1. **概述**

RK平台wifi部分在原生android基础上有三种适配形式：一种是静态适配，在编译时通过对android平台层进行配置，编译时根据配置的wifi模块类型，通过宏的形式控制执行代码流程；第二种是动态适配，适配层dts中配置的wifi\_type，在hal层根据wifi\_type 加载不同的driver，启动不同的wpa服务；

Framework

WifiNative

JNI()

Hal层

lib\_driver\_cmd\_bcmdhd

libhardware\_legacy

libwifi-hal-bcm

Wpa

内核层

80211

Wifi驱动

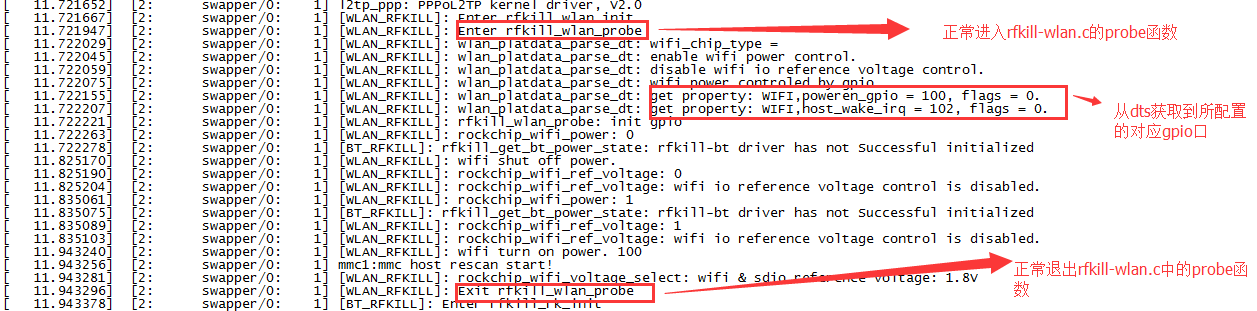
1. **配置**

参看相关sdk下的 配置说明

1. **调测**

* 确保平台或者dts中配置起效

dts文件的配置wifi部分是在net/rfkill-wlan.c中进行配置；先通过内核启动日志确认相关配置是否有正常解析，如果解析过程出现异常，确认是所配置的gpio是否存在冲突；



* 检查供电是否正常

确认wifi的供电控制是否受控

Echo 0 > /sys/class/rkwifi/power //对wifi模块掉电

Echo 1 > /sys/class/rkwifi/power//对wifi模块上电

如果执行上面命令对模块进行上下电，而 实际测量对应管脚不受控，可以通过io 命令读取对应的寄存器，确认是否写入，如果正确写入但是实际测量不受控请检查硬件部分；

* 扫描模块初始化模块

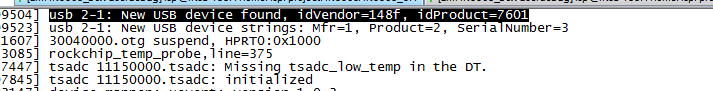
检查内核中是否配置CONFIG\_WIFI\_LOAD\_DRIVER\_WHEN\_KERNEL\_BOOTUP=y；调测wifi时请把该宏配置为n；

执行 echo 1 > /sys/class/rkwifi/driver 命令会调用模块的驱动的初始化操作，初始化成功后看到wlan0 节点；

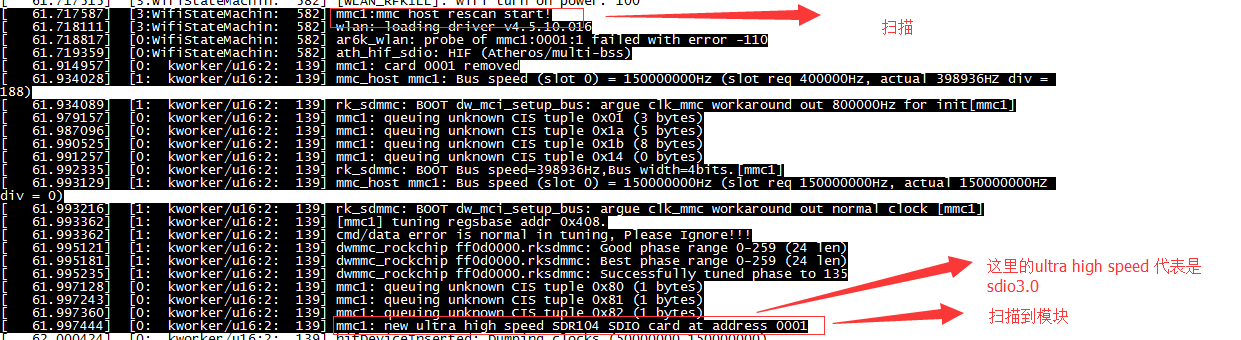
如何判断是否识别到模块

Usb接口的模块

对于usb接口的模块成功识别到模块，能够看到类似如下截图的信息

 sdio接口的模块，

对于sdio接口的模块，执行” echo 1 > /sys/class/rkwifi/driver”命令 ，正常情况下 sdio\_clk 和sdio\_cmd 能够测量到相关波形，内核打印上能够看到如下打印，如果没有测量到波形也没有看到如下打印，根据配置文档检查是否正确配置sdio;



