

密级状态：绝密() 秘密() 内部() 公开(☒)

RK Android WiFi BT 配置及常见问题说明

(技术部, MID 组)

文件状态： [] 正在修改 [<input checked="" type="checkbox"/>] 正式发布	当前版本：	V1.3
	作 者：	胡卫国、高伟龙，陈美友
	完成日期：	2013-05-26
	审 核：	
	完成日期：	2013-05-26

福州瑞芯微电子有限公司

Fuzhou Rockchips Semiconductor Co., Ltd

(版本所有, 翻版必究)

版本历史

版本号	作者	修改日期	修改说明	备注
V1.0	高伟龙	2013-2-28	初始版本	
V1.1	陈美友	2013-5-23	增加 Bluetooth 部分	
V1.2	胡卫国	2013-5-26	修改整理	
V1.3	胡卫国			

目 录

1 RK Android 平台 WiFi 架构.....	5
1.1 Android WiFi 基本架构.....	5
1.1.1 Android 平台 WiFi 的层次结构.....	5
1.1.2 开源库 wpa_supplicant.....	6
1.2 RK 平台 WiFi 架构.....	7
1.2.1 WiFi 驱动.....	7
1.2.1.1 驱动编译.....	7
1.2.1.2 WiFi 电源控制.....	8
1.2.2 WiFi Firmware.....	9
1.2.3 WiFi Chip Type 节点.....	10
1.2.4 WiFi 硬件抽象层.....	10
1.2.5 WiFi PCBA 测试.....	10
2 RK Android Bluetooth 基本架构.....	11
2.1 Android 4.1.....	11
2.1.1 源码目录说明.....	11
2.1.2 Android Bluetooth 架构图.....	11
2.1.3 RK 平台 Bluetooth 具体配置.....	12
2.1.3.1 Bluetooth 驱动.....	12
2.1.3.2 Bluetooth 电源控制.....	13
3 RK 平台 WiFi BT 具体配置.....	16
3.1 kernel WiFi 框架相关部分.....	16
3.1.1 WiFi 接口驱动配置.....	16
3.1.2 WiFi 网络协议栈配置.....	16
3.2 Kernel 蓝牙框架相关部分.....	17
3.2.1 蓝牙接口驱动配置.....	17
3.2.2 蓝牙框架部分配置.....	17
3.3 RK903 & RK901 & AP6xxx 系列配置.....	17
3.3.1 Kernel Memuconfig 配置.....	18
3.3.2 Android 部分配置.....	19
3.3.3 Android 4.2 支持 BT4.0 配置（可选）.....	20
3.4 RTL8188 配置.....	20
3.4.1 Kernel Memuconfig 配置.....	21
3.4.2 Android 部分配置.....	21
3.5 MT5931&MT6622 配置.....	22
3.5.1 Kernel Memuconfig 配置.....	22
3.5.2 模块电源脚配置.....	23
3.5.3 Android 部分配置.....	23
3.5.4 串口硬件流控配置.....	25

3.6 MT7601 配置.....	26
3.6.1 Kernel Memuconfig 配置.....	26
3.6.2 Android 部分配置.....	26
3.7 RTL8723AU(USB 接口)配置.....	27
3.7.1 Kernel Memuconfig 配置.....	27
3.7.2 Android 部分配置.....	28
3.8 RTL8723AS(SDIO 接口)配置.....	29
3.8.1 Kernel Memuconfig 配置.....	29
3.8.2 Android 部分配置.....	30
3.9 ESP8089 配置.....	31
3.9.1 Kernel menuconfig.....	31
3.9.2 Android 配置.....	34
3.10 RDA5876 配置.....	35
3.10.1 Kernel menuconfig.....	35
3.10.2 电源控制.....	35
3.10.3 Android 配置.....	36
3.11 BK3515 蓝牙配置.....	36
3.11.1 Kernel menuconfig.....	36
3.11.2 蓝牙 GPIO 配置.....	37
3.11.3 UART clock 设置.....	39
3.11.4 Android 配置.....	39
3.11.5 BK3515A 支持 2M bps.....	39
4 RK 平台 WiFi BT 问题汇总.....	41
4.1 平台 WiFi 公共性问题汇总.....	41
4.1.1 状态栏 WiFi 信号显示的颜色.....	41
4.1.2 WiFi 打开死机.....	41
4.1.3 WiFi 休眠策略.....	42
4.1.4 机器休眠时后台文件下载中断.....	42
4.1.5 机器休眠再唤醒后 WiFi 无法使用.....	43
4.1.6 使用 SDIO 接口 WiFi 机器不断重启.....	43
4.2 RK903 & RK901 & AP6xxx 问题汇总.....	44
4.2.1 设置中 WiFi 打不开.....	44
4.2.2 信号强度差.....	45
4.2.3 RK903 设置中可以打开 WiFi 但是扫描列表中没有扫描到 AP.....	46
4.2.4 WiFi 使用中出现异常不能连接 AP，需要关闭再打开才能恢复.....	46
4.2.5 WiFi 热点打不开.....	48
4.2.6 WiFi 连接不上某些 AP 或上网速度很慢.....	48
4.2.7 BT 能够描述到设备但是无法匹配上.....	48
4.2.8 BPLUS 导致的 CTS 测试失败.....	48
4.2.9 AP6476 打开失败.....	49

4.2.10 AP6330 打开失败.....	49
4.3 RTL8188CUS & RTL8188EUS 问题汇总.....	49
4.3.1 设置中 WiFi 打不开.....	49
4.3.2 设置中打开 WiFi 很慢（十几秒钟）.....	50
4.3.3 WiFi 热点打不开.....	50
4.3.4 信号强度差、扫描的 AP 数量较少.....	51
4.3.5 信号较差时上网慢.....	51
4.4 RT5370 & MT7601 问题汇总.....	52
4.5 MT5931 & MT6622 问题汇总.....	53
4.5.1 设置中 WiFi 打不开.....	53
4.5.2 BT 打不开.....	53
4.5.3 BT 工作不稳定.....	54
4.6 ESP8089 & BK3515 问题汇总.....	55
4.6.1 蓝牙打开失败.....	55
4.6.2 蓝牙不能收发文件，听音乐卡.....	55
4.7 RK Bluetooth(Android 4.1/4.2)问题汇总.....	55
4.7.1 前提.....	55
4.7.2 命令行启动蓝牙.....	56
4.7.3 蓝牙打开失败.....	58
4.7.4 蓝牙 UART 通信失败.....	59
4.7.5 蓝牙设备配对失败.....	60
4.7.6 文件传输失败.....	60
4.7.7 蓝牙默认设备名字修改.....	61
4.7.8 蓝牙键盘无法使用.....	62
5 WiFi 主要指标说明.....	63
5.1 WiFi 吞吐率以及吞吐率的测量.....	63
5.2 RF 硬件指标.....	63
5.2.1 频偏（Frequency Error）.....	63
5.2.2 矢量误差幅度（EVM）.....	63
5.2.3 发射功率（Transmitter power）.....	63
5.2.4 接收灵敏度（Receiver Sensitivity）.....	64
5.3 WiFi RFTTest（定频测试）.....	64

1 RK Android 平台 WiFi 架构

1.1 Android WiFi 基本架构

1.1.1 Android 平台 WiFi 的层次结构

随着 Android 平台的更新，WiFi 相关的功能不断改变，相关的代码也都不尽相同，甚至差异甚大，但是 Android 整个 WiFi 的层次框架基本不变。

Android 中 WiFi 是使用层次结构设计的，因此执行过程基本是在接到用户命令后，先从上到下，再从下到上，完成用户与 WiFi 设备的交互。下图 1-1 就是 WiFi 功能的详细执行过程示意图。

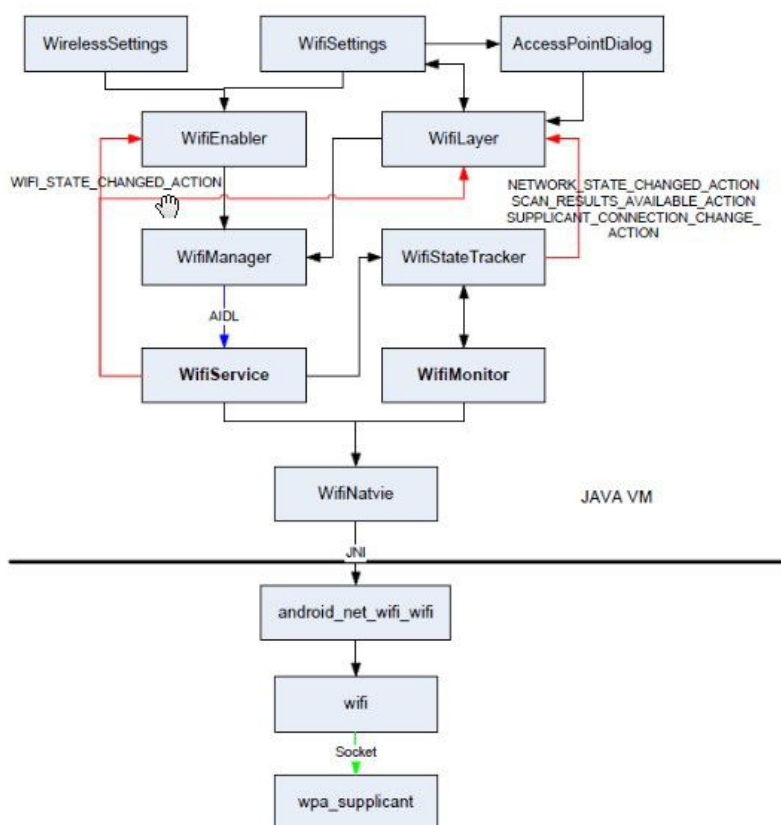


图 1-1

WifiService 和 WifiMonitor 是整个模块的核心。WifiService 负责启动关闭 wpa_supplicant、启动关闭 WifiMonitor 监视线程和把命令下发给 wpa_supplicant，而 WifiMonitor 则负责从 wpa_supplicant 接收事件通知。

当用户按下 Wifi 按钮后，Android 会调用 WifiEnabler 的 onPreferenceChange，再由 WifiEnabler 调用 WifiManager 的 setWifiEnabled 接口函数，通过 AIDL，实际调用的是 WifiService 的 setWifiEnabled 函数，WifiService 接着向自身发送一条 MESSAGE_ENABLE_WIFI 消息，在处理

该消息的代码中做真正的使能工作：

首先装载 WIFI 内核模块（该模块的位置硬编码为"/system/lib/modules/xxx.ko"），然后启动 wpa_supplicant（配置文件硬编码为"/data/misc/wifi/wpa_supplicant.conf"），再通过 WifiStateTracker 来启动 WifiMonitor 中的监视线程。

1.1.2 开源库 wpa_supplicant

它是一个开源的库，加入到 Android 源码中，经过修改后成为 Android 实现 WiFi 功能的基础。它的代码位于./external/wpa_supplicant_X(X 对应 wpa_supplicant 的版本，4.0 开始使用 8 版本) 文件夹中，主要用 C 和 C++ 写成，实现了从上层接到命令后，发送给硬件驱动程序，接着操作硬件完成需要的操作，这里是通过 socket 来与硬件驱动进行通信的。下图 1-2 是 wpa_supplicant 的框架图。

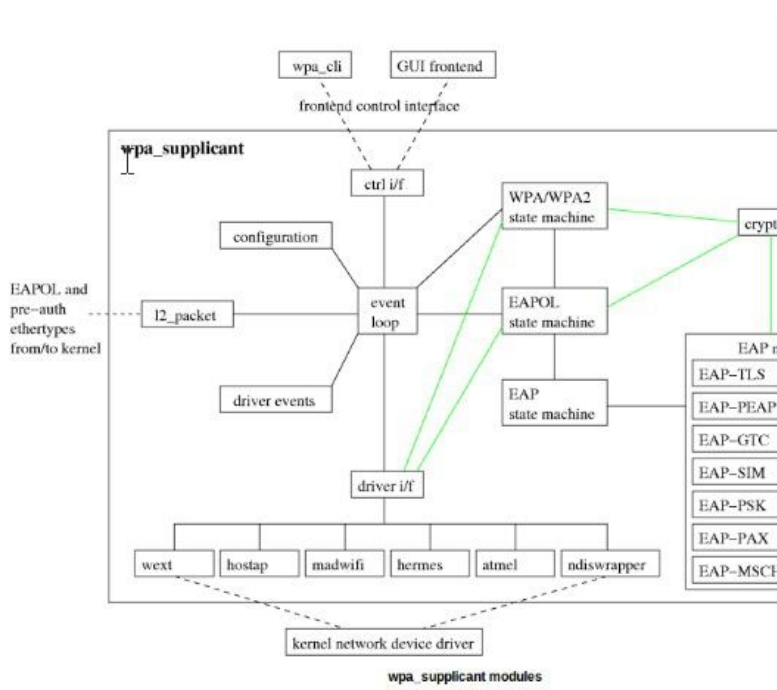


图 1-2

wpa_supplicant 包含不同的协议借口，可以在 wpa_supplicant 的配置文件中配置，打开对应协议的支持比如 NL80211、WEXT 等。wpa_supplicant 的配置文件位置在./external/wpa_supplicant_8/wpa_supplicant/android.conf。

wpa_supplicant 使用的协议栈定义在 init.rkxx.rc 文件中 wpa_supplicant 的 service 中指定。例如 RK30SDK 中的定义如下所示：

```
service wpa_supplicant /system/bin/wpa_supplicant \
-Dnl80211 -iwlan0 -c/data/misc/wifi/wpa_supplicant.conf

#-Dnl80211 -iwlan0 -puse_p2p_group_interface=1 -e/data/misc/wifi/entropy.bin

# we will start as root and wpa_supplicant will switch to user wifi

# after setting up the capabilities required for WEXT

# user wifi

# group wifi inet keystore
```



```
class main

socket wpa_wlan0 dgram 660 wifi wifi

disabled

Oneshot
```

绿色标识位置便是指定使用 NL80211 接口和内核进行通信，其中 D 为 driver 的意思，i 为 interface 使用的是 wlan0（android 指定的 WiFi 网络接口），c 为 configuration file。

1.2 RK 平台 WiFi 架构

1.2.1 WiFi 驱动

1.2.1.1 驱动编译

WiFi 驱动不 Build In 到 kernel.img 中，而是单独编译成模块 ko，放到 system/lib/modules 目录下，在打开 WiFi 时 insmod 驱动，关闭 WiFi 时 rmmod 驱动。

在 Android 4.1 平台上，ko 文件存放在：

device/rockchip/rk30sdk/

在 Android 4.2 平台上，ko 文件存放在：

device/rockchip/common/wifi/lib/，

其中，modules 目录用于存放单核芯片 RK292X 使用的 ko

modules_smp 目录用于存放多核芯片 RK3066&RK31XX 等使用的 ko

在最终的 MID 中,ko 文件统一放在/system/lib/modules 目录下

相应的 ko 文件对应的 WiFi 芯片如下：

rkwifi.ko	RK901, RK903, AP6xxx
8188eu.ko	RTL8188EUS, RTL8188ETV
8189es.ko	RTL8189ES
8192cu.ko	RTL8188CU, RTL8188CTV
8723as.ko	RTL8723AS(SDIO 接口)
8723au.ko	RTL8723AU(USB 接口)
mt5931.ko	MT5931
mt7601sta.ko	MT7601
rt5370sta.ko	MT5370
wlan.ko	早期 WiFi 驱动 build-in 到 kernel 时的引导代码，现在不再使用

移植到 rk 平台的 WiFi 驱动都加了以下格式的版本信息（以 RTL8188EU 为例）：

```
=====
==== Launching Wi-Fi driver! (Powered by Rockchip) ====
=====
```

Realtek 8188EU USB WiFi driver (Powered by Rockchip, **Ver 3.20.WFD**) init.

