SSW118GE
SSTAR_IOT_SUPPLICANT 使用说明

Application Note Version 2.0

REVISION HISTORY

Revision No.	Description	Date
1.0	• 初始版本	09/24/2020
1.1	• 更新 DEMO 调试、TCP 心跳包测试方法，新增获取 RSSI 指令	12/10/2020
1.2	• 添加获取当前网络时间接口（SDIO）	12/31/2020
1.3	• 新增客户支持配置说明	01/11/2021
1.4	• 新增清空 WIFI 配置项及获取 ETF 收包期间统计数据功能	02/04/2021
1.5	• 新增 netpattern、wk_ssid 等新指令	04/20/2021
1.6	• 新增客户证书存储、sdio 保活指令	10/15/2021
1.7	• 新增大通道传输反馈功能，用于双向传输大数据，补充新旧卸载方式说明	01/11/2022
1.8	• 新增 debug 调试打印控制、强制 118 重启及快速配网设置功能	03/11/2022
1.9	• 修改部分适配 118 指令说明	11/28/2024
2.0	• 修正 118 etf 配置指令说明	12/19/2024

TABLE OF CONTENTS

1. 说明

本文档针对于 SSTAR_IOT_SUPPLICANT 工具的使用进行了详细的说明。
文档旨在介绍工具使用过程中的指令配置及使用方法，后续章节将详细介绍具体内容。
文档所属指令功能均已进行了测试，并且粘贴了相应的测试结果。
另外，SSTAR_IOT_SUPPLICANT 提供了 DEMO 版本，用于模拟实际的网络连接与断开、驱动加载与卸载情况下的视频数据发送流程。

2. 工具启动

本工具启动依赖于驱动模块，使用时请将 Sstar_iot_supplicant 与 ssw118ge_wifi_sdio.ko 置于同一目录。
本工具包含了两个组成部分：守护后台（Sstar_iot_supplicant）和指令终端（Sstar_iot_cli），在设计理念上与操作方法上都与 wpa_supplicant 保持了极大的互通性。因此，对于 wpa_supplicant 有一定了解的人员使用起来会更加地得心应手。另外，在流程上工具配备了 demo 版本 Sstar_iot_supplicant_demo，用于调试数据控制功能，在第 5 部分有详细的使用介绍。
启动方法可参考 wpa_supplicant 的使用，后续的启动打印也会将整个流程详细地显示出来。
另外，工具启动过程采用了 GPIO 触发方式，由 GPIO 属性值触发驱动的加载与卸载流程，相关流程明细见下方打印，使用指令终端工具即可发送指令：

```
[root@sinlinx /]#./Sstar_iot_supplicant &          ->启动守护后台
942
*** Welcome to use Sstar_iot_supplicant ***

VER: 2024-1114-1500.          –>工具版本号

[    276.996487]
[    276.996493] DRIVER VER: 2024-1125-1120.          –>驱动版本号
[    277.003520] [Sstar_log]:Sstar_init_firmware
[    277.007989] [Sstar_log]:----drvier support chip RHEA
[    277.013534] [Sstar_wifi]: Sstar_wifi module power set by axp.
[    277.019816] create_regulator: axp22_ldo2: Failed to create debugfs directory
[    277.027640] [Sstar_wifi]: regulator on.
[    277.032522] [Sstar_wifi]: succeed to set gpio Sstar_wifi_shdn to 1 !
[    277.039445] [Sstar_wifi]: sdio wifi power state: on
[    277.045197] sdc1 set ios: clk 0Hz bm PP pm UP vdd 3.3V width 1 timing LEGACY(SDR12) dt B
[    277.069070] sdc1 set ios: clk 400000Hz bm PP pm ON vdd 3.3V width 1 timing LEGACY(SDR12) dt B
[    277.149712] sdc1 set ios: clk 400000Hz bm PP pm ON vdd 3.3V width 1 timing LEGACY(SDR12) dt B
[    277.161648] sdc1 set ios: clk 400000Hz bm PP pm ON vdd 3.3V width 1 timing LEGACY(SDR12) dt B
[    277.172504] ********************Try sdio********************
[    277.178929] sdc1 set ios: clk 400000Hz bm PP pm ON vdd 3.3V width 1 timing LEGACY(SDR12) dt B
[    277.189791] SDIO_CCCR_CCCR>>>>>>>>>>>>>>>>>>>>>>>>32
[    277.195533] SDIO_CCCR_CAPS>>>>>>>>>>>>>>>>>>>>>>>=1f
[    277.208862] sdc1 set ios: clk 400000Hz bm PP pm ON vdd 3.3V width 1 timing SD-HS(SDR25) dt B
[    277.218293] sdc1 set ios: clk 50000000Hz bm PP pm ON vdd 3.3V width 1 timing SD-HS(SDR25) dt B
[    277.249610] [Sstar_log]:Sstar: mmc2 is not found.
[    277.283021] sdc1 set ios: clk 50000000Hz bm PP pm ON vdd 3.3V width 4 timing SD-HS(SDR25) dt B
[    277.293578] mmc1: new high speed SDIO card at address 0001
[    277.300714] [Sstar_log]:Probe called      ->设备 match probe 成功挂载
[    277.304677] [Sstar_log]:Sstar_sdio_probe:v12
```

```
[    277.310546] [Sstar_log]:Sstar_after_load_firmware++
[    277.310560] [Sstar_log]:Sstar_sdio_rx_thread
[    277.330798] [Sstar_log]:set_block_size=256
[    277.335365] [Sstar_log]:mdelay wait wsm_startup_done   !!
[    277.339027] [Sstar_log]:rx timeout
[    277.345186] [Sstar_log]:firmwareCap c03d
[    277.349461] [Sstar_log]:firmwareCap2 0
[    277.353524] CAPABLITIES_SDIO_TX_LIST_CHECK     4
[    277.358355] CAPABLITIES_TX_CONFIRM_NOTIFY      8
[    277.363108] CAPABLITIES_HOST_CONFIG            16
[    277.367459] CAPABLITIES_HOST_BA_PARAM                  32
[    277.372018] [Sstar_log]:wsm_caps.firmwareCap c03d
[    277.376958] [Sstar_log]:apollo wifi WSM init done.
[    277.376963]       Input buffers: 24 x 1632 bytes
[    277.376967]       Hardware: 0.25697
[    277.376971]       WSM firmware [=IOT=RF=Ares_AX   2GHZ Nov 28 2024 09:40:49], ver: 22570, build: 0,
api: 22570, cap: 0xC03D Config[30004]   expection 9007094, tx prog 24   expection 9007094, tx prog 24
->成功获取 startup 枚举信息
[    277.429049] [Sstar_log]:Sstar_sdio_tx_thread
[    277.433614] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(23),used(1)
[    277.440692] sync hera ba parames ok.
[    277.440764] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(22),used(1)
[    277.451030] [Sstar_log]:Sstar_get_mac_address.a8:29:fe:d4:c5
[    277.462893] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(21),used(1)
[    277.469056] Sstar_ioctl_open cost time: 110 ms
[    277.474305] [Sstar_log]:wifi mode:STA
[    277.478292] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(20),used(1)
[    277.484512] [Sstar_log]:[Sstar_wtd]:set wtd_probe = 1                    ->驱动加载完成
[WIFI_MODE] STA
CTRL-EVENT-STATE-CONNECTED - Connection to e2:88:f2:b6:8a:84 completed   ->配置网络信息
[    278.514954] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(19),used(1)
[    278.521462] [Sstar_log]:Sstar_sta_connect_event: associated
[    278.527496] update ba params idx=0.
[    278.531409] ba_tid_params:action1,linkid1,tid0,ssn0.
[    278.536924] [Sstar_log]:Sstar_get_mac_address.a8:29:fe:d4:c5
[    278.543048] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(17),used(1)
[    278.549701] [Sstar_log]:tcp_port_filter addcnt.1:
[    278.554846] [Sstar_log]:Sstar_open
[    278.590258] [Sstar_wifi]: Sstar_wifi module power set by axp.
[    278.596505] create_regulator: axp22_ldo2: Failed to create debugfs directory
[    278.604358] [Sstar_wifi]: regulator on.
[    278.609164] [Sstar_wifi]: succeed to set gpio Sstar_wifi_shdn to 1 !
[    278.616024] [Sstar_wifi]: sdio wifi power state: on
[WAKEUP] IO wakeup
```

```
[  288.929023] wlan0: no IPv6 routers present
```

上述为一个完整的正常驱动加载流程。

3. 工具休眠、唤醒控制与卸载说明

本工具针对于固件端与 Linux 端的休眠控制采用了 Linux 端为主控制层的方法，即 Linux 端享有休眠操作的决定权。

主要处理逻辑分为如下几类：

- 固件端接收到休眠指令（按键/发包控制）：

1) Linux 端可忽略此项通知继续相关数据传输，待操作完毕后发送休眠指令触发固件端与 Linux 端休眠；
2) Linux 端可忽略此项通知继续相关数据传输，待服务器断连后自动触发固件端与 Linux 端休眠。

- 固件端未接收到休眠指令（按键/发包控制）：

1) Linux 端无法触发休眠指令。

关于服务器部分在后续的第 5 部分将会加以介绍，另外服务器断连触发休眠的方法仅适用于 demo 版本。
在 demo 版本中所包含的发包控制休眠唤醒的方法同样适用于此部分描述。

额外需要介绍的是 Linux 端的两种状态：可休眠状态和唤醒状态。
可休眠状态：Linux 端接收到固件端的休眠通知后的状态。
命令行提示：linux driver can rmmod
唤醒状态：Linux 端接收到固件端的唤醒通知后或从未接收过休眠通知的状态。
命令行提示：linux driver can not rmmod

示例休眠流程如下：

```
[  277.376971]     WSM firmware [=IOT=RF=Ares_AX  2GHZ Nov 28 2024 09:40:49], ver: 22570, build: 0,
api: 22570, cap: 0xC03D Config[30004]  expection 9007094, tx prog 24  expection 9007094, tx prog 24
->成功获取 startup 枚举信息
[  277.429049] [Sstar_log]:Sstar_sdio_tx_thread
[  277.433614] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(23),used(1)
[  277.440692] sync hera ba parames ok.
[  277.440764] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(22),used(1)
[  277.451030] [Sstar_log]:Sstar_get_mac_address.a8:29:fe:d4:c5
[  277.462893] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(21),used(1)
[  277.469056] Sstar_ioctl_open cost time: 110 ms
[  277.474305] [Sstar_log]:wifi mode:STA
[  277.478292] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(20),used(1)
[  277.484512] [Sstar_log]:[Sstar_wtd]:set wtd_probe = 1      ->驱动加载完成
[WIFI_MODE] STA
CTRL-EVENT-STATE-CONNECTED - Connection to e2:88:f2:b6:8a:84 completed     ->配置网络信息
[  278.514954] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(19),used(1)
[  278.521462] [Sstar_log]:Sstar_sta_connect_event: associated
[  278.527496] update ba params idx=0.
[  278.531409] ba_tid_params:action1,linkid1,tid0,ssn0.
[  278.536924] [Sstar_log]:Sstar_get_mac_address.a8:29:fe:d4:c5
```

```
[    278.543048] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(17),used(1)
[    278.549701] [Sstar_log]:tcp_port_filter addcnt.1:
[    278.554846] [Sstar_log]:Sstar_open
[    278.590258] [Sstar_wifi]: Sstar_wifi module power set by axp.
[    278.596505] create_regulator: axp22_ldo2: Failed to create debugfs directory
[    278.604358] [Sstar_wifi]: regulator on.
[    278.609164] [Sstar_wifi]: succeed to set gpio Sstar_wifi_shdn to 1 !
[    278.616024] [Sstar_wifi]: sdio wifi power state: on
[WAKEUP] IO wakeup
[    288.929023] wlan0: no IPv6 routers present
[root@sinlinx /]#
[root@sinlinx /]#
[root@sinlinx /]# >> linux driver can rmmod ...              ->IO 电平触发休眠
[     289.999567] Can't write config register.


[ 1358.272470] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(4),used(1)
[ 1358.289026] sleep lock wait cmd over...
[root@sinlinx /]# [ 1358.293444] [Sstar_log]:Sstar_sdio_xmit_deinit
[ 1358.299626] [Sstar_log]:Sstar_sdio_tx_thread:exit
[ 1358.304818] [Sstar_log]:Sstar_sdio_rev_deinit
[ 1358.309508] [Sstar_log]:Sstar_sdio_rx_thread:exit
[ 1358.314568] Sstar_wsm_fw_sleep ok! new rmmod version...
[root@sinlinx /]# [ 1358.449030] [Sstar_log]:Sstar_stop
[ 1358.465220] [Sstar_log]:Sstar_sdio_disconnect
[ 1358.469701] [Sstar_log]:Sstar_unregister_common.++
[ 1358.589968] [Sstar_log]:Sstar_unregister_common.--
[ 1358.595532] [Sstar_log]:Sstar_release_firmware
[ 1358.596132] mmc1: card 0001 removed
[ 1358.596549] sdc1 set ios: clk 0Hz bm OD pm OFF vdd 3.3V width 1 timing LEGACY(SDR12) dt B      ->卸载
完成
[root@sinlinx /]#
```

