APPLICATION NOTE

ATBM 驱动配置说明_FAQ

ATBM603X

1x1 802.11b/g/n Wi-Fi 芯片

Table of contents

作者	版本	说明
Yuzhihuang	V0.1	该文档适用于 SVN1359 版本以后的驱动
yuzhihuang	V0.2	增加调试等级信息说明
Yuzhihuang	V0.3	增加加载驱动固件方法说明
Yuzhihuang	V0.4	小改
Yuzhihuang	V0.5	小改
YUZHIHUAN	V1.1	正式版本
Yuzhihuang	V1.2	新增配置项，适用 1584 以后驱动
Yuzhihuang	V1.3	驱动增加 firmware 目录，对该目录的说明
Yuzhihuang	V1.4	增加 sdio 移植配置说明
Yuzhihuang	V1.5	刷新下说明
Yuzhihuang	V1.6	添加一些调试方法
Yuzhihuang	V1.7	添加 ATBM6012B 兼容配置

1 单独编译驱动的编译方法

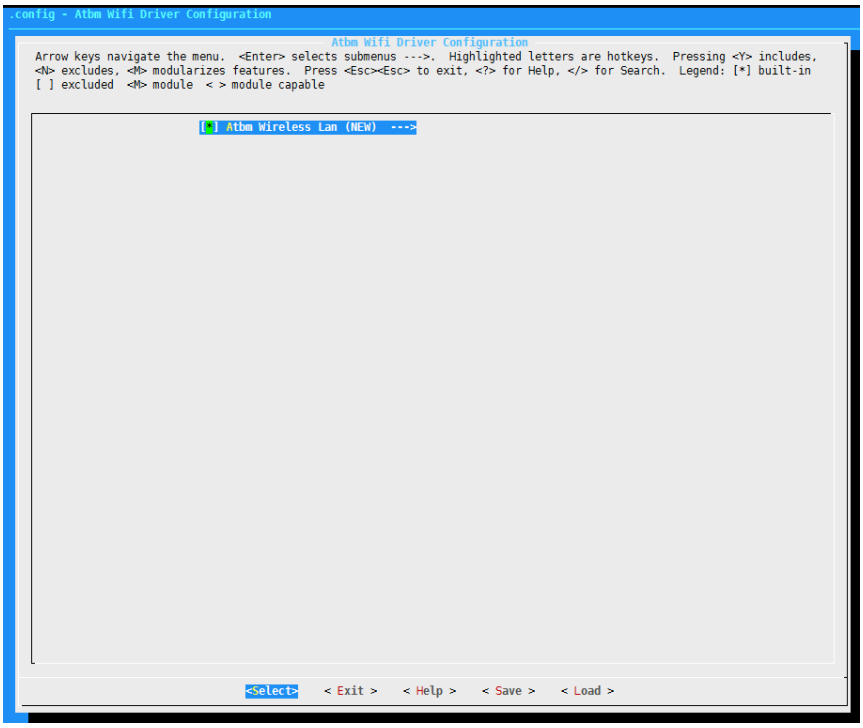
1.1 驱动通过 Make menuconfig 进行配置相关参数

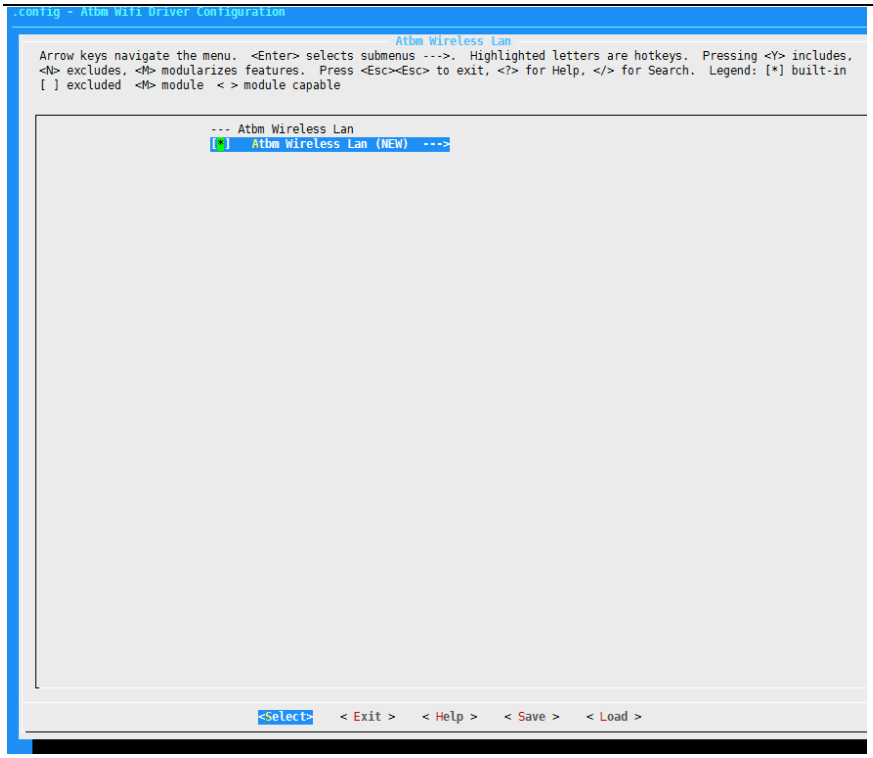
(1) 配置编译环境

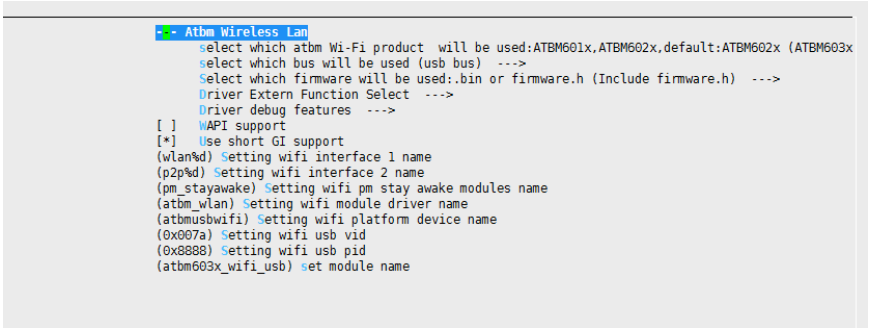
修改驱动根目录的 Makefile,配置对应的内核路径，平台架构以及交叉编译工具链

```
#PLATFORM_HI3516EV200                    19
#PLATFORM_XUNWEI_2G                       20
#PLATFORM_NVT98517       21
export
platform ?= PLATFORM_SIGMASTAR
#Android
#Linux
sys ?= linux
#arch:arm or arm64 or mips(NVT98517)
arch ?= arm
#export
#ATBM_WIFI__EXT_CCFLAGS = -DATBM_WIFI_PLATFORM=$(platform)

ifeq ($(CUSTOMER_SUPPORT_USED),y)
MAKEFILE_SUB ?= Makefile.build.customer
else
MAKEFILE_SUB ?= Makefile.build
endif

ifeq ($(platform),PLATFORM_SIGMASTAR)
KERDIR:=/usr/lchome/yuzhihuang/Mstar/325/kernel
CROSS_COMPILE:=/usr/lchome/yuzhihuang/Mstar/325/arm-buildroot-linux-uclibcgnueabihf-4.9.4/bin/arm-buildroot-linux-uclibcgnueabi
hf-
arch = arm
ATBM_WIFI__EXT_CCFLAGS = -DATBM_WIFI_PLATFORM=18
```

