SSV6051_Porting_Guide

Revision	Date	Author	Description
1.0	2015/11/09	Eason Li	Initial Version

深圳市南方硅谷微电子有限公司

Shenzhen South Silicon Valley Microelectronic Co. ,Ltd

目录

➢ 特别说明

- ◆ 本 Guide 仅适用于英卡 RK3126 Single-WIFI-SDK。
- ◆ 修改的地方，用红色粗体字标示。
- ◆ 修改文件，用绿色字体标示。
- ◆ 本项目的烧录和软件编译方式，请参考 RK3126 烧录以及编译。

本 Guide 带有 new/old 对比文件，如有不清楚的，使用 compare 查看。

SSV6051 Wifi Android Instructions

RK3126 使用 3.10 版本的内核，跟其他平台(3.0.x)相比有较大的变化，对于客户来说，在项目开发阶段进行内核配置时，对于 Wifi 部分建议直接使用参考的配置文”arch/arm/condigs/rockchip_rk312x_axp_defconfig”,而不要去修改其他涉及到 Wifi 的配置项，除了具体的 Wifi 模组选择以外，另外 RK3126 SDK 中融入 SSV6051 芯片驱动支持，需要修改和增加较多的文件，后面章节将作具体的分析。

1. 内核配置与修改

1.1 内核配置

Rk3126 方案带有两种不同的硬件设置，主要区别是 PMU 采用不同厂家的型号，一方案采用 PMU 为 AXP，二方案采用 PMU 为 8931。

针对两种方案，内核配置便出现两种，当然也可以手动更改配置，但为了方便使用和区分，提高生产效率，在这里单独提供两份 defconfig :

1, rockchip_rk312x_axp_6051_defconfig (PMU : AXP)

2, rockchip_rk312x_8931_6051_defconfig (PMU : 8931)

这两组文件将单独提供，存放在 new/kernel/arch/arm/configs/ 下。

针对 wifi 部分的修改，两个配置文件主要增加 SSV6051 的芯片支持:

1, CONFIG_SSV6051=m

2, CONFIG_SSV6051_P2P=y

3, CONFIG_SSV6051_SDIO=y

此配置增加需要在 kernel/drivers/net/wireless/Kconfig 中增加

source "drivers/net/wireless/rockchip_wlan/ssv6xxx/Kconfig"

针对以上路径增加 ssv6xxx 目录，创建 Kconfig , Kconfig 内容如下:

```
menu "iComm WLAN support"
config SSV6051
        tristate "ssv6051 Wireless driver"
        depends on MMC
        select CFG80211
        select MAC80211
        select NL80211_TESTMODE
        ---help---
                Enable iComm ssv6051 WLAN kernel driver.
config SSV6051_P2P
        default y
        bool "SSV6051_P2P"
        ---help---
                Enable iComm ssv6051 P2P interface.
config SSV6051_SDIO
        default y
        bool "SSV6051_SDIO"
        ---help---
                iComm ssv6051 sdio setting.
endmenu
```

1.2 内核 dts 设备树配置以及修改

刚才讨论了 rk3126 带有两种硬件方案，当然每种硬件也同时单独提出独自的 dts 来支持设备，本 dts 也同时提供两种:

1, rk3126-v2.1.dts (PMU : AXP)

2, rk3126-v1.0.dts (PMU: 8931)

针对 wifi 部分的修改，两个 dts 配置文件主要增加对 SSV6051 型号的支持，同时针对此文件，修改 wifi 开关有时无法正常打开问题。

两个文件共同修改部分:

Wireless-wlan

1, wifi_chip_type 修改为 wifi_chip_type = "ssv6051";