上面 WiFi 驱动打印信息可在打开 WiFi 时看到，Ver3.20.WFD 为 rk 定义的驱动版本信息，借助此信息可方便调试定位问题。

1.2.1.2 WiFi 电源控制

系统中注册 bcmhdhd_wlan 设备，WiFi 驱动加载时会调用此设备的以下接口：

```
static struct wifi_platform_data rk29sdk_wifi_control = {
    .set_power = rk29sdk_wifi_power,
    .set_reset = rk29sdk_wifi_reset,
    .set_carddetect = rk29sdk_wifi_set_carddetect,
    .mem_prealloc = rk29sdk_mem_prealloc,
};
```

其中, **rk29sdk_wifi_power** 为 WiFi 电源控制函数，包括上电与下电。

WiFi 电源一般需要控制 power 与 reset 两个脚(例如 braodcom sdio WiFi)，也可能只需要控制 power（例如 realteck usb wifi），相应的 GPIO 配置位置如下：

1. RK3168 & RK3188

位于 **board-rk31xx-xx-sdmmc-conifg.c**，需要根据不同的 WiFi 来配置：

```
#define RK30SDK_WIFI_GPIO_POWER_N          RK30_PIN3_PD0    // 电源脚
#define RK30SDK_WIFI_GPIO_POWER_ENABLE_VALUE GPIO_HIGH      // 有效电平
#define RK30SDK_WIFI_GPIO_RESET_N          RK30_PIN2_PA7     // 复位脚
#define RK30SDK_WIFI_GPIO_RESET_ENABLE_VALUE GPIO_HIGH      // 有郊电平
```

注意：在 RK3188 平台上增加了 **sdio 电平驱动强度的设置**，有 **1.8V** 与 **3.3V** 两种电压可以配置，这里设置的电平需要与实际的硬件相匹配。

int rk31sdk_get_sdio_wifi_voltage(void)

```
{
    int voltage;

    /*****

    ** Please tell me how much wifi-module uses voltage in your project.

    *****/

    #if defined(CONFIG_BCM4330) || defined(CONFIG_BCM4329) || defined(CONFIG_BCM4319) || defined(CONFIG_RK903) ||
    defined(CONFIG_RK901)

        voltage = 1800 ; //power 1800mV

    #elif defined(CONFIG_MT5931_MT6622)||defined(CONFIG_MT5931)

        voltage = 1800 ; //power 1800V

    #elif defined(CONFIG_MT6620)

        voltage = 2800 ; //power 2800V

    #elif defined(CONFIG_RDA5990)||defined(CONFIG_RTL8723AS)
```

```

        voltage = 3300 ; //power 3300V

    #else

        //default, sdio use 3.0V

        voltage = 3000 ; //power 3000V

    #endif

    return voltage;

}

```

2. RK3066

位于 **board-rk30-sdk-sdmmc.c**,

先找到以下代码

```

#else //default for RK30,RK3066 SDK

    // refer to file /arch/arm/mach-rk30

    #define WIFI_HOST_WAKE RK30_PIN3_PD2

```

再根据不同的 WiFi 来配置

```

//power

#define RK30SDK_WIFI_GPIO_POWER_N          RK30_PIN3_PC6    // 电源脚

#define RK30SDK_WIFI_GPIO_POWER_ENABLE_VALUE  GPIO_HIGH      // 有效电平

//reset

#define RK30SDK_WIFI_GPIO_RESET_N          RK30_PIN3_PD1     // 复位脚

#define RK30SDK_WIFI_GPIO_RESET_ENABLE_VALUE  GPIO_HIGH      // 有效电平

```

3. RK2926 & RK2928

位于 **board-rk2928-sdk-sdmmc.c**,

先找到以下代码

```

#ifdef CONFIG_ARCH_RK2928 //refer to file ./arch/arm/mach-rk2928/include/mach/iomux.h

    #define WIFI_HOST_WAKE RK2928_PIN3_PC0

```

再根据不同的 WiFi 来配置

```

#define RK30SDK_WIFI_GPIO_POWER_N          RK2928_PIN3_PD2    // 电源脚

#define RK30SDK_WIFI_GPIO_POWER_ENABLE_VALUE  GPIO_HIGH      // 有效电平

#define RK30SDK_WIFI_GPIO_RESET_N          RK2928_PIN3_PD5    // 复位脚

#define RK30SDK_WIFI_GPIO_RESET_ENABLE_VALUE  GPIO_HIGH      // 有效电平

```

1.2.2 WiFi Firmware

Firmware:

WiFi 芯片内部有一个小系统，用来运行 802.11 协议，此部分代码就叫 Firmware。

有些芯片(例如 broadcom)的 Firmware 是以文件的形式存放的，有些芯片(例如 realteck)的 Firmware 是做到驱动代码中的。

Nvram:

WiFi 芯片需要作相应的 RF 参数校准，这个校准值等信息一般放到 Nvram 中。例如，同一个芯片 bcm4330，做成不同的模块时，需要不同的 Nvram。另外，有些芯片(例如 realtek)将 RF 参数校准等信息写到芯片的 EEPROM 中，这部分工作在模块出厂时完成。

WiFi 芯片工作前，需要 host 先下载 Firmware 文件到 WiFi 芯片中，此部分工作在 WiFi 驱动中完成。

Firmware 与 Nvram 文件存放于 external/wlan_loader/firmware/目录中，在最终的 MID 中，存放于/system/etc/firmware。

1.2.3 WiFi Chip Type 节点

通过节点/sys/class/rkwifi/chip 可以获取机器使用的 WiFi 芯片型号信息，以便上层可以动态兼容多种 WiFi 芯片的支持。

这部分实现代码在 kernel/drivers/net/wireless/wifi_sys/rkwifi_sys_iface.c.

1.2.4 WiFi 硬件抽象层

位于 hardware/libhardware_legacy/wifi/wifi.c 中，rk 修改了其中 wifi_load_driver 函数，修改后的流程为：读取/sys/class/rkwifi/chip 节点获取 WiFi 芯片类型，根据不同的芯片类型加载不同的驱动来实现 WiFi 的动态兼容。

注意：有些 WiFi，例如 mt5931, mt6620，由于 WiFi 硬件抽象层跟原生 android 差异很大，难以实现动态兼容。目前通过静态编译方法来实现，也就是需要先配置相应的 mk 文件，再编译，具体见后面的配置章节。

1.2.5 WiFi PCBA 测试

WiFi pcba 测试的代码位于 external/rk-pcba-test/wlan_test.c，其测试流程为：

- 1) 判断 WiFi 芯片类型，加载相应的驱动
- 2) 使用 iwlist 工具扫描
- 3) 显示测试结果

2 RK Android Bluetooth 基本架构

2.1 Android 4.1

2.1.1 源码目录说明

Android 4.1 及之前的版本，蓝牙协议栈使用 bluez，其源码：

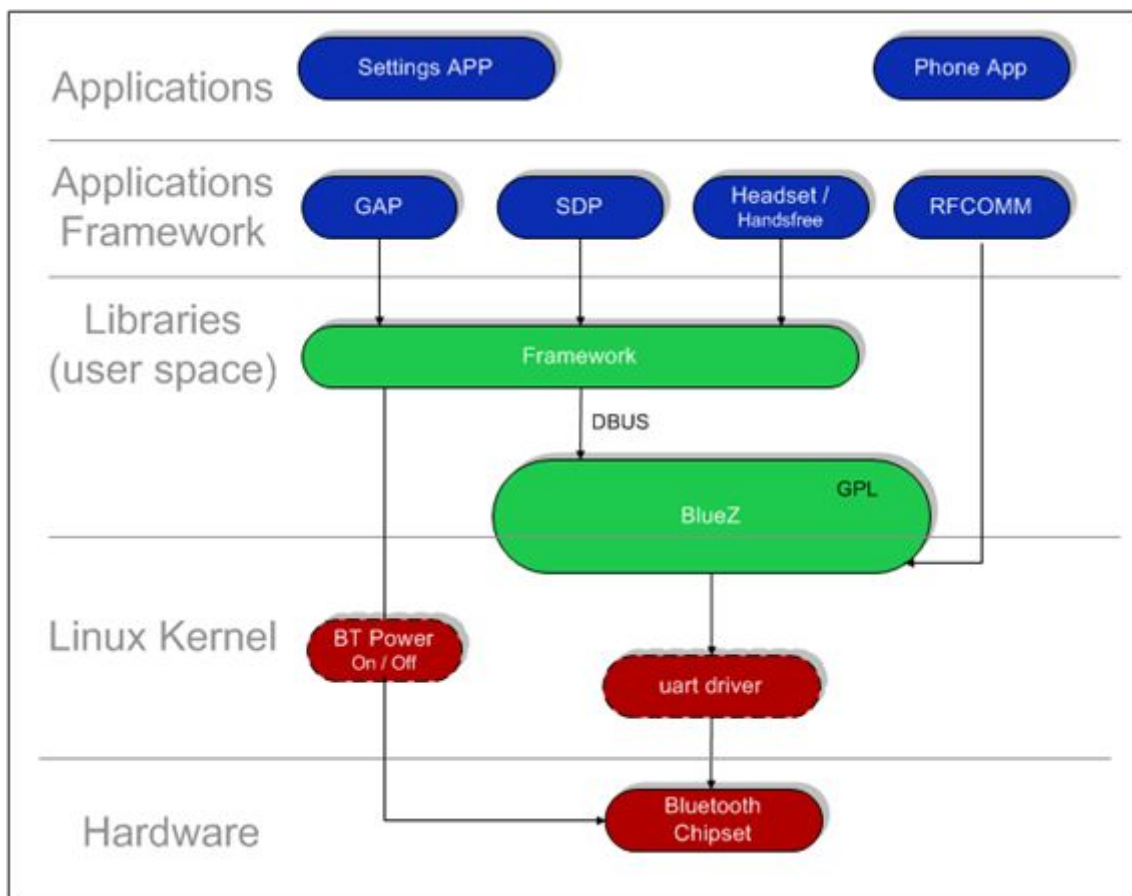
kernel/net/bluetooth/	实现蓝牙的各种传输协议
kernel/drivers/bluetooth/	各种接口的蓝牙设备的驱动
external/bluetooth/bluez/	位于用户空间，提供了函数库及应用接口给第三方程序调用，便于开发蓝牙应用

Android 4.1 及之前的版本，蓝牙应用程序，其源码：

system/bluetooth/	通过 rtkill 驱动接口使用蓝牙电源的开关以及初始化
frameworks/base/core/java/android/server/	位于 Android 上层的蓝牙服务
frameworks/base/core/java/android/bluetooth/	位于 Android 上层的蓝牙 profile 的实现
frameworks/base/packages/SystemUI/src/com/android/systemui/statusbar/policy/BluetoothController.java	状态栏上蓝牙图标
packages/apps/Bluetooth/	OPP、PBAP 这两个 profile 的实现

2.1.2 Android Bluetooth 架构图

Android 蓝牙系统分为四个层次，内核层、BlueZ 库、Bluetooth 的适配库、Bluetooth 的 JNI 部分、Java 框架层、应用层。

**Linux kernel 层:**

Bluez 协议栈、uart 驱动，h4 协议，hci, l2cap, sco, rfcmm

Bluez 层:

这是 bluez 用户空间的库，开源的 bluetooth 代码，包括很多协议，生成 libbluetooth.so

Library 层:

Libbluedroid.so 等

Framework 层:

实现了 Headset/Handfree 和 A2DP/AVRCP profile，但其实现方式不同 Headset/Handfree 是直接在 bluez 的 RFCOMM Socket 上开发的，没有利用 bluez 的 audio plugin，而 A2DP/AVRCP 是在 bluez 的 audio plugin 基础上开发的，大大降低了实现的难度。

2.1.3 RK 平台 Bluetooth 具体配置

2.1.3.1 Bluetooth 驱动

各种 Bluetooth 模块使用统一的 HCI 驱动，因此调试 Bluetooth 驱动时，只需要设置电源控制就可。

2.1.3.2 Bluetooth 电源控制

Rfkill 用于控制蓝牙的电源以及睡眠、唤醒等，其源码：

kernel/net/rfkill/rfkill-rk.c **rockchip's rfkill driver**

kernel/include/linux/rfkill-rk.h **rockchip's rfkill driver 头文件**

kernel/arch/arm/mach-rk30/board-rk30-sdk.c **rockchip's rfkill driver IO 配置**

具体电源控制如下：

1. RK903 & AP6xxx 系统 & RTL8723AS(SDIO)

使用 linux 标准的 RFKILL，具体配置在相关板级文件中（例如 board-rk30-sdk.c）。定义以下设备 **rfkill_rk_platdata**：

```
static struct rfkill_rk_platform_data rfkill_rk_platdata = {
    .type                = RFKILL_TYPE_BLUETOOTH,
```

// 蓝牙的电源控制

```
.poweron_gpio          = { // BT_REG_ON
    .io                  = RK30_PIN3_PC7,
    .enable              = GPIO_HIGH,
    .iomux               = {
        .name            = GPIO3C7_SDMMC1WRITEPRT_NAME,
        .fgpio           = GPIO3C_GPIO3C7,
    },
},
```

// 蓝牙的复位控制

```
.reset_gpio            = { // BT_RST
    .io                  = RK30_PIN3_PD1, // set io to INVALID_GPIO for disable it
    .enable              = GPIO_LOW,
    .iomux               = {
        .name            = GPIO3D1_SDMMC1BACKENDPWR_NAME,
        .fgpio           = GPIO3D_GPIO3D1,
    },
},
```

// Host 控制蓝牙的睡眠/唤醒

```
.wake_gpio             = { // BT_WAKE, use to control bt's sleep and wakeup
    .io                  = RK30_PIN3_PC6, // set io to INVALID_GPIO for disable it
    .enable              = GPIO_HIGH,
    .iomux               = {
        .name            = GPIO3C6_SDMMC1DETECTN_NAME,
```

```

        .fgpio      = GPIO3C_GPIO3C6,
    },
},

```

// 蓝牙通过该 IO 唤醒 HOST

```

.wake_host_irq    = { // BT_HOST_WAKE, for bt wakeup host when it is in deep sleep
    .gpio          = {
        .io        = RK30_PIN6_PA7, // set io to INVALID_GPIO for disable it
        .enable    = GPIO_LOW, // set GPIO_LOW for falling, set GPIO_HIGH for rising
        .iomux     = {
            .name    = NULL,
        },
    },
},

```

// HOST 睡眠前，通过该 IO 阻止蓝牙发数据过来

// HOST 唤醒后，再设置该 IO 允许蓝牙发数据过来

```

.rts_gpio         = { // UART_RTS, enable or disable BT's data coming
    .io            = RK30_PIN1_PA3, // set io to INVALID_GPIO for disable it
    .enable        = GPIO_LOW,
    .iomux         = {
        .name      = GPIO1A3_UART0RTSN_NAME,
        .fgpio     = GPIO1A_GPIO1A3,
        .fmux      = GPIO1A_UART0_RTS_N,
    },
},
};

```

2. MT6622

不使用 RFKILL，具体配置在相关板级文件中（例如 board-rk30-sdk.c）。定义以下设备 **mt6622_platdata**：

```
static struct mt6622_platform_data mt6622_platdata = {
```

// 蓝牙的电源控制

```

.power_gpio       = { // BT_REG_ON
    .io            = RK30_PIN3_PC7, // set io to INVALID_GPIO for disable it
    .enable        = GPIO_HIGH,
    .iomux         = {
        .name      = "mt6622_power",
        .fgpio     = GPIO3_C7,
    },
},

```


// 蓝牙的复位控制

```
.reset_gpio      = { // BT_RST
    .io           = RK30_PIN3_PD1,
    .enable       = GPIO_HIGH,
    .iomux        = {
        .name     = NULL,
    },
},
```

// 蓝牙通过该 IO 唤醒 HOST

```
.irq_gpio        = {
    .io           = RK30_PIN0_PA5,
    .enable       = GPIO_LOW,
    .iomux        = {
        .name     = NULL,
    },
}
};
```

3 RK 平台 WiFi BT 具体配置

3.1 kernel WiFi 框架相关部分

这部分是 RK SDK 默认打开的，一般情况下不需要再去配置。

3.1.1 WiFi 接口驱动配置

1、SDIO 接口

选择 “RK29 SDMMC1 controller support(sdio)”

Location:

- > Device Drivers
- > MMC/SD/SDIO card support
- > RK29 SDMMC controller support

```
<*> RK29 SDMMC1 controller support(sdio)
[ ] write-protect for SDMMC1
[ ] sdio-irq from gpio
```

2、USB 接口

选择 “Rockchip USB 2.0 host controller”

Location:

- > Device Drivers
- > USB support

```
< > Rockchip USB 1.1 host controller
<*> Rockchip USB 2.0 host controller
[*] ---usb2.0 host controller enable
```

3.1.2 WiFi 网络协议栈配置

选择 “Wireless”

Location:

- > Networking support

```
--- wireless
<*> cfg80211 - wireless configuration API
[ ] nl80211 testmode command
[ ] enable developer warnings
[ ] cfg80211 regulatory debugging
[*] enable powersave by default
[ ] cfg80211 debugFS entries
[ ] use statically compiled regulatory rules database
[*] cfg80211 wireless extensions compatibility
[*] wireless extensions sysfs files
< > Common routines for IEEE802.11 drivers
[ ] Allow reconnect while already connected
< > Generic IEEE 802.11 Networking Stack (mac80211)
```

3.2 Kernel 蓝牙框架相关部分

这部分是 RK SDK 默认打开的，一般情况下不需要再去配置。

3.2.1 蓝牙接口驱动配置

1. UART 接口：

选择 “Serial port 0 support”

