

WS73V100 常见问题

FAQ

文档版本 05

发布日期 2024-10-21

前言

概述

本文档主要介绍 WS73V100 解决方案中常见的问题处理和解决办法。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
WS73	V100


读者对象





本文档主要适用于以下工程师：

- 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

符号	说明
	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
05	2024-10-21	<ul style="list-style-type: none">更新 “2.1.1 SDIO 设备通信异常” 小节内容。新增 “4.3 常见性能问题及影响参数” 章节内容。
04	2024-06-07	新增 “4 Wi-Fi 性能问题排查方法” 章节内容。
03	2024-04-17	更新 “3.3 DebugKits 使用失败” 小节内容。
02	2024-02-26	更新 “3.2 wpa_supplicant 运行失败” 小节内容。
01	2023-12-11	第一次正式版本发布。 更新 “2.1.3 USB 设备通信异常” 小节内容。
00B01	2023-11-28	第一次临时版本发布。

目 录

前言	i
1 SDK 开发环境和 SDK 使用类	1
1.1 SDK 开发环境问题	1
1.1.1 Python 版本不兼容冲突	1
1.1.2 依赖包问题	2
1.2 SDK 使用问题	3
1.2.1 Python 版本错误	3
1.2.2 SDK 编译出错未指定编译器	3
1.2.3 SDK 编译出错未指定编译方式 (GCC or CLANG)	4
2 系统异常类	6
2.1 Host 侧和 Device 侧通信异常	6
2.1.1 SDIO 设备通信异常	6
2.1.2 SDIO 总线卸载 plat_soc 后重新加载异常	8
2.1.3 USB 设备通信异常	9
2.1.4 FRW 消息传输异常	10
3 软件常见问题及定位方法	11
3.1 驱动加载失败	11
3.1.1 提示: insmod: short read	11
3.1.2 提示: Exec format error	11
3.1.3 提示: Firmware 下载失败	12
3.1.4 提示: Unknown symbol cfg80211_xxx	13
3.1.5 提示: Unknown symbol hci_free_dev	14
3.2 wpa_supplicant 运行失败	15

3.3 DebugKits 使用失败.....	15
4 Wi-Fi 性能问题排查方法.....	18
4.1 性能问题概述.....	18
4.2 性能问题整体定位思路.....	19
4.2.1 环境排查.....	19
4.2.2 缩小问题范围.....	23
4.2.3 差异分析.....	24
4.2.4 干扰场景.....	27
4.3 常见性能问题及影响参数.....	29
4.3.1 TCP RX 打流性能波动较大.....	29
4.3.2 TCP RX 打流性能异常低下.....	29
4.3.3 弱信号/远距离场景性能较低.....	30
4.3.4 UDP/TCP RX 打流性能较低.....	30
4.3.5 打流过程中 CPU 占用较高.....	30

1

SDK 开发环境和 SDK 使用类

本章节主要介绍 SDK 开发环境搭建以及使用时出现的常见问题处理和解决办法。

1.1 SDK 开发环境问题

1.2 SDK 使用问题

1.1 SDK 开发环境问题

- 请参考《WS73V100 Linux 平台驱动移植 用户指南》中“Python 环境安装”章节，安装 SDK 依赖 Python 及 Python 库。
- 本章节中图例为说明问题使用的测试代码，与实际使用版本有差异，以构建脚本要求版本为准。

1.1.1 Python 版本不兼容冲突

问题描述

图1-1 在 SDK 下执行 make 失败

```
$ make BLE_ANDROID=y
File "/build/scripts/mconfig.py", line 18
    self._old_dir: str = os.getcwd()
    ^
SyntaxError: invalid syntax
make: *** [prepare] Error 1

#### make failed to build some targets ####
```

在 SDK 中执行“make xxx”命令，报错：SyntaxError: invalid syntax。

问题分析

Python 3.8 引入了海象运算符 (Walrus Operator)，它使用符号 “:=” 表示，它可以将表达式的值赋给变量，并且在同一表达式中使用该变量。

若当前使用的 Python 版本低于 3.8，会导致无法识别符号 “:=” 而报错。

解决方法

安装 Python 3.8+版本。

- 以命令行安装：

```
sudo apt-get install python3.8
```

- 源码包编译安装：

```
tar -xf Python-3.8.5.tgz
cd Python-3.8.5
./configure
sudo make
sudo make install
```

1.1.2 依赖包问题

问题描述

依赖包版本与实际使用版本不一致，如图 1-2 所示。

图1-2 Python 依赖包不匹配

```
scons: Reading SConscript files ...
3.7.3 (default, Dec 27 2019, 03:17:52)
[GCC 5.4.0 20160609]
ModuleNotFoundError: No module named 'Crypto':
  File "/home/xubinhua/Hi3861V100R001C01SPC020B030/SConstruct", line 7:
    from scripts import common_env, scons_utils, scons_app, scons_env_cfg, pkt_builder
  File "/home/xubinhua/Hi3861V100R001C01SPC020B030/build/scripts/pkt_builder.py", line 12:
    import make_upg_file as MAKE_IMAGE
  File "/home/xubinhua/Hi3861V100R001C01SPC020B030/build/scripts/make_upg_file.py", line 13:
    from Crypto.Hash import SHA
```

问题分析

Pycryptodome 未安装，或者安装的版本与 Python 版本不匹配而引发的问题时常发生，因为 Linux 系统允许安装多个 Python 版本，依赖包根据不同 Python 版本安装后，不会共享依赖包。用户在安装依赖包时，容易使用错误的 Python 版本安装。

解决方法

SDK 中依赖包需要支持 python3.8 版本，确认已安装依赖包与编译要求的 Python 版本相匹配。如果没有安装，请使用 python3.8 安装。

1.2 SDK 使用问题

1.2.1 Python 版本错误

问题描述

引发报出 Python 版本错误的因素有多种，例如：“1.1 SDK 开发环境问题”中介绍的安装库版本不匹配问题。以下提供一些常用的排查思路和解决方法。

解决方法

通过“python --version”查看默认 Python 版本（要求为 python3.8+）。

- 如果没有安装 python3.8，需先安装。
- 如果无法修改默认 Python 默认使用版本，需在个人目录下，建立 python3.8 的软链接（例如：~/bin），将此目录添入“PATH”系统参数中，并且此目录路径排在 PATH 最前，重新登陆终端，再次查看 Python 版本。（如把当前目录添加到环境变量，并且优先使用，export PATH=\$(pwd):\$PATH）

1.2.2 SDK 编译出错未指定编译器

问题描述

编译 SDK 版本时，上报类似编译链、内核路径不存在等问题，如图 1-3 所示。

图1-3 编译器配置异常示例

```
@yanta:~/sdk_ws73/tmp$ make all
kconfig header saved to '/output/bin/autoconfig.h'
INI file generate success in /output/bin/ws73_cfg.ini!
cd /driver/platform && \
BUILD_DEVICE_WITH_ROM_REPO=yes\
make -j48 && cp -f plat_soc.ko /output/bin/
make[1]: Entering directory '/driver/platform'
make -C "" ARCH="arm" CROSS_COMPILE="" M=/driver/platform modules -j KBUILD_MODPOST_WARN=1
make: Entering an unknown directory
```

SDK 编译前会检查编译器是否已经指定正确。出现错误的情况有：

- 未将编译器路径加到\$PATH 中，或者指定不正确。
- \$PATH 指定了多款编译器路径，导致检查出错。
- 编译器未完整安装。编译器 bin 目录中不包含 “arm-unknown-linux-gcc”。

解决方法

检查环境变量\$PATH，将编译器路径添加到服务器环境变量。

编译器检查有以下条件：

- 路径以 “bin” 或 “bin/” 结尾。
- bin 目录中需包含 “arm-unknown-linux-gcc”。
- 尽量勿将多个版本的编译器放在同一路径下，因为系统默认使用第一个被找到的编译器，可能与所需要版本不匹配。

图1-4 编译器配置正常示例

```
#
# build
#
# Configure the compilation environment
#
WSCFG_CROSS_COMPILE="arm-himix100-linux-"
WSCFG_KERNEL_DIR="/kernel/linux-4.9.y"
WSCFG_ARCH_ARM=y
# WSCFG_ARCH_CUSTOM is not set
WSCFG_ARCH_NAME="arm"
BOARD_ASIC=y
BOARD_ASIC_WIFI=y
WSCFG_BUS_SDI0=y
```

1.2.3 SDK 编译出错未指定编译方式（GCC or CLANG）

问题描述

编译 SDK 时会检查是否指定了编译方式，如果未添加 WSCFG_USING_GCC 及 WSCFG_USING_LLVM_CLANG 选项，会出现编译问题，如图 1-5 所示。

