

SWT6XXX 无线芯片 BSP KERNEL 移植指导

SWT6XXX 无线芯片 BSP KERNEL 移植指导.....	1
1. 修订记录	2
2. Makefile 环境变量配置	3
2.1. KSRC_PATH 指定 kernel 源码路径。	3
3. 驱动编译	3
3.1. driver 目录结构	3
3.2. 目录结构说明	4
3.3. Drivers 目录	4
3.4. Firmware 目录	5
3.5. DTS 添加说明	5
4. LOG 的输出	8
4.1. LOG 目录	8
4.2. RELEASE 版本如何关闭 LOG	9
4.3. RELEASE 版本如何开启 LOG	10
4.4. 客户问题复现抓取 LOG 的方法	10
5. 常见问题分析 FAQ:	14
5.1. 编译问题	14
5.2. SDIO 扫卡失败	15
5.3. Load firmware 失败	15
5.4. Boot 完成后没有收到版本信息	18
5.5. USB 枚举失败	19
5.6. USB 枚举成功, 但是 USB boot 报错	19
5.7. USB boot 完成后, 没有回复版本信息	20
5.8. 关闭 recoverymode 复现问题	21
6. AP 平台的 check list:	22
6.1. Rockchip 相关的平台的 checklist	22
6.2. Allwinner 相关平台的 checklist	23

1. 修订记录

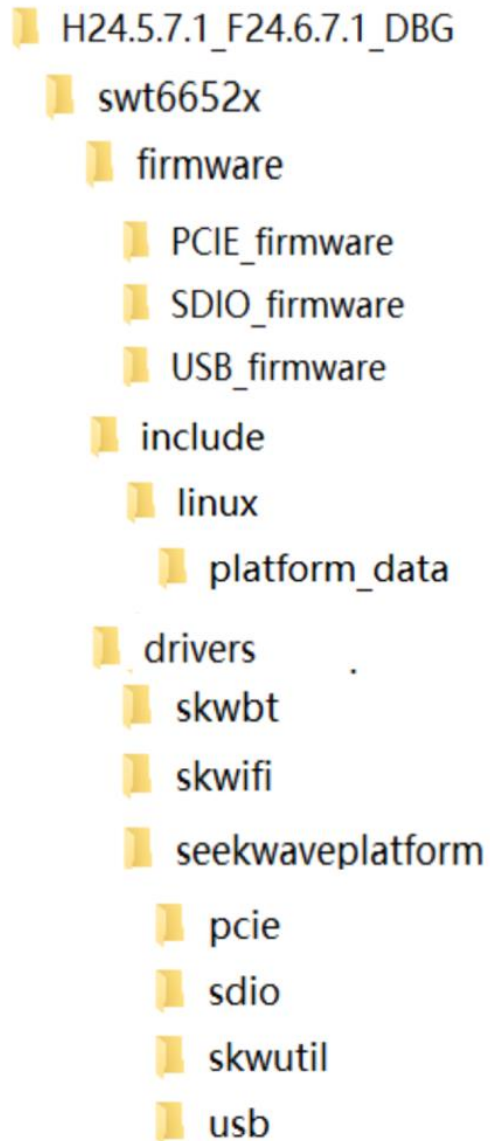
版本	修订日期	作者	描述
1.0.0	2023-6-5	JUNWEI.JIANG	SWT6652 芯片 BSP KERNEL 移植文档初稿
1.1.0	2023-6-5	Jiayong.yang	增加 sdio_pwrseq node 修改。
1.1.1	230907	JUNWEI.JIANG	增加关闭 recover 复现问题 debug 方法
1.1.1	240402	JUNWEI.JIANG	增加 firmware 的客户汇总
1.1.2	240515	JUNWEI.JIANG	增加 AP 主控的配置 checklist(初稿)
1.1.2	240531	JUNWEI.JIANG	更新客户常见问题分析,firmware load fail
1.1.2	240531	JUNWEI.JIANG	更新客户问题 log 复现抓取的方法
1.1.2	240531	JUNWEI.JIANG	更新客户问题手动 dump mem 的方法
1.1.3	240628	Cheng.Lu	更新 PCIe 部分

2. Makefile 环境变量配置

2.1. KSRC_PATH 指定 kernel 源码路径。

3. 驱动编译

3.1.driver 目录结构



3.2. 目录结构说明

根目录名 H24.5.7.1_F24.6.7.1，是 patch 的版本号，根目录包含三个子目录，分别为 drivers, firmware 和 kernel。

drivers 包含所有的驱动代码，

firmware 为固件文件，

kernel 目录包含所有对 kernel 原生代码的 patch，目前包含 DTS patch 和 sdio driver patch，这部分代码需要合入到 kernel 代码中。

3.3. Drivers 目录

Drivers 包含 3 个部分，

分别对应 WiFi Driver, Bus driver 和 boot driver 编译后生成 3 个 KO：

对于 SDIO 模式，分别生成

```
[seekwave]
WiFi driver: skw.ko;

BT driver: skwbt.ko

Boot & Bus driver: skw_sdio.ko.
```

对于 USB 模式，分别生成 WiFi driver: skw.ko; BT driver: skwbt.ko 以及 USB boot & bus driver: skw_usb.ko.

这 3 个 KO 的加载顺序为： bus & boot driver、wifi driver、bt driver

```
[seekwave]
WiFi driver: skw.ko;

BT driver: skwbt.ko

Boot & Bus driver: skw_usb.ko.
```

对于 PCIe 模式，编译后生成 3 个 KO

```
[seekwave]
WiFi driver: skw.ko;

BT driver: skwbt.ko

Boot & Bus driver: skw_pcie.ko.
```

3.4. Firmware 目录

包含三个子目录分别对应 SDIO 模式的 firmware 文件、USB 模式的 firmware 文件和 PCIE 模式的 firmware 文件。固件加载是 boot & bus driver 中通过 request_firmware 机制加载 firmware 文件，因此 firmware 保存。

```
/lib/firmware/
```

注：在使用 request_firmware()之前，要在 menuconfig 中勾选相应的配置：

```
Symbol: FW_LOADER [=y]
| Type : tristate
| Prompt: Userspace firmware loading support
| Location:
|     -> Device Drivers
| (1)   -> Generic Driver Options
|     Defined at drivers/base/Kconfig:77
```

或者直接在.config 中配置：

```
...
CONFIG_FW_LOADER=y
...
```

3.5. DTS 添加说明

1)SDIO mode:

如果 host 平台支持 DTS，需要将以下 DST 节点增加到 DTS 中。

```
seekwcn_boot:seekwcn_boot {
    compatible = "seekwave,sv6160";
    dma_type = <1>; /*1:ADMA,2:SDMA*/
    seekwave_nv_name = "SEEKWAVE_NV_SWT6652.bin";
    status = "okay";
    gpio_host_wake = <&gpio0 10 GPIO_ACTIVE_LOW>; /*GPIO0_B2 CP2AP*/
    gpio_chip_wake = <&gpio0 3 GPIO_ACTIVE_HIGH>; /*GPIO0_A3 AP2CP*/
    gpio_chip_en = <&gpio0 9 GPIO_ACTIVE_HIGH>; /*GPIO0_B1 CP POWERON*/
};
```

在 DST node 中 dma_type，描述 SDIO DMA 的属性，默认配置 1。seekwave_nv_name 是配置 bt 的天线 usb speed config 等，根据 host 平台的 SDIO feature 决定。通常不改动。

后三个属性是 GPIO 的编号，主要功能描述如下：

Gpio_host_wake, WiFi 用来唤醒 Host 的 GPIO，在 host 端是作为输入模式，这个管脚连接到模块的 GPIO0 (WL_WAKE_HOST pin34)，并且对应 host 端的 gpio 对应的 PIN 要有 EIC 功能。

Gpio_chip_wake, Host 用来唤醒 WiFi 的 GPIO，在 host 端是作为输出模式，这个管脚连接到模块的 GPIO1 (HOST_WAKE_WL pin13)。

Gpio_chip_en, Host 用来给 WiFi Chip 上电控制的，连接到模块的 CHIP_EN (PMU_enable)，在 host 端是作为输出模式，输出高时 WiFi chip power on，输出低时 WiFi chip power off。