Location:

- > Device Drivers
- > Character devices
- > Serial drivers
- > RockChip RK29/RK30 serial port support

3.2.2 蓝牙框架部分配置

选择以下协议：

```
[*] Networking support --->
    <*> Bluetooth subsystem support --->
        [*] L2CAP protocol support
        [*] SCO links support
        <*> RFCOMM protocol support
        [*] RFCOMM TTY support
        <*> BNEP protocol support
        <*> HIDP protocol support
        Bluetooth device drivers --->
            <*> HCI UART driver
                [*] UART (H4) protocol support
                [*] HCILL protocol support
            <*> Bluetooth auto sleep
    <*> RF switch subsystem support --->
        [*] Power off on suspend
        [*] Rockchips RFKILL driver
```

3.3 RK903 & RK901 & AP6xxx 系列配置

包括以下模块：RK901、RK903、AP6181、AP6210、AP6330、AP6476、BCM4330 等。以上模块都是 broadcom 芯片，使用同一份驱动 rkwifi，对应的 Firmware 与 Nvram 文件不一样。

3.3.1 Kernel Memuconfig 配置

1、 选择 “RK901/RK903/BCM4330 wireless cards support”

Location:

- > Device Drivers
- > Network device support
- > Wireless LAN
- > WiFi device driver support

```
--- wireless LAN
[*] wireless LAN (IEEE 802.11)
[*] WiFi device driver support (RK901/RK903/BCM4330 wireless cards support) ---->
    RK901/RK903/BCM4330 wireless cards support
    [*] Enable NL80211 support (NEW)
    Select the wifi module (RK903) ---->
    Select the wifi module crystal freq (26M) ---->
```

Select the wifi module:

选择相应的模块

Select the wifi modules crystal freq:

选择模块使用的外围晶体频率，默认是 26M，绝大部分情况下不需要修改

2、 关闭 “nl80211 testmode command”

Location:

- > Networking support
- > Wireless
- > cfg80211 - wireless configuration API

```
--- wireless
[*] cfg80211 - wireless configuration API
[*] nl80211 testmode command
[*] enable developer warnings
[*] cfg80211 regulatory debugging
```

3、 选择串口硬件流控 “Serial port 0 CTS/RTS support”

Location:

- > Device Drivers
- > Character devices
- > Serial drivers
- > RockChip RK29/RK30 serial port support
- > Serial port 0 support

```
[*] RockChip RK29/RK30 serial port support
[*] Serial port 0 support
[*] Serial port 0 CTS/RTS support
(0) Serial port 0 DMA support (EXPERIMENTAL)
```

RK3066、RK292x、RK31xx 默认蓝牙使用的是 UART0，如果硬件上有变化，配置时需要选择对应的 UART。

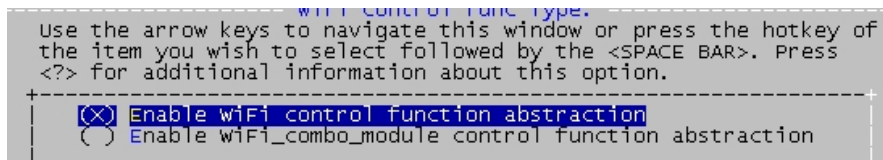
Serial port 0 DMA support 里的值需要修改成 0，其它值只在软件流控时使用。

4、选择“Enable WiFi control function abstraction”

Location:

-> System Type

-> wifi control func Type



3.3.2 Android 部分配置

1、Android 4.1

在 device/rockchip/rkxxsdk/BoardConfig.mk 中，需要保证以下 mtk 相关配置为 false

```
MT5931_WIFI_SUPPORT := false
```

```
MT6622_BT_SUPPORT := false
```

```
#modified by XBW at 2012-12-01
```

```
BOARD_HAVE_MTK_MT6620 := false
```

注：如果这几个选项有修改，记得要 make clean 后重新编译

2、Android 4.2

在 device/rockchip/rkxxsdk/wifi_bt.mk 中，需要保证以下 broadcom 相关配置为 true，其它都为 false

```
# broadcom
```

```
BROADCOM_WIFI_SUPPORT := true
```

```
BROADCOM_BT_SUPPORT := true
```

```
# mt5931 && mt6622
```

```
MT5931_WIFI_SUPPORT := false
```

```
MT6622_BT_SUPPORT := false
```

```
# rtl8723
```

```
RTL8723_BT_SUPPORT := false
```

```
# rda5876
```

```
RDA587X_BT_SUPPORT := false
```

```
# rtl8723au(USB interface)
```

```
RTL8723_BTUSB_SUPPORT := false
```

在 device/rockchip/rkxxsdk/BoardConfig.mk 中，将以下配置设置成 true

```
ifeq ($(strip $(BROADCOM_BT_SUPPORT)),true)
```

```
BOARD_HAVE_BLUETOOTH ?= true
```

注：如果这几个选项有修改，记得要 **make clean** 后重新编译

注意：以上配置只支持 **BT3.0**（一般情况下够用）

3.3.3 Android 4.2 支持 BT4.0 配置（可选）

如果想要支持 **BT4.0**，需要打开 **bplus** 配置：

如果使用 AP6476 的 **GPS 功能**，也必须要打开 **bplus** 配置：

使用 **bplus**：

确认 device/rockchip/rk30sdk/wifi_bt.mk 文件如下选项：

```
BROADCOM_WIFI_SUPPORT := true
```

```
BROADCOM_BT_SUPPORT := true
```

确认 device/rockchip/rk30sdk/BoardConfig.mk 文件中如下配置

```
ifeq ($(strip $(BROADCOM_BT_SUPPORT)),true)
```

```
BOARD_HAVE_BLUETOOTH ?= true
```

```
BOARD_HAVE_BLUETOOTH_BCM ?= true
```

```
BLUETOOTH_USE_BPLUS ?= true
```

注：如果这几个选项有修改，记得要 **make clean** 后重新编译

手工指定蓝牙 **firmware** **HCD** 文件

修改 external/bluetooth/bluedroid/conf/bt_vendor.conf 文件，在其中指定 FwPatchFileName 的文件名：

```
FwPatchFileName=rk903.hcd # RK903, 37.4MHz 晶振
```

```
FwPatchFileName=rk903_26M.hcd # RK903, 26MHz 晶振，大多数用这个
```

```
FwPatchFileName=bcm2076b1.hcd # AP6476
```

```
FwPatchFileName=bcm20710a1.hcd # AP6210
```

```
FwPatchFileName=bcm40183b2.hcd # AP6330, AP6493
```

蓝牙默认设备名称修改

在文件 external/bluetooth/bluedroid/conf/bt_stack_40.conf 中指定蓝牙名称
Name=RK_BT_4.0

3.4 RTL8188 配置

包括以下模块：RTL8192CU、RTL8188CTV、RTL8188EU、RTL8188ETV。

其中 RTL8192CU、RTL8188CTV 使用同样的驱动“**Realtek 8192CU USB WiFi Support**”。

RTL8188EU、RTL8188ETV 使用同样的驱动“**Realtek 8188EU USB WiFi Support**”。

3.4.1 Kernel Memuconfig 配置

- 1、 选择 “**Realtek 8192CU USB WiFi Support**” 或
“**Realtek 8188EU USB WiFi Support**”

Location:

- > Device Drivers
- > Network device support
- > Wireless LAN
- > WiFi device driver support

```
--- wireless LAN
[*] wireless LAN (IEEE 802.11)
  Enable WIFI AIDS(Automatic Identification USB wifi Type) (NEW)
  [*] WiFi device driver support (Realtek 8188EU USB WiFi Support) --->
```

- 2、 关闭 “**nl80211 testmode command**”

Location:

- > Networking support
- > Wireless
- > cfg80211 - wireless configuration API

```
--- wireless
[*] cfg80211 - wireless configuration API
  [*] nl80211 testmode command
  [*] enable developer warnings
  [*] cfg80211 regulatory debugging
```

- 3、 选择 “**Enable WiFi control function abstraction**”

Location:

- > System Type
- > wifi control func Type

```
WiFi control func type.
Use the arrow keys to navigate this window or press the hotkey of
the item you wish to select followed by the <SPACE BAR>. Press
<?> for additional information about this option.
+-----+
| (X) Enable wifi control function abstraction |
| ( ) Enable wifi_combo_module control function abstraction |
+-----+
```

3.4.2 Android 部分配置

1、Android 4.1

在 device/rockchip/rkxxsdk/BoardConfig.mk 中，需要保证以下 mtk 相关配置为 false

```
MT5931_WIFI_SUPPORT := false
```

```
MT6622_BT_SUPPORT := false
```

```
#modified by XBW at 2012-12-01
```

```
BOARD_HAVE_MTK_MT6620 := false
```

注：如果这几个选项有修改，记得要 **make clean** 后重新编译

2、Android 4.2

在 device/rockchip/rkxxsdk/wifi_bt.mk 中，需要保证以下 **broadcom** 相关配置为 **true**，其它都为 **false**

```
# broadcom

BROADCOM_WIFI_SUPPORT := true

BROADCOM_BT_SUPPORT    := true

# mt5931 && mt6622

MT5931_WIFI_SUPPORT    := false

MT6622_BT_SUPPORT      := false

# rtl8723

RTL8723_BT_SUPPORT     := false

# rda5876

RDA587X_BT_SUPPORT     := false

# rtl8723au(USB interface)

RTL8723_BTUSB_SUPPORT := false
```

注：如果这几个选项有修改，记得要 **make clean** 后重新编译

3.5 MT5931&MT6622 配置

3.5.1 Kernel Memuconfig 配置

- 选择 “MediaTek MT5931 WiFi” 单 WiFi 模块 或
“MediaTek MT5931 & MT6622 WiFi Bluetooth Combo” WiFi BT 二合一模块

Location:

- > Device Drivers
- > Network device support
- > Wireless LAN
- > WiFi device driver support

```
--- wireless LAN
[*] wireless LAN (IEEE 802.11)
    [*] WiFi device driver support (MediaTek MT5931 & MT6622 WiFi Bluetooth Combo) ---->
```

- 选择串口软件流控 “Serial port 0 DMA support (EXPERIMENTAL) ”

Location:

- > Device Drivers
- > Character devices
- > Serial drivers
- > RockChip RK29/RK30 serial port support

-> Serial port 0 support

```
[*] Rockchip RK29/RK30 serial port support
[*]   Serial port 0 support
[*]     Serial port 0 CTS/RTS support
(2)   Serial port 0 DMA support (EXPERIMENTAL)
```

RK3066、RK292x、RK31xx 默认蓝牙使用的是 UART0，如果硬件上有变化，配置时需要选择对应的 UART。

Serial port 0 DMA support 里的值需要修改成 2。

串口的 CTS, RTS 脚可能会干扰到串口通信, 造成 BT 打不开, 主控的 CTS, RTS 需要与模块断开。

3、 选择 “Enable WiFi control function abstraction”

Location:

-> System Type

-> wifi control func Type

```
WiFi control func type.
Use the arrow keys to navigate this window or press the hotkey of
the item you wish to select followed by the <SPACE BAR>. Press
<?> for additional information about this option.
+-----+
| (X) Enable wifi control function abstraction |
| ( ) Enable wifi_combo_module control function abstraction |
+-----+
```

3.5.2 模块电源脚配置

1、中龙通模块:

WiFi power: 模块第 2 脚
 WiFi reset: 模块第 1 脚
 BT power: 模块第 33 脚
 BT reset: 模块第 6 脚
 BT interrupt: 模块第 8 脚

2、芯动力等 M500 模块:

WiFi power: 模块第 30 脚
 WiFi reset: 与 BT reset 复用，放在 BT 驱动 probe 中统一控制
 BT power: 模块第 1 脚
 BT reset: 模块第 22 脚
 BT interrupt: 模块第 41 脚

3.5.3 Android 部分配置

1、Android 4.1

在 device/rockchip/rkxxsdk/BoardConfig.mk 中，**需要保证以下相关配置为 true**

MT5931_WIFI_SUPPORT := true

MT6622_BT_SUPPORT := true

以下为 false

#modified by XBW at 2012-12-01

BOARD_HAVE_MTK_MT6620 := false

修改后需要将以下模块重新编译:

mmm hardware/libhardware_legacy/ -B

mmm system/netd/ -B

mmm system/bluetooth/bluedroid/ -B

mmm external/bluetooth/bluez/tools/ -B

mmm frameworks/base/core/jni/ -B

mmm packages/apps/Bluetooth/ -B

删除 rm -rf out/target/product/rk30sdk/system

再 make

2、Android 4.2

在 device/rockchip/rkxxsdk/wifi_bt.mk 中, 需要保证以下相关配置为 true, 其它都为 false

broadcom

BROADCOM_WIFI_SUPPORT := false

BROADCOM_BT_SUPPORT := false

mt5931 && mt6622

MT5931_WIFI_SUPPORT := true

MT6622_BT_SUPPORT := true

rtl8723

RTL8723_BT_SUPPORT := false

rda5876

RDA587X_BT_SUPPORT := false

rtl8723au(USB interface)

RTL8723_BTUSB_SUPPORT := false

修改后需要将以下模块重新编译:

mmm hardware/libhardware_legacy/ -B

mmm system/netd/ -B

mmm hardware/libhardware/ -B

mmm external/bluetooth/bluedroid/ -B

mmm frameworks/base/services/java/ -B

mmm packages/apps/Bluetooth/ -B

mmm device/common/libbt/ -B

删除 rm -rf out/target/product/rk30sdk/system/

删除 rm -rf out/target/product/rk30sdk/root/

再 make

3.5.4 串口硬件流控配置

目前 MT6622 使用得都是串口软件流控, 这可能造成串口通信不稳定, 导致 BT 通信异常问题。硬件流控现已经调试成功, 需要修改以下三点, 以支持硬件流控。

具体请参考《MT6622 串口使用硬件流控补丁》。

1. UART 配置成硬件流控

选择串口硬件流控 “Serial port 0 CTS/RTS support”

Location:

- > Device Drivers
- > Character devices
- > Serial drivers
- > RockChip RK29/RK30 serial port support
- > Serial port 0 support

```
[*] RockChip RK29/RK30 serial port support
[*]   Serial port 0 support
[*]   Serial port 0 CTS/RTS support
(0)   Serial port 0 DMA support (EXPERIMENTAL)
```

2. android 修改

Android 4.1

device/rockchip/rk30sdk/init.rk30board.rc

```
-service mtk_hciattach /system/bin/hciattach -n -t 10 -s 115200 /dev/ttyS0 mtk 1500000 noflow
```

```
+service mtk_hciattach /system/bin/hciattach -n -t 10 -s 115200 /dev/ttyS0 mtk 1500000 flow
```

Android4.2

更新 device/rockchip/common/bluetooth/mt6622/libbluetooth_mtk.so 文件

1) 通过以下操作编译生效:

```
rm -rf out/target/product/rk30sdk/system
make
```

2) 或者直接使用 adb push 来验证

```
adb push libbluetooth_mtk.so system/lib
重启机器
```

3. 硬件修改

由于 MT6622 的 RTS 脚输出一直为高电平, 导致串口在切换波特率后通信失败。需要将模块

的 RTS 脚强制接地。具体修改为：将模块端的 RTS 悬空，对应主控端的 CTS 接地（串 10K 电阻）。

3.6 MT7601 配置

3.6.1 Kernel Memuconfig 配置

1、选择 “Mediatek MT7601 USB WiFi Support”

Location:

- > Device Drivers
- > Network device support
- > Wireless LAN
- > WiFi device driver support

```
--- wireless LAN
[*] wireless LAN (IEEE 802.11)
    WiFi device driver support (Mediatek MT7601 USB WiFi support) --->
```

2、关闭 “nl80211 testmode command”

Location:

- > Networking support
- > Wireless
- > cfg80211 - wireless configuration API

```
--- wireless
-*- cfg80211 - wireless configuration API
    [ ] nl80211 testmode command
    [ ] enable developer warnings
    [ ] cfg80211 regulatory debugging
```

3、选择 “Enable WiFi control function abstraction”

Location:

- > System Type
- > wifi control func Type

```
WiFi control func Type.
Use the arrow keys to navigate this window or press the hotkey of
the item you wish to select followed by the <SPACE BAR>. Press
<?> for additional information about this option.
+-----+
| (X) Enable WiFi control function abstraction |
| ( ) Enable WiFi_combo_module control function abstraction |
+-----+
```

3.6.2 Android 部分配置

1、Android 4.1

在 device/rockchip/rkxxsdk/BoardConfig.mk 中，需要保证以下相关配置为 false

MT5931_WIFI_SUPPORT := false

```
MT6622_BT_SUPPORT := false
```

```
#modified by XBW at 2012-12-01
```

```
BOARD_HAVE_MTK_MT6620 := false
```