图1-5 未配置编译方式示例

```
hi3873-master$ make
warning: CONFIG_PLAT_DFR_OUTPUT_PATH (defined at driver/platform/kconfig:54) has leading or trailing whitespace in its prompt
No change to Kconfig header in /hi3873-master/output/bin/autoconfig.h
warning: CONFIG_PLAT_DFR_OUTPUT_PATH (defined at driver/platform/kconfig:54) has leading or trailing whitespace in its prompt
INI file generate success in /home/x00383584/code/TR5/hi3873-master/output/bin/ws73.cfg.ini!
cd /hi3873-master/driver/platform &&
BUILD_DEVICE_WITH_ROM_REPO=yes
make -j48 && cp -f plat_soc.ko /hi3873-master/output/bin/
make[1]: Entering directory '/hi3873-master/driver/platform'
Makefile:382: *** "error please specify an available compiler!". Stop.
make[1]: Leaving directory '/hi3873-master/driver/platform'
make: *** [Makefile:87: platform] Error 2
x00383584@yanfa:~/code/TR5/hi3873-master$
```

SDK 编译前会检查编译方式是否已经指定正确。出现错误的情况有：

- 既未指定为 GCC 编译模式又未指定未 CLANG 编译模式。
- 指定了 GCC 模式，但是未配置工具链。
- 指定了 CLANG 模式，但是未配置 CLANG PATH。

解决方法

正确指定编译方式，并且指定配套的编译工具路径，如下示例图 1-6 及：

图1-6 GCC 配置正确示例

```
# Generate by menuconfig

#
# build
#
#
# Configure the compilation environment
#
WSCFG_USING_GCC=y
# WSCFG_USING_LLVM_CLANG is not set
WSCFG_CROSS_COMPILE="/usr/bin/arm-linux-gnueabihf-"
WSCFG_KERNEL_DIR="/kernel/KERNEL_0B7"
```

图1-7 CLANG 配置正确示例

```
# Generate by menuconfig

#
# build
#
#
# Configure the compilation environment
#
# WSCFG_USING_GCC is not set
WSCFG_USING_LLVM_CLANG=y
WSCFG_CROSS_COMPILE="/usr/bin/arm-linux-gnueabihf-"
WSCFG_KERNEL_DIR="/kernel/KERNEL_0B7"
WSCFG_KERNEL_PATH="/usr/bin/arm-linux-gnueabihf-"
WSCFG_KERNEL_FLAGS=""
WSCFG_EXTRA_PARAMS="LLVM=1 LLVM_IAS=1"
# WSCFG_ARCH_ARM is not set
```

2 系统异常类

本章节主要介绍异常触发时看门狗相关问题和异常导致死机信息打印的相关问题。

2.1 Host 侧和 Device 侧通信异常

2.1 Host 侧和 Device 侧通信异常

2.1.1 SDIO 设备通信异常

问题描述

加载 plat_soc.ko 时，显示 “sdio load fail”，如图 2-1 所示。

图2-1 SDIO 加载异常

```
/komod # insmod plat_soc.ko
plat_soc: loading out-of-tree module taints kernel.
plat_soc:EMERG]0:emerg log.
plat_soc:ALERT]1:alert log.
plat_soc:CRIT]2:crit log.
plat_soc:E]3:err log.
plat_soc:W]4:warning log.
plat_soc:E]custom ini[plat kern log level]: 7.
plat_soc:E]board_power_gpio_init::succ, power_gpio_idx=[40]!
plat_soc:E]board_wakeup_gpio_init::succ, wkup_gpio_idx=[70]!
plat_soc:E]tty unlocked_ioctl() ret: 0
plat_soc:E]zdiag_uart rx task enter
plat_soc:E]tcp rx task

[SDIO][E]sdio load fail.
[sdio_trigger_probe:2215]
[SDIO][E]sdio first probe failed![hcc_adapt_sdio_load:2547]
plat_soc:E]hcc init: fail
plat_soc:E][SH_PROC] task exit.
plat_soc:E]zdiag_uart: channel exit enter.
```

解决方法

1. 检测硬件焊接，SDIO 设备与主控设备焊接是否正常。
2. 执行“cat /proc/mci/mci_info”命令，查看是否可以识别到 SDIO 设备。
3. 检查“Host work clock”配置频率，不能超过 50MHz。

图2-2 SDIO 设备示例

```
~ # cat /proc/mci/mci_info
MCIO: 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
```

2.1.2 SDIO 总线卸载 plat_soc 后重新加载异常

问题描述

SDIO 总线形态下，卸载掉已加载的所有 ko 后，重新加载 plat_soc.ko 时，显示 “sdio load fail”，如图 2-3 所示。

图2-3 SDIO 加载异常

```
/komod # insmod plat_soc.ko
plat_soc: loading out-of-tree module taints kernel.
plat_soc:EMERG]0:emerg log.
plat_soc:ALERT]1>alert log.
plat_soc:CRIT]2:crit log.
plat_soc:E]3:err log.
plat_soc:W]4:warning log.
plat_soc:E]custom ini[plat kern log level]: 7.
plat_soc:E]board_power_gpio_init::succ, power_gpio_idx=[40]!
plat_soc:E]board_wakeup_gpio_init::succ, wkup_gpio_idx=[70]!
plat_soc:E]tty unlocked_ioctl() ret: 0
plat_soc:E]zdiag uart rx task enter
plat_soc:E]tcp rx task

[SDIO][E]sdio load fail.
[sdio_trigger_probe:2215]
[SDIO][E]sdio first probe failed![hcc_adapt_sdio_load:2547]
plat_soc:E]hcc init: fail
plat_soc:E][SH_PROC] task exit.
plat_soc:E]zdiag_uart: channel exit enter.
```

解决方法

在卸载掉所有业务 ko 时，会触发 WS73 芯片的复位，接下来再卸载 plat_soc.ko 时，会失去驱动对 Linux 内核 SDIO 设备的控制句柄，在下次加载时，由于 WS73 芯片复位 SDIO 断开，需要调用 SDIO 的 rescan 接口进行重新探卡，但是此时 SDIO 设备句柄已经丢失，无法重新探卡，SDIO 不能正常通信，导致重新加载 plat_soc.ko 失败。

一般主控板使用的 SDK，会在 Linux 原生代码的基础上额外封装无需设备参数的 SDIO 探卡接口，如 3518 主控板的 SDK 中，在 Linux 内核的 drivers\mmc\host 目录下有封装 hisi_sdio_rescan() 接口，通过该接口可以触发 SDIO 重新探卡，如图 2-4 所示。

1. 在主控 SDK 的 mmc 目录下找到 SDIO 重新探卡接口。
2. 在 WS73SDK 的 driver/platform/hcc/host/hcc_sdio_host.c 文件，sdio_detectcard_change() 函数中调用 SDIO 重新探卡接口，如图 2-5 所示。

3. 重新编译驱动，验证是否可以正常重新加载，若不行则可以调整入参的 SDIO ID 重新尝试。

图2-4 3518SDK 中 SDIO 重新探卡接口

```
/* This api is for wifi driver rescan the sdio device,
 * ugly but it is needed */
int hisi_sdio_rescan(int slot)
{
    struct mmc_host *mmc = mci_host[slot];

    if (mmc == NULL) {
        pr_err("invalid mmc, please check the argument\n");
        return -EINVAL;
    }

    mmc_detect_change(mmc, 0);
    return 0;
}
EXPORT_SYMBOL_GPL(hisi_sdio_rescan);
```

图2-5 在 WS73 SDK 中添加 SDIO 重新探卡接口

```
2211: // 声明主控封装的SDIO重新扫描接口
2212: extern int hisi_sdio_rescan(int slot);
2213:
2214:
2215: /* sdio-first.enum, wifi power on, must down later */
2216: #if defined(_PRE_OS_VERSION_LINUX) && defined(_PRE_OS_VERSION) && (_PRE_OS_VERSION_LINUX == _PRE_OS_VERSION)
2217: static td_s32 sdio_detectcard_change(td_void)
2218: {
2219:     // 调用主控封装的SDIO重新扫描接口
2220:     hisi_sdio_rescan(1);
2221:     return EXT_ERR_SUCCESS;
2222: }
```

2.1.3 USB 设备通信异常

问题描述

加载 plat_soc.ko 驱动时，加载失败。。

解决方法

WS73 模组默认 ID 号为 ffff:3733，执行 “lsusb”，若主控未识别到相关设备，检测设备连接是否异常。

图2-6 WS73 USB 模组示例

```
/komod # lsusb
Bus 001 Device 001: ID 1d6b:0002
Bus 001 Device 002: ID ffff:3733
Bus 002 Device 001: ID 1d6b:0003
```

