

侯总说：“我干了快 20 年的软件开发了， 我得回去琢磨一下， 我怎么能把软件做的深入再深入，做个 10 年规划，可能会更很不错！”

我也站起来说：“我们做硬件 20 年了，低功耗也研究了快 10 年了，合宙再降 10 年的功耗吧哈哈”。

在本章的结尾，我把业务成功的三要素送给你：
(1)方向正确， (2) 像傻子一样勤奋，(3)不期待短期回报。

跟你说句实话：
我在看到这个三要素之后，我的内心确实轻松了，释然了，焦虑似乎也少了很多。

如果你看到我和老陆的朋友圈发的内容的话， 你可能会更加理解我的心情。

下一章，我会讲一个更有意思的模型。

7 不可能的行动变成可能，也有方法

有件事情，说出来你可能不相信：
合宙在 2014 年创办，但是直到 2019 年，都没有销售团队，没有一个销售员。

在这几年时间里面， 都是靠着客户们主动来追着我们要货。

这也是合宙的发展一直都很慢的原因之一。

其实，我本人不懂销售， 也是长期没有销售团队的重要原因。
因为不懂，所以逃避。

从 2019 年开始， 我发现这事不能拖了，于是合宙开始有了销售团队。

销售团队建立之后， 效果是真的好。

客户们有了专职销售做沟通， 做服务， 信息一下子通畅了， 整个公司的运营效率像是提速了一样。

客户的数量，突然之间，呈现了数量级的增长。

问题来了。

客户的增加， 带来工作量的急剧增加，曾经的慢节奏的工作， 无法适应当前的需求了，整个团队都需要提速。

团队提速最大的挑战，就是信息的高效同步。

在 2021 年，逐字稿的导入， 就大幅度的提升了内部的信息沟通效率。

但是，还是不够。

尤其是在客户信息汇报的环节，总是会遇到比较大的困难。

举一个简单的例子:
10 个销售同事， 可能有两三个，能把客户的情况，很有逻辑的向大家书面汇报。
客户的项目情况，项目进度，遇到的困难，客户的关键人，关键人的想法， 客户后续的规划， 都可以做详细的回报，这样后方的支撑团队，就能够对客户进行针对性的服务，提高服务效率，让客户受益，合宙也受益。

但是，要让团队的每个销售， 都做到这一点， 能行吗?

开始，我和老陆都认为并不难。

老陆制定了详细的汇报模板， 写了详细的制度， 开了培训会议， 手把手的教同事，怎么做拜访前沟通， 聊天的重点，从客户公司离开后的邮件怎么写。

老陆给的内容， 我感觉已经详细到令人发指了。

执行起来太容易了！

于是，我就等着看报告了。

然而， 还是之前的两三个人，写的更好了，按照老陆给的指导，提升了报告的质量，很完美。

其他人，还是外甥打灯笼，照旧。

我感到怒不可遏， 都这么培训了，就差喂到嘴边了，还不行吗？

我一个个的单独私信，打电话， 询问为啥不改进报告质量呢？

被我问到的同事，无一例外，都对我说：

“老秦， 报告非常重要，太重要了，对公司很有价值！我下不为例，一定改进！”

我听了感到非常的欣慰，一颗心放到了肚子里面。

等了几天。

还是没有报告。

我又把几个销售总监， 拉过来单独开会， 单独询问为啥会这样。

总监全都答复说：“大家还没开始适应，对书面的工作，有一点不习惯，先慢慢锻炼起来，过两个月就好了。”

我想想有道理， 对他们说：“你们一定要加强辅导，认真监督，务必不能原地踏步！”

总监说：“好的，老秦，一定会改进的，看我们的行动吧！”

三个月过去了，没有改进。

半年过去了，还是没有改进。

而开始就输出很优秀的两三个同事， 书面报告，写的更优秀了，而且持续在坚持，从不间断。

我意识到，可能问题不仅仅是培训的问题， 一定是还有其他的问题。

正好，一堂的一个同学，陈真同学，给我推送了一个模型思维的视频， 我看的如痴如醉。

这个视频讲到了一个模型， 叫做“福格行为模型”， 公式是 B = MAP。
behavior = Motivation + Ability + Prompt。

翻译成中文就是：
行为 = 动机 + 能力 + 提示

这个模型，让我一下子明白了，为什么， 我跟老陆，苦口婆心的劝导， 效果不佳， 是因为方法不对。

同事输出高质量的客户报告， 是行动。

这个行动呢， 是由三个要素构成的：

第一个，是动机，
同事的动机， 是比较强烈的， 不缺的。
同事都想赚钱，都想得到客户的认可，也都想得到公司的认可，都想得到同事的认可。

这个动机， 是足够的，不是瓶颈。

在制度透明，公平的情况下，动机是够的。

第二个，是能力，

销售同事的能力， 分为两个维度： 对行业的专业度和沟通的共情能力。

这两个能力， 合宙的销售，都是比较强的。

经过持续不断的产品线的培训， 合宙销售同事们的专业度， 是相当强的。

而销售同事们的沟通的共情能力， 也都是很强，都比我好非常多。

所以，第二个因素， 能力维度，也不是瓶颈。

第三个，提示。

提示，是不断的刺激。

提示包括不断的表扬，业绩的对比，邮件的事务提醒，对过错的批评，高频次的培训，部门内和组内的讨论等等，这些都是对同事的不同的提示。

因为我对销售团队工作的不熟悉，不专业， 所以我往往只重视了动机和能力， 忽视了提示这个维度。

而恰恰是提示， 是最难做到的。

因为提示是需要高频次的，是每天，每周，甚至每个小时，都要出现，让同事们感受到的。

为什么游戏能让人上瘾?