2, sdio 参考电压 sdio_vref 修改为 sdio_vref = <3300>

Wireless-bluetooth

- ssv6051 不支持 bt 功能，将禁止 wireless-bluetooth，将 status 修改成

Status = “disable”;

backlight: backlight

- 此部分是解决 wifi 有事在屏幕熄灭进入休眠后，唤醒屏幕再次操作 wifi，会导致功耗高，此解决方法是: A1 的控制休眠拉低，唤醒拉高。

enable-gpio = <&gpio0 GPIO_D3 GPIO_ACTIVE_LOW>;

修改成

enable-gpio = <&gpio1 GPIO_A1 GPIO_ACTIVE_HIGH>;

两个文件异同修改部分:

由于两种方案对 wifi 部分的硬件电路均匀异同

1, PMU axp 方案针对 wifi ssv6051 芯片，带有控制引脚，ldo_en，接到硬件的 axp 的 gpio3，所以需要设置为

WIFI,axp_gpio = <3>;

axp_gpio_level = <1>;

status = "okay";

2, PMU 8931 方案对 wifi ssv6051 芯片暂无控制，硬件上直接串联 RC 上拉供电，于是需要屏蔽或者删除控制部分:

// WIFI,axp_gpio = <3>;

// axp_gpio_level = <1>;

status = "okay";

1.3 内核相关驱动以及 mac80211 修改

1.3.1 内核 mmc 驱动修改

- FILE: kernel/drivers/mmv/host/rk_sdmmc.c & Kconfig

rk_sdmmc.c 中增加 command complete 时的延时处理并处理 sdio wifi 不进入 sd 卡部分代码，以免出现其他意想不到的问题。

1，在 rk_sdmmc.c 文件第 2169 行作以下修改:

```
2169
2170 #ifdef CONFIG_SSV6051_SDIO
2171         mdelay(20);
2172 #else
2173         /* newer ip versions need a delay between retries */
2174         if (host->quirks & DW_MCI_QUIRK_RETRY_DELAY)
2175             mdelay(20);
2176 #endif
```

2，在 rk_sdmmc.c 文件第 4134 和 4168 行，两处作同样的修改:

```
4168         (get_wifi_chip_type() >= WIFI_SSV6051 || get_wifi_chip_type() > WIFI_AP6XXX_SERIES))
4169         return 0;
```

3，Kconfig 中增加对 MMC_DW_IDMAC 的限制，开启此配置，会启动 internal DMAC 来处理数据，我们芯片不需要开启，开启后可能会导致死机或者重启等现象，不开启则使用 external DMA 接口来处理，这正好是我们需要的。

```
528 config MMC_DW_IDMAC
529         bool "Internal DMAC interface"
530         depends on MMC_DW && !SSV6051_SDIO
```

1.3.2 rochchip wifi system interface control

- FILE : kernel/drivers/net/wireless/rockchip_wlan/wifi_sys/rkwifi_sys_iface.c
  Kernel/include/linux/rfkill-wlan.h

本部分主要增加 ssv6051 芯片的控制，识别出 ssv6051 芯片，以便作 android 层的相关数据的处理。

1， rfkill-wlan.h 增加 ssv6051 芯片信号

```
68      WIFI_ESP8089,
69      WIFI_SSV6051,
70      TYPE_MAX,
```

在 rfkill-wlan.c 中增加型号的识别处理。

```
else if (strcmp(wifi_chip_type_string, "ssv6051") == 0) {
        type = WIFI_SSV6051;
 }
```