Wifi驱动会根据扫描到的sdio模块的vid pid 进行驱动匹配，rtl的驱动会根据读取到的vid，pid进行驱动匹配；其中正基系列的模块会根据后面从data数据线上读取到F1 function 读取的数值进行 驱动与固件匹配（正基目前的驱动兼容所有sdio接口正基模块，根据F1 function 读取的值 匹配固件）；

如果能够扫描模块但是初始化过程看到data fifo error,检查下 sdio接口电平是否一致；方法如下：

echo 1 > /sys/class/rkwifi/power

测量 VDDIO sdio\_clk sdio\_cmd sdio\_data0~sdio\_data3 的电压；正常情况下 sdio\_clk 为 0V，sdio其他五根线与vddio电压一致；

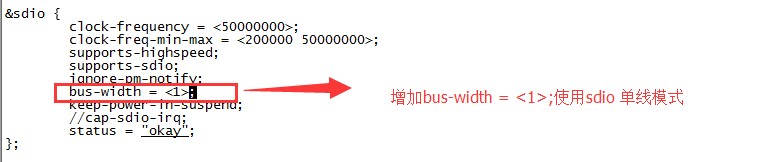
如果电压不一致：312x平台确认下 sdio接口的内部上下拉是否禁掉，参看文档RK Kernel 3.10平台WiFi BT不工作异常排查.pdf Part C；其他平台考虑加外部上拉(注clk绝对不要加外部上拉)；

同时测量执行echo 1 > /sys/class/rkwifi/driver 时 外部晶体是否有起振，如果扫描时没有起振检查下硬件；同时建议测量外部晶体频偏，频偏比较大情况下，会出现能扫描到模块但是初始化失败；除检查晶振外，正基系列还需要外部32k，测量32k的峰峰值（峰峰值>=0.7\*VDDIO && 峰峰值 <= 1\*VDDIO）；【注：频偏和峰峰值一定要测量检查，频偏过大峰峰值不对会影响wifi（扫描连接热点）和蓝牙（扫描连接设备））】

电压一致情况下，晶振频偏和32k的峰峰值没有问题（正基系列的要考虑晶振频偏与32k峰峰值，具体结合自己电路实际情况）但是初始化依然出问题；

考虑降低sdio\_clk ，重新测试；如果降低clk可以，考虑硬件上走线；

如果降低clk依然不行，考虑使用sdio单线模式方法如下



如果单线模式可以而使用4线模式不行，检查硬件上sdio\_data0~sdio\_data3 四根线的线序是否弄错；

如果降低clk，使用单线模式均不可以检查下是否是使用最新的sdk代码和最新的wifi驱动（ftp服务器上有相关patch）；

上述检查均无结果，check 图纸 是否周围器件有贴错器件；

* 检查模块能否处于工作状态

netcfg wlan0 up 或busybox ifconfig wlan0 up //执行完成后检查 wlan0 是否处于up状态；如果没有处于up状态;做如下检查确认

1 确认相关固件是否存在(正基系列，通过看内核日志可以看到)，固件不存在考虑到ftp下载固件；此时如果还报其他错误从两个方面排查1 上电时序，2检查sdio部分走线；

2 尝试使用原始最新的sdk代码做测试；（有客户出现过，上层做了相关修改导致wifi初始化成功，但是执行netcfg wlan0 up 报告无法识别 ioctl 命令等奇怪错误，原生sdk生成的sysytem.img 没有问题）

执行iwlist wlan0 scanning ，测试扫描热点是否正常（3368平台下执行iwlist 命令有问题，忽略此步骤）

* 确认Android层是否能够打开

上述检查各个步骤可以工作，而通过上层settings界面打开失败；以下几个方面排查

1 dts中的wifi\_type配置是否正确;cat /sys/class/rkwifi/chip 确认 下 打印的结果和你的模块是否匹配

2 确认 wpa\_supplicant 相关服务是否生成，libhardware\_leacy 启动的wpa服务是否正确；

3 抓取logcat 日志上传readmine

1. **其他功能性问题**

1 连接热点问题

无法连接热点，正基系列模块检查确认晶振频偏和32k峰峰值；rtl模块考虑驱动配置是否正确，是否匹配；

2 检查确认p2p wlan0 的mac地址是否一致如果检查是否有调用rockchip\_wifi\_mac\_addr读取mac地址，如果有考虑直接在该函数中return -1；