当 kernel 不支持 DTS，可以修改 seekwave_ea6652X\seekwaveplatform\skwutil 的 Boot_config.h 中的宏来定义各个 GPIO 编号。

```
+++ b/boot_config.h
@@ -0,0 +1,3 @@
#define MODEM_ENABLE_GPIO      -1 //变更为自己的 HW 相对应的 GPIO NO
#define HOST_WAKEUP_GPIO_IN    -1 //默认为-1，可在定义后启用
#define MODEM_WAKEUP_GPIO_OUT -1
#define SEEKWAVE_NV_NAME       "SEEKWAVE_NV_SWT6652.bin" //NO dts NV config
//iram dram file for no dts seekwave firmware_request api
#define SKW_IRAM_FILE_PATH     "/data/ROM_EXEC_KERNEL_IRAM.bin"
#define SKW_DRAM_FILE_PATH     "/data/RAM_RW_KERNEL_DRAM.bin"
```

如果不要求 low power 模式，DTS 中 Gpio_host_wake 和 Gpio_chip_wake 可以删除。

除了 boot 的 DST 节点需要配置外，kernel 原生 sdio_pwrseq 的 DST node 需要修改，其中的 reset-gpio 需要配置和 chip_en 相同的 GPIO。

```
&sdio {
    max-frequency = <200000000>;
    no-sd;
    no-mmc;
    bus-width = <4>;
    disable-wp;
    cap-sd-highspeed;
    cap-sdio-irq;
    keep-power-in-suspend;
    mmc-pwrseq = <&sdio_pwrseq>;
    non-removable;
    pinctrl-names = "default";
    pinctrl-0 = <&sdio_pins>;
    sd-uhs-sdr104;
    post-power-on-delay-ms = <50>;
    status = "okay";
};"
```

```

sdio_pwrseq: sdio-pwrseq {
    compatible = "mmc-pwrseq-simple";
    // clocks = <&hym8563>;
    // clock-names = "ext_clock";
    pinctrl-names = "default";
    pinctrl-0 = <&wifi_enable_h>;
    /*
     * On the module itself this is one of these (depending
     * on the actual card populated):
     * - SDIO_RESET_L_WL_REG_ON
     * - PDN (power down when low)
     */
    post-power-on-delay-ms = <200>;
    reset-gpios = <&gpio0 RK_PC4 GPIO_ACTIVE_LOW>;
};

```

```

sdio-pwrseq {
    wifi_enable_h: wifi-enable-h {
        rockchip,pins = <0 RK_PC4 RK_FUNC_GPIO &pcfg_pull_up>;
    };
};

```

对于非 DTS 项目，可以修改 Boot_config.h 宏为-1。且可在 skw_boot.c 中，可注释掉引用 CONFIG_OF 宏。

```
#ifndef CONFIG_OF
static void seekwave_release(struct device *dev)
{
}
static struct platform_device seekwave_device = {
    .name = "sv6160",
    .dev = {
        .release = seekwave_release,
    }
};
#endif
static int seekwave_boot_init(void)
{
    btboot_pdev = NULL;
    skw_ucom_init();
#ifndef CONFIG_OF
    platform_device_register(&seekwave_device);
#endif
    return platform_driver_register(&seekwave_driver);
}

static void seekwave_boot_exit(void)
{
    skw_ucom_exit();
#ifndef CONFIG_OF
    platform_device_unregister(&seekwave_device);
#endif
    platform_driver_unregister(&seekwave_driver);
}

/*****
```

可以通过关闭 Selinux 的总开关，允许模组的 log 临时保存到 data 的目录下面。可以在 data 目录下生成 log000 和

4. LOG 的输出

4.1.LOG 目录

在 android 产品形态中，可以通过关闭 Selinux 的总开关，允许模组的 log 临时保存到 data 的目录下面。可以在 data 目录下生成 log000 和 log111。

```
修改尚未加入提交 (使用 "git add" 和/或 "git commit -a")
yk163@yk163:~/incar_work/chs/android12_zx/01/rockchip2_s/system$ git diff core/init/selinux.cpp
diff --git a/system/core/init/selinux.cpp b/system/core/init/selinux.cpp
index 29c0ff3baa..66f0b6fb6c 100644
--- a/system/core/init/selinux.cpp
+++ b/system/core/init/selinux.cpp
@@ -114,6 +114,7 @@ EnforcingStatus StatusFromProperty() {
 }

 bool IsEnforcing() {
+ return false;
   if (ALLOW_PERMISSIVE_SELINUX) {
     return StatusFromProperty() == SELINUX_ENFORCING;
   }
 }
yk163@yk163:~/incar_work/chs/android12_zx/01/rockchip2_s/system$
```

通过 adb 查看生成在 data 目录下面的 log000 和 log111

```
rk3890_Android12:/data # ls
adb          backup          fonts          media
anr          bootchart      gsi            mediadrms
apex         cache          gsi_persistent_data misc
app          code_mem_100000_7a000 incremental  misc_ce
app-asec     cscb_mem_e000ed00_300 local         misc_de
app-ephemeral dalvik-cache   log000        nfc
app-lib      data           log111        ota
app-private  data_mem_20200000_40000 log_store     ota_package
app-staging  drm           lost+found    per_boot
```

这里的 log000，log111 可以通过 adb pull 到本地的 debug 目录下面。用于 debug 分析模组异常的问题。请将此文件及时发送给 seekwave。

4.2.RELEASE 版本如何关闭 LOG

关闭模组 log 的方法，可以通过在对应编译工程的 defconfig 中添加 seekwave 的 CONFIG_SEEKWAVE_PLD_RELEASE=y,来关闭模组的 log 输出到 AP 侧。实例操作如下：

```
diff --git a/drivers/misc/seekwaveplatform/Kconfig b/drivers/misc/seekwaveplatform/Kconfig
index 7e060a747..cb50cab07 100644
--- a/drivers/misc/seekwaveplatform/Kconfig
+++ b/drivers/misc/seekwaveplatform/Kconfig
@@ -6,6 +6,12 @@ menuconfig SEEKWAVE_BSP_DRIVERS
     if you want to buildin bsp driver
     please say "y"
     Thanks.
+config SEEKWAVE_PLD_RELEASE
+    bool "seekwave Platform support chip recovery mode"
+    depends on SEEKWAVE_BSP_DRIVERS
+    default n
```

需要注意：

在 USER 版本中的 defconfig 中需要配置 CONFIG_SEEKWAVE_PLD_RELEASE=y, 可以关闭模块的 log, 可以用于模组相关的功耗和性能测试。

4.3. RELEASE 版本如何开启 LOG

测试前执行:

insmod skw_bootcoms.ko log_path=XXX, 指定 log 路径

3 个 KO 驱动加载完后, 在执行一下脚本:

echo enable > /sys/kernel/debug/skwsdio/CPLog

echo START > /dev/LOG

echo disable > /sys/kernel/debug/skwsdio/recovery。

注: /sys/kernel/debug/skwsdio 路径不唯一, 与当前配置使用 boot & bus 相关。

4.4. 客户问题复现抓取 LOG 的方法

1, 查看一下是否有 selinux 的权限问题。

setenforce 0

2, 手动加载一下 seekwave 的 ko

insmod skw_sdio.ko 或者是 insmod skw_usb.ko

insmod skw_bootcoms.ko

insmod skw.ko (linux 系统必须这样操作)

3, 如果 debugfs 没有打开的情况下:

mount -t debugfs none /sys/kernel/debug

4, 手动执行一下开 log 的命令

sdio 的方案

echo enable > /sys/kernel/debug/skwsdio/CPLog

如果是 usb 的方案。指令是

echo enable > /sys/kernel/debug/skwusb/CPLog

```

SRWIFI/ SRWUSB/
2|rk3588_t_skw_pcie_lga50:/data # cd /sys/kernel/debug/skwusb/
rk3588_t_skw_pcie_lga50:/sys/kernel/debug/skwusb # ls -al
total 0
drwxr-xr-x  2 root root 0 2024-04-19 21:34 .
drwxr-xr-x 46 root root 0 1970-01-01 00:00 ..
-rw-rw-rw-  1 root root 0 2024-04-19 21:34 BT_ANT
-rw-rw-rw-  1 root root 0 2024-04-19 21:34 BT_UART1
-rw-rw-rw-  1 root root 0 2024-04-19 21:34 CLog
-rw-rw-rw-  1 root root 0 2024-04-19 21:34 Statistic
-rw-rw-rw-  1 root root 0 2024-04-19 21:34 USB_SPEED
-rw-rw-r--  1 root root 0 2024-04-19 21:34 Version
-rw-rw-rw-  1 root root 0 2024-04-19 21:34 log_level
-rw-rw-rw-  1 root root 0 2024-04-19 21:34 recovery
rk3588_t_skw_pcie_lga50:/sys/kernel/debug/skwusb # |