2.1.4 FRW 消息传输异常

问题描述

成功加载驱动后，显示消息传输超时，如图 2-7 所示。

图2-7 frw 消息超时

```
44.555781] [B1_DEV_INF0] bt_misc_dev_open:111
45.397918] frw_host_post_sync: thread id 0 TIMEOUT! msg_id 85 seq 104
45.404708] {frw_thread_dump:: thread name: wifi_frwrmsg, is_running: 1}
45.411697] {frw_thread_dump:: queue_id:0, msg_cnt:5}
45.416881] {frw_thread_dump:: queue_id:2, msg_cnt:12}
45.422234] {frw_thread_dump:: thread name: wifi_frwr_txdata, is_running: 0}
45.429455] {hmac_vap_sync::vap[2] sync msg failed[203].}
45.435536] Oh no,LinkLoss!! BSSID[A8:5A:E0:E8:XX:XX]
45.442835] hmac_single_hal_device_scan_complete:vap[2] time[10005] chan_cnt[13] chan_0[1] back[1] event[7] mode[11]
46.202284] input: 0 as /devices/virtual/input/input11
47.445905] frw_host_post_sync: thread id 0 TIMEOUT! msg_id 357 seq 41
47.455142] {frw_thread_dump:: thread name: wifi_frwrmsg, is_running: 1}
47.462535] {frw_thread_dump:: queue_id:0, msg_cnt:5}
47.467912] {frw_thread_dump:: queue_id:2, msg_cnt:13}
47.473980] {frw_thread_dump:: thread name: wifi_frwr_txdata, is_running: 0}
47.509917] frw_host_post_sync: thread id 0 TIMEOUT! msg_id 85 seq 105
47.516708] {frw_thread_dump:: thread name: wifi_frwrmsg, is_running: 1}
47.523883] {frw_thread_dump:: queue_id:0, msg_cnt:5}
47.529113] {frw_thread_dump:: queue_id:2, msg_cnt:13}
47.534416] {frw_thread_dump:: thread name: wifi_frwr_txdata, is_running: 0}
47.541811] {hmac_vap_sync::vap[2] sync msg failed[203].}
```

问题分析

驱动 Host 侧和 Device 侧传输的 MSG 消息 ID 不匹配，导致通信异常。

解决方法

检测驱动 ko 和 firmware 文件是否为同一 SDK 版本。

3 软件常见问题及定位方法

3.1 驱动加载失败

3.2 wpa_supplicant 运行失败

3.3 DebugKits 使用失败

3.1 驱动加载失败

3.1.1 提示: insmod: short read

问题描述

insmod 驱动加载失败, 提示 "insmod: short read"。

问题分析

加载驱动时, 提示 "insmod: short read", 通常是 ko 文件下载不完整导致。

解决方法

检查下载 ko 到板端的工具是否正常运行、链路是否联通。确定链路联通状态下, 重新下载驱动到板端。

3.1.2 提示: Exec format error

问题描述

insmod 驱动加载失败, 提示 "Exec format error"。

问题分析

Exec format error 通常是由于尝试运行不兼容的二进制文件或脚本文件而引起的。此外，操作系统架构和二进制文件架构不匹配也可能导致此错误。例如将 32 位交叉编译器生成的二进制文件放到 64 位系统上运行会导致 Exec format error。

解决方法

在目标开发板上输入 “uname -a” 查看板端系统信息；在 SDK 编译服务器上输入 “file xxx.ko” 查看驱动文件信息。检测两者 OS 版本、CPU 架构是否一致。

3.1.3 提示：Firmware 下载失败

问题描述

加载 wifi_soc.ko 时，下载固件时，无法找到对应 bin 文件，导致驱动加载失败，如图 3-1 所示。

图3-1 wifi 驱动加载失败

```
/komod # insmod wifi_soc.ko
[FRW] frw_timer_init enter
frw_start_hcc_service type:6, ret=0
frw_start_hcc_service type:4, ret=0
[FRW] frw_main_init etc end
plat_soc:E]check if var and value is NULL or blankplat_soc:E]check if var and value is NULL or blank
hwifi_custom_host_read_cfg_init finish!
hwifi_cfg_host_global_init_param: rx_stbc[1] ret[0]
hwifi_cfg_tcp_ack_param_init: tcp_ack_filter_enable[1] ret[0]
hwifi_cfg_tcp_ack_param_init: tcp_ack_max_num_start_process[2] ret[0]
hwifi_cfg_host_global_init_param: user_num[8] ret[0]
hwifi_cfg_scan_probe_ie_cfg_init ret 0 priv_value 1
hwifi_cfg_scan_probe_del_wps_ie_cfg_init ret 0 priv_value 1
g_scan_probe_req_del_wps_ie = 1
hwifi_cfg_random_mac_addr_scan_init ret 0 priv_value 1
hwifi_cfg_random_mac_addr_scan_oui_init ret 65535 priv_value 0
hwifi_cfg_pcap_file_len_max_cfg_init ret 0 priv_value 12
hwifi_cfg_data_sample_param_init: data_sample[0] ret[0]
hwifi_cfg_host_global_init_param: apf_enable[1] ret[0]
hwifi_cfg_host_global_init_param: llax_protocol_enable[0] ret[0]
hwifi_cfg_host_global_init_param: wow_event[15] ret[0]
hwifi_cfg_host_global_init_param: wow_enable[1] ret[0]
hwifi_cfg_host_global_init_param: smooth_phase_en[1] ret[0]
hwifi_cfg_host_global_init_param: over_ds_en[1] ret[0]
dfx_user_rd_init success!
hmac_register_pm_callback_etc:wlan_pm_get_wifi_srv_handler_etc is null
plat_soc:E]pm_svc_open::first svc open, pm_svc_power_on!
[SDIO][I]sdio state changed, tx[off=>on ],rx[off=>on ][sdio_print_state:284]
plat_soc:E]firmware download etc begin
plat_soc:E]filp open [/etc/ws73/ws73.bin] fail!!, fp= (null)
plat_soc:E][[/etc/ws73/ws73.bin] sha256sum check fail!!
plat_soc:E]parse file fail!
plat_soc:E]download firmware fail
plat_soc:W]download firmware, count [1], current time [19511]us, max time [19511]us
plat_soc:E]wait bsp ready timeout 1000 ms
```

问题分析

wifi 驱动加载时，无法找到对应的 bin 文件，导致通信设备。

解决方法

在 SDK 根目录下，修改.config 文件，编辑 firmware 文件路径。

图3-2 修改.config 文件文件路径

```
#
# Configure the loading path of files such as firmware
#
CONFIG_FIRMWARE_BIN_PATH="/etc/firmware/ws73.bin"
CONFIG_FIRMWARE_WIFICALI_PATH="/etc/firmware/wifi_calib.bin"
CONFIG_FIRMWARE_BSLECALI_PATH="/etc/firmware/btc_calib.bin"
CONFIG_FIRMWARE_WOW_PATH="/etc/firmware/wow.bin"
CONFIG_INI_FILE_PATH="/etc/firmware/ws73_cfg.ini"
# end of Configure the loading path of files such as firmware
```

固件存放位置

ini文件位置

3.1.4 提示：Unknown symbol cfg80211_xxx

问题描述

insmod wifi_soc.ko 失败，提示 “Unknown symbol cfg80211_xxx”，如图 3-3 所示。

图3-3 未识别 cfg80211 相关符号

```
/komod # insmod wifi_soc.ko
wifi_soc: Unknown symbol __cfg80211_alloc_event_skb (err 0)
wifi_soc: Unknown symbol wiphy_register (err 0)
wifi_soc: Unknown symbol cfg80211_remain_on_channel_expired (err 0)
wifi_soc: Unknown symbol cfg80211_vendor_cmd_reply (err 0)
wifi_soc: Unknown symbol cfg80211_del_sta_sinfo (err 0)
wifi_soc: Unknown symbol wiphy_unregister (err 0)
wifi_soc: Unknown symbol cfg80211_connect_bss (err 0)
wifi_soc: Unknown symbol cfg80211_ch_switch_notify (err 0)
wifi_soc: Unknown symbol cfg80211_ready_on_channel (err 0)
wifi_soc: Unknown symbol __ieee80211_get_channel (err 0)
wifi_soc: Unknown symbol wiphy_free (err 0)
wifi_soc: Unknown symbol wiphy_new_nm (err 0)
wifi_soc: Unknown symbol cfg80211_get_bss (err 0)
wifi_soc: Unknown symbol __cfg80211_send_event_skb (err 0)
wifi_soc: Unknown symbol cfg80211_disconnected (err 0)
wifi_soc: Unknown symbol cfg80211_michael_mic_failure (err 0)
wifi_soc: Unknown symbol cfg80211_scan_done (err 0)
```