注：如果这几个选项有修改，记得要 **make clean** 后重新编译

2、Android 4.2

在 device/rockchip/rkxxsdk/wifi_bt.mk 中，需要保证以下 **mt7601** 相关配置为 **true**，其它都为 **false**

```
BROADCOM_WIFI_SUPPORT := false
```

```
BROADCOM_BT_SUPPORT := true
```

```
# mt5931 && mt6622
```

```
MT5931_WIFI_SUPPORT := false
```

```
MT6622_BT_SUPPORT := false
```

```
# rtl8723
```

```
RTL8723_BT_SUPPORT := false
```

```
# rda5876
```

```
RDA587X_BT_SUPPORT := false
```

```
# rtl8723au(USB interface)
```

```
RTL8723_BTUSB_SUPPORT := false
```

```
# mt7601U
```

```
MT7601U_WIFI_SUPPORT := true
```

注：如果这几个选项有修改，记得要 **make clean** 后重新编译

3.7 RTL8723AU(USB 接口)配置

3.7.1 Kernel Memuconfig 配置

1、选择 “Realtek 8723AU USB WiFi Support ”

Location:

-> Device Drivers

-> Network device support

-> Wireless LAN

-> WiFi device driver support

```
--- wireless LAN
[*] wireless LAN (IEEE 802.11)
[*] Enable WIFI AIDS(Automatic Identification USB wifi type) (NEW)
WiFi device driver support (Realtek 8723AU USB WiFi support) --->
```

2、关闭 “nl80211 testmode command”

Location:

- > Networking support
- > Wireless
- > cfg80211 - wireless configuration API

```
--- wireless
-*-  cfg80211 - wireless configuration API
[ ]  nl80211 testmode command
[ ]  enable developer warnings
[ ]  cfg80211 regulatory debugging
```

3、 选择 “Enable WiFi control function abstraction”

Location:

- > System Type
- > wifi control func Type

```
Use the arrow keys to navigate this window or press the hotkey of
the item you wish to select followed by the <SPACE BAR>. Press
<?> for additional information about this option.
+-----+
(X) Enable WiFi control function abstraction
( ) Enable WiFi_combo_module control function abstraction
```

4. Bluetooth 相关配置

GPIO 配置

目前不需要配置该模块的 GPIO，但要求该模块常供电

Bluetooth 协议配置

- ```
[*] Networking support --->
<*> Bluetooth subsystem support --->
[*] L2CAP protocol support
[*] SCO links support
<*> RFCOMM protocol support
[*] RFCOMM TTY support
<*> BNEP protocol support
<*> HIDP protocol support
```

不需要配置 rfkill / UART 等的配置，如果这些配置已经选上也没有关系，RTL8723AU 不会用到这些。

## 3.7.2 Android 部分配置

### 1. Android 4.2

在 device/rockchip/rkxxsdk/wifi\_bt.mk 中，**需要保证以下相关配置为 true，其它都为 false**

```
BROADCOM_WIFI_SUPPORT := true
BROADCOM_BT_SUPPORT := false

mt5931 && mt6622

MT5931_WIFI_SUPPORT := false
MT6622_BT_SUPPORT := false
```

```
rtl8723

RTL8723_BT_SUPPORT := false

rda5876

RDA587X_BT_SUPPORT := false

rtl8723au(USB interface)

RTL8723AS_VAU_BT_SUPPORT := true
```

注：如果这几个选项有修改，记得要 **make clean** 后重新编译

## 3.8 RTL8723AS(SDIO 接口)配置

### 3.8.1 Kernel Memuconfig 配置

#### 1、选择 “Realtek 8723AS SDIO WiFi Support”

Location:

- > Device Drivers
- > Network device support
- > Wireless LAN
- > WiFi device driver support

```
--- wireless LAN
[*] wireless LAN (IEEE 802.11)
 wifi device driver support (Realtek 8723AS SDIO w1Fi support) --->
```

#### 2、关闭 “nl80211 testmode command”

Location:

- > Networking support
- > Wireless
- > cfg80211 - wireless configuration API

```
--- wireless
-*-- cfg80211 - wireless configuration API
 [] nl80211 testmode command
 [] enable developer warnings
 [] cfg80211 regulatory debugging
```

#### 3、选择 “Enable WiFi control function abstraction”

Location:

- > System Type
- > wifi control func Type

```
Use the arrow keys to navigate this window or press the hotkey of
the item you wish to select followed by the <SPACE BAR>. Press
<?> for additional information about this option.
+-----+
| (X) Enable w1Fi control function abstraction |
| () Enable w1Fi_combo_module control function abstraction |
+-----+
```

#### 4、选择串口**硬件流控** “Serial port 0 CTS/RTS support”

Location:

- > Device Drivers
- > Character devices
- > Serial drivers
- > RockChip RK29/RK30 serial port support
- > Serial port 0 support

```
[*] Rockchip RK29/RK30 serial port support
[*] Serial port 0 support
[*] Serial port 0 CTS/RTS support
(0) Serial port 0 DMA support (EXPERIMENTAL)
```

RK3066、RK292x、RK31xx 默认蓝牙使用的是 UART0，如果硬件上有变化，配置时需要选择对应的 UART。

**Serial port 0 DMA support** 里的值需要修改成 0，其它值只在软件流控时使用。

### 5. Bluetooth 相关配置

GPIO 配置

只需要配置 power\_on 即可，将 rkill\_rk\_platdata 中的其它 gpio 设置成 INVALID\_GPIO

Bluetooth 协议配置

[\*] Networking support --->

<\*> Bluetooth subsystem support --->

[\*] L2CAP protocol support

[\*] SCO links support

<\*> RFCOMM protocol support

[\*] RFCOMM TTY support

<\*> BNEP protocol support

<\*> HIDP protocol support

Bluetooth device drivers --->

<\*> HCI UART driver

[\*] Realtek H5 protocol support

[\*] HCILL protocol support

<\*> RF switch subsystem support --->

[\*] Power off on suspend

[\*] Rockchips RFKILL driver

### 3.8.2 Android 部分配置

#### 1. Android 4.2

在 device/rockchip/rkxxsdk/wifi\_bt.mk 中，需要保证以下相关配置为 true，其它都为 false

```
BROADCOM_WIFI_SUPPORT := true
```



```
BROADCOM_BT_SUPPORT := false

mt5931 && mt6622

MT5931_WIFI_SUPPORT := false

MT6622_BT_SUPPORT := false

rtl8723

RTL8723AS_BT_SUPPORT := true

rda5876

RDA587X_BT_SUPPORT := false

rtl8723au(USB interface)

RTL8723_BTUSB_SUPPORT := false
```

注：如果这几个选项有修改，记得要 **make clean** 后重新编译

## 3.9 ESP8089 配置

### 3.9.1 Kernel menuconfig

#### 1、选择 “ESP8089 SDIO WiFi Support”

Location:

```
-> Device Drivers
 -> Network device support
 -> Wireless LAN
 -> WiFi device driver support
 WiFi device driver support (Espressif 8089 sdio Wi-Fi) --->
 Espressif 8089 sdio Wi-Fi
```

此时，还会弹出新的一个配置项

```
[] Espressif 8089 sdio Wi-Fi combo with BK3515A bluetooth (NEW)
```

如果是与 BK3515A 组合的二合一或者三合一，那么就需要勾选此项。如果是单 wifi，则不可勾选。

#### 2、电源配置

内核部分还需要增加对电源的修改，找到你们用 board-xxx-sdmmc-config.c 文件，用 CONFIG\_ESP8089 将你们定义管脚的地方包起来，定义你们所用的 wifi 供电管脚，同时还需要在 rkxxsdk\_get\_sdio\_wifi\_voltage 中将供电电压设好，一般都是 3V，如下

```
+ #elif defined(CONFIG_ESP8089)
+ #define RK30SDK_WIFI_GPIO_POWER_N RK30_PIN3_PDO
+ #define RK30SDK_WIFI_GPIO_POWER_ENABLE_VALUE GPIO_HIGH
+
+ #define RK30SDK_WIFI_GPIO_RESET_N RK30_PIN3_PD1
+ #define RK30SDK_WIFI_GPIO_RESET_ENABLE_VALUE GPIO_HIGH
```

```
release 1888 / / power 1888
+ #elif defined(CONFIG_ESP8089)
+ voltage = 3000 ; //power 3000V
```

需要注意的是，如果是 ESP8089+BK3515A 的组合(二合一或者三合一模块)，发现不打开 BT 的情况下，无法打开 WIFI，先检查下 BT 的管脚定义是否控制了 reset 脚。BK3515A 不需要控制 reset，而是 wifi 需要控制。

### 3 配置文件

ESP8089 驱动(目前最新版本 1.15)，支持动态读取配置文件，完成一些特殊功能，比如 26M 和 40M 晶振的动态支持，比如关闭 WIFI 是否有 clk 输出等等。配置文件在系统的 system/lib/modules/init\_data.conf 中，

注意：只需要修改文件中参数的值，而不要随意修改文件的格式和参数的名称

目前有以下几个参数可以配置

#### 1、crystal\_26M\_en

表示是否使用 26M 晶振，有效的值有 0 和 1

默认值 0

0: 40MHz

1: 26MHz

2: 24MHz

#### 2、test\_xtal

表示 wifi 芯片输出 crystal clock 用于蓝牙或其他芯片使用的方式，有效的值有 0,1,2

此配置用于 wifi 芯片调试测试，一般客户无须使用

默认值 0

0: None

1: GPIO0 PIN is the output of the crystal clock

2: U0RXD PIN is the output of the crystal clock

#### 3、sdio\_configure

表示 host sdio 模式，有效值 0,1,2

默认值 2, 如果你们是 SPI 接口，请用 1

0: Auto by pin strapping

1: SDIO dataoutput is at negative edges (SDIO V1.1)

2: SDIO dataoutput is at positive edges (SDIO V2.0)

#### 4、bt\_configure

表示 wifi，蓝牙共存接线模式，有效值 0,1

如果 esp8089 是与 bk3515a 组合成二合一或者三合一，则此项需配置为 2

默认值 0

0: None

1: GPIO0 -> WLAN\_ACTIVE

MTMS -> BT\_ACTIVE

MTDI -> BT\_PRIORITY  
XPD\_DCDC -> BT\_CLK\_REQ  
U0TXD -> ANT\_SEL\_BT  
U0RXD -> ANT\_SEL\_WIFI

2: None, have blurtooth

### 5、bt\_protocol

表示 wifi，蓝牙共存使用的协议，有效值 0,1,2,3

默认值 0

0: WiFi-BT are not enabled. Antenna is for WiFi

1: WiFi-BT are not enabled. Antenna is for BT

### 6、dual\_ant\_configure

表示双天线配置模式，有效值 0,1,2,3

默认值 0

0: None

1: U0RXD + XPD\_DCDC

2: U0RXD + GPIO0

3: U0RXD + U0TXD

### 7、test\_uart\_configure

表示芯片 uart 输出模式,有效值 0,1,2

默认值 0

0: None

1: GPIO2 PIN is the output of the UART Tx Data

2: U0TXD PIN is the output of the UART Tx Data

### 8.share\_xtal:

This option is to share crystal clock for BT

The state of Crystal during sleeping

表示是否需要在关闭 wifi 时，继续提供震荡电路以提供给外部使用

如果是与 BK3515A 组合成的二合一或者三合一，此项需要勾选为 1

0: Off

1: Forcely On

2: Automatically On according to XPD\_DCDC

默认值:0

### 9.gpio\_wake

During sleeping, the chip can be waken up by a PIN

0: None

1: XPD\_DCDC  
2: GPIO0  
3: both XPD\_DCDC and GPIO0  
默认值:0

#### 10、no\_auto\_sleep

0:auto sleep, for pad  
1:no auto sleep, for TV BOX  
默认值: 0, 如果是 **BOX**, 请用 1

#### 11、ext\_rst

rst chip by ext gpio  
0:no gpio can reset chip  
1:there is a gpio can reset chip  
默认值:0

### 4 wifi 芯片指标参数文件

路径位于/system/lib/modules/esp\_init\_data.bin, 是 wifi 原厂留给客户的; 如果客户所用的 wifi 芯片在 PCB 板上工作指标不好, 需要到原厂或者模组商那边调板子, 获得新的 esp\_init\_data.bin 文件替换默认的文件, 来最好的匹配不同板子的射频指标。

### 5 信号指标平滑性

当 wifi 连接固定路由器之后, 同一个半径内活动, 信号指标是不一样的; 即便是站在固定位置, 信号指标也是会变化的, 这些是正常的现象。如果有非常特殊需求, 想要做信号指标”优化”, 使得信号看起来非常整齐, “漂亮”, 可使用方法, 但是 Rockchips 方面对于这些优化持保留意见, 如果优化后造成任何纠纷产生, Rockchips 方面不附带任何义务和责任:

新建文件/system/bin/rssi\_filter.conf, 可写入这几种参数之一: fast、normal、slow。示例如下:

- (1) echo fast >/system/bin/rssi\_filter.conf 过滤强度低, 波动较大
  - (2) echo normal >/system/bin/rssi\_filter.conf 过滤强度低, 轻微波动
  - (3) echo slow >/system/bin/rssi\_filter.conf 过滤强度高, 几乎无波动
- 若无 rssi\_filter.conf 或者参数不对, 则以 normal 级别为准。

注意: 当阅读到这里, rockchips 默认您已经阅读了上述声明, 谢谢。

## 3.9.2 Android 配置

### 1、Android 4.2

在 device/rockchip/rkxxsdk/wifi\_bt.mk 中, 需要保证以下 **broadcom** 相关配置为 **true**, 其它都为 **false**

# broadcom

```
BROADCOM_WIFI_SUPPORT := true
BROADCOM_BT_SUPPORT := true

mt5931 && mt6622
MT5931_WIFI_SUPPORT := false
MT6622_BT_SUPPORT := false

rtl8723
RTL8723_BT_SUPPORT := false

rda5876
RDA587X_BT_SUPPORT := false

rtl8723au(USB interface)
RTL8723_BTUSB_SUPPORT := false
```

注：如果这几个选项有修改，记得要 **make clean** 后重新编译

## 3.10 RDA5876 配置

### 3.10.1 Kernel menuconfig

Menuconfig 里面配置，去掉 CONFIG\_RFKILL\_RK，选上 CONFIG\_TCC\_BT\_DEV:

```
-CONFIG_RFKILL_RK=y
+CONFIG_TCC_BT_DEV=y
+CONFIG_UHID=y
```

### 3.10.2 电源控制

在项目的 board 文件里面加上 ldo 与 bt\_host\_wake pin 脚的匹配:

```
+static struct tcc_bt_platform_data tcc_bt_platdata = {
+
+ .power_gpio = { // ldoon
+ .io = RK2928_PIN3_PC0,
+ .enable = GPIO_HIGH,
+ .iomux = {
+ .name = NULL,
+ },
+ },
+
+ .wake_host_gpio = { // BT_HOST_WAKE, for bt wakeup host when it is in deep sleep
+ .io = RK2928_PIN0_PC5, // set io to INVALID_GPIO for disable it
```

```

+ .enable = IRQF_TRIGGER_RISING, // set IRQF_TRIGGER_FALLING for falling,
set IRQF_TRIGGER_RISING for rising
+ .iomux = {
+ .name = NULL,
+ },
+ },
+ };
+};

```

### 3.10.3 Android 配置

#### 1、Android 4.2

在 device/rockchip/rkxxsdk/wifi\_bt.mk 中，需要保证 **RDA587X\_BT\_SUPPORT** 配置为 **true**，其它都为 **false**

```

broadcom

BROADCOM_WIFI_SUPPORT := true

BROADCOM_BT_SUPPORT := true

mt5931 && mt6622

MT5931_WIFI_SUPPORT := false

MT6622_BT_SUPPORT := false

rtl8723

RTL8723_BT_SUPPORT := false

rda5876

RDA587X_BT_SUPPORT := true

rtl8723au(USB interface)

RTL8723_BTUSB_SUPPORT := false

```

注：如果这几个选项有修改，记得要 **make clean** 后重新编译

## 3.11 BK3515 蓝牙配置

### 3.11.1 Kernel menuconfig

配置方面， 确保如下选项有勾选上

CONFIG\_RFKILL=y

CONFIG\_RFKILL\_PM=y

CONFIG\_RFKILL\_RK=y

CONFIG\_BT=y

```
CONFIG_BT_L2CAP=y
CONFIG_BT_SCO=y
CONFIG_BT_RFCOMM=y
CONFIG_BT_RFCOMM_TTY=y
CONFIG_BT_BNEP=y
CONFIG_BT_HIDP=y

CONFIG_BT_HCIUART=y
CONFIG_BT_HCIUART_H4=y
CONFIG_BT_HCIUART_BCSP=y
CONFIG_BT_HCIUART_LL=y

CONFIG_SERIAL_CORE=y
CONFIG_SERIAL_RK29=y
CONFIG_UART0_RK29=y
CONFIG_UART0_CTS_RTS_RK29=y
CONFIG_UART0_DMA_RK29=0
```