因为只要进入到游戏里面， 就充斥了持续不断的提示， 提醒你又得到了一个宝物，提醒你又过了一关，提醒你又面对一个老怪，让你无法自拔。

为什么抖音能让人上瘾?

你只要打开抖音，各种画面都在提示你，通过观看这个短视频，就能让你提升，让你获得做饭技能，让你得知国际大事，持续不断的提示你，让你感觉很爽。

所以， 同事们的行为参差不齐， 是因为动机的强度不一样，能力的强度也不一样， 从而对提示的需求度也不同。

通过每天不断的提示， 结合已经存在的动机和专业的能力， 是有希望让同事们的行为，产生一个本质的提升的。

看到这里， 我有点豁然开朗，以前总是认为大公司做的哪些排行榜，勋章，奖状，都是虚头巴脑的，没用的东西。

原来，是我见识浅了而已。

明白了这点后， 我在公司内部群里面， 每周都发布销售业绩排行榜，邮件公布业绩表格，每天把销售的订单汇总，发给每个销售的同事。

做了这些动作之后， 同事们的客户汇报的质量，果然有了明显的提升， 60%以上的同事，都能够输出高质量的报告了。

对于增加行动的提示，你有什么经验吗？

希望你也能教教我。

8 基本功也有降维打击吗？

1 从基本功练起

第三章提到小麻的公司在疫情之间关闭了，好在并没有产生什么债务。

最近联系才知，小麻在家里休息了几个月之后，又重新开了一家公司。不做硬件产品了，**只给其他公司做定制开发，收开发费。**

小麻自己是研发的主力，同时招聘了七八个毕业生，靠着以前的一些客户积累，也接了一些项目做。

小麻说现在这个生意，赚不到什么大钱，但是因为经营的成本比较低，所以也很难亏钱，比做产品感觉踏实很多。

一天，侯总和小麻，一起来找我喝茶，我听到了下面的对话：

小麻问：“老侯，有什么办法让我的这些年轻的同事们，跟我一样的努力，一样的为公司着想呢？”

老侯听了，看了小麻好几秒钟，才回答他：“你为啥要让你的同事们跟你一样的努力？”

小麻说：“这不显而易见的吗，如果只有我一个人努力，不可能赚钱呀。越努力，不是挣钱越多的吗？”

老侯说：“问题是，同事都已经觉得自己非常努力了啊。**只是你自己认为的努力，跟同事认为的努力，是不一样的。**”

小麻：“有没有什么办法，让同事做到跟我一样努力的程度嘞？”

老侯：“我干公司差不多也有快 10 年了，我感觉吧，你想的方向可能错了。”

老侯喝了口茶，接着说：“小麻你技术很厉害，能很快的把解决方案实现出来。你的其他同事经验肯定没你多，**他们会不会有一种跟不上你的绝望的感觉呢？**”

小麻眼睛一瞪，说：“经验不足的话，足够勤奋也能弥补呀，怎么可能会绝望呢？谁还没年轻过啊？“

老侯斜了小麻一眼，吐了口烟圈，弹了下烟灰，“像你这样的人，是少数派，好不好？”

小麻不抽烟，被老侯的眼圈呛了一下，咳嗽了两下，用手扇了下空气：“那你说咋办？我都快累死了！同事还是不紧不慢的，老是让我救火！”

老侯不紧不慢的说：“你问问老秦，老秦上次不是说过，一个组织，最重要的是要练基本功吗？**我感觉你需要的不是让同事努力，而是让同事练习基本功。**”

小麻马上说：“是说过基本功，也知道基本功是挺重要的。但是吧，哪个公司能没有基本功呀？”

小麻顿了一下，说：“我怎么感觉，基本功像是一句废话呀？”

老侯笑了：“其实，打字速度，也是一个很重要的基本功。”

小麻：“老侯你开玩笑的吧，打字速度跟我们有啥关系？不是文员的基本功吗？”

老侯笑的嘴老大：“现在是什么年代了，你打字速度不快的话，你跟客户交流都跟不上趟！打字慢的话，跟一个客户交流都费劲。**打字速度快的话，你可以跟 10 个客户同时聊！**”

小麻：“这不就是客服吗？”

