

WS73V100 Linux Wi-Fi、BLE 软件

开发指南

文档版本 10

发布日期 2025-02-10

前言

概述

本文档详细介绍了 WS73V100 Wi-Fi 软件 STA、SoftAp 和 BLE 接口功能以及开发流程。

本文主要为软件开发工程师、测试工程师提供 Wi-Fi 和 BLE 配置开发提供参考。

产品版本

与本文档对应的产品版本如下。

产品名称	产品版本
WS73	V100





读者对象

本文档主要适用以下工程师：

- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示如不可避免则将会导致死亡或严重伤害的具有高等级风险的危害。
 警告	表示如不可避免则可能导致死亡或严重伤害的具有中等级风险的危害。
 注意	表示如不可避免则可能导致轻微或中度伤害的具有低等级风险的危害。
须知	用于传递设备或环境安全警示信息。如不可避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
 说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修改记录

文档版本	发布日期	修改说明
10	2025-01-21	<ul style="list-style-type: none">更新“4 BLE 软件开发”章节内容。删除“自研协议栈 Sample 示例”章节内容。新增“6 缺省编译配置说明”章节内容。
09	2024-10-08	<ul style="list-style-type: none">更新“3.5 SoftAp 模式基础操作示例”章节内容。更新“4.1.2 开发流程”章节内容。
08	2024-08-05	新增“自研协议栈 Sample 示例”章节内容。
07	2024-05-23	<ul style="list-style-type: none">更新“4 BLE 软件开发”章节内容。
06	2024-04-26	新增“4 BLE 软件开发”章节内容。
05	2024-04-17	<ul style="list-style-type: none">更新“3.5 SoftAp 模式基础操作示例”章节内容。

文档版本	发布日期	修改说明
04	2024-03-21	更新“3.5 SoftAp 模式基础操作示例”章节内容。
03	2024-02-26	更新“3.1.1 SDIO Wi-Fi 设备检测”章节内容。
02	2024-01-15	新增“错误码”章节内容。
01	2023-12-11	第一次正式版本发布。 更新“5 蓝牙业务开发指南”章节内容。
00B04	2023-12-01	<ul style="list-style-type: none">更新“3.4 STA 模式基础操作示例”章节内容。更新“3.5 SoftAp 模式基础操作示例”章节内容。更新“3.7 Sta/P2P 共存基础操作示例”章节内容。
00B03	2023-11-07	更新“3.1.2 USB Wi-Fi 设备检测”章节内容。
00B02	2023-10-24	<ul style="list-style-type: none">新增“2 设备检测”章节内容。更新“3.4 STA 模式基础操作示例”章节内容。更新“3.5 SoftAp 模式基础操作示例”章节内容。更新“3.6 P2P(Wi-Fi Peer-to-Peer)模式基础操作示例”章节内容。新增“3.7 Sta/P2P 共存基础操作示例”章节内容。新增“3.8 Sta&Softap 共存基础操作示例”章节内容。
00B01	2023-08-25	第一次临时版本发布。

目 录

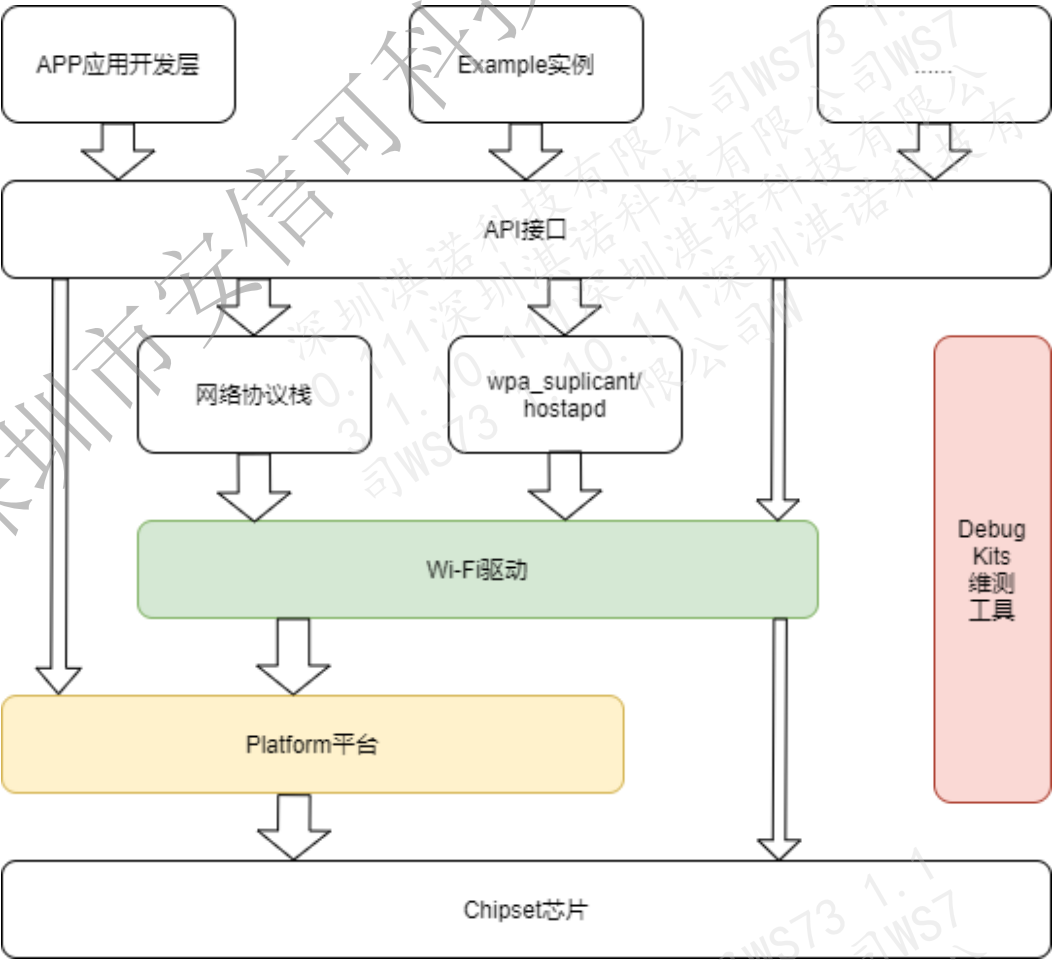
前言	i
1 概述.....	1
2 设备检测.....	3
3 Wi-Fi 业务开发指南.....	4
3.1 Wi-Fi 设备检测	4
3.1.1 SDIO Wi-Fi 设备检测	4
3.1.2 USB Wi-Fi 设备检测.....	5
3.2 加载驱动.....	5
3.3 载入工具.....	7
3.4 STA 模式基础操作示例	9
3.5 SoftAp 模式基础操作示例	11
3.6 P2P(Wi-Fi Peer-to-Peer)模式基础操作示例	12
3.7 Sta/P2P 共存基础操作示例.....	14
3.8 Sta&Softap 共存基础操作示例.....	14
4 BLE 软件开发	16
4.1 BLE 功率档位定制	16
4.1.1 概述.....	16
4.1.2 开发流程.....	17
5 蓝牙业务开发指南	18
5.1 蓝牙设备检测	18
5.1.1 SDIO 蓝牙设备检测	18
5.1.2 USB 蓝牙设备检测.....	19

5.2 加载驱动.....	19
5.3 载入工具.....	20
5.4 Bluez 蓝牙业务基础操作示例.....	22
6 缺省编译配置说明	25
A 缩略语.....	37

1 概述

WS73V100 通过 API（Application Programming Interface）面向开发者提供 Wi-Fi 功能的开发和应用接口，包括芯片初始化、资源配置、Station 创建和配置、扫描、关联以及去关联、状态查询等一系列功能，框架结构如图 1-1 所示。

图1-1 WS73 API 接口控制流框图



各功能模块说明如下：

- APP 应用开发层：用户基于 API 接口的二次开发。
- Example 示例：SDK 提供的功能开发示例。
- API 接口：提供基于 SDK 的通用接口。
- wpa_supplicant（含 hostapd）：Wi-Fi 管理模块。
- 网络协议栈：操作系统网络转发协议栈。
- Wi-Fi 驱动：802.11 协议实现模块。
- Platform 平台：芯片驱动公共模块（SDIO、USB、消息通信）。
- Debug Kits 维测工具：主要用于问题定位时日志查看&收集。