```

echo START > /dev/LOG

```

rk3588_t_skw_pcie_lga50:/ # cd /dev/
rk3588_t_skw_pcie_lga50:/dev # ls
ATC                                cpu_variant:arm64
BTBOOT                            cpuctl
BTCMD                            cpuset
BTISOC                            crypto
LOG                               device-mapper
__properties__                   dm-user
ashmem                            dma_heap
ashmem57cf35d0-e76c-4011-8a0f-c4deca2c747d  dri
binder                            drm_dp_aux0
binderfs                         drm_dp_aux1
blkio                             event-log-tags
block                             fd
boringssl                        fscklogs
bus                               full
cgroup_info                      fuse
console                          gpiochip0
cpu_dma_latency                  gpiochip1
cpu_variant:arm                  gpiochip2
rk3588_t_skw_pcie_lga50:/dev # echo START > LOG

```

5, 查看/data/下面的 log size 的变化

看到/data/目录下面的 log000 或者是 log00*等文件的 size 在不停的增加。

#ls -al /data/log000

```
rk3588_t_skw_pcie_lga50:/data # ls -al
total 4078
drwxrwx--x 50 system system 4096 2024-04-20 01:49 .
drwxr-xr-x 28 root root 4096 2024-04-19 21:51 ..
drwx----- 2 root root 3452 1970-01-01 00:00 adb
drwxrwxr-x 2 system system 3452 1970-01-01 00:00 anr
drwxr-xr-x 8 root system 3452 1970-01-01 00:00 apex
drwxrwx--x 2 system system 3452 1970-01-01 00:00 app
drwx----- 2 root root 3452 1970-01-01 00:00 app-asec
drwxrwx--x 2 system system 3452 1970-01-01 00:00 app-ephemeral
drwxrwx--x 2 system system 3452 1970-01-01 00:00 app-lib
drwxrwx--x 2 system system 3452 1970-01-01 00:00 app-private
drwxr-x--x 2 system system 3452 1970-01-01 00:00 app-staging
drwx----- 4 system system 3452 2024-04-19 21:21 backup
drwxr-xr-x 2 system system 3452 1970-01-01 00:00 bootanim
drwxr-xr-x 2 shell shell 3452 1970-01-01 00:00 bootchart
-rwxrwxrwx 1 root root 1024 2024-04-19 21:33 btbt_mem_41000400_400
-rwxrwxrwx 1 root root 1024 2024-04-19 21:33 btdm_mem_41000000_400
-rwxrwxrwx 1 root root 49152 2024-04-19 21:33 btem_mem_41022000_c000
-rwxrwxrwx 1 root root 1024 2024-04-19 21:33 btle_mem_41000800_400
drwxrwx--- 5 system cache 3452 1970-01-01 00:00 cache
-rwxrwxrwx 1 root root 499712 2024-04-19 21:33 code_mem_100000_7a000
-rwxrwxrwx 1 root root 768 2024-04-19 21:33 cscb_mem_e000ed00_300
drwxrwx--x 4 root root 3452 1970-01-01 00:00 dalvik-cache
drwxrwx--x 113 system system 20480 2024-04-19 21:21 data
-rwxrwxrwx 1 root root 262144 2024-04-19 21:33 data_mem_20200000_40000
drwxrwx--- 3 drm drm 3452 2024-04-19 21:21 drm
-rwxrwxrwx 1 root root 4224 2024-04-19 21:33 edma_mem_40188000_1080
drwxrwx--x 4 root root 3452 1970-01-01 00:00 fonts
drwx----- 5 root root 3452 1970-01-01 00:00 gsi
-rw----- 1 system system 1 1970-01-01 00:00 gsi_persistent_data
drwxrwx--x 2 system system 3452 1970-01-01 00:00 incremental
drwxr-x--x 5 root root 3452 1970-01-01 00:00 local
-rwxrwxrwx 1 root root 1352651 2024-04-20 01:48 log000
-rwxrwxrwx 1 root root 1351691 2024-04-20 01:48 log111
-rwxrwxrwx 1 root root 1 2024-04-20 01:48 log_store
```

6, 确认固件 mem 的是否有效

确认一下是否 dump 信息是否有效在串口或者是 adb shell 中

```
#echo "at+pldassert=1\r" > /dev/ATC
```

或者是

```
#echo 1 > /d/skwifi/chip1.sdio/assert
```

将/data/下面的 log000 否则 log00*发送给 SEEKWAVE 的研发人员确认。

7, 如果是 GKI 的版本如何获取固件 LOG

需要 cat /dev/LOG > log000

将 log000 文件发送给 SEEKWAVE 的研发

8, 客户复现问题的操作。

1) 开打模组的 log

- *SDIO mode*:

```
echo enable > /sys/kernel/debug/skwsdio/CPLo
```

- *USB mode*

```
echo enable > /sys/kernel/debug/skwusb/CPLo
```

- *PCIe mode*

```
echo enable > /sys/kernel/debug/skwpcie/CPLo
```

2) 开启 log 的 task

```
echo START > /dev/LOG
```

3) 关闭 recovery 功能 (其它 mode 参考 1)) :

```
echo disable > /sys/kernel/debug/skwsdio/recovery_debug
*SWT6652*
```

```
echo disable > /sys/kernel/debug/skwsdio/recovery
```

9, 支持手动 dump 模组的 mem 的方法 (其它 mode 参考 SDIO) :

首先要确认一下当前的版本是否有 dumpmem 的 debugfs 的节点

```
C:\Users\xw>adb shell
rk3399_Android12:/ # echo START > /dev/LOG
rk3399_Android12:/ # echo disable > /d/skwsdio/
CPLo      Statistic      Version      WiFi      dumpmem      log_level      recovery_debug
rk3399_Android12:/ # echo disable > /d/skwsdio/recovery_debug
rk3399_Android12:/ # echo 1 > /d/
```

操作如下:

```
# echo START > /dev/LOG
```

```
# echo disable > /sys/kernel/debug/skwsdio/recovery_debug
```

```
#echo dump > /sys/kernel/debug/skwsdio/dumpmem
```

在 console 可以看到 dumpmem 的信息

```

serial-com24 x
506.059711] [SKWIFI WARN] skw_cmd_tx_allowed: skw->flags: 0x8701, extra_flags: 0x0
506.059733] [SKWIFI ERROR] skw_sched_scan_start: failed, ret: -5
506.322371] [SKWSDIO INFO] skw_sdio_dumpmem: the dump status =1
506.322495] [SKWSDIO INFO] skw_sdio_dumpmem: dump mem start
506.322533] [SKWBOOT]:bt_state_event_notifier event = 5
506.322554] [SKWBOOT]:BT do nothing !!!!
506.322582] [SKWIFI DBG] skw_bsp_notifier: action: 5, skw flags: 0x8701
506.322672] [SKWBOOT]:modem_event_notifier event = 5
506.322724] [SKWLOG]:skw_modem_dumpmodem_start_rec enter
506.322786] [SKWLOG]:The -----Enter -----
506.324484] type=1400 audit(1716845179.810:951): avc: denied { read write } for cc
tem_data_root_file:s0 tclass=file permissive=1
506.324939] type=1400 audit(1716845179.810:952): avc: denied { open } for comm="st
tem_data_root_file:s0 tclass=file permissive=1
506.416138] [SKWLOG]:Dump data_mem_20200000_40000 memory done !!
506.416206] [SKWLOG]:the file close!!!
506.656495] [SKWLOG]:Dump code_mem_100000_7a000 memory done !!
506.656568] [SKWLOG]:the file close!!!
506.657292] [SKWLOG]:Dump cscb_mem_e000ed00_300 memory done !!
506.657320] [SKWLOG]:the file close!!!
506.689892] [SKWLOG]:Dump umem_mem_40b00000_c000 memory done !!
506.689935] [SKWLOG]:the file close!!!
506.691399] [SKWLOG]:Dump sdio_mem_401e0000_800 memory done !!
506.691427] [SKWLOG]:the file close!!!
506.692239] [SKWLOG]:Dump btdm_mem_41000000_400 memory done !!
506.692265] [SKWLOG]:the file close!!!
506.693907] [SKWLOG]:Dump btbt_mem_41000400_400 memory done !!
506.693937] [SKWLOG]:the file close!!!
506.695088] [SKWLOG]:Dump btle_mem_41000800_400 memory done !!
506.695115] [SKWLOG]:the file close!!!
506.774284] [SKWLOG]:Dump btem_mem_41022000_c000 memory done !!
506.774383] [SKWLOG]:the file close!!!
506.805384] [SKWLOG]:Dump wreg_mem_40820000_4000 memory done !!
506.805440] [SKWLOG]:the file close!!!
506.826536] [SKWLOG]:Dump phyr_mem_40830000_4000 memory done !!
506.826621] [SKWLOG]:the file close!!!
561.369328] [SKWLOG]:Dump smem_mem_40a00000_58000 memory done !!
561.369428] [SKWLOG]:the file close!!!
562.399551] [SKWLOG]:skw_modem_dumpmodem_stop_rec enter 0
562.399659] [SKWBOOT]:DUMP MEM EVENT Comming in !!!!