(2) 进到驱动根目录执行，make menuconfig

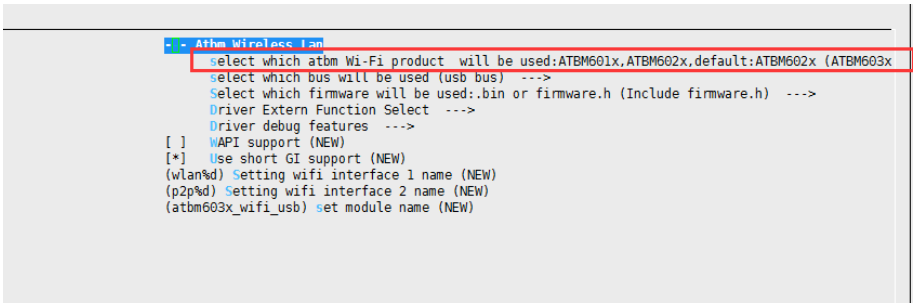


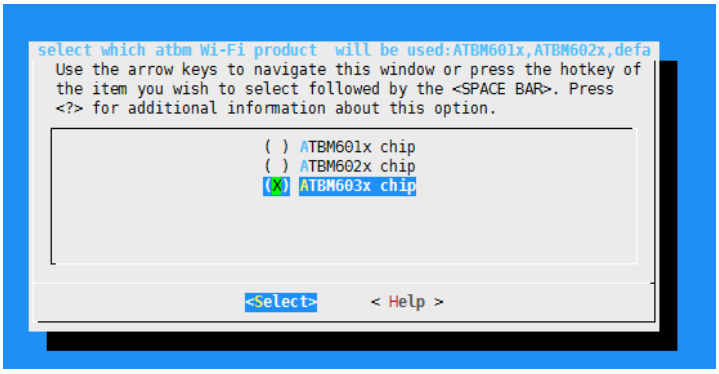




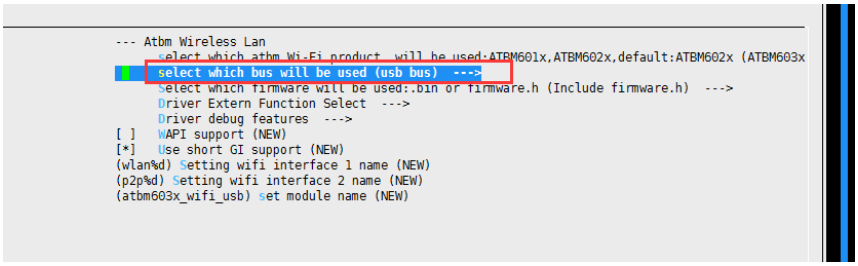
1.2 配置说明

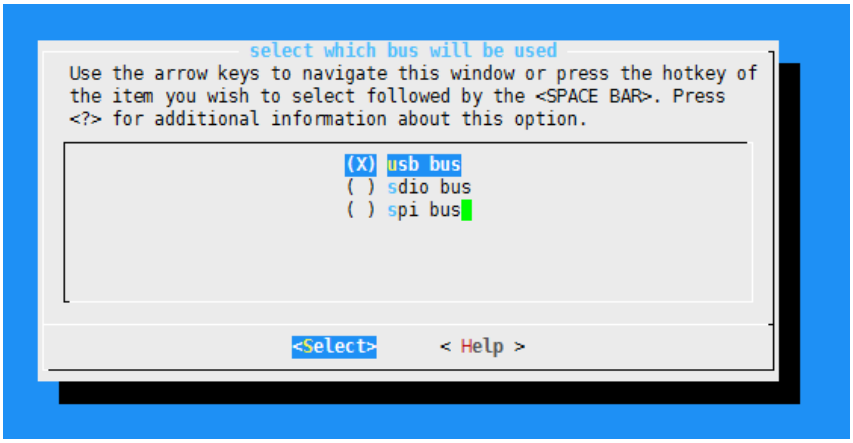
(1) 选择 **wifi** 芯片型号



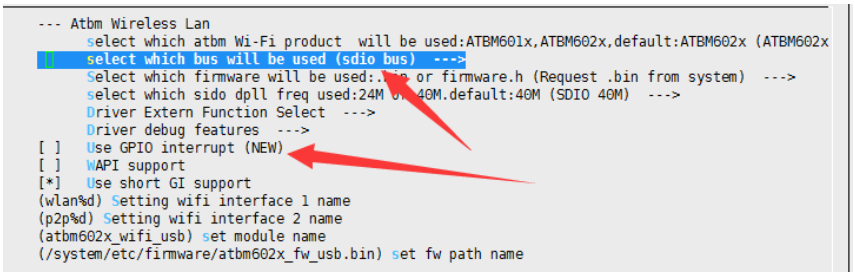


（2） 选择通信总线接口





如果选择 sdio 总线，需要选择是 sdio 中断还是 gpio 中断



a） 如果需要使用 GPIO 中断，

需要做如下修改，否则跳过此部分 ：

hal_apollo/Makefile 中，根据 platform 添加 GPIO 中断的宏。

```
ccflags-y += -DOPER_CLOCK_USE_SEM
ccflags-y += -DEXIT_MODULE_RESET_USB=0
ccflags-y += -DATBM_VIF_LIST_USE_RCU_LOCK
#ccflags-y += -DATBM_SUPPORT_SMARTCONFIG

ifeq ($(platform),PLATFORM_AMLOGIC_S805)
ccflags-y += -DCONFIG_ATBM_APOLLO_USE_GPIO_IRQ
endif
ifeq ($(platform),PLATFORM_FH_IPC)
ccflags-y += -DCONFIG_ATBM_APOLLO_USE_GPIO_IRQ
endif

ifeq ($(platform),PLATFORM_AMLOGIC)
#ccflags-y += -DCONFIG_ATBM_APOLLO_USE_GPIO_IRQ
endif
```

hal_apollo/apollo_plat.h 添加宏,该宏的值和顶层 Makefile 中 export 出来的值一样

```
*PLATFORM_FRIENDLY:based on linux3.086
*
*********************************************************
*/
#define PLATFORM_XUNWEI                         (1)
#define PLATFORM_SUN6I                          (2)
#define PLATFORM_FRIENDLY                       (3)
#define PLATFORM_SUN6I_64                       (4)
#define PLATFORM_CDLINUX                        (12)
#define PLATFORM_AMLOGIC_S805           (13)
#define PLATFORM_AMLOGIC_905            (8)
#define PLATFORM_FH_IPC                         (18)

#ifndef  ATBM_WIFI_PLATFORM
#define ATBM_WIFI_PLATFORM                      PLATFORM_XUNWEI
#endif
```

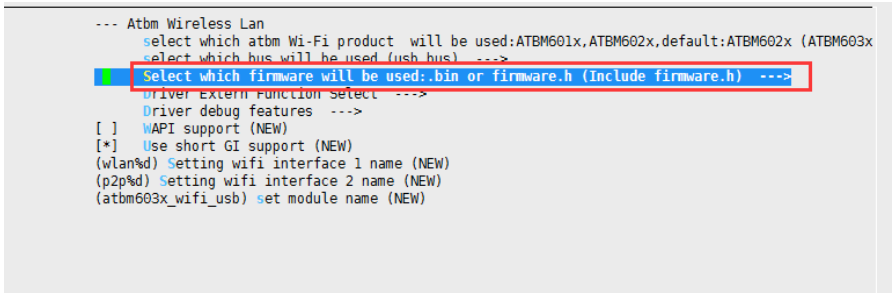
hal_apollo/atbm_platform.c 添加 GPIO 中断引脚号

```
^M
#endif  //CONFIG_ATBM_APOLLO_USE_GPIO_IRQ^M
#endif^M
struct atbm_platform_data platform_data = {^M
#if (ATBM_WIFI_PLATFORM == 10)^M
        .mmc_id         = "mmc1",^M
#else^M
        .mmc_id         = "mmc2",^M
#endif^M
        .clk_ctrl       = NULL,^M
        .power_ctrl     = atbm_power_ctrl,^M
        .insert_ctrl    = atbm_insert_crtl,^M
#if(ATBM_WIFI_PLATFORM == PLATFORM_XUNWEI)^M
        .irq_gpio       = EXYNOS4_GPX2(4),^M
        .power_gpio     = EXYNOS4_GPC1(1),^M
#endif^M
#if(ATBM_WIFI_PLATFORM == PLATFORM_AMLOGIC_S805)^M
        .irq_gpio       = INT_GPIO_4,^M
        .power_gpio     = 0,^M
#endif^M
#if(ATBM_WIFI_PLATFORM == PLATFORM_AMLOGIC_905)^M
        .irq_gpio       = 100,^M
        .power_gpio     = 0,^M
#endif^M
#if(ATBM_WIFI_PLATFORM == PLATFORM_FRIENDLY)^M
        .power_gpio     = EXYNOS4_GPK3(2),^M
#endif^M
#if(ATBM_WIFI_PLATFORM == PLATFORM_FH_IPC)^M
        .irq_gpio       = 0,^M
#endif^M
        .reset_gpio = 0,^M
};^M
^M
struct atbm_platform_data *atbm_get_platform_data(void)^M
{^M
        return &platform_data;^M
```