2 设备检测

SDIO Wi-Fi 设备检测

使用 SDIO Wi-Fi 前，需要在 Host 端检测到 SDIO 设备：

- 如果 Wi-Fi 芯片是默认上电的，则 Linux 启动时会检测到 SDIO 设备，显示“mmc1: new SDIO card at address 0001”打印信息，说明 SDIO 检测成功。

可以通过执行“cat /proc/mci/mci_info”命令查看是否检测到 SDIO 设备，如图 2-1 所示。

图2-1 SDIO 设备示例

```
~ # cat /proc/mci/mci_info
MCI0: plugged_disconnected
MCI1: plugged_connected
      Type: SDIO card Mode: HS
      Speed Class: Class 0
      Uhs Speed Grade: Less than 10MB/sec(0h)
      Host work clock: 25MHz
      Card support clock: 25MHz
      Card work clock: 25MHz
      Card error count: 0
MCI2: invalid
```

说明

使用的 SDIO 通路以内核配置为准，当前示例以 mmc1 做为 SDIO Wi-Fi 连接通路

3

Wi-Fi 业务开发指南

3.1 Wi-Fi 设备检测

3.2 加载驱动

3.3 载入工具

3.4 STA 模式基础操作示例

3.5 SoftAp 模式基础操作示例

3.6 P2P(Wi-Fi Peer-to-Peer)模式基础操作示例

3.7 Sta/P2P 共存基础操作示例

3.8 Sta&Softap 共存基础操作示例

3.1 Wi-Fi 设备检测

在完成芯片上电后，驱动加载实现对芯片寄存器的初始配置、SDIO、USB 识卡等基础操作。

3.1.1 SDIO Wi-Fi 设备检测

使用 SDIO Wi-Fi 前，需要在 Host 端检测到 SDIO 设备：

如果 Wi-Fi 芯片是默认上电的，则 Linux 启动时会检测到 SDIO 设备，如打印 “mmc1: new SDIO card at address 0001” 打印信息，说明 SDIO 探卡成功。

可以通过执行 “cat /proc/mci/mci_info” 命令查看是否检测到 SDIO 设备，如图 3-1 所示。

图3-1 SDIO 设备示例

```
~ # cat /proc/mci/mci_info
MCI0: plugged_disconnected
MCI1: plugged_connected
      Type: SDIO card Mode: HS
      Speed Class: Class 0
      Uhs Speed Grade: Less than 10MB/sec(0h)
      Host work clock: 25MHz
      Card support clock: 25MHz
      Card work clock: 25MHz
      Card error count: 0
MCI2: invalid
```

说明

- 使用的 sdio 通路以内核配置为准，当前示例以 mmc1 做为 sdio Wi-Fi 连接通路。
- mmc 为 linux 内核实现的驱动管理系统。

3.1.2 USB Wi-Fi 设备检测

USB Wi-Fi 在使用前，需要在 Host 端先检测到 USB 设备：

- 如果主控平台支持 USB 功能且是 build-in 模式，则 Linux 启动时会检测到 USB 设备，显示 “xHCI Host Controller” 打印信息，说明探测到 USB 设备。
- 如果主控平台支持 USB 功能且需要加载驱动，则需要加载 USB 驱动后，再加载 WS73 驱动。

可以通过执行 “lsusb” 命令查看是否检测到 USB 设备，如图 3-2 所示。

图3-2 USB 设备示例

```
~ # lsusb
Bus 001 Device 001: ID 1d6b:0002
Bus 001 Device 006: ID ffff:3733 ← USB Wi-Fi
Bus 002 Device 001: ID 1d6b:0003
```

说明

device ID 为全 f 表示单板经过 efuse 烧写，属于正常现象。

3.2 加载驱动

步骤 1 将 WS73 所需的文件放至单板对应目录下，如表 3-1 所示。

表3-1 驱动加载所需文件及存放路径

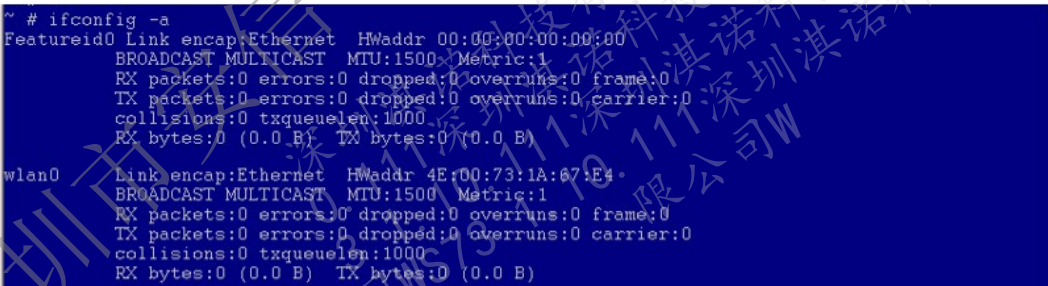
文件	存放路径（默认）
firmware/ws73.bin	/etc/ws73
firmware/wifi_calib.bin	/etc/ws73
firmware/btc_calib.bin	/etc/ws73
firmware/wow.bin	/etc/ws73
output/ws73_cfg.ini	/etc
output/plat_soc.ko	任意目录，如：/usr/komod
output/wifi_soc.ko	任意目录，如：/usr/komod

步骤 2 依次加载 plat_soc.ko wifi_soc.ko。

```
$ insmod plat_soc.ko
$ insmod wifi_soc.ko
```

步骤 3 执行 ifconfig -a 命令，若出现 wlan0 网口，则说明 Wi-Fi 驱动初始化成功。

图3-3 wlan0 初始化成功



----结束

📖 说明

- 若内核选项 CONFIG_CFG80211=m，则需提前加载内核编译的 cfg80211.ko；若内核选项 CONFIG_CFG80211=y，则无需加载 cfg80211.ko。
- 编译 Host 侧驱动时，可配置 bin 文件和 ini 文件存放的目录，当拷贝 bin 文件和 ini 文件时，需要保证拷贝路径与 Host 配置路径一致。

3.3 载入工具

开源组件 wpa_supplicant 和 hostapd 均不在 sdk 发布范围，以下流程中的 wpa_supplicant 和 hostapd 以及 lib 库均为下载开源组件编译产生。

步骤 1 将 SDK 中 output/wifi_service 目录下 “libnl-3.so.200.26.0” 和 “libnl-genl-3.so.200.26.0” 拷贝至单板的 /lib 目录下，并创建软链接。

```
ln -s libnl-3.so.200.26.0 libnl-3.so.200
ln -s libnl-genl-3.so.200.26.0 libnl-genl-3.so.200
```

步骤 2 将 SDK 中 output/wifi_service 目录下 “wpa_supplicant” “wpa_cli” “hostapd” 拷贝至单板的 /bin 目录下，添加可执行权限。

```
chmod a+x wpa_supplicant
chmod a+x wpa_cli
chmod a+x hostapd
```

步骤 3 创建 wpa_supplicant.conf 文件。

wpa_supplicant.conf 文件是启动 wpa_supplicant 进程时需要使用到的配置文件。在单板上新建一个该文件，文件内容如下：

```
ctrl_interface=/etc/Wireless/wpa_supplicant
update_config=1
```

步骤 4 创建 p2p_supplicant.conf 文件。

p2p_supplicant.conf 文件是启动 wpa_supplicant P2P 功能需的配置文件。在单板上新建一个该文件，文件内容如下：

```
ctrl_interface=/etc/Wireless/wpa_supplicant
update_config=1
device_name=p2p_test
device_type=10-0050F204-5
config_methods=display push_button keypad virtual_push_button physical_display
p2p_go_he=1
p2p_group_idle=10
p2p_no_group_iface=1
```

步骤 5 创建 hostapd.conf 文件。

hostapd.conf 文件是启动 hostapd 功能需的配置文件。在单板上新建一个该文件，文件内容如下：

```
interface=wlan0
```

```

driver=nl80211
ctrl_interface=/var/hostapd
ssid=AP_Test
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=12345678
wpa_pairwise=CCMP
channel=6
hw_mode=g
ieee80211n=1
ht_capab=[SHORT-GI-20]