3检查确认是否有做RF指标测试以及天线匹配测试

4上述检查没有问题，做如下测试 （首先用给手机连接所测试的热点做确认）1 连接无加密热点 2 连接加密热点 测试能否连接成功，并记录对应的logcat 日志与内核日志（开机到打开wifi以及连接热点的整个过程）

2 softap 无法打开（正基系列的）

1 查看打开热点时的内核日志，确认下 下载固件是否正确 ，正基系列的模块 softap 下载的固件一般是带ap后缀结尾的；

2 固件下载没有问题 ，考虑使用原始的sdk代码做测试

3 P2p 问题

P2p 无法打开，确认是否有p2p节点，有p2p节点的检查确认mac地址是否与wlan0 一样，如果一样按照热点问题中的step 2 处理；

3 第一次开机能够打开，重启后无法打开：考虑检查上电时序，目前遇到都是rtl的模块出现过，问题在于chipen 脚不受控，建议做成受控，在重启时对chipen脚下电；可以通过如下方法实现net/rfkill-wlan.c中的rfkill\_wlan\_driver 中增加shutdown函数 在该函数中对chip\_en 下电；

4 休眠唤醒出现wifi无法打开

1 对比检查休眠唤醒前后 sdio 的iomux 是否发生变更

2 对比 休眠前后以及休眠中 wifi的外围供电是否发生变更

1. **吞吐率问题**
2. pcb检查，一定要让模块原厂检查确认 pcb是否存在问题
3. RF指标确认是否ok
4. 天线是否做过匹配
5. Sdio 接口的可以考虑 提到sdio的clk 启用sdio3.0【前提 平台支持 sdio3.0 ,模块支持sdio3.0】

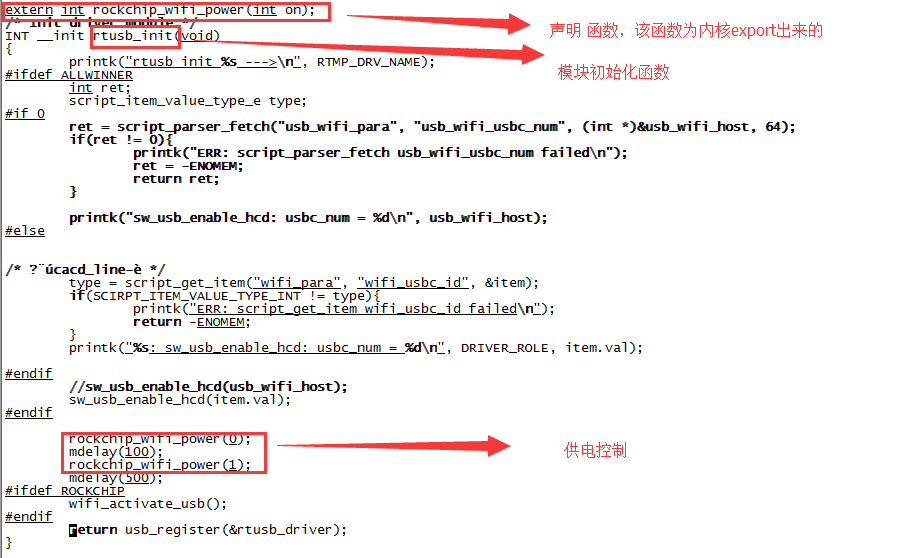
**六 新模块移植**

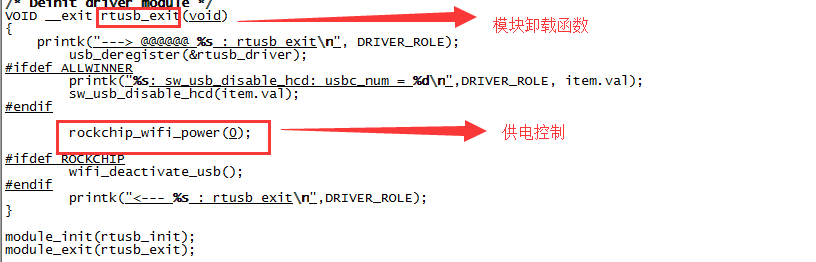
* 内核层

1 驱动层移植请按照模块厂商给出的porting文档进行处理，同时记得在模块驱动层的初始化函数入口地方记得加入对模块供电函数的调用，生成的ko文件放置在device/rockchip/common/wifi/lib/modules(请根据自己平台进行相关确认)

