Android 15 Wi-Fi/BT
开发指南

版本号: 1.9
发布日期: 2024.10.9

版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.11.10	AWA0989	初始版本
1.1	2020.12.30	AWA0989	修改配置文件路径，适配 SDK-v1.2
1.2	2021.05.12	AWA0989	增加 AW869A/AW869B/AIC8800 配置说明
1.3	2021.11.29	AWA0989	新增 Android12 支持
1.4	2022.05.09	AWA1828	针对 H618 Android12 修改 dts、BoardConfig.mk、vnd_generic.txt 示例，更新 init.wireless.xxx.rc 内容
1.5	2022.11.27	AWA1828	修改 menuconfig 配置示图、dts 配置示例、方案配置目录下配置文件示例
1.6	2023.03.13	AWA0989	1. 增加 PCIE Wi-Fi 配置说明 2. 增加 Bluetooth profile 定制说明 3. 增加 Wi-Fi random mac address 配置说明 4. 增加 Connectivity Captive Portal Detection Server 配置说明
1.7	2023.12.13	AWA1828	1. 增加 GRF 模式注意事项 2. 修改硬件配置 dts 示例 3. 增加适配 aic8800d80/aw869c 的 PCM dts 配置 4. 增加 aic8800d80/aic8800dc 固件路径提示
1.8	2024.2.22	AWA1828	1. 增加 CLK 注意事项小节 2. 增加新模组适配范例章节 3. 支持部分内容的章节跳转
1.9	2024.10.9	AWA1828	1. 增加双天线配置方法 2. 增加 sdio wifi tuning 配置方法

目　录

插　图

1 概述

介绍 Wi-Fi/BT 模组配置方法，目的是让 Wi-Fi/BT 模块的开发和使用人员可以根据该文档完成一些 Wi-Fi/BT 的常规配置工作，解决常见问题。本文档将介绍 xradio、realtek、broadcom 及 unisoc 模组的配置方法。

说明

全志已经完成了所列举模组的 Android 适配，但发布 SDK 并不一定带有相关内核驱动，若无，请联系对应模组厂或全志获取相关支持。

注意

从 AndroidQ 开始，全志的硬件资源配置大部分已经由 sys_config.fex 转换为 board.dts，请确保 sys_config.fex 中不要保留重复的配置。

