

WS73V100 Linux Wi-Fi、BLE 软件

# 开发指南

文档版本 09

发布日期 2024-10-08

# 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
09	2024-10-08	更新 “3.5 SoftAp 模式基础操作示例” 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	• 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	• 更新 “3.5 SoftAp 模式基础操作示例” 章节内容。
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。

文档版本	发布日期	修改说明
		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"><li>更新“3.4 STA 模式基础操作示例”章节内容。</li><li>更新“3.5 SoftAp 模式基础操作示例”章节内容。</li><li>更新“3.7 Sta/P2P 共存基础操作示例”章节内容。</li></ul>
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"><li>新增“2 设备检测”章节内容。</li><li>更新“3.4 STA 模式基础操作示例”章节内容。</li><li>更新“3.5 SoftAp 模式基础操作示例”章节内容。</li><li>更新“3.6 P2P(Wi-Fi Peer-to-Peer)模式基础操作示例”章节内容。</li><li>新增“3.7 Sta/P2P 共存基础操作示例”章节内容。</li><li>新增“3.8 Sta&amp;Softap 共存基础操作示例”章节内容。</li></ul>
00B01	2023-08-25	第一次临时版本发布。

# 目 录

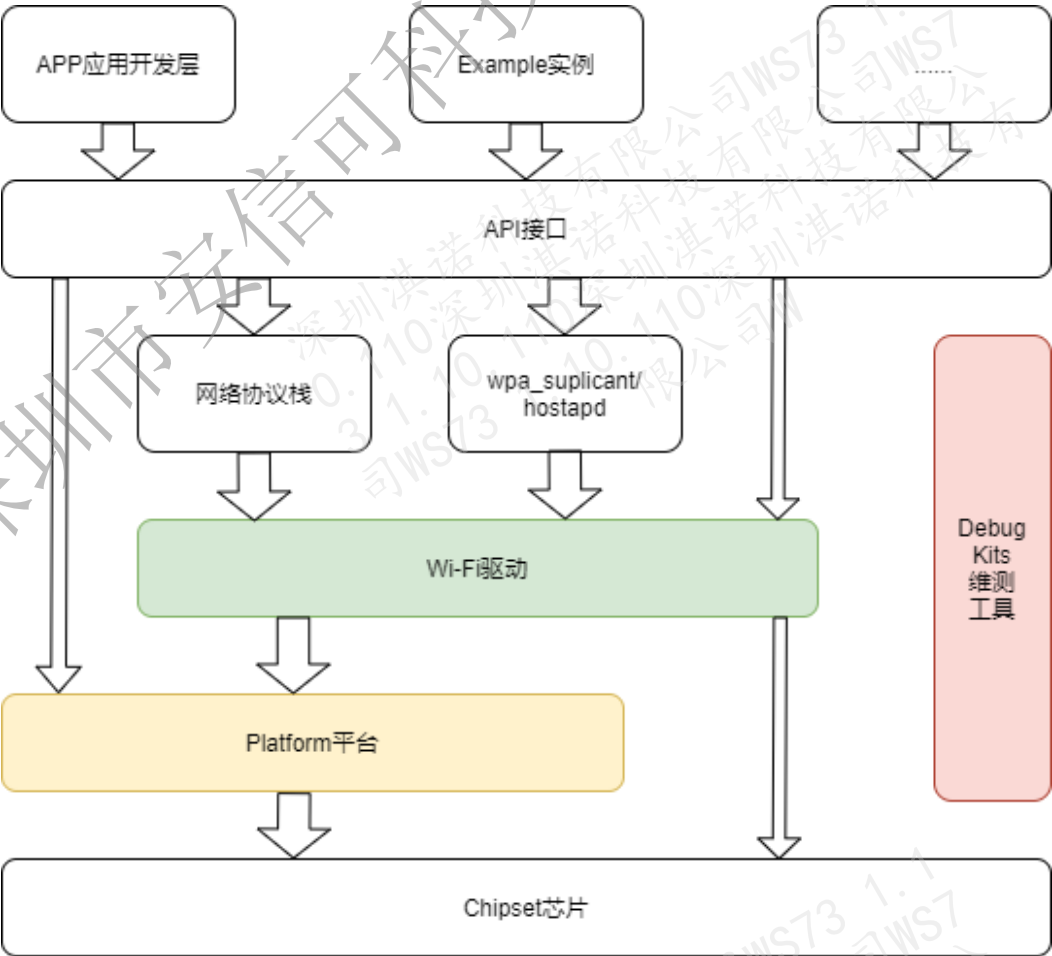
前言 .....	i
1 概述 .....	1
2 设备检测 .....	3
3 Wi-Fi 业务开发指南 .....	4
3.1 Wi-Fi 设备检测 .....	4
3.1.1 SDIO Wi-Fi 设备检测 .....	4
3.1.2 USB Wi-Fi 设备检测 .....	5
3.2 加载驱动 .....	5
3.3 载入工具 .....	7
3.4 STA 模式基础操作示例 .....	9
3.5 SoftAp 模式基础操作示例 .....	11
3.6 P2P(Wi-Fi Peer-to-Peer)模式基础操作示例 .....	12
3.7 Sta/P2P 共存基础操作示例 .....	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 .....	16
5 蓝牙业务开发指南 .....	17
5.1 蓝牙设备检测 .....	17
5.1.1 SDIO 蓝牙设备检测 .....	17
5.1.2 USB 蓝牙设备检测 .....	18

5.2 加载驱动 .....	18
5.3 载入工具 .....	19
5.4 Bluez 蓝牙业务基础操作示例 .....	21
<b>A 缩略语 .....</b>	<b>24</b>

# 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_calibin	/etc/ws73
firmware/btc_calibin	/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

```
$ udhpcp -i wlan0  
$ ifconfig wlan0
```

步骤 7 卸载驱动。

1. 退出 wpa\_supplicant、udhpcp 等应用
2. 卸载驱动：

```
$ killall wpa_supplicant  
$ killall udhpcp  
$ 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
rfkill: 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 - Connected to 3a:bc:1a:cb:8d:90 completed [1d=0 id str=
<3>P2P-GROUP-STARTED p2p-p2p0-0 client_ssid='DIRECT-ra-MeizunoteM1' freq=2462 psk=04535780efe25e2055d6e82de075d101e7380bfac3d75f67794f46517566bb 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：

```

$ udhcpc -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、udhcpc 等应用。

```

$ killall udhcpd

```

```
$ killall wpa_supplicant  
$ killall udhcpc
```

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

## 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 txpower 值, 选择 $\leq$ txpower 的最大档位。

- 当 ws73\_cfg.ini 的 bt\_maxpower=7 时:
  - txpower=10, 则选中第 4 档发送广播的功率为 10dbm
  - txpower=9, 则选中第 3 档发送广播的功率为 6dbm
  - txpower=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:
  - txpower=10, 则选中第 4 档发送广播的功率为 6dbm
  - txpower=9, 则选中第 3 档发送广播的功率为 6dbm
  - txpower=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_cali.bin	/etc/ws73
firmware/btc_cali.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

# 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	保护无线电脑网络安全系统
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