老侯“切”了一声，“啥客服，**老板跟客户也要聊好不好，销售也要聊，技术也要聊，产品经理更要聊！**这是啥年代了，你跟客户不聊，你吃亏吧，吃亏死你！”

小麻：“好像是这么回事。。。。。，我最近跟客户聊的好像也挺多”

小麻停了会，又说：“最近我在试着用逐字稿，这玩意也很费键盘，也需要打字速度快。”

老侯说：“就是嘛！你现在跟客户聊天，你发语音包，人家都很烦的，谁有空听你的语音！你发一段文字给人家，客户才轻松嘛！”

小麻说：“怪不得，我发微信语音，好多人不回复。”

2 **基本功的四个类型**

我听了半天，实在是有点忍不住想炫技了，我正好听过基本功的完整课程，是一堂的 CEOTruman 讲了 10 小时的课程。

我给他俩倒上了茶，“打字确实是基本功的基本功，基本功是有挺多种的，但是咱们也不用搞太多基本功。”

小麻说：“那你看我们公司适合啥基本功？”

我说：“你公司不到 10 人，**成立时间不长，比较适合练习找机会的基本功。**”

老侯又喷了一口烟：“找机会是啥基本功？”

小麻也瞪大眼睛看着我。

我说：“**找机会，是创造型的基本功。**”

小麻说：“创造型的基本功，听上去有点牛逼啊，还有啥其他型的基本功？”

我略微有点得意的说：“**创造性的基本功，就是最难的，解决开放性问题的。**比如你们公司选择做啥，还是不做啥，就是开放性的，掌握这个技能，就是掌握了创造型的基本功。”

小麻说：“做啥不做啥，不都是我的事情吗？我同事也决定不了吧？”

我说：“假如你和你的两个同事，你们三个人，都能很清晰的知道你们做啥不做啥，你们公司是不是能上个台阶？”

小麻立刻说：“对啊！我最开始说的就是这个意思！我就是想让同事跟我一起做大事啊！我自己感觉有点力不从心了啊！”

老侯悠悠的说：“我大概明白了，**做啥不做啥，是开放性的，很难；你怎么定义产品，也很难，也是开放性的；怎么去找新客户，也很难。**这些都是所谓的创造型基本功吧？”

我说：“对对对！还是老侯厉害！”

小麻说：“那除了创造型基本功，还有啥基本功？”

我想了下，说：“还有……**专业型基本功、领先型基本功、标准型基本功**。”

小麻：“听上去好复杂……我有点晕啊”

我：“猛一看上去确实有点晕，但是你已经知道了创造型基本功，其他三个都是更好理解的了。”

小麻：“专业型基本功是什么？”

我：“**专业型基本功，不是开放性的工作，是封闭性的工作，是难度比较大的封闭性工作，是专家类型的工作。**”

小麻说：“这么说，我也做了专业型的基本功了呀。”

我：“对的，研发工作，几乎都是专业型的基本功。”

老侯说：“创造型的基本功，怎么练呢？基本功怎么算是练好了，怎么算是没练好？”

小麻：“这能有标准吗？就练呗。”

我说：“创造型基本功，衡量练好没练好，确实没有标准，但是有练习的方向。”

老侯把往烟灰缸倒了点茶水，把烟头扣在一个烟灰缸里面，又拧了一下，慢慢的说：“练习的方向？这玩意能有啥方向？”

我说：“**创造型基本功练习的方向，就是通过持续的练习，让创造型基本功，成为你公司的专业型基本功。**”

小麻一拍大腿：“靠，太牛了！我怎么没想到呢！”

我被小麻吓了一跳：“你想不到很正常啊，这是人家研究了好几年的成果，我们拿来用就好了。”

小麻又问：“那专业型的基本功，练习的方向是啥？”

老侯又点了一根华子，深吸一口，吐出来，眯着眼说：“专业型的基本功，练习的方向就是变成傻瓜型基本功哈哈哈哈。”

我被老侯的烟呛得不行，强忍着说：“别瞎说，**专业型的基本功，要练成领先型的基本功。**”

老侯说：“啥叫领先型的基本功？”

我端起一杯茶，一饮而尽：”**领先型的基本功，就是注重效率的基本功。**电商运营、产品文档、销售、逐字稿，这些都是领先型的基本功。“

小麻说：“领先型的基本功，看样子都是有套路的，通过练习可以掌握套路的啊！”

我说：“小麻你太聪明了！领先型的基本功，通过训练之后，可以达成标准型的基本功的效果。”

老侯听得忘记了抽烟，被烟烧到了手指头，赶紧把烟放到烟灰缸里面，说：“标准型的基本功又是啥？”

我说：“**标准型的基本功，就是需要强调执行力，动作标准的。**测试类、事务类、跟进类，都是标准型基本功。”

小麻问："标准型的基本功，训练的目标是不是 SOP 化？"

我惊讶了："小麻你简直是天才，你怎么知道是 SOP 化？太厉害了！"

小麻哈哈笑了："SOP 就是标准操作流程呀，有了 SOP，效率自然就高了呀，也不容易错了呀！"

老侯也不点烟了，拿起手机开始记，一边记一边说：

创造型基本功专业化，

专业型基本功领先化，

领先型基本功标准化，

标准型基本功 SOP 化!

