密级状态：绝密（　　）　秘密（　　）　内部（ √ ）　公开（　）

RK Kernel 3.10 平台 WiFi BT 不工作异常排查

（技术部，系统一部）

文件状态： [　] 正在修改 [√] 正式发布	当前版本：	V1.0
	作　　者：	胡卫国
	完成日期：	2015-02-09
	审　　核：	
	完成日期：	

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	胡卫国	2015-02-09	初始版本	

目 录

1 WiFi 打不开问题

表现出的异常现象为：设置中打开 WiFi，一直显示正在打开，或者打开后自动关闭。

需要通过以下几点排查：

1. 确认 WLAN_RFKILL 驱动是否加载成功

加载成功会有如下打印：

[WLAN_RFKILL]: Exit rfkill_wlan_probe

如果失败，一般都是 dts 配置异常导致，可跟踪代码排查，驱动代码在：kernel/net/rfkill/rfkill-wlan.c

2. 确认上层是否调用到了 kernel 中的 WiFi 驱动

上层调用到 WiFi 驱动会有如下打印（以 AP6335 为例）：

==========

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

==========

WiFi driver (Powered by Rockchip,Ver 5.00.WFD.OOB) init.

如果没有这个打印，那可能是上层配置不对，需要参考《RKXX_Android 4.4&5.0 Kernel 3.10 WiFi BT 配置说明 V1.0.pdf》进行配置

3. sdio 识卡，或 usb 枚举设备是否正常

1) **如果是 sdio 接口 wifi，正常识卡会有如下打印：**

mmc2: new high speed SDIO card at address 0001

如果识别异常，需要从以下几点排查：

A) **确定 WiFi 模块是否正常工作**，比如 VCCIO_WL 供电是正常正常，由 GPIO 控制的模块使能脚是否被正常控制到

B) **如果是 AP6xxx 系列模块**，还需要确认外部晶体是否正常工作，外部供的 32.768K clock 是否正常，32.768K clock 指标满足要求（参考模块 spec）

C) 如果是 RK312x 平台，SDIO 接口需要接外部上拉电阻

并将默认芯片的上下拉禁掉

```
+++ b/arch/arm/boot/dts/rk312x-pinctrl.dtsi
                        sdio0_cmd: sdio0_cmd {
                                rockchip,pins = <MMC1_CMD>;
-                               rockchip,pull = <VALUE_PULL_DISABLE>;
+                               rockchip,pull = <VALUE_PULL_UPDOWN_DISABLE>;
                        };
                        sdio0_clk: sdio0_clk {
                                rockchip,pins = <SDMMC_CLKOUT>;
-                               rockchip,pull = <VALUE_PULL_DISABLE>;
+                               rockchip,pull = <VALUE_PULL_UPDOWN_DISABLE>;
                        };
                        sdio0_bus1: sdio0-bus-width1 {
                                rockchip,pins = <SDMMC_DATA0>;
-                               rockchip,pull = <VALUE_PULL_DISABLE>;
+                               rockchip,pull = <VALUE_PULL_UPDOWN_DISABLE>;
                        };
                        sdio0_bus4: sdio0-bus-width4 {
                                rockchip,pins = <SDMMC_DATA0>,
                                                <SDMMC_DATA1>,
                                                <SDMMC_DATA2>,
                                                <SDMMC_DATA3>;
-                               rockchip,pull = <VALUE_PULL_DISABLE>;
+                               rockchip,pull = <VALUE_PULL_UPDOWN_DISABLE>;
                        };

                        sdio0_gpio: sdio0_gpio{
                                                <GPIO1_A2>, //data1
                                                <GPIO1_A4>, //data2
                                                <GPIO1_A5>; //data3
-                               rockchip,pull = <VALUE_PULL_DISABLE>;
+                               rockchip,pull = <VALUE_PULL_UPDOWN_DISABLE>;
                        };

                };
```

D) **如果是 RK3288 平台，主控的 SDIO 电平有 1.8、3.3V 可配置**，需要根据实际硬件(VCCIO_WL 电压）来配置 dts 中的：sdio_vref = <1800>;