```

----结束

表3-2 开源组件文件及存放路径一览

文件	默认存放路径
wifi_service/libnl-3.so.200.26.0	/lib
wifi_service/libnl-genl-3.so.200.26.0	/lib
wifi_service/libnl-route-3.so.200	/lib
wifi_service/libcrypto.so.1.1	/lib
wifi_service/libssl.so.1.1	/lib
wifi_service/wpa_supplicant	任意路径，如/bin 目录
wifi_service/wpa_cli	任意路径，如/bin 目录
wifi_service/hostapd	任意路径，如/bin 目录
wpa_supplicant.conf	任意路径，如/etc/Wireless 目录，需和启动 wpa_supplicant 命令中指定 conf 参数的路径一致，wpa_supplicant -i wlan0 -D nl80211 -c /etc/Wireless/wpa_supplicant.conf &
p2p_supplicant.conf	任意路径，如/etc/Wireless 目录，需和启动 wpa_supplicant 命令中指定 conf 参数的路径一致，wpa_supplicant -ip2p0 -D nl80211 -c /etc/Wireless/p2p_supplicant.conf &
hostapd.conf	任意路径，如/etc/Wireless 目录，需和启动 hostapd 命令中指定 conf 参数的路径一致，hostapd

文件	默认存放路径
	/etc/Wireless/hostapd.conf &

3.4 STA 模式基础操作示例

连接 AP 需要启动 wpa_supplicant 进程。Linux 采用 wpa_supplicant 进行 Wi-Fi 连接过程，wpa_supplicant 是开源代码，包含了 WEP、WPA/WPA2、WPA-PSK/WPA2-PSK、WAPI、WPS、P2P、EAP 等协议。

步骤 1 加载驱动。

```
$ insmod plat_soc.ko
$ insmod wifi_soc.ko
```

步骤 2 启动 wpa_supplicant。命令如下：

```
wpa_supplicant -iwlan0 -Dnl80211 -c/etc/Wireless/wpa_supplicant.conf &
```

- -iwlan0：使用 wlan0 网口。
- -Dnl80211：使用 cfg80211 接口（用户态的接口为 libnl，内核中为 cfg80211）。
- -c/etc/Wireless/wpa_supplicant.conf：指定 wpa_supplicant 的配置文件。

说明

执行 wpa_supplicant 命令后，可使用 ps 命令查看 wpa_supplicant 进程是否存在。若存在，则启动正常；若不存在，则通过提高 wpa_supplicant 打印等级，从 log 中查找问题：

```
wpa_supplicant -iwlan0 -Dnl80211 -c/etc/Wireless/wpa_supplicant.conf -ddd &
```

步骤 3 启动 wpa_cli。命令如下：

```
wpa_cli -iwlan0 -p/etc/Wireless/wpa_supplicant
```

说明

“wpa_cli”命令执行成功会出现 “>” 符号。如果出现 “Could not connect to wpa_supplicant - re-trying”，则表示 “wpa_cli” 不能与 wpa_supplicant 建立 socket 连接，此时需要检查：

- wpa_supplicant 进程是否还存在。
- 查看是否有 “/etc/Wireless/wpa_supplicant/wlan0”。
- 检查 wpa_supplicant.conf 文件中 “ctrl_interface” 是否为 “/etc/Wireless/wpa_supplicant”。
- 确认 “/tmp” 目录是否可以写入权限（“wpa_supplicant” 进程启动后，会在 “/tmp” 目录下创建临时文件）。

步骤 4 执行扫描。

1. 在 ">" 后执行 "scan" 命令，驱动扫描流程。
2. 收到 "CTRL-EVENT-SCAN-RESULTS" 后，执行 "scan_results"，获得扫描结果。

```
> scan
> scan_results
```

步骤 5 执行连接。

1. 在 ">" 后执行 "add_network" 命令，该命令会返回一个数字，表示添加的网络 id 号。
2. 执行 "set_network 网络 id ssid "AP 的 SSID" " 命令，配置网络 ID 的 SSID。
3. 执行 "set_network 网络 id key_mgmt NONE" 命令，配置网络 ID 的加密方式。
4. 执行 "select_network 网络 id" 命令，选择并网络 ID 进行连接。
5. 收到 "CTRL-EVENT-CONNECTED" 表示连接成功。

```
> add_network
> set_network 0 ssid "sta-test"
> set_network 0 key_mgmt WPA-PSK
> set_network 0 psk "12345678"
> select_network 0
> q
```

图3-4 STA 链接过程



```
Interactive mode
> scan
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>CTRL-EVENT-NETWORK-NOT-FOUND
scan_results
> bssid / frequency / signal level / flags / ssid
5e:c0:a0:8a:72:4c 2437 -20 [WPA2-PSK-CCMP][ESS]
> add_network
0
> set_network 0 ssid
OK
> set_network 0 key_mgmt WPA-PSK
OK
> set_network 0 psk "12345678"
OK
> select_network 0
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
wlan0: Trying to associate with 5e:c0:a0:8a:72:4c (SSID=' ' freq=2437 MHz)
<3>Trying to associate with 5e:c0:a0:8a:72:4c (SSID=' ' freq=2437 MHz)
wlan0: Associated with 5e:c0:a0:8a:72:4c
wlan0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
<3>Associated with 5e:c0:a0:8a:72:4c
<3>CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
wlan0: WPA: Key negotiation completed with 5e:c0:a0:8a:72:4c [PTK=CCMP GTK=CCMP]
<3>WPA: Key negotiation completed with 5e:c0:a0:8a:72:4c [PTK=CCMP GTK=CCMP]
wlan0: CTRL-EVENT-CONNECTED - Connection to 5e:c0:a0:8a:72:4c completed [id=0 id_str=]
<3>CTRL-EVENT-CONNECTED - Connection to 5e:c0:a0:8a:72:4c completed [id=0 id_str=]
status
```

步骤 6 获取 IP。

退出 wpa_cli 后，启动 dhcp client 端，查看 wlan0 获取的 ip 地址


```
$ udhcpc -i wlan0  
$ ifconfig wlan0
```

步骤 7 卸载驱动。

1. 退出 wpa_supplicant、udhcpc 等应用
2. 卸载驱动:

```
$ killall wpa_supplicant  
$ killall udhcpc  
$ rmmod wifi_soc.ko  
$ rmmod plat_soc.ko
```

----结束

3.5 SoftAp 模式基础操作示例

步骤 1 加载驱动。

```
$ insmod plat_soc.ko  
$ insmod wifi_soc.ko
```

步骤 2 启动 hostapd, 命令如下:

```
$ hostapd /etc/Wireless/hostapd.conf &
```

步骤 3 启动 udhcpd, 命令如下:

```
$ ifconfig wlan0 192.168.49.1  
$ udhcpd -S /etc/Wireless/udhcpd.conf &
```

说明

1. 使用 wlan1 端口起 softap 时, 需要将 hostapd.conf 和 udhcpd.conf 配置文件中 interface 的值改为 wlan1, 否则可能会导致 softap 无法启动。
2. 如果 softap 需要支持 11ax wifi6 协议, 请在 配置文件 hostapd.conf 中 ieee80211n=1 后增加一行 ieee80211ax=1。
3. 请使用 hostapd 起 softap, 使用 killall hostapd 方式关闭 softap。

步骤 4 卸载驱动。

1. 退出 hostapd、udhcpd 等应用。

```
$ killall udhcpd  
$ killall hostapd
```

2. 卸载驱动。

```
$ rmmod wifi_soc.ko
```

```
$ rmmod plat_soc.ko
```

----结束

3.6 P2P(Wi-Fi Peer-to-Peer)模式基础操作示例

步骤 1 加载驱动。

```
$ insmod plat_soc.ko  
$ insmod wifi_soc.ko
```

步骤 2 启动 wpa_supplicant, 命令如下:

```
$ wpa_supplicant -ip2p0 -Dnl80211 -c/etc/Wireless/p2p_supplicant.conf &
```

- -ip2p0: 使用 p2p0 网口。
- -Dnl80211: 使用 cfg80211 接口 (用户态的接口为 libnl, 内核中为 cfg80211)。
- /etc/Wireless/p2p_supplicant.conf: wpa_supplicant 使用 p2p 功能的配置文件 (必须保证该文件已经存在)。

说明

执行 wpa_supplicant 命令后, 可使用 ps 命令查看 wpa_supplicant 进程是否存在。若存在, 则启动正常; 若不存在, 则通过提高 wpa_supplicant 打印等级, 从 log 中查找问题:

```
wpa_supplicant -ip2p0 -Dnl80211 -c/etc/Wireless/p2p_supplicant.conf -ddd &
```

步骤 3 启动 wpa_cli, 命令如下:

```
wpa_cli -ip2p0 -p/etc/Wireless/wpa_supplicant
```

说明

wpa_cli 命令执行成功会出现 ">" 符号。如果出现 "Could not connect to wpa_supplicant -retrying", 则表示 wpa_cli 不能与 wpa_supplicant 建立 socket 连接, 此时需要检查:

1. wpa_supplicant 进程是否还存在。
2. 查看是否有/etc/Wireless/wpa_supplicant/p2p0。
3. 检查 wpa_supplicant.conf 文件中是否是 ctrl_interface=/etc/Wireless/wpa_supplicant。

步骤 4 发现设备。

1. 在 ">" 后执行 "p2p_find" 命令, 驱动开始发现其他 p2p 设备。
2. 收到 "P2P-DEVICE-FOUND" 后, 获得发现 p2p 设备信息。

步骤 5 连接设备。

1. 在 “>” 后执行 “p2p_connect XX:XX:XX:XX:XX:XX pbc persistent go_intent=15 freq=2412” 命令，其中 XX:XX:XX:XX:XX:XX 为要关联 p2p 设备的 MAC 地址；go_intent 的值取决于预期作为 GO(值较大，比如 14)或 GC(值较小，比如 1)；freq 的值表示工作信道的频率。
2. 在要关联的设备上选择接受关联后，双方开启关联流程。
3. 双方根据协商结果确定 GO/GC 的角色，收到 “CTRL-EVENT-CONNECTED” 表示连接成功。

图3-5 连接 p2p 示例

```
P2P-FIND-STOPPED
> OK
<3>P2P-FIND-STOPPED
P2P-GO-NEG-SUCCESS role=client freq=2462 ht40=0 peer_dev=3a:bc:1a:cb:8d:90 peer_iface=3a:bc:1a:cb:8d:90 wps_method=PBC
<3>P2P-OK
GO-NEG-SUCCESS role=client freq=2462 ht40=0 peer_dev=3a:bc:1a:cb:8d:90 peer_iface=3a:bc:1a:cb:8d:90 wps_method=PBC
rtkill: Cannot open RFKILL control device
p2p-p2p0-0: WPS-PBC-ACTIVE
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
p2p-p2p0-0: Trying to associate with 3a:bc:1a:cb:8d:90 (SSID='DIRECT-ra-MeizunoteM1' freq=2462 MHz)
p2p-p2p0-0: Associated with 3a:bc:1a:cb:8d:90
p2p-p2p0-0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
p2p-p2p0-0: CTRL-EVENT-EAP-STARTED EAP authentication started
p2p-p2p0-0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=14122 method=1
p2p-p2p0-0: CTRL-EVENT-EAP-METHOD EAP vendor 14122 method 1 (WSC) selected
p2p-p2p0-0: WPS-CRED-RECEIVED
p2p-p2p0-0: WPS-SUCCESS
P2P-GROUP-FORMATION-SUCCESS
<3>P2P-GROUP-FORMATION-SUCCESS
p2p-p2p0-0: CTRL-EVENT-EAP-FAILURE EAP authentication failed
p2p-p2p0-0: CTRL-EVENT-DISCONNECTED bssid=3a:bc:1a:cb:8d:90 reason=3 locally_generated=1
p2p-p2p0-0: Trying to associate with 3a:bc:1a:cb:8d:90 (SSID='DIRECT-ra-MeizunoteM1' freq=2462 MHz)
p2p-p2p0-0: Associated with 3a:bc:1a:cb:8d:90
p2p-p2p0-0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
p2p-p2p0-0: WPA: Key negotiation completed with 3a:bc:1a:cb:8d:90 [PTK=CCMP GTK=CCMP]
p2p-p2p0-0: CTRL-EVENT-CONNECTED - Connection to 3a:bc:1a:cb:8d:90 completed [id=0 id_str=]
<3>P2P-GROUP-STARTED p2p-p2p0-0 client ssid="DIRECT-ra-MeizunoteM1" freq=2462 psk=0e4535780efe25c2053dce82de075d10c7380bfcd75f67794f46517566bb go_dev_addr=3a:bc:1a:cb:8d:90 [PERSISTENT]
P2P-GROUP-STARTED p2p-p2p0-0 client ssid="DIRECT-ra-MeizunoteM1" freq=2462 go_dev_addr=3a:bc:1a:cb:8d:90 [PERSISTENT]
```

步骤 6 获取 IP 地址。

1. 在 “>” 后输入 “q”，退出 wpa_cli 程序。
2. 如果设备作为 GO，则启动 dhcp server 端，命令如下：
\$ ifconfig p2p0 192.168.49.1
\$ udhcpd -S /etc/Wireless/udhcpd_go.conf
3. 如果设备作为 GC，则启动 dhcp client 端，命令如下：
\$ udhccp -i p2p0

配置 IP 后，通过 ping 网关判断 IP 地址配置是否正确。

步骤 7 卸载驱动。

1. 进入 wpa_cli 程序，在 “>” 后执行 “p2p_group_remove *” 命令，释放小组资源。
2. 退出 wpa_supplicant、udhcpd、udhccp 等应用。
\$ killall udhcpd
\$ killall wpa_supplicant
\$ killall udhccp

3. 卸载驱动。

```
$ rmmod wifi_soc.ko
```

```
$ rmmod plat_soc.ko
```

----结束

3.7 Sta/P2P 共存基础操作示例

步骤 1 加载驱动。

```
$ insmod plat_soc.ko
```

```
$ insmod Wi-Fi_soc.ko
```

步骤 2 参考“3.4 STA 模式基础操作示例”章节启动 STA 并连接 AP。

步骤 3 参考“3.6 P2P(Wi-Fi Peer-to-Peer)模式基础操作示例”章节启动 P2P 并作为 GO 或 GC，建立 P2P 连接。

步骤 4 卸载驱动。

1. 退出 wpa_supplicant、udhcpd、udhcpc 等应用。

```
$ killall udhcpd
```

```
$ killall wpa_supplicant
```

```
$ killall udhcpc
```

2. 卸载驱动。

```
$ rmmod wifi_soc.ko
```

```
$ rmmod plat_soc.ko
```

----结束

3.8 Sta&Softap 共存基础操作示例

步骤 1 加载驱动。

```
$ insmod plat_soc.ko
```

```
$ insmod wifi_soc.ko
```

步骤 2 参考“3.4 STA 模式基础操作示例”章节启动 STA 并连接 AP。

步骤 3 参考“3.5 SoftAp 模式基础操作示例”章节启动 SoftAp。

📖 说明

STA 和 SoftAp 共存时，SoftAp 的信道会跟随 STA 的信道，即 SoftAp 将和 STA 工作在相同的信道上。

步骤 4 卸载驱动。

1. 退出 wpa_supplicant、udhcpd、udhcpc 等应用。

```
$ killall udhcpd
$ killall hostapd
$ killall wpa_supplicant
$ killall udhcpc
```

2. 卸载驱动。

```
$ rmmod wifi_soc.ko
$ rmmod plat_soc.ko
```

----结束

4 BLE 软件开发

须知

WS73 提供了开源蓝牙和闭源蓝牙两种蓝牙业务开发方式。

WS73 SDK 默认编译生成的 `ble_soc.ko` 为开源蓝牙，其中的开源组件不在 SDK 的发布范围。

闭源蓝牙编译参考文档《WS73V100 BLE 和 Wi-Fi 小型化开发指南》，若使用闭源蓝牙，请申请对应主控使用的 `lib` 库（SDK 中作为参考在 `application` 文件夹下提供了部分主控型号和工具链的蓝牙工具及静态库）。

4.1 BLE 功率档位定制

4.1 BLE 功率档位定制

4.1.1 概述

WS73V100 支持 8 个档位：-6、-2、2、6、10、14、16、20。

支持通过 `ws73_cfg.ini` 的 `bt_maxpower` 配置最高功率档位：

`bt_maxpower=7`，（默认值）表示有 8 个档位，分别是 -6、-2、2、6、10、14、16、20

`bt_maxpower=5`，表示有 6 个档位，分别是 -6、-2、2、6、10、14

4.1.2 开发流程

根据 BLE 官方协议要求，通过广播下发的 tx_power 值，选择 \leq tx_power 的最大档位。
tx_power 在广播参数中下发，下发需遵循蓝牙协议。

- 当 ws73_cfg.ini 的 bt_maxpower=7 时：
tx_power=10，则选中第 4 档发送广播的功率为 10dbm。
tx_power=9，则选中第 3 档发送广播的功率为 6dbm。
tx_power=0x7F，则选中最高档位发送广播的功率为 20dbm。
- 当 ws73_cfg.ini 的 bt_maxpower=3 时：
tx_power=10，则选中第 4 档发送广播的功率为 6dbm。
tx_power=9，则选中第 3 档发送广播的功率为 6dbm。
tx_power=0x7F，则选中最高档位发送广播的功率为 6dbm。