2， rkwifi_sys_iface.c 中增加识别到的 wifi chip，便开启其他 wifi chip 的宏开关，避免配置不当或者其他，导致编译出错等。

```
115     if(type == WIFI_ESP8089) {
116         count = sprintf(_buf, "%s", "ESP8089");
117         printk("Current WiFi chip is ESP8089.\n");
118     }
119     if(type == WIFI_SSV6051) {
120         count = sprintf(_buf, "%s", "SSV6051");
121         printk("Current WiFi chip is SSV6051.\n");
122     }
123
```

```
161 #ifdef CONFIG_RKWIFI
162     if (type < WIFI_AP6XXX_SERIES) {
163         if (enable > 0)
164             ret = rockchip_wifi_init_module_rkwifi();
165         else
166             rockchip_wifi_exit_module_rkwifi();
167         return ret;
168     }
169 #endif
170 #ifdef CONFIG_RTL_WIRELESS_SOLUTION
171     if (type < WIFI_RTL_SERIES) {
172         if (enable > 0)
173             ret = rockchip_wifi_init_module_rtkwifi();
174         else
175             rockchip_wifi_exit_module_rtkwifi();
176         return ret;
177     }
178 #endif
179 #ifdef CONFIG_ESP8089
180     if (type == WIFI_ESP8089) {
181         if (enable > 0)
182             ret = rockchip_wifi_init_module_esp8089();
183         else
184             rockchip_wifi_exit_module_esp8089();
185         return ret;
186     }
187 #endif
```

1.3.3 Add p2p0 interface in Mac80211

➢ FILE: kernel/net/mac80211/main.c

Mac80211 是 linux 系统中的一个子系统，实现了 soft-mac/half-mac wireless devices 代码共享,主要包括 MLME 和其他的 code，是驱动开发者用来写 softMac 无线设备驱动框架，softMAC 设备可以让系统更好的控制硬件，允许用软件实现帧的管理。

Mac80211 为 softMac 设备实现了 cfg80211 回调函数,mac80211 也就能依靠 cfg80211 来注册到网络子系统，并且配置设备。在 mac80211 中,MLME 在内核以 STA 模式实现，在用户空间以 AP 模式实现。

Kernel/net/mac80211/main.c 是主模块的入口点，驱动回调的入口，在 main.c 中的 ieee80211_register_hw 函数增加对 SSV6051 的 STA Mode P2P Client 功能支持。

```
//#ifdef CONFIG_ESP8089
#if defined(CONFIG_ESP8089) || defined(CONFIG_SSV6051_P2P)
    if (local->hw.wiphy->interface_modes &(BIT(NL80211_IFTYPE_P2P_GO) |
BIT(NL80211_IFTYPE_P2P_CLIENT))) {
            result = ieee80211_if_add(local, "p2p%d", NULL,
                                      NL80211_IFTYPE_STATION, NULL);
                if (result)
                    wiphy_warn(local->hw.wiphy,
                               "Failed to add default virtual iface\n");
            }

#endif
```

➢ FILE: kernel/net/mac80211/mlme.c

根据 WiFi 休眠后的连接处理，本部分对应于 kernel/net/wireless/sysfs.c 的修改。

```
void ieee80211_mgd_quiesce(struct ieee80211_sub_if_data *sdata)
{
    struct ieee80211_if_managed *ifmgd = &sdata->u.mgd;
    u8 frame_buf[IEEE80211_DEAUTH_FRAME_LEN];

    mutex_lock(&ifmgd->mtx);

    if (ifmgd->auth_data) {
        /*
```

```
            * If we are trying to authenticate while suspending, cfg80211
            * won't know and won't actually abort those attempts, thus we
            * need to do that ourselves.
            */
           ieee80211_send_deauth_disassoc(sdata,
                                          ifmgd->auth_data->bss->bssid,
                                          IEEE80211_STYPE_DEAUTH,
                                          WLAN_REASON_DEAUTH_LEAVING,
                                          false, frame_buf);
           ieee80211_destroy_auth_data(sdata, false);
           cfg80211_send_deauth(sdata->dev, frame_buf,
                                IEEE80211_DEAUTH_FRAME_LEN);
       }

       mutex_unlock(&ifmgd->mtx);
}
```