E) 如果使用了 sdmmc 作为 wifi sdio 口

需要修改如下（以 rk312x 平台为例)：

将 rk312x-sdk.dtsi 中的 sdmmc 修改成如下定义

```
&sdmmc {
        clock-frequency = <37500000>;
        clock-freq-min-max = <200000 37500000>;
        supports-highspeed;
        supports-sdio;
        ignore-pm-notify;
        keep-power-in-suspend;
        cap-sdio-irq;
};
```

在板级 dts 中增加

```
&sdmmc {
        status = "okay";
};
```

增加以下代码，关掉 jtag 自动 iomux 成 jtag 功能（注意，此修改只针对 RK312x 平台）

```
+++ b/drivers/mmc/host/rk_sdmmc.c
@@ -3845,6 +3845,9 @@ int dw_mci_probe(struct dw_mci *host)
        int init_slots = 0;
        u32 regs;

+         if(cpu_is_rk312x())
+             grf_writel(((1 << 8) << 16) | (0 << 8), RK3036_GRF_SOC_CON0);
+
        if (!host->pdata) {
                host->pdata = dw_mci_parse_dt(host);
                if (IS_ERR(host->pdata)) {
```

E) **可尝试降低 SDIO Clock 试试：**

在 dts 中增加以下部分

```
&sdio{
    clock-frequency = <37500000>; //  修改成 12M 或更低
}
```

F) sdio iomux 是否正常：

RK3288 WiFi 默认使用 SDIO0（需要跟实际硬件相匹配）：

SDIO0 （标记为红色的数字表示 io 切换成了 SDIO0 口）

```
io -4 0xff770044
```

ff770044: 00005555 // 第 8-15 bit

io -4 0xff770048

ff770048: 00000005 // 第 0-3 bit

RK312x WiFi 默认使用 SDIO1（需要跟实际硬件相匹配）:

SDIO0 （标记为红色的数字表示 io 切换成了 SDIO0 口）

io -4 0x200080bc

200080bc: 00004028 // 第 14bit

io -4 0x200080c0

200080c0: 00005555 // 第 0bit, 4-11bit

SDIO1 （标记为红色的数字表示 io 切换成了 SDIO1 口）

io -4 0x200080b8

00000a2a // 第 0-5 , 8-11 bit

io -4 0x200080a8

0000a085 // 第 6-7 bit

2) 如果 sdio 识别正常，有如下打印：

rksdmmc: data FIFO error

那可能是 sdio data 线有异常，需要按上一节中的 C 至 F 部分排查

3) **如果是 USB WiFi，正常枚举到 USB 设备会有如下类似打印**：以 RTL8723BU 为例

[5.339323] usb 2-1: New USB device found, idVendor=0bda, idProduct=b720

[5.339342] usb 2-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3

[5.339358] usb 2-1: Product: 802.11n WLAN Adapte

[5.339371] usb 2-1: Manufacturer: Realtek

如果识别异常，需要从以下几点排查：

A) **确认 USB WiFi 电源是否正常供着**（如果有 GPIO 使能脚，控制是否正常？）

B) **确认 USB Host 是否正常**，可以接上 U 盘等设备测试下

4. 确认网络接口 wlan0 是否正常注册

使用 netcfg 命令查看 wlan0 接口是否正常注册，（如下正常的打印）

```
wlan0    UP          192.168.8.102/24   0x00001043 6c:fa:a7:21:91:e0
eth0     UP          0.0.0.0/0          0x00001003 fe:53:d4:58:75:9c
sit0     DOWN        0.0.0.0/0          0x00000080 00:00:00:00:00:00
lo       UP          127.0.0.1/8        0x00000049 00:00:00:00:00:00
```

如果 wlan0 正常注册，可以使用 iwlist 命令查看是否可以正常扫描到路由器(AP)

```
iwlist wlan0 scan
wlan0     Scan completed :
          Cell 01 - Address: F4:EC:38:61:05:CA
                    ESSID:"TP-LINK_8E"
```

Cell 02 - Address: C4:A8:1D:84:D9:F4
ESSID:"D-Link_DIR-868L"

Cell 03 - Address: 08:57:00:F9:D9:C8
ESSID:"TP-LINK_HWG"

Cell 04 - Address: C0:3F:0E:08:10:BA
ESSID:"FZ_NETGEAR"