注意

在 GRF 开发模式下，如无特殊说明，下文的修改均指代 Vendor SDK 里的内容。

2 内核驱动配置

本章节主要说明公共驱动、各可选模块的 wifi/bt 驱动如何通过内核配置菜单来使能/禁用。

2.1 公共驱动

公共驱动所有厂家模组都必须配置。具体编译进内核或是编译成模块，需要根据系统需求定义。一般的，在支持 GSI 的平台上必须配置成模块其他平台可选。

2.1.1 sunxi-rf 驱动

此驱动为电控制接口的公共驱动，所有模组都必须配置。

图 2-1: sunxi-rf 驱动配置

2.1.2 mac 地址管理驱动

此驱动为 mac 定制地址管理驱动，所有模组都必须配置。

图 2-2: sunxi-addr 驱动配置

2.2 xradio 模组

适用于 xr819/xr829 模组

功能：Wi-Fi（station/softap/p2p）+ BT

接口类型：SDIO + UART

说明

XR819 不支持 BT，无 UART 接口，无需配置 btlpm 驱动。

以下章节以 xr829 为例进行说明。

2.2.1 Wi-Fi 驱动

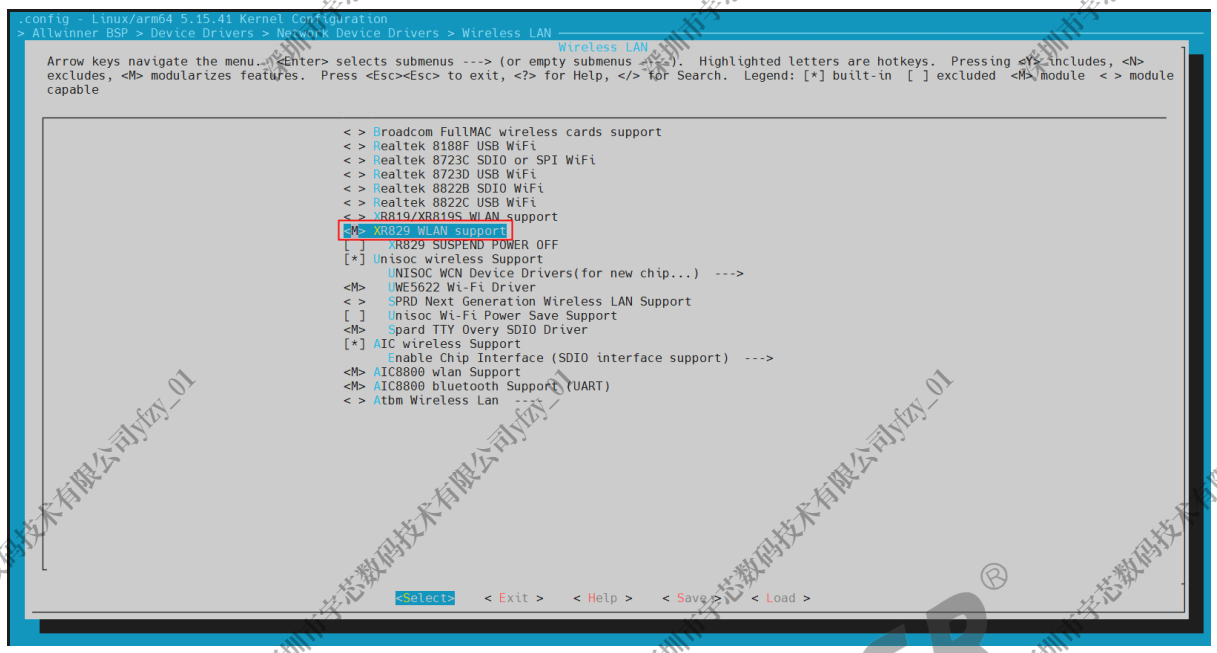

图 2-3: xradio Wi-Fi 驱动配置

2.2.2 btlpm 驱动

图 2-4: xradio btlpm 驱动配置

2.3 realtek 模组

适用于 rtl8723bs(cs)/rtl8723bs-vq0/rtl8703as-vq0

功能：Wi-Fi（station/softap/p2p）+ BT

接口类型：SDIO + UART

以下章节以 rtl8723cs 为例进行说明。

2.3.1 Wi-Fi 驱动

图 2-5: Realtek Wi-Fi 驱动配置

2.3.2 btlpm 驱动

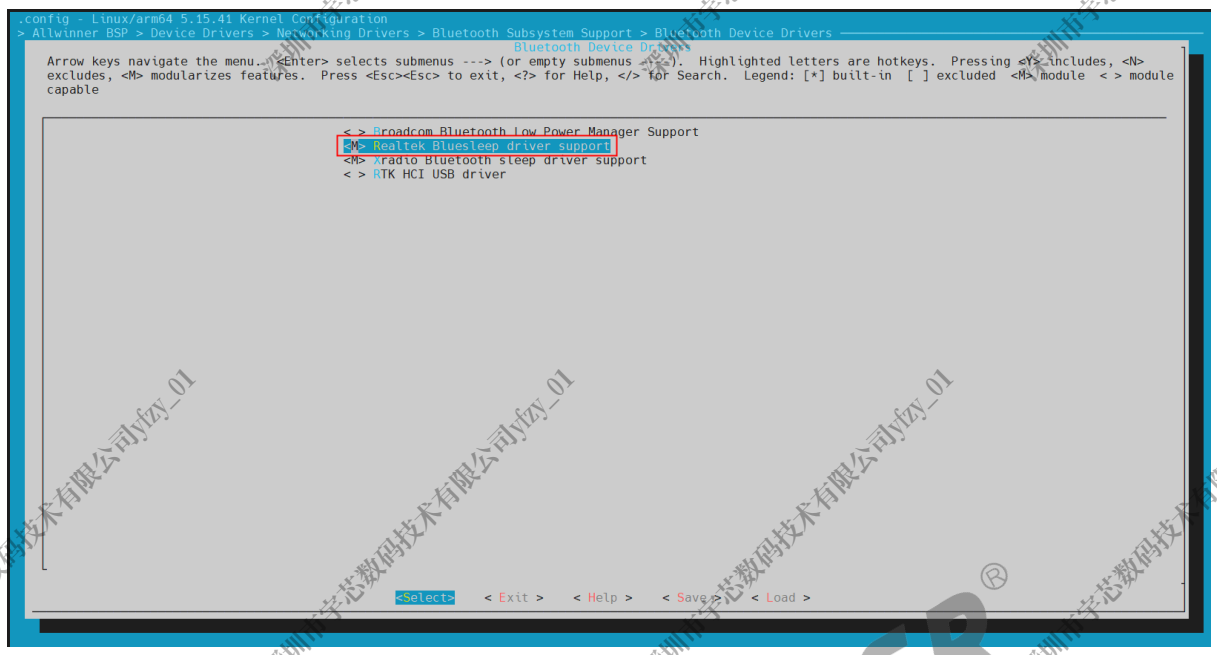

图 2-6: Realtek btlpm 驱动配置

2.4 Broadcom 模组

适用于 AP6181/AP6212/AP6255/AP6330/AP6335 等模组

功能：Wi-Fi（station/softap/p2p）+ BT

接口类型：SDIO + UART

以下章节以 AP6330 为例进行说明。

2.4.1 Wi-Fi 驱动

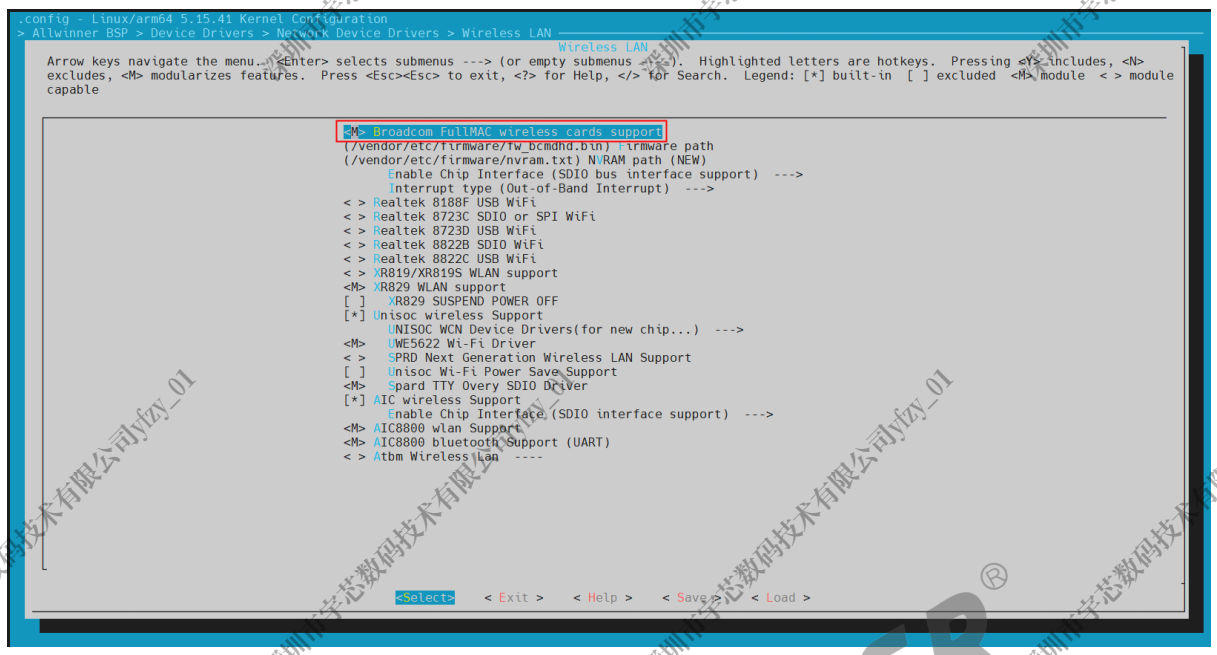

图 2-7: Broadcom Wi-Fi 驱动配置

2.4.2 btlpm 驱动

图 2-8: Broadcom btlpm 驱动配置

2.5 unisoc 模组

适用于 AW859A/UWE5622

功能：Wi-Fi（station/softap/p2p）+ BT

接口类型：SDIO

说明

两款模组使用相同的驱动和配置文件。

2.5.1 Wi-Fi BSP/Wi-Fi/BT 驱动

图 2-9: SPRD Wi-Fi 驱动配置

说明

只有打开 Spreadtrum wireless Support 时才会显示其下面的选项。需要将 UWE5622 Wi-Fi Driver 及 Sprd TTY Overy SDIO Driver 选中编译为模块，其他选项保持默认即可。

2.6 aic 模组

适用于 AW869A/AW869B/AIC8800

功能：Wi-Fi（station/softap/p2p）+ BT

接口类型：SDIO + UART

说明

几款模组使用相同的驱动和配置文件。

2.6.1 Wi-Fi BSP/Wi-Fi/BT 驱动

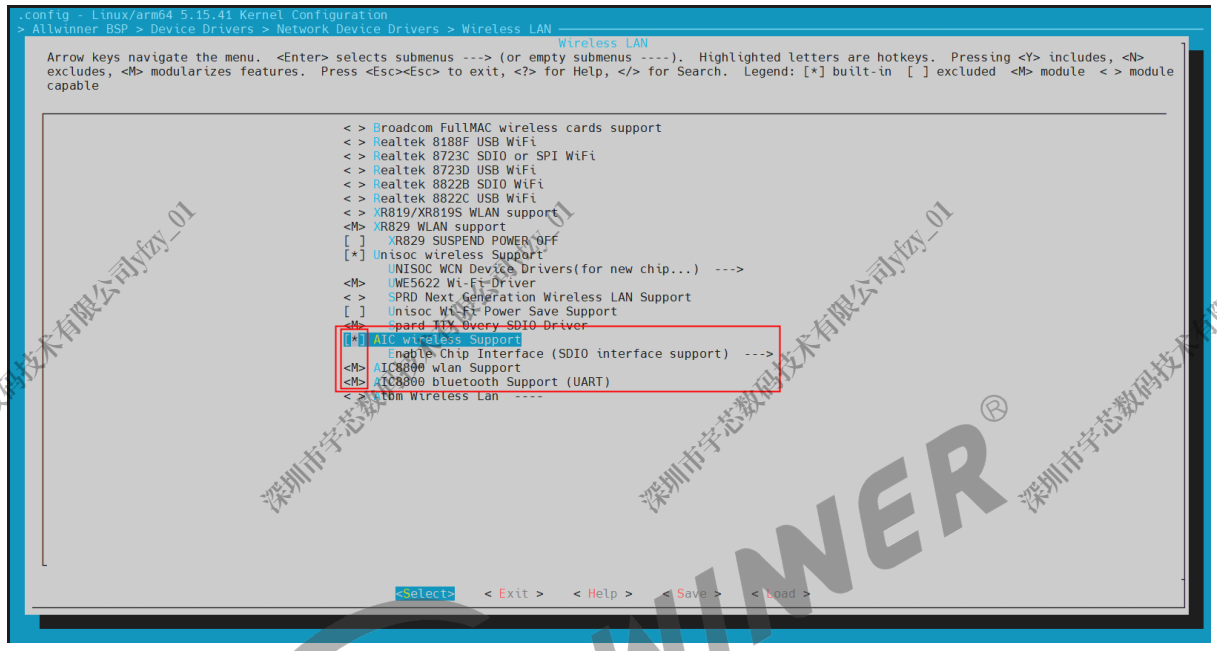

图 2-10: AIC Wi-Fi 驱动配置

说明

只有打开 AIC wireless Support 时才会显示其下面的选项。需要将 AIC8800 wlan Support 及 AIC8800 bluetooth Support 选中编译为模块。