```

5. 常见问题分析 FAQ:

5.1. 编译问题.

- Kernel5.10 文件系统的 API: filp_open/kernel_read/kernel_write 接口限制内核使用, 导致出现编译错误:

```

96 MODPOST modules-only.symvers
97 ERROR: modpost: module skwbt uses symbol kernel_write from namespace VFS_internal_I_am_really_a_filesystem_and_am_NOT_a_driver, but does not import it.
98 ERROR: modpost: module skwbt uses symbol kernel_read from namespace VFS_internal_I_am_really_a_filesystem_and_am_NOT_a_driver, but does not import it.
99 ERROR: modpost: module skwbt uses symbol filp_open from namespace VFS_internal_I_am_really_a_filesystem_and_am_NOT_a_driver, but does not import it.
100 make[3]: *** [scripts/Makefile.modpost:169: modules-only.symvers] Error 1

```

可以在使用这些 API 的接口中增加:

驱动里添加:

```
MODULE_IMPORT_NS(VFS_internal_I_am_really_a_filesystem_and_am_NOT_a_driver);
```

- GKI 版本完全限制了文件接口的使用, 这将导致 driver 无法保存 CP log。

解决方案是在 defconfig 中注掉 CONFIG_NO_GKI

5.2. SDIO 扫卡失败

检查 DTS 中以下 sdio 的中 pwrseq 设置,

```
&sdio0 {
    max-frequency = <200000000>;
    supports-sdio;
    cap-sdio-irq;
    bus-width = <4>;
    disable-wp;
    cap-sd-highspeed;
    keep-power-in-suspend;
    mmc-pwrseq = <&sdio_pwrseq>;
    non-removable;
    num-slots = <1>;
    pinctrl-names = "default";
    pinctrl-0 = <&sdio0_bus4 &sdio0_cmd &sdio0_clk>;
    sd-uhs-sdr104;
    status = "okay";
};
```

以及 sdio_pwrseq 的设置:

```
sdio_pwrseq: sdio-pwrseq {
    compatible = "mmc-pwrseq-simple";
    clocks = <&rk808 1>;
    clock-names = "ext_clock";
    pinctrl-names = "default";
    pinctrl-0 = <&wifi_enable_h>;

    /*
     * On the module itself this is one of these (depending
     * on the actual card populated):
     * - SDIO_RESET_L_WL_REG_ON
     * - PDN (power down when low)
     */
    reset-gpios = <&gpio0 9 GPIO_ACTIVE_LOW>; /* GPIO0_B2 */
    chip-en-gpios = <&gpio0 9 GPIO_ACTIVE_LOW>; /* GPIO0_B1 */
};
```

其中 GPIO 的应该连接到 WiFi 芯片的 CHIP_EN pin12

5.3. Load firmware 失败

```

5.836236] cfg80211: failed to load regulatory.db
5.858618] [SKWSDIO INFO] skw_sdio_io_init: skw_sdio_init entry
5.859304] [SKWSDIO INFO] skw_sdio_scan_card: sdio_scan_card
5.859480] skw_sdio_probe: func->class=0, vendor=0x0000, device=0x0000, func_num=0x0001, clock=200000000 blksize=0x200 max_bkcnt 2048
5.859535] [SKWSDIO INFO] skw_sdio_probe: sdio.enable_func ret=0 type 0
5.859572] [SKWSDIO INFO] skw_sdio_probe: enable_func1 done
5.859637] [SKWSDIO INFO] check_chipid: chip id:sw6160 used sdio10
5.864061] [SKWSDIO INFO] skw_sdio_scan_card: scan end
5.864085] [SKWSDIO INFO] skw_sdio_io_init: skw_sdio_bus_init ok
5.868770] [SKWSDIO INFO] skw_sdio_update_thread: : the line :1201 Enter
udhpc: sending discover
5.881709] register char device:ATC 242:0
5.885381] register char device:LOG 242:1
5.887806] [SKWLOG] skw_sdio_log_init enter
5.891451] [SKWLOG] skw_sdio_log_start_rec enter
5.891617] register char device:LOOPCHECK 242:7
5.891720] [SKWLOG] log path = /data
5.891787] [SKWLOG_ERR] open log_store /data/log_store failed:-2
5.893485] register char device:BTDATA 242:2
5.893498] register char device:BTCMD 242:3
5.895934] register char device:BTAUDIO 242:4
5.897595] [SKWBOOT_ERR] seekwave boot_parse_dt: gpio_host request fail ret=-16
5.897631] [SKWBOOT] seekwave boot_parse_dt: gpio out:67 gpio in:70 state = 0
5.897682] (NULL device *): Direct firmware load for RAW_RW_KERNEL_DRAM failed with error -2
5.898190] request_firmware RAW_RW_KERNEL_DRAM fail
5.898208] [SKWBOOT] image_size=0, ret=-2
5.898723] register char device:BTBOOT 242:8
5.900120] [SKWSDIO INFO] skw_sdio_cpdebug_boot: not download CP from AP!!!!
5.924555] [SKWSDIO INFO] skw_sdio_set_dma_type: dma_type=1,adma_rx_enable=1
5.924555]
5.924634] [SKWSDIO INFO] skw_sdio_host_irq_init: gpio_in:70,gpio_out:67 irq 64
5.924690] [SKWSDIO INFO] skw_sdio_cpdebug_boot: CP DUEGBOOT Done!!!
5.924710] [SKWSDIO INFO] skw_sdio_cp_service_ops: -----The ENTER-----
5.924717] [SKWSDIO INFO] skw_sdio_cp_service_ops: Have no service ops!
5.924724] [SKWSDIO INFO] skw_boot_loader: boot loader ops end!!!
5.924729] [SKWBOOT] skw_doubleting_First_boot: first boot pass
5.924834] [SKWSDIO ERROR] skw_sdio_adma_parser: skw_sdio_adma_parser[0]:ch:255 len:0x8c 0xffffffffff 0xffffffffff 0xffffffffff 0xffffffffff 0xffffffffff
5.924864] [SKWSDIO INFO] skw_sdio_adma_parser: 1:1:20322:7eb3e2a (5.10.110)
5.9248823] [SKWSDIO INFO] skw_drv_probe: MAC: 00:00:00:00:00:00
5.924883] [SKWBOOT] skw_download_signal_ops line:613 download data ops done
5.924884] [SKWSDIO INFO] skw_sdio_cp_service_ops: -----The ENTER-----
5.924885] [SKWSDIO INFO] skw_wifi_service_start: Enter STARTWiFi 0
6.004318] [SKWSDIO INFO] send_modem_service_command: ret = 0 cmd 0
6.004318] [SKWLOG_ERR] open log_store /data/log_store failed:-2
7.014459] [SKWSDIO ERROR] skw_check_cp_ready: check CP-ready time out
7.014496] [SKWSDIO ERROR] skw_sdio_cp_service_ops: skw_wifi_service_start time:0 failed, error:-62
7.014518] [SKWSDIO INFO] skw_wifi_service_start: Enter STARTWiFi 0
7.041179] [SKWSDIO INFO] send_modem_service_command: ret = 0 cmd 0
7.041179] [SKWLOG_ERR] open log_store /data/log_store failed:-2
8.054458] [SKWSDIO ERROR] skw_check_cp_ready: check CP-ready time out
8.054492] [SKWSDIO ERROR] skw_sdio_cp_service_ops: skw_wifi_service_start time:1 failed, error:-62
8.054512] [SKWSDIO INFO] skw_wifi_service_start: Enter STARTWiFi 0
8.081170] [SKWSDIO INFO] send_modem_service_command: ret = 0 cmd 0
8.941290] [SKWLOG_ERR] open log_store /data/log_store failed:-2
udhpc: sending
9.094476] [SKWSDIO ERROR] skw_check_cp_ready: check CP-ready time out
9.094536] [SKWSDIO ERROR] skw_sdio_cp_service_ops: skw_wifi_service_start time:2 failed, error:-62
9.094561] [SKWSDIO ERROR] skw_boot_loader: line:1232 skw_boot_loader fail ret=-1
9.094584] [SKWBOOT_ERR] skw_start_wifi_service,line:758 boot fail
9.094584] [SKWLOG_ERR] open log_store /data/log_store failed:-2
10.96170] [SKWLOG_ERR] open log_store /data/log_store failed:-2
11.147813] [SKWIFI ERROR] skw_msg_xmt_timeout: skw_cmd_syn_version[2], seq: 1, ret: -110, flags: 0x300 timeout:600
11.174532] [SKWSDIO ERROR] send_modem_assert_command: skw_cmd_syn_version[2], seq: 1, ret: -110, flags: 0x300 timeout:600
11.174569] [SKWIFI ERROR] skw_sync_cmd_event_version: ret: -110