修改为：

```
void ieee80211_mgd_quiesce(struct ieee80211_sub_if_data *sdata)
{
    struct ieee80211_if_managed *ifmgd = &sdata->u.mgd;
    u8 frame_buf[IEEE80211_DEAUTH_FRAME_LEN];

    mutex_lock(&ifmgd->mtx);

    if (ifmgd->auth_data || ifmgd->assoc_data) {
        const u8 *bssid = ifmgd->auth_data ?
                                ifmgd->auth_data->bss->bssid :
                                ifmgd->assoc_data->bss->bssid;
        printk("%s: ifmgd->auth_data || ifmgd->assoc_data\n", __func__);
        /*
         * If we are trying to authenticate while suspending, cfg80211
         * won't know and won't actually abort those attempts, thus we
         * need to do that ourselves.
         */
        ieee80211_send_deauth_disassoc(sdata,
                                       bssid,
                                       IEEE80211_STYPE_DEAUTH,
```

```
                                                WLAN_REASON_DEAUTH_LEAVING,
                                                false, frame_buf);
            if (ifmgd->assoc_data)
                ieee80211_destroy_assoc_data(sdata, false);
            if (ifmgd->auth_data)
                ieee80211_destroy_auth_data(sdata, false);

            cfg80211_send_deauth(sdata->dev, frame_buf,
                                 IEEE80211_DEAUTH_FRAME_LEN);
        }

        mutex_unlock(&ifmgd->mtx);
    }
```

➢ FILE: kernel/net/wireless/sysfs.c

将休眠后的连接处理开启，默认被屏蔽。

```
        //if (!rdev->wowlan)
        //    cfg80211_leave_all(rdev);
```

修改为:

```
        if (!rdev->wowlan)
            cfg80211_leave_all(rdev);
```

1.4 Android Device Configure

➢ FILE: device/rockchip/rk312x/wifi_bt.mk

具体配置如下:

```
# iCommCooperation:
#       ssv6051              #only wifi
#
BOARD_CONNECTIVITY_VENDOR := iCommCooperation
BOARD_CONNECTIVITY_MODULE := ssv6051
```

➢ FILE: device/rockchip/rksdk/wifi_bt_common.mk

增加 ssv6051 使用的 wpa_supplicant 版本以及私有库信息

```
#BOARD_CONNECTIVITY_VENDOR := Broadcom
#BOARD_CONNECTIVITY_MODULE := ap6xxx

# Wifi related defines

ifeq ($(strip $(BOARD_CONNECTIVITY_VENDOR)), iCommCooperation)
FORCE_WIFI_WORK_AS_ANDROID4_2 := false
BOARD_WIFI_VENDOR := iCommCooperation
BOARD_WPA_SUPPLICANT_DRIVER := NL80211
WPA_SUPPLICANT_VERSION      := VER_0_8_X
BOARD_WPA_SUPPLICANT_PRIVATE_LIB := lib_driver_cmd_icomm
BOARD_HOSTAPD_DRIVER        := NL80211
BOARD_HOSTAPD_PRIVATE_LIB   := lib_driver_cmd_icomm
BOARD_WLAN_DEVICE           := ssv6051
WIFI_DRIVER_FW_PATH_STA     := ""
WIFI_DRIVER_FW_PATH_P2P     := ""
WIFI_DRIVER_FW_PATH_AP      := ""
endif
```