3 硬件资源配置

本章节旨在给出设备树文件的相关配置参考，以及指出需要注意的硬件配置细节。文件路径：longan/device/config/chips/{IC}/configs/{BOARD}/linux-{LINUX-VERSON}/board.dts

3.1 RF-KILL

RF-KILL 上电控制相关参考配置如下:

```
&rfkill {
    compatible = "allwinner,sunxi-rfkill";
    chip_en  = <&r_pio PM 5 GPIO_ACTIVE_HIGH>;
    power_en = <&r_pio PL 7 GPIO_ACTIVE_HIGH>;
    pinctrl-0;
    pinctrl-names;
    status = "okay";

    /* wlan session */
    wlan {
        compatible   = "allwinner,sunxi-wlan";
        clocks;
        clock-names;
        wlan_power   = "axp2202-aldo3", "axp2202-bldo1", "axp2202-bldo2"; /* vcc-pl/vcc-pg/vcc-pm */
        wlan_power_vol= <3300000>, <1800000>, <1800000>;
        wlan_busnum  = <0x1>;
        wlan_regon   = <&r_pio PM 1 GPIO_ACTIVE_HIGH>;
        wlan_hostwake = <&r_pio PM 0 GPIO_ACTIVE_HIGH>;
        wakeup-source;
    };

    /* bt session */
    bt {
        compatible   = "allwinner,sunxi-bt";
        clocks;
        clock-names;
        bt_power   = "axp2202-aldo3", "axp2202-bldo1", "axp2202-bldo2"; /* vcc-pl/vcc-pg/vcc-pm */
        bt_power_vol= <3300000>, <1800000>, <1800000>;
        bt_rst_n     = <&r_pio PM 2 GPIO_ACTIVE_LOW>;
    };
};
```

表 3-1: RF-KILL 配置项说明

属性路径	配置项	值含义
rfkill	chip_en	模组使能引脚，硬件未使用时则无需配置
rfkill	power_en	模组外部的电源开关控制引脚，硬件未使用则无需配置

属性路径	配置项	值含义
rfkill	pinctrl-0	SOC 特殊引脚配置，如 DCXO pin 复用等，由平台定义
rfkill	pinctrl-names	同上
rfkill	status	表示使用 rfkill 驱动
rfkill/wlan	compatible	固定值，请勿修改
rfkill/wlan	clocks	Wi-Fi 使用的 SOC DCXO 时钟配置，如不使用，则无需配置
rfkill/wlan	clock-names	Wi-Fi clocks 相对应的时钟名字，仅为标识用，可不配置
rfkill/wlan	wlan_power	Wi-Fi 供电的 regulator 名称，可支持多个电配置
rfkill/wlan	wlan_power_vol	Wi-Fi wlan_power 对应电压，uV，留空则为 uboot 默认电压
rfkill/wlan	wlan_busnum	Wi-Fi 所使用的 SDIO 控制器/USB 控制器号
rfkill/wlan	wlan_regon	Wi-Fi 模组 power on 控制引脚
rfkill/wlan	wlan_hostwake	Wi-Fi 唤醒主控的 GPIO
rfkill/wlan	wakeup-source	Wi-Fi 唤醒主控功能有效
rfkill/wlan	regulator-boot-on	PCIE 模组需要设置该项，其他接口无需设置
rfkill/wlan	regulator-always-on	暂未使用
rfkill/bt	compatible	固定值，请勿修改
rfkill/bt	clocks	BT 使用的 SOC DCXO 时钟配置，如不使用，则无需配置
rfkill/bt	clock-names	BT clocks 相对应的时钟名字，仅为标识用，可不配置
rfkill/bt	bt_power	BT 供电的 regulator 名称，可支持多个电配置
rfkill/bt	bt_power_vol	BT bt_power 对应电压，uV，留空则为 uboot 默认电压
rfkill/bt	bt_rst_n	BT 对应的 reset 控制引脚
rfkill/bt	regulator-boot-on	PCIE 模组需要设置该项，其他接口无需设置
rfkill/bt	regulator-always-on	暂未使用

说明

对于无 BT 的模组，rfkill/bt 部分可不配置

注意

以上所有项必须参看原理图进行配置，配置与原理图实际使用的资源保持一致。

3.1.1 CLK 注意事项

模组工作需要时钟，一般有：模组内部自带时钟源、模组外接晶振、模组外接 soc 时钟信号三种方式，可通过原理图中 clk 的走线来判断。

其中，模组外接 soc 时钟信号是需要在 dts 中配置 clk 资源的（一般如 xr819、xr829、broadcom 系列会需要在外部接上 soc 时钟信号，或接上外接晶振）。以 xr829 外接 soc 时钟信号为例，其 dts clk 配置方式参考如下：

```
&rfkill {
    compatible = "allwinner,sunxi-rfkill";
    chip_en;
    power_en;
    pinctrl-0;
    pinctrl-names;
    status = "okay";

    /* wlan session */
    wlan {
      compatible   = "allwinner,sunxi-wlan";
      clocks       = <&rtc_ccu CLK_DCXO24M_OUT>, <&rtc_ccu CLK_OSC32K_OUT>;
      clock-names  = "dcxo24M-out", "osc32k-out";
      ...
    };

    /* bt session */
    bt {
      compatible   = "allwinner,sunxi-bt";
      clocks       = <&rtc_ccu CLK_DCXO24M_OUT>, <&rtc_ccu CLK_OSC32K_OUT>;
      clock-names  = "dcxo24M-out", "osc32k-out";
      ...
    };
  };
```

上述配置中，32K 为 LSC，这个时钟使用于模组的电源管理工作；24M 为 HSC，主业务依赖于这个时钟。注意，该示例仅供参考，每个模组对 LSC & HSC 的频率、LSC 是否外接、HSC 是否外接都有不同的要求，需按照实际情况来进行配置。

一般来说，比起外接 soc 时钟信号，使用外接晶振对于 HSC 来说可以提高时钟的精准度、减少干扰、对于 LSC 则可以提高模组的稳定性，但相对而言成本也会更高。

3.2 BT-LPM 部分

LPM 参考配置如下：

```
&btlpm {
  compatible  = "allwinner,sunxi-btlpm";
  uart_index  = <0x1>;
  bt_wake     = <&r_pio PM 3 GPIO_ACTIVE_HIGH>;
  bt_hostwake = <&r_pio PM 4 GPIO_ACTIVE_HIGH>;
  wakeup-source;
  status      = "okay";
};
```

表 3-2: BT-LPM 配置项说明

属性路径	配置项	值含义
btlpm	compatible	固定值，请勿修改
btlpm	uart_index	表示 BT 所使用的 UART 编号
btlpm	bt_wake	表示主控唤醒 BT 的 GPIO

属性路径	配置项	值含义
btlpm	bt_hostwake	表示 BT 唤醒主控的 GPIO
btlpm	wakeup-source	表示使能 BT 唤醒主动功能
btlpm	status	表示使用 btlpm 驱动

说明

对于无 BT 的模组，btlpm 部分可不配置

⚠ 注意

1. 请正确配置 bt_wake/bt_hostwake 的 ACTIVE。
2. 以上所有项必须参看原理图进行配置，配置与原理图实际使用的资源保持一致。

3.3 其他注意事项

3.3.1 控制引脚及 Wi-Fi POWER/IO 供电

Wi-Fi 5 (80211ac) 模组，如 AP6255/AW859A/UWE5622 IO 电压为 1.8V，AP 侧其控制引脚电压也应该为 1.8V。上面示例配置中控制引脚为 PG 口，请确保 PG 电压为 1.8V。