需要注意，要把 CONFIG\_BT\_AUTOSLEEP 该选项去掉

由于 BK3515 使用的是 WIFI 那边提供的 26M 时钟，请确保 WIFI 配置为 ESP8089，如下：

```
CONFIG_ESP8089=y
CONFIG_BK3515A_COMBO=y
```

### 3.11.2 蓝牙 GPIO 配置

在项目的 board 文件里面修改蓝牙 IO 配置，如下的示：

```
static struct rfkill_rk_platform_data rfkill_rk_platdata = {
 .type = RFKILL_TYPE_BLUETOOTH,

 .poweron_gpio = { // BT_REG_ON, 蓝牙使能脚
 .io = RK30_PIN3_PD1,
 .enable = GPIO_HIGH,
 .iomux = {
 .name = "bt_poweron",
 .fgpio = GPIO3_D1, //GPIO3_C7,
 },
 },

 .reset_gpio = { // BT_RST, 与 poweron 配合，很多模块没有用到该 PIN
```

```

 .io = INVALID_GPIO, // set io to INVALID_GPIO for disable it
 .enable = GPIO_LOW,
 .iomux = {
 .name = "bt_reset",
 .fgpio = GPIO3_D1,
 },
 },

 .wake_gpio = { // BT_WAKE, 与 HOST_UART_TX 复用
 .io = RK30_PIN1_PA1, // set io to INVALID_GPIO for disable it
 .enable = GPIO_LOW,
 .iomux = {
 .name = "gpio1a1_uart0sout_name",
 .fgpio = GPIO1_A1,
 .fmux = UART0_SOUT,
 },
 },

 .wake_host_irq = { // BT_HOST_WAKE, 与 HOST_UART_RX 复用
 .gpio = {
 .io = RK30_PIN1_PA0, // set io to INVALID_GPIO for disable it
 .enable = GPIO_LOW, // set GPIO_LOW for falling, set 0 for rising
 .iomux = {
 .name = "uart0_rx",
 .fgpio = GPIO1_A0,
 .fmux = UART0_SIN,
 },
 },
 },

 .rts_gpio = { // UART_RTS, enable or disable BT's data coming
 .io = RK30_PIN1_PA3, // set io to INVALID_GPIO for disable it
 .enable = GPIO_LOW,
 .iomux = {
 .name = "bt_rts",
 .fgpio = GPIO1_A3,
 .fmux = UART0_RTSEN,
 },
 },
};

```



### 3.11.3 UART clock 设置

目前有发现部分样机的串口 clock 没有配置好导致 UART 波特率偏差太多，蓝牙无法工作；此时需要修改串口的 clock：

修改 kernel 下的 board-xxx.c 文件，在 machine\_rk30\_board\_init()函数中设置 uart0 的 clock 为 16M:

```
clk_set_rate(clk_get_sys("rk_serial.0", "uart"), 16*1000000);
```

### 3.11.4 Android 配置

#### 1、Android 4.2

在 device/rockchip/rkxxsdk/wifi\_bt.mk 中，**需要保证 BK3515\_BT\_SUPPORT 配置为 true，其它都为 false**

```
broadcom
BROADCOM_WIFI_SUPPORT := true
BROADCOM_BT_SUPPORT := false

BK3515
BK3515_BT_SUPPORT := true
```

注：如果这几个选项有修改，记得要 **make clean** 后重新编译

### 3.11.5 BK3515A 支持 2M bps

BK3515A 串口速率默认使用 1Mbps，当提升到 2M bps 时候，上/下行速度会有所提升，依不同机器提升的幅度并不相同。

需要修改的地方如下：

Kernel:

```
Arch/arm/mach-xxx/board-xxxx.c:
static void __init machine_rk30_board_init(void)
{

+ #if defined(CONFIG_BK3515A_COMBO)
+ clk_set_rate(clk_get_sys("rk_serial.0", "uart"), 32*1000000);
+ #endif
}
```

Android:

```
device/common/libbt_beken/include/vnd_rk30sdk.txt
- UART_TARGET_BAUD_RATE = 1000000
+ UART_TARGET_BAUD_RATE = 2000000
device/common/libbt_beken/src/hardware.c
```

```

- //u_int8_t Beken_change_uart_2000000_cmd[15] = {0x01, 0xe0, 0xfc, 0x0b, 0x08,
0x80, 0xba, 0x8c, 0x01, 0x80, 0x84, 0x1e, 0x00, 0x00, 0x00}; //2000000
- serial_vendor_write((uint8_t *)Beken_change_uart_1M_cmd, 15);
+ u_int8_t Beken_change_uart_2M_cmd[15] = {0x01, 0xe0, 0xfc, 0x0b, 0x08, 0x80,
0xba, 0x8c, 0x01, 0x80, 0x84, 0x1e, 0x00, 0x00, 0x00}; //2000000
+ serial_vendor_write((uint8_t *)Beken_change_uart_2M_cmd, 15);

```

## 4 RK 平台 WiFi BT 问题汇总

### 4.1 平台 WiFi 公共性问题汇总

#### 4.1.1 状态栏 WiFi 信号显示的颜色

Android 4.0 以后版本，在连上 WiFi 之后状态栏上显示的信号有灰色和蓝色区分。蓝色表示后台 Google 服务在请求网络数据。

有客户对这个存在疑问，甚至有出现连上 WiFi 之后图标一直都是灰色不会变蓝的情况。这种情况下一些 Google 的应用例如 play store 连不上，但是网络并没有异常，可以正常上网看网络视频，认为是因为 WiFi 的问题导致。

#### 4.1.2 WiFi 打开死机

如果出现 WiFi 在设置中一打开机器就死机或者重启（出现 kernel panic），那可能有以下两种情况：

##### 1) 内核和 WiFi 驱动 ko 文件不匹配导致

ko 编译的内核如果跟当前使用的内核有差异，便可能导致此驱动 ko 文件在当前驱动运行出现一些异常的情况。如果是这种情况，则需要更新 ko 或者重新在当前的内核上编译出新的驱动 ko 文件即可。

##### 2) rk29\_sdmmc\_interrupt 中出现空指针异常

在串口中如下 kernel panic:

```
[6.097388] Unable to handle kernel NULL pointer dereference at virtual address 00000380
[6.112683] pgd = d77c4000
[6.115392] [00000380] *pgd=00000000
[6.118980] Internal error: Oops: 5 [#1] PREEMPT SMP
[6.123950] CPU: 0 Tainted: G C (3.0.36+ #165)
[6.129535] PC is at __raw_spin_lock_irqsave+0x2c/0xb8
[6.134678] LR is at try_to_wake_up+0x28/0x354
[6.767978] [<c0927208>] (__raw_spin_lock_irqsave+0x2c/0xb8) from [<c0473cd4>] (try_to_wake_up+0x28/0x354)
[6.777644] [<c0473cd4>] (try_to_wake_up+0x28/0x354) from [<c0775ec0>] (rk29_sdmmc_interrupt+0x140/0x488)
[6.787223] [<c0775ec0>] (rk29_sdmmc_interrupt+0x140/0x488) from [<c04b9eac>] (handle_irq_event_percpu+0x6c/0x2ac)
[6.797582] [<c04b9eac>] (handle_irq_event_percpu+0x6c/0x2ac) from [<c04ba128>] (handle_irq_event+0x3c/0x5c)
[6.807419] [<c04ba128>] (handle_irq_event+0x3c/0x5c) from [<c04bc4dc>] (handle_fasteoi_irq+0xbc/0x164)
[6.816821] [<c04bc4dc>] (handle_fasteoi_irq+0xbc/0x164) from [<c04b97f8>] (generic_handle_irq+0x28/0x3c)
[6.826400] [<c04b97f8>] (generic_handle_irq+0x28/0x3c) from [<c043d04c>] (asm_do_IRQ+0x4c/0xac)
[6.835195] [<c043d04c>] (asm_do_IRQ+0x4c/0xac) from [<c0442cc4>] (__irq_usr+0x44/0xe0)
```

这可能有以下两种原因：

a) 可能是模块的 IO 电平与主控的 SDIO IO 电平不匹配造成的(只针对 RK31x8)，需要将两者配置成一样。模块端是由模块的 IO 供电来决定，主控制端可以通过函数 `rk31sdk_get_sdio_wifi_voltage` 来调整(具体见 1.2.1.2 节)

b) 可能模块焊接异常，导致 **WiFi SDIO DATA1** 数据线被异常拉底

针对这个 **panic** 问题，软件上可通过以下修改防止死机：

```
diff --git a/include/linux/mmc/host.h b/include/linux/mmc/host.h
@@ -355,6 +355,7 @@ static inline void mmc_signal_sdio_irq(struct mmc_host *host)
{
 host->ops->enable_sdio_irq(host, 0);
 host->sdio_irq_pending = true;
+ if(host && host->sdio_irq_thread)
 wake_up_process(host->sdio_irq_thread);
}
```

### 4.1.3 WiFi 休眠策略

在设置->WiFi 高级设置中的“在休眠状态下保持 WiFi 连接”选项及具体意思：

#### 1 始终：

在机器休眠时保持 WiFi 始终连接，不断线。

#### 2 仅限充电时

在机器休眠并且充电时保持 WiFi 始终连接，不断线。

#### 3 永不（会增加数据网络流量）

在机器休眠时会将 WiFi 继线，继线时间大概为休眠后 15 分钟。

注意：只要 WiFi 的打开的，不管有没有连线，在机器进入休眠时，WiFi 模块的供电是不会断的，也不能断开，一旦供电被断开，WiFi 就会工作异常。表现出的现象是，机器唤醒后 WiFi 无法使用，需要先关闭再打开才能恢复正常。

### 4.1.4 机器休眠时后台文件下载中断

目前 WiFi 默认没有打开 wake up host 功能，也就是机器进入二级休眠后，WiFi 有接收到消息时，无法将主控唤醒。可通过修改以下部分来支持 wake up host 功能：(注意：这个修改只针对带有 wake up host 功能的 WiFi，例如 RK901, RK903, AP6xxx, MT5931 等，USB WiFi 目前无法支持。)

## 1. menuconfig 配置

选择 “sdio-irq from gpio ”

Location:

-> Device Drivers

-> MMC/SD/SDIO card support

-> RK29 SDMMC controller support

-> RK29 SDMMC1 controller support(sdio)

```
<*> RK29 SDMMC1 controller support(sdio)
[] write-protect for SDMMC1
[*] sdio-irq from gpio
```

## 2. 在 arch/arm/mach-rk30/board-xxx-sdmmc-config.c 中配置如下中断:

```
#define RK30SDK_WIFI_GPIO_WIFI_INT_B RK30_PIN3_PD2 // WiFi wake up host 中断脚
#define RK30SDK_WIFI_GPIO_WIFI_INT_B_ENABLE_VALUE GPIO_HIGH // 中断有效电平
```

## 3. 增加唤醒主控制代码

```
diff --git a/drivers/mmc/host/rk29_sdmmc.c b/drivers/mmc/host/rk29_sdmmc.c
@@ -3840,7 +3840,8 @@ static int rk29_sdmmc_probe(struct platform_device *pdev)
 host->errorstep = 0x8D;

 goto err_dmaunmap;
}

disable_irq_nosync(host->sdio_irq);
+ enable_irq_wake(host->sdio_irq);
}
```

### 4.1.5 机器休眠再唤醒后 WiFi 无法使用

机器休眠再唤醒后 WiFi 无法使用，需要重新关闭再打开 WiFi 才能恢复正常。  
这一般是因为 WiFi 的 power 或 reset 被异常关掉导致的，机器休眠时，WiFi 驱动不会将 power 与 reset 关掉。需要查一下 power 与 reset 是不是被其它模块干扰到了。

### 4.1.6 使用 SDIO 接口 WiFi 机器不断重启

在串口中如下 kernel panic:

```
[6.097388] Unable to handle kernel NULL pointer dereference at virtual address 00000380
[6.112683] pgd = d77c4000
[6.115392] [00000380] *pgd=00000000
[6.118980] Internal error: Oops: 5 [#1] PREEMPT SMP
[6.123950] CPU: 0 Tainted: G C (3.0.36+ #165)
[6.129535] PC is at __raw_spin_lock_irqsave+0x2c/0xb8
```

```
[6.134678] LR is at try_to_wake_up+0x28/0x354
[6.767978] [<c0927208>] (__raw_spin_lock_irqsave+0x2c/0xb8) from [<c0473cd4>] (try_to_wake_up+0x28/0x354)
[6.777644] [<c0473cd4>] (try_to_wake_up+0x28/0x354) from [<c0775ec0>] (rk29_sdmmc_interrupt+0x140/0x488)
[6.787223] [<c0775ec0>] (rk29_sdmmc_interrupt+0x140/0x488) from [<c04b9eac>] (handle_irq_event_percpu+0x6c/0x2ac)
[6.797582] [<c04b9eac>] (handle_irq_event_percpu+0x6c/0x2ac) from [<c04ba128>] (handle_irq_event+0x3c/0x5c)
[6.807419] [<c04ba128>] (handle_irq_event+0x3c/0x5c) from [<c04bc4dc>] (handle_fasteoi_irq+0xbc/0x164)
[6.816821] [<c04bc4dc>] (handle_fasteoi_irq+0xbc/0x164) from [<c04b97f8>] (generic_handle_irq+0x28/0x3c)
[6.826400] [<c04b97f8>] (generic_handle_irq+0x28/0x3c) from [<c043d04c>] (asm_do_IRQ+0x4c/0xac)
[6.835195] [<c043d04c>] (asm_do_IRQ+0x4c/0xac) from [<c0442cc4>] (__irq_usr+0x44/0xe0)
```

这可能有以下两种原因：

- 1) 可能是模块的 IO 电平与主控的 SDIO IO 电平不匹配造成的(只针对 RK31x8)，需要将两者配置成一样。模块端是由模块的 IO 供电来决定，主控制端可以通过函数 `rk31sdk_get_sdio_wifi_voltage` 来调整(具体见 1.2.1.2 节)

## 4.2 RK903 & RK901 & AP6xxx 问题汇总

### 4.2.1 设置中 WiFi 打不开

设置中打开 WiFi 无法打开，一般情况下主要有以下几个原因，可按如下步骤进行排查：

A、连接串口或者通过 ADB 连接打印内核消息，打开 wifi 查看内核的打印信息，确保打开 wifi 时加载的驱动为 `rkwifi.ko`，判断依据如图 4-1 所示。并确保内核配置无误，硬件上使用 RK903 则内核必须配置 RK903 的选项，RK903、RK901 和 BCM4330 等都不可混用。

```
=====
==== Launching Wi-Fi driver! (Powered by Rockchip) ====
=====
RKWIFI WiFi driver (Powered by Rockchip, Ver 4.25) init.
dhd_module_init: Enter
===== WLAN placed in POWER ON =====
ANDROID-ERROR) ## wifi_probe
ANDROID-ERROR) wifi_set_power = 1
rk29sdk_wifi_power: 1
wifi turn on power
ANDROID-ERROR) wifi_set_carddetect = 1
rk29sdk_wifi_set_carddetect:1
```

图 4-1

B、若确定驱动和内核配置无误，若打开 wifi 时，内核驱动出现如下图 4-2 所示的打印信息说明内核 SDIO 驱动没有配置。需要配置好 SDIO 驱动，具体参考“RK 平台中 WiFi 的内核配置”。

```
=====
==== Launching Wi-Fi driver! (Powered by Rockchip) ====
=====
RKWIFI WiFi driver (Powered by Rockchip, Ver 4.25) init.
dhd_module_init: Enter
===== WLAN placed in POWER ON =====
ANDROID-ERROR) ## wifi_probe
ANDROID-ERROR) wifi_set_power = 1
rk29sdk_wifi_power: 1
acc_open
acc_release
wifi turn on power
ANDROID-ERROR) wifi_set_carddetect = 1
rk29sdk_wifi_set_carddetect:1
rk29sdk_wifi_set_carddetect, nobody to notify
Linux Kernel SDIO/MMC Driver

Dongle Host Driver, version 5.90.195.26.1.9
Compiled in drivers/net/wireless/bcmdhd on Sep 28 2012 at
```