搜索入口的方法 搜索” module\_init”

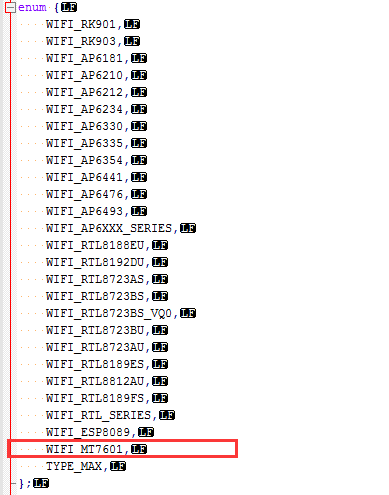
以MT7601为例



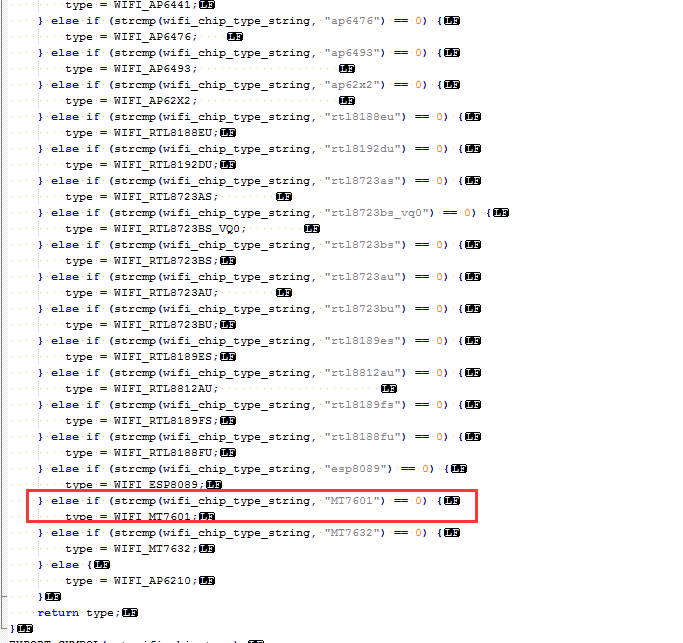


2 增加对新模块类型的识别，

kernel\include\linux\rfkill-wlan.h



net\rfkill\rfkill-wlan.c中int get\_wifi\_chip\_type(void)函数中对



* Hal移植

Hal层移植主要在hardware/libhardware\_legacey/wifi/目录下进行

主要修改以下函数

wifi\_load\_driver() 实现加载驱动

int wifi\_start\_supplicant(int p2p\_supported) 实现启动wpa服务

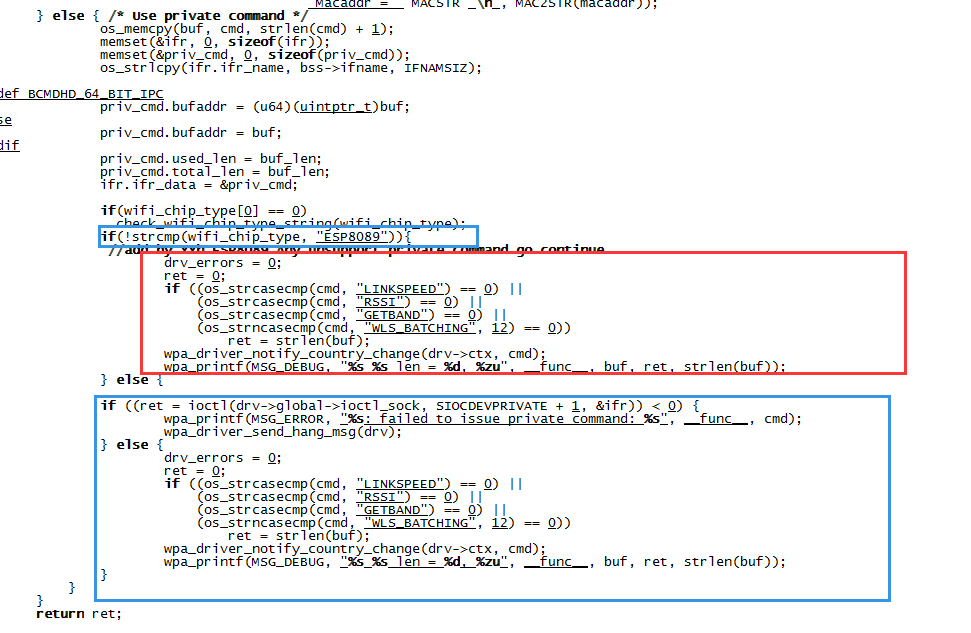
int wifi\_stop\_supplicant(int p2p\_supported) 停止wpa服务