请确保 Wi-Fi VCC-WIFI/VCC-WIFI-IO 电压配置正确，否则有损坏模组的风险，如 AW859A/UWE5622 电源电压应配置为 3.3V，IO 电压应配置为 1.8V。请根据原理图电路连接关系在 dts 中正确配置 Wi-Fi POWER/IO 电压。为了安全，建议 uboot 阶段即配置正确的供电电压及 IO 电压。

uboot dts 路径为：longan/brandy/brandy-2.0/u-boot-2018/arch/arm/dts/{IC}-{BOARD}-board.dts。设置电压的典型配置如下：

```
&power_sply {
    aldo3_vol = <1001800>;
    dldo1_vol = <1800>;
};
```

⚠ 注意

特别注意的是，全志平台 PL 口同时用于 PMU I2C，当改变 PL 默认 IO 电压时，某些平台（如 A133）需要额外的配置以保证在电压切换前后 PMU 能正常工作。这些平台通常需要在 uboot dts 中添加如下额外的配置：

```
&gpio_bias {
    device_type = "gpio_bias";
    pl_bias = <1800>;
    pl_supply = "aldo3_vol";
};

&power_delay {
```

```
    device_type = "power_delay";
    aldo3_vol_delay = <50000>;
};
```

3.3.2 SDIO 配置

Wi-Fi 5 (80211ac) 模组，如 AP6255/AW859A/UWE5622 使用 SDIO 3.0，请确保 SDIO 引脚供电配置为 1.8V，同时，dts 中所使用的 SDC 控制器中应包含如下 SDIO 3.0 特有配置：

```
&pio {
    vcc-pg-supply = <&reg_pio1_8>;
};

&sdc1 {
    bus-width = <4>;
    no-mmc;
    no-sd;
    cap-sd-highspeed;
    /*sd-uhs-sdr12*/
    sd-uhs-sdr25;
    sd-uhs-sdr50;
    sd-uhs-ddr50;
    sd-uhs-sdr104;
    /*sunxi-power-save-mode;*/
    sunxi-dis-signal-vol-sw;
    cap-sdio-irq;
    keep-power-in-suspend;
    ignore-pm-notify;
    max-frequency = <208000000>;
    sunxi-dly-208M  = <0 0 0 0 0 0>;
    ctl-spec-caps = <0x428>;
    status = "okay";
};
```

如果使用的是 aic8800 模组，sdc 节点中，还需增加如下配置：

```
978
979 &sdc1 {
980 ›   bus-width = <4>;
981 ›   no-mmc;
982 ›   no-sd;
983 ›   cap-sd-highspeed;
984 ›   /*sd-uhs-sdr12*/
985 ›   sd-uhs-sdr25;
986 ›   sd-uhs-sdr50;
987 ›   sd-uhs-ddr50;
988 ›   sd-uhs-sdr104;
989 ›   /*sunxi-power-save-mode;*/
990 ›   sunxi-dis-signal-vol-sw;
991 ›   cap-sdio-irq;
992 ›   keep-power-in-suspend;
993 ›   ignore-pm-notify;
994 ›   /*
995 ›    * For AIC Wi-Fi SDIO3.0,
996 ›    * when max-frequency = <208000000>, set sunxi-dly-208M = <0 0 0 0 0 0>
997 ›    * when max-frequency = <150000000>, set sunxi-dly-208M = <1 1 0 0 1 1>
998 ›    */
999 ›   max-frequency = <208000000>;
1000 ›  sunxi-dly-208M  = <0 0 0 0 0 0>;
1001 ›  ctl-spec-caps = <0x428>;
1002 ›  status = "okay";
1003 };
1004
```

图 3-1: aic8800_sdio_ 特殊配置

另外需要注意，soc 如 A523、A733 支持 soc read 方向的 tuning。对于使用 sdio 协议的 wifi，强烈建议开启 tuning，配置方式为：

在 dtsi 文件的 sdc 节点中，加入 execute_tuning_in_kernel 配置，示例如下：

```
sdc1: sdmmc@4021000 {
  compatible = "allwinner,sunxi-mmc-v5p3x";
  device_type = "sdc1";

  ... // 部分省略

  /*sunxi-dly-104M  = <0xff 0xff 0xff 0xff 0xff 0xff>;*/
  /*sunxi-dly-208M  = <0xff 0xff 0xff 0xff 0xff 0xff> ;*/
  /*sunxi-dly-104M-ddr  = <0xff 0xff 0xff 0xff 0xff 0xff>;*/
  /*sunxi-dly-208M-ddr  = <0xff 0xff 0xff 0xff 0xff 0xff>;*/
  sunxi-dly-208M  = <0xff 0x1 0xff 0xff 0xff 0xff>;
  execute_tuning_in_kernel;
  ctl-spec-caps = <0x428>;
  status = "okay";
};
```

3.3.3 PCM 配置

在 AIC8800D80/AW869C 中，以 PCM 的方式支持蓝牙通话，i2s1_mach 配置参考如下：

```
&i2s1_mach {
  soundcard-mach,format     = "dsp_a";
  soundcard-mach,frame-master = <&i2s1_cpu>;
  soundcard-mach,bitclock-master  = <&i2s1_cpu>;
```

```
    /* soundcard-mach,frame-inversion; */
    soundcard-mach,bitclock-inversion;
    soundcard-mach,slot-num    = <2>;
    soundcard-mach,slot-width  = <16>;
    status     = "okay";
    i2s1_cpu: soundcard-mach,cpu {
      sound-dai = <&i2s1_plat>;
      /* note: pll freq = 24.576M or 22.5792M * pll-fs */
      soundcard-mach,pll-fs   = <1>;
      /* note:
       * mclk freq = mclk-fs * 12.288M or 11.2896M    (when mclk-fp ture)
       * mclk freq = mclk-fs * pcm rate      (when mclk-fp false)
       */
      soundcard-mach,mclk-fp;
      soundcard-mach,mclk-fs  = <1>;
    };
    i2s1_codec: soundcard-mach,codec {
      soundcard-mach,pll-fs   = <1>;
    };
};
```

说明

每一项配置的具体释义说明，可参考发布文档《Linux_Audio_ 开发指南》。

4 Android配置

本章节将展示方案配置、公共配置等目录下的一些可选配置项，可用以开关 wifi/bt 的功能特性或设置属性值。

4.1 BoardConfig.mk

文件路径：android/device/softwinner/{DEVICE}/{PRODUCT}

说明

BoardConfig.mk 文件决定 android 要加载哪一款 Wi-Fi 模组，以及是否支持蓝牙。

以 AIC8800 为例，典型的配置如下：

```
# wifi and bt configuration
# 1. Wifi Configuration
BOARD_WIFI_VENDOR := aic
BOARD_USR_WIFI   := aic8800
WIFI_DRIVER_MODULE_PATH := "/vendor/lib/modules/aic8800_fdrv.ko"
WIFI_DRIVER_MODULE_NAME := "aic880_fdrv"
WIFI_DRIVER_MODULE_ARG  := ""

# 2. Bluetooth Configuration
BOARD_BLUETOOTH_VENDOR   := aic
BOARD_HAVE_BLUETOOTH_NAME := aic8800
```

如果不规定某一款特定模组，使用模组自动识别的配置如下：

```
# wifi and bt configuration
# 1. Wifi Configuration
BOARD_WIFI_VENDOR := common
BOARD_USR_WIFI   := common
WIFI_DRIVER_MODULE_PATH :=
WIFI_DRIVER_MODULE_NAME :=
WIFI_DRIVER_MODULE_ARG  :=

# 2. Bluetooth Configuration
BOARD_BLUETOOTH_VENDOR   := common
BOARD_HAVE_BLUETOOTH_NAME :=
```

表 4-1: Android Wi-Fi/BT 配置项说明

配置项	值含义	注意事项
BOARD_WIFI_VENDOR	Wi-Fi 模组厂商	-
BOARD_USR_WIFI	Wi-Fi 模组型号	-
WIFI_DRIVER_MODULE_PATH	Wi-Fi 驱动 ko 的路径	-