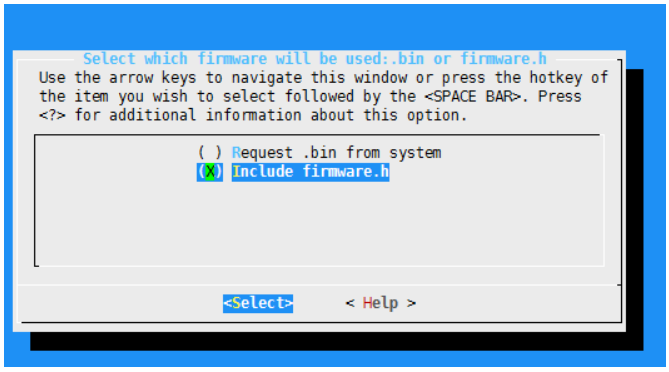
b) 整体设定完毕后，在驱动源码下执行 make 即可。

(3) 选择固件方式

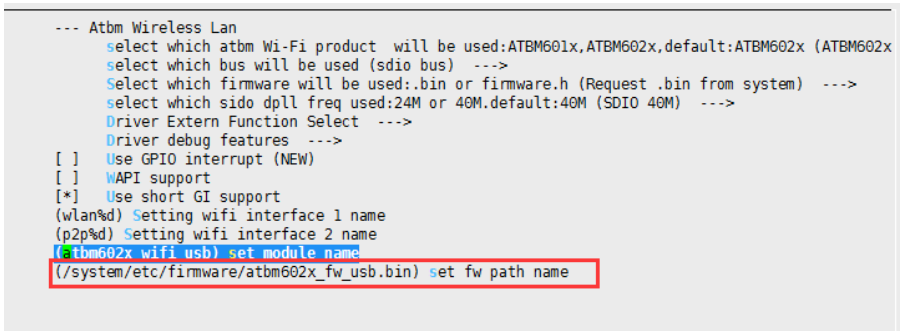
可以选择独立于驱动之外的 bin 文件，或者选择包含在驱动里面的 hal_apollo/firmware.h



IPC 考虑到使用方便的原因一般选择 firmware.h，安卓平台一般都是用的 bin 文件

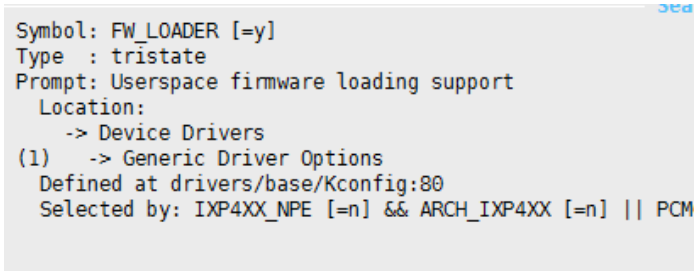


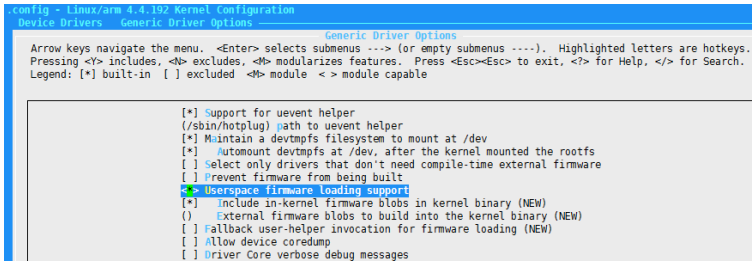
出现下图的提示说明固件使用的是独立的 bin 文件，需要将红框固件名前缀去掉，只留下 atbm602x_fw_usb.bin。



PS:

1）固件使用 bin 文件需要内核支持下载 firmware,所以在内核需要打开 FW_LOADER 宏,不打开没法正常下载固件。





2）最终的固件只能放在内核预定义的路径

2.1）该路径定义于：kernel/drivers/base/firmware_class.c

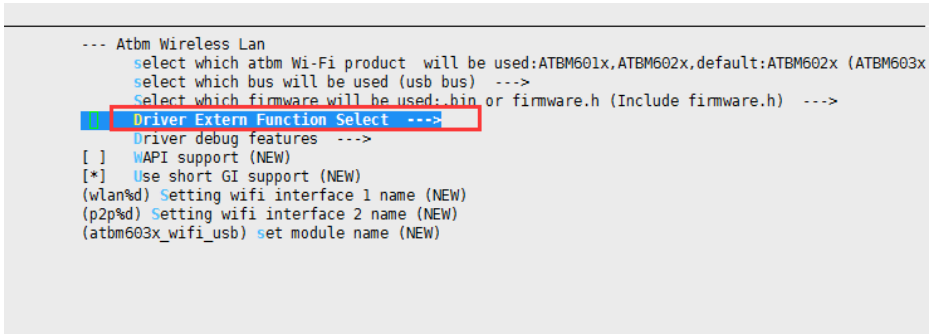
```
/* direct firmware loading support */
static char fw_path_para[256];
static const char * const fw_path[] = {
        fw_path_para,
        "/lib/firmware/updates/" UTS_RELEASE,
        "/lib/firmware/updates",
        "/lib/firmware/" UTS_RELEASE,
        "/lib/firmware"
};
```

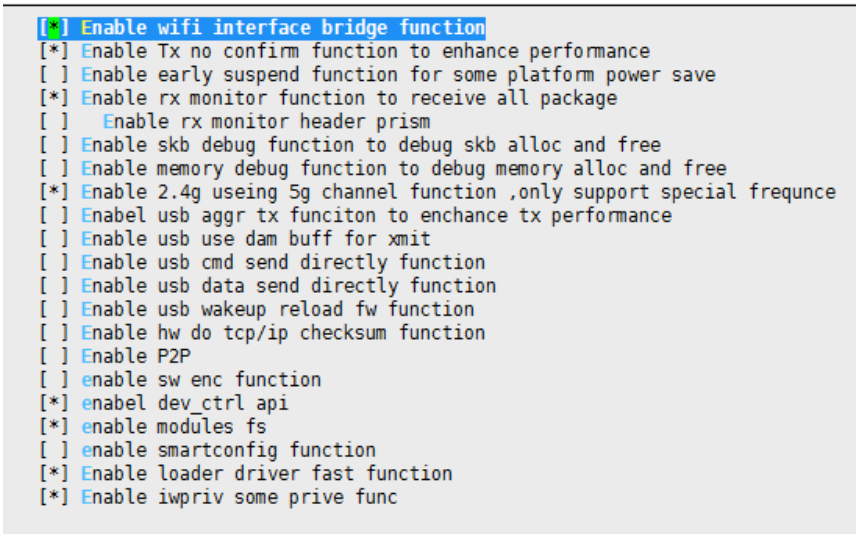
2.2）在运行系统里面添加存放固件的路径

例如固件放的路径为/mnt/sdcard/firmware/添加路径的方法如下：

```
echo /mnt/sdcard/firmware/ > /sys/module/firmware_class/parameters/path
```