4. 指令详解

SSTAR_IOT_SUPPLICANT 工具目前主要集成了 POWER SAVE、AP CONNECT、AP CONFIG、TCP FILTER、SCAN、WIFI 相关参数设定、SMARTCONFIG、AP TOUCH、ETF RX&TX 、ADAPTIVE、VERSION、FW UPDATE、FW 指令发送、FW 打印穿透等功能，其他指令会在后续的开发中继续并入。
（以下指令除 rmmod 指令（休眠卸载指令）外，均需在驱动加载状态下操作）

4.1. set_ps

配置 power save 属性（ps_level 及 ps_type 需同时设置，即时生效）。该命令一般不推荐客户使用，系统已经有设置了 sleep 的默认值。

- **参数结构：**

ps_level：省电等级，最大 level 限制为 2，level 分级为 0、1；level 0: max throughput; level 1: max ps time, 驱动中已经默认为 level 模式；
ps_type：省电类型，包含 3 种模式，NO_SLEEP、MODEM_SLEEP、LIGHT_SLEEP。

- **示例：**

设置 power save 等级为 1，power save 模式为 light_sleep：

set_ps ps_level 1 ps_type LIGHT_SLEEP

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli set_ps ps_level 1 ps_type LIGHT_SLEEP
cmd_line: set_ps ps_level 1 ps_t ype LIGHT_SLEEP
[ 5027.565928] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(12),used(1)
OK
[root@sinlinx /]#
```

4.2. list_networks

显示当前用户配置的网络。

- **参数结构：**

无参

- **示例：**

显示当前配置的网络：

list_networks

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli list_networks
cmd_line: list_networks
ssid      /     key     / flags
```

```
                    NONE
OK
[root@sinlinx /]#
```

4.3. set_network

配置 STA 模式网络参数信息。

- **参数结构：**

ssid：所要连接 AP 的 ssid，即无线名称；
key：无线密钥；
key_id：key id；
key_mgmt：安全密钥类型，包含 4 种方式：NONE、WEP、WPA、WPA2。

- **示例：**

配置无线名称为"honor"，无线密码为"12345678"，采用的 WPA2 加密方式：

set_network ssid honor
set_network key 12345678
set_network key_mgmt WPA2

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli set_network ssid honor
cmd_line: set_network ssid honor
OK
[root@sinlinx /]# ./Sstar_iot_cli set_network key 12345678
cmd_line: set_network key 12345678
OK
[root@sinlinx /]# ./Sstar_iot_cli set_network key_mgmt WPA2
cmd_line: set_network key_mgmt WPA2
OK
[root@sinlinx /]#
```

4.4. enable_network

启动网络连接。

- **参数结构：**

无参

- **示例：**

启动网络连接：

enable_network

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli enable_network
cmd_line: enable_network
[ 5113.223975] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(11),used(1)
```

```
[root@sinlinx /]# OK
CTRL-EVENT-STATE-DISCONNECTED
[ 5113.368876] [Sstar_log]:Sstar_stop
[ 5116.834060] [Sstar_log]:__ieee80211_rx_handle_packet: mac[04:79:70:ba:04:28] not associate
[ 5116.843160] [Sstar_log]:ieee80211_rx_handlers_result:drop (4208)
[ 5116.869818] [Sstar_log]:__ieee80211_rx_handle_packet: mac[04:79:70:ba:04:28] not associate
[ 5116.878913] [Sstar_log]:ieee80211_rx_handlers_result:drop (4208)
[ 5116.918017] [Sstar_log]:__ieee80211_rx_handle_packet: mac[04:79:70:ba:04:28] not associate
[ 5116.927109] [Sstar_log]:ieee80211_rx_handlers_result:drop (4208)
[ 5117.897609] hostevent->bssid 04:79:70:ba:04:28 ipaddr 9f03a8c0
[ 5117.904259] [Sstar_log]:Sstar_sta_connect_event: associated
CTRL-EVENT-STATE-CONNECTED - Connection to 04:79:70:ba:04:28 completed
ip addr:192.168.3.159
[ 5117.921740] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(10),used(1)
[ 5117.928211] [Sstar_log]:Sstar_get_mac_address.0:1:3d:5c:c6
[ 5117.934148] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(9),used(1)
[ 5117.940625] [Sstar_log]:tcp_port_filter addcnt.1:
[ 5117.945749] dump_mem len 24
[ 5117.945753]
[ 5117.950417] 00 00 00 00 |03 00 01 01
[ 5117.954484] 13 89 00 00 |00 00 00 00
[ 5117.958549] 13 89 00 00 |00 00 00 00
[ 5117.962629]
[ 5117.964477] [Sstar_log]:Sstar_open

[root@sinlinx /]# [ 5128.428866] wlan0: no IPv6 routers present

[root@sinlinx /]#
```

4.5. clear_network

清除已配置网络。

■ **参数结构：**

无参

■ **示例：**

清除已配置网络：

clear_network

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli list_networks
cmd_line: list_networks
ssid      /     key     / flags
honor     12345678          WPA2
```

```
OK
[root@sinlinx /]# ./Sstar_iot_cli clear_network
cmd_line: clear_network
OK
[root@sinlinx /]# ./Sstar_iot_cli list_networks
cmd_line: list_networks
ssid      /     key     / flags
                        NONE
OK
[root@sinlinx /]#
```

4.6. status

获取网络状态。

- **参数结构：**

无参

- **示例：**

获取网络连接状态：

status

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli status
cmd_line: status
[ 5174.593971] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(17),used(1)
[root@sinlinx /]# wifi_mode=STA
wpa_state=ACTIVE
ssid=honor
ip_address=192.168.3.159
mac_address=04:79:70:ba:04:28
ip_mask=255.255.255.0
gate_way=192.168.3.1
OK

[root@sinlinx /]#
```

4.7. set_ip

设置板端 ip。

- **参数结构：**

无参

- **示例：**

设置板端 ip，ip 信息由网络连接成功后自动获取：

set_ip

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli set_ip
cmd_line: set_ip
Line:526 can not set ip now, please change ip_auto mode
FAIL
[root@sinlinx /]#
```

4.8. set_ip_auto

设置板端 ip 配置方式。

- **参数结构：**

1：自动配置板端 IP，默认为此项；
0：手动配置板端 IP。

- **示例：**

设置板端 ip 配置方式为自动配置：

set_ip_auto 1

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli set_ip_auto 1
cmd_line: set_ip_auto 1
OK
[root@sinlinx /]#
```

4.9. list_filters

显示当前已生效的 TCP 过滤端口号。

- **参数结构：**

无参

- **示例：**

显示当前已生效的端口：

list_filters

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli list_filters
cmd_line: list_filters
[ 5235.963969] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(16),used(1)

<TCP FILTER CONFIGURED>
SRC PORT:5001
DES PORT:5001
ICMP: SET
OK
```

```
[root@sinlinx /]#
```

4.10. set_filter

配置 filter 端口（118 端使用），源端口与目的端口最大均可配置 4 个，用户请按照实际情况配置。

■ **参数结构：**

SRC：设置源端口号，作为客户端配置；

DES：设置目的端口号，作为服务器端配置；

ICMP：配置 ICMP 过滤，默认不过滤，设置方式 1 或 0。

■ **示例：**

作为服务器端，有 5003、1028、1304 三个端口，则配置过滤目的端口号为 5003、1028、1304，另使能 ICMP 过滤：

set_filter DES 5003 1028 1304

set_filter ICMP 1

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli set_filter DES 5003 1028 1304
cmd_line: set_filter DES 5003 1028 1304
Now, you set
SRC (client):
DES (server):5003 1028 1304
ICMP: NOT SET
OK
[root@sinlinx /]# ./Sstar_iot_cli set_filter ICMP 1
cmd_line: set_filter ICMP 1
Now, you set
SRC (client):
DES (server):5003 1028 1304
ICMP: SET
OK
[root@sinlinx /]#
```

4.11. enable_filter

使能已配置 filter 端口信息。

■ **参数结构：**

无参

■ **示例：**

使能当前配置的 filter 端口信息：

enable_filter

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli enable_filter
```

```
cmd_line: enable_filter
[ 5309.523952] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(15),used(1)

OK
[root@sinlinx /]#
[root@sinlinx /]# ./Sstar_iot_cli list_filters
cmd_line: list_filters
[ 5320.193946] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(14),used(1)

<TCP FILTER CONFIGURED>
SRC PORT:
DES PORT:5003 1028 1304
ICMP: SET
OK
[root@sinlinx /]#
```

4.12. scan

启动 AP 扫描（STA 模式）。

- **参数结构：**

无参

- **示例：**

启动 AP 扫描：

scan

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli scan
cmd_line: scan
[ 5331.603951] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(13),used(1)

[root@sinlinx /]# CTRL-EVENT-SCAN-RESULTS
OK

[root@sinlinx /]#
```

4.13. scan_results

显示 AP 扫描结果（STA 模式）

扫描结果以 30 个为一组进行打印，OVER 代表显示完毕，REMAIN 代表包含多组显示，见执行显示红标）。

- **参数结构：**

无参

- **示例：**

显示 AP 扫描结果：

scan_results

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli scan_results
cmd_line: scan_results
[ 5347.173958] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(12),used(1)
[root@sinlinx /]#
get scan cnt: 25 (OVER)
bssid / level / channel / ssid
40:b0:76:ae:75:18 -46 1 16
d0:76:e7:1f:83:96 -49 1 mic_2_2.4G
54:75:95:5b:ce:ae -32 1 wifi_test_ap49
04:79:70:ba:04:28 -18 1 honor
48:0e:ec:5b:27:d8 -39 1 abcde
d8:24:bd:78:60:f0 -39 1 tt
50:fa:84:9e:f0:20 -44 1 wifi_test_ap56
b0:89:00:0d:51:54 -33 3 18
d0:76:e7:1f:83:a8 -53 4 mic_2.4G
84:34:97:ad:a8:41 -72 4 HP-Print-41-Officejet Pro 8600
50:fa:84:15:30:a8 -49 6 test
8a:25:93:7d:e6:aa -54 6 GT_East
ec:ad:e0:c2:32:a1 -47 6 wifi_test_ap67
cc:40:d0:f9:21:e0 -56 10 wifi_IOT_East_2.4G
10:44:00:3e:e0:86 -62 10 HUAWEI-B315-E086
4e:e0:10:26:04:77 -64 11 超晶科技
48:7d:2e:94:d1:79 -66 11 GT_2.4G
60:f4:45:e8:04:84 -64 11 41
c8:3a:35:2f:3e:b8 -50 13 p30ap
40:77:a9:62:af:6b -31 13
40:77:a9:62:af:6d -32 13 wifi_test_ap63
2c:61:04:fb:55:45 -62 13 wifi_test_ap61
80:89:17:ed:ba:a7 -41 13 wifi_test_ap14
40:b0:76:2d:0a:98 -52 13 wifi_IOT_2.4
40:b0:76:2c:67:88 -54 13 wfi_prj_2.4G
OK