配置项	值含义	注意事项
WIFI_DRIVER_MODULE_NAME	Wi-Fi 驱动名称	加载驱动后 lsmod 的名字
WIFI_DRIVER_MODULE_ARG	Wi-Fi 驱动加载所需参数	-
BOARD_BLUETOOTH_VENDOR	蓝牙模组厂商	-
BOARD_HAVE_BLUETOOTH_NAME	蓝牙模组型号	非必须项

说明

1. **BOARD_WIFI_VENDOR/BOARD_BLUETOOTH_VENDOR 目前支持 xradio/realtek/broadcom/sprd/common，其他值则认为是禁用 Wi-Fi/BT。**
2. **BOARD_WIFI_VENDOR/BOARD_BLUETOOTH_VENDOR 配置为 common 时表示使用模组自动识别，此时其余项请留空不配。**
3. **模组自动识别原理及配置方法请参考模块自动识别部分流程简介。**

4.2 device-common.mk

文件路径：android/device/softwinner/{DEVICE}/{PRODUCT}

说明

本文件用户仅需关注 ro.hardware.bt.ant 属性，剩余其他配置与无线模组无关。首先需要注意，该属性仅对 aic 的模组生效，通常情况下是默认不配置的，此时表示设备为单天线方案，wifi & bt 共用同一根天线。如果像下文一样，将属性配置为 2，则表示使用双天线的方案，新增蓝牙天线工作在 bt only 模式。

```
# set aic bluetooth to bt_only ant mode
PRODUCT_VENDOR_PROPERTIES += ro.hardware.bt.ant=2
```

4.3 bt_vendor.conf

文件路径：android/device/softwinner/{DEVICE}/common/wireless/bluetooth

说明

此文件仅 xradio、broadcon 使用，一般无需修改。

4.3.1 xradio

```
# UART hci commnication bandrate
Uartbandrate=1500000
```

4.3.2 broadcom

```
# UART device port where Bluetooth controller is attached
UartPort = /dev/ttyAS1

# Firmware patch file location
FwPatchFilePath = /vendor/etc/firmware/

# Firmware name
# Do not specify FwPatchFileName = xxx.hcd to enable FwAutoDetection
# FwPatchFileName = bcm43438a0.hcd
```

4.4 配置 bdroid_buildcfg.h

文件路径：android/device/softwinner/{DEVICE}/common/wireless/bluetooth/bdroid_buildcfg.h

说明

本文件主要配置 COD、Stack 编译宏等，一般不需要修改。

说明

Android13 及后续版本弃用，仅针对 12 及之前版本。

```
/*
 * Copyright (C) 2012 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file  except in  compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http:// www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,  software
 * distributed  under the License is distributed  on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for  the specific  language governing permissions and
 * limitations  under the License.
 */

#ifndef _BDROID_BUILDCFG_H
#define _BDROID_BUILDCFG_H

/*
#define BTM_DEF_LOCAL_NAME        "Generic Bluetooth"
 */

/*
 * Default class of  device
 * {SERVICE_CLASS, MAJOR_CLASS, MINOR_CLASS}
 * SERVICE_CLASS: 0x1A (Bit17 - Networking, Bit19 - Capturing, Bit20 - Object Transfer)
 * MAJOR_CLASS  : COMPUTER
 * MINOR_CLASS  : DIGITIZING_TABLET
 */
```

```
#define BTA_DM_COD                  {0x1A, BTM_COD_MAJOR_COMPUTER, BTM_COD_MINOR_DIGITIZING_TABLET}

#define BTA_GATT_DEBUG              FALSE

#define PORT_RX_BUF_LOW_WM          (10)
#define PORT_RX_BUF_HIGH_WM         (40)
#define PORT_RX_BUF_CRITICAL_WM     (45)

#define BTM_BLE_SCAN_SLOW_INT_1     (144)
#define BTM_BLE_SCAN_SLOW_WIN_1     (16)
#define BTM_MAX_VSE_CALLBACKS       (6)

#define BTM_BLE_CONN_INT_MIN_DEF    0x06
#define BTM_BLE_CONN_INT_MAX_DEF    0x0C
#define BTM_BLE_CONN_TIMEOUT_DEF    200

#define BLE_VND_INCLUDED            TRUE

#define BTA_DISABLE_DELAY           1000 /* in milliseconds */

/* avdtp log define*/
#define AVDT_DEBUG                  FALSE

/*A2DP SINK ENABLE*/
#define BTA_AV_SINK_INCLUDED        FALSE
#define BLE_LOCAL_PRIVACY_ENABLED   TRUE
/*page timeout */
#define BTA_DM_PAGE_TIMEOUT         8192
#define BTM_LOCAL_IO_CAPS_BLE       BTM_IO_CAP_KBDISP

#endif
```

4.5 配置 vnd_generic.txt

文件路径：android/device/softwinner/{DEVICE}/common/wireless/bluetooth

说明

1. **此文件仅 xradio、broadcom、AIC 使用**
2. **本文件为蓝牙编译期间的配置文件，配置硬件通信接口、波特率、Firmware 路径、LPM/DEBUG 状态、PCM 接口等。除硬件通信接口及波特率外，一般不需要修改。如需支持 hfp 功能，请按 AP 端 PCM 接口参数配置 SCO 参数**

```
#Set baudrate to 1500000
UART_TARGET_BAUD_RATE = 1500000
BLUETOOTH_UART_DEVICE_PORT = "/dev/ttyAS1"
FW_PATCHFILE_LOCATION = "/vendor/etc/firmware/"
LPM_SLEEP_MODE = TRUE
BT_WAKE_VIA_PROC = TRUE
USE_CONTROLLER_BDADDR = FALSE
BTVND_DBG = TRUE
BTCMD_DBG = TRUE
BTHW_DBG = TRUE
UPIO_DBG = TRUE
VNDUSERIAL_DBG = TRUE
#SCO_PCM_ROUTING = 0x00
#SCO_PCM_IF_CLOCK_RATE = 0x04
#SCO_PCM_IF_FRAME_TYPE = 0x00
```

```
#SCO_PCM_IF_SYNC_MODE  = 0x00
#SCO_PCM_IF_CLOCK_MODE  = 0x00
#PCM_DATA_FMT_SHIFT_MODE = 0x00
#PCM_DATA_FMT_FILL_BITS = 0x00
#PCM_DATA_FMT_FILL_METHOD = 0x00
#PCM_DATA_FMT_FILL_NUM = 0x03
#PCM_DATA_FMT_JUSTIFY_MODE = 0x0
```

4.6 配置 rtkbt.conf

文件路径：android/device/softwinner/{DEVICE}/common/wireless/bluetooth/rtkbt.conf

说明

1. **此文件仅 realtek 使用**
2. **本文件为蓝牙配置文件，配置蓝牙名称、硬件通信接口路径、COD 等。除硬件通信接口，一般不需要修改。**

```
# RELEASE NAME: 20171107_BT_ANDROID_8.x
# Bluetooth Device Name; NULL or comment means "ro.product.model"
# Name=Realtek Bluetooth

# Indicate USB or UART driver bluetooth
# For usb device:
# BtDeviceNode=/dev/rtk_btusb
# For uart device:
BtDeviceNode=/dev/ttyAS1

# Device Class
DevClassServiceClass=0x1A
DevClassMajorClass=0x01
DevClassMinorClass=0x1C

# Enable BtSnoop logging function
# valid value : true, false
RtkBtsnoopDump=false

# BtSnoop log output file
BtSnoopFileName=/data/misc/bluedroid/btsnoop_hci.cfa

# Preserve existing BtSnoop log before overwriting
BtSnoopSaveLog=true

#bit0 = 1,donnot show heartbeat packet in btsnoop
RtkbtLogFilter=1

# configuration for uart card to save HCI log for slave
H5LogOutput=0

# Enable Coex log
BtCoexLogOutput=0

# Enable net btsnoop Dump
RtkBtsnoopNetDump=false

# Enable auto restart bt
RtkBtAutoRestart=true
```