小麻说："这要是都练成了，不就是上个台阶，形成降维打击了吗？"

老侯说："说着挺容易，做起来太难了吧！"

我叹了口气，说："是啊，太难了！每一个岗位，想持续练习都要花太多精力，何况上个台阶！"

老侯说："无论怎样，还是需要练的，练了比不练的好。"

老侯跟小麻告辞走了，我在手机备忘录写了今天的心得：

越小的公司，越要进行创造型基本功的训练，把创造型基本功往专业方向练；

长大一点，就要创造型和专业型基本功一起练，专业型基本功练成领先型；

大型公司，就必须是四类基本功都要练，都上升一个台阶。

练基本功的目的，就是提升基本功的层次，形成降维打击！

今天的故事讲完了，欢迎给我留言，说说你对基本功的看法。

9 建立客户信任还有公式？

1 信任公式的四个维度

刚刚讨论完基本功，没想到第二天，小麻又上门了。

不仅来了，还把老侯一起拉过来了。

刚坐下，小麻就说："两位大哥，咱们来讨论下：**我们要跟客户建立信任，建立长期的信任，以后能紧密合作。**这种基本功，是什么类型的基本功？"

老侯点了根烟，抽了一口，从鼻子里悠悠呼出来，说道："是专业类的基本功吧，因为这个建立的目标很明确，不算是开放性的工作。又不需要拼效率，有点像是销售专家性质的。**所以，专业类的基本功比较合适。**"

我点点头："我认为老侯说的非常对！"

小麻站起来说："既然是专业类基本功，怎么能练到领先类基本功？建立客户信任这个事儿，得看人的吧？厉害的人就是能建立信任，我好像就不行。"

老侯把烟灰摁到桌上："**能拆解，就能练；不能拆解，就不容易练。**"

小麻："老侯你是说的刻意练习吗？"

老侯点点头说："刻意练习确实是需要拆解的，但是我不确定刻意练习能不能帮助建立客户信任，我好像也不知道怎么拆解。"

我说："我记得有个信任公式，应该是叫做麦肯锡信任公式。"

小麻重新坐下："说说看看？啥公式？"

我说："信任=专业度×可靠度×亲密度×利他度"

老侯说：“你这个公式，感觉还有点复杂啊！”

小麻又站起来：“让我先写到白板上！”

老侯看着白板说：“专业度，这个好理解，就是销售员的行业专业度。”

小麻用手指点着白板说：“可靠度，应该是指销售员传递给客户的个人形象和公司形象吧，让对方绝对踏实靠谱，可信赖，其实就是品牌形象吧？”

我开始泡铁观音，端给他俩一人一杯：“我也是这么理解的，**可靠度，就是销售员传递给对方的品牌形象，个人的形象也代表了品牌形象。**”

老侯说：“亲密度，这个也好理解，就是销售员跟对方的关键人，是否能形成一种个人的人际关系。但是人际关系，还不能完全代表信任度啊，这个真有意思！”

小麻坐回到座位上，喝了一杯茶说：“这个确实是颠覆我的认知了，**我以前一直以为，客户信任就是做关系，但是这个公式显然更科学啊！**”

我说：“人际关系，能促进信任度，但是又不是信任的全部。而促进人际关系，也可以继续拆解。。。。。。有一位济南的周大哥给我讲过怎么拆解，特别精辟。”

老侯又抽了一口，说：“我们先不扯远了，先看看这个公式。”

小麻问：“利他度，这个有点费解了啊，是做好事吗？”

老侯说：“利他度应该算是长期的行为吧？长期让别人觉得，他是能够分享利益的人，能够双赢的一家公司？”

我哈哈笑了下，说：“利他度，好像也跟短期也有关系。我认识的一个牛人，跟一个刚刚接触的客户，就能做到雪中送炭。**下雨天，主动给对方送一把雨伞，送完就走，让对方非常感动。**这就是很典型的利他标签。”

小麻一拍桌子：“哇！这太棒了！我好像就从来没有做过这样的利他的事情！我得认真学习了！”

老侯很不好意思的说：“我去送伞，我都送不出去，我很难为情。”

我想了想：“我应该也送不出去。”

小麻有点激动了：“我来送！我肯定行！”

老侯：“年轻就是好，我这还是太喜欢面子了。“

2 如何提升客户信任

我看向白板：“我们再看看这个信任公式，**4 个变量：专业度、可靠度、亲密度、利他度，如何提升也很关键。**”

老侯：“专业度经过培训专业尝试，培训话术，应该是能够提升的。”

小麻说：“可靠度，挺难训练的，怎么看上去可靠？怎么看上去不可靠？还是主要依赖于人吧？”

老侯：“可靠度的打造，可能人不是最关键的一环，最关键的还是公司在前几年的形象占第一位；其他的比如销售员能力，有帮助，**可能不如前几年积累的形象来的真实。**”