[root@sinlinx /]#
```

4.14. wifi_mode

设置 wifi 工作模式。

■ **参数结构：**

wifi_mode：工作模式，STA 和 AP。

■ **示例：**

设置 wifi 工作模式为 AP：

wifi_mode AP

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli wifi_mode AP
cmd_line: wifi_mode AP
[ 5367.024112] [Sstar_log]:Sstar_wsm_set_wifi_mode:[3][1][ee6ebc80]
[root@sinlinx /]# [ 5367.032793] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(11),used(1)
OK
[ 5367.308880] [Sstar_log]:Sstar_stop
[ 5367.602584] hostevent->bssid 04:79:70:ba:04:28 ipaddr 12ba8c0
[ 5367.609029] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(10),used(1)
AP Mode.
[ 5367.615475] [Sstar_log]:wifi mode:AP
ip addr:192.168.43.1
[ 5367.620032] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(9),used(1)
[ 5367.628279] [Sstar_log]:Sstar_configure_wifi_mode:aes
[ 5367.635493] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(8),used(1)
[ 5367.641886] [Sstar_log]:Sstar_get_mac_address.0:1:3d:5c:c6
[ 5367.647798] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(7),used(1)
[ 5367.654307] [Sstar_log]:tcp_port_filter addcnt.3:
[ 5367.659449] dump_mem len 24
[ 5367.659453]
[ 5367.664097] 00 00 00 00 |03 00 00 03
[ 5367.668163] 00 00 00 00 |00 00 00 00
[ 5367.672244] 13 8b 04 04 |05 18 00 00
[ 5367.676307]
[ 5367.678181] [Sstar_log]:Sstar_open
[ 5367.682231] ADDRCONF(NETDEV_UP): wlan0: link is not ready

[root@sinlinx /]#
```

4.15. clear_ap_cfg

清空已配置 AP 模式参数信息。

- **参数结构：**

无参

- **示例：**

清空当前配置的 AP 模式参数信息：

clear_ap_cfg

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli clear_ap_cfg
cmd_line: clear_ap_cfg
OK
[root@sinlinx /]#
```

4.16. set_ap_cfg

设置 AP 模式参数信息。

- **参数结构：**

ssid：AP 模式下的 ssid，即无线名称；
key：无线密钥；
key_id：key id；
key_mgmt：安全密钥类型，包含四种方式：NONE、WEP、WPA、WPA2。

- **示例：**

设置 AP 模式下无线名称为“hello_Sigmastar”，无线密码为“12345678”，采用的 WPA2 加密方式：

set_ap_cfg ssid hello_Sigmastar
set_ap_cfg key 12345678
set_ap_cfg key_mgmt WPA2

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli set_ap_cfg ssid hello_Sigmastar
cmd_line: set_ap_cfg ssid hello_Sigmastar
OK
[root@sinlinx /]# ./Sstar_iot_cli set_ap_cfg key 12345678
cmd_line: set_ap_cfg key 12345678
OK
[root@sinlinx /]# ./Sstar_iot_cli set_ap_cfg key_mgmt WPA2
cmd_line: set_ap_cfg key_mgmt WPA2
OK
[root@sinlinx /]#
```

4.17. enable_ap_cfg

使能 AP 模式参数配置，并启动 AP 模式。

- **参数结构：**

无参

- **示例：**

使能 AP 模式参数配置，启动 AP 模式：

enable_ap_cfg

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli enable_ap_cfg
cmd_line: enable_ap_cfg
```

```
[ 5495.833935] [Sstar_log]:Sstar_wsm_set_ap_cfg:aes
[root@sinlinx /]# [ 5495.841181] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(6),used(1)
[ 5496.505407] hostevent->bssid 04:79:70:ba:04:28 ipaddr 12ba8c0
[ 5496.554799] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(5),used(1)
AP Mode.
[ 5496.561176] [Sstar_log]:wifi mode:AP
ip addr:192.168.43.1
[ 5496.565756] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(4),used(1)
[ 5496.574011] [Sstar_log]:Sstar_configure_wifi_mode:aes
OK
[root@sinlinx /]# ./Sstar_iot_cli status
cmd_line: status[ 5520.144073] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(3),used(1)

wifi_mode=AP
wpa_state=ACTIVE
ssid=hello_Sigmastar
ip_address=192.168.43.1
mac_address=00:00:01:3d:5c:c6
ip_mask=255.255.255.0
gate_way=192.168.43.1
OK
[root@sinlinx /]#
```

4.18. switch_channel

切换 wifi 信道。

■ **参数结构：**

channel：信道，1~14。

■ **示例：**

切换 wifi 信道至 6：

switch_channel 6

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli switch_channel 6
cmd_line: switch_channel 6
[ 5540.924137] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(2),used(1)
[root@sinlinx /]# [ 5541.389261] hostevent->bssid 04:79:70:ba:04:28 ipaddr 12ba8c0
[ 5542.096634] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(1),used(1)
AP Mode.
[ 5542.103006] [Sstar_log]:wifi mode:AP
ip addr:192.168.43.1
[ 5542.107584] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(0),used(1)
[ 5542.115863] [Sstar_log]:Sstar_configure_wifi_mode:aes
```

```
[ 5542.121441] hostevent->bssid 04:79:70:ba:04:28 ipaddr 12ba8c0
[ 5542.127912] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(17),used(1)
AP Mode.
[ 5542.134504] [Sstar_log]:wifi mode:AP
ip addr:192.168.43.1
[ 5542.138941] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(16),used(1)
[ 5542.147253] [Sstar_log]:Sstar_configure_wifi_mode:aes
[ 5542.198906] [Sstar_log]:Sstar_stop
[ 5542.203577] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(15),used(1)
[ 5542.210058] [Sstar_log]:Sstar_get_mac_address.0:1:3d:5c:c6
[ 5542.215975] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(14),used(1)
[ 5542.222534] [Sstar_log]:tcp_port_filter addcnt.3:
[ 5542.227658] dump_mem len 24
[ 5542.227662]
[ 5542.232330] 00 00 00 00 |03 00 00 03
[ 5542.236397] 00 00 00 00 |00 00 00 00
[ 5542.240484] 13 8b 04 04 |05 18 00 00
[ 5542.244547]
[ 5542.246394] [Sstar_log]:Sstar_open
[ 5542.250475] ADDRCONF(NETDEV_UP): wlan0: link is not ready
OK

[root@sinlinx /]#
```

4.19. set_country

设置国家码。

■ **参数结构：**

country_no：国家码，包含：CHINESE、USA、EUROPE、JAPAN、CANADA、AUSTRALIA、ISRAEL、MEXICO、FRANCE。

■ **示例：**

设置国家码为 USA：

set_country USA

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli set_country USA
cmd_line: set_country USA
[ 5562.374375] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(13),used(1)
set country: 1
OK
[root@sinlinx /]#
```

4.20. get_country

获取国家码。

■ **参数结构：**

无参

■ **示例：**

获取国家码：

get_country

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli get_country
cmd_line: get_country[ 5573.433968] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(12),used(1)

country: USA
OK
[root@sinlinx /]#
```

4.21. get_stalist

获取 AP 模式下已连接的 STA 列表。

■ **参数结构：**

无参

■ **示例：**

获取当前 STA 列表：

get_stalist

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# [ 5610.617149] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(11),used(1)
[ 5610.623619] Sstar_sta_add_event   mac_addr:54:ba:d6:6a:80:c4
[ 5610.629739] [Sstar_log]:add[54:ba:d6:6a:80:c4],aid[1],enc[1],key_type(4),iv(8),icv(8)
[ 5610.638690] ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready

[root@sinlinx /]# [ 5621.158865] wlan0: no IPv6 routers present

[root@sinlinx /]# ./Sstar_iot_cli get_stalist
cmd_line: get_stalist
[ 5629.544126] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(0),used(1)
[root@sinlinx /]# STA List:
54:ba:d6:6a:80:c4

OK
[root@sinlinx /]#
```

4.22. smart_cfg_start

启动智能联网。

■ **参数结构：**

无参

■ **示例：**

启动智能联网：

smart_cfg_start

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli smart_cfg_start
cmd_line: smart_cfg_start
[ 5740.164221] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(12),used(1)
[root@sinlinx /]# OK[ 5740.895371] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(11),used(1)

[ 5740.901984] Sstar_sta_loss_event   mac_addr:54:ba:d6:6a:80:c4
[ 5741.038875] [Sstar_log]:Sstar_stop

[root@sinlinx /]#
```

4.23. smart_cfg_stop

停止智能联网。

■ **参数结构：**

无参

■ **示例：**

停止智能联网：

smart_cfg_stop

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli smart_cfg_stop
cmd_line: smart_cfg_stop
[ 5763.834167] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(10),used(1)
[root@sinlinx /]# OK

[root@sinlinx /]#
```

4.24. ap_touch_start

启动 AP Touch 智能联网。

■ **参数结构：**

notify_type：通知类型，包含：AP_NOTIFY、NO_NOTIFY，默认无通知。

■ **示例：**

启动 AP Touch 智能联网，AP 通知模式：

ap_touch_start AP_NOTIFY

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli ap_touch_start AP_NOTIFY
cmd_line: ap_touch_start AP_NOTIFY
[ 5797.784175] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(9),used(1)
OK
[root@sinlinx /]# [ 5798.282277] hostevent->bssid 54:ba:d6:6a:80:c4 ipaddr 12aa8c0
[ 5798.288698] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(8),used(1)
AP Mode.[ 5798.295108] [Sstar_log]:wifi mode:AP

ip addr:192.168.42.1
[ 5798.299430] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(7),used(1)
[ 5798.307808] [Sstar_log]:Sstar_configure_wifi_mode:aes
[ 5798.315319] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(6),used(1)
[ 5798.321701] [Sstar_log]:Sstar_get_mac_address.0:1:3d:5c:c6
[ 5798.327614] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(5),used(1)
[ 5798.334071] [Sstar_log]:tcp_port_filter addcnt.3:
[ 5798.339214] dump_mem len 24
[ 5798.339218]
[ 5798.343862] 00 00 00 00 |03 00 00 03
[ 5798.347928] 00 00 00 00 |00 00 00 00
[ 5798.352010] 13 8b 04 04 |05 18 00 00
[ 5798.356073]
[ 5798.357920] [Sstar_log]:Sstar_open
[ 5798.361989] ADDRCONF(NETDEV_UP): wlan0: link is not ready

[root@sinlinx /]#
```

4.25. ap_touch_stop

停止 AP Touch 智能联网。

■ **参数结构：**

无参

■ **示例：**

停止 AP Touch 智能联网：

ap_touch_stop

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli ap_touch_stop
cmd_line: ap_touch_stop[ 5820.114105] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(4),used(1)
```

```
[root@sinlinx /]# OK
[ 5821.018878] [Sstar_log]:Sstar_stop
[ 5822.248159] [Sstar_log]:__ieee80211_rx_handle_packet: mac[04:79:70:ba:04:28] not associate
[ 5822.257288] [Sstar_log]:ieee80211_rx_handlers_result:drop (4208)
[ 5822.276902] [Sstar_log]:__ieee80211_rx_handle_packet: mac[04:79:70:ba:04:28] not associate
[ 5822.286021] [Sstar_log]:ieee80211_rx_handlers_result:drop (4208)
[ 5822.318909] [Sstar_log]:__ieee80211_rx_handle_packet: mac[04:79:70:ba:04:28] not associate
[ 5822.328013] [Sstar_log]:ieee80211_rx_handlers_result:drop (4208)
[ 5823.297236] hostevent->bssid 04:79:70:ba:04:28 ipaddr 9f03a8c0
[ 5823.303771] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(3),used(1)
AP Mode.
[ 5823.310163] [Sstar_log]:wifi mode:STA
ip addr:192.168.3.159
[ 5823.322104] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(2),used(1)
[ 5823.328504] [Sstar_log]:Sstar_sta_connect_event: associated
[ 5823.334522] [Sstar_log]:Sstar_get_mac_address.0:1:3d:5c:c6
[ 5823.340453] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(1),used(1)
[ 5823.346916] [Sstar_log]:tcp_port_filter addcnt.3:
[ 5823.352057] dump_mem len 24
[ 5823.352061]
[ 5823.356705] 00 00 00 00 |03 00 00 03
[ 5823.360787] 00 00 00 00 |00 00 00 00
[ 5823.364854] 13 8b 04 04 |05 18 00 00
[ 5823.368932]
[ 5823.370779] [Sstar_log]:Sstar_open