问题分析

在 insmod wifi_soc.ko 前，cfg80211 模块相关符号未注册给 Linux 内核。

解决方法

请参见《WS73V100 Linux 平台驱动移植 用户指南》中的“驱动加载出错”章节。

3.1.5 提示：Unknown symbol hci_free_dev

问题描述

insmod ble_soc.ko 失败，提示“Unknown symbol hci_free_dev”，如图 3-4 所示。

图3-4 内核缺少蓝牙模块

```
[15403.297093] ble_soc: Unknown symbol hci_free_dev (err -2)
[15403.297473] ble_soc: Unknown symbol hci_alloc_dev (err -2)
[15403.297750] ble_soc: Unknown symbol hci_unregister_dev (err -2)
[15403.298017] ble_soc: Unknown symbol hci_rcv_frame (err -2)
[15403.298240] ble_soc: Unknown symbol hci_reset_dev (err -2)
[15403.298440] ble_soc: Unknown symbol hci_register_dev (err -2)
[15403.337373] ble_soc: Unknown symbol hci_free_dev (err -2)
[15403.337746] ble_soc: Unknown symbol hci_alloc_dev (err -2)
[15403.338077] ble_soc: Unknown symbol hci_unregister_dev (err -2)
[15403.338327] ble_soc: Unknown symbol hci_rcv_frame (err -2)
[15403.338558] ble_soc: Unknown symbol hci_reset_dev (err -2)
[15403.338726] ble_soc: Unknown symbol hci_register_dev (err -2)
```

问题分析

内核编译未开启蓝牙相关选项。

解决方法

重新编译 linux 内核，开启蓝牙相关选项。

- 1、进入内核编译目录，执行 make ARCH=arm CROSS_COMPILE=arm-himix100-linux- menuconfig 进入配置页。
- 2、进入 Networking support -> Bluetooth subsystem support -> Bluetooth Low Energy (LE) features 选项，输入 y 勾选。
- 3、重新编译内核镜像后烧录。

📖 说明

ARCH 为相应的处理器架构

CROSS_COMPILE 为相应的编译链信息

3.2 wpa_supplicant 运行失败

问题描述

执行 “wpa_supplicant -iwlan0 -Dnl80211 -c/etc/wpa_supplicant.conf &” 命令后，无法创建 socket 程序和后台 wpa_supplicant 通信。

解决方法

图3-5 wpa_supplicant.conf 配置文件示例

```
ctrl_interface=/var/run/wpa_supplicant #必须配置
update_config=1 #强制更新覆盖配置
```

建立 “wpa_supplicant” 通信，可依次排查下列场景：

1. 为 wpa_supplicant 添加可执行权限。
2. 执行 “wpa_supplicant” 命令时，确定指定路径下存在 “wpa_supplicant.conf” 配置文件，如上述命令中需保证 “/etc” 目录下，存在 “wpa_supplicant.conf” 文件。
3. “wpa_supplicant.conf” 文件 “ctrl_interface” 会记录对应 socket 端口存放的位置，确保该目录下具有读写权限。

3.3 DebugKits 使用失败

问题描述

网线连接 DebugKits 工具输出日志，存在如下编译问题


```
[ 460.725144] [2024:03:27 11:55:11][pid:16138,cpu1,vpluginserver][tzdriver] (16138, vpluginserver)post_proc_smc_return:smc call ret 0xffff0001,cmd ret
[ 460.746596] [pid:16138,cpu1,vpluginserver][tzdriver] (16138, vpluginserver)proc_open_session:smc call returns error,ret=0xa
[ 460.758206] [pid:16138,cpu1,vpluginserver][tzdriver] (16138, vpluginserver)open_session_and_switch_ret:open session failed,code=0xffff0001,origin=4
[ 461.217800] [pid:16145,cpu0,insmod]plat_soc:EMERG0:emerg log.
[ 461.225196] [pid:16145,cpu1,insmod]plat_soc:ALERT1:alert log.
[ 461.234187] [pid:16145,cpu1,insmod]plat_soc:CRIT2:crit log.
[ 461.240144] [pid:16145,cpu1,insmod]plat_soc:E3:err log.
[ 461.246165] [pid:16144,cpu0,ndroid.nearlink]adittr:rate limit exceeded
[ 461.253162] [pid:16145,cpu1,insmod]plat_soc:E3:custom init[plat kern log level]: 7.
[ 461.303704] [pid:16145,cpu1,insmod]plat_soc:E3:board power gpio init:succ, power_gpio_idx=[40]!
[ 461.313629] [pid:16145,cpu1,insmod]plat_soc:E3:board wakeup gpio init:succ, wkup_gpio_idx=[70]!
[ 461.329441] [pid:16145,cpu3,insmod]plat_soc:E3:tty unlocked_ioctl() ret: -14
[ 461.338208] [pid:16169,cpu0,diag_uart_rx]plat_soc:E3:diag_uart_rx task enter
[ 461.352804] [pid:16145,cpu0,insmod]plat_soc:E3:diag_log service init ok
[ 461.459919] [pid:16169,cpu0,diag_uart_rx]Unable to handle kernel NULL pointer dereference at virtual address 0000000000000000
[ 461.462596] [pid:16169,cpu0,diag_uart_rx]Mem abort info:
[ 461.467914] [pid:16169,cpu0,diag_uart_rx] ESR = 0x80000005
[ 461.473728] [pid:16169,cpu0,diag_uart_rx] EC = 0x21: IABT (current EL), IL = 32 bits
[ 461.481711] [pid:16169,cpu0,diag_uart_rx] SET = 0, RWX = 0
[ 461.487462] [pid:16169,cpu0,diag_uart_rx] EA = 0, S1PTW = 0
[ 461.493300] [pid:16169,cpu0,diag_uart_rx]user pgtable: 4k pages, 39-bit VAS, pgdp=000000011c470000
[ 461.502517] [pid:16169,cpu0,diag_uart_rx]Internal error: Oops: 86000005 [1] PREEMPT SMP
[ 461.515962] [pid:16169,cpu0,diag_uart_rx]Modules linked in: plat_soc(O)
[ 461.529969] [pid:16169,cpu0,diag_uart_rx]CPU: 0 PID: 16169 comm: diag_uart_rx Tainted: G 0 5.10.43-hi3751v811+ #3
[ 461.542081] [pid:16169,cpu0,diag_uart_rx]Hardware name: hi3751v811 (DT)
[ 461.548664] [pid:16169,cpu0,diag_uart_rx]pstate: 60000005 (nZcv daif -PAN -UAO -TCO BTYP=)
[ 461.557151] [pid:16169,cpu0,diag_uart_rx]xpc: 0x0
[ 461.561800] [pid:16169,cpu0,diag_uart_rx]x1r: 0x0000000000000000 [diag_uart_rx_task+0x1dc/0x3e8 [plat_soc
[ 461.569824] [pid:16169,cpu0,diag_uart_rx]xsp: 0xfffffc02460bce0
[ 461.575625] [pid:16169,cpu0,diag_uart_rx]x29: 0xfffffc02460be10 x28: 0xfffffd0996bd358
[ 461.583424] [pid:16169,cpu0,diag_uart_rx]x27: 0xfffffd0996bba70 x26: 0xfffffd13daa9280
[ 461.591216] [pid:16169,cpu0,diag_uart_rx]x25: 0xfffffd0996bba80 x24: 0xfffffd15722dbf8
[ 461.599008] [pid:16169,cpu0,diag_uart_rx]x23: 0xfffffd0996baee0 x22: 0xfffffd15722db80
[ 461.606801] [pid:16169,cpu0,diag_uart_rx]x21: 0xfffffd0996bb8b0 x20: 0xffffffffffffffff
[ 461.614592] [pid:16169,cpu0,diag_uart_rx]x19: 0xfffffd0996bd690 x18: 0000000000000000
[ 461.622384] [pid:16169,cpu0,diag_uart_rx]x17: 0000000000000016 x16: 0000000000000016
[ 461.630175] [pid:16169,cpu0,diag_uart_rx]x15: 0000000000000000 x14: 0000000000000400
[ 461.637967] [pid:16169,cpu0,diag_uart_rx]x13: 0000000000000004 x12: 000000028ec79ea3
```