4.7 Bluetooth profile 配置

4.7.1 Android12 及之前

请在方案目录下使用 overlay 机制覆盖 packages/apps/Bluetooth/res/values/config.xml 资源文件。

4.7.2 Android13 及之后

请修改 device/softwinner/common/product.prop 文件中的相关 profile 选项。

4.8 Wi-Fi random mac address 配置

Android10 及之后，为保护用户隐私，防止联网的设备被 AP 记录和追踪，Android 实现了 Wi-Fi random mac address 功能。

Android12/13 上 AOSP 代码该功能默认是关闭的，如需开启，请在方案下使用 overlay 机制覆盖 vendor/aw/public/package/rro/WifiOverlay/res/values/config.xml 资源文件。

4.9 Connectivity Captive Portal Detection Server 配置

设备联网后，需要检测该网络是否需要登录，及判断该网络是否具有公共网络访问能力。此机制叫 Captive Portal Detection。该机制使用一个 204 服务器来实现，当通过 http request 向该服务器发送请求时，若返回空页面或 204 的状态码，则表明该网络是具有公共网络访问能力的；如果检测到 head 中的跳转标识时，表明该网络需要登录，会跳转到新页面中进行登录认证。

Android12/13 上 AOSP 代码默认使用了 http://www.google.com/generate_204、https://www.google.com/generate_204 这两个服务器地址，在国内该地址基本时不可访问的，从 UI 上来看，会提示该网络无访问，并显示带 x 或无信号强度的图标。为减轻该问题带来的负面影响，Allwinner SDK 默认修改该地址为：http://www.google.cn/generate_204、https://www.google.cn/generate_204。

针对 Android12/13，如果客户有自己搭建该 204 服务器，请在方案下使用 overlay 机制覆盖 vendor/aw/public/package/rro/ NetworkStackConfig/res/values/config.xml 以实现对 204 服务器地址的定制。

4.10 Firmware 路径

表 4-2: 各厂商模组 Firmware 路径

厂商	模块	Firmware 路径
xradio	Wi-Fi	android/hardware/xradio/wlan/kernel-firmware
xradio	BT	android/hardware/xradio/bt/firmware
realtek	Wi-Fi	No need
realtek	BT	android/hardware/realtek/bluetooth/firmware
broadcom	Wi-Fi	android/hardware/aw/wireless/parter/ampak/firmware/*.bin
broadcom	BT	android/hardware/aw/wireless/parter/ampak/firmware/*.hcd
unisoc	Wi-Fi	android/hardware/sprd/wlan/firmware/uwe5622/wcnmodem.bin
unisoc	BT	android/hardware/sprd/wlan/firmware/uwe5622/wcnmodem.bin
aic	Wi-Fi	android/hardware/aic/wlan/firmware/aic8800/*.bin
aic	BT	android/hardware/aic/wlan/firmware/aic8800/*.bin

说明

unisoc Wi-Fi/BT 功能集成到一个 Firmware 中。

说明

aic8800dc/aic8800d80 Wi-Fi/BT 的 Firmware 在原 aic 固件目录的下一级同名子目录中。如 android/hardware/aic/wlan/firmware/aic8800/aic8800d80/

4.11 其他公共配置文件

此部分由 Allwinner 整合，一般无需修改，只需要确认存在即可。

4.11.1 initrc 文件

当 BoardConfig.mk 指定具体模组时，使用 android/device/softwinner/common/config/wireless/initrc 下的 initrc 文件；当 BoardConfig.mk 配置为自动识别时，使用 android/hardware/aw/wireless/xx/config 下的文件。

4.11.1.1 Wi-Fi initrc 文件

创建必要的目录，设置其正确的访问权限，注册 wpa_supplicant 服务。

android/device/softwinner/common/config/wireless/initrc/init.wireless.wlan.rc：

```
on post-fs-data
    insmod /vendor/lib/modules/sunxi_rfkill.ko
    # Create the directories used by the Wireless subsystem
    mkdir /data/vendor/wifi 0771 wifi wifi
    mkdir /data/vendor/wifi/wpa 0770 wifi wifi
    mkdir /data/vendor/wifi/wpa/sockets 0770 wifi wifi

# broadcom/realtek/xradio wifi sta p2p concurrent service
service wpa_supplicant /vendor/bin/hw/wpa_supplicant \
    -O/data/vendor/wifi/wpa/sockets -dd \
    -g@android:wpa_wlan0
    interface aidl android.hardware.wifi.supplicant.ISupplicant/default
    socket wpa_wlan0 dgram 660 wifi wifi
    class main
    disabled
    oneshot
```

android/hardware/aw/wireless/wlan/config/init.wlan.common.rc：

```
on early-init
    insmod /vendor/lib/modules/sunxi_rfkill.ko
    # rfkill control for wifi
    chmod 0666 /sys/devices/virtual/misc/sunxi-wlan/rf-ctrl/power_state
    chmod 0666 /sys/devices/virtual/misc/sunxi-wlan/rf-ctrl/scan_device

on post-fs-data
    # Create the directories used by the Wireless subsystem
    mkdir /data/vendor/wifi 0771 wifi wifi
    mkdir /data/vendor/wifi/wpa 0770 wifi wifi
    mkdir /data/vendor/wifi/wpa/sockets 0770 wifi wifi

# broadcom/realtek/xradio wifi sta p2p concurrent service
service wpa_supplicant /vendor/bin/hw/wpa_supplicant \
    -O/data/vendor/wifi/wpa/sockets -dd \
    -g@android:wpa_wlan0
    interface aidl android.hardware.wifi.supplicant.ISupplicant/default
    socket wpa_wlan0 dgram 660 wifi wifi
    class main
    disabled
    oneshot

on property:persist.log.tag.wlan_vendor=broadcom
    setprop wifi.direct.interface p2p-dev-wlan0

on property:persist.log.tag.wlan_vendor=realtek
    setprop wifi.direct.interface p2p0

on property:persist.log.tag.wlan_vendor=xradio
    setprop wifi.direct.interface p2p-dev-wlan0

on property:persist.log.tag.wlan_vendor=sprd
    setprop wifi.direct.interface p2p-dev-wlan0

on property:persist.log.tag.wlan_vendor=aic
    setprop wifi.direct.interface p2p-dev-wlan0

on property:persist.log.tag.wlan_vendor=atbm
    setprop wifi.direct.interface p2p-dev-wlan0
```