图 4-2

C、若确定驱动和内核配置无误，若打开 wifi，内核加载驱动过程中是否出现如下打印信息：

```
rk29_sdmmc_command_complete..2935...CMD55(arg=0x0), hoststate=1, errorTimes=1, errorStep=0x1e ! [sdio]
rk29_sdmmc_command_complete..2935...CMD55(arg=0x0), hoststate=1, errorTimes=1, errorStep=0x1e ! [sdio]
rk29_sdmmc_command_complete..2935...CMD55(arg=0x0), hoststate=1, errorTimes=1, errorStep=0x1e ! [sdio]
rk29_sdmmc_command_complete..2935...CMD55(arg=0x0), hoststate=1, errorTimes=1, errorStep=0x1e ! [sdio]
rk29_sdmmc_command_complete..2935...CMD55(arg=0x0), hoststate=1, errorTimes=1, errorStep=0x1e ! [sdio]
rk29_sdmmc_command_complete..2935...CMD1(arg=0x0), hoststate=1, errorTimes=1, errorStep=0x1e ! [sdio]
```

如果有则说明当前 SDIO 没有识别到网卡设备，一般是因为模组上电异常或者硬件上连线的问题。分别进行排查，在设置中打开 WiFi 马上使用万用表或者示波器测量模组的第 30 脚 VBAT 和第 18 脚 VDDIO 电压是否正常，若模组使用 RK901 建议 VDDIO 使用 3.0V。

如果模组供电正常，请检查硬件连线包括 4 根 DATA 线和 CMD 连线是否有误（RK2818 SDIO 没有内部上拉，需要使用上拉电阻对 4 根 DATA 线和 CMD 线进行外部上拉）。检查晶振是否起振。

另外，在 PCB 画板阶段请确保 SDIO 和 WiFi 模组的连线等长并避免干扰。避免数据通讯的问题。

D、若 WiFi 供电无异常，请检查 WiFi 模组第 31 脚 WL\_REG\_ON 是否在设置中打开 WiFi 时有被拉高。该脚对 RK90X 内部进行上下电的控制，由主控的一个 GPIO 进行电平控制，若该脚没有拉高则控制有误，请检查内核文件：

arch/arm/mach-rkxx/board-rk30-sdk-sdmmc.c 中 RK30SDK\_WIFI\_GPIO\_POWER\_N 的定义是否和实际硬件设计一致。

## 4.2.2 信号强度差

WiFi 的信号强度主要和整机硬件和 WiFi 的 RF 天线设计有关，如果对机器 WiFi 的信号有疑问可以取使用相同 WiFi 的机器和手机等在同一个位置进行对比，包括对比扫描 AP 的数量以及 AP 的信号强度。

当发现 WiFi 信号强度对比明显较差，可通过以下步骤进行排查：



- A、WiFi 天线是否已经匹配过，天线阻抗不匹配会导致功率衰减影响通讯效能。
  - B、WiFi 的 RF 指标是否 OK，若还没有确认过，建议先测试并调整下指标。
  - C、若机器为金属外壳，将外壳拆除看信号是否正常，若正常，则说明是因为模具设计不当导致。
  - D、查看机器 PCB 板是否已经加屏蔽处理，查看 DDR 频率是否会影响 WiFi，计算方法可以将 DDR 频率 \* n (n 为整数，如 6、7 分别代表 6 倍频和 7 倍频)。DDR 频率 n 次倍频若落在 WiFi 的工作频段内(2.412 GHz — 2.484GHz) 会对 WiFi 存在一定的影响。
- WiFi 信号容易受机器其他硬件辐射干扰，如果机器外壳是金属则硬件设计不当将导致 WiFi 信号被极大影响。在机器 PCB 设计以及整机模具设计之时应该注意这方面的问题。

#### 4.2.3 RK903 设置中可以打开 WiFi 但是扫描列表中没有扫描到 AP

设置中可以打开 WiFi，但是在扫描列表中没有扫描到 AP，若当前周边确实存在范围内的 AP，需要确认下硬件上 WiFi 模组使用的晶振频率和内核实际配置的晶振是否一致，RK903 支持 26M 和 37.4M 两种规格的晶振，在内核中通过配置项进行区分，如下图 4-3 所示。使用的晶振频率和内核配置不一致会出现可以打开 WiFi 但是无法扫描到 AP 这个情况。

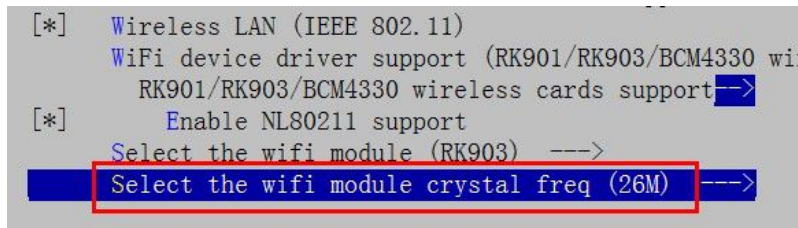


图 4-3

#### 4.2.4 WiFi 使用中出现异常不能连接 AP，需要关闭再打开才能恢复

如果在 WiFi 的正常使用中突然出现网络断开，并不能重新连接 AP，确认下当前 AP 是否正常，用其他设备例如手机连接这台 AP 试试，如果 AP 正常，将机器用串口或者 ADB 打印出内核信息，如果内核打印中有出现如下图 4-4 所示的内容则说明当前 WiFi 模组出现异常导致模组崩溃。

```
dhdcdc_query_ioctl: dhdcdc_msg failed w/status -110
dhd_check_hang: Event HANG send up due to re=0 te=8 e=-110 s=0
CFG80211-ERROR) wl_cfg80211_hang : In : chip crash eventing
dhd_open: failed with code -110
dhdsdio_isr : bus is down. we have nothing to do

Dongle Host Driver, version 5.90.195.26.1.9
Compiled in drivers/net/wireless/bcmdhd on Sep 28 2012 at 16:18:05
Fianl fw_path=/system/etc/firmware/fw_RK901a0.bin
Fianl nv_path=/system/etc/firmware/nvram_RK901.txt
dhdsdio_isr : bus is down. we have nothing to do
dhdsdio_isr : bus is down. we have nothing to do
dhdsdio_isr : bus is down. we have nothing to do
dhdsdio_isr : bus is down. we have nothing to do
dhdsdio_isr : bus is down. we have nothing to do
```

图 4-4

这种情况根据出现的概率的大小有不同的处理方法。

- A、如果概率较高，上网看网络视频等很容易出现，则说明当前模组工作状态不稳定，需要检查下模组供电电压纹波，包括 WiFi 模组的 VBAT 和 VDDIO。如果供电负债不够，在上网等大数据传输时，WiFi 功耗增大，可能拉低电压导致模组供电不足而崩溃。如果使用的是 RK901 而 VDDIO



电压为 1.8V，可尝试将 VDDIO 电压提高到 3V 再测试。

如果电压正常，则需要检查是否 SDIO 和模组的 data 连线不等长或者 PCB 布线受干扰导致数据通讯异常。

B、如果概率非常低，很难重现，而模组电压，布线都没有问题，那可能是因为模组本身的系统不稳定，可打上下面的补丁来实现自我恢复功能。当模组出现异常导致崩溃时，上层会自动重启 WiFi 使恢复正常工作状态。修改的文件路径为：

```
Android/external/wpa_supplicant_8/wpa_supplicant/src/drivers/driver_nl80211.c
--- a/src/drivers/driver_nl80211.c
+++ b/src/drivers/driver_nl80211.c

@@ -410,7 +410,17 @@ static int no_seq_check(struct nl_msg *msg, void *arg)
{
 return NL_OK;
}

+static int drv_errors = 0;

+static void wpa_driver_send_hang_msg(struct wpa_driver_nl80211_data *drv)
+{
+ drv_errors++;
+ if (drv_errors > DRV_NUMBER_SEQUENTIAL_ERRORS) {
+ drv_errors = 0;
+ wpa_msg(drv->ctx, MSG_INFO, WPA_EVENT_DRIVER_STATE "HANGED");
+ }
+}

static int send_and_recv(struct nl80211_global *global,
 struct nl_handle *nl_handle, struct nl_msg *msg,
@@ -1179,6 +1189,11 @@ static void mlme_event_disconnect(struct wpa_driver_nl80211_data *drv,
 data.disassoc_info.reason_code = nla_get_u16(reason);
 data.disassoc_info.locally_generated = by_ap == NULL;
 wpa_supplicant_event(drv->ctx, EVENT_DISASSOC, &data);
+// fix hang issue
+ if (data.disassoc_info.reason_code == WLAN_REASON_UNSPECIFIED) {
+ drv_errors = 4;
+ wpa_driver_send_hang_msg(drv);
+ }
}

@@ -8834,6 +8849,7 @@ typedef struct android_wifi_priv_cmd {
 int total_len;
} android_wifi_priv_cmd;
```

```
+/*

static int drv_errors = 0;

static void wpa_driver_send_hang_msg(struct wpa_driver_nl80211_data *drv)
@@ -8844,7 +8860,7 @@ static void wpa_driver_send_hang_msg(struct wpa_driver_nl80211_data *drv)
 wpa_msg(drv->ctx, MSG_INFO, WPA_EVENT_DRIVER_STATE "HANGED");
}
}
+*/
```

#### 4.2.5 WiFi 热点打不开

WiFi 能够打开，但是 WiFi 热点打不开，主要分以下几点进行分析：

A、如果使用的是 RK901，则先确认是否驱动、nvram 和 firmware 都已经更新到最新版本。

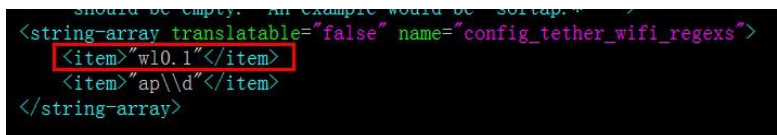
B、是否 Android 代码有过人为修改，对比发布的 SDK 代码中以下的几个文件查看差异的代码，如果存在差异而客户无法判断可将对比的结果发 RK 的 WiFi 工程师进行分析。

Android/hardware/libhardware\_legacy/wifi/wifi.c

Android/system/netd/CommandListener.cpp

Kernel/drivers/net/wireless/wifi\_sys/rkwifi\_sys\_iface.c

C、在文件 Android/frameworks/base/core/res/res/values/config.xml 中查找字符串“config\_tether\_wifi\_regexs”，查看如下图 4-5 的内容是否存在图中红色方框标识的内容。



```
<string-array translatable="false" name="config_tether_wifi_regexs">
 <item>wl0.1</item>
 <item>ap\\d</item>
</string-array>
```

图 4-5

#### 4.2.6 WiFi 连接不上某些 AP 或上网速度很慢

具体现象为：能够扫描到 AP，信号也很好，但是始终连接不上。这很可能是 WiFi 模块晶体偏频过大引起的。

由于 RK903 及 AP6xxx 系列需要外接晶体，晶体匹配电容如果没有校准好的话，很容易造成晶体偏频过大。需要拿机器到我们公司校准一下频偏。

#### 4.2.7 BT 能够描述到设备但是无法匹配上

这很可能是模块晶体偏频过大引起的。解决方法同上。

#### 4.2.8 BPLUS 导致的 CTS 测试失败

打开了 BLUETOOTH\_USE\_BPLUS 之后，在进行 CTS 测试时候报如下的错误。

Compatibility Test Package: android.security

Test	Result	Details
android.security.cts.ListeningPortsTest		
-- testNoListeningLoopbackTcpPorts	fail	junit.framework.AssertionFailedError: Found port listening on addr=127.0.0.1, port=10001, UID=1002 in /proc/net/tcp

处理方法：最近的代码更新 bplus 为 1.0.6.1 版本，请更新。

#### 4.2.9 AP6476 打开失败

在上电时候，BT\_HOST\_WAKE 需要高电平

#### 4.2.10 AP6330 打开失败

目前发现有部分样机，在打开了 BPLUS 之后，蓝牙打开失败，可按如下方法修正：

把/etc/bluetooth/bt\_vendor.conf 文件中的 RegonDelay 时间从原来的 500ms 缩短为 200ms

### 4.3 RTL8188CUS & RTL8188EUS 问题汇总

#### 4.3.1 设置中 WiFi 打不开

USB WiFi 如果出现设置中打不开的情况，可以从以下几个步骤进行排查：

A、首先仍然要确认在打开 WiFi 时内核加载的驱动和实际使用的 WiFi 是配对的，因为 rtl8188cus 和 rtl8188eus 本身名字很相似，但是本身使用不同的两份驱动，出现过不少客户机器使用的是 rtl8188cus，但是内核配置的驱动是 rtl8188cus。所以非常有必要确认清楚，查看机器上 WiFi 模組的名字。然后在设置中打开 WiFi 查看内核打印信息，加载驱动时会打印驱动的版本和驱动的类型，如下图 4-6 所示，红色框为驱动版本号，黄色框为驱动的类型。

```
==== Launching Wi-Fi driver! (Powered by Rockchip) ====
Realtek 8188EU USB Wi-Fi driver (Powered by Rockchip, Ver 1.11) init.
wifi_usb_init
wifi_activate_usb
usbcore: registered new interface driver rtl8188eu
```

图 4-6

B、在确定驱动没有问题之后，如果模组是 GPIO 控制上下电（原则上都要求硬件上实现 WiFi 上下电可控），在设置中打开 WiFi，然后测量 WiFi 模組的供电脚电压是否正常，若供电正常应该

可以在内核打印中出现如下图 4-7 所示的内容，为 USB host 枚举设备的信息。

```
usb 1-1: new high speed USB device number 3 using usb20_host
usb 1-1: New USB device found, idVendor=0bda, idProduct=8179
usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 1-1: Product: 802.11n NIC
usb 1-1: Manufacturer: Realtek
usb 1-1: SerialNumber: 00E04C0001
```

图 4-7

如果没有出现图 2-8 的内容，说明当前 USB host 跟模组的连线有问题，或者模组本身有异常。检查硬件连线或者更换模组测试。

正常情况下只要 USB host 可以枚举到设备，然后驱动无误，那么 WiFi 便可以正常执行初始化并打开。但有一种情况，如果给 WiFi 模组供电的电源负载能力较差，在 WiFi 执行较大功率操作的时候会出现因为负载不足而拉低电压导致模组崩溃的现象，这个情况下可以在内核的打印信息中出现如下图 4-8 所示的内容。这个时候 host 跟网卡直接连接断开了，WiFi 驱动运行就会出现异常导致错误。

```
usb 1-1: USB disconnect, device number 2
```

图 4-8

C、如果模组的供电是一直供电的方式，目前发现在 rtl8188cus 使用一直供电的方式不会有异常，而如果在 rtl8188eus 使用一直供电的方式，会出现恢复出厂设置或者反复开关 WiFi 导致 USB host 没有成功枚举到网卡设备或者因为 host 和网卡设备断开连接而 WiFi 无法使用必须重启机器（原则上要求所有的 WiFi 都具备上下电控制功能）。

### 4.3.2 设置中打开 WiFi 很慢（十几秒钟）

如果使用 USB WiFi 出现在设置中打开 WiFi 需要很久（十几秒）才能打开，打开之后 WiFi 都正常，则只要将 android 上层硬件抽象层 wifi.c 文件更新下即可，更新到目前 SDK Git 服务器最新的即可解决。

### 4.3.3 WiFi 热点打不开

不同厂家的 WiFi 针对 Softap 实现的方式不一样，在 /system/netd/ 目录下集合了关于 Softap 的控制办法，目前包含了 Broadcom、Realtek 和 MTK。通过 CommandListener.cpp 文件中进行对应的方式选择，如下图 4-9 所示。

```

if (!sSoftapCtrl) {
 int chip_type = check_wifi_chip_type();
#ifdef MT5931
 sSoftapCtrl = new SoftapController();
#elif defined(MTK_MT6620_HOTSPOT_SUPPORT)
 sSoftapCtrl = new SoftapController();
#else
 if((chip_type == RTL8188CU) || (chip_type == RTL8188EU) || (chip_type == RTL8723AS))
 sSoftapCtrl = new SoftapController_rtl();
 else if(chip_type == RT5370)
 sSoftapCtrl = new SoftapController_rt5370();
 else
 sSoftapCtrl = new SoftapController();
#endif
}

```

图 4-9

所以，如果 Softap 打不开可以先通过 android 的 log 判断当前是否走的正确的分支，例如如果 WiFi 使用的是 RTL8188XX 的 WiFi，那么当打开热点时，在 android log 中执行的 SoftapController 应该是 SoftapController\_rtl.cpp 中的对象。打印信息中可以看到“SoftapController\_rtl”的打印信息。否则说明当前分支不正确，需要去检查函数“check\_wifi\_chip\_type()”执行是否有误。