小麻：“这么看，可靠度，其实是个偏长期的变量，靠着前期的积累，才能发挥出来优势。”

我：“对的对的，可靠度，是靠长期积累的，短期确实不是太可能打造很好的可靠度。”

小麻又走向白板：“**长期积累的可靠度，是品牌资产。**现在就开始做品牌运营，看样子是很有必要的，可是怎么做品牌呢？”

老侯：“小麻你别急，我们先把这个公式说完呗”

我：“对对对对，我们继续。亲密度，就是做关系了。做关系可以这么来拆解。。。。。”

老侯：“老秦打住打住，我们先把这个公式说完，我还是有点懵逼啊~“

小麻看着老侯，疑惑的说：“这个公式不难理解啊！**就是保证 4 个维度的变量，至少不能有特别低的分数，就是中上等的水平了。**假如再有其中一个变量，是全球前几名，甚至第一名，这信任度，不就牛逼大发了？”

老侯：“你看看这 4 个变量，也就只有专业度是可以被你培训的。多培训几次后，团队的专业度肯定是会提升的。”

小麻说：“没错，回去就做培训计划！”

我说：“专业度相对难度小一点，但是可靠度，就不容易了！”

小麻说:“为啥可靠度不一样？不是一样的做法吗？"

老侯说：“**可靠度，不但要长期形成，至少一年。这一年内，你得每天保持相同的行为，看看是不是撑得住。**”

我说：“亲密度，刚才说了，是客户关系，可以拆解。”

小麻说：“先不急着拆解亲密度，先看看这个利他度，这个训练的话，有具体的方法吗？”

我跟老侯你看看我，我看看你，感觉拿不出方案来的话，有点没面子。

老侯说：“利他度应该是很难拆解的。”

我说：“利他度肯定是能拆解的，不过可能拆出来去分解执行，也肯定没有一个人能搞定亲密度的成本高。”

小麻：“以前从来没有听到过这个公式。今天学到了！我回去就让同事们继续拆解！”

老侯重新点上一根烟，说道：“今天学到的这个信任公式，不管这个公式是否能执行好，**客户信任最起码有方向了！**”

小麻接着说：“通过这个公式，拆开 4 个变量，把 4 个变量都分别提升，有希望达到领先类基本功吗？”

我说：“这四个变量再拆分，再分别练，好像就是达到领先型基本功的路径，这个路径应该是非常靠谱的！”

创造型基本功专业化，

专业型基本功领先化，

领先型基本功标准化，

标准型基本功 SOP 化！

老侯说：“把信任关系，从专业类基本功，做到的领先类基本功，也是没谁了！“

小麻说：“今天太值了！没想到客户信任也是基本功，而且是可以专业基本功领先化！受教了！”

说完，小麻就喜滋滋的走了。

送走了小麻后，我问老侯：“小麻能练成客户信任基本功吗？”

老侯说：“我看难，练好亲密度这一项就很难！反正我是没戏了！”

我说：“不管咋说，**信任=专业度×可靠度×亲密度×利他度，是公司里面很值得实操推广的模型。**下周我就在合宙开始推广。”

对建立信任很想做到，就是摸不到门。东一锤西一锤的，效果不稳定。

麦肯锡信任模型，可以解决这个问题。

10 亲密度关系的分类和拆解

亲密度关系的分类和拆解

这天下午，我正在思考，**接下来到底是冲量保市场份额，还是先保利润再要份额？**想了半个多小时，也没个头绪，就想先泡一壶老枞水仙，希望自己纷乱的脑袋，能被水仙的香气给清洗一下。

刚坐好，就有一位一堂同学来访，赶紧招呼坐上贵宾席，准备泡茶。

刚泡好第一杯，还没喝上，小麻就来了。

我诧异的说：“前天昨天，你不是都来了，怎么今天又来了？”

小麻说：“我昨晚没睡好啊，一直想着你说的信任公式。”

我给小麻也倒了杯水仙：“昨天不是都说清楚了吗，你还有啥烦恼的？”

小麻也不喝茶，快速说道：“昨天你说的信任模型，我感觉好有道理啊！就是有个**亲密度**，就是做关系，你不是说能拆解的吗？**但是没说怎么拆啊？**”

我慢慢的喝了一口，第一泡的水仙，香气沁人，回味悠长，让我都不舍得说话了。

小麻见我不吭气，又问：“我回忆了好久，昨天好像你准备说来着，但是被老侯给打断了。**你现在倒是给我说说啊！**”

我又倒了一杯，然后对着小麻抬了抬下巴："先喝茶，凉了就不好喝了。"

小麻端起茶杯一饮而尽，继续看着我，等着我说话。

我又喝了一口，第二泡的香气，不再是直入鼻腔，而是回味到了喉咙深处。

我咽下去之后，对小麻说："昨天被老侯打断之后，我也忘记了。你这么一说，我才想起来。这个拆解，也是说来话长的。"