(4) 驱动一些扩展功能





a) Enable wifi interface bridge function

选择驱动是否支持桥接

b) Enable Tx no confirm function to enhance performance
这个功能默认是打开的。

c) Enable early suspend function for some platform power save
与平台相关，安卓系统层支持休眠时候需要打开。

d) Enable rx monitor function to receive all package
驱动是否支持进入监听状态的功能，默认打开
子选项：Enable rx monitor header prism
Monitor 头部修改为 PRISM 格式，默认为 RATIO 格式

e) Enable skb debug function to debug skb alloc and free
打开 skb 泄露 debug 的功能，此功能通常不打开

f) Enable memory debug function to debug memory alloc and free
打开 memory 泄露 debug 的功能，此功能通常不打开

g) Enable 2.4g useing 5g channel function ,only support special frequnce
是否使用 5G 信道作为特殊频点，默认关闭状态

h) Enabel usb aggr tx funciton to enchance tx performance
打开 usb 聚合发送数据包的功能，目前关闭，cpu 频率较低时可以打开

i) Enable usb use dam buff for xmit
使用 usb 的 dma buff,一般打开 usb 聚合时打开此功能

j) Enable usb cmd send directly function
cpu 频率较低时打开此功能

k) Enable usb data send directly function
cpu 频率较低时打开此功能

l) Enable usb wakeup reload fw function
android 平台休眠唤醒时是否重新 load 固件，默认关闭

m) Enable hw do tcp/ip checksum function
是否使能 aresB 的 check sum 功能，默认关闭

n) Enable P2P
使能 P2P mirecast 功能

o) enable sw enc function

支持 enc 功能

p) enabel dev_ctrl api
使能 dev ioctl 相关命令功能

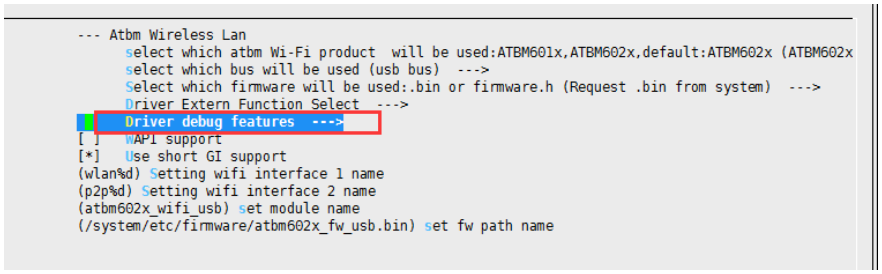
q) enable modules fs
使能 modules fs 功能,打开此功能会在/sys/module/<driver_name>/目录下生成 atbm_fs 目录
该目录可以进行功能调试

r) enable smartconfig function
使能 smart config 功能

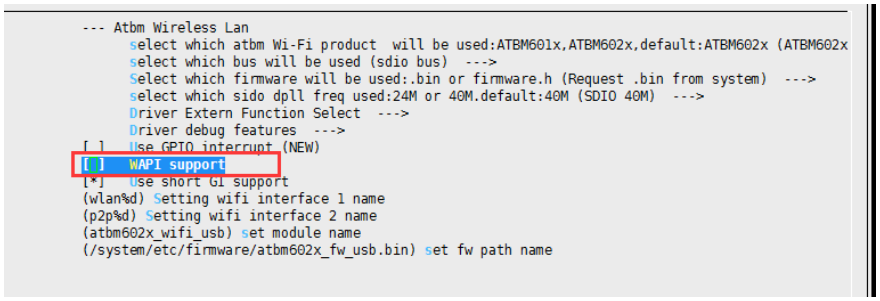
s) Enable loader driver fast function
打开此功能可以缩短加载 usb 驱动的时间

t) Enable iwpriv some prive func
打开此功能支持私有协议功能

(5) 内部调试使用



(6) 是否支持 WAPI



WAPI 是我国首个在计算机宽带无线网络通信领域自主创新并拥有知识产权的安全接入技术标准。WAPI 同时也是中国无线局域网强制性标准中的安全机制。

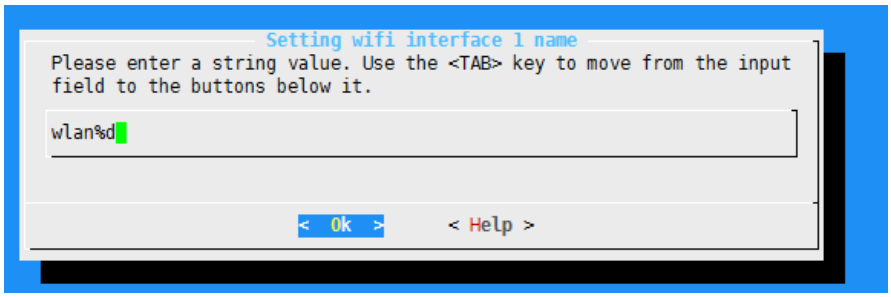
(7) 是否需要 short GI

```
--- Atbm Wireless Lan
      select which atbm Wi-Fi product  will be used:ATBM601x,ATBM602x,default:ATBM602x (ATBM602x
      select which bus will be used (sdio bus)  --->
      Select which firmware will be used:.bin or firmware.h (Request .bin from system)  --->
      select which sido dpll freq used:24M or 40M.default:40M (SDIO 40M)  --->
      Driver Extern Function Select  --->
      Driver debug features  --->
[ ]   Use GPIO interrupt (NEW)
[ ]   WAPI support
[*]   Use short GI support
(wlan%d) Setting wifi interface 1 name
(p2p%d) Setting wifi interface 2 name
(atbm602x_wifi_usb) set module name
(/system/etc/firmware/atbm602x_fw_usb.bin) set fw path name
```

该功能默认打开。

(8) 修改 wifi 接口名称

```
--- Atbm Wireless Lan
      select which atbm Wi-Fi product  will be used:ATBM601x,ATBM602x,default:ATBM602x (ATBM602x
      select which bus will be used (sdio bus)  --->
      Select which firmware will be used:.bin or firmware.h (Request .bin from system)  --->
      select which sido dpll freq used:24M or 40M.default:40M (SDIO 40M)  --->
      Driver Extern Function Select  --->
      Driver debug features  --->
[ ]   Use GPIO interrupt (NEW)
[ ]   WAPI support
[*]   Use short GI support
(wlan%d) Setting wifi interface 1 name
(p2p%d) Setting wifi interface 2 name
(atbm602x_wifi_usb) set module name
(/system/etc/firmware/atbm602x_fw_usb.bin) set fw path name
```



wifi 接口默认的名称是：wlan0 以及 p2p0.

客户可以根据需要执行修改：

1）客户想使用 wlan0 以及 wlan1,

可直接修改 p2p%d 修改为 wlan%d 即可。

2）客户想使用 wlan 以及 p2p 接口

将 wlan%d 修改为 wlan

将 p2p%d 修改为 p2p

(9) 修改驱动名称以及挂载结点名称

```
--- Atbm Wireless Lan
      select which atbm Wi-Fi product  will be used:ATBM601x,ATBM602x,default:ATBM602x (ATBM602x
      select which bus will be used (sdio bus)  --->
      Select which firmware will be used:.bin or firmware.h (Request .bin from system)  --->
      select which sido dpll freq used:24M or 40M.default:40M (SDIO 40M)  --->
      Driver Extern Function Select  --->
      Driver debug features  --->
[ ]   Use GPIO interrupt (NEW)
[ ]   WAPI support
[*]   Use short GI support
(wlan%d) Setting wifi interface 1 name
(p2p%d) Setting wifi interface 2 name
(atbm602x_wifi_usb) set module name
(/system/etc/firmware/atbm602x_fw_usb.bin) set fw path name
```