[root@sinlinx /]# [ 5834.118867] wlan0: no IPv6 routers present
[root@sinlinx /]#
```

4.26. set_etf_cfg

设置 etf_rx 与 etf_tx 的参数信息。

■ **参数结构：**

channel：信道，1~14；

mode：速率模式，0-11b，1-11g，2-11n，3-HE_SU，4-HE_ER_SU（mode 设置需要先于 rate）；

rate：传输速率值，与 mode 搭配。

mode 为 0 时，rate 为 0~3，即 1M、2M、5M、11M；

mode 为 1 时，rate 为 0~7，即 6M、9M、12M、18M、24M、36M、48M、54M；

mode 为 2 时，rate 为 0~7，即 MCS0~MCS7；

mode 为 3 时，rate 为 0~11，即 MCS0~MCS11；

mode 为 4 时，rate 为 0~2，即 MCSER0~MCSER2；

40m：使能 40M 带宽，使能为 1，默认不使能；

greenfield：使能 greenfield 模式，使能为 1，默认不使能；

len：发包长度；

ldpc：0 或 1，用于配置是否使能 LDPC。

- **示例：**

设置 etf 参数，2 信道，5.5Mbps，发包长度 300：

set_etf_cfg channel 2
set_etf_cfg mode 0
set_etf_cfg rate 2
set_etf_cfg len 300

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli set_etf_cfg channel 2
cmd_line: set_etf_cfg channel 2
OK
[root@sinlinx /]# ./Sstar_iot_cli set_etf_cfg mode 0
cmd_line: set_etf_cfg mode 0
OK
 [root@sinlinx /]# ./Sstar_iot_cli set_etf_cfg rate 2
cmd_line: set_etf_cfg rate 2
OK
[root@sinlinx /]# ./Sstar_iot_cli set_etf_cfg len 300
cmd_line: set_etf_cfg len 300
OK
[root@sinlinx /]#
```

4.27. etf_start_tx

启动射频发包模式（通道信息由 set_etf_cfg 配置）。

- **参数结构：**

无参

- **示例：**

启动射频发包模式：

etf_start_tx

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli etf_start_tx
cmd_line: etf_start_tx[ 5934.243992] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(8),used(1)

Current channel:2   rate:5   len:300   is40M:0   greenfield:0
mode:3 ldpc:0
[root@sinlinx /]# [ 5934.488874] [Sstar_log]:Sstar_stop
OK

[root@sinlinx /]#
```

4.28. etf_stop_tx

停止射频发包模式。

- **参数结构：**

无参

- **示例：**

停止射频发包模式：

etf_stop_tx

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli etf_stop_tx
cmd_line: etf_stop_tx[ 5956.294170] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(7),used(1)

[root@sinlinx /]# OK

[root@sinlinx /]#
```

4.29. etf_singletone

发送定频（通道信息由 set_etf_cfg 配置）。

- **参数结构：**

无参

- **示例：**

发送定频：

etf_singletone

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli etf_singletone
cmd_line: etf_singletone
[ 5971.464047] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(6),used(1)
Current channel:2
[root@sinlinx /]# OK

[root@sinlinx /]#
```

4.30. etf_start_rx

启动射频收包模式（通道信息由 set_etf_cfg 配置，需停止 etf_tx 流程）。

- **参数结构：**

无参

- **示例：**

启动射频收包模式：

etf_start_rx

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli etf_start_rx
cmd_line: etf_start_rx
[    46.504482] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(10),used(1)

Current channel:3   is40M:0
[WAKEUP] IO wakeup
CTRL-EVENT-STATE-DISCONNECTED
[root@sinlinx /]#
[    46.548986] [Sstar_log]:Sstar_sta_disconnect_event: deauth
[    46.554892] clear ba params.
[root@sinlinx /]# [    46.618957] [Sstar_log]:Sstar_stop
OK
```

4.31. etf_start_rx_no_drop

启动射频收包模式（monitor 模式，通道信息由 set_etf_cfg 配置，需停止 etf_tx 流程）。

- **参数结构：**

无参

- **示例：**

启动射频收包模式（monitor 模式）：

etf_start_rx_no_drop

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli etf_start_rx_no_drop
cmd_line: etf_start_rx_no_drop[    139.724357] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(8),used(1)

Current channel:3   is40M:0
OK
[root@sinlinx /]#
```

4.32. etf_stop_rx

停止射频收包模式。

- **参数结构：**

无参

- **示例：**

停止射频收包模式：

etf_stop_rx

- **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli etf_stop_rx
cmd_line: etf_stop_rx
[   182.094390] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(7),used(1)
[root@sinlinx /]#
OK
[root@sinlinx /]#
```

4.33. etf_reset_rx

清零射频收包统计数据（在 etf_start_rx 状态下使用）。

■ **参数结构：**

无参

■ **示例：**

清零射频收包统计数据：

etf_reset_rx

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli etf_reset_rx
cmd_line: etf_reset_rx[    213.104264] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(5),used(1)

OK
[root@sinlinx /]#
```

4.34. adaptive

配置自适应功能。

■ **参数结构：**

flag：自适应启动标识，包含：ON 和 OFF。

■ **示例：**

设置自适应开启：

adaptive ON

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli adaptive ON
cmd_line: adaptive ON[ 6078.444186] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(17),used(1)

OK
[root@sinlinx /]#
```

4.35. get_ver

获取系统版本信息。

■ **参数结构：**

无参

■ **示例：**

获取系统版本信息：

get_ver

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli get_ver
cmd_line: get_ver[   245.404245] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(3),used(1)

[HW FW Version]
 SSTAR:6461 fwVer:22570 SDK ver:0.1.0

OK
[root@sinlinx /]#
```

4.36. get_sdk_ver

获取客户 SDK 版本信息。

■ **参数结构：**

无参

■ **示例：**

获取客户 SDK 版本信息：

get_sdk_ver

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli get_sdk_ver
cmd_line: get_sdk_ver
[   287.354338] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(2),used(1)
[root@sinlinx /]#
[SDK Version]
 0.2.4

OK
[root@sinlinx /]#
```

4.37. update_fw

升级固件（烧写完毕固件断电重启或手动输入 AT 指令控制固件板重启后即版本生效）。

■ **参数结构：**

path：固件路径。

■ **示例：**

升级存放在同级目录的 iot_ota5.bin：

update_fw iot_ota5.bin

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli update_fw iot_ota5.bin
file:iot_ota5.bin    size:279064 bytes
[    811.290180] [WSM] >>> 0x0020 (1044)
[    811.296820] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(21),used(1)
[    812.814164] [WSM] >>> 0x0020 (1044)
[    812.818052] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(20),used(1)
[    812.824583] [WSM] >>> 0x0020 (1044)
[    812.828471] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(19),used(1)
[    812.964729] [WSM] >>> 0x0020 (1044)
[    812.968615] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(10),used(1)
[    813.042116] [WSM] >>> 0x0020 (1044)
[    813.046002] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(5),used(1)
[    813.057723] [WSM] >>> 0x0020 (1044)
[    813.061636] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(4),used(1)
[    813.073172] [WSM] >>> 0x0020 (1044)
[    813.077058] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(3),used(1)
[    813.088677] [WSM] >>> 0x0020 (1044)
[    813.092571] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(2),used(1)
[    813.104110] [WSM] >>> 0x0020 (1044)
[    813.107997] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(1),used(1)
[    813.119610] [WSM] >>> 0x0020 (1044)
[    813.123496] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(0),used(1)
[    813.135033] [WSM] >>> 0x0020 (1044)
[    813.138918] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(22),used(1)
[    813.306299] [WSM] >>> 0x0020 (1044)
[    813.310205] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(11),used(1)
[    813.321856] [WSM] >>> 0x0020 (1044)
[    813.321880] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(10),used(1)
[    813.327518] [WSM] >>> 0x0020 (1044)
[    813.327543] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(9),used(1)
[    813.333020] [WSM] >>> 0x0020 (1044)
[    813.333044] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(8),used(1)
[    813.338652] [WSM] >>> 0x0020 (1044)
[    813.338676] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(7),used(1)
[    814.692638] [WSM] >>> 0x0020 (1044)
[    814.692663] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(2),used(1)
[    814.721140] [WSM] >>> 0x0020 (1044)
[    814.721164] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(0),used(1)
```

```
[    815.032556] [WSM] >>> 0x0020 (1044)
[    815.032583] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(15),used(1)
[    815.038123] [WSM] >>> 0x0020 (1044)
[    815.038147] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(14),used(1)
[    815.043745] [WSM] >>> 0x0020 (1044)
[    815.043769] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(13),used(1)
[    815.049287] [WSM] >>> 0x0020 (1044)
[    815.111953] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(7),used(1)
[    815.117451] [WSM] >>> 0x0020 (1044)
[    815.117475] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(6),used(1)
[    815.123133] [WSM] >>> 0x0020 (1044)
[    815.123157] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(5),used(1)
[    815.163134] [WSM] >>> 0x0020 (1044)
[    815.163157] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(1),used(1)
[    815.168729] [WSM] >>> 0x0020 (1044)
[    815.168752] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(0),used(1)
[    828.408199] [WSM] >>> 0x0020 (1044)
[    828.412242] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(22),used(1)
OK
[root@sinlinx /]#
```

4.38. listen_itvl

设置 beacon 监听间隔。

- **参数结构：**

interval：监听间隔，最大为 20（unit:100ms）

- **示例：**

设置 beacon 监听间隔为 1s：

listen_itvl 10

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli listen_itvl 10
cmd_line: listen_itvl 10
[ 6432.044140] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(11),used(1)
[root@sinlinx /]# OK

[root@sinlinx /]#
```

4.39. fw_cmd

发送固件 AT 指令。

- **参数结构：**

固件 AT 指令。

- **示例：**

设置固件指令，使能 LIGHT_SLEEP：

fw_cmd AT+LIGHT_SLEEP 1

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli fw_cmd AT+LIGHT_SLEEP 1
cmd_line: fw_cmd AT+LIGHT_SLEEP 1
[ 6460.374258] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(10),used(1)
OK
[root@sinlinx /]#
```

4.40. fw_print

设置固件打印穿透使能。

- **参数结构：**

0：不穿透；

1：穿透。

- **示例：**

设置固件打印穿透：

fw_print 1

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
cmd_line: fw_print 1
[ 6478.134136] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(9),used(1)
[root@sinlinx /]# OK
[ 6489.718430] AT+LIGHT_SLEEP 1
[ 6489.718436] AT_LightSleep:1
[ 6489.718440] +OK
[ 6494.022082] AT+LIGHT_SLEEP 1
[ 6494.022087] AT_LightSleep:1
[ 6494.022090] +OK

[root@sinlinx /]# ./Sstar_iot_cli fw_print 0
cmd_line: fw_print 0[ 6500.694141] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(8),used(1)

OK
[root@sinlinx /]#
```