解决方法

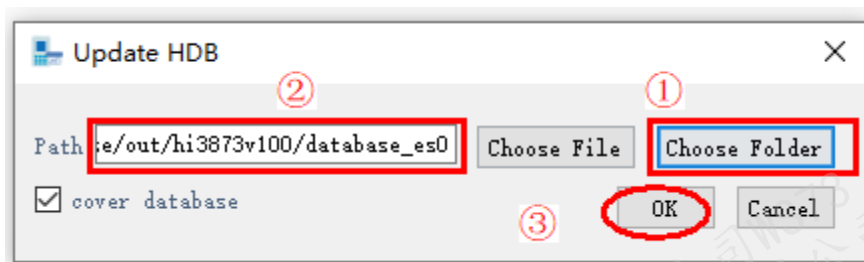
1.配置 ws73_default.cfg 中 CONFIG_DIAG_SUPPORT_SOCKET 宏为开, 关闭 CONFIG_DIAG_SUPPORT_UART 和 CONFIG_DFX_SUPPORT_SHELL_PROC 宏如下, 注意关闭宏需要配置为# xxx is not set 的形式, 直接注释掉无效。

```
119 WSCFG_ONEIMAGE=y
120 WSCFG_PLAT_VARIABLE_FEATURE=y
121 WSCFG_PLAT_DIAG_LOG_OUT=y
122 + CONFIG_DIAG_SUPPORT_SOCKET=y
123 # CONFIG_DIAG_SUPPORT_UART is not set
124 CONFIG_DIAG_SUPPORT_LOCAL_LOG=y
125 # CONFIG_DFX_SUPPORT_SHELL_PROC is not set
126 CONFIG_PLAT_SUPPORT_DFR=y
127 CONFIG_DEVICE_INIT_STANDBY=y
128 CONFIG_PLAT_DFR_OUTPUT_PATH="/etc/ws73"
```

2.网线连接单板与 PC, 将单板配置与 PC 到相同的网段, pc 与单板能相互 ping 通

3.替换日志数据库 database, Options->Update HDB->Choose Folder

注: 其中 database 由 sdk 编译时生成, 在 out 目录下文件夹名为 database_es0, database 单独编译命令为: make hso, 详见《Ws73V100 Linux 平台驱动移植 用户指南》2.3.2 章



3. 打开 DebugKits 工具, Connection->Connect, 选择 NetPort, 填写单板 IP 地址, 点击 OK 连接



其它请参见《WS73V100 DebugKits 工具 使用指南》中“连接设置”章节。

4 Wi-Fi 性能问题排查方法

说明

1. 本章节中提到的测试或调试命令均对 WS73 芯片生效，但本章节命令仅展示关键参数，CCPRIV 命令格式需使用者自行进行转换。
2. 本文中介绍的命令格式："wlan0 set_udata_rate_mode fixed"转换为 WS73 命令：echo "wlan0 set_udata_rate_mode fixed" > /sys/ccsys/ccpriv
3. 命令均以 wlan0 作为举例，实际使用中可换成实际使用的 vap 名字（如 ap0, wlan1 等）

4.1 性能问题概述

4.2 性能问题整体定位思路

4.3 常见性能问题及影响参数

4.1 性能问题概述

性能问题特征

WIFI 性能问题与功能问题相比，在现象上有较为明显的差异：

- 功能问题：一般可通过设备的异常行为直接识别出来，比如：死机、关联失败、ping 不通、断流、串口打印异常等；
- 性能问题：一般需要通过与参考对象对比才能发现，比如：在同等测试条件下与标杆进行对比，吞吐量低于标杆、跑流曲线比标杆波动大、覆盖范围比标杆差等；

性能相关测试场景

WIFI 性能相关的测试场景可以大致分为以下几种类型：

- 屏蔽环境：使用屏蔽箱、屏蔽房进行测试，测试环境内无测试相关设备外的其他信号；
- 开放环境：办公室环境、户外环境等，环境中存在其他设备或其他干扰源（WiFi 同频/邻频/叠频干扰、微波炉干扰、蓝牙干扰、LTE 干扰等）；
- 不同距离/衰减大小：近距离（低衰减）、中距离（中衰减）、远距离（高衰减）；
- 不同打流业务类型：上行/下行/双向、TCP/UDP、视频业务、混合业务等；
- 不同用户数/流数：单用户、多用户、满规格用户等；
- 不同设备工作模式：单 STA、STA+softAP、P2P、DBAC 等；

4.2 性能问题整体定位思路

如上文描述，Wi-Fi 性能的测试场景众多，不同的测试场景下，多种因素的影响下均可能产生不同的性能表现。因此，在遇到性能问题时，需快速针对该问题发生的场景、周围环境、关键影响因素进行排查，缩小该问题的定位范围。此外，由于 Wi-Fi 性能与 Wi-Fi 算法中为报文发送选择的参数紧密相关，本文将介绍一些针对关键参数的常见调试手段，可在遇到性能问题时快速进行一系列调优尝试。

4.2.1 环境排查

由于性能问题往往与周围环境有着强依赖关系，本节记录了一些常见的性能环境排查项及排查方法。

打流方式及工具排查

- 测试周期

由于环境突变，打流的测试结果同样会出现上下浮动，在环境干扰较多且变化较快的情况下该浮动会更加明显。因此，为确保测试结果的准确性，需保障待测设备与竞品测试的时间相邻，并测试多组数据，防止结果存在瞬时性而影响最终判断。

- 打流工具及命令：

使用 iperf 命令进行测试时，需关注以下几个关键参数是否配置正确。

若使用 UDP 进行测试，则需添加合适的 -b 命令（如 -b 100M），防止发送端发送的吞吐过小，整体性能不达预期。

若使用 TCP 进行测试，则无需配置 -b 命令，TCP 协议会自行进行吞吐调整；但需注意 TCP 窗口的大小（可在 iperf 发送及接收端界面上看到该值），过小的窗口同样会影

响性能。如默认使用的 TCP 窗口较小，可使用 -w 命令（如 -w 2M）来调整 TCP 窗口大小。

其他业务影响排查

因打流性能同样会受到其他业务的影响，若打流过程中出现了如下业务，需重点关注：

- 扫描

扫描业务会切离当前的工作信道，并多次切换信道并停留，极大影响当前工作信道的吞吐，因此如果 WIFI 性能出现周期性的下降或者掉 0，需要排查是否与扫描有关。比如，WS73 默认没有使能背景扫描，但是如果主控适配时打开了背景扫描业务，1 分钟会扫描一次，从吞吐上可以很明显观察到性能每分钟都有一次掉坑现象。

WS73 在扫描完成后会有如下打印，从串口打印也可以判断是否有扫描出现：

```
hmac_single_hal_device_scan_complete:vap[xx] time[xx] chan_cnt[xx] chan_0[xx]  
back[xx] event[xx] mode[xx]
```

```
[95140.906279] hmac_single_hal_device_scan_complete:vap[2] time[610] chan_cnt[13] chan_0[1] back[1] event[6] mode[1]
```

- 漫游

在弱信号场景中（rssi 弱于 -80），WS73 会有几率触发漫游，漫游业务会有切离工作信道扫描的过程，影响 WIFI 性能。因此在测试 WIFI 极限性能时，若发现当前打流过程中触发了漫游（串口中可观察到扫描相关的打印），则可以暂时改变触发漫游的条件，关闭漫游，排除漫游影响：

方法 1：使用 ccpriv 命令 - "wlan0 roam_cfg 0 -128 -128"。

方法 2：ini 文件中关闭漫游 roam_trigger_rssi_2g 配置为 -128。

- DBAC

DBAC（异信道共存），多应用于 STA+P2P GO 场景，DBAC 场景下，WS73 会不断切换信道，暂停当前信道收发，对 WIFI 极限性能有非常大的影响。通过 ifconfig 命令，查看是否存在多个收发包 VAP，并通过 DBAC 维测命令，判断当前是否处于 DBAC 状态：

```

wlan0    Link encap:Ethernet  HWaddr d2:00:73:0e:2e:f0
          inet addr:192.168.100.120  Bcast:192.168.100.255  Mask:255.255.255.0
          inet6 addr: fe80::d000:73ff:fe0e:2ef0/64 Scope: Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1006599 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1041360 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1439392694 TX bytes:1105062814

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope: Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:88 errors:0 dropped:0 overruns:0 frame:0
          TX packets:88 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:12523 TX bytes:12523

eth0      Link encap:Ethernet  HWaddr 3a:ae:43:af:5b:b0  Driver hieth
          inet addr:192.168.20.141  Bcast:192.168.20.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:17279 errors:0 dropped:2303 overruns:0 frame:0
          TX packets:875 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2667463 TX bytes:86489
          Interrupt:26

p2p0      Link encap:Ethernet  HWaddr d2:00:73:0e:31:f0
          inet addr:192.168.49.1  Bcast:192.168.49.255  Mask:255.255.255.0
          inet6 addr: fe80::d000:73ff:fe0e:31f0/64 Scope: Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 TX bytes:0
  