```

Firmware image 路径放在:

Android 版本 default 是在/vendor/etc/firmware

Linux 版本建议放在/lib/firmware 目录

Load firmware 成功时有如下 log:

```

[20230607 16:54:10:436] 3.650203] register char device:BTCMD 511:3
[20230607 16:54:10:436] 3.650382] register char device:BTAUDIO 511:4
[20230607 16:54:10:436] 3.650913] [SKWBOOT] seekwave boot_parse_dt: gpio out:74 gpio in:79 state = 0
[20230607 16:54:10:442] 3.652990] [SKWBOOT] boot data dram_img_data 000000001db4b24a
[20230607 16:54:10:455] 3.674726] [SKWBOOT] image_size=490736,234256, ret=0
[20230607 16:54:10:455] 3.674756] [SKWBOOT] skw_boot_init line:684,the tail_offset --0x128,the head_offset --0xd4 ,iram_addr=0x100000,dram_addr=
[20230607 16:54:10:455] 3.674761] [SKWBOOT] skw_boot_init line:688 analysis the img module
[20230607 16:54:10:455] 3.674767] [SKWBOOT] skw_boot_init line:697 dl_addr=0x110000, write_addr=0x110000, index=0x1,data_size=0x245c
[20230607 16:54:10:455] 3.674771] [SKWBOOT] skw_boot_init line:697 dl_addr=0x20200000, write_addr=0x20200000, index=0x1,data_size=0x8410
[20230607 16:54:10:455] 3.674776] [SKWBOOT] skw_boot_init line:697 dl_addr=0x112600, write_addr=0x112600, index=0x2,data_size=0x30920
[20230607 16:54:10:460] 3.674781] [SKWBOOT] skw_boot_init line:697 dl_addr=0x2020ac00, write_addr=0x2020ac00, index=0x2,data_size=0x13a88
[20230607 16:54:10:460] 3.674785] [SKWBOOT] skw_boot_init line:697 dl_addr=0x143000, write_addr=0x143000, index=0x3,data_size=0x34c0
[20230607 16:54:10:460] 3.674790] [SKWBOOT] skw_boot_init line:697 dl_addr=0x20232000, write_addr=0x20232000, index=0x3,data_size=0x4a38
[20230607 16:54:10:479] 3.674795] [SKWBOOT] skw_boot_init line:697 dl_addr=0x20238f08, write_addr=0x20238f08, index=0x3,data_size=0x408
[20230607 16:54:10:479] 3.675120] register char device:BTBOOT 511:8
[20230607 16:54:10:505] 3.701508] [SKWSDIO INFO] skw_sdio_set_dma_type: dma_type=1,adma_rx_enable=1
[20230607 16:54:10:505] 3.701508]
[20230607 16:54:10:517] 3.728378] [SKWSDIO INFO] skw_sdio_host_irq_init: gpio_in:79,gpio_out:74 irq 107

```

常见的客户问题汇总

固件的放置问题:

1, 固件的路径在/sys/class/modules/firmware.class/paramter/path

直接修改 firmware_class 模块参数/sys/module/firmware_class/parameters/path

2,固件的放置路径在各个 board 中。


```

junwei.jiang@ip-10-10-6-130:/mnt/efs/junwei.jiang/rockchip/rockchip12_mp/vendor/rockchip/common$ cd wifi/
junwei.jiang@ip-10-10-6-130:/mnt/efs/junwei.jiang/rockchip12_mp/vendor/rockchip/common/wifi$ ls
firmware iwconfig iwlist modules ssv6xxx wifi.mk
junwei.jiang@ip-10-10-6-130:/mnt/efs/junwei.jiang/rockchip12_mp/vendor/rockchip/common/wifi$ cd firmware/
junwei.jiang@ip-10-10-6-130:/mnt/efs/junwei.jiang/rockchip12_mp/vendor/rockchip/common/wifi/firmware$ ls
4359 cypress_auto.clm_blob ctm_bcm43752a2_ag.blob fw_RK903b2.bin fw_bcm43241b4_ag_p2p.bin fw_bcm43438a0_apsta.bin
AP6275P_NVRAM_V1.1_20200702.txt ctm_bcm43752a2_pcie_ag.blob fw_RK903b2_apsta.bin fw_bcm4330.bin fw_bcm43438a0_p2p.bin
EA65210F_SEEKWAVE_R00005.bin ctm_bcm4375b4_pcie_ag.blob fw_RK903b2_p2p.bin fw_bcm4330_apsta.bin fw_bcm43438a1.bin
EA65210T_SEEKWAVE_R00005.bin dpd_matrix.ini fw_awnb108.bin fw_bcm43341b0_ag.bin fw_bcm43438a1_apsta.bin
EA66210T_SEEKWAVE_R00005.bin dpd_param.ini fw_awnb108_apsta.bin fw_bcm43341b0_ag_apsta.bin fw_bcm43438a1_p2p.bin
EA66210T_SEEKWAVE_R00000.bin fw_RK901.bin fw_bcm40181a2.bin fw_bcm43341b0_ag_p2p.bin fw_bcm43455c0_ag.bin
EA66210T_SEEKWAVE_R00000.in fw_RK901a0.bin fw_bcm40181a2_apsta.bin fw_bcm43341b1_ag.bin fw_bcm43455c0_ag_apsta.bin
EA66210T_SEEKWAVE_R00002.bin fw_RK901a0_apsta.bin fw_bcm40183b2.bin fw_bcm43341b1_ag_apsta.bin fw_bcm43455c0_ag_p2p.bin
EA66210T_SEEKWAVE_R00002.in fw_RK901a2.bin fw_bcm40183b2_ag.bin fw_bcm43341b1_ag_p2p.bin fw_bcm43456c5_ag.bin
EA66210T_SEEKWAVE_R00005.bin fw_RK901a2_apsta.bin fw_bcm40183b2_ag_apsta.bin fw_bcm4339a0_ag.bin fw_bcm43456c5_ag_apsta.bin
RT2870AP.dat fw_RK901a2_p2p.bin fw_bcm40183b2_ag_p2p.bin fw_bcm4339a0_ag_apsta.bin fw_bcm4354a1_ag.bin
RT2870APCard.dat fw_RK903.bin fw_bcm40183b2_apsta.bin fw_bcm4339a0_ag_p2p.bin fw_bcm4354a1_ag_apsta.bin
RT2870STA.dat fw_RK903_ag.bin fw_bcm40183b2_p2p.bin fw_bcm4339a0_ag_p2p.bin fw_bcm4354a1_ag_p2p.bin
RT2870STACard.dat fw_RK903_ag_apsta.bin fw_bcm43013c1_ag.bin fw_bcm43436b0_ag_apsta.bin fw_bcm4356a2_ag.bin
ctm_bcm43013c1_ag.blob fw_RK903_ag_p2p.bin fw_bcm43241b4_ag.bin fw_bcm43436b0_p2p.bin fw_bcm4356a2_ag_apsta.bin
ctm_bcm4359c51a2_ag.blob fw_RK903_p2p.bin fw_bcm43241b4_ag_apsta.bin fw_bcm43438a0_p2p.bin fw_bcm4356a2_ag_p2p.bin
junwei.jiang@ip-10-10-6-130:/mnt/efs/junwei.jiang/rockchip12_mp/vendor/rockchip/common/wifi/firmware$

```

3, firmware 的 path 的配置方法

这里是在 devices 下面的配置。根据 board 的配置不同配置。

```

./rockchip/common/BoardConfig.mk:107:BOARD_KERNEL_CMDLINE := console=ttyFIQ0 firmware class.path=/vendor/etc/firmware