如果这个部分没有问题，则可参考 RK903 的 Softap 问题问题的排查方法进行排查。

#### 4.3.4 信号强度差、扫描的 AP 数量较少

信号的问题基本都是硬件问题，单纯软件上无法解决，类似增大发射功率等无需考虑，功率加大会导致 EVM 变差会降低信噪比对实际增强通讯能力并没有益处。USB WiFi rtl8188cus 和 rtl8188eus 都是使用 PCB 模组的方式，在出厂时都已经校准好指标，一般情况下指标问题不大，所以如果出现信号强度较差，扫描的 AP 数量较少，可以从以下几个方面进行排查：

A、天线是否已经匹配过，不同的硬件和模组对于天线的阻抗匹配存在差异，一个新的机器应该重新进行天线的匹配，包括匹配电阻电容，匹配过的天线才能发挥更高的功率效率。

B、是否金属外壳，采用金属外壳如果硬件设计不当会导致信号的削弱，可以尝试将外壳拆掉测试，如果拆掉外壳后 WiFi 信号恢复正常，则说明模具设计等不合理，需要排查硬件或者模具的问题。

C、WiFi 容易受 PCB 上其他部分硬件的电磁辐射干扰，建议对 DDR 等容易产生干扰的电路加屏蔽罩处理，在 PCB 设计之初就对 WiFi 的干扰考虑在内，尽量靠近外围边缘，附近尽量不要有大电感元件或者较强的辐射源存在，模组引出来的 RF PCB 线路避免使用直角，预留 RF 匹配电感电容座。

#### 4.3.5 信号较差时上网慢

在信号较差（1 格到 2 格或者 RSSI 小于 -75dbm）时，上网比较卡，或者打不开网页。这种情况说明当前 WiFi 模组和 AP 之间数据通讯能力差，一般情况下跟硬件设计和环境的干扰会有较大关系，如果周围 AP 数量较多，同一个通道存在多个 AP，这样会产生较大的信号干扰，信号较差时，容易出现丢包误码的情况。



同时跟机器本身硬件和 WiFi 天线设计有关，不同型号的模组也可能存在差异，不过一般差异不大都在一个指标范围内，只是灵敏度方面可能有些差异。在实际使用上 rtl8188cu 相比 rtl8188eu 效果相对会好一些。

## 4.4 RT5370 & MT7601 问题汇总

RT5370&MT7601 跟 RTL8188EUS 一样都是 USB 接口的 WiFi，针对这个问题可以参考 RTL8188EUS 的处理办法。

需要特别注意是

<1>MT7601 如果在使用 Soft AP 以及 Direct 功能的时候，出现界面卡顿，Wi-Fi 无法再关闭和打开，需要重启机器的现象时，KMSG 出现如下异常信息提示

```
72.181015] -----[cut here]-----
[72.181133] WARNING: at kernel/softirq.c:159 local_bh_enable_ip+0x90/0xc4()
[72.181267] [<044d4e4>] (unwind_backtrace+0x0/0xf8) from [<047d584>] (warn_slowpath_common+0x4c/0x64)
[72.181398] [<047d584>] (warn_slowpath_common+0x4c/0x64) from [<047d5b8>] (warn_slowpath_null+0x1c/0x24)
[72.181526] [<047d5b8>] (warn_slowpath_null+0x1c/0x24) from [<0484080>] (local_bh_enable_ip+0x90/0xc4)
[72.181688] [<0484080>] (local_bh_enable_ip+0x90/0xc4) from [<08daa3c>] (nl80211_dump_scan+0x3e0/0x564)
[72.181843] [<08daa3c>] (nl80211_dump_scan+0x3e0/0x564) from [<07cb6bc>] (netlink_dump+0x54/0x1c4)
[72.181972] [<07cb6bc>] (netlink_dump+0x54/0x1c4) from [<07cbdc8>] (netlink_dump_start+0x19c/0x1f8)
[72.182100] [<07cbdc8>] (netlink_dump_start+0x19c/0x1f8) from [<07ce6ec>] (genl_rcv_msg+0xc4/0x1f4)
[72.182227] [<07ce6ec>] (genl_rcv_msg+0xc4/0x1f4) from [<07cdd0c>] (netlink_rcv_skb+0x9c/0xd8)
[72.182347] [<07cdd0c>] (netlink_rcv_skb+0x9c/0xd8) from [<07ce620>] (genl_rcv+0x1c/0x24)
[72.182463] [<07ce620>] (genl_rcv+0x1c/0x24) from [<07cd680>] (netlink_unicast+0x2a0/0x2fc)
[72.182582] [<07cd680>] (netlink_unicast+0x2a0/0x2fc) from [<07cda10>] (netlink_sendmsg+0x294/0x30c)
[72.182716] [<07cda10>] (netlink_sendmsg+0x294/0x30c) from [<07935f4>] (sock_sendmsg+0x9c/0xb0)
[72.182841] [<07935f4>] (sock_sendmsg+0x9c/0xb0) from [<07953a8>] (_sys_sendmsg+0x2bc/0x2d4)
[72.182962] [<07953a8>] (_sys_sendmsg+0x2bc/0x2d4) from [<0795e64>] (sys_sendmsg+0x3c/0x68)
[72.183082] [<0795e64>] (sys_sendmsg+0x3c/0x68) from [<0447f80>] (ret_fast_syscall+0x0/0x30)
[72.183186] ---[end trace 19a386a826e02380]---
```

此时需要客户确认，在所使用的内核中是否有选择 NL80211\_TESTMODE 这个配置。

<2>如果客户在测试 Direct 连接过程中发现比较难以连接，请客户确认两件事情

(1) 客户的天线是否有匹配好，硬件上是否有干扰源

(2) Wpa\_supplicant 是否是针对 7601 做个处理的，可以确认以下文件的代码段是否存在，如果没有相关的配置段，请更新服务器

```
#(!!!!!!)MT7601 SHOULD NOT use ANDROID_P2P definition!!!!!!
if [$(strip ${MT7601U_WIFI_SUPPORT})]; then
L_CFLAGS = -DWPA_IGNORE_CONFIG_ERRORS
L_CFLAGS += -DVERSION_STR_POSTFIX=\"-${PLATFORM_VERSION}\"
L_CFLAGS += -DANDROID_LOG_NAME=\"wpa_supplicant\"
endif
```

<3> 如果客户无法使用 station 功能进行正常的上网(也就是打开 wifi 失败)，并出现如下 log

```
[28.619908] mt7601Usta: Unknown symbol RTMPQMemAddr (err 0)
```

则请客户如下几点

(1) system/lib/modules/目录下是否存在 mtprealloc7601Usta.ko 这个文件，并确保权限是 0644

(2) Init.rk30board.rc 是否存在加载此驱动的配置段落，如果没有请添加 insmod /system/lib/modules/mtprealloc7601Usta.ko

<4>如果概率性的打开 softap 失败，请确认/system/net/softapcontroller\_mt7601.cpp 中

int SoftapController\_mt7601::startSoftap() 函数内

```
85 ret = wifi_load_ap_driver();
86 sleep(3); //是否有加了延时
87 if(ret < 0)
88 {
89 ALOGE("wifi_load_ap_driver *****");
90 }
```

## 4.5 MT5931 & MT6622 问题汇总

### 4.5.1 设置中 WiFi 打不开

#### 1. 出现 kernel 如下 SDIO 通信出错

```
rk29_sdmmc_command_complete..2935...CMD5(arg=0x0), hoststate=1, errorTimes=1, errorStep=0x1e ! [sdio]
rk29_sdmmc_command_complete..2935...CMD55(arg=0x0), hoststate=1, errorTimes=1, errorStep=0x1e ! [sdio]
rk29_sdmmc_command_complete..2935...CMD55(arg=0x0), hoststate=1, errorTimes=1, errorStep=0x1e ! [sdio]
rk29_sdmmc_command_complete..2935...CMD55(arg=0x0), hoststate=1, errorTimes=1, errorStep=0x1e ! [sdio]
rk29_sdmmc_command_complete..2935...CMD55(arg=0x0), hoststate=1, errorTimes=1, errorStep=0x1e ! [sdio]
rk29_sdmmc_command_complete..2935...CMD1(arg=0x0), hoststate=1, errorTimes=1, errorStep=0x1e ! [sdio]
```

需要确认以下几点：

- 1) 在打开 WiFi 时，模块的 power 与 reset 脚是否上电成功。
- 2) SDIO (DATA0-3, CMD, CLK) 电平是否正常，一般 SDIO 电平需要 3.3V（或 1.8V）左右，是否需要接上拉电阻。

### 4.5.2 BT 打不开

使用 `logcat -s [BT]` 打印：

A. 出现如下打印后就没有其它打印：

**GORMcmd\_HCC\_Read\_Local\_Version: GORMcmd\_HCC\_Read\_Local\_Version**

这表示主控制发出第一条 UART 命令后，BT 模块没有响应。

首先确认 BT 模块的 power 与 reset 脚是否上电了。

如果正常上电了，那么需要确认如下 UART 情况：

- 1) 默认配置的是 UART0 口，请确认跟硬件是否配置
- 2) UART 口的 TX, RX 连接是否正常，是否可能接反了
- 3) 32K RTC\_CLK 是否接上
- 4) 由于没有硬件流控，UART 可能受 RTC, CTS 脚的影响，需要主控与模块的 RTC, CTS 脚断

开。或者软件上将 RTS, CTS 切换成 GPIO 并拉底。

B. 出现如下打印后就没有其它打印：

**set\_speed: standard baudrate: 1500000 -> 0x0000100a**

这表示 UART 波特率从 115200 切换到 1500000 后，通信出错。

这是因为 UART clock 配置不准确导致分不出精确的 1500000 的波特率。

在 arch/arm/mach-rkxx/board-xxx-sdk.c，中的

\_\_init machine\_rk30\_board\_init 函数中添加

**clk\_set\_rate(clk\_get\_sys("rk\_serial.0", "uart"), 24\*1000000);**

### 4.5.3 BT 工作不稳定

出现无法扫描或者无法传送文件，或者传送文件失败；

出现无法接收文件，或者概率性无法接收文件，接收文件中断；

可能有以下几种原因：

A. 可能 BT 中断脚有异常，

可通过 cat proc/interrupts 查看 BT 的中断情况

282:                    2                    0                    0                    0                    GPIO   BT\_INT\_B

1). 如果查看到中断数为 0，需要先查一下是否产生硬件中断：

BT\_INT\_B 中断脚如果通过三极管反相到主控，有可能是三极管脚状态不稳定导致的，可以去掉三极管排查一下

2). 软件里设置的默认中断有效电平是高电平，有可能跟硬件不匹配，将板级文件里的 mt6622\_platdata 设备里的**中断有效电平配置设置成低电平有效试试。**

B. 32.768KHz RTC Clock 异常

32.768KHz RTC Clock 的幅度有要求，不能过小，需要 1V 以上

C. BT\_INT\_B 所使用的 GPIO 是否可能被其它模块共用了，导致无法产生中断。

D. 如果是 RK3066 平台，出现接收文件或发送文件中断，需要修改如下：

```
--- a/kernel/arch/arm/mach-rk30/include/mach/board.h
+++ b/kernel/arch/arm/mach-rk30/include/mach/board.h

@@@ -75,7 +75,7 @@@ enum_codec_pll {

else

-#define RK30_CLOCKS_DEFAULT_FLAGS (CLK_FLG_MAX_I2S_12288KHZ/*(CLK_FLG_EXT_27MHZ*)
+#define RK30_CLOCKS_DEFAULT_FLAGS (CLK_FLG_MAX_I2S_12288KHZ|CLK_CPU_HPCLK_11/*(CLK_FLG_EXT_27MHZ*)

#if (RK30_CLOCKS_DEFAULT_FLAGS&CLK_FLG_UART_1_3M)
```



## 4.6 ESP8089 & BK3515 问题汇总

### 4.6.1 蓝牙打开失败

蓝牙的 CTS/RTS 没有与 HOST 端连接，飞线连接上后正常。

MTK 二合一的模块，没有使用硬件流控；当机型是参照 MTK 模块来布板时，会存在这个问题

### 4.6.2 蓝牙不能收发文件，听音乐卡

测量串口速率只有115200， 经确认后是 libbt 用 broadcom 的，没有使用 beken 的 libbt 导致，重新编译 device/common/libbt\_beken/ 得到 libbt-vendor.so

## 4.7 RK Bluetooth(Android 4.1/4.2)问题汇总

注意：4.7.1-4.7.3 节部分只适用于 **broadcom** 系列蓝牙芯片，例如 RK903, AP6xxx 系列

### 4.7.1 前提

在查问题之前，需先掌握这些东西：

#### 捕捉 LOG

Kernel 的启动 LOG，在串口中会有输出，蓝牙的 rfkill-rk 驱动，其打印以字符串"**[BT\_RFKILL]**"作为前缀，可搜索之；如果你用“adb shell”来查看 log，可先切换到 root 后，用“cat /proc/kmsg”打印 kernel 的 LOG。

Android 中关于蓝牙的 LOG，可通过如下命令打印：

```
logcat -s blue* Blue* Bt* bt* &
```

如果你的 Android 的 logcat 命令不支持星号通配符，请修改 system/core/liblog/logprint.c 文件，如下所示：

```
static android_LogPriority filterPriForTag(
 AndroidLogFormat *p_format, const char *tag)
{
 FilterInfo *p_curFilter;

 for (p_curFilter = p_format->filters;
 p_curFilter != NULL;
 p_curFilter = p_curFilter->p_next)
 {
 if ((p_curFilter->mTag[strlen(p_curFilter->mTag)-1] == '*' &&
 0 == strcmp(tag, p_curFilter->mTag, strlen(p_curFilter->mTag)-1)) ||
 (0 == strcmp(tag, p_curFilter->mTag)))
 {
 return p_curFilter->mPriority;
 }
 }
 return ANDROID_LOG_DEFAULT;
}
```

```
 }
 if (p_curFilter->mPri == ANDROID_LOG_DEFAULT) {
 return p_format->global_pri;
 } else {
 return p_curFilter->mPri;
 }
}
}
}

return p_format->global_pri;
}
```

#### 4.7.2 命令行启动蓝牙

通过 su 命令切换到 root 用户

- 1、先确认 RFKILL 驱动已经加载

```
ls /sys/class/rfkill/rfkill0/
```

如果没有找到 rfkill0 这个目录，说明蓝牙驱动有问题。

请检查 kernel 中的蓝牙选项是否有勾选了

请查看 kernel 的打印信息中以“[BT\_RFKILL]”打头的信息。

- 2、关闭蓝牙：

A. 在 Settings 界面中关闭蓝牙

B. 给蓝牙设备下电：

```
echo 0 > /sys/class/rfkill/rfkill0/state
```

C. 关闭 bluetoothd 和 hciattach 这两个 service:

```
setprop ctl.stop bluetoothd
```

```
setprop ctl.stop hciattach
```

D. 对于 Android4.2，除了上述三步之外，还需要关闭进程 com.android.bluetooth

```
busybox killall com.android.bluetooth
```

- 3、确定蓝牙已经关闭之后，手动给蓝牙上电：

```
echo 1 > /sys/class/rfkill/rfkill0/state
```

[UART 状态] 蓝牙上电之后，HOST\_UART\_CTS 应当为低电平，且 HOST\_UART\_TX/RX 都应当为高电平

- 4、下载蓝牙固件

Broadcomd 系列芯片使用命令：

RK30:

```
brcm_patchram_plus --patchram bychip --baudrate 1500000 --enable_lpm
```

```
--enable_hci /dev/ttyS0 -d &
```

RK29:

```
 brcm_patchram_plus --patchram bychip --baudrate 1500000 --enable_lpm
--enable_hci /dev/ttyS2 -d &
```

如果蓝牙硬件及驱动没有问题，那么在这步执行完成后可以看到打印：

```
E/bluetooth_brcm(402): Done setting line discipline
```

如果没有出现这行打印，说明蓝牙硬件或软件方面存在问题。

Realtek RTL8723AS 使用命令：

```
/system/bin/hciattach -n -s 115200 /dev/ttyS0 rtk_h5
```

[UART 状态] 在这一过程中，HOST 开始与蓝牙通过 UART 通信，HOST\_UART\_RTS 应当为低电平，且 HOST\_UART\_TX/RX 都应当为高电平，用示波器应能测量到通信的方波。

5、确认 hci0 interface 已经创建：

```
hciconfig -a
```

```
hci0: Type: BR/EDR Bus: UART
 BD Address: 20:00:00:00:01:09 ACL MTU: 1021:7 SCO MTU: 64:1
 DOWN
 RX bytes:485 acl:0 sco:0 events:18 errors:0
 TX bytes:95 acl:0 sco:0 commands:18 errors:0
 Features: 0xff 0xff 0x8f 0xfe 0x9b 0xff 0x79 0x87
 Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
 Link policy: RSWITCH HOLD SNIFF PARK
 Link mode: SLAVE ACCEPT
```