4.41. rmmod

发送休眠（卸载）指令（仅在可休眠状态下生效）。

- **参数结构：**

无参

- **示例：**

触发休眠：

rmmod

■ **返回值：**

无

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli rmmod
cmd_line: rmmod
[   333.914362] Sstar_wsm_pre_rmmod new rmmod version...
[   333.921284] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(1),used(1)
[root@sinlinx /]#
[   333.939026] sleep lock wait cmd over...
[root@sinlinx /]# [   333.943439] [Sstar_log]:Sstar_sdio_xmit_deinit
[   333.949713] [Sstar_log]:Sstar_sdio_tx_thread:exit
[   333.954776] [Sstar_log]:Sstar_sdio_rev_deinit
[   333.959447] [Sstar_log]:Sstar_sdio_rx_thread:exit
[   333.964507] Sstar_wsm_fw_sleep ok! new rmmod version...
[   333.992752] [Sstar_log]:Sstar_sdio_disconnect
[   333.997209] [Sstar_log]:Sstar_unregister_common.++
[   334.109869] [Sstar_log]:Sstar_unregister_common.--
[   334.115450] [Sstar_log]:Sstar_release_firmware
[   334.116044] mmc1: card 0001 removed
[   334.116456] sdc1 set ios: clk 0Hz bm OD pm OFF vdd 3.3V width 1 timing LEGACY(SDR12) dt B
```

4.42. set_rmmod_auto

设置自动休眠（卸载）使能（默认使能自动）。

■ **参数结构：**

1：使能自动；

0：使能手动。

■ **示例：**

设置手动控制休眠：

set_rmmod_auto 0

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
>> --- ip connected
[   332.950427] wlan0: no IPv6 routers present
>> linux driver can rmmod ..  可休眠状态
[   333.848498] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(16),used(1)   自动进入休眠流程

[   333.939026] sleep lock wait cmd over...
[root@sinlinx /]# [   333.943439] [Sstar_log]:Sstar_sdio_xmit_deinit
[   333.949713] [Sstar_log]:Sstar_sdio_tx_thread:exit
[   333.954776] [Sstar_log]:Sstar_sdio_rev_deinit
```

```
[    333.959447] [Sstar_log]:Sstar_sdio_rx_thread:exit
[    333.964507] Sstar_wsm_fw_sleep ok! new rmmod version...
[    333.992752] [Sstar_log]:Sstar_sdio_disconnect
[    333.997209] [Sstar_log]:Sstar_unregister_common.++
[    334.109869] [Sstar_log]:Sstar_unregister_common.--
[    334.115450] [Sstar_log]:Sstar_release_firmware
[    334.116044] mmc1: card 0001 removed
[    334.116456] sdc1 set ios: clk 0Hz bm OD pm OFF vdd 3.3V width 1 timing LEGACY(SDR12) dt B

[    421.920174]
[    421.920179] DRIVER VER: 2024-1125-1120.     重新加载驱动
[    421.927122] [Sstar_log]:Sstar_init_firmware
[    421.931623] [Sstar_log]:----drvier support chip RHEA
[    421.937135] [Sstar_wifi]: Sstar_wifi module power set by axp.
[    421.943372] create_regulator: axp22_ldo2: Failed to create debugfs directory
[    421.951214] [Sstar_wifi]: regulator on.
[    421.955951] [Sstar_wifi]: succeed to set gpio Sstar_wifi_shdn to 1 !
[    421.962833] [Sstar_wifi]: sdio wifi power state: on
[    421.968603] sdc1 set ios: clk 0Hz bm PP pm UP vdd 3.3V width 1 timing LEGACY(SDR12) dt B
[    421.998966] sdc1 set ios: clk 400000Hz bm PP pm ON vdd 3.3V width 1 timing LEGACY(SDR12) dt B
[    422.079654] sdc1 set ios: clk 400000Hz bm PP pm ON vdd 3.3V width 1 timing LEGACY(SDR12) dt B
[    422.091667] sdc1 set ios: clk 400000Hz bm PP pm ON vdd 3.3V width 1 timing LEGACY(SDR12) dt B
[    422.102546] ********************Try sdio********************
[    422.108975] sdc1 set ios: clk 400000Hz bm PP pm ON vdd 3.3V width 1 timing LEGACY(SDR12) dt B
[    422.119818] SDIO_CCCR_CCCR>>>>>>>>>>>>>>>>>>>>>>>32
[    422.125559] SDIO_CCCR_CAPS>>>>>>>>>>>>>>>>>>>>>>>=1f
[    422.138889] sdc1 set ios: clk 400000Hz bm PP pm ON vdd 3.3V width 1 timing SD-HS(SDR25) dt B
[    422.148317] sdc1 set ios: clk 50000000Hz bm PP pm ON vdd 3.3V width 1 timing SD-HS(SDR25) dt B
[    422.169537] [Sstar_log]:Sstar: mmc2 is not found.
[    422.209024] sdc1 set ios: clk 50000000Hz bm PP pm ON vdd 3.3V width 4 timing SD-HS(SDR25) dt B
[    422.219568] mmc1: new high speed SDIO card at address 0001
[    422.226677] [Sstar_log]:Probe called
[    422.230682] [Sstar_log]:Sstar_sdio_probe:v12
[    422.236570] [Sstar_log]:Sstar_after_load_firmware++
[    422.236584] [Sstar_log]:Sstar_sdio_rx_thread
[    422.256815] [Sstar_log]:set_block_size=256
[    422.258955] [Sstar_log]:rx timeout
[    422.265074] [Sstar_log]:mdelay wait wsm_startup_done   !!
[    422.270910] [Sstar_log]:Sstar_sdio_irq_period:Miss
[    422.276303] [Sstar_log]:firmwareCap c03d
[    422.280580] [Sstar_log]:firmwareCap2 0
[    422.284642] CAPABLITIES_SDIO_TX_LIST_CHECK     4
[    422.289498] CAPABLITIES_TX_CONFIRM_NOTIFY      8
[    422.294235] CAPABLITIES_HOST_CONFIG             16
```

```
[    422.298585] CAPABLITIES_HOST_BA_PARAM                          32
[    422.303147] [Sstar_log]:wsm_caps.firmwareCap c03d
[    422.308088] [Sstar_log]:apollo wifi WSM init done.
[    422.308093]       Input buffers: 24 x 1632 bytes
[    422.308098]       Hardware: 0.25697
[    422.308101]       WSM firmware [=IOT=RF=Ares_AX    2GHZ Nov 28 2024 09:40:49], ver: 22570, build: 0,
api: 22570, cap: 0xC03D Config[30004]    expection 9007094, tx prog 24
[    422.358968] [Sstar_log]:Sstar_sdio_tx_thread
[    422.363532] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(23),used(1)
[    422.370523] sync hera ba parames ok.
[    422.374564] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(22),used(1)
[    422.381013] [Sstar_log]:Sstar_get_mac_address.a8:29:fe:d4:c5
[    422.393008] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(21),used(1)
[    422.399515] [Sstar_log]:wifi mode:STA
[    422.403495] [Sstar_log]:[Sstar_wtd]:set wtd_probe = 1
[    422.408968] Sstar_ioctl_open cost time: 120 ms
[    422.413879] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(20),used(1)
[WIFI_MODE] STA
./Sstar_iot_cli set_rmmod_auto 0    使能手动
cmd_line: set_rmmod_auto 0
OK
[root@sinlinx /]# >> linux driver can rmmod ...
[root@sinlinx /]#  可休眠状态未自动休眠
```

4.43. get_rate

获取通信速率。

■ **参数结构：**

无参

■ **示例：**

获取当前的通信速率：

get_rate

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli get_rate
cmd_line: get_rate
[    609.174527] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(19),used(1)
[root@sinlinx /]#
rate: 68
Line:227 fail
FAIL
[root@sinlinx /]#
```

4.44. send_ipc_data

透传 IPC_DATA。

■ **参数结构：**

server：服务器地址

port：端口号

data：IPC_DATA

cnt：发送次数（默认 1 次）

■ **示例：**

传输数据 i_am_a_apple 到服务器 192.168.3.147，服务器端口号为 10010，发送 3 次：

./Sstar_iot_cli send_ipc_data server=192.168.3.147 port=10010

./Sstar_iot_cli send_ipc_data data=i_am_a_apple cnt=3

或

./Sstar_iot_clit_cli send_ipc_data server=192.168.3.147 port=10010 data=i_am_a_apple cnt=3

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli send_ipc_data server=192.168.3.147 port=10010    先发送服务器信息
cmd_line: send_ipc_data server=192.168.3.147 port=10010
ipc data err or not dound.
FAIL
[root@sinlinx /]#
[root@sinlinx /]# ./Sstar_iot_cli send_ipc_data data=i_am_a_apple cnt=3   再发送数据
cmd_line: send_ipc_data data=i_am_a_apple
[   499.894162] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(9),used(1)
OK
[root@sinlinx /]#
```

或

```
[root@sinlinx/]# ./Sstar_iot_clit_cli send_ipc_data server=192.168.3.147 port=10010 data=i_am_a_apple
cnt=3   服务器信息和数据一起发送
cmd_line: send_ipc_data server=192.168.3.147 port=10010 data=i_am_a_apple
[   539.914314] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(7),used(1)
OK
[root@sinlinx /]#
```

4.45. get_rssi

获取 rssi。

■ **参数结构：**

无参

■ **示例：**

获取 rssi：

./Sstar_iot_cli get_rssi

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx/]# ./Sstar_iot_clit_cli get_rssi
cmd_line: get_rssi
[   539.914314] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(8),used(1)
current rssi: -26
OK
[root@sinlinx /]#
```

4.46. get_time

获取当前网络时间。

■ **参数结构：**

无参

■ **示例：**

获取当前网络时间：

./Sstar_iot_cli get_time

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx/]# ./Sstar_iot_clit_cli get_time
cmd_line: get_time
[   70.895184] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(17),used(1)
Current time:
Thu Jan 01 00:00:00 1970
OK
[root@sinlinx /]#
```

4.47. clear_wifi_cfg

清空网络配置并断开 AP 连接。

■ **参数结构：**

无参

■ **示例：**

清空网络配置：

./Sstar_iot_cli clear_wifi_cfg

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx/]# ./Sstar_iot_clit_cli clear_wifi_cfg
cmd_line: clear_wifi_cfg
[   455.273304] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(8),used(1)
[root@sinlinx /]# CTRL-EVENT-STATE-DISCONNECTED
[   455.558065] [Sstar_log]:Sstar_stop
```

```
OK
[root@sinlinx /]#
```

4.48. get_etf_rx_info

获取射频收包期间的统计数据（仅在射频收包期间有效）。

■ **参数结构：**

无参

■ **示例：**

配置完毕并启动射频收包，获取射频收包期间的统计数据：

./Sstar_iot_cli get_etf_rx_info

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli set_etf_cfg channel 3
cmd_line: set_etf_cfg channel 3
OK
[root@sinlinx /]# ./Sstar_iot_cli etf_start_rx
cmd_line: etf_start_rx
[   724.074487] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(17),used(1)
Current channel:3   is40M:0
OK
[root@sinlinx /]#
[root@sinlinx /]#
[root@sinlinx /]#
[root@sinlinx /]# ./Sstar_iot_cli get_etf_rx_info
cmd_line: get_etf_rx_info
[   728.594437] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(16),used(1)

current etf rx: success_cnt334, fcs_err_cnt:317, per:48%, rssi:-52
OK
[root@sinlinx /]#
```

4.49. add_netpattern

添加网络文本，用于休眠下唤醒匹配。

■ **参数结构：**

index：网络文本索引，0~3

ip：匹配 ip 地址

port：匹配端口号

protocol：匹配协议，TCP or UDP

pattern：文本数据

■ **示例：**

配置网络文本信息，索引 0，ip 信息为 192.168.2.3，端口号 1234，采用 TCP 协议，匹配文本数据为“123”：

./Sstar_iot_cli add_netpattern 0 192.168.2.3 1234 TCP 123

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]#./Sstar_iot_cli add_netpattern 0 192.168.2.3 1234 TCP 123
cmd_line: add_netpattern 0 192.168.2.3 1234 TCP 123
[ 2063.986220] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(10),used(1)
OK
[root@sinlinx /]#
```

4.50. del_netpattern

删除已配置网络文本。

■ **参数结构：**

index：网络文本索引，0~3

■ **示例：**

删除对应索引为 0 的网络文本：

./Sstar_iot_cli del_netpattern 0

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]#./Sstar_iot_cli del_netpattern 0
cmd_line: del_netpattern 0
[ 2092.094917] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(8),used(1)
[root@sinlinx /]# OK
```

4.51. add_conn

添加 socket 保活通信配置（不启动连接）。

■ **参数结构：**

index：配置索引号，0~3

ip：欲通信的 ip 地址

port：欲通信的端口号

itvl：通信数据发送周期，单位毫秒

protocol：协议，TCP or UDP

text：欲发送的文本数据

■ **示例：**

添加 socket 连接配置，使用索引 0，对端信息为 192.168.3.155，端口号为 1234，数据发送周期为 1 秒，采用 TCP 协议，数据信息为 hello_world：