根据客户需要执行修改，编译出来生成的驱动名称。
以及在系统中挂载驱动 lsmod 看到的驱动名称。

1.3 ATBM6012B 配置说明

带 6012B 名称的 .h 文件为 ATBM6012B 运行固件。

(1) 驱动同时兼容 ATBM6032ix & ATBM6012B

当配置为 ATBM603x 同时是 USB 总线此时配置界面会增加 ATBM6012B chip 的配置项

```
--- Atbm Wireless Lan
[ ]   support wireless wext
      select which atbm Wi-Fi product  will be used:ATBM601x,ATBM602x,ATBM603x,ATBM604x,default:ATBM602x (ATBM603x chip)  --
        ATBM603x chip
[*]       ATBM6012B chip
      select which bus will be used (usb bus)  --->
      Select which firmware will be used:.bin or firmware.h (Include firmware.h)  --->
      Driver Extern Function Select  --->
      Driver debug features  --->
[ ]   WAPI support
[*]   Use short GI support
(wlan%d) Setting wifi interface 1 name
[*]   ENABLE scond interface
(p2p%d) Setting wifi interface 2 name
(pm_stayawake) Setting wifi pm stay awake modules name
(atbm_wlan) Setting wifi module driver name
(atbm_dev_wifi) Setting wifi platform device name
(0x007a) Setting wifi usb vid
(0x8888) Setting wifi usb pid
(ATBM_313E_HT40) set module name
```

此时驱动需要包含两份固件，驱动会根据芯片型号自动加载对应的 firmware
驱动包提供的 firmware 目录下找到如下固件：

AresM_6012B_IPC_NOTXCONRIM_USB_svnXXXX_24M_comb.h
Ares_B_Chip_IPC_NOTXCONRIM_USB_svnXXXX_24M.h

需要执行如下步骤：

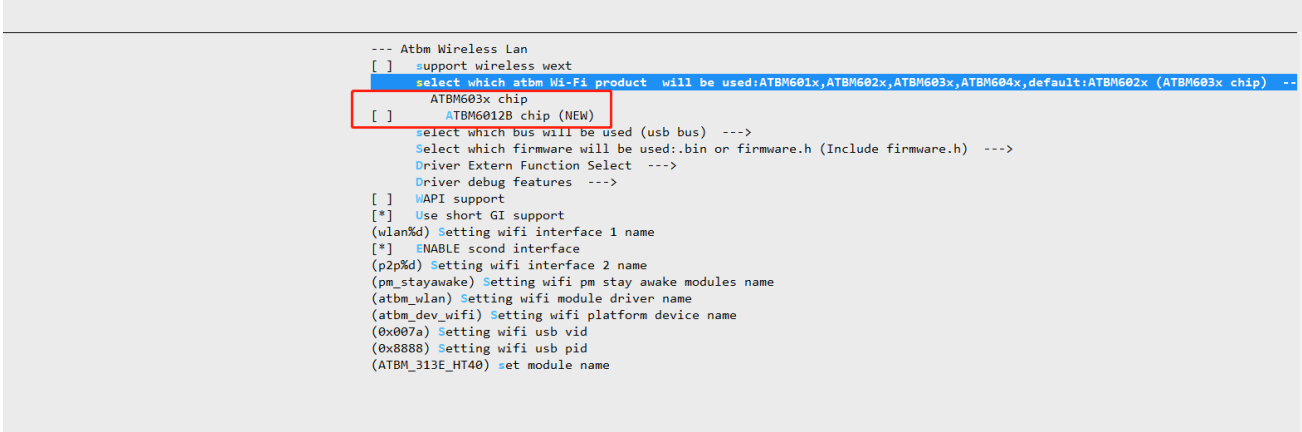
a.在配置界面选上 ATBM6012B 配置

b.将 AresM_6012B_IPC_NOTXCONRIM_USB_svnXXXX_24M_comb.h 更名为 firmware_usb_6012b.h 放到驱动根目录下的 hal_apollo 目录下

c.将 Ares_B_Chip_IPC_NOTXCONRIM_USB_svnXXXX_24M.h 更名为 firmware_usb.h 放到驱动根目录下的 hal_apollo 目录下

(2) 驱动仅支持 ATBM6012B

不需要配置上 ATBM6012B



驱动包提供的 firmware 目录下找到如下固件：

AresM_6012B_IPC_NOTXCONRIM_USB_svnXXXX_24M.h

将 AresM_6012B_IPC_NOTXCONRIM_USB_svnXXXX_24M.h 改名为 firmware_usb.h 更名为 firmware_usb.h 放到驱动根目录下的 hal_apollo 目录下

1.4 SDIO WIFI 移植配置说明

sdio 通信有 sdio 中断和 GPIO 中断通信方式。

(1) 注意

这里需要注意下如果 mmc host 进行了如下配置了：

```
void mmc_rescan(struct work_struct *work)
{
    struct mmc_host *host =
        container_of(work, struct mmc_host, detect.work);
    int i;

    if (host->trigger_card_event && host->ops->card_event) {
        host->ops->card_event(host);
        host->trigger_card_event = false;
    }

    if (host->rescan_disable)
        return;

    /* If there is a non-removable card registered, only scan once */
    if ((host->caps & MMC_CAP_NONREMOVABLE) && host->rescan_entered)
        return;
    host->rescan_entered = 1;

    mmc_bus_get(host);
```

代表 mmc host 不允许 sdio 从设备进行热插拔，调用 mmc rescan 直接返回,不会去扫卡。所以此时驱动直接加载卸载即可。

有一些平台刚上电 sdio 不稳定就需要复位一下 sdio wifi，那么就需要增加复位扫卡的动作。

(2) SDIO 中断方式

说明：

以君正 T31 平台为例。

1.4.2.1 修改驱动根目录的 Makefile

自定义一个 platform

```
#PLATFORM_SIGMASTAR                    18
#PLATFORM_HI3516EV200                  19
#PLATFORM_XUNWEI_2G                     20
#PLATFORM_NVT98517      21
#PLATFORM_ANYKA_SDIO            22
#PLATFORM_INGENICT31            23
export
platform ?= PLATFORM_INGENICT31
#Android
#Linux
sys ?= linux
#arch:arm or arm64 or mips(NVT98517)
arch ?= arm
#export
#ATBM_WIFI__EXT_CCFLAGS = -DATBM_WIFI_PLATFORM=$(platform)
```

增加一个编译配置项，增加内核路径和工具链路径用于单独编译驱动使用

需要注意 ATBM_WIFI_PLATFORM 值为 23

```
arch = arm
ATBM_WIFI__EXT_CCFLAGS = -DATBM_WIFI_PLATFORM=22
endif

ifeq ($(platform),PLATFORM_INGENICT31)
ifeq ($(sys),linux)
KERDIR:=/usr/lchome/yuzhihuang/ankai/Linux/anyka3918ev500/AnyCloudV500_PDK_V1.02/PDK/SDK/sdk_release_
CROSS_COMPILE:=/usr/lchome/yuzhihuang/ankai/Linux/anyka3918ev500/AnyCloudV500_PDK_V1.02/PDK/SDK/sdk_r
endif
export
arch = arm
ATBM_WIFI__EXT_CCFLAGS = -DATBM_WIFI_PLATFORM=23
endif
```

修改 hal_apollo/apollo_plat.h

增加一个平台定义宏值为 23

```
25: */
26: #define PLATFORM_XUNWEI             (1)
27: #define PLATFORM_SUN6I              (2)
28: #define PLATFORM_FRIENDLY           (3)
29: #define PLATFORM_SUN6I_64           (4)
30: #define PLATFORM_CDLINUX            (12)
31: #define PLATFORM_AMLOGIC_S805       (13)
32: #define PLATFORM_AMLOGIC_905        (8)
33: #define PLATFORM_ANYKA_SDIO         (22)
34: #define PLATFORM_INGENICT31         (23)
35:
36:
37:
38: #ifndef  ATBM_WIFI_PLATFORM
39: #define ATBM_WIFI_PLATFORM          PLATFORM_INGENICT31
40: #endif
41:
42: #define APOLLO_1505  0
```