➢ FILE:device\rockchip\rksdk\device.mk

增加 ssv6051 设备的配置支持, 取消使用 rksdk 下的 init.connectivity.rc

增加 wpa_supplicat_icomm 和 hostapd_icomm 的支持。

```
ifneq ($(strip $(BOARD_CONNECTIVITY_VENDOR)), MediaTek_mt7601)
ifneq ($(strip $(BOARD_CONNECTIVITY_VENDOR)), MediaTek)
ifneq ($(strip $(BOARD_CONNECTIVITY_VENDOR)), RealTek)
ifneq ($(strip $(BOARD_CONNECTIVITY_VENDOR)), iCommCooperation)
#ifneq ($(strip $(BOARD_CONNECTIVITY_VENDOR)), Espressif)
PRODUCT_COPY_FILES += \
    device/rockchip/rksdk/init.connectivity.rc:root/init.connectivity.rc
#endif
endif
endif
endif
endif
```

```
include hardware/rk29/camera/Config/rk32xx_camera.mk
include hardware/rk29/camera/Config/user.mk

ifeq ($(strip $(BOARD_CONNECTIVITY_VENDOR)), iCommCooperation)
include hardware/icomm/wlan/config/config-icomm.mk
include hardware/icomm/wlan/firmware/config-icomm.mk
endif

ifeq ($(strip $(TARGET_BOARD_PLATFORM_GPU)), PVR540)
include device/rockchip/common/gpu/PVR540.mk
endif
```

```
# NTFS support
PRODUCT_PACKAGES += \
    ntfs-3g

PRODUCT_PACKAGES += \
   wpa_supplicant_icomm \
   hostapd_icomm

PRODUCT_PACKAGES += \
    com.android.future.usb.accessory
```

➢ FILE: device/rockchip/common/rk30_wifi.mk

本文件增加驱动 ko 编译时的 copy 动作，请更新 ko 时，务必替换 device/rockchip/common/lib/modules & modules_smp 目录下的 ssv6051.ko。

```
        device/rockchip/common/wifi/lib/modules/ssv6051.ko:system/lib/modules/ssv6051.ko
else
PRODUCT_COPY_FILES += \
        device/rockchip/common/wifi/lib/modules_smp/wlan.ko:system/lib/modules/wlan.ko \
        device/rockchip/common/wifi/lib/modules_smp/rkwifi.ko:system/lib/modules/rkwifi.ko \
        device/rockchip/common/wifi/lib/modules_smp/rkwifi.oob.ko:system/lib/modules/rkwifi.oob.ko \
        device/rockchip/common/wifi/lib/modules_smp/8188eu.ko:system/lib/modules/8188eu.ko \
        device/rockchip/common/wifi/lib/modules_smp/8192cu.ko:system/lib/modules/8192cu.ko \
        device/rockchip/common/wifi/lib/modules_smp/mt5931.ko:system/lib/modules/mt5931.ko \
    device/rockchip/common/wifi/lib/modules_smp/8723as.ko:system/lib/modules/8723as.ko \
        device/rockchip/common/wifi/lib/modules_smp/8723au.ko:system/lib/modules/8723au.ko \
        device/rockchip/common/wifi/lib/modules_smp/8189es.ko:system/lib/modules/8189es.ko \
        device/rockchip/common/wifi/lib/modules_smp/mt7601sta.ko:system/lib/modules/mt7601sta.ko \
        device/rockchip/common/wifi/lib/modules_smp/mt7601ap.ko:system/lib/modules/mt7601ap.ko \
        device/rockchip/common/wifi/lib/modules_smp/mtprealloc7601Usta.ko:system/lib/modules/mtprealloc7601Usta.ko \
        device/rockchip/common/wifi/lib/modules_smp/esp8089.ko:system/lib/modules/esp8089.ko \
        device/rockchip/common/wifi/lib/modules_smp/ssv6051.ko:system/lib/modules/ssv6051.ko
endif
```

1.5 External Doc Configure

Android WiFi tools ，对于 Android 这个庞大的系统，其源代码占据了 10 多 G 的地盘。例如要进行蓝牙模块，音视频开发，或者 wifi 模块移植，那么 external 目录下的源代码就很重要了，在此我们使用 wpa_supplicant 802.11x 无线网络管理工具，也是 android 系统下广泛使用的 wifi 工具，当然 wireless_tools 也是一个不错的工具，其在 wifi router 中使用广泛。

修改 wpa_supplicant_8,增加对 SSV6051 的 WiFi P2P 网络接口功能支持。

➢ FILE: external/wap_supplicant_8/wpa_supplicant/Android.mk
externale/wap_supplicant_8/ hostapd/Android.mk

```
ifeq ($(BOARD_WLAN_DEVICE), mrvl)
L_CFLAGS += -DANDROID_P2P
endif
ifeq ($(BOARD_WIFI_VENDOR), Espressif)
L_CFLAGS += -DANDROID_P2P
L_CFLAGS += -DWIFI_EAGLE
endif

ifeq ($(BOARD_WIFI_VENDOR),iCommCooperation)
L_CFLAGS += -DANDROID_P2P
L_CFLAGS += -DWIFI_EAGLE
endif
```