./Sstar_iot_cli add_conn 0 192.168.3.155 1234 1000 TCP hello_world

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]#./Sstar_iot_cli add_conn 0 192.168.3.155 1234 1000 TCP hello_world
cmd_line: add_conn 0 192.168.3.155 1234 1000 TCP hello_world
[ 2122.575009] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(7),used(1)
```

```
OK
[root@sinlinx /]#
```

4.52. del_conn

删除已添加的 socket 保活通信配置。

- **参数结构：**

index：配置索引号，0~3

- **示例：**

删除索引为 0 的保活通信配置：

./Sstar_iot_cli del_conn 0

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]#./Sstar_iot_cli del_conn 0
cmd_line: del_conn 0
[ 2140.394912] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(6),used(1)
OK
[root@sinlinx /]#
```

4.53. conn_lose

获取当前断连的 conn 映射索引表。

- **参数结构：**

无参

- **示例：**

获取断连 conn 映射索引表：

./Sstar_iot_cli conn_lose

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]#./Sstar_iot_cli conn_lose
cmd_line: conn_lose
[ 2154.424786] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(5),used(1)
conn lose map: 0x0
OK
```

4.54. conn_switch

启动或关闭已配置的 socket 保活通信设置。

- **参数结构：**

index：conn 配置索引，0~3

enable：使能配置，ON or OFF

- **示例：**

启动已配置索引 0 的 socket 连接：

./Sstar_iot_cli conn_switch 0 ON

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli conn_switch 0 ON
cmd_line: conn_switch 0 ON
[ 2176.155028] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(4),used(1)
OK
```

4.55. add_wk_ssid

添加唤醒 ssid，用于休眠下断网情况扫描 ssid 唤醒支持。

- **参数结构：**

ssid：SSID

- **示例：**

配置唤醒 ssid 信息：

./Sstar_iot_cli add_wk_ssid hello_wkup

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]#./Sstar_iot_cli add_wk_ssid hello_wkup
cmd_line: add_wk_ssid hello_wkup
[ 2244.824908] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(1),used(1)
OK
[root@sinlinx /]#
```

4.56. del_wk_ssid

删除已配置的唤醒 ssid。

- **参数结构：**

ssid：SSID

- **示例：**

删除已配置唤醒 ssid 信息：

./Sstar_iot_cli del_wk_ssid hello_wkup

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]#./Sstar_iot_cli del_wk_ssid hello_wkup
cmd_line: del_wk_ssid hello_wkup
[ 2261.504944] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(0),used(1)
OK
[root@sinlinx /]#
```

4.57. wkup_event

设置唤醒事件支持（用于使能唤醒控制，如 ssid 唤醒、netpattern 唤醒）。

- **参数结构：**

event：唤醒事件，TCP_PATTERN、UDP_PATTERN、WK_SSID、MAGIC_PKT、NONE（清除设置）

- **示例：**

配置使能 ssid 唤醒支持：

./Sstar_iot_cli wkup_event WK_SSID

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]#./Sstar_iot_cli wkup_event WK_SSID
cmd_line: wkup_event WK_SSID
[ 2280.784875] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(17),used(1)
OK
[root@sinlinx /]#
```

4.58. auto_reconn

配置 ap 断开自动重连使能。

- **参数结构：**

enable：使能，0 or Other

- **示例：**

配置使能自动重连：

./Sstar_iot_cli auto_reconn 1

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]#./Sstar_iot_cli auto_reconn 1
cmd_line: auto_reconn 1
[ 2323.854937] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(16),used(1)
OK
[root@sinlinx /]#
```

4.59. customer_cmd（预留）

客户定制化指令。

- **参数结构：**

cmd_id：指令索引

cmd_buff：指令数据

- **示例：**

发送 cmd_id 为 0 的指令：

./Sstar_iot_cli customer_cmd 0

- **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli customer_cmd 0
cmd_line: customer_cmd 0
[ 2348.094937] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(15),used(1)
[ 2348.103404] cmdid:0   data:hello123     ---------------->预设指令 0 的回馈
OK
[root@sinlinx /]#
```

4.60. customer_cert

客户证书存储。

■ **参数结构：**

type：证书类型，0 CERT，1PRIV_KEY，2ROOT_CA

file_path：证书文件路径

■ **示例：**

存储 cert 证书：

./Sstar_iot_cli customer_cert CERT ./1.cert

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli customer_cert CERT ./1.cert
cmd_line: customer_cert CERT ./1.cert
[ 2368.094937] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(20),used(1)
file:./1.cert   size:121 bytes
[root@sinlinx/]#save cert over.
OK
[root@sinlinx /]#
```

4.61. start_check_alive

启动 SDIO 保活。

■ **参数结构：**

period：单次保活周期，秒

tmo_count：超时计次，需>1

■ **示例：**

启动 sdio 保活，周期 3s，3 次超时设置：

./Sstar_iot_cli start_check_alive 3 3

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli start_check_alive 3 3
cmd_line: start_check_alive 3 3
[ 2367.034657] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(19),used(1)
OK
[ 2397.044656] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(18),used(1)
```

```
[ 2427.054657] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(17),used(1)
[ 2457.064656] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(16),used(1)
[ 2487.074655] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(15),used(1)
[root@sinlinx /]#
```

4.62. direct_trans

双向传输通道 demo 测试，附带发送及回传共用的 1600 buffer。

- **参数结构：**

str：要传输的数据

- **示例：**

使用双向通道发送 abcdefg，预期 118 端重写 buffer 为 abcdefg get_OK：

./Sstar_iot_cli direct_trans abcdefg

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli direct_trans abcdefg
cmd_line: direct_trans abcdefg
[   256.744437] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(12),used(1)
str: abcdefg
get new str: abcdefg get_OK
OK
[root@sinlinx /]#
```

4.63. control_debug

设置 debug 打印掩码。

- **参数结构：**

debug_mask：debug 打印掩码，目前支持的有：RX(bit0)、TX(bit1)、SLEEP(bit2)

- **示例：**

设置 debug mask 为 1，即使能 RX 调试打印：

./Sstar_iot_cli control_debug 1

- **返回值：**

成功 OK，失败 FAIL

- **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli control_debug 1
cmd_line: control_debug 1
[   415.954930] [Sstar_log]:Sstar_wsm_set_control_debug_mask: update debug mask 1.
New control debug: 0x1
OK
[root@sinlinx /]#
```

4.64. force_reboot

强制 118 重启。

■ **参数结构：**

无

■ **示例：**

强制 118 重启：

./Sstar_iot_cli force_reboot

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli force_reboot
cmd_line: force_reboot
[ 1690.604880] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(6),used(1)
[root@sinlinx /]# [ 1690.629628] sleep lock wait cmd over...
[ 1690.633909] [mmc]: *** sw_mci_dump_errinfo(L604): smc 1 err, cmd 53,  RTO !!
[ 1690.641755] [mmc]: *** sw_mci_request_done(L662): In data read operation
[ 1690.649197] [mmc]: found data error, need to send stop command !!
[ 1690.655979] [mmc]: *** sw_mci_send_manual_stop(L575): sdc 1 send stop command failed
[ 1690.664605] [Sstar_log]:Sstar_clear_wakeup_reg: set_wakeup: can't read control register.
[ 1690.673437] [mmc]: *** sw_mci_dump_errinfo(L604): smc 1 err, cmd 53,  RTO !!
[ 1690.681266] [mmc]: *** sw_mci_request_done(L662): In data write operation
[ 1690.688802] [mmc]: found data error, need to send stop command !!
[ 1690.695568] [mmc]: *** sw_mci_send_manual_stop(L575): sdc 1 send stop command failed
[ 1690.704175] [Sstar_log]:Sstar_clear_wakeup_reg: set_wakeup: can't write control register.
[ 1690.713197] [mmc]: *** sw_mci_dump_errinfo(L604): smc 1 err, cmd 52,  RTO !!
[ 1690.721054] [Sstar_log]:Sstar_sdio_xmit_deinit
[ 1690.725806] [Sstar_log]:Sstar_sdio_tx_thread:exit
[ 1690.730868] [Sstar_log]:Sstar_sdio_rev_deinit
[ 1690.735550] [Sstar_log]:Sstar_sdio_rx_thread:exit
[ 1690.740614] Sstar_wsm_fw_sleep ok! new rmmod version...
OK
[root@sinlinx /]# [root@sinlinx /]#
```

4.65. fast_cfg_recv

使能 probe 快速配网接收端。

■ **参数结构：**

enable：使能

time：接收持续时间，0~65535s，0 为永久接收

■ **示例：**

启动 probe 快速配网接收，持续 60s：

./Sstar_iot_cli fast_cfg_recv enable 1 time 60

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli fast_cfg_recv enable 1 time 60
cmd_line: fast_cfg_recv enable 1 time 60
[   124.795053] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(19),used(1)
OK
[root@sinlinx /]# [   166.214467] hostevent->bssid e2:3f:70:c2:f6:a0 ipaddr 903a8c0
[   166.220934] [Sstar_log]:Sstar_sta_connect_event: associated
CTRL-EVENT-STATE-CONNECTED - Connection to e2:3f:70:c2:f6:a0 completed
ip addr:192.168.3.9
[   166.238685] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(17),used(1)
[   166.245210] [Sstar_log]:Sstar_get_mac_address.29:19:0:12:39
[   166.251244] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(16),used(1)
[   166.257818] [Sstar_log]:tcp_port_filter addcnt.1:
[   166.262965] [Sstar_log]:Sstar_open
[root@sinlinx /]#
```

4.66. fast_cfg_send

使能 probe 快速配网发送端。

■ **参数结构：**

enable：使能

time：发送持续时间，0~65535s，0 为永久发送

ssid：配置 ssid 信息，可选项（未设置下默认为当前已连接网络）

pwd：配置 ssid 对应的密码，可选项（未设置下默认为当前已连接网络）

■ **示例：**

启动 probe 快速配网发送，持续 60s（或指定 ssid）：

./Sstar_iot_cli fast_cfg_send enable 1 time 60

或

./Sstar_iot_cli fast_cfg_send enable 1 time 60 ssid wifi_IOT_2.4G pwd Sigmastar

■ **返回值：**

成功 OK，失败 FAIL

■ **执行显示：**

```
[root@sinlinx /]# ./Sstar_iot_cli fast_cfg_send enable 1 time 60
cmd_line: fast_cfg_send enable 1 time 60
[    50.876129] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(11),used(1)
OK
[root@sinlinx /]#
[root@sinlinx /]#
[root@sinlinx /]# ./Sstar_iot_cli fast_cfg_send enable 1 time 60 ssid wifi_IOT_2.4G pwd Sigmastar
cmd_line: fast_cfg_send enable 1 time 60 ssid wifi_IOT_2.4G pwd Sigmastar
[   954.145081] [Sstar_log]:Sstar_sdio_tx_bh:cmd free(8),used(1)
OK
[root@sinlinx /]#
```

4.67. quit

退出守护进程。

- **参数结构：**

无参

- **示例：**

退出守护进程：

quit

- **返回值：**

无

- **执行显示：**