1.4.2.2 修改 hal_apollo/atbm_platform.c

如果需要增加复位扫卡动作见后面【1.3.4 复位&扫卡动作】

(3) GPIO 中断方式

说明：

以 amlogic s905 平台为例

1.4.3.1 修改驱动根目录的 Makefile

自定义一个 platform

```
28: #PLATFORM_SIGMASTAR                    18
29: #PLATFORM_HI3516EV200                  19
30: #PLATFORM_XUNWEI_2G                     20
31: #PLATFORM_NVT98517      21
32: #PLATFORM_ANYKA_SDIO                22
33: #PLATFORM_INGENICT31                23
34: export
35: platform ?= PLATFORM_AMLOGIC_905
36: #Android
37: #Linux
38: sys ?= linux
39: #arch:arm or arm64 or mips(NVT98517)
40: arch ?= arm
```

增加一个编译配置项，增加内核路径和工具链路径用于单独编译驱动使用

需要注意 ATBM_WIFI_PLATFORM 值为 8

```
256: ifeq ($(platform),PLATFORM_AMLOGIC_905)
257: ifeq ($(sys),Android)
258: #KERDIR:=/wifi_prj/staff/zhouzhanchao/amlogic/release_s905_l/out/target/product/p200/obj/KERNEL_OBJ/
259: KERDIR:=/wifi_prj/staff/mengxuehong/s905l/S905L/out/target/product/p201_iptv/obj/KERNEL_OBJ/
260: CROSS_COMPILE:=/ssd-home/mengxuehong/buildTool1/gcc-linaro-aarch64-linux-gnu-4.9-2014.09_linux/bin/aa
261: else
262: #KERDIR:=/wifi_prj/staff/panxuqiang/64bi_driver/cqa64_linux_qt5.3.2/lichee/linux-3.10/
263: endif
264: export
265: ATBM_WIFI__EXT_CCFLAGS = -DATBM_WIFI_PLATFORM=8
266: arch:=arm64
267: endif
```

在最后的增加一个配置项用于在内核目录直接编译 modules 时候的配置

需要主要 ATBM_WIFI_PLATFORM 值为 8

```
347: ifeq ($(platform),PLATFORM_AMLOGIC_905)
348: export
349: ATBM_WIFI__EXT_CCFLAGS = -DATBM_WIFI_PLATFORM=8
350: endif
```

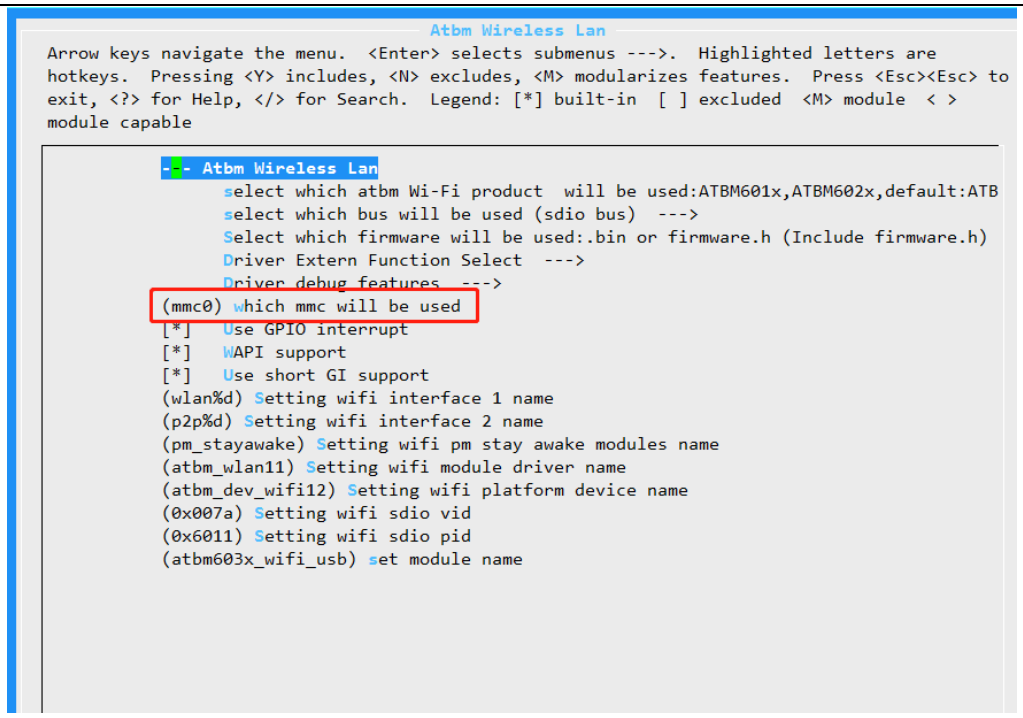
1.4.3.2 打开支持 GPIO 中断配置

```
Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are
hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >
module capable

          --- Atbm Wireless Lan
                select which atbm Wi-Fi product  will be used:ATBM601x,ATBM602x,default:ATB
                select which bus will be used (sdio bus)  --->
                Select which firmware will be used:.bin or firmware.h (Include firmware.h)
                Driver Extern Function Select  --->
                Driver debug features  --->
          (mmc0) which mmc will be used
          [*]   Use GPIO interrupt
          [*]   WAPI support
          [*]   Use short GI support
          (wlan%d) Setting wifi interface 1 name
          (p2p%d) Setting wifi interface 2 name
          (pm_stayawake) Setting wifi pm stay awake modules name
          (atbm_wlan11) Setting wifi module driver name
          (atbm_dev_wifi12) Setting wifi platform device name
          (0x007a) Setting wifi sdio vid
          (0x6011) Setting wifi sdio pid
          (atbm603x_wifi_usb) set module name
```

1.4.3.3 修改使用平台的 mmc 口

根据实际使用的 mmc 口进行配置。

一般平台有两个 mmc 口，mmc0 或者是 mmc1。



1.4.3.4 修改 hal_apollo/apollo_plat.h

增加一个平台宏定义值为 8

```
26: #define PLATFORM_XUNWEI             (1)
27: #define PLATFORM_SUN6I              (2)
28: #define PLATFORM_FRIENDLY           (3)
29: #define PLATFORM_SUN6I_64           (4)
30: #define PLATFORM_CDLINUX            (12)
31: #define PLATFORM_AMLOGIC_S805       (13)
32: #define PLATFORM_AMLOGIC_905        (8)
33: #define PLATFORM_ANYKA_SDIO         (22)
34: #define PLATFORM_INGENICT31         (23)
35:
36:
37:
38: #ifndef  ATBM_WIFI_PLATFORM
39: #define ATBM_WIFI_PLATFORM          PLATFORM_AMLOGIC_905
40: #endif
```

1.4.3.5 修改 hal_apollo/atbm_platform.c

增加一个编译时候打印的信息

```
53: #if (ATBM_WIFI_PLATFORM == PLATFORM_AMLOGIC_905)
54: #define PLATFORMINF     "amlogic_905"
55: #endif
```

如果有不同版本的内核并且有较差异需要增加进来

```
#if ((ATBM_WIFI_PLATFORM == PLATFORM_AMLOGIC_S805) || (ATBM_WIFI_PLATFORM == PLATFORM_AMLOGIC_905))

#if (LINUX_VERSION_CODE < KERNEL_VERSION(3, 14, 0))
extern void wifi_teardown_dt(void);
extern int wifi_setup_dt(void);
#endif
#endif //#if (ATBM_WIFI_PLATFORM == PLATFORM_AMLOGIC_S805)
```