5

蓝牙业务开发指南

5.1 蓝牙设备检测

5.2 加载驱动

5.3 载入工具

5.4 Bluez 蓝牙业务基础操作示例

5.1 蓝牙设备检测

5.1.1 SDIO 蓝牙设备检测

SDIO 蓝牙在使用，需要在 Host 端检测到 SDIO 设备：

- 如果蓝牙芯片是默认上电的，则 Linux 启动时会检测到 SDIO 设备，显示 “mmc1: new SDIO card at address 0001” 打印信息，说明 SDIO 检测成功。
- 如果蓝牙芯片是默认不上电的，则需要在蓝牙驱动中添加上电 SDIO 检测相关流程，再继续执行。

平台可以通过执行 “cat /proc/mci/mci_info” 命令查看是否检测到 SDIO 设备，如图 5-1 所示。

图5-1 SDIO 设备示例

```

~ # cat /proc/mci/mci_info
MCI0: plugged_disconnected
MCI1: plugged_connected
      Type: SDIO card Mode: HS
      Speed Class: Class 0
      Uhs Speed Grade: Less than 10MB/sec(0h)
      Host work clock: 50MHz
      Card support clock: 50MHz
      Card work clock: 50MHz
      Card error count: 0
MCI2: invalid
.. ..

```

说明

- 使用的 sdio 通路以内核配置为准，当前示例以 mmc1 做为 sdio BT 连接通路。
- mmc 为 linux 内核实现的驱动管理系统。

5.1.2 USB 蓝牙设备检测

USB 蓝牙在使用前，需要在 Host 端先检测到 USB 设备：

- 如果主控平台支持 USB 功能且是 build-in 模式，则 Linux 启动时会检测到 USB 设备，显示 “xHCI Host Controller” 打印信息，说明探测到 USB 设备。
- 如果主控平台支持 USB 功能且需要加载驱动，则需要加载 usb 驱动后，再加载 WS73 驱动。

平台可以通过执行 “lsusb” 命令查看是否检测到 USB 设备，如图 5-2 所示。

图5-2 USB 设备示例

```

~ # lsusb
Bus 001 Device 001: ID 1d6b:0002
Bus 001 Device 006: ID ffff:3733 ← USB BT
Bus 002 Device 001: ID 1d6b:0003
~ ..

```

说明

device ID 为全 f 表示单板经过 efuse 烧写，属于正常现象。

5.2 加载驱动

步骤 1 将 WS73 所需的文件放至单板对应目录下，如表 5-1 所示。

表5-1 驱动加载所需文件及存放路径

文件	存放路径（默认）
firmware/ws73.bin	/etc/ws73
firmware/wifi_calib.bin	/etc/ws73
firmware/btc_calib.bin	/etc/ws73
firmware/wow.bin	/etc/ws73
output/ws73_cfg.ini	/etc
output/plat_soc.ko	任意目录，如：/usr/komod
output/wifi_soc.ko	任意目录，如：/usr/komod

步骤 2 依次加载 plat_soc.ko ble_soc.ko。

```
$ insmod plat_soc.ko
$ insmod ble_soc.ko
```

步骤 3 串口打印如图 5-3 所示，则说明蓝牙驱动初始化成功。

图5-3 蓝牙驱动加载成功

```
komod # [HCC] enter hci_bt_open
[HCC] hci_bt_setup
[HCC] hci_bt_flush[HCC] hci_bt_close

----结束
```

5.3 载入工具

开源组件 dbus-daemon、bluetoothd、bluetoothctl 均不在 sdk 发布范围，以下流程中的文件及 lib 库均为下载开源组件编译产生。

步骤 1 将如下编译生成库文件拷贝至单板的/lib 目录下：

```
libglib-2.0.so.0
libexpat.so.1
libpcre.so.1
libdbus-1.so.3
libintl.so.8
libreadline.so.8
```

步骤 2 将 bluetoothd、bluetoothctl、dbus-daemon 拷贝至单板的/bin 目录下，并修改为可执行权限：

```
chmod a+x bluetoothd
chmod a+x bluetoothctl
chmod a+x dbus-daemon
```

步骤 3 将 session.conf 文件拷贝至单板的/vendor/share/dbus-1 目录。

----结束

表5-2 开源组件文件及存放路径一览

文件	默认存放路径
/vendor/lib/libglib-2.0.so.0	/lib
/vendor/lib/libexpat.so.1	/lib
/vendor/lib/libpcre.so.1	/lib
/vendor/lib/libdbus-1.so.3	/lib
/vendor/lib/libreadline.so.8	/lib
/vendor/lib/libintl.so.8	/lib
/vendor/libexec/bluetooth/bluetoothd	任意路径，如/bin 目录
/vendor/bin/bluetoothctl	任意路径，如/bin 目录
/vendor/bin/dbus-daemon	任意路径，如/bin 目录
/vendor/share/dbus-1/session.conf	任意路径，如/vendor/share/dbus-1/session.conf 目录，需和启动 dbus-daemon 命令中指定 config-file 参数的路径一致，如 dbusresult=`dbus-daemon --config-file=/vendor/share/dbus-1/session.conf --print-address --fork`

5.4 Bluez 蓝牙业务基础操作示例

步骤 1 加载驱动。

```
$ insmod plat_soc.ko  
$ insmod ble_soc.ko
```

步骤 2 启动 dbus，命令如下：

```
dbusresult=`dbus-daemon --config-file=/vendor/share/dbus-1/session.conf --print-address --fork`  
export DBUS_SESSION_BUS_ADDRESS=$dbusresult  
export DBUS_SYSTEM_BUS_ADDRESS=$dbusresult
```

步骤 3 启动 bluetoothd，命令如下：

```
bluetoothd -n &
```

步骤 4 启动 bluetoothctl，命令如下：

```
bluetoothctl
```

说明

bluetoothctl 命令执行成功会出现 “[bluetooth]#” 符号。

步骤 5 启动蓝牙设备

- 在 “#” 后执行 power on
- 启动成功

```
[bluetooth]# power on
```

图5-4 启动成功

```
Changing power on succeeded  
[CHG] Controller 1C:1D:2B:33:EF:EE Powered: yes
```

步骤 6 执行扫描

- 在 “#” 后执行 scan on，开启扫描

```
[bluetooth]# scan on
```

图5-5 开启扫描

```
Discovery started
[CHG] Controller 1C:1D:2B:33:EF:EE Discovering: yes
[NEW] Device 00:1B:66:F3:FE:4D MOMENTUM TW 2
[NEW] Device C4:64:E3:F5:34:42 B44326A933E0
[NEW] Device 63:C6:F2:94:2F:1C 63-C6-F2-94-2F-1C
[NEW] Device 70:0E:71:4A:16:29 70-0E-71-4A-16-29
[NEW] Device 44:27:F3:03:13:5D Xiaomi Watch S2 135D
```

- 在“#”后执行 devices，查看扫描结果

```
[bluetooth]# devices
```

图5-6 扫描结果

```
[bluetooth]# devices
Device 00:1B:66:F3:FE:4D MOMENTUM TW 2
Device C4:64:E3:F5:34:42 B44326A933E0
Device 62:41:81:94:24:FF 62-41-81-94-24-FF
Device 44:27:F3:03:13:5D Xiaomi Watch S2 135D
```

步骤 7 执行广播

- 在#后执行 advertise on

```
[bluetooth]# advertise on
```

- 回显结果

图5-7 广播成功

```
Advertising object registered
Tx Power: off
Name: off
Appearance: off
Discoverable: on
```

步骤 8 执行连接

- 在#后执行 connect XX:XX:XX:XX:XX:XX

```
[bluetooth]# connect XX:XX:XX:XX:XX:XX
```

图5-8 连接成功

```
Attempting to connect to 4F:FE:C5:42:67:E6
[CHG] Device 4F:FE:C5:42:67:E6 Connected: yes
[4F-FE-C5-42-67-E6]# bluetoothd[1399]: src/device.c:load_gatt_db() Unable to load key file from /vendor/var/lib/bluetooth/1C:1D:2B:33:EF:EE/cache/4F:FE:C5:42:67:E6: (No such file or directory)
bluetoothd[1399]: src/device.c:load_gatt_db() No cache for 4F:FE:C5:42:67:E6
Connection successful
```