```

DBAC 维测: `echo "wlan0 alg_cfg dbac_stats_info" > /sys/ccsys/ccpriv`

```

console:/vendor/lib/modules # echo "wlan0 alg_cfg dbac_stats_info" > /sys/ccsy
[96067.087903] dbac type[0] state[1] pause[0] dual_sta_mode[0]
[96067.093580] timer_cnt[0], noa_start_cnt[0] noa_end_cnt[0] sync_sta_tsft_to_ap_cnt[0]
[96067.101341] netbuf_alloc_fail[0] led_tbt channel_recovery[0] noa_slot_num_err_cnt[0]
[96067.109259] vap[0], vap_tbt_isr_cnt[0], vap_tbt_channel_mismatch[0]
[96067.115797] vap[0], noa_start_bias[0], noa_end_bias[0]
[96067.120997] vap[0], ch_mismatch[0], pause[0] out_err[0] netbuf[0]
[96067.127153] vap[0], channel_switch[0], channel_preempt[0] one_packet_fail[0]
[96067.134302] vap[0], vap_tbt_isr_cnt[0], vap_tbt_channel_mismatch[0]
[96067.141222] vap[0], noa_start_bias[0], noa_end_bias[0]
[96067.146564] vap[0], ch_mismatch[0], pause[0] out_err[0] netbuf[0]
[96067.152753] vap[0], channel_switch[0], channel_preempt[0] one_packet_fail[0]
[96067.159876] OK
  
```

```

/* DBAC状态定义 */
typedef enum {
    DBAC_STATE_NOT_INIT = 0,
    DBAC_STATE_INITED = 1,
    DBAC_STATE_SWITCH_READY = 2,
    DBAC_STATE_BUTT
} alg_dbac_state_enum;
  
```

可以通过 down 掉其中一个设备, 退出 DBAC 状态: `ifconfig p2p0 down`。

周围环境排查

- 测试摆放位置: 确认与竞品摆放位置相同。
- 信号强度对比:

在确认摆放位置相同后, 预期待测设备感知到的对端信号强度同样接近。为获取 WS73 芯片感知到的对端信号强度, 可使用扫描命令 (`scan + scan_results`) 进行查看。

- 干扰信息获取：

如果在开放环境，则可通过以下命令获取环境的干扰信息：

"wlan0 alg_cfg intf_type_enquiry 1"

```
/komod #  
/komod #  
/komod # echo "wlan0 alg_cfg intf_type_enquiry 1" > /sys/ccsys/ccpriv  
coch_intf_type = 0, adjch_intf_type = 0, state = 0, noise_floor = 0  
OK.  
/komod #
```

该命令的返回值中可以读取到当前的干扰强度、干扰状态、及底噪等信息。如上图示例，从该命令的回显中，可读取到当前的同频干扰状态 coch_intf_type (0-无干扰, 1-有干扰)，临叠频干扰状态 adjch_intf_type (0-无干扰, 1-中等干扰, 2-强干扰, 3-认证/强干扰)，与当前底噪值 noise_floor (值越大表示当前空口噪声越大)。

- 温度排查：

为了保障芯片能在高温下正常工作，驱动内部在检测到芯片高温后，会进行一系列降低功耗的操作，如降低发包频率、降低功率等；此类操作均会对性能造成影响。因此，若当前测试环境温度较高（如放在温箱内，或紧挨高温物品），则需关注当前的芯片温度，判断当前芯片是否进入了过温保护模式。WS73 提供了如下命令以获取当前的芯片温度：

echo "get_temp" > /sys/ccsys/ccpriv

```
/komod # echo "get_temp" > /sys/ccsys/ccpriv  
OK  
temperature 30  
OK.
```

如果获取温度高于 100 度，则表示当前驱动可能已经启动了过温保护策略，性能会开始逐步下降。

如果获取温度高于 120 度，则表示当前温度已经逼近了芯片的最高承受温度，驱动会启动关闭 PA 的保护手段，关闭 PA 后业务会被中断，可能会出现打流掉零或断连的现象。温度下降后会恢复 PA。

硬件排查

- 单板：明确当前问题为单一芯片问题还是通用类问题（多块单板均有出现问题）。
- 天线：
排查天线的位置摆放，是否为垂直向上、高度与路由平齐、天线是否拧紧等；同时，可确认当前是否与竞品使用同一天线，或同类天线，以排除天线差异。
- 主控：明确当前问题为单一主控问题还是通用类问题。
- 电源：确保当前使用的电源规格满足芯片设计的要求。

4.2.2 缩小问题范围

在排除上述环境相关的因素初步确认问题确实存在后，为帮助后续精准定位，可通过以下几类方法快速缩小问题的范围：

协议模式

该问题影响的协议模式及带宽（如 20M/40M，legacy/HT/HE），路由器页面配置

打流模式

- 该问题主要针对发送方向 or 接收方向。
- 该问题是否仅在 TCP 打流模式时出现问题，还是同样影响 UDP；若仅影响 TCP，可调整 TCP 打流窗口大小（iperf -w 参数 eg. 窗口大小调整为 2M：-w 2M）或打流流数（iperf -P 参数 eg.打四条流：-P 4），观察性能结果是否会改变。

打流现象

观察当前问题场景下的打流结果。最终的性能结果低下可能来源于：

- 整体性能偏低，打流过程中无较高的瞬时吞吐。
- 性能波动较大，有较大的瞬时吞吐，但由于出现掉坑或掉零现象，导致最终整体的性能统计值较低。

对端设备

该项主要目的在于确认该问题是否为对接特定设备时的兼容性问题。设备可为不同型号的路由（WS73 作 STA），或不同型号的手机（WS73 作 softAP）。

- 该问题为通用问题，即更换多款对端设备时均存在该问题。
- 该问题为兼容性问题，即仅在对接特定设备时会出现性能问题，对接其余设备时表现正常；若发现为兼容性问题，则需确认以下几点：
 - 同等规格下，竞品对接该问题设备时是否存在同样的问题现象。
 - 问题设备的规格（型号、天线数、wi-fi 芯片厂商等）。

4.2.3 差异分析

通过上述环境排查及影响范围确认，已经可以基本确认问题是否存在，以及在何种场景、打流模式、协议模式下存在。接下来，就可以在上述分析中的问题场景下对问题现象进行初步的分析，并针对以下这些常见的性能影响因素进行逐一排查。

报文抓包分析

影响 Wi-Fi 性能的一大因素为报文的发包参数（如速率、聚合度等），而报文的相应发包参数可通过空口中抓到的报文体现出来。因此，通过报文抓包的差异分析，我们可以从以下几个角度排查可能会影响到性能的一些关键参数：

● 速率参数差异

抓包观察 WS73 待测设备与竞品发送的报文是否有明确的速率差异，如竞品发送的多数为 MCS7，而 WS73 使用的大多为 MCS5 等。若当前环境下无抓包工具，可使用如下命令获取当前 WS73 所使用的瞬时发送速率：

"wlan0 get_tx_params [对端 MAC 地址]"

```
cho "wlan0 get_tx_params 5e:c0:a0:8a:74:90" > /sys/ccsys/ccpriv
[96786.242863] wal_get_tx_params: user_id = 2, tx_rate = 150000 kbps
[96786.249606] OK.
```

此外，由于报文发送速率同样与报文发送的协议模式带宽等参数相关，也可通过抓包观察待测设备是否与竞品使用了相同的协议模式及带宽。

若发现待测设备与竞品使用了不同的发送参数，则可通过固定速率参数的形式，将 WS73 配置成与竞品使用相同的上述参数，并观察是否会带来性能上的提升。固定参数命令如下：

"wlan0 set_udata_rate_mode [fixed/auto]" 使用固定/动态速率模式

格式	echo "\$vap_name set_udata_rate_mode \$value" > /sys/ccsys/ccpriv
参数说明	<ul style="list-style-type: none">\$vap_name: 需要维测的vap名字，通常用wlan0等。\$value: 配置模式。<ul style="list-style-type: none">0: auto（自动速率）；1: fixed（固定速率）。
示例	echo "wlan0 set_udata_rate_mode [0 1]" > /sys/ccsys/ccpriv
响应	~
注意	<p>第一次改固定速率的时候，没有默认值，需要配合配置Wi-Fi固定速率参数使用：</p> <pre>echo "\$vap_name set_udata_fix_rate \$protocol_bw \$rate_index" > /sys/ccsys/ccpriv</pre> <p>后续改固定速率会以上一次配置的固定速率参数进行发送。</p>