中断相关的声明定义

```
#if(ATBM_WIFI_PLATFORM == PLATFORM_AMLOGIC_905)

#if (LINUX_VERSION_CODE >= KERNEL_VERSION(3, 14, 0))
extern int wifi_irq_num(void);
#endif

u32 atbm_wlan_get_oob_irq(void)
{
    u32 host_oob_irq = 0;

#if (LINUX_VERSION_CODE < KERNEL_VERSION(3, 14, 0))
    host_oob_irq = INT_GPIO_4;
#else
    host_oob_irq = wifi_irq_num();
#endif
    atbm_printk_platform("host_oob_irq: %d \r\n", host_oob_irq);

    return host_oob_irq;
}
#endif
```

struct atbm_platform_data platform_data 结构中进行中断号初始化

```
#if(ATBM_WIFI_PLATFORM == PLATFORM_AMLOGIC_905)
     .irq_gpio    = 100,
     .power_gpio = 0,
#endif
```

在 atbm_plat_request_gpio_irq 函数中增加 GPIO 中断的初始化

```
#if(ATBM_WIFI_PLATFORM == PLATFORM_AMLOGIC_905)
    bgf_irq = atbm_wlan_get_oob_irq();
    atbm_printk_platform("atbm_plat_request_gpio_irq \n");
    /* Request the IRQ */
    ret =  request_threaded_irq(bgf_irq, atbm_gpio_hardirq,
                atbm_gpio_irq,
                IORESOURCE_IRQ | IORESOURCE_IRQ_HIGHEDGE | IORESOURCE_IRQ_SHAREABLE,
                "atbm_wlan_irq", self);

    if (WARN_ON(ret))
        goto ↓err;
```

在 atbm_plat_free_gpio_irq 函数增加 GPIO 中断的反初始化

```
#if(ATBM_WIFI_PLATFORM == PLATFORM_AMLOGIC_905)
    disable_irq(atbm_bgf_irq);
    free_irq(atbm_bgf_irq,self);
#elif (ATBM_WIFI_PLATFORM == PLATFORM_AMLOGIC_S805)
    //do nothing
    disable_irq(atbm_bgf_irq);
    free_irq(atbm_bgf_irq,self);
#else
    disable_irq_wake(atbm_bgf_irq);
    free_irq(atbm_bgf_irq,self);
    gpio_free(pdata->irq_gpio);
#endif
```

(4) 复位&扫卡动作

1.4.4.1 注册

这里主要是在 hal_apollo/atbm_platform.c 里面的 struct atbm_platform_data 结构体的两个函数实现：

```
406: struct atbm_platform_data platform_data = {
407: #ifdef SDIO_BUS
408:     .mmc_id        = CONFIG_ATBM_SDIO_MMC_ID,
409:     .clk_ctrl      = NULL,
410:     .power_ctrl    = atbm_power_ctrl,
411:     .insert_ctrl   = atbm_insert_crtl,
412: #if(ATBM_WIFI_PLATFORM == PLATFORM_XUNWEI)
413:     .irq_gpio    = EXYNOS4_GPX2(4),
414:     .power_gpio = EXYNOS4_GPC1(1),
415: #endif
416: #if(ATBM_WIFI_PLATFORM == PLATFORM_AMLOGIC_S805)
417:
418:     .irq_gpio    = INT_GPIO_4,
419:     .power_gpio = 0,
420: #endif
421: #if(ATBM_WIFI_PLATFORM == PLATFORM_AMLOGIC_905)
422:     .irq_gpio    = 100,
423:     .power_gpio = 0,
424: #endif
425: #if(ATBM_WIFI_PLATFORM == PLATFORM_FRIENDLY)
```

增加平台的扫卡函数以及控制 wifi 复位的 GPIO 号。

```
18 #if (ATBM_WIFI_PLATFORM == PLATFORM_INGENICT31)
19: #define PLATFORMINF "ingenict31"
20: extern int jzmmc_manual_detect(int index, int on);
21: static int WL_REG_EN = 32+25;
22: #endif
23:
```

1.4.4.2 复位

在 hal_apollo/atbm_platform.c 里面 atbm_power_ctrl --> atbm_platform_power_ctrl 函数中增加对 wifi 的复位操作

```
#if (ATBM_WIFI_PLATFORM == PLATFORM_INGENICT31)
{
    if(enabled){
        atbm_printk_platform("[%s] reset altobeam wifi !\n",__func__);

        gpio_request(WL_REG_EN, "sdio_wifi_power_on");

        atbm_printk_platform("PLATFORM_INGENICT31 SDIO WIFI_RESET 0 \n");
        gpio_direction_output(WL_REG_EN, 0);
        msleep(300);
        atbm_printk_platform("PLATFORM_INGENICT31 SDIO WIFI_RESET 1 \n");
        gpio_direction_output(WL_REG_EN, 1);
        msleep(100);
    }
}
#endif//(ATBM_WIFI_PLATFORM == PLATFORM_INGENICT31)
```

1.4.4.3 扫卡

1) 如果平台有提供扫卡函数那么直接调用即可

在 hal_apollo/atbm_platform.c 里面 atbm_insert_crtl --> atbm_platform_insert_crtl 函数增加扫卡的动作

```
#if (ATBM_WIFI_PLATFORM == PLATFORM_INGENICT31)
{

    mdelay(100);
    jzmmc_manual_detect(1, enabled);
    atbm_printk_platform("============platform insert crtl====== enable=%d\n", enabled);

}
```

2) 如果平台没有提供对外的 mmc rescan 函数那么需要自己实现一个

在 hal_apollo/apollo_sdio.c 的 atbm_sdio_init 函数里面已经实现

```
1646: #if ((ATBM_WIFI_PLATFORM != 10) && (ATBM_WIFI_PLATFORM != PLATFORM_AMLOGIC_S805)\
1647:     && (ATBM_WIFI_PLATFORM != PLATFORM_AMLOGIC_905))
1648:
1649:     ret = atbm_sdio_on(pdata);
1650:     if (ret)
1651:         goto err_on;
1652: #endif
1653:     atbm_wtd_init();
1654:     return 0;
1655:
1656: #if ((ATBM_WIFI_PLATFORM != 10) && (ATBM_WIFI_PLATFORM != PLATFORM_AMLOGIC_S805)\
1657:     && (ATBM_WIFI_PLATFORM != PLATFORM_AMLOGIC_905))
1658:
1659: err_on:
1660:     if (pdata->power_ctrl)
1661:         pdata->power_ctrl(pdata, false);
1662: #endif
1663: err_power:
1664:     if (pdata->clk_ctrl)
1665:         pdata->clk_ctrl(pdata, false);
1666: err_clk:
1667:     sdio_unregister_driver(&sdio_driver);
1668: err_reg:
1669:     return ret;
```

```
902: #if ((ATBM_WIFI_PLATFORM != 10) && (ATBM_WIFI_PLATFORM != PLATFORM_AMLOGIC_S805) \
903:     && (ATBM_WIFI_PLATFORM != PLATFORM_AMLOGIC_905))
904:
905: static int atbm_sdio_on(const struct atbm_platform_data *pdata)
906: {
907:     int ret = 0;
908:     if (pdata->insert_ctrl)
909:         ret = pdata->insert_ctrl(pdata, true);
910:     msleep(200);
911:     atbm_detect_card(pdata);
912:     return ret;
913: }
914: #endif //#if ((ATBM_WIFI_PLATFORM != 10) && (ATBM_WIFI_PLATFORM != PLATFORM_AMLOGIC_S805))
```