```

4, firmware 的 kernel 的添加新的路径的方法

kernel-5.10/drivers/base/firmware_loader/main.c 中的路径:

```

rk3128-86v_h1401.dts  firmware_class.c  wifi_bt_common.mk  device_rockchip_...  device_rockchip_...
264
265 /* direct firmware loading support */
266 static char fw_path_para[256];
267 static const char * const fw_path[] = {
268     "/vendor/firmware",
269     "/vendor/firmware/",
270     "/etc/firmware/",
271     "/etc/firmware",
272     "/vendor/rockchip/common/SDIO-Firmware",
273     fw_path_para,
274     "/lib/firmware/updates/" ,//UTS_RELEASE,
275     "/lib/firmware/updates",
276     "/libA/firmware/" ,//UTS_RELEASE,
277     "/lib/firmware",
278     "/vendor/rockchip/common/SDIO-Firmware",
279     "/vendor/rockchip/common/SDIO-Firmware/",
280     "../../../../vendor/rockchip/common/SDIO-Firmware/",

```

5, 各个厂家的配置方法:

```

^C
junwei.jiang@tp-10-10-6-130:/mnt/efs/junwei.jiang/rk_android13$ cd kernel-5.10/
junwei.jiang@tp-10-10-6-130:/mnt/efs/junwei.jiang/rk_android13/kernel-5.10$ grep "vendor/etc/firmware" -nr ./
./config:2174:CONFIG_BCM2HD_FW_PATH="/vendor/etc/firmware/fw_bcm2hd.bin"
./config:2175:CONFIG_BCM2HD_NVRAM_PATH="/vendor/etc/firmware/nvram.txt"
./include/config/auto.conf:525:CONFIG_BCM2HD_FW_PATH="/vendor/etc/firmware/fw_bcm2hd.bin"
./include/config/auto.conf:525:CONFIG_BCM2HD_NVRAM_PATH="/vendor/etc/firmware/nvram.txt"
./include/generated/autoconf.h:975:#define CONFIG_BCM2HD_FW_PATH "/vendor/etc/firmware/fw_bcm2hd.bin"
./include/generated/autoconf.h:975:#define CONFIG_BCM2HD_NVRAM_PATH "/vendor/etc/firmware/nvram.txt"
./tmp.config.w00pspxfs:2000:CONFIG_BCM2HD_FW_PATH="/vendor/etc/firmware/fw_bcm2hd.bin"
./tmp.config.w00pspxfs:2001:CONFIG_BCM2HD_NVRAM_PATH="/vendor/etc/firmware/nvram.txt"
Binary file ./GTAGS matches
./arch/arm/configs/rv1126-evb.config:18:CONFIG_BCM2HD_FW_PATH="/vendor/etc/firmware/fw_bcm2hd.bin"
./arch/arm/configs/rv1126-evb.config:11:CONFIG_BCM2HD_NVRAM_PATH="/vendor/etc/firmware/nvram.txt"
./arch/arm64/boot/dts/rockchip/rk3399-blend-android-mpl.dts:17: svb160_dram_path = "/vendor/etc/firmware/RAM_RW_KERNEL_DRAM";
./arch/arm64/boot/dts/rockchip/rk3399-blend-android-mpl.dts:18: svb160_iram_path = "/vendor/etc/firmware/ROM_EXEC_KERNEL_IRAM";
./kernel/config.data:2174:CONFIG_BCM2HD_FW_PATH="/vendor/etc/firmware/fw_bcm2hd.bin"
./kernel/config.data:2175:CONFIG_BCM2HD_NVRAM_PATH="/vendor/etc/firmware/nvram.txt"
./config.old:1977:CONFIG_BCM2HD_FW_PATH="/vendor/etc/firmware/fw_bcm2hd.bin"
./config.old:1977:CONFIG_BCM2HD_NVRAM_PATH="/vendor/etc/firmware/nvram.txt"
./drivers/misc/seekwaveplatform/skwtll/skw_boot.c:847: //boot_data->iram_file_path = "/vendor/etc/firmware/ROM_EXEC_KERNEL_IRAM.bin";
./drivers/misc/seekwaveplatform/skwtll/skw_boot.c:847: //boot_data->dram_file_path = "/vendor/etc/firmware/RAM_RW_KERNEL_DRAM.bin";
./drivers/net/wireless/rockchip_wlan/rkwtll/Makefile:409: EXTRA_CFLAGS += -DUSE_MAP_PATH="/vendor/etc/firmware/wifi_efuse_${MODULE_NAME}.map"
Binary file ./drivers/net/wireless/rockchip_wlan/rkwtll/bcm2hd/bcm2hd.o matches
Binary file ./drivers/net/wireless/rockchip_wlan/rkwtll/bcm2hd/bcm2hd.o matches
./drivers/net/wireless/rockchip_wlan/rkwtll/Kconfig:42: default "/vendor/etc/firmware/fw_bcm2hd.bin"
./drivers/net/wireless/rockchip_wlan/rkwtll/Kconfig:48: default "/vendor/etc/firmware/nvram.txt"
./drivers/net/wireless/rockchip_wlan/rkwtll/bcm2hd_indep_power/Kconfig:16: default "/vendor/etc/firmware/fw_bcm2hd.bin"
./drivers/net/wireless/rockchip_wlan/rkwtll/bcm2hd_indep_power/Kconfig:17: default "/vendor/etc/firmware/nvram.txt"
./drivers/net/wireless/rockchip_wlan/ssv6xxx/ssvdevice.c:83:char DEFAULT_CFG_PATH[] = "/vendor/etc/firmware/ssv6051-wifi.cfg";
./drivers/net/wireless/rockchip_wlan/ssv6xxx/firmware/ssv6051-wifi.cfg:12:firmware_path = /vendor/etc/firmware/
./drivers/net/wireless/rockchip_wlan/infineon/bcm2hd/dhd_linux.c:9802: fw = "/vendor/etc/firmware/fw_firmware_pcie.bin";
./drivers/net/wireless/rockchip_wlan/infineon/bcm2hd/dhd_linux.c:9803: nv = "/vendor/etc/firmware/nvram.txt";
./drivers/net/wireless/rockchip_wlan/infineon/bcm2hd/dhd_linux.c:9806: fw = "/vendor/etc/firmware/fw_bcm8459_pcie.bin";
./drivers/net/wireless/rockchip_wlan/infineon/bcm2hd/dhd_linux.c:9807: nv = "/vendor/etc/firmware/nvram_cy80459.txt";
./drivers/net/wireless/rockchip_wlan/infineon/bcm2hd/dhd_common.c:5720: clm_blob_path = "/vendor/etc/firmware/4359_cypress_auto.clm_blob";//VENDOR_PATH CONFIG_BCM2HD_CLM_PATH;
./drivers/net/wireless/rockchip_wlan/cywdh/bcm2hd/dhd_linux.c:431:char clm_path[MOD_PARAM_PATHLEN] = "/vendor/etc/firmware/cyfmac4373-sdio.clm_blob";
./drivers/net/wireless/rockchip_wlan/cywdh/bcm2hd/dhd_linux.c:9807:#define DEFAULT_BCM2HD_FW_PATH "/vendor/etc/firmware/"
./drivers/net/wireless/rockchip_wlan/cywdh/bcm2hd/dhd_linux.c:9807:#define DEFAULT_BCM2HD_NVRAM_PATH "/vendor/etc/firmware/"
./drivers/net/wireless/rockchip_wlan/cywdh/bcm2hd/dhd.h:404:#define CONFIG_BCM2HD_CLM_PATH "/vendor/etc/firmware/bcm2hd.clm.blob"
./drivers/net/wireless/rockchip_wlan/cywdh/bcm2hd/dhd.h:406:#define CONFIG_BCM2HD_CLM_PATH "/vendor/etc/firmware/bcm2hd.clm.blob"
./drivers/net/wireless/rockchip_wlan/cywdh/bcm2hd/Makefile:269: DHD_FLAGS += -DHD_FIRMWARE_DIR_PATH="/vendor/etc/firmware/"
./drivers/net/wireless/rockchip_wlan/rkwtll/Makefile:409: EXTRA_CFLAGS += -DUSE_MAP_PATH="/vendor/etc/firmware/wifi_efuse_${MODULE_NAME}.map"
junwei.jiang@tp-10-10-6-130:/mnt/efs/junwei.jiang/rk_android13/kernel-5.10$