小麻起身，给我倒了一杯茶："你可以先简单说下，然后再详细展开说。"

我说："这个亲密度的拆解，**是一堂的同学，济南的周广智大哥，亲自传授给我的。**"

小麻嘴巴张的老大，说："这位广智大哥，也太牛了吧，跟人的关系，都能拆解？"

我喝完杯子里的茶，接着说："这位广智大哥，做大 B 的生意好多年，提炼了非常精辟的方法论。**客户关系的拆解，仅仅是他的总结里面的很小的一环。**"

小麻说："有机会见到这位大哥真人就好了！"

我忍不住笑了，说："**这位大哥那是远在天边，近在眼前啊！**你旁边喝茶这位，就是周广智大哥！"

广智对小麻伸出手来，说道："幸会幸会！"

广智大哥是山东济南人氏，年龄跟我差不多，但是外貌看着要比我更加稳重踏实，一看就是政府机关待过很多年的那种气质。

小麻面对这么稳如泰山的大哥，稍微有一点点害羞，连忙伸出手说："广智大哥，幸会幸会！"

我说："广智大哥啊，你以前跟我讲过，人际关系，**尤其是在面向大客户的时候，更要有逻辑清晰的人际关系，**今天能不能详细的跟我讲一下呀？"

广智笑的非常憨厚，略显黝黑的国字脸，显得非常轻松，笑起来有一口异常洁白的牙齿："有秦鹏大哥邀请，我很高兴能够分享哈哈。"

广智喝了口茶，接着说："要说人际关系啊，我把人际关系分为**三种：第一种是裙带关系，第二种是利益关系，第三种是价值共振的关系。**"

小麻一边给广智和我倒茶，一边说："哇，听上去就感觉好牛啊，第一次听到人际关系能有这样的分类！"

我等着小麻倒好茶，喝了一杯说："大 B 的业务，因为客户的决策周期比较长，决策的环节比较多，所以更要把客户的团队的人的因素分析清楚，才能更好的帮到客户。"

小麻说："**我们面对的是中小 B，是不是要求会低一点？**"

我说："你学会了大 B 的客户关系的方法，再去做中小客户的关系，岂不是降维打击吗？"

小麻一听笑了，连连点头，又忙着给广智倒了一杯茶："广智大哥，您继续哈"。

广智接着说：“裙带关系，就是你跟客户的关键人，是亲戚关系，表弟啊表舅啊，都算，或者是发小啊，老战友关系，或者是老同学关系。“

我说：“裙带关系，好像是可遇不可求的吧？”

广智说：“对的对的，这些关系啊，跟血缘亲属，个人经历有关，**无法复制或者很难复制，也无法经营。**所以这一类关系啊，也不用继续拆解了，有这层关系就可以继续用，没有的话，也不用强求。”

小麻说：“是的，越是草根出身，越是缺少裙带关系。”

广智说：“缺少是的，不过也没关系，咱们还有第二种和第三种。”

小麻问：“第二种是啥来着？刚才我听到了，现在又忘记了。”

广智说：“第二种是利益关系。啥意思呢？**所谓的利益关系，就是你跟对方是利益的绑定关系，也就是金钱利益的关系。**”

我说：“好像很多销售的大佬，很会经营这类关系。”

广智说：“是的，利益绑定关系非常常见，但是我不倡导跟客户形成这类关系。”

小麻说：“为啥？我之前见过一个销售大佬，跟我说过，跟客户形成利益绑定是一种能力啊！”

广智说：“跟客户的关键人形成利益绑定的话，很可能只是形成很短期的关系，对长期的合作很不利。并且，你跟其中的某个人形成了利益绑定之后，跟其他人就很难形成真正的紧密关系，最终你可能会受到反噬。最后，**最关键的是，利益绑定是不可逆的，这是大忌。**”

小麻说：“你这个观点有点震撼到我了，颠覆了我对于客户关系的认知啊！”

我笑了：“这算啥，后面还有更震撼的！”

广智说：“总之，这个第二种关系利益绑定的关系，我们公司是严格限制的，即使在公司发展初期，为了生存免不了这种关系，也是压缩到最低限度的。这种关系，对企业的发展实在是一种很大的隐患。”

我说：“是的，我也深有体会！咱们还是要重点发展第三种关系对吧？”

广智说：“对的，**第三种关系，就是价值共振关系，也就是跟客户形成心灵碰撞，跟客户在一起能够共同排遣孤独的这种关系。**”

小麻说：“听上去这很高级啊，但是，是不是特别难呀？”

广智说：“其实没那么难的。你跟老秦之间，不就是这种关系吗？”

小麻疑惑的看向我说：“是吗，老秦？我们是这种关系？”

我被小麻看的有点脸红了：“肯定是啊，我们这种沟通，不就是在排遣孤独、抱团取暖吗？”

小麻松了口气说：“哦，原来如此啊！那就是谈得来的关系呗！”

广智说：“是的是的，**谈得来就是个开始，而你把谈得来，经营为一种长期的谈得来的关系，就是价值共振的关系了！**”