Cell 05 - Address: D4:EE:07:03:BC:14
ESSID:"HiWiFi_03BC14"

Cell 06 - Address: C0:A0:BB:1D:F1:70
ESSID:"mike-dlink"

如果使用 iwlist 命令可以正常扫描到 AP，那可能是上层配置不对，需要参考《RKXX_Android 4.4&5.0 Kernel 3.10 WiFi BT 配置说明 V1.0.pdf》进行配置

5. 如果上面步骤排查都正常，仍然有问题，需要提供以下 log 供我们分析：

logcat -v time, cat proc/kmsg （注意这些 log 需要从开机开始，不要有丢失）

1.1 机器休眠唤醒后 SDIO WiFi 异常

具体表现为以下两种情况：

1) 打开 wifi 连接路由器，让机器进入二级休眠，再唤醒后，wifi 出现异常；

2) 休眠前先关闭 wifi，让机器进入二级休眠，唤醒后，wifi 打不开

具体原因是休眠唤醒后，sdmmc1 脚被切换成了 GPIO

```
--- a/arch/arm/boot/dts/rk3126-sdk.dts
+++ b/arch/arm/boot/dts/rk3126-sdk.dts
@@ -55,9 +55,9 @@
                        //|RKPM_CTR_VOL_PWM2
                        )
                >;
-               rockchip,pmic-suspend_gpios = <
-                       GPIO1_A1
-                       >;
+               //rockchip,pmic-suspend_gpios = <
```

```
+                //        GPIO1_A1
+                //         >;
         };
 };

diff --git a/arch/arm/mach-rockchip/pm-rk312x.c b/arch/arm/mach-rockchip
/pm-rk312x.c
index 475f6d0..9e95d95 100644
--- a/arch/arm/mach-rockchip/pm-rk312x.c
+++ b/arch/arm/mach-rockchip/pm-rk312x.c
@@ -829,7 +829,7 @@ static inline void  rkpm_slp_mode_set_resume(void)

        grf_writel(grf_soc_con0 | (1 << (SOC_REMAP + 16)), GRF_SOC_CON0);

-       if ((pmic_sleep_gpio == 0) || (pmic_sleep_gpio == 0x1a10))
+       if (/*(pmic_sleep_gpio == 0) || */(pmic_sleep_gpio == 0x1a10))
                grf_writel(0X000C0000 | gpio_pmic_sleep_mode, 0xb8);
        /*rk3126 GPIO1A1 : RK3128 GPIO3C1 iomux pmic-sleep*/
        if (pmic_sleep_gpio == 0x3c10)
```

1.2 reboot 重启后或者 WiFi 反复开关后出现 wifi 概率性打不开

机器 reboot 重启后 wifi 出现概率性打不开，或者反复开关 wifi 出现概率性打不开。

特别是 rtl8703as (rtl8723bs-vq0)模块，特别容易出这个问题。

需要确认 wifi 模块的电源控制，一般 wifi 模块有一路 gpio 控制的电源。有些客户可能会做成常供电，这可能会造成问题。

1. 需要在开机时确保这个电源复位一次（也就是先断电再上电）。例如 AP6xxx 系列的模块，在开机时需要先将 WL_REG_ON 脚拉底，再拉高。客户遇到问题时，需要先确认开机时针对这个脚是否已经有上面的逻辑控制（实际测量为准）。

2. 另外在 WiFi 关闭时，需要将 WL_REG_ON 脚拉底，打开时再拉高。

以上控制逻辑在 rfkill-wlan.c 中已经实现，只要正确配置 dts 就可。

值得注意的是：后期的 RK3368 Android 5.1 将 wifi 驱动修改成开机就初始化，这样子在后面开关 WiFi 时，其 WL_REG_ON 脚都是不会被关闭的（AP6xxx 模块除外）。

1.3 AP6330 扫描不到 5G 的路由器

需要打开以下开关，不然 AP6330 只工作在 2.4G 模式：

drivers/net/wireless/rockchip_wlan/rkwifi/bcmdhd/Makefile

DHDCFLAGS += -DBAND_AG

2 BT 打不开问题

首先确认 BOARD_HAVE_BLUETOOTH 设置成 true

例如 312x 平台在以下目录设置：

device/rockchip/rk312x/BoardConfig.mk

不然设置中都看不到蓝牙开关的选项。

表现出的异常现象为：设置中打开 BT，一直显示正在打开，或者打开后自动关闭。

需要通过以下几点排查：

1. 确认 BT_RFKILL 驱动是否加载成功

加载成功会有如下打印(以 AP6335 为例)：

[BT_RFKILL]: ap6335 device registered.