```
[root@sinlinx /]#./Sstar_iot_cli quit
cmd_line: quit
[root@sinlinx /]#./Sstar_iot_cli quit
connect err
```

5. DEMO 调试

工具额外增加了 tcp_demo 发包测试工具（Sstar_iot_supplicant_demo，工具可后接 ip 地址，默认 192.168.3.147），用于仿真连接 AP 成功后的视频数据传输与网络指令控制休眠唤醒。
基本测试方法主要包括如下几部分：

1、 配置虚拟服务端（指令端）
2、 配置虚拟服务端（收包端）
3、 启动测试板测试

依赖工具首选 socketTool V4，或自行上网下载其他相关 socket 工具（sokit 工具测试服务端无法进行数据发送）。

测试前提：

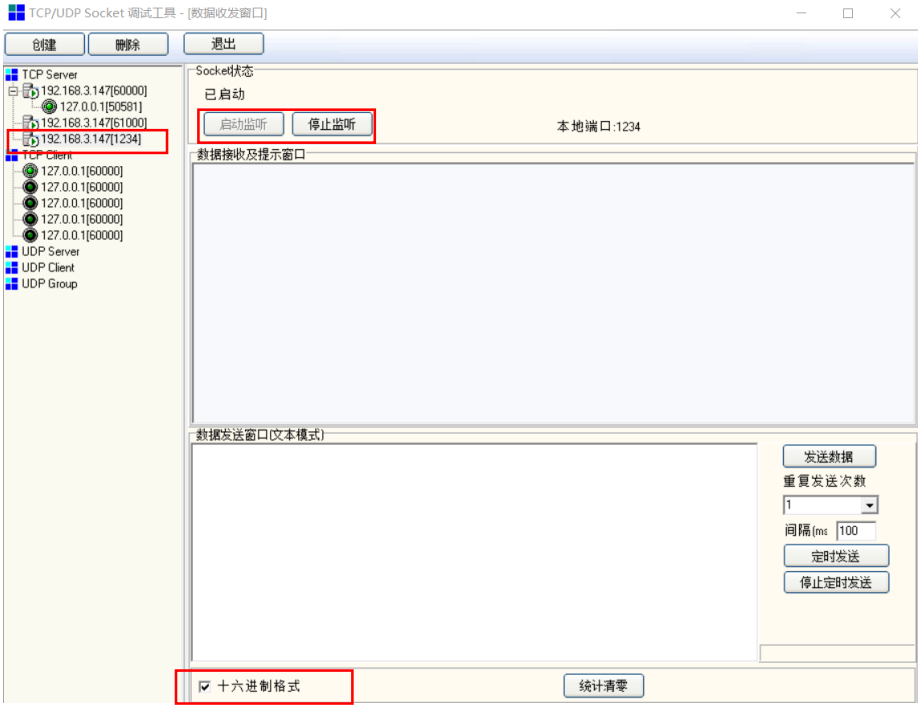
- ✧ 测试板与测试所用笔记本均正常连接到同一网络。

注意事项：

- ✧ 测试指令请确保 IO 为休眠态配置。

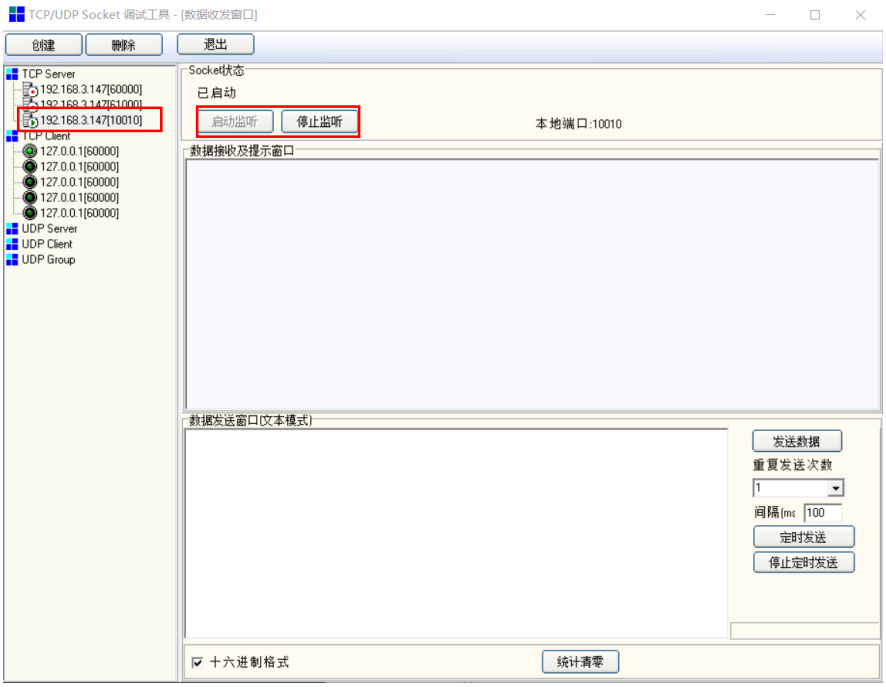
5.1. 配置虚拟服务端（指令端）

笔记本端配置：打开 socketTool V4 工具，选中服务器配置。
选中 TCP Server，新建服务端，配置端口号为 1234（可自行定义），并启动服务端监听。
注：测试休眠唤醒命令，需将左下角十六进制格式取消勾选。



5.2. 配置虚拟服务端（收包端）

流程与 5.1 基本一致，Sstar_iot_supplicant_demo 软件内写死发包端口为 10010，因此配置收包服务端端口号保持一致即可。



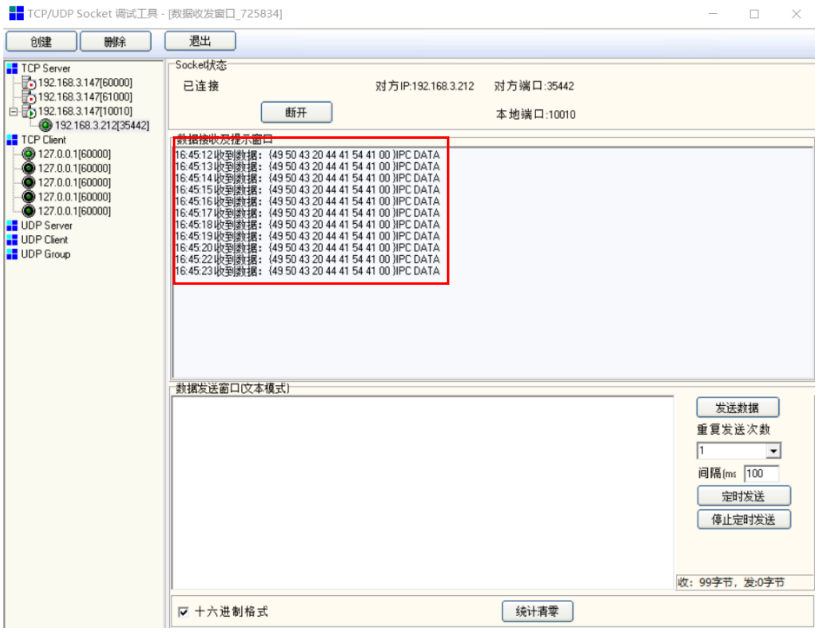
5.3. 启动测试板测试

测试板端配置：

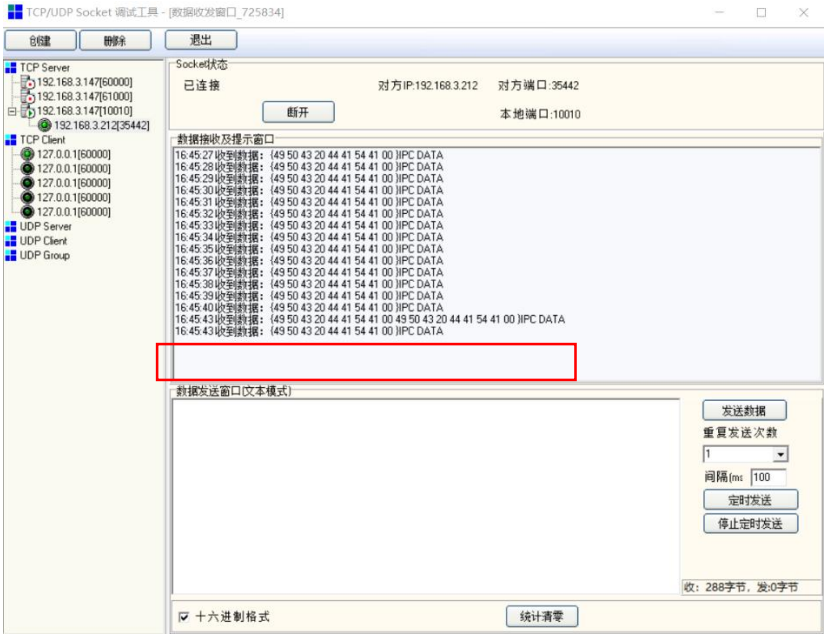
1、测试板上电并正常启动，确认与笔记本连接至同一网络；

2、启动心跳包指令，建立 TCP 连接（详细操作见 6 TCP 心跳包测试）。

上述步骤执行完毕，可以在虚拟服务端（收包端）观察到数据的传输，与下图一致。

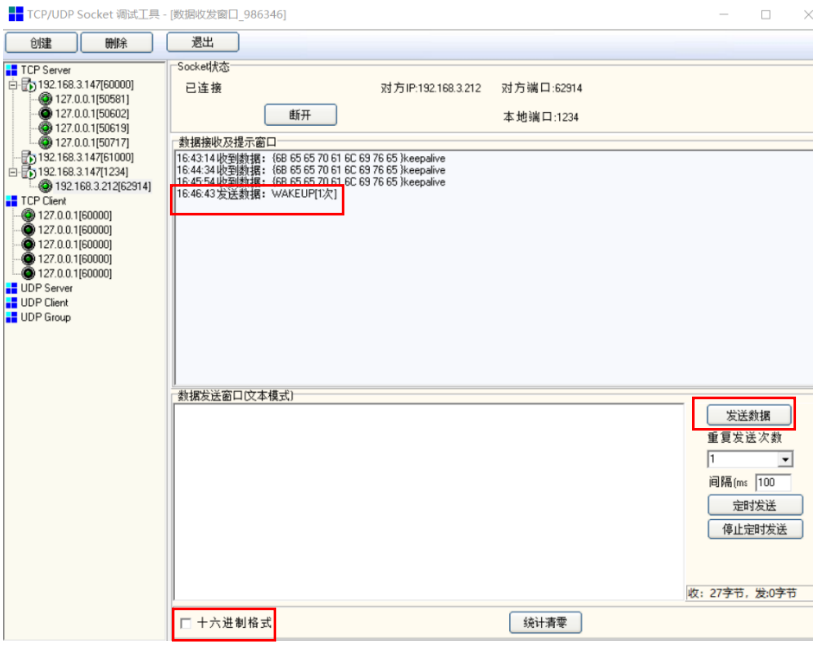


3、IO 控制驱动进入休眠状态，开始进行网络指令测试流程。此时 IPC 属于发送停止态，如下图所示（数据接收窗口底部按回车无数据更新）。

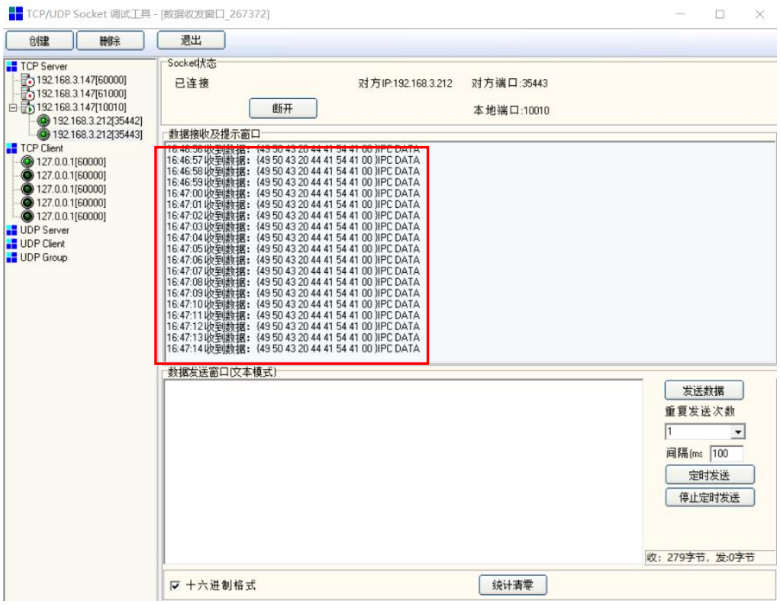


笔记本端控制交互流程：

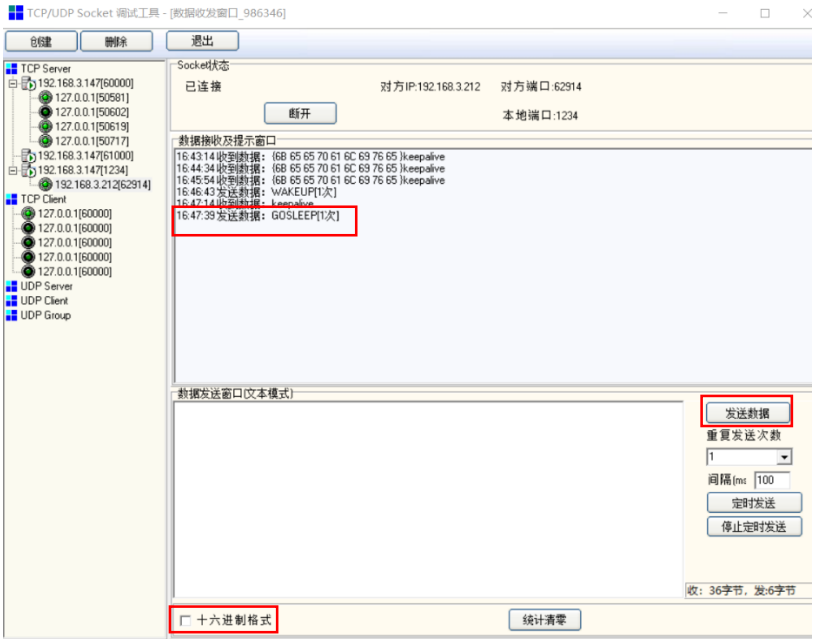
1. 虚拟服务端（指令端）发送唤醒指令 WAKEUP。



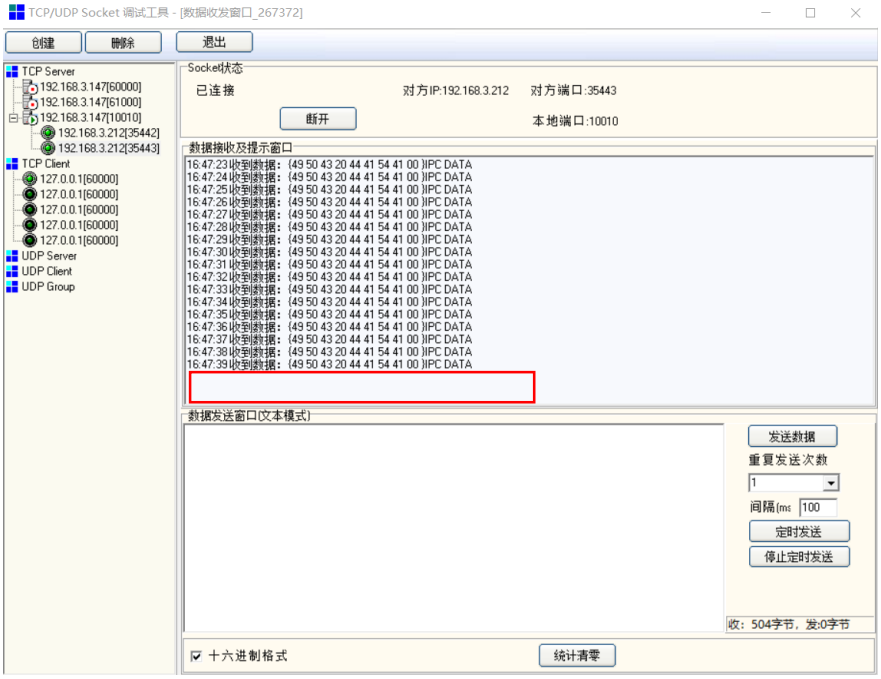
此时，可观察到虚拟服务端（收包端）又有新的数据到来，如下图示。



2. 虚拟服务端（指令端）发送休眠指令 GOSLEEP。



此时 IPC 属于发送停止态，如下图所示（数据接收窗口底部按回车无数据更新）。



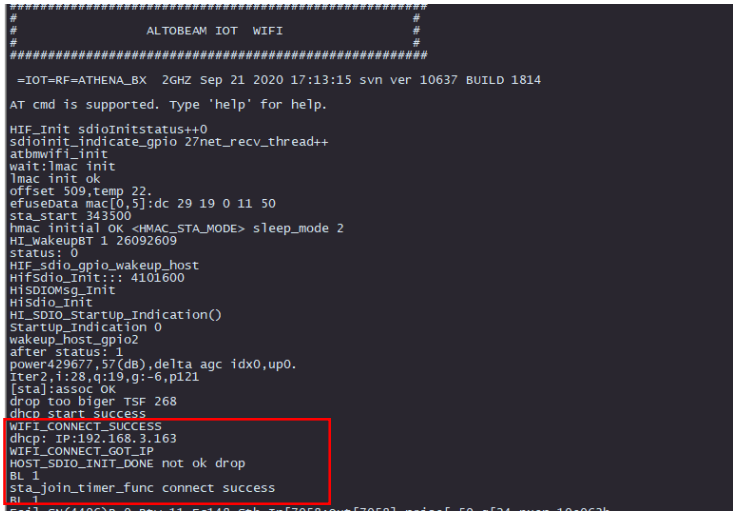
3. 重复上述步骤，测试完成。

6. TCP 心跳包测试

TCP 心跳包测试采用了固件板发 AT 指令的方式，且仅在固件板连接 AP 成功后指令有效。主要用于启动保活及网络命令控制。

■ **测试方法**

A. 固件板烧录包含了心跳包测试功能的固件版本，上电启动，连接指定 AP 成功，如下图示：

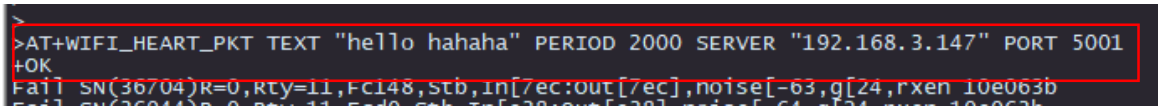


B. 参照 **5 DEMO 调试**所讲述的方法，使用 socketTool V4（或 sokit）工具建立服务端，配置相关 IP 及端口号，并启动监听。