小麻说：“谈得来，好像跟好多人都行。但是让销售员，跟客户的关键人，长期谈得来，是不是难度又太大了？”

广智说：“这种谈得来的方式，是可以有方法的，可以拆解的。”

我站起来说：“广智拆解这个价值共振的长期的关系，可以拆的非常详细，都六点多钟了，我们先去吃饭吧，边吃边拆咋样？”

11 经营良性可持续的长期关系

到了饭店，广智大哥、小麻和我三人落座。

还没点菜，小麻就迫不及待的问：“广智大哥，**价值共振关系，怎么长期经营啊？**我怎么感觉有点不太可能啊？”

广智大哥笑了，他的洁白的牙齿，闪着晶莹的亮光：“不仅仅是有可能，而且是经过我的实践，是可以达成的。”

我说：“广智大哥，无论是央企一号位，还是小企业的经理，是不是都适用的？”

广智说：“对的，经过我自己的亲身验证，都是适用的。”

广智喝了口水说：“这里我插一句啊，刚才在老秦办公室说是有三种关系，不是特别完整。**其实还有第四种关系的，叫做酒肉朋友关系。**”

小麻说：“哦对，这种关系就是经常喝酒的那种朋友对吧？”

广智说：“对的，就是经常一起喝酒泡吧夜总会，经常一起玩的关系。离开了这些场子，就没什么话题了。”

我说：“这类关系，是短期还是长期的？”

广智说：“这类酒肉朋友的关系，**绝大多数，都是短期的关系，一般是跟生意伴随的关系。**生意不做了，酒肉的场子就没有了，关系也就断了。”

小麻说：“这类关系也不是良性的对吧？”

广智说：“这类关系，不是长期的，只是一种消遣的形式，因为它**不太符合人的求知上进的本性**，所以往往会有一种比较浅的愧疚感，人跟人之间通过酒肉建立起来的关系，将来往往会倾向于不再来往。”

我说：“那这样岂不是很不划算啊，辛辛苦苦建立起来的关系，后面就全都丢了。”

广智说：“是的是的，所以酒肉的关系，也是要尽量避免的，所以我都把它给忘记了，现在觉得还是要提一下，免得遗漏了。”

小麻说：“知道了大哥，那您继续说，怎么经营价值共振的关系呗。”

广智说：“价值共振的关系的特点，也是最重要的一个特点，就是长期性。**价值共振的关系是长期的，并且是可以长期经营的。**”

我说：“好像确实是有那么一些朋友，做过生意，现在很多年不做了，但是还是会长期的沟通，碰撞思想，感觉还是不错的。”

广智说：“是的，这种价值共振的关系，**不仅仅对于个人来说是长期良性的，对于发展业务也是良性的，**不会因为关系的亲密，就会做有损双方的企业的行为，所以它是可以长期保持和经营的。”

小麻说：“广智哥你说的这点，我也有体会了。我跟有些客户的沟通，就感觉很舒适，有些客户沟通就感觉有点别扭，但是又必须捏着鼻子沟通，可能就是我还没有经营长期价值共振的意识。”

广智说：“价值共振的关系的经营，关键在于，**每次跟客户的沟通，都要遵循一个沟通三段论。**”

小麻说：“有一个三段论，我这回听仔细点哈哈。”

广智说：“我把这个三段论总结为：**第一、现实满足；第二、价值同步；第三、沟通推进。**每一次的跟对方沟通，都要遵循这个三段论，这样你的长期的良性的关系，就建立起来了！”

小麻说：“完全不懂啊大哥，能不能解释一下呀？”广智说：“别急别急。我一个一个的跟你讲。”

这时候，服务员开始上菜了。

大家都有一点饿了，一边吃一边说。

广智说：“我先说三段论的第一段，**什么叫做现实满足。**”

广智吃了一口土豆丝，继续说：“你在跟对方沟通的过程中，总是会有一个话题，让双方聊得都非常愉快，达到会心一笑的地步。“

小麻若有所思，给广智夹了一块口水鸡，没有说话。

广智接着说：“比如说，你刚到客户办公室，**看到墙上挂着客户的创始团队的合影，你就跟对方聊创业起步阶段的打江山的过程。**对方大概率会滔滔不绝的跟你讲，他们当时是多么的风光，多么的艰难的拿下一个山头。而你就除了赞同他的执行力之外，还要说下

你对这个行业当前形势的看法，尽管生意不那么好做了，但是凭借你的团队能力，怎么能够进行下个阶段的升级。”

小麻说：“好像经常跟客户一见面，都会扯闲篇扯个 20 分钟。。。。。”

广智说：“你说的很对，**扯闲篇，就是在做现实满足。**甚至你们聊到孩子教育，对方也是很感兴趣，意犹未尽的。有些是聊到技术细节，也是很来劲的，有些是聊到对宏观形势的看法，反正都行，都是现实满足。这个难度不大的，而且是经常发生的。”