4.11.1.2 Bluetooth initrc 文件

BT 资源和服务配置相关的文件。根据不同 vendor 加载不同模组所需要的驱动，创建必要的目录，设置其正确的访问权限。

android/device/softwinner/common/config/wireless/initrc/init.wireless.bluetooth.rc：

```
on boot
    # UART device
    chmod 0660 ${persist.vendor.bluetooth_port}
    chown bluetooth net_bt_admin ${persist.vendor.bluetooth_port}

    # bluetooth power up/down interface
    chmod 0660 /sys/class/rfkill/rfkill0/state
    chmod 0660 /sys/class/rfkill/rfkill0/type
    chown bluetooth net_bt_admin /sys/class/rfkill/rfkill0/state
    chown bluetooth net_bt_admin /sys/class/rfkill/rfkill0/type
    write /sys/class/rfkill/rfkill0/state 0

    # bluetooth MAC address programming
    chown bluetooth net_bt_admin ${ro.bt.bdaddr_path}

on property:persist.vendor.bluetooth_vendor=broadcom
    insmod /vendor/lib/modules/bcm_btlpm.ko assert_level=1
    setprop vendor.init.lpm.load 1

on property:persist.vendor.bluetooth_vendor=realtek
    insmod /vendor/lib/modules/rtl_btlpm.ko assert_level=1
    setprop vendor.init.lpm.load 1

on property:persist.vendor.bluetooth_vendor=xradio
    insmod /vendor/lib/modules/xradio_btlpm.ko assert_level=3
    setprop vendor.init.lpm.load 1

on property:persist.vendor.bluetooth_vendor=sprd
    insmod /vendor/lib/modules/uwe5622_bsp_sdio.ko
    insmod /vendor/lib/modules/sprdbt_tty.ko
    setprop vendor.init.lpm.load 1

on property:persist.vendor.bluetooth_vendor=aic
    insmod /vendor/lib/modules/aic8800_bsp.ko
    insmod /vendor/lib/modules/aic8800_btlpm.ko assert_level=3
    setprop vendor.init.lpm.load 1

on property:vendor.driver.lpm.load=1
    setprop vendor.init.lpm.load 1

on property:vendor.init.lpm.load=1
    chmod 0660 /proc/bluetooth/sleep/lpm
    chmod 0660 /proc/bluetooth/sleep/btwrite
    chmod 0660 /proc/bluetooth/sleep/btwake
    chown bluetooth net_bt_admin /proc/bluetooth/sleep/lpm
    chown bluetooth net_bt_admin /proc/bluetooth/sleep/btwrite
    chown bluetooth net_bt_admin /proc/bluetooth/sleep/btwake

    # only for sprd device
    chmod 0666 /sys/devices/platform/mtty/rfkill/rfkill1/state
    chmod 0666 /sys/devices/platform/mtty/rfkill/rfkill1/type
```

```
    chmod 0660 /dev/ttyBT0
    chown bluetooth net_bt_admin /dev/ttyBT0
    # only for aic device
    chmod 0666 /sys/devices/platform/aic-bsp/rfkill/rfkill1/state
    chmod 0666 /sys/devices/platform/aic-bsp/rfkill/rfkill1/type

on property:persist.vendor.bluetooth_vendor=realtek && property:sys.boot_completed=1
    setprop persist.vendor.bluetooth.rtkcoex true

on property:persist.vendor.bluetooth_vendor=realtek && property:sys.boot_completed=0
    setprop persist.vendor.bluetooth.rtkcoex false

on property:persist.vendor.bluetooth_vendor=xradio && property:vold.post_fs_data_done=1
    mkdir /data/vendor/bluetooth 0771 bluetooth bluetooth
    mkdir /data/vendor/bluetooth/sdd 0770 bluetooth bluetooth
```

说明

下述文件中提到的 btmode 节点，对应前面 device-common.mk 文件里的 ro.hardware.bt.ant 属性。一般单天线方案下，节点值为 -1，表示 wifi & bt 共用一根天线。如果设置了 ro.hardware.bt.ant 属性值，则该节点值与之对应。例如属性值设置为 2，则节点值也被设为 2，表示 bt only 模式。

android/hardware/aw/wireless/bluetooth/config/init.bluetooth.common.rc：

```
on boot
    # UART device
    chmod 0660 ${persist.vendor.bluetooth_port}
    chown bluetooth net_bt_admin ${persist.vendor.bluetooth_port}

    # bluetooth power up/down interface
    chmod 0660 /sys/class/rfkill/rfkill0/state
    chmod 0660 /sys/class/rfkill/rfkill0/type
    chown bluetooth net_bt_admin /sys/class/rfkill/rfkill0/state
    chown bluetooth net_bt_admin /sys/class/rfkill/rfkill0/type
    write /sys/class/rfkill/rfkill0/state 0

    # bluetooth MAC address programming
    chown bluetooth net_bt_admin ${ro.bt.bdaddr_path}

on property:persist.log.tag.bluetooth_vendor=broadcom
    insmod /vendor/lib/modules/bcm_btlpm.ko assert_level=1
    setprop vendor.init.lpm.load 1

on property:persist.log.tag.bluetooth_vendor=realtek
    insmod /vendor/lib/modules/rtl_btlpm.ko assert_level=1
    insmod /vendor/lib/modules/rtk_btusb.ko
    setprop vendor.init.lpm.load 1

on property:persist.log.tag.bluetooth_vendor=xradio
    insmod /vendor/lib/modules/xradio_btlpm.ko assert_level=3
    setprop vendor.init.lpm.load 1

on property:persist.log.tag.bluetooth_vendor=sprd
    insmod /vendor/lib/modules/uwe5622_bsp_sdio.ko
    insmod /vendor/lib/modules/sprdbt_tty.ko
    setprop vendor.init.lpm.load 1

on property:persist.log.tag.bluetooth_vendor=aic
    insmod /vendor/lib/modules/aic8800_bsp.ko
    insmod /vendor/lib/modules/aic8800_btlpm.ko assert_level=3
    insmod /vendor/lib/modules/aic8800_btusb.ko
```

```
    setprop vendor.init.lpm.load 1

on property:vendor.driver.lpm.load=1
    setprop vendor.init.lpm.load 1

on property:vendor.init.lpm.load=1
    chmod 0660 /proc/bluetooth/sleep/lpm
    chmod 0660 /proc/bluetooth/sleep/btwrite
    chmod 0660 /proc/bluetooth/sleep/btwake
    chown bluetooth net_bt_admin /proc/bluetooth/sleep/lpm
    chown bluetooth net_bt_admin /proc/bluetooth/sleep/btwrite
    chown bluetooth net_bt_admin /proc/bluetooth/sleep/btwake

    # only for sprd device
    chmod 0666 /sys/devices/platform/mtty/rfkill/rfkill1/state
    chmod 0666 /sys/devices/platform/mtty/rfkill/rfkill1/type
    chmod 0660 /dev/ttyBT0
    chown bluetooth net_bt_admin /dev/ttyBT0
    # only for aic device
    chmod 0666 /sys/devices/platform/aic-bsp/rfkill/rfkill1/state
    chmod 0666 /sys/devices/platform/aic-bsp/rfkill/rfkill1/type
    chmod 0660 /dev/aicbt_dev
    chown bluetooth net_bt_admin /dev/aicbt_dev
    # only for realtek usb device
    chmod 0660 /dev/rtkbt_dev
    chown bluetooth net_bt_admin /dev/rtkbt_dev

on property:persist.log.tag.bluetooth_vendor=realtek && property:sys.boot_completed=1
    setprop persist.vendor.bluetooth.rtkcoex true

on property:persist.log.tag.bluetooth_vendor=realtek && property:sys.boot_completed=0
    setprop persist.vendor.bluetooth.rtkcoex false

on property:persist.log.tag.bluetooth_vendor=xradio && property:vold.post_fs_data_done=1
    mkdir /data/vendor/bluetooth 0771 bluetooth bluetooth
    mkdir /data/vendor/bluetooth/sdd 0770 bluetooth bluetooth

on property:persist.log.tag.bluetooth_vendor=aic && property:vendor.init.lpm.load=1 && property:ro.hardware.bt.ant=*
    write /sys/devices/platform/aic-bsp/aicbsp_info/btmode ${ro.hardware.bt.ant}
```

4.11.2 manifest 文件

路径：android/device/softwinner/common/config/wireless/manifest，此为兼容性矩阵文件，对应 Wi-Fi/BT HIDL HAL 的配置，文件一般无需修改。

4.11.3 wireless_config.mk

路径：android/device/softwinner/common/config/wireless

本文件一般不需要修改，只需要确认存在即可。其作用是：

1. 解析 BoardConfig.mk 里面的 Wi-Fi/BT 的配置;

2. 把一些零散的 Wi-Fi/BT 配置集中管理，并能根据不同的模组厂完成相应的配置。

5 新模组适配示例

假设要支持一款名为 WExmp 的新模组，大致流程可分为以下几步。

⚠ 注意

本章节仅适用于已有相关支持的厂商的新模组。若需要适配一个全新的厂商的模组，请咨询 AW 工作人员。

5.1 方案目录配置部分

主要是针对 mk 的修改，添加新模组的名称、需要使用上的库名、编译文件的名称路径等；以及对 initrc 的修改，如在 initrc 中添加对应的 btlpm 驱动加载、修改节点权限等。示例如下：

Android/device/softwinner/common/common.mk

```
// wpa_supplicant 库添加
PRODUCT_CFI_INCLUDE_PATHS += \
+ hardware/WExmp/wlan/wpa_supplicant_8_lib \
```