如果失败，一般都是 dts 配置异常导致，可跟踪代码排查，驱动代码在：kernel/net/rfkill/rfkill-bt.c

2. 确认 UART 设备是否打开正常

如果打开正常会有如下 logcat 打印：

userial vendor open: opening /dev/ttyS0

注意：RTL8723AU/BU 使用的是 USB 接口，所以不使用这个接口，使用：/dev/rtk_btusb

如果打开失败，需要确认：

1) 是否存在这个节点

2) 是否有权限访问，一般在 init.connectivity.rc 中修改相应的属性(类似如下)

chmod 0660 /dev/ttyS0

chown bluetooth net_bt_stack /dev/ttyS0

3. 确认 bt stack bluedroid 初始化是否成功

Bluedroid 初始化成功依赖于 bt 模块正常工作，如果操作成功，会有如下打印：

02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_HCI

02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_L2CAP

02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_RFCOMM
02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_AVDT
02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_AVRC
02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_A2D
02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_BNEP
02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_BTM
02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_GAP
02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_PAN
02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_SDP
02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_GATT
02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_SMP
02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_BTAPP
02-10 01:15:56.438 I/ (1257): BTE_InitTraceLevels -- TRC_BTIF

如果没有操作成功需要从以下几点排查：

A) **确定 BT 模块是否正常工作**，比如 VCCIO_WL 供电是正常正常，由 GPIO 控制的模块使能脚是否被正常控制到

B) **如果是 AP6xxx 系列模块**，还需要确认外部晶体是否正常工作，外部供的 32.768K clock 是否正常，32.768K clock 的精度及幅度是否满足要求（参考模块 spec）

C) **确认 UART 工作是否有异常**

比如有如下打印表示 UART 接收到的数据校验出错，

serial 20060000.serial: error:lsr=0xd9

这一般是 UART TX, RX 数据线没有维持在高电平导致异常，可能跟芯片内部上下拉有关系，如果是 Rk312x 平台，可以通过以下修改禁掉 UART0 口内部的下拉，类似如下：

```
uart0_xfer: uart0-xfer {
    rockchip,pins = <UART0_SIN>,
            <UART0_SOUT>;
    rockchip,pull = <VALUE_PULL_UPDOWN_DISABLE>;
};
```

D) **如果是 RK3128 box 平台，BT 默认使用的是 UART1 口**，但是代码里配置的是 UART0，需要修改代码：

1.

device/common/bluetooth/libbt/ // AP6xxxBT 模块

device/common/bluetooth/libbt_rtk8723bs/ // Realtek BT 模块

找到所有文件中的 ttyS0（注意不一定是.c,.h 文件），将其修改成 ttyS1

2. 针对 Android 4.4 还需要修改：

device/rockchip/rksdk/

```
diff --git a/init.connectivity.rc b/init.connectivity.rc
index 6afc7b8..e82db56 100755
--- a/init.connectivity.rc
+++ b/init.connectivity.rc
@@ -44,10 +56,10 @@ on boot

 # bluetooth power up/down interface
     chmod 0660 /dev/ttyS0
-    chmod 0660 /dev/ttyS2
+    chmod 0660 /dev/ttyS1
     chmod 0660 /dev/vflash
     chown bluetooth net_bt_stack /dev/vflash
-    chown bluetooth net_bt_stack /dev/ttyS2
+    chown bluetooth net_bt_stack /dev/ttyS1
     chown bluetooth net_bt_stack /dev/ttyS0
     chown bluetooth net_bt_stack /sys/class/rfkill/rfkill0/type
     chown bluetooth net_bt_stack /sys/class/rfkill/rfkill0/state
```

hardware/realtek/wlan

```
diff --git a/wlan/config/init.realtek.rc b/wlan/config/init.realtek.rc
index 90d226c..3c3ddbb 100755
--- a/wlan/config/init.realtek.rc
+++ b/wlan/config/init.realtek.rc
@@ -23,10 +23,10 @@ on boot

 # bluetooth power up/down interface
     chmod 0660 /dev/ttyS0
-    chmod 0660 /dev/ttyS2
```