C. 在固件板命令行输入心跳包指令：

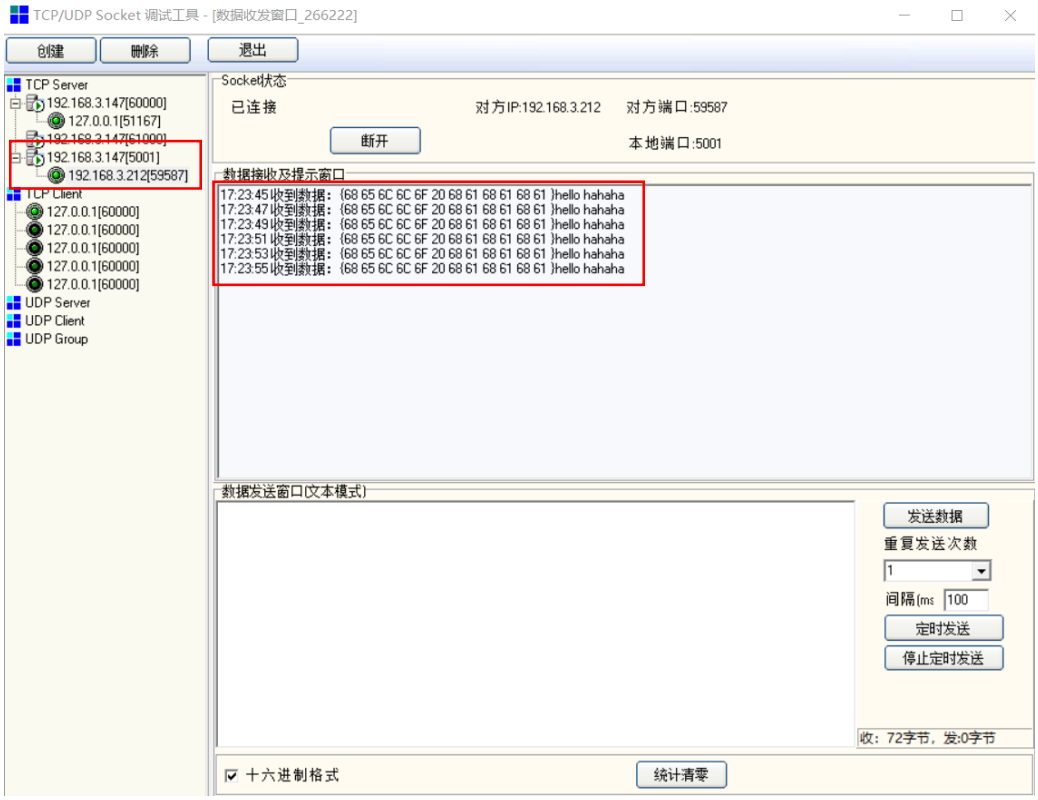
AT+WIFI_HEART_PKT TEXT [text] PERIOD [peri] SERVER [addr] PORT [port]

示例参考如下，设置周期为 2000ms，并连接指定服务端，发送包数据“hello hahaha”。



如上参数在指令执行后，依然可自行变更。

D. 在监听的服务端即可周期性地收到相关数据，如下图示：



7. 客户支持配置

在释放的 118 SDK 中提供了一些适配客制化需求的修改，主要包含了硬件板 IO 配置以及软件流程控制及修改等对外开发的流程代码。
主要可配部分见下述详解。

7.1. 端口过滤功能

端口过滤主要包含了 TCP 端口、UDP 端口以及其他部分通用协议配置（ICMP、ARP、DNS 通过驱动配置）。
在 4 指令详解部分已经包含了过滤器的设置指令，但是其因个数有限无法用于大量端口配置情况。而此处所介绍的方法则显得尤为方便。无论是设置单端口亦或端口域，设置起来都会更加得心应手。
配置过滤流程详见 user_main.c sstarwifi_tcpfilter 与 sstarwifi_udpfilter 接口实现。

7.2. 网络指令控制

网络指令目前已经集成的默认指令为"WAKEUP"和"GOSLEEP"，主要用于联网状态下由远端服务器控制 WIFI 芯片进入唤醒或休眠状态，其指令可自行修改或根据个体需求增减。
网络指令配置流程详见 tcpapp.c handle_tcp_app_rx 接口实现。

7.3. 服务器保活

保活功能实现是完全开放出来的一项客户自制化功能，可基于原始 demo 流程修改适配不同场景下的需求。
保活流程详见 user_main.c sstar_wifi_start_tcp_heart_packet 以及对应的 AT 指令实现 app_wifi_AT_cmd.c AT_TCPHeartPacket 接口实现。

7.4. WIFI 唤醒电平配置

目前唤醒电平配置有两种：高电平唤醒和低电平唤醒。客户可根据具体硬件设计自行配置。
唤醒电平配置详见 hi_sdio.h 宏定义 WAKEUP_LOW_LEVEL。

7.5. UART IO 配置

目前 UART 配置存在两种：IO4-5 搭配和 IO0-1 搭配，修改方法见于 SDK 根目录 config.include。
宏定义 CONFIG_UART_01。

7.6. 休眠唤醒 IO 配置

休眠唤醒 IO 主要包含了两部分：WIFI 芯片本身的休眠唤醒控制 IO 和控制 HOST 加载驱动 IO。
按照现有逻辑，唤醒由 WIFI 芯片唤醒 IO 与网络指令共同参与，满足其中一种条件即可实现芯片唤醒；而休眠流程则全权由 HOST 控制决定休眠时机。
值得注意的一点是 IO8~12 无 IO 唤醒功能。
相关 IO 配置详见 hi_sdio.h 宏定义 WAKEUP_WIFI_GPIO_ID 和 WAKEUP_HOST_GPIO_ID。

7.7. 上电强制枚举配置

原初始方案为 WIFI 芯片第一次上电进行枚举流程是由 IO 控制的（此时无网络指令配置）。为适配客户硬件设计需求，新增了此项功能，即上电后 WIFI 芯片主动进入枚举流程，而不受 IO 高低影响。
上电强制枚举配置详见 hi_sdio.h 宏定义 FORCE_STARTUP_INDICATION。

7.8. WIFI SDIO 状态机流程

此部分主要包含了 SDIO 状态机从起始枚举等待态到枚举态以至最后的休眠态的整个状态条件切换流程。方便客户理解其实现原理以及相关的考究问题。
另，虽然此部分虽然已经开放出来，但无特殊需求无需修改。
主要流程详见 sdio_demo_main.c HI_WakeupBT_Process 接口。