说明

- XX:XX:XX:XX:XX:XX 为对端蓝牙地址

步骤 9 关闭蓝牙设备

- 在#后执行 power off

```
[bluetooth]# power off
```

图5-9 关闭成功

```
[HCC] hci_bt_close
Changing power off succeeded
```

步骤 10 卸载驱动

1. 退出 bluetoothd、bluetoothctl 等应用
2. 卸载驱动:

```
$ rmmod ble_soc.ko
$ rmmod plat_soc.ko
```

----结束

6 缺省编译配置说明

配置项	默认值	功能
编译环境/平台配置		
WSCFG_USING_GC C=y # WSCFG_USING_LL VM_CLANG is not set WSCFG_CROSS_C OMPILER="" WSCFG_KERNEL_D IR="" WSCFG_EXTRA_CF LAGS="" WSCFG_EXTRA_PA RAMS="" WSCFG_ARCH_AR M=y # WSCFG_ARCH_CU STOM is not set WSCFG_ARCH_NA ME="arm"	-	配置编译器类型为 GCC; 配置编译器类型为 LLVM_CLANG; 配置编译器存放路径; 配置内核存放路径; 配置编译选项; 配置编译参数; 配置 CPU 架构类型为 ARM; 若非 ARM 架构, 可关闭 WSCFG_ARCH_ARM, 打开 WSCFG_ARCH_CUSTOM 宏, 并修改 WSCFG_ARCH_NAME, 如 WSCFG_ARCH_NAME="mips".
BOARD_ASIC	y	区分调试阶段和回片后的宏, 默认开启, 无需修 改。
BOARD_ASIC_WIFI	y	区分调试阶段和回片后的宏, 默认开启, 无需修 改。
WSCFG_BUS_SDIO =y	sdio	配置 WS73 总线模式, 支持配置 sdio/usb/uart 三

配置项	默认值	功能
# WSCFG_BUS_USB is not set		种模式。
# WSCFG_BUS_UART is not set		
_PRE_OS_VERSION	0	适配不同 OS，默认为 0，编译 Linux OS： _PRE_OS_VERSION_LINUX=0 _PRE_OS_VERSION_WIN32=1 _PRE_OS_VERSION_WINDOWS=2 _PRE_OS_VERSION_RAW=3 _PRE_OS_VERSION_HIRTOS=4 _PRE_OS_VERSION_WIN32_RAW=5 _PRE_OS_VERSION_LITEOS=6 _PRE_OS_VERSION_ANDROID=7
ini 配置 (conf ini 覆盖 ini 文件中的配置)		
WSCFG_INI_DBAC_STA_GC_RATIO	50	同 ini 文件中的 dbac_sta_gc_ratio。 dbac 共存 gc vap 占用空口的比例，默认 50(取值范围 20-80)。
WSCFG_INI_DBAC_STA_GO_RATIO	30	同 ini 文件中的 dbac_sta_go_ratio。 dbac 共存 go vap 占用空口的比例，默认 STA 30(取值范围 20-80)。
CONFIG_INI_HOST_GPIO	40	配置 power on 管脚号。同 ini 文件中的 power_gpio_idx，主控 power_on WS73 的 GPIO 管脚配置，配合 CONFIG_INI_HOST_GPIO_POWER_ON_LEVEL(同 ini 的 power_on_level)使用。
CONFIG_INI_HOST_GPIO_POWER_ON_LEVEL	1	配置 power on 管脚的默认配置电平。同 ini 文件中的 power_on_level，配合 CONFIG_INI_HOST_GPIO 使用，用来定义主控 power on WS73 的电平。
CONFIG_INI_DEVICE_AWAKE_GPIO_IDX	10	配置唤醒主控的 WS73 侧 GPIO 管脚号。同 ini 文件中的 device_awake_host_gpio_idx，WS73 唤醒主控的 GPIO 配置。

配置项	默认值	功能
CONFIG_INI_DEVICE_AWAKE_GPIO_LEVEL	1	配置唤醒主控的 WS73 侧 GPIO 管脚默认配置电平。同 ini 文件中的 device_awake_host_gpio_level, WS73 唤醒主控的 GPIO 电平配置。
CONFIG_INI_WAKE_UP_GPIO_IDX	70	配置唤醒主控的主控侧 GPIO 管脚号。同 ini 文件中的 wkup_gpio_idx, WS73 唤醒主控的 GPIO 配置, 配合 CONFIG_INI_DEVICE_AWAKE_GPIO_IDX 使用。
CONFIG_INI_WAKE_UP_GPIO_LEVEL	1	配置唤醒主控的主控侧 GPIO 管脚默认配置电平。同 ini 文件中的 wkup_gpio_level, device 唤醒 host 的 GPIO 的电平, GPIO10 配置高电平, 唤醒主控。
CONFIG_INI_PLAT_REBOOT_TYPE_GPIO	0	配置主控复位时 WS73 的执行策略。同 ini 文件中的 plat_reboot_type, reboot 操作类型 0: 默认复位 WS73; 1: 启动 BLE 休眠唤醒。
CONFIG_INI_BT_COEX_MODE	1	同 ini 文件中的 bt_coex_mode。支持 0: 片外 / 1: 片内两种共存模式, 默认配置片内共存。
CONFIG_INI_BSLE_USE_FLASH	0	同 ini 文件中的 bsle_use_flash。配置是否使用 flash 代替 efuse。
Wi-Fi 特性宏配置		
WSCFG_WLAN_INIT_VAP_COUNT	2	最大可以创建 wlan 设备的数量, 配合 ini 文件中的 wlan_ifname0, wlan_ifname1, wlan_ifname2 使用; 默认为 2, 允许创建 2 个 wlan 设备。
WIFI_DEBUG	y	配置使能以下宏定义, 提供更多的 DEBUG 信息和调试手段: _PRE_WIFI_DEBUG _PRE_WLAN_CFGID_DEBUG _PRE_WIFI_PRINTK _PRE_WLAN_FEATURE_SPECIAL_PKT_LOG

配置项	默认值	功能
		。
WLAN_WFA_TEST	n	WFA 测试相关的定制，配置使能宏 <code>_PRE_WLAN_FEATURE_WFA_CODE</code> ，WFA 测试版本建议开启。
WIFI_CSA	y	802.11h 定义的无线信道切换通告（channel switch Announcement）特性开关，配置使能 <code>_PRE_WLAN_FEATURE_CSA</code> 宏，编译文件 <code>hmac_csa_ap.c</code> 、 <code>hmac_csa_sta.c</code> 。
WIFI_WPS	y	WPS 特性宏，配置使能编译 <code>hmac_wps.c</code> 文件，默认支持 WPS 功能。
<code>_PRE_WLAN_FEATURE_WPA3</code>	n	WPA3 加密特性宏，控制是否支持 WPA3 加密，默认关闭。
WSCFG_WIFI_ENHANCE	y	<p>以下特性宏的总开关。关闭后，下列特性全部关闭；使能后，下列特性根据特性宏可以单独开启、关闭：</p> <p>WIFI_BSRP_NFRP、WIFI_SLP、WIFI_APF、WIFI_BASE_ROAM、WIFI_11KVR、WIFI_MBO、WIFI_AUTO_ADJUST_FREQ、WIFI_SINGLE_PROXYSTA、WIFI_CSI、WIFI_M2U、WIFI_BLACKLIST、WIFI_WAPI、WIFI_SDP、WIFI_LATENCY_STAT、WIFI_PROMISC、WIFI_TX_AMSDU、WIFI_UAPSD、WIFI_ANT_SEL、WIFI_PSD、WIFI_TWT、WIFI_DNB、WIFI_SR_STA、WIFI_ALG_CCA、WIFI_ALG_TEMP_PROTECT、WIFI_ALG_TXBF、WIFI_ALG_EDCA、WIFI_ALG_ANTI_INTF、<code>_PRE_WLAN_FEATURE_TX_CLASSIFY_LAN_TO_WLAN</code>、WIFI_WOW、WIFI_ALWAYS_TX、<code>_PRE_WLAN_DFT_STAT</code>、<code>_PRE_WLAN_FEATURE_SNIFFER</code>、</p>