int wifi\_unload\_driver() 卸载driver

部分厂商驱动兼容性不够好，出现通过上层连接热点时，settings界面出现wifi不断开关的情况；大部分在于厂商驱动后没有处理SIOCDEVPRIVATE + 1 命令；此时我们可以对lib\_driver\_cmd\_bcmdhd 做兼容修正根据wifi\_type 进行修正；如下示例

int wpa\_driver\_nl80211\_driver\_cmd(void \*priv, char \*cmd, char \*buf,

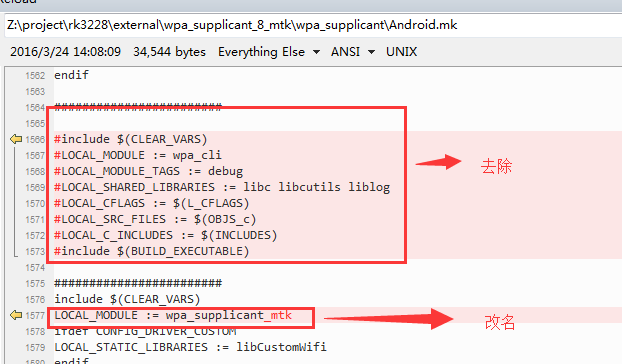
size\_t buf\_len )

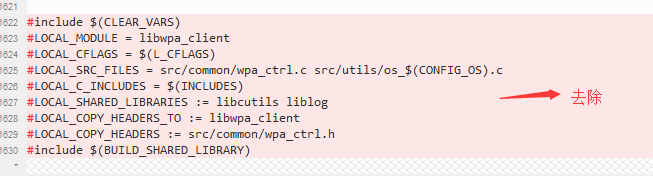


* wpa 部分

建议默认先尝试使用android自带原生的wpa，如果自带wpa使用存在问题再尝试使用模块厂商提供的wpa；

该部分主要修改部分mk文件消除编译的冲突



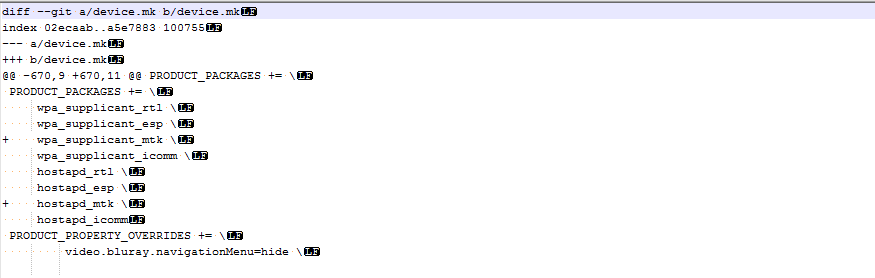




Wpa部分如果单独编译没有问题，但是整个系统整体编译出现wpa部分的报l2\_packet\_linux.c已定义的错误，可以通过修正wpa\_supplicnat/Android.mk中做如下修正来消除



* 指定系统编译时编译wpa服务与rc脚本中声明服务

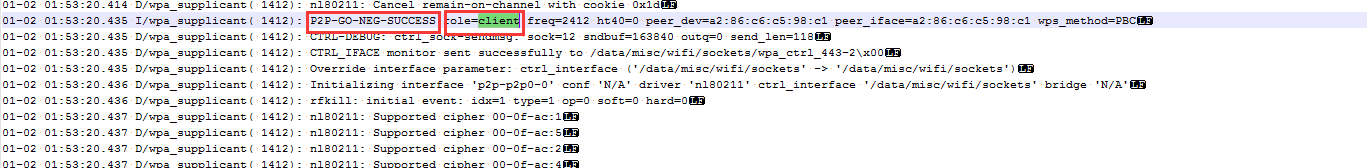




* 其他

部分驱动p2p问题，部分厂商的驱动移植完成后打开p2p 无法connect,通过

wpa\_cli IFNAME=wlan0 log\_level excessive 放开wpa部分的全部日志，发现在连接时会看到p2p-p2p0-0: Failed to initialize driver interface 类似错误；并且能够看到类似字样



可以通过尝试如下方法fix

1 确认 libhardware\_legacey中启动的wpa服务是否是你对应的服务，如果确认是对应的服务，查看rc脚本中的服务声明中，尝试 去掉 -puse\_p2p\_group\_interface=1 参数