6、激活蓝牙

```
hciconfig hci0 up
```

7、确认蓝牙激活成功

```
hcitool dev
```

Devices:

```
 hci0 20:00:00:00:01:09
```

8、 蓝牙激活成功后，可以开始扫描周围的蓝牙设备

```
hcitool scan
```

Scanning ...

```
 90:C1:15:0F:C2:78 Xperia neo
```

### 4.7.3 蓝牙打开失败

可从如下步骤去查：

#### A. 软件配置是否正确？

先确定是否按第 3 节的 android, kernel 具体配置。

#### B. 从命令行启动蓝牙，观察出错的位置

```
E/bluetooth_brcm(1615): Cannot open "/data/misc/bluetoothd/bt_addr": No such file or directory
```

```
D/bluetooth_brcm(1615): bd_addr: 20:00:00:00:01:0B
```

```
D/bluetooth_brcm(1615): Read default bdaddr of 20:00:00:00:01:0B
```

```
D/bluetooth_brcm(1615): Get hcd file by BT chip!
```

```
D/bluetooth_brcm(1615): hcd file: /system/etc/bluez/rk903.hcd
```

```
D/bluetooth_brcm(1615): /dev/ttyS0
```

```
--- a 发送 reset 指令 ---
```

```
E/bluetooth_brcm(1615): writing
```

```
E/bluetooth_brcm(1615): 01 03 0c 00
```

```
E/bluetooth_brcm(1615): received 7
```

```
E/bluetooth_brcm(1615): 04 0e 04 01 03 0c 00
```

```
E/bluetooth_brcm(1615): writing
```

```
E/bluetooth_brcm(1615): 01 2e fc 00
```

```
E/bluetooth_brcm(1615): received 7
```

```
E/bluetooth_brcm(1615): 04 0e 04 01 2e fc 00
```

```
--- b 读取蓝牙固件的内容并发送给蓝牙芯片 ---
```

```
.....
```

```
--- c 发送 reset 指令 ---
```

```
E/bluetooth_brcm(2072): writing
```

```
E/bluetooth_brcm(2072): 01 03 0c 00
```

```
E/bluetooth_brcm(2072): received 7
```

```
E/bluetooth_brcm(2072): 04 0e 04 01 03 0c 00
```

```
--- d 复位完成后，设置波特率 ---
```

```
E/bluetooth_brcm(2072): writing
```

```
E/bluetooth_brcm(2072): 01 18 fc 06 00 00 60 e3 16 00
```

```
E/bluetooth_brcm(2072): received 7
```

```
E/bluetooth_brcm(2072): 04 0e 04 01 18 fc 00
```

```
E/bluetooth_brcm(2072): Done setting baudrate
```

```
.....
```

```
E/bluetooth_brcm(1615): writing
```

```
E/bluetooth_brcm(1615): 01 27 fc 0c 01 01 01 01 01 01 00 00 00 00 00
```

```
E/bluetooth_brcm(1615): received 7
```

```
E/bluetooth_brcm(1615): 04 0e 04 01 27 fc 00
```

E/bluetooth\_brcm( 1615): Done setting line discipline

D/bluetooth\_brcm( 1615): total used:3.605s

在 a 出错：意味着与蓝牙芯片通过 UART 通信失败，可能问题：

蓝牙实际没有上电，IO 配置错误

UART 接线有问题，UART 的状态不对，可测量 UART 的 4 个脚

在 b 出错：说明蓝牙芯片上电成功了，但 UART 通信存在问题，应检查 UART

在 c 出错：在 RK29+BCM4329 中，由于 UART 的 Rx/Tx 没有加上拉，要求在执行 reset 之前要先读取一个 byte:

```
read(uart_fd, buffer, 1);
```

在 d 出错：可能是 BT\_WAKE 这个 IO 的被设置成了 disable，导致蓝牙挂起，无法通讯；可测量该 IO 电平。

C. 如果查不出问题，此时应检查晶振的频偏及振幅是否满足要求

#### 4.7.4 蓝牙 UART 通信失败

调试蓝牙时经常遇到 UART 通信失败，也就是主控发第一条 UART 命令给蓝牙芯片的操作失败。这可以通过以下步骤排查。

在 Android 4.2 平台中，开机时上层默认会去初始化 BT 芯片，就是通过 UART 发一系列命令给 BT 芯片。也可以通过 kill com.android.bluetooth 进程，让上层不断去初始化 BT 芯片。

A. 用示波器测量主控的 UART\_TX 信号，确认是否有发出第一条命令，如果没有发出，那么可能有以下原因：

- 1) UART\_TX 的 IO 脚可能被复用成其它 IO 功能了；
- 2) 流控信号 UART\_CTS 的状态不对，这是 BT 模块输出，高电平表示，BT 模块未准备好，不能发 UART 数据给它；低电平表示，BT 模块已经准备好，可以发数据给它。
  - i) BT 模块的 power, reset 未控制，没有给它上电或复位；
  - ii) BT 模块的 32.768KHz sleep clock 没有供给它，或者幅度过大，过小；
- 3) 软件没有走到蓝牙初始化代码；
- 4) 硬件连接问题；

B. 主控的 UART\_TX 信号有发出，但是 BT 模块未响应，这种情况很可能是 BT 模块没有工作起来：

- 1) BT 模块的 power, reset 未控制，没有给它上电或复位
- 2) BT 模块的 32.768KHz sleep clock 没有供给它，或者幅度过大，过小；
- 3) 流控信号 UART\_RTS 的状态不对，这是主控的输出，高电平表示，主控未准备好，不能发 UART 数据给它；低电平表示，主控已经准备好，可以发数据给它。

可能原因：UART\_RTS 的 IO 脚可能被复用成其它 IO 功能了。

C. 前面的 UART 命令都操作成功，但是切换 UART 波特率后出现异常

- 1) 目前大部分 BT 正常工作需要从初始化时的 115200bps 切换到 1500000bps，出现切换后通

信异常，可能是 uart clock 设置不合理，导致 uart 控制器分不出精确的波特率（误差在 3%以内），最后造成通信失败。

可通过 cat proc/clocks 查看当前 uart clock，可按以下方法设置合理的 uart clock：

设置原则是：UART 的 clock 等于串口波特率乘以 16 的倍数，例如，

波特率为 1.5M，那么 UART0clock 需要设置成  $1.5M * 16 * n = 24 M (n=1, 2, 3, \dots)$

```
@@ -2231,7 +2231,7 @@ static void __init machine_rk30_board_init(void)
 rk29sdk_wifi_bt_gpio_control_init();

#endif

#ifdef CONFIG_MT5931_MT6622
- clk_set_rate(clk_get_sys("rk_serial.0", "uart"), 16*1000000);
+ clk_set_rate(clk_get_sys("rk_serial.0", "uart"), 24*1000000);
#endif
}
```

#### 4.7.5 蓝牙设备配对失败

确认 32.768k rtc clock 的精度与幅度

晶振：

32.768KHz 晶振，我们的推荐是 从 PMU 引出，RK29 中没有使用 PMU，是从 RTC 取 clock datasheet 中推荐

Parameter	LPO Clock	Units
Nominal input frequency	32.768	kHz
Frequency accuracy	± 200	ppm
Duty cycle	30 - 70	%
Input signal amplitude	200 to 1800	mV, p-p
Signal type	Square-wave or sine-wave	-
Input impedance	>100k <5	Ω pF
Clock jitter (integrated over 300Hz – 15KHz)	<1	Hz

其峰峰值： 0.2 ~ 1.8 V

推荐频偏 200ppm， 即 32762 ~ 32774， 1ppm=1/10<sup>6</sup>

如果是 RK903 模块，要求 26MHz 晶振，其频偏不能偏超过 520Hz

#### 4.7.6 文件传输失败

确认文件的类型，是否 Android 支持的类型：

如果需要增加接受文件类型的支持，需要在 packages/apps/Bluetooth/src/com/android

/bluetooth/opp/Constants.java 的 ACCEPTABLE\_SHARE\_INBOUND\_TYPES 数组里增加相应

的 data style;

```
public static final String[] ACCEPTABLE_SHARE_INBOUND_TYPES = new String[] {
 "image/*",
 "video/*",
 "audio/*",
 "text/x-vcard",
 "text/plain",
 "text/html",
 "application/zip",
 "application/vnd.ms-excel",
 "application/msword",
 "application/vnd.ms-powerpoint",
 "application/pdf",
};
```

如果需要增加发送文件类型的支持，需要在 **AndroidManifest.xml** 里增加相应的 **style**:

```
<action android:name="android.intent.action.SEND" />
<category android:name="android.intent.category.DEFAULT" />
<data android:mimeType="image/*" />
<data android:mimeType="video/*" />
<data android:mimeType="audio/*" />
<data android:mimeType="text/x-vcard" />
<data android:mimeType="text/plain" />
<data android:mimeType="text/html" />
<data android:mimeType="application/zip" />
<data android:mimeType="application/vnd.ms-excel" />
<data android:mimeType="application/msword" />
<data android:mimeType="application/vnd.ms-powerpoint" />
<data android:mimeType="application/pdf" />
</intent-filter>
```

#### 4.7.7 蓝牙默认设备名字修改

##### 1. Android 4.2

针对不同的模块具体修改如下:

MT6622:

device/common/libbt\_mtk6622/bdroid\_buildcfg.h

RK903, AP6xxx, RTL8723:

device/rockchip/rk30sdk/bluetooth/bdroid\_buildcfg.h

RDA 587x:

device/common/libbt\_rda/bdroid\_buildcfg.h

默认定义为"rk30sdk"

#define BTM\_DEF\_LOCAL\_NAME "rk30sdk"

修改后重新编译以下目录:

mmm external/bluetooth/bluedroid/ -B

#### 4.7.8 蓝牙键盘无法使用

请打开 kernel 配置中的以下配置:

CONFIG\_UHID=y



## 5 WiFi 主要指标说明

### 5.1 WiFi 吞吐率以及吞吐率的测量

吞吐率指单位时间内在网络上传输的数据量。是衡量网络性能的主要指标。通常情况下测量 WiFi 的吞吐率主要使用 iperf 工具，理想环境要求在屏蔽房中进行，避免干扰。实际测量中通常直接在办公环境下测试，所得数据和理想环境下测试结果会有较大的差距。

在理想环境下，iperf 测试的实际吞吐率可以达到 WiFi 实际连接速率的 1/2，例如连接速率是 65Mbps，实际吞吐率可以达到 30Mbps 左右。

只支持 HT20(20M 带宽)的模块，连接速率只能达到 72Mbps，例如 RK903 等，在办公环境下近距离测试大概在 10~20Mbps 左右。

支持 HT40(40M 带宽)的模块，连接速率最高可到 150Mbps，例如 RTL8188，在办公环境下近距离测试大概是在 30~40Mbps 左右。

这些数据的前提是 WiFi 的天线匹配较好，排除本身设备的硬件问题。

### 5.2 RF 硬件指标

#### 5.2.1 频偏 (Frequency Error)

频率误差表征射频信号偏离该信号所处信道中心频率的大小，通常以 IQview 来测量，单位为 ppm。

频偏要求：<±25ppm (802.11b)； <±20ppm (802.11g/a/n)

#### 5.2.2 矢量误差幅度 (EVM)

EVM 表征的是一个给定时刻理想无误差基准信号和实际发射信号的向量差。

在 AP 的无线指标中，EVM 是一个发送状态时一个非常重要的指标，它是表征信号发送质量的好坏的一个指标。

EVM 与发射功率相关联：发射功率越大，矢量误差被放大的就越多，也就是 EVM 越大即发送信号质量越差。

在实际应用中，我们要在发送功率和 EVM 间去一个折中，这就是在测试 g mode 和 n mode 时，发送信号功率不能太大的原因。

EVM 要求：< -25dbm (802.11g)； < -28dbm (802.11n)

#### 5.2.3 发射功率 (Transmitter power)

发射功率在保持 EVM 性能的前提下，功率越大，WiFi 性能越好，在实际应用中表现为无线

覆盖范围越大。

Rx Power 要求: [13.5, 16.5] (802.11g); [12.5, 15.5] (802.11n)

## 5.2.4 接收灵敏度 (Receiver Sensitivity)

接收灵敏度越好, 其接收到的有用信号就越多, 其无线覆盖范围越大。

接收灵敏度要求: -69dbm (802.11n MSC7); -72dbm (802.11g 54Mbps)

## 5.3 WiFi RFTTest (定频测试)

WiFi 定频测试就是让 WiFi 模块发出特定的信号(11b, 11g, 11n, channel 1-13, 各种 WiFi 速率 [11Mbps, 54Mbps, 65Mbps]), 以便于测试仪器进行 RF 指标测试。

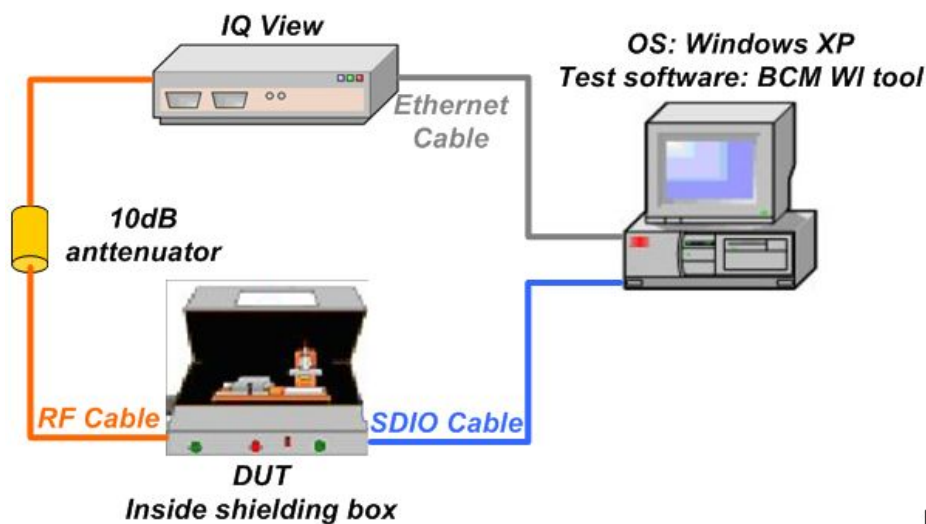


图 5-1

IQ View 为测试仪器:

DUT 为待测试的平板电脑, 需要放在屏蔽箱内, 与 IQ View 通过 RF Cable 连接。

WiFi 定频一般需要特定的驱动程序, 这个驱动程序与正常使用的 WiFi 驱动程序不兼容。

目前定频工具有两类:

一是做成 UI 形式的, 例如 RK903

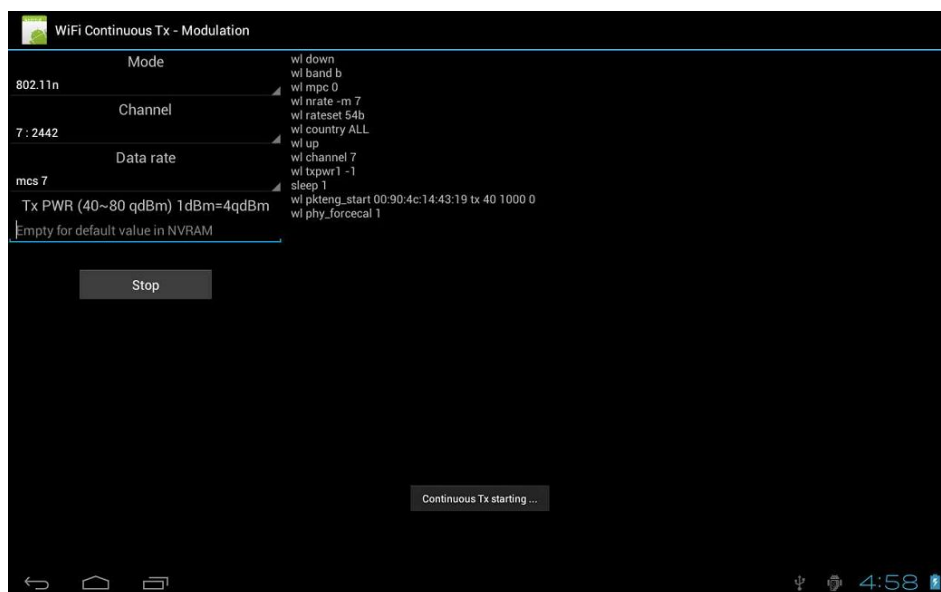


图 5-2

二是命令形式的，例如 **mtk WiFi**

通过 adb 执行特定的命令，让 WiFi 模块进入定频发射。