"wlan0 set_udata_fix_rate [11n20M] [MCS7]" 配置想要固定的协议模式\带宽\速率

格式	echo "\$vap_name set_uda_rate \$protocol_bw \$rate_index" > /sys/ccsys/ccpriv
参数说明	<ul style="list-style-type: none"> \$vap_name: 需要维测的vap名字, 通常用wlan0等。 \$protocol_bw: 输入协议带宽的字符串。 <p>目前支持输入格式有以下列表, 对应协议和带宽: 11g、11b、11n20M、11n40M、11ax20M、11axer20M、11axer106tone</p> <ul style="list-style-type: none"> \$rate_index: 配置各个协议的速率, 具体字符串如下: <p>11b: 1M、2M、5.5M、11M</p> <p>11g: 6M、9M、12M、18M、24M、36M、48M、54M</p> <p>11n: mcs0、mcs1、mcs2、mcs3、mcs4、mcs5、mcs6、mcs7</p> <p>11ax/11axer: mcs0、mcs1、mcs2、mcs3、mcs4、mcs5、mcs6、mcs7、mcs8、mcs9</p>
示例	echo "wlan0 set_uda_rate 11b 1M" > /sys/ccsys/ccpriv
响应	串口打印协议和带宽。
注意	该命令只是配置固定速率的参数, 不会改变发送速率模式, 需要配置Wi-Fi发送速率模式为固定速率: echo "\$vap_name set_uda_rate_mode 1" > /sys/ccsys/ccpriv后才能生效。

● 聚合参数差异

若发现 WS73 与竞品使用的发送参数无明显差异, 但性能依旧低于竞品时, 可进一步排查聚合度大小 (即一个 ampdu 中有多少 mpdu) 对其性能大小的影响。如果大多数情况下, WS73 发送报文的聚合度与竞品相差不大, 那么可以排除聚合度的影响。

需要说明的是, 虽然在理论情况下聚合个数的增多可带来性能的收益, 但针对 IoT 这类运行资源受限的设备而言, 由于聚合个数同样受报文在队列的最大缓存个数、聚合窗口大小、与整体的资源运转流水相关, 较低的聚合度可能可以带来更稳定更高的性能。若通过比较发现竞品实用的聚合个数较低, 且想限制当前的最大聚合个数, 可通过 ini 或 NV 进行限制:

-- ini 文件中限制最大聚合个数 配置 ampdu_max_tx_mpdu_num

除上述说明的 AMPDU 聚合之外, 如当前发送的报文包长较短 (<128 字节), 则 Wi-Fi 驱动可能会将小包报文进行 AMSDU 聚合处理以试图提升性能。在 TCP 打流的 RX 方向下, 由于 WS73 需回复 TCP ack 报文, 且该报文多为小包, 可重点关注该影响。在排查该 AMSDU 聚合影响时, 可通过 ini 或 NV 将 AMSDU 功能关闭, 并进行性能对比, 关闭方法如下:

-- ini 文件中关闭 amsdu 功能 配置 amsdu_tx_on 为 0

● RTS 开启比例

除报文的发送参数与聚合表现外, 抓包中也可以看出 WS73 与竞品发送 RTS 的行为差异, 如 WS73 仅发送了少量的 RTS, 而竞品几乎每次发送数据帧前均发了 RTS, 或是 WS73 频繁发送 RTS, 而竞品仅发送了少量的 RTS。若报文抓包中存在以上差异, 则可以使用如下命令对比强制开启 RTS 或关闭 RTS 时的性能表现:

"wlan0 alg_cfg rts_mode [auto/enable/disable]" 强制开启 RTS 或关闭 RTS

格式	echo "\$vap_name alg_cfg rts_mode \$arg" > /sys/ccsys/ccpriv
参数说明	<ul style="list-style-type: none"> \$vap_name: 需要维测的vap名字, 通常用wlan0等。 \$arg: 配置RTS的模式: auto、enable、disable、rank0off、threshold。 auto: 速率等级0动态调整; enable: 所有速率等级强制开启RTS; disable: 所有速率等级关闭RTS; rank0off: 速率等级0关闭RTS; threshold: 所有速率等级的RTS开关由MIB值dot11RTSThreshold来决策。
示例	echo "wlan0 alg_cfg rts_mode enable" > /sys/ccsys/ccpriv
响应	~
注意	如果最后一个参数配成mib, 可以用配置RTS阈值: echo "\$vap_name rts_threshold \$value" > /sys/ccsys/ccpriv进行改变dot11RTSThreshold值。

发送功率影响

除上述介绍的几个因素之外, 报文发送功率的大小同样会影响性能, 尤其是在干扰或是远距离场景下。因此, 针对报文的发送功率, WS73 提供了下列两种方式进行功率调试: 1) 通过命令在现有基础上调整功率大小; 2) 通过修改配置文件的方式修改默认功率配置。

- 命令方式:

"wlan0 set_cal_rpwr \$protocol \$rate \$offset" 参数说明及配置方法见下图

格式	echo "\$vap_name set_cal_rpwr \$protocol \$rate \$offset" > /sys/ccsys/ccpriv
参数说明	<ul style="list-style-type: none"> \$vap_name: 需要维测的vap名字, 通常用wlan0等。 \$protocol: 设置业务的协议模式。 0: 11b协议; 1: 11g协议; 2: 11n/11ax 20M协议; 3: 11n40M协议。 \$rate: 设置速率索引。 11b: 0~3 表示1, 2, 5.5, 11Mbps; 4 表示全速率修改; 11g: 0~7 表示6, 9, 12, 18, 24, 36, 48, 54Mbps; 8 表示全速率修改; 11n/11ax 20M: 0~9 表示mcs0~mcs9; 10 表示全速率修改; 11n 40M: 0~9 表示mcs0~mcs9, 10 表示mcs32; 11 表示全速率修改。 \$offset: 设置功率偏移-100~+40, 单位0.1dB。
示例	echo "wlan0 set_cal_rpwr 2 6 20" > /sys/ccsys/ccpriv
响应	<ul style="list-style-type: none"> 成功: OK 失败: INPUT_ERROR or CMD_NOT_FOUND
注意	- 11n 20M/40M 实际支持到mcs7。

使用上述命令, 可实时针对特定协议模式及速率进行功率调整, 调大 1db 或调小 1db, 观察功率变化对性能的影响。

- 定制化文件方式:

可修改 ini 中的默认功率配置值

```
119 # max power 0.1dBm
120 rf_chip_max_power_2g=230;
121 # target power 0.5dBm
122 ## 1M 2M 5.5M 11M 11b
123 target_tx_power_2g_11b=0x2E,0x2E,0x2E,0x2B;
124 ## 6M 9M 12M 18M 24M 36M 48M 54M 11g
125 target_tx_power_2g_11g=0x2A,0x2A,0x2A,0x2A,0x2A,0x2A,0x28,0x26;
126 ## MCS0~MCS9 11n(MCS7)/11ax
127 target_tx_power_2g_20m=0x28,0x28,0x28,0x25,0x25,0x25,0x25,0x24,0x21,0x1E;
128 ## MCS0~MCS9 11n(MCS7)/11ax MCS32 11n
129 target_tx_power_2g_40m=0x28,0x28,0x28,0x25,0x25,0x25,0x25,0x24,0x21,0x1E,0x16;
```

4.2.4 干扰场景

干扰同样是影响 Wi-Fi 性能的一个关键因素。如果当前测试场景是在屏蔽箱内或屏蔽环境中，可以忽略本章节内容。

若当前测试环境为开放环境，或环境中存在其他的 Wi-Fi 测试设备或可能产生干扰信号的设备（如微波炉、蓝牙、LTE 等），则需要记录当前的环境干扰信息，并根据 WS73 提供的几条常见的抗干扰调试命令，结合当前的干扰情况进行初步的抗干扰调试。

干扰信息获取

可在 ping 包的同时通过如下命令获取芯片采集到的干扰信息：

"wlan0 alg_cfg intf_det_enquiry 1"

```
/komod #
/komod #
/komod # echo "wlan0 alg_cfg intf_type_enquiry 1" | sys/ccsys/ccpriv
coch_intf_type = 0, adjch_intf_type = 0, state = 0, noise_floor = 0
OK.
/komod #
```

如上图示例，从该命令的回显中，可读取到当前的同频干扰状态 `coch_intf_type`（0-无干扰，1-有干扰），临叠频干扰状态 `adjch_intf_type`（0-无干扰，1-中等干扰，2-强干扰，3-认证/强干扰），与当前底噪值 `noise_floor`（值越大表示当前空口噪声越大）。

MAC 抗干扰相关参数调试

- EDCA 退避参数调试

EDCA 机制，通过控制报文发送至空口前的退避相关参数（如 `CWmin/max`，`Aifsn`，`txop` 等），可影响报文的抢占空口能力。一般情况下，为防止过多的空口碰撞，EDCA 参数不建议配置过于激进（参数数值更小）；但当空口出现竞争时，更为激进的 EDCA 参数配置可能会增大设备成功抢到空口的机会，从而提升竞争下的性能。因此，WS73