小麻说：“是的是的，听上去确实难度不大，我有时候对这种扯闲篇比较焦虑，觉得是不务正业，耽误时间，原来这是必要的沟通步骤呀！这下子懂了！”

我说：“现实满足，就是跟客户拉近距离，也可以说是比破冰更加进一步的沟通形式。”

广智说：“是的是的。现实满足就是这个作用。**那么，价值同步是什么呢？价值同步，是更进一步的心灵沟通。**”

广智喝了口鸡汤，继续说：“在你们继续聊的过程中，双方共同发现和建立了一个对生命，或者人生，或者经营的抽象的看法，更产生了相见恨晚的感觉。这种感觉，让双方的沟通产生了升华，这就叫做价值同步。“

小麻说：“我的天，我好像除了跟老秦之外，其他人还没有过呀！”

广智说：“其实也不难。比如说，你去见对方董事长，跟董事长说明来意，不是来推销的，而是来请教问题的。你说你的发小，经营企业六七年了，一直很累，对经营企业产生了深深的怀疑，以及个人的自我怀疑，想请教下董事长，面对这个问题，我该怎么开导我的发小？

事实上，这个问题，是你经过调查，其实就是这个董事长经历过的。

于是，董事长就开始滔滔不绝的跟你讲一个多小时，他是怎么创业，怎么经历至暗时刻，怎么走出低谷，怎么建立当前的局面的。

讲完之后，意犹未尽，他会再加一句：小麻，咱们边吃边聊吧，你的那个方案啊，回头我再跟王工说说，怎么改进一下，赶紧合作起来！”

小麻有点惊呆了，说：”我以前就没想过还能这样的！不过好像确实可以啊！“

广智说：“是的，这就是需要我们做有心人，**真正的关心对方，给对方良性的情绪，**也让我们自己在聊的过程中学到不少东西。”

小麻说：“价值同步现在也知道了，下一段是沟通推进，该怎么说？”

广智说：“在你们将要沟通完毕，还有几分钟要离开的时候，**你要针对聊的话题，预留一个伏笔，不要一次说完了，为下次沟通留下一个话题的引子。**

比如说，你们聊到孩子的教育，对方说孩子不爱学习，就爱打游戏。你也说，我孩子也是这样，学习完全不上心，游戏一打一个通宵，但是最近 3 个月，游戏几乎不打了，成绩也提高挺多。等到下次有机会，我好好跟你分享一下哈。”

小麻说：“真妙啊！我怎么没想到这点！”

广智说：“这样你在跟他分开后，可以通过微信发给对方一些你的思考内容，你的经验，你的阅读材料，对方就能感觉到你对他的关心了。而你下次再约他的时候，可以约他就这个话题，继续深入讨论，对方一定会百忙之中会跟你见面的。”

我说：“广智大哥传授的的太妙了！现实满足、价值同步、沟通推进，这个三段论，听大哥讲完，好像也没那么难呀！”

广智说：“是不难，就是要多练。熟能生巧，多练，就会变得容易了。”

小麻问：“广智大哥，**跟客户的关系，保持的好还是不好，有什么标准吗？**”

广智说：“你这个问题问的真好，还真有标准。**维持关系的目标，是不让客户合法的伤害你。**”

我之前听广智跟我讲过合法伤害的话题，觉得特别精辟，赶紧竖起耳朵，仔细听第二遍。

小麻说：“合法的伤害怎么理解？”

广智说：“你跟客户喝酒的时候，客户就只灌你酒，还联合其他人一起灌你，但是对其他人就会网开一面。**这就是合法伤害你的信号，**你要警惕了，很可能是你关系没有做到位呀。”

小麻说：“哦，这样啊，我很少喝酒，好像感受不太深刻，还有其他的标准吗？”

广智说：“你要交付客户产品吧？不管是硬件还是软件还是文档，你交付的时候，客户验收都有制度和规则，但是也都有弹性。**客户是利用弹性善待你，还是利用弹性刁难你？**如果是利用制度刁难你，就是在合法的伤害你。”

小麻说：“这听起来有点可怕呀！”

广智说：“这是很常见的！假如你要给客户送一批货，客户的仓管员，以一个非常合理的理由，让你的货进不了仓库，导致你入库延迟，从而影响验收，以及后续的结算。仓管员的做法完全合法，他只是没有通过更加帮你忙的手段让你入库而已。而你的这次交付，就在客户处留下了一个不可逆的污点。这就是一次合法的伤害。”

我说：“小麻你交付客户的电路板，客户那边验收测试的时候，**可以 2 天测试完，也可以 60 天测试完。**2 天测试完，就是你关系到位了，人家帮你；60 天才测试完，就是你关系不到位，人家合法的伤害你。”

小麻：“明白了！怪不得有些客户总是对我们的配合，爱理不理的，原来是我们关系没到位呀！这下彻底懂了！”

我：“感谢广智大哥的传授，我再听一遍，更是真是受益匪浅！”

广智说：“我能给兄弟们讲一遍，也很享受，下次我们继续交流哈哈！”