Android/device/softwinner/common/config/wireless/wireless_config.mk

```
// 添加至已支持模组名单
SUPPORTED_WIFI_VENDOR     := broadcom realtek xradio sprd aic ssv WExmp common
SUPPORTED_BLUETOOTH_VENDOR := broadcom realtek xradio sprd aic WExmp common

// 如果用户指定使用 WExmp 模组，则需要以下配置来添加库名、p2p 接口名、 firmware 存放路径等
// 以下为 wifi 相关配置
ifeq ($(BOARD_WIFI_VENDOR),xxx)
...
+ else ifeq ($(BOARD_WIFI_VENDOR),WExmp)
+    BOARD_WLAN_DEVICE        := WExmp
+    BOARD_WPA_SUPPLICANT_PRIVATE_LIB := lib_driver_cmd_WExmp
+    BOARD_HOSTAPD_PRIVATE_LIB  := lib_driver_cmd_WExmp
+    BOARD_WIRELESS_PROPERTIES  += wifi.direct.interface=p2p-dev-wlan0 //（根据实际情况配置）
+    WIFI_HIDL_FEATURE_DUAL_INTERFACE := true
+    -include hardware/WExmp/wlan/firmware/$(BOARD_USR_WIFI)/device-WExmp.mk
+ endif
// 以下为 bt 相关配置
ifeq ($(BOARD_BLUETOOTH_VENDOR),xxx)
...
+ else ifeq ($(BOARD_BLUETOOTH_VENDOR),WExmp)
+    BOARD_HAVE_BLUETOOTH_WExmp := true
+    BOARD_CUSTOM_BT_CONFIG     := $(BOARD_CUSTOM_BT_CONFIG_TMP)
+    -include hardware/WExmp/libbt/firmware/$(BOARD_HAVE_BLUETOOTH_NAME)/WExmp-bt.mk
```

Android/device/softwinner/common/config/wireless/initrc/init.wireless.wlan.rc

```
// 如果有什么差异化处理可在此处添加，没有的话可以忽略
+ on property:persist.vendor.wlan_vendor=WExmp
+ 差异化处理
```

Android/device/softwinner/common/config/wireless/initrc/init.wireless.bluetooth.rc

```
// 加载 WExmp 对应的 btlpm 驱动
+ on property:persist.vendor.bluetooth_vendor=WExmp
+ insmod /vendor/lib/modules/WExmp_btlpm.ko assert_level=3 //（根据实际情况配置）
+ setprop vendor.init.lpm.load 1

// 其他差异化处理，如果节点权限等
+ 差异化处理
```

5.2 板级目录配置部分

与 Wi-Fi/BT 相关的 dts 配置节点介绍如下：

- rfkill：配置 Wi-Fi/BT IC 的供电、片选、唤醒引脚
- btlpm：配置 BT 的主从唤醒引脚
- sdc1：配置 Wi-Fi 使用的 sdio 口，如总线宽、工作模式、最大频率等

对于 dts 的配置，一般是随主控 IC 更改，适用同一板级的新模组，引脚对应，无需修改 dts。如果是未支持过的全新模组，需要和模组厂对接清楚参数，根据实际情况配置。

5.3 框架目录配置部分

具体路径在 Android/frameworks/opt/net/wifi/libwifi_hal/Android.mk，此部分上承框架，下接 HAL 层，只需填充正确的厂商名及其对应库名即可，示例如下：

```
// 在判断 LIB_WIFI_HAL 处添加下述配置
else ifeq ($(BOARD_WLAN_DEVICE), WExmp)
  # support WExmp WIFI HAL
  LIB_WIFI_HAL := libwifi-hal-WExmp
```

5.4 模块自动识别部分

5.4.1 流程简介

1. init 进程启动，读取解析特定平台的 rc 文件，初始化属性服务 property
2. 判断属性服务中是否含有 wifi 模组信息，若无，则给 wifi 模组上电，扫描 wifi 设备
3. 通过 uevent 向上层发送扫描到的模组的信息
4. 上层获取到 wifi 模组的 vid pid 信息，将其保存到 property 中
5. 后续根据 property 的值来差异化加载 wifi/bt 所需资源

5.4.2 配置方法

主要维护 Android/hardware/aw/wireless/hwinfo/hwinfo.xml，配置新模组的 id、接口类型、模组名、库名等信息，示例如下：

```
// wifi hal
<!-- Wi-Fi Hal config -->
<wifi-hal>
+ <entry vendor="WExmp" name="libwifi-hal-WExmp.so" />  //配置 wifi 相关库名

// bt hal
<!-- Bluetooth libbt-vendor config -->
<libbt>
+ <entry vendor="WExmp" name="libbt-WExmp.so" />  //配置 bt 相关库名

<!-- Preload driver module for vendor -->  // 配置驱动预加载（根据实际情况，按需填写）
<driver>
+ <entry vendor="WExmp" >
+ <preload name="WExmp_preload" />
+ </entry>

<!-- Hardware info for detection -->  // 配置模组硬件信息（根据实际情况配置）
<module>
+ <entry id="0x12345678" bus="0x1" name="WExmp" driver="WExmp" module="WExmp" vendor
="WExmp" btsup="1" />
</module>
```

其中，上述 module 信息具体释义如下：

- bus：接口类型，0 -> usb，1 -> sdio
- name：模组名
- driver：内核驱动名
- module：内核驱动名
- vendor：厂商名
- btsup：是否支持蓝牙，0 -> 不支持，1 -> 支持
- drvpreload：是否启用驱动预加载，省略 / 0 -> 不启用，1 -> 启用

5.5 HAL 部分

HAL 代码由厂商提供，在新模组支持时用户主要工作是调整编译文件。需要修改的地方如下：

Android/hardware/aw/wireless/wlan/config/init.wlan.common.rc

```
// 根据当前模组，做一些设置 p2p 接口名等差异化的操作
+ on property:persist.log.tag.wlan_vendor=WExmp
+    setprop wifi.direct.interface p2p-dev-wlan0 //（根据实际情况配置）
+    其他差异化操作
```

Android/hardware/aw/wireless/wlan/firmware/WExmp/device-WExmp.mk

```
// 主要负责固件、配置文件的拷贝
FW_BASE_PATH := hardware/WExmp/wlan/firmware // 需要与下文提到的存放厂商 HAL 代码的路径对应

ifneq (,$(wildcard $(FW_BASE_PATH)))

// 差异化操作，主要查找厂商提供的 bin 文件、配置文件等等
XXX_FILES:= $ 差异化操作
// 文件找到后，做一个拷贝，放到所需要输出的目录，如 TARGET_COPY_OUT_VENDOR/etc/firmware，根据实际情况配置
PRODUCT_COPY_FILES += $ 差异化操作

endif
```

Android/hardware/aw/wireless/wlan/wifi_hal/common/Android.mk

```
// 添加新模组的 so 文件前缀
+ LOCAL_VENDOR_LIST += hardware/WExmp/wlan/wifi_hal:libwifi-hal-WExmp
```

Android/hardware/aw/wireless/wlan/wifi_hal/vendor/WExmp/Android.mk

```
// WExmp 这个目录需要自行创建，Android.mk 也是自行添加的，用来处理 hardware/WExmp/wlan/wifi_hal 下的文件编
   译，大体框架如下
LOCAL_PATH := hardware/WExmp/wlan/wifi_hal
ifneq (,$(wildcard $(LOCAL_PATH)))
  // 差异化操作，主要是添加厂商提供的 HAL 代码、依赖的库等等
  LOCAL_MODULE := libwifi-hal-WExmp
endif
```

剩下的 wpa_supplicant 部分、bluetooth 部分结构一致，根据上述介绍的 wifi 部分的规律去逐一添加即可，不在此赘述。

Android/hardware/WExmp

这部分由厂商提供，用户确认路径正确即可。

5.6 驱动与固件

由厂商提供，通常将驱动置于 bsp目录: driver/net/wireless/ 目录下，固件置于android目录: hardware/WExmp/wlan/firmware 及 android目录: hardware/WExmp/libbt/firmware 目录下。正确放置文件后，参考本文的内核驱动配置 与硬件资源配置 章节来配置即可。

著作权声明

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明





（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。