```

5.4. Boot 完成后没有收到版本信息

正常 boot 完成后，默认会有以下 log：

```

[SKWSDIO INFO] skw_sdio_handle_packet: LOOPCHECK channel received: trunk_W23.20.2-rev24520-rev24520-rev24490 202:
[SKWSDIO INFO] skw_sdio_handle_packet: firmware version: trunk_W23.20.2-rev24520-rev24520-rev24490 20230522-10:5:
:trunk_W23.20.2-rev24520-rev24520-rev24490 20230522-10:50:54

```

如果 boot 完成后，没有这条 log 输出，可能是 GPIO 配置不正确，建议按照 3.5 DTS 配置说明检查 DTS 中 GPIO 的配置和 HW 设计是否一致。

1) WiFi command timeout

[41.730967] [SKWIFI ERROR] skw_sync_cmd_event_version: ret: -110

WiFi driver 超时，检查检查 DTS 中 GPIO 的配置 HW 设计是否一致。

如果 GPIO 设置没有问题，可以调整 GPIO_wake_chip 的驱动能力试试。

2) SDIO 模式，boot 完成后，单独 BT 启动正常，单独开启 WiFi，WiFi scan 失败。

```

[ 86.595261] [SKWSDIO INFO] skw_sdio_rx_thread: line:1054 total:115 next_pac0, valid len:22 cnt 1
[ 86.595276] [SKWSDIO INFO] skw_sdio_adma_parser: skw_sdio_adma_parser[0]:ch:5 len:0x16 0x00000000 0x00160000 : 0x0000000A 0xffff0000 0xffffffff
[ 86.595314] RX thread suspending
[ 86.596390] skw_gpio_irq_handler enter...
[ 86.596484] RX thread resume
[ 86.596506] skw_sdio_rx_thread GPIO_state = 1
[ 86.596569] [SKWSDIO INFO] skw_sdio_rx_thread: line:1016 cp fifo status(106,105) ret=0
[ 86.596646] [SKWSDIO ERROR] skw_sdio_start_transfer: skw_sdio_start_transfer:CMD53 read failed error=-5
[ 86.596745] skw_gpio_irq_handler enter...
[ 86.597203] dwmmc_rockchip fe2c0000.dwmmc: All phases bad!
[ 86.597221] mmc1: tuning execution failed: -5
[ 86.597249] [SKWSDIO ERROR] skw_sdio_abort: SDIO Abort, SDIO_VER, CCCR:0xff
[ 86.597263] [SKWSDIO ERROR] skw_sdio_rx_thread: skw_sdio_rx_thread adma read fail ret:-5
[ 86.597276] RX thread suspending
[ 86.597284] RX thread resume
[ 86.597296] skw_sdio_rx_thread GPIO_state = 1
[ 86.597319] [SKWSDIO INFO] skw_sdio_rx_thread: line:1016 cp fifo status(255,106) ret=-5
[ 86.597331] RX thread suspending
[ 86.603940] [SKWIFI DBG] skw_android_cmd: wlan0: BTCOEEXSCAN-STOP
[ 86.604645] [SKWIFI DBG] skw_android_cmd: wlan0: RXFILTER-STOP
[ 86.605484] [SKWIFI DBG] skw_android_cmd: wlan0: RXFILTER-ADD 2
[ 86.606044] [SKWIFI DBG] skw_android_cmd: wlan0: RXFILTER-START
[ 86.606682] [SKWIFI DBG] skw_android_cmd: wlan0: RXFILTER-STOP
[ 86.607109] [SKWIFI DBG] skw_android_cmd: wlan0: RXFILTER-ADD 3
[ 86.607390] [SKWIFI DBG] skw_android_cmd: wlan0: RXFILTER-START
[ 86.607761] [SKWIFI DBG] skw_android_cmd: wlan0: SETSUSPENDMODE 1
[ 86.608325] [SKWIFI DBG] skw_set_power_mgmt: wlan0, enabled: 1, timeout: -1
[ 86.645484] [SKWIFI DBG] skw_android_cmd: wlan0: SETSUSPENDMODE 0
[ 89.087623] healthd: battery l=41 v=7432 t=18.8 h=2 st=2 c=-557000 fc=5000000 chg=
[ 94.697868] [SKWIFI DBG] skw_scan_done: inst: 0, aborted: 1, scan result: 0
[ 94.698260] [SKWSDIO INFO] wifi_send_cmd: wifi_send_cmd port5 sg_num=2 total=512 0x5007e00 0xb0600
[ 96.746836] [SKWIFI ERROR] skw_msg_xmit_timeout: SKW_CMD_STOP_SCAN[6], seq: 11, ret: -110, flags: 0x700 timeout:2000
[ 96.747137] [SKWSDIO ERROR] send_modem_assert_command: send_modem_assert_command ret=0 cmd: 0x5007e00 0xb0600 0x8 :-205312--203263
[ 96.748228] [SKWIFI DBG] dev_dump_config wlan0: idx: 0

```

建议调整 WIFI

RF 的功率试试

5.5. USB 枚举失败

USB 模组调试，不建议飞线方式连接模组，USB 模式枚举不可靠，
当贴上 USB 模组后，Kernel 启动后，lsusb 没有新的 USB 设备出现
检查 chip_en（pin12）的状态是否为高

5.6. USB 枚举成功，但是 USB boot 报错

当贴上 USB 模组后，lsusb 有新的 USB 设备出现,比如：0x0483:0x5721。
但是 USB boot 失败

```
[ 46.861914] [SKWBOOT]:boot data dram_img_data d1840000
[ 46.930021] [SKWBOOT]:image_size=485248,234040, ret=0
[ 46.930033] [SKWBOOT]:skw_boot_init line:686,the tail_offset ---0x128, the head_offset --0xd4 ,iram_addr=0x100000,dram_addr=0x20200000,
[ 46.930036] [SKWBOOT]:skw_boot_init line:690 analysis the img module
[ 46.930041] [SKWBOOT]:skw_boot_init line:699 dl_addr=0x110000, write_addr=0x110000, index=0x1,data_size=0x2434
[ 46.930046] [SKWBOOT]:skw_boot_init line:699 dl_addr=0x20200000, write_addr=0x20200000, index=0x1,data_size=0x6eb4
[ 46.930051] [SKWBOOT]:skw_boot_init line:699 dl_addr=0x112600, write_addr=0x112600, index=0x2,data_size=0x2e534
[ 46.930055] [SKWBOOT]:skw_boot_init line:699 dl_addr=0x2020ac00, write_addr=0x2020ac00, index=0x2,data_size=0x112b0
[ 46.930060] [SKWBOOT]:skw_boot_init line:699 dl_addr=0x143000, write_addr=0x143000, index=0x3,data_size=0x33780
[ 46.930065] [SKWBOOT]:skw_boot_init line:699 dl_addr=0x20232000, write_addr=0x20232000, index=0x3,data_size=0x4968
[ 46.930070] [SKWBOOT]:skw_boot_init line:699 dl_addr=0x20238e30, write_addr=0x2023f800, index=0x3,data_size=0x408
[ 46.930494] skw_ucom: probe of skw_ucom.1:auto failed with error -16
[ 46.930509] [SKWUSB INFO] skw_boot_loader: status:0 , chip_en_gpio=1
[ 46.930509] [SKWUSB INFO] skw_boot_loader: USB FIRST BOOT...
[ 46.930514] [SKWBOOT]:skw_doubleimg_first_boot first boot pass
[ 46.943437] [SKWIFI INFO] VERSION: 1.1.230423.cce9bd6 (4.9.127_s5)
[ 47.174303] type=1400 audit(1682580344.692:579): avc: denied { read } for pid=1742 comm="HotLugThread" scontext=u:cameraserver:s0 tcontext=u:
[ 47.194048] type=1400 audit(1682580344.692:579): avc: denied { read } for pid=1742 comm="HotLugThread" scontext=u:cameraserver:s0 tcontext=u:
[ 47.197948] usb 1-3: USB disconnect, device number 3
[ 47.198358] dlloader_send_command send cmd error ret -108 actual_len 0 command_len 4
[ 47.198362] [SKWUSB INFO] skw_usb_io_disconnect: interface[0] disconnected 0
[ 47.198366] usb 1-3: get version error
[ 47.198372] dlloader_send_command send cmd error ret -5 actual_len 0 command_len 20
[ 47.198376] usb 1-3: start download command failed
[ 47.198380] [SKWUSB INFO] dlloader_work: dlloader_work dram download img fail !!!
[ 47.198384] dlloader_send_command send cmd error ret -5 actual_len 0 command_len 20
[ 47.198388] usb 1-3: start download command failed
[ 47.198392] dlloader_send_command send cmd error ret -5 actual_len 0 command_len 16
[ 47.198396] usb 1-3: exec command is error
```