配置项	默认值	功能
		WIFI_BASE_DFR、WIFI_DAQ、 _PRE_WLAN_SUPPORT_CCPRIV_CMD、 WIFI_DFX_CUSTOMIZATION。
WIFI_BSRP_NFRP	y	11AX BSRP/NFRP 特性开关，支持响应 AP 发送的 BSRP/NFRP 报文，使能后配置 _PRE_WLAN_FEATURE_BSRP 宏，编译文件 hmac_bsrp_nfrp.c。
WIFI_SLP	y	spark link 定位特性开关，使能后配置 _PRE_WLAN_FEATURE_SLP 宏，编译文件 hmac_slp.c。
WIFI_APF	y	Android packet filter 特性开关，支持根据配置的规则过滤报文，使能后配置 _PRE_WLAN_FEATURE_APF，编译文件 hmac_apf.c。
WIFI_BASE_ROAM	y	基础漫游特性开关，支持非 11K/V/R 的漫游，使能后配置 _PRE_WLAN_FEATURE_ROAM，编译文件 hmac_roam_alg.c、 hmac_roam_connect.c、hmac_roam_main.c。
WIFI_11KVR	y	11K/V/R 漫游特性开关，11K：无线电环境测量并上报，11V：AP 引导漫游，11R：快速加密连接，使能后配置 _PRE_WLAN_FEATURE_11K、 _PRE_WLAN_FEATURE_11V、 _PRE_WLAN_FEATURE_11R 宏，编译文件 hmac_11k.c、hmac_11v.c、hmac_11r.c。
WIFI_MBO	y	11AX MBO 特性开关，基于 11KVR 的增强漫游功能，使能后配置 _PRE_WLAN_FEATURE_MBO 宏，编译文件 hmac_mbo.c。
WIFI_AUTO_ADJUST_FREQ	y	自动调频特性开关，支持根据流量调整 CPU 主频；使能后编译文件 hmac_auto_adjust_freq.c、 hal_auto_adjust_freq.c，功能生效还需要下发命令打开自动调频功能。

配置项	默认值	功能
WIFI_SINGLE_PROXYSTA	y	3 地址 repeater 特性开关, 使能后配置 <code>_PRE_WLAN_FEATURE_SINGLE_PROXYSTA</code> , 编译文件 <code>hmac_single_proxysta.c</code> 。
WIFI_CSI	y	信道状态信息上报特性开关, 使能后配置 <code>_PRE_WLAN_FEATURE_CSI</code> , 编译文件 <code>hmac_csi.c</code> 、 <code>hal_csi.c</code> 。
WIFI_M2U	y	多播转单播特性开关, 使能后配置 <code>_PRE_WLAN_FEATURE_M2U</code> , 编译文件 <code>hmac_m2u.c</code> 。
WIFI_BLACKLIST	y	配置 Wi-Fi 连接的黑白名单, STA 场景下限制接入 AP 的范围, softap 场景下限制接入 STA 的范围, 使能后编译文件 <code>hmac_blacklist.c</code> 。
WIFI_WAPI	n	WAPI 特性开关, 使能后编译文件 <code>hmac_wapi.c</code> 、 <code>hmac_wapi_sms4.c</code> 、 <code>hmac_wapi_wpi.c</code> , 默认关闭。
WIFI_SDP	y	SDP 特性开关, wifi aware(NAN)功能, 使能后编译文件 <code>hmac_sdp.c</code> 、 <code>hmac_sdp_test.c</code> , 默认开启。
WIFI_LATENCY_STAT	y	转发时延统计特性开关, 使能后配置 <code>_PRE_WLAN_LATENCY_STAT</code> 宏, 编译文件 <code>hmac_latency_stat.c</code> , 编译后需要使用命令行开启时延统计功能。
WIFI_PROMISC	y	混杂模式即 sniffer 特性开关, 使能后编译文件 <code>hmac_promisc.c</code> 。
WIFI_TX_AMSDU	y	小包 A-MSDU 聚合特性开关, 使能后编译文件 <code>hmac_tx_amsdu.c</code> 。
WIFI_UAPSD	y	U-APSD 节能特性开关, 使能后配置 <code>_PRE_WLAN_FEATURE_STA_UAPSD</code> 宏, 编译 <code>hmac_uapsd_sta.c</code> 、 <code>hmac_uapsd.c</code> 。
WIFI_ANT_SEL	y	天线分集选择特性开关, 使能后配置

配置项	默认值	功能
		_PRE_WLAN_FEATURE_ANT_SEL 宏，编译 hmac_ant_sel.c、hal_ant_sel.c，WS73 不支持此特性。
WIFI_PSD	y	Wi-Fi 功率谱密度测量上报特性开关，使能后配置 _PRE_WLAN_FEATURE_PSD 宏，编译文件 hmac_psd.c。
WIFI_TWT	y	11AX TWT 节能特性开关，使能后配置 _PRE_WLAN_FEATURE_TWT 宏，编译文件 hmac_twt.c。
WIFI_DNB	y	动态窄带特性开关，开启后对接特定 AP 时，支持 8-LTF 格式的 HE 报文接收，使能后编译文件 hmac_dnb_sta.c。
WIFI_SR_STA	y	11AX Spatial reuse（空间复用）特性开关，使能后配置 _PRE_WLAN_FEATURE_SR 宏，编译文件 hmac_sr_sta.c。
WIFI_ALG_CCA	y	CCA（空闲信道评估）主动调整算法特性开关，使能后编译文件 alg_cca_intrf_mode.c、alg_cca_optimize.c。
WIFI_ALG_TEMP_PROTECT	y	温度保护算法特性开关，使能后编译文件 alg_temp_protect.c。
WIFI_ALG_TXBF	y	TxBF 波束增强（RX）特性开关，使能后编译文件 alg_txbf.c。
WIFI_ALG_EDCA	y	EDCA（增强型分布式信道访问）算法特性开关，使能后编译文件 alg_edca_intrf_mode.c、alg_edca_opt.c。
WIFI_ALG_ANTI_INTERF	y	若干扰免疫算法特性开关，使能后编译文件 alg_anti_interference.c。
_PRE_WLAN_FEATURE_TX_CLASSIFY_LAN_TO_WLAN	y	待 WLAN 发送的报文类型识别功能，识别无 QOS 配置的 RTP 报文并映射 VI/VO 队列发送，使能后编译文件 hmac_traffic_classify.c。

配置项	默认值	功能
WIFI_WOW	y	WOW 低功耗唤醒特性开关，使能后配置 <code>_PRE_WLAN_FEATURE_WOW_OFFLOAD</code> 、 <code>_PRE_WLAN_FEATURE_DYNAMIC_OFFLOAD</code> 宏，编译文件 <code>hmac_wow.c</code> 。
WIFI_ALWAYS_TX	y	常发常收特性开关，使能后配置 <code>_PRE_WLAN_FEATURE_ALWAYS_TX</code> 宏，编译文件 <code>hmac_al_tx_rx.c</code> 。
<code>_PRE_WLAN_DFT_STAT</code>	y	维测信息特性宏，开启后记录更多维测信息，提供更多维测命令。
<code>_PRE_WLAN_FEATURE_SNIFFER</code>	n	Wi-Fi sniffer 特性开关，开启后支持空口抓包，需要配合混杂模式使用。
WIFI_BASE_DFR	y	基础 DFR 自愈特性总开关，使能后配置 <code>_PRE_WLAN_FEATURE_DFR</code> 、 <code>_PRE_WLAN_DFR_STAT</code> 宏。
<code>_PRE_WLAN_FEATURE_DFR</code>	y	Wi-Fi 驱动自愈特性开关，配置后支持 Wi-Fi 驱动 DFR 自愈。
<code>_PRE_WLAN_DFR_STAT</code>	y	Wi-Fi 驱动自愈状态上报特性开关，配置后支持上报自愈事件。
WIFI_DAQ	y	Wi-Fi MAC&PHY 数采特性开关，使能后定义 <code>_PRE_WLAN_FEATURE_DAQ</code> 宏，编译文件 <code>hmac_sample_daq.c</code> 、 <code>hmac_sample_daq_phy.o</code> 、 <code>hal_daq.c</code> 。
<code>_PRE_WLAN_SUPPORT_CCPRIV_CMD</code>	y	CCPRIV 命令特性开关，开启后支持解析 CCPRIV 命令。
WIFI_BTCOEX	y	Wi-Fi/BT 共存特性开关，使能后定义 <code>_PRE_WLAN_FEATURE_BTCOEX</code> ，编译文件 <code>hmac_btcoex.c</code> 、 <code>hmac_btcoex_ba.c</code> 、 <code>hmac_btcoex_btsta.c</code> 、 <code>hmac_btcoex_m2s.c</code> 、 <code>hmac_btcoex_notify.c</code> 、 <code>hmac_btcoex_ps.c</code> 、 <code>hal_coex_reg.c</code> 。