+ chmod 0660 /dev/ttyS1

chmod 0660 /dev/vflash

chown bluetooth net_bt_stack /dev/vflash

- chown bluetooth net_bt_stack /dev/ttyS2

+ chown bluetooth net_bt_stack /dev/ttyS1

chown bluetooth net_bt_stack /dev/ttyS0

chown bluetooth net_bt_stack /sys/class/rfkill/rfkill0/type

chown bluetooth net_bt_stack /sys/class/rfkill/rfkill0/state

E) 如果是 RTL8723BU/AU USB 接口模块

先确认 CONFIG_BT_RTKBTUSB=y

另外确认 USB 设备是否被正常枚举到了

H) uart iomux 是否正常：

RK3288:

如果使用 UART0 （标记为红色的数字表示 io 切换成了 UART0 口）

io -4 0xff770044

ff770044: 00005555 // 低 8bit

RK312x :

如果使用 UART0 （标记为红色的数字表示 io 切换成了 UART0 口）

io -4 0x200080d4

200080d4: 000008a0 // 第 4-11 bit

io -4 0x200080b0

200080b0: 00000108 // 第 2,3 bit

如果使用 UART1 （标记为红色的数字表示 io 切换成了 UART1 口）

io -4 0x200080bc

200080d4: 000040aa // 第 0-7 bit

4. 如果 1, 2 两点都没问题（只针对 RTL8723BS 系列芯片）

确认下是否有如下异常打印：

I/ActivityManager(460): Process com.android.bluetooth (pid 1518) has died.

那么需要确认主控 UART_CTS 脚是否在悬空状态？如果是，需要将其接地(建议串 10K 电阻接地)，或者接到 rtl8723bs 对应脚上。

4. 如果上面步骤排查都正常，仍然有问题，需要提供以下 log 供我们分析：

logcat -v time, cat proc/kmsg （注意这些 log 需要从开机开始，不要有丢失）

5. bt snoop log 记录方法

除了上面的 logcat 外，还可以通过以下方法记录 bt stack 部分的通信协议数据，以便进一步分析

目前 Android 平台上的 bt snoop 记录方法：

修改

system/etc/bluetooth/bt_stack.conf

里的

BtSnoopLogOutput=false

为

BtSnoopLogOutput=true

重启机器，BT 操作时的所有 bt stack 通信数据会记录到以下文件：

/sdcard/btsnoop_hci.log

bt_snoop_hci.log 是二进制内容，需要修改成 pcap 后缀，可使用 wireshark 打开查看。

2.1 机器休眠后 BT 无法接收文件或 BT 键盘等 HID 设备无法唤醒机器

需要确认 dts 中以下中断配置是否正常，GPIO，中断有效电平，建议实际量一下。

BT,wake_host_irq = <&gpio4 GPIO_D7 GPIO_ACTIVE_HIGH>;

3 WiFi BT 简单测试

3.1 WiFi 部分

1. WiFi 是否可以正常打开，连接上路由器并成功上网
2. WiFi Direct 功能是否正常
3. WiFi 热点功能是否正常
4. 打开 WiFi 重启机器，WiFi 是否自动成功打开
5. 打开 WiFi 连接路由器，休眠唤醒后，WiFi 是否正常，是否可以正常上网
6. 恢复出厂设置后，WiFi 是否可以正常打开
7. 重启机器后，WiFi MAC 地址是否变掉了
8. 两台同样的机型 WiFi MAC 地址是否不一样

3.2 BT 部分

1. BT 是否可以传送，接收文件
2. BT 是否可以连接上蓝牙音箱或耳机，并出声音
3. BT 是否可以连接蓝牙键盘鼠标
4. 两台同样的机型 BT MAC 地址是否不一样
5. 恢复出厂设置后，BT 是否可以正常打开