这是由于开机过

程枚举了 2 次，这可能是由于 chip_en 处于常高，但是再开机的 uboot 阶段枚举过，kernel 启动后重新枚举，导致 USB 下载失败。

解决办法，chip_en 是 driver 可控制，通过 driver 可以对芯片 reset。

5.7. USB boot 完成后，没有回复版本信息

USB load firmware 完成后，firmware 运行并且枚举正常，但是没有收到 firmware 的 verison，同时出现以下 error:

```
[ 29.022773] [SKWLOG]: open /data/log111 for CP log record
[ 29.282772] [SKWUSB INFO] bulkin_complete: endpoint8 actual = 0 status -71
[ 29.282855] [SKWUSB INFO] bulkin_complete: endpoint7 actual = 0 status -71
[ 29.282925] [SKWLOG_ERR]:skw_sdio_log_to_file_work read log data err:-71
```

```
[ 28.170609] register char device:B1BOOI 245:12
[ 28.170710] [SKWUSB INFO] skw_boot_loader: status:0 , chip_en_gpio=93
[ 28.170769] [SKWUSB INFO] skw_boot_loader: USB FIRST BOOT...
[ 28.170820] [SKWBOOT]:skw_doubleimg_first_boot first boot pass
[ 28.171123] usb 1-1: dloader connect susscess...
[ 28.456198] usb 1-1: USB disconnect, device number 2
[ 28.456312] [SKWUSB INFO] skw_usb_io_disconnect: interface[0] disconnected 0
[ 28.844930] usb 1-1: new high-speed USB device number 3 using xhci-hcd
[ 29.018032] usb 1-1: intf[0] is registered: ep count 2 WIFICMD
[ 29.018195] [SKWUSB INFO] usb_port_entry: usb_port_entry0 (MPC 2 buffer_size 0xc40 )is running
[ 29.018523] usb 1-1: intf[1] is registered: ep count 2 WIFIDATA
[ 29.018776] [SKWUSB INFO] usb_port_async_entry: usb_port_async_entry 1 running packet 16 ...
[ 29.019081] usb 1-1: intf[2] is registered: ep count 2 BTDATA
[ 29.019442] usb 1-1: intf[3] is registered: ep count 2 BTCMD
[ 29.019961] usb 1-1: intf[4] is registered: ep count 2 BTAUDIO
[ 29.020521] register char device:ATC 245:5
[ 29.020607] usb 1-1: intf[5] is registered: ep count 2 ATC
[ 29.021227] register char device:LOG 245:6
[ 29.021327] [SKWLOG]:skw_sdio_log_init enter
[ 29.022050] [SKWLOG]:skw_sdio_log_start_rec enter
[ 29.022074] [SKWLOG]:log path = /data
[ 29.022088] usb 1-1: intf[6] is registered: ep count 2 LOG
[ 29.022654] register char device:LOOP 245:7
[ 29.022735] usb 1-1: intf[7] is registered: ep count 2 LOOP
[ 29.022773] [SKWLOG]: open /data/log111 for CP log record
[ 29.282772] [SKWUSB INFO] bulkin_complete: endpoint8 actual = 0 status -71
[ 29.282855] [SKWUSB INFO] bulkin_complete: endpoint7 actual = 0 status -71
[ 29.282925] [SKWLOG_ERR]:skw_sdio_log_to_file_work read log data err:-71
[ 29.282935] [SKWUSB INFO] bulkin_complete: endpoint1 actual = 0 status -71
[ 29.282949] [SKWUSB INFO] bulkin_async_complete: endpoint2 actual = 0 status -71
[ 29.282974] usb 1-1: usb_port_async_entry bulkin read status=-71 state=1
[ 29.282979] [SKWUSB INFO] bulkin_async_complete: endpoint2 actual = 0 status -71
[ 29.282982] usb 1-1: usb_port_async_entry-port1 is stopped
[ 29.282991] usb 1-1: bulkin read len=-71
[ 29.282997] usb 1-1: usb_loopcheck_entry-port7 is stopped
[ 29.283003] usb 1-1: usb_loopcheck_entry write_context = (null)
[ 29.283070] [SKWUSB INFO] bulkin_async_complete: endpoint2 actual = 0 status -2
```

建议和 HW 核

对电源功率是否满足要求：

3.3V 的平均电流：500mA，峰值 1A

5.8. 关闭 recoverymode 复现问题

首先是要在 userdebug 的版本，kernel 要支持 debugfs，这样我们可以在

SDIO 通道模型下可以通过/sys/kernel/debug/skwsdio/recovery 节点关闭 recovery 功能。这样保证业务异常后停在现场。

操作如下：echo disable > /sys/kernel/debug/skwsdio/recovery

USB 通道，PCIE 通道同上。

5.9. PCIe link fail 或枚举失败

1) 检查主控 PCIe 管脚#PERSTn 是否配置正确

- 2) 不同模组，芯片的 chip_en 管脚连接有差异，如 NGFF 和 LGA50 封装的模组，NGFF 模组上芯片的 chip_en 管脚会上拉，给芯片供电 3.3V，芯片就可以正常启动；而 LGA50 封装的模组上，芯片 chip_en 管脚从模组引脚引出接到主控芯片的管脚（GPIO）上，需要主控芯片在 PCIe training 之前拉高对应的 GPIO，否则无法 link up

6. AP 平台的 check list:

6.1.Rockchip 相关的平台的 checklist

Rockchip 关于 WiFi 模组这类外设的处理逻辑一般分为两种：

一种是不依赖 rfkill（linux 的 rfkill 子系统提供了用于禁用系统中任何无线电发射器的通用接口）的 non-removable 的方式，sdio 作为内部 card 处理。这里 sdio host 端在 mmc 初始化的时候就会对 WiFi 模组就行 scan card 的操作。

一种是依赖 rfkill 的外部 card 的处理方式。这里需要在 AP 的 sdio 相关的 dts 的配置中添加 non-removable 配置项添加上，

```
&sdhci {
    bus-width = <8>;
    no-sdio;
    no-sd;
    non-removable;
    max-frequency = <200000000>;
    mmc-hs400-1_8v;
    mmc-hs400-enhanced-strobe;
    full-pwr-cycle-in-suspend;
    status = "okay";
};
```

这里需要在 skw_sdio_main.c 中的 skw_sdio_io_init 中添加对 wifi 模组的上电操作逻辑

```
static int __init skw_sdio_io_init(void)
{
    struct skw_sdio_data_t *skw_sdio;
    int ret = 0;
    skw_sdio_debugfs_init();
    skw_sdio_log_level_init();

    rockchip_wifi_power(1); //rk 的上电操作逻辑
    mdelay(200);
    rockchip_wifi_set_carddetect(1);
    mdelay(200);
}
```


6.2.Allwinner 相关平台的 checklist

Allwinner 的平台都是依赖 rfkill (linux 的 rfkill 子系统提供了用于禁用系统中任何无线电发射器的通用接口)，所以在我们适配的时候一定要在相应的 AP 的 dts 将 non-removable 配置项进行关闭处理。使用

```
&mmc1 {
    vmmc-supply = <&reg_cldo3>;
    vqmmc-supply = <&reg_aldo1>;
    mmc-pwrseq = <&wifi_pwrseq>;
    bus-width = <4>;
non-removable; //删除此配置项
    status = "okay";
};
```

在 sdio 的 skw_sdio_main.c 中添加

```
static int __init skw_sdio_io_init(void)
{
    struct skw_sdio_data_t *skw_sdio;
    int ret = 0;
    skw_sdio_debugfs_init();
    skw_sdio_log_level_init();

    sunxi_wlan_set_power(1); //allwinner 的上电操作逻辑
    mdelay(200);
    sunxi_mmc_rescan_card(1);
    mdelay(200);
}
```

在 skw_sdio_main.c 中的 skw_sdio_io_exit Api 中需要添加下电逻辑

```
static void __exit skw_sdio_io_exit(void)
{
    struct skw_sdio_data_t *skw_sdio = skw_sdio_get_data();

    skw_sdio_debugfs_deinit();
    skw_sdio_stop_thread();

    if (SKW_CARD_ONLINE(skw_sdio)) {
        skw_sdio_remove_card();
    }

    sunxi_wlan_set_power(0); //allwinner 的下电逻辑
    mdelay(100);
    sunxi_mmc_rescan_card(1);
    mdelay(200);
}
```