提供了 EDCA 相关的抗干扰模式命令，可供在存在较强同频干扰的测试环境下调试使用：

"wlan0 alg_intrf_mode edca_switch 1" 开启或关闭 EDCA 抗干扰模式

格式	echo "\$vap_name alg_intrf_mode \$intrf_mode \$value" > /sys/ccsys/ccpriv
参数说明	<ul style="list-style-type: none">\$vap_name: 需要维测的vap名字，通常用wlan0等。\$intrf_mode: 干扰模式，字符串列表如下： 11b_switch、cca_switch、edca_switch、11n_switch、no_11b_switch、long_range_intrf_switch\$value: <ul style="list-style-type: none">0: 关闭;1: 开启。
示例	echo "wlan0 alg_intrf_mode 11b_switch 1" > /sys/ccsys/ccpriv
响应	~
注意	~

此外，WS73 还提供了两套 EDCA 相关命令，一套可用于读取当前使用的 EDCA 各项参数数值，一套可用于针对相应参数进行配置：

"wlan0 get_edca_params \$para \$prio"

格式	echo "\$vap_name get_edca_params \$para \$prio" > /sys/ccsys/ccpriv
参数说明	<ul style="list-style-type: none">\$vap_name: 需要维测的vap名字，通常用wlan0。\$para: 查询EDCA BSS广播参数、MIB值和寄存器值，包括AIFS、CwMin、CwMax、TXOP、qAIFS、qCwMin、qCwMax、qTXOP。\$prio: 配置\$para对应的优先级，可配置0/1/2/3，表示BE/BK/VI/VO队列。
示例	echo "wlan0 get_edca_params cwmin 1" > /sys/ccsys/ccpriv
响应	<ul style="list-style-type: none">成功: OK失败: INPUT_ERROR or CMD_NOT_FOUND
注意	~

"wlan0 set_edca_params \$para \$prio \$val"

格式	echo "\$vap_name set_edca_params \$para \$prio \$val" > /sys/ccsys/ccpriv
参数说明	<ul style="list-style-type: none">\$vap_name: 需要维测的vap名字，通常用wlan0。\$para: 配置EDCA BSS广播参数，可配置EDCA BSS广播参数，包括AIFS、CwMin、CwMax、TXOP；可配置EDCA参数MIB值和寄存器值，包括qAIFS、qCwMin、qCwMax、qTXOP。\$prio: 配置\$para对应的优先级，可配置0/1/2/3，表示BE/BK/VI/VO队列。\$val: <ul style="list-style-type: none">AIFS配置范围: 0~15;CwMin配置范围: 0~10;CwMax配置范围: 0~10;TXOP配置范围: 0~65535。
示例	echo "wlan0 set_edca_params cwmin 1 3" > /sys/ccsys/ccpriv
响应	<ul style="list-style-type: none">成功: OK失败: INPUT_ERROR or CMD_NOT_FOUND
注意	~

● CCA 阈值调试

CCA 门限主要作用于控制芯片发送报文时的检测，若当前环境中存在强能量，CCA 机制会判断当前信道的空口条件较差，减少自身的发包机会；即只有检测到当前能量值小于该阈值时，芯片才会发送报文。因此，若通过上述的干扰信息获取发现当前环境下存在较高的底噪值（如>-50）时，则可以关注 CCA 阈值对干扰性能的影响，进行 CCA 相关的抗干扰模式调试：

"wlan0 alg_intrf_mode cca_switch [1/0]" 开启或关闭 CCA 抗干扰模式

格式	echo "\$vap_name alg_intrf_mode \$intrf_mode \$value" > /sys/ccsys/ccpriv
参数说明	<ul style="list-style-type: none">• \$vap_name: 需要维测的vap名字, 通常用wlan0等。• \$intrf_mode: 干扰模式, 字符串列表如下: 11b_switch、cca_switch、edca_switch、11n_switch、no_11b_switch、long_range_intrf_switch• \$value: 0: 关闭; 1: 开启。
示例	echo "wlan0 alg_intrf_mode 11b_switch 1" > /sys/ccsys/ccpriv
响应	~
注意	~

4.3 常见性能问题及影响参数

4.3.1 TCP RX 打流性能波动较大

- 适用场景: TCP RX 打流出现明显波动, 但其他打流场景表现正常 (如 TCP TX、UDP T/RX 等)
- 尝试方法: 关闭 ini 配置文件中的 TCP ACK FILTER 功能
(tcp_ack_filter_enable=1 改为 tcp_ack_filter_enable=0)

该配置项为 WiFi 驱动中的 TCP ACK 帧过滤功能, 其通过过滤 TX 方向 TCP ACK 的数量来减轻空口回复 TCP ACK 报文的压力, 从而获取一定性能上的提升。但是, 延迟发送甚至丢弃部分 TCP ACK 报文, 可能会使得 TCP 发送端 (即对端) 无法及时获取本端 TCP 报文发送情况的反馈, 从而导致性能产生波动。

4.3.2 TCP RX 打流性能异常低下

- 适用场景: TCP RX 打流性能异常低下, 但其他打流场景表现正常 (如 TCP TX、UDP T/RX 等)
- 尝试方法: 关闭 ini 配置文件中的 AMSDU 功能 (amsdu_tx_on=1 改为 amsdu_tx_on=0)

该配置项为 WiFi 驱动中的 AMSDU 帧聚合功能。在 TCP RX 场景下, 该功能可通过将一些小包聚合起来成为一个大包一起发送来减少空口中发送 TCP ACK 报文的次数, 从而获取一定性能上的提升。该功能为协议要求项, 正常情况下不会存在问题, 但是如果内核适配存在问题, 导致 AMSDU 报文上层解析失败的话, 驱动发送的 TCP ACK 报文会传输失败, 导致打流出现异常。

(*若确认打流异常与该项配置相关, 说明内核适配存在问题, 需及时修复, 修复后需重新开启 AMSDU 功能。)

4.3.3 弱信号/远距离场景性能较低

- 适用场景：弱信号或远距离下进行性能相关业务（打流、数据传输、出图等）
- 尝试方法：修改 ini 配置文件中的漫游触发门限值（roam_trigger_rssi_2g=-78 改为 roam_trigger_rssi_2g=-128）

该配置为漫游触发的门限值，即当 rssi 低于当前配置值后，驱动会开启漫游扫描，尝试漫游关联至信号更强的 AP。因扫描会对性能产生影响，在弱信号或是远距离场景下可考虑关闭漫游。（*若当前产品默认未包含漫游功能，则当前配置项不会生效，无需修改）

4.3.4 UDP/TCP RX 打流性能较低

- 适用场景：屏蔽环境下，UDP 或 TCP RX 打流性能较低，且通过 cat /proc/net/snmp 发现内核丢包较多
- 尝试方法：通过调大下列内核参数来增大内存设置（*调整前需先观察一下原本的默认值，具体设置的值可根据原本默认值做适当调整），如：

```
sysctl -w net.ipv4.udp_mem="858 1145 1716"  
sysctl -w net.core.rmem_default=716800  
sysctl -w net.core.rmem_max=716800  
sysctl -w net.ipv4.tcp_rmem="4096 87380 444384"
```

RX 打流过程中，除开空口丢包、驱动内部丢包之外，内核中也可能会存在丢包，一般为内核设置的接收内存大小不足导致。若在 cat /proc/net/snmp 中发现内核中存在较多丢包，则可以尝试临时调整上述内存参数查看性能是否上升。

4.3.5 打流过程中 CPU 占用较高

- 适用场景：打流过程中发现 hcc_ap 线程占用较高 CPU（如>50%），或打流已经停止但该线程 CPU 占用依旧很高
- 尝试方法：修改 hcc_flow_ctrl.c 中的 hcc_flow_ctrl_credit_update 函数，在发现 remain_pkt_nums 为 0 时（即 Device 侧报文出现拥塞时），调用 osal_msleep 接口主动让出线程

在 Device 侧内存空闲时，HCC 线程可以顺畅往 Device 侧发包；但当 Device 侧由于空口环境变差出现拥塞时，HCC 无法及时将报文发往 Device 侧。当前方案下，HCC 会通过循环一直尝试发包，试图能在 Device 侧出现空闲时第一时间将报文送至 Device 侧。然而，在 Device 侧拥塞严重时，Device 侧内存存在短时间内不会出现空闲，

则循环中的大部分尝试都将是无效操作，而一直等到 Device 侧空时 HCC 才会让出线程。因此，该方案下容易出现 CPU 占用较高的问题。

深圳市安信可科技有限公司