在 wpa_supplicant/Android.mk 下，修改 LOCAL_MODULE

LOCAL_MODULE := wpa_supplicant_icomm

在 hostapd/Android.mk 下，修改 LOCAL_MODULE

LOCAT_MODULE := hostapd_icomm

- FILE: external/sepolicy/ file_contexts

由 selinux 影响，在 netd 进程中只能启动 hostapd，如果要启动 hostapd_icomm 等需要修改 sepolicy 修改相关安全策略文件,增加 hostapd_icomm 支持。

```
/system/bin/hostapd_icomm      u:object_r:hostapd_icomm_exec:s0
```

- FILE: external/sepolicy/hostapd_icomm.te

创建 hostapd_icomm.te 文件，输入以下内容

```
type hostapd_icomm, domain;
permissive hostapd_icomm;
type hostapd_icomm_exec, exec_type, file_type;

init_daemon_domain(hostapd_icomm)
net_domain(hostapd_icomm)
unconfined_domain(hostapd_icomm)
```

- FILE: external/sepolicy/netd.te

在第 48 行后，增加以下内容

```
domain_auto_trans(netd, hostapd_icomm_exec, hostapd_icomm)
allow netd hostapd_icomm:process signal;
```

1.6 Hareware Doc Modify

1.6.1 增加 icomm 目录

Hareware/icomm/目录集成了 config, firmware, wpa_supplicant_lib，
此目录将单独提供，存放在 guide 文档文件同目录中。

Config:

系统配置文件 init.connectivity.rc，wpa_supplicant 配置文件
wpa_supplicant.conf & p2p_supplicant_icomm.conf。

Firmware：

ssv6051 驱动相关文件。

Wpa_supplicant_lib:

External 下的 Wpa_supplicant_8 WiFi tools 使用的私有库，在
device/rockchip/rksdk/wifi_bt_common.mk 中有指定 lib_driver_cmd_icomm

1.6.2 HAL 层中 wifi control

- FILE: hardware/libhardware_legacy/include/hardware_legacy/wifi.h
  在 wifi chip type list 中增加 ssv6051 型号，如下：

```
enum WIFI_CHIP_TYPE_LIST{
    RTL8188CU = 0,
    RTL8192CU,
    RTL8188EU,
    BCM4330,
    RK901,
    RK903,
    AP6335,
    AP6234,
    AP6441,
    MT7601,
    RTL8723AS,
    RTL8723AU,
    RTL8723BS,
    RTL8723BU,
    RTL8192DU,
    MT6620,
    ESP8089,
    SSV6051,
    NUM_MAX,
};
```

- FILE: hardware/libhardware_legacy/wifi/Android.mk

  定义 WIFI_SSV6051 宏

```
LOCAL_SRC_FILES += wifi/rk_wifi_ctrl.c
ifeq ($(strip $(BOARD_CONNECTIVITY_VENDOR)),iCommCooperation)
LOCAL_CFLAGS += -DWIFI_SSV6051
endif
ifeq ($(strip $(BOARD_CONNECTIVITY_VENDOR)),Espressif)
LOCAL_CFLAGS +=-DWIFI_ESP8089
endif
```

- FILE: hardware/libhardware_legacy/wifi/rk_wifi_ctrl.c

  在芯片检测函数中增加 ssv6051 芯片类型的判断。

```
    }
    else if (0 == strncmp(buf,"SSV6051",strlen("SSV6051")))
    {
        wifi_chip_type = SSV6051;
        ALOGD("Read wifi chip type OK ! wifi_chip_type = SSV6051");
    }
```