配置项	默认值	功能
WIFI_ONLINE_CALI	y	射频前端在线校准特性开关，开启后支持射频前端在线校准。
_PRE_WLAN_WIRELESS_EXT	y	兼容 iwlist 接口修改，使能后支持 iwlist 命令执行扫描等操作。
WIFI_TCP_ACK_FILTER	y	TCP ACK 过滤特性开关，通过调整发送 TCP ACK 的策略，提升 TCP RX 的性能，使能后定义 _PRE_WLAN_TCP_OPT、_PRE_WLAN_FEATURE_OFFLOAD_FLOWCTL 宏。
CONTROLLER_CUSTOMIZATION	n	新增宏定义，解决升级内核 5.15 引入的 CFI（接口定义和声明不一致）问题。
平台特性宏配置		
_PRE_PLAT_SHA256SUM_CHECK	y	firmware 完整性校验。
CONFIG_SUPPORT_SHA256_MEM_USING_VMAALLOC	n	在 kmalloc 不支持大内存申请的情况下，启用大块连续地址内存校验 firmware 完整性。
WSCFG_ONEIMAGE	y	在 WS73 所有平台函数加 ws73_前缀，避免与主控函数同名冲突。
WSCFG_PLAT_VARIABLE_FEATURE	y	平台增强功能总开关。HSO 日志、dfr、hcc_test 功能等均依赖该特性宏。
WSCFG_PLAT_DIAG_LOG_OUT	y	HSO 日志总开关。
CONFIG_DIAG_SUPPORT_SOCKET	n	HSO 日志支持网口连接 DebugKits 工具。
CONFIG_DIAG_SUPPORT_UART	n	HSO 日志支持串口连接(需适配串口号)DebugKits 工具。
CONFIG_DIAG_SUPPORT_LOCAL_LOG	y	支持开启 HSO 离线日志(写本地日志文件)功能。
CONFIG_PLAT_SUPPORT_DFR	y	WS73 发生异常时支持 DFR 自愈功能。
CONFIG_PLAT_DFR_OUTPUT_PATH="/etc/ws73"	-	配置 DFR 日志文件输出路径。

配置项	默认值	功能
CONFIG_PLAT_SUP PORT_DFR_TRIGG ER	n	支持 ccpriv 命令触发 DFR 自愈功能。
CONFIG_HCC_SUP PORT_TEST	y	支持 hcc_test 测试与 WS73 通信测试。
CONFIG_HCC_ERR OR_PRINT	y	支持 HCC 错误提示串口打印开关。
_PRE_PLAT_HCC_S DIO	y	支持 WS73 与主控使用 sdio 通信，仅 WS73S 需 要配置该项。
CONFIG_HCC_SDIO _SUPPORT_SCATT ER	y	支持 sdio 的 scatter 传输。
BT_EM_BUFFER_C ALI_SUPPORT	y	支持 BT 校准。
_PRE_PLAT_SLP_U ART_FORWARD	n	支持通过 WS73 建立主控和 SLP 芯片通信的转发 通道。
CONFIG_DFX_SUPP ORT_SYFS	y	支持文件系统命令存储 HSO 日志调试信息。
CONFIG_SDIO_RES CAN	y	若产品存在反复加载卸载 WS73 plat_soc.ko 的需 求，则需要主控提供 sdio 重新探卡的接口在 sdio_detectcard_change 中调用，仅 WS73S 反 复加/卸载 ko 的场景需要关注。
CONFIG_STR_RES UME_ASYNC	n	支持主控待机唤醒恢复时，WS73 异步恢复，提 升待机唤醒的恢复速度。
CONFIG_STR_SUSP END_WAIT_USB_DI SCONNECT	n	支持主控进入待机唤醒时，保证 WS73 通信时 序，有 STR 待机唤醒的场景推荐开启该功能。
CONFIG_ANDROID_ FEATURE	n	未使用。
CONFIG_SUPPORT _RESET_DEVICE_IN _INSMOD	n	支持开启业务时复位 WS73，建议 WS73U 且支 持 GPIO 复位 WS73 的场景下开启。
CONFIG_SUPPORT _STATIC_FIRMWAR E_MEM	n	支持 firmware 下载等大内存使用静态 BSS 段内 存。
CONFIG_GPIO_USI NG_HI_DRV	n	支持海思媒体芯片 GPIO 来操作 WS73 复位的接

配置项	默认值	功能
		口。
CONFIG_SUPPORT_HCC_CONN_CHECK	n	支持启动业务过程中，WS73 上电后检测 HCC 通路，若不通复位 WS73 并重新上电尝试，可以提供系统启动兼容性，仅 WS73U 支持。
CONFIG_GPIO_ADAPTIVE_POWER_ON_LEVEL	n	主控通过一个 GPIO 连接 WS73 的 power on 管脚，该选项支持根据驱动初始化阶段自动检测到的电平，来作为 WS73 保持工作的 power on 电平。
CONFIG_MFG_GPIO_CTRL	n	支持使用 ccpriv 命令配置 WS73 的指定 GPIO 管脚的电平。
CONFIG_INI_STR_LPM_MODE	0	主控待机时配置 WS73 进入 lowpower 模式或直接下电，配置 0 为直接下电，配置 1 为进入 lowpower 模式，默认配置为 0。
CONFIG_NO_SECURITY	n	配置是否使用安全函数。
PLAT_LITE_EXTREME	n	主控侧驱动代码小型化，裁剪部分特性，减少 code size。
CONFIG_PRE_BSLE_GATEWAY	n	支持 BSLE 网关，仅 WS73E 打开。
CONFIG_HCC_SUPPORT_HEARTBEAT	n	HCC 通信支持心跳机制。
firmware 固件存放路径配置		
CONFIG_FIRMWARE_BIN_PATH="/etc/ws73/ws73.bin"	-	存放 WS73 firmware 固定的路径，驱动加载后，从对应路径读取固定，下载到 WS73。
CONFIG_FIRMWARE_WIFICALI_PATH="/etc/ws73/wifi_cali.bin"	-	
CONFIG_FIRMWARE_BSLECALI_PATH="/etc/ws73/btc_cali.bin"	-	
CONFIG_FIRMWARE_WOW_PATH="/etc"	-	

配置项	默认值	功能
/ws73/wow.bin"		
CONFIG_INI_FILE_PATH="/etc/ws73_cfg.ini"	-	存放 WS73 ini 配置文件的路径，驱动加载后，从对应路径读取并解析配置文件。
BLE 特性宏配置		
WSCFG_BLE_COMPILE_BY_DEFAULT	y	打开默认编译 BLE 相关的内核模块文件。
CONFIG_BLE_MAC_FORK	y	打开根据 WIFI 地址派生蓝牙地址。
CONFIG_INI_BLE_DISABLE_LL_PRIVACY	-	同 ini 文件中的 ble_disable_ll_privacy。
CONFIG_INI_BSLE_SUSPEND_MODE	-	同 ini 文件中的 bsle_suspend_mode。
CONFIG_INI_BSLE_SUSPEND_SCAN_INTERVAL	300	同 ini 文件中的 bsle_suspend_scan_interval。
CONFIG_INI_BSLE_SUSPEND_SCAN_WINDOW	30	同 ini 文件中的 bsle_suspend_scan_window。
CONFIG_INI_BSLE_FRONT_SWITCH	-	同 ini 文件中的 bsle_front_switch。
SLE 特性宏配置		
WSCFG_SLE_COMPILE_BY_DEFAULT	y	打开默认编译 SLE 相关的内核模块文件。

A 缩略语

C		
CPU	Central Processing Unit	中央处理器单元
E		
EAP	Extensible Authentication Protocol	可扩展认证协议
H		
HCI	Human-Computer Interaction	人机交互
P		
P2P	Wi-Fi Peer-to-Peer	Wi-Fi 直连
S		
SDIO	Secure Digital Input and Output	安全数字输入输出接口
U		
UART	Universal Asynchronous Receiver/Transmitter	通用异步收发器
W		
WEP	wired equivalent privacy	有线等效保密

WPA	Wi-Fi Protected Access	保护无线网络安全系统
WAPI	Wireless LAN Authentication and Privacy Infrastructure	无线局域网鉴别和保密基础结构
WPS	Wi-FiProtected Setup	Wi-Fi 保护设置