- FILE: hardware/libhardware_legacy/wifi/wifi.c

  有于本文件修改地方众多，在这里只要解释这个文件的修改方法，此文件将单独提供。

  1，增加定义,包括驱动名称定义，驱动文件所在位置定义，以及驱动文件使用的配置文件定义。

  2，增加 service 名称的支持，此名称对应在 init.connectivity.rc 中，例如：ssv_p2p_supp。

  3，增加在 wifi load drvier 中加载 ssv6051.ko 驱动文件。

  4，在 start_wpa_supplicant 中增加 ssv6051 芯片的筛选，以保证启动正确的 service，确保 wpa_supplicant_icomm 能正常启动。

1.7 System Doc modify

- FILE:system/netd/Android.mk

  创建一个宏，wifi_chip_type_ssv6051。

```
ifeq ($(strip $(BOARD_WIFI_VENDOR)), iCommCooperation)
LOCAL_CFLAGS += -DWIFI_CHIP_TYPE_SSV6051
endif
```

- System/netd/SoftapController.cpp

  增加对 hostapd_icomm 的支持，确保开启 softap 后顺利调用 hostapd_icomm。

```
static const char HOSTAPD_CONF_FILE[]     = "/data/misc/wifi/hostapd.conf";
static const char HOSTAPD_BIN_FILE[]     = "/system/bin/hostapd";
static const char HOSTAPD_SSV_BIN_FILE[]     = "/system/bin/hostapd_icomm";

  if (!pid) {
      ensure_entropy_file_exists();
      if (execl(HOSTAPD_SSV_BIN_FILE, HOSTAPD_SSV_BIN_FILE,
                "-e", WIFI_ENTROPY_FILE,
                HOSTAPD_CONF_FILE, (char *) NULL)) {
          ALOGE("execl failed (%s)", strerror(errno));
      }
      ALOGE("SoftAP failed to start");
      return ResponseCode::ServiceStartFailed;
  } else {
      mPid = pid;
      ALOGD("SoftAP started successfully");
      usleep(AP_BSS_START_DELAY);
  }
  return ResponseCode::SoftapStatusResult;
```

➢ FILE:system/netd/TetherController.cpp

增加--dhcp-authoritative 参数支持，加快 wifi 连接速度。dhcp authoritative 缺失，可能会导致客户端长时间等待 dhcp 分配 ip 超时。

```
#ifdef WIFI_CHIP_TYPE_SSV6051
        int num_processed_args = 8 + (num_addrs/2) + 1; // 1 null for termination
        char **args = (char **)malloc(sizeof(char *) * num_processed_args);
        args[num_processed_args - 1] = NULL;
        args[0] = (char *)"/system/bin/dnsmasq";
        args[1] = (char *)"--keep-in-foreground";
        args[2] = (char *)"--no-resolv";
        args[3] = (char *)"--no-poll";
        // TODO: pipe through metered status from ConnService
        args[4] = (char *)"--dhcp-option-force=43,ANDROID_METERED";
        args[5] = (char *)"--pid-file";
        args[6] = (char *)"--dhcp-authoritative";
        args[7] = (char *)"";

        int nextArg = 8;

#else
```

总结

1， 移植过程中，如果 wifi 打不开，或者打开后无法扫描，请检查 wifi.c 文件。并在系统中通过 lsmod 参考驱动是否有加载，如果没有请检查 wifi.c 中的 driver load 函数，如果 lsmod 查到 ssv6051，但无法扫描 ap，请执行 top 查看 wpa_supplicant_icomm

程序是否有启动，如果没有，请查看 wifi.c 中的 start_supplicant 函数和 init.connectivity.rc 中的 service “ssv_p2p_supp”。

2， 编译完成后请务必在 out 目录下检查 wpa_supplicant.conf, p2p_supplicant_icomm.conf, ssv6051-sw.bin, ssv6051-wifi.cfg, ssv6051.ko 文件是否存在。