1.5 编译

```
make;make strip
```

2 驱动放置在内核中的编译方法

2.1 将驱动放置在内核中

进入内核目录下的 drivers/net/wireless/子目录修改 Makefile 和 Kconfig 文件

```
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
fugui@ubuntu200:/wifi_prj/staff/fugui/platform/iTop4412_Kernel_3.0$ cd drivers/net/wireless/
fugui@ubuntu200:/wifi_prj/staff/fugui/platform/iTop4412_Kernel_3.0/drivers/net/wireless$
```

a) 修改 Makefile

```
obj-$(CONFIG_HOSTAP)            += hostap/
obj-$(CONFIG_B43)               += b43/
obj-$(CONFIG_B43LEGACY)         += b43legacy/
obj-$(CONFIG_ZD1211RW)          += zd1211rw/
obj-$(CONFIG_RTL8180)           += rtl818x/
obj-$(CONFIG_RTL8187)           += rtl818x/
obj-$(CONFIG_RTLWIFI)           += rtlwifi/
obj-$(CONFIG_ATBM_WIRELESS)     += atbm_wifi_40M/

# 16-bit wireless PCMCIA client drivers
obj-$(CONFIG_PCMCIA_RAYCS)      += ray_cs.o
obj-$(CONFIG_PCMCIA_WL3501)     += wl3501_cs.o
```

b) 修改 Kconfig

```
source "drivers/net/wireless/rt2x00/Kconfig"
source "drivers/net/wireless/rtlwifi/Kconfig"
source "drivers/net/wireless/wl1251/Kconfig"
source "drivers/net/wireless/wl12xx/Kconfig"
source "drivers/net/wireless/zd1211rw/Kconfig"
source "drivers/net/wireless/mwifiex/Kconfig"
source "drivers/net/wireless/atbm_wifi_40M/Kconfig"

menuconfig MTK_WIRELESS_SOLUTION
        bool "MTK wireless chip configuration"
        help
          "enable/disable and config MTK wireless solution"

if MTK_WIRELESS_SOLUTION
source "drivers/net/wireless/combo_mt66xx/Kconfig"
source "drivers/net/wireless/mt5931/Kconfig"
endif # MTK_WIRELESS_SOLUTION
endif # WLAN
```

修改完 Makefile 和 Kconfig 文件后回到内核顶层目录

将 atbm_wifi_40M 的驱动源码目录复制到内核目录下的 drivers/net/wirelesee/

c) 修改 atbm_wifi_40M 中的 Makefile,指定相关平台选择，默认指定为 platform_other,

PS：注意#PLATFORM_OTHER 20 ，这个值用户自定义，不要和已定义的冲突即可

```
Makefile
28 #PLATFORM_OTHER                20
29 export
30 platform ?= PLATFORM_OTHER      选择平台
31 #Android
32 #Linux
33 sys ?= Linux
34 #arch:arm or arm64
35 arch ?= arm                     选择系统及架构
36 #export
37 #ATBM_WIFI__EXT_CCFLAGS = -DATBM_WIFI_PLATFORM=$(platform)
38
39 ifeq ($(CUSTOMER_SUPPORT_USED),y)
40 MAKEFILE_SUB ?= Makefile.build.customer
41 else
42 MAKEFILE_SUB ?= Makefile.build
43 endif
44
45 ifeq ($(KERNELRELEASE),)
46
47 ifeq ($(platform),PLATFORM_HS_IPC)
48 KERDIR:=/wifi_prj/staff/zhouzhanchao/ankai_hs_ipc/kernel/kernel_testXMFlash/kernel/
49 CROSS_COMPILE:=/wifi_prj/staff/zhouzhanchao/ankai_hs_ipc/bin/arm-2009q3/bin/arm-none-linux-gnueabi-
50 ATBM_WIFI__EXT_CCFLAGS = -DATBM_WIFI_PLATFORM=17
51 arch = arm
```

在同文件在底下，需要修改 PLATFORM_OTHER 的值

```
331 ATBM_WIFI__EXT_CCFLAGS = -DATBM_WIFI_PLATFORM=12
332 endif
333 ifeq ($(platform),PLATFORM_AMLOGIC_S805)
334 export
335 ATBM_WIFI__EXT_CCFLAGS = -DATBM_WIFI_PLATFORM=13
336 endif
337 ifeq ($(platform),PLATFORM_ROCKCHIP_3229)
338 export
339 ATBM_WIFI__EXT_CCFLAGS = -DATBM_WIFI_PLATFORM=10
340 endif
341 ifeq ($(platform),PLATFORM_XIONGMAI)
342 export
343 ATBM_WIFI__EXT_CCFLAGS = -DATBM_WIFI_PLATFORM=21
344 endif
345 ifeq ($(platform),PLATFORM_OTHER)
346 export
347 ATBM_WIFI__EXT_CCFLAGS = -DATBM_WIFI_PLATFORM=20
348 endif
349 export
350 include $(src)/Makefile.build.kernel
351 endif
```

d) 通过 make menuconfig 配置 atbm_wifi 驱动支持的相关配置

➔ 请参考文档的【一.2)】：配置驱动。

e) 通过平台相关编译方式编译得到 atbm 的驱动 ko 文件。

f) 编译出来的驱动有点大 需要 strip 缩小体积

Arm-linux-xxx-strip --strip-debug atbm_wifixxx.ko

3 出错调试信息&解决

3.1 编译出错

在编译驱动时，有可能出现编译限制等级较为严格导致出错。

```
  ^
  CC [M]  drivers/net/wireless/atbm_HS_svn950/hal_apollo/pm.o
drivers/net/wireless/atbm_HS_svn950/hal_apollo/atbm_platform.c:90:9: note: #pragma message: xunwei
 #pragma message(PLATFORMINF)
         ^
drivers/net/wireless/atbm_HS_svn950/hal_apollo/atbm_platform.c:403:2: error: implicit declaration of function 'EXYNOS4_GPX2' [-Werror=implicit-function-declaration]
  .irq_gpio = EXYNOS4_GPX2(4),
  ^
drivers/net/wireless/atbm_HS_svn950/hal_apollo/atbm_platform.c:403:2: error: initializer element is not constant
drivers/net/wireless/atbm_HS_svn950/hal_apollo/atbm_platform.c:403:2: error: (near initialization for 'platform_data.irq_gpio')
drivers/net/wireless/atbm_HS_svn950/hal_apollo/atbm_platform.c:404:2: error: implicit declaration of function 'EXYNOS4_GPC1' [-Werror=implicit-function-declaration]
  .power_gpio = EXYNOS4_GPC1(1),
  ^
drivers/net/wireless/atbm_HS_svn950/hal_apollo/atbm_platform.c:404:2: error: initializer element is not constant
drivers/net/wireless/atbm_HS_svn950/hal_apollo/atbm_platform.c:404:2: error: (near initialization for 'platform_data.power_gpio')
cc1: some warnings being treated as errors
make[5]: *** [drivers/net/wireless/atbm_HS_svn950/hal_apollo/atbm_platform.o] Error 1
make[5]: *** Waiting for unfinished jobs....
drivers/net/wireless/atbm_HS_svn950/hal_apollo/hwio_usb.c: In function 'atbm_before_load_firmware':
```

修改 kernel/Makefile

```
KBUILD_CFLAGS   += $(call cc-option,-Wno-stringop-truncation)
KBUILD_CFLAGS   += $(call cc-option,-Wno-sizeof-pointer-memaccess)
KBUILD_CFLAGS   += $(call cc-option,-Wno-array-bounds)
KBUILD_CFLAGS   += $(call cc-option,-Wno-packed-not-aligned)

# disable warnings in gcc 7.2.1
#KBUILD_CFLAGS   += $(call cc-option,-Wno-switch-unreachable)
#KBUILD_CFLAGS   += $(call cc-option,-Wno-misleading-indentation)

# use the deterministic mode of AR if available
KBUILD_ARFLAGS := $(call ar-option,D)

#KBUILD_CFLAGS   += -Werror

include scripts/Makefile.kasan
include scripts/Makefile.extrawarn
include scripts/Makefile.ubsan

# Add any arch overrides and user supplied CPPFLAGS, AFLAGS and CFLAGS as
# last assignments
KBUILD_CPPFLAGS += $(ARCH_CPPFLAGS) $(KCPPFLAGS)
KBUILD_AFLAGS   += $(ARCH_AFLAGS)   $(KAFLAGS)
KBUILD_CFLAGS   += $(ARCH_CFLAGS)   $(KCFLAGS)

# Use --build-id when available.
LDFLAGS_BUILD_ID = $(patsubst -Wl$(comma)%,%,\
                             $(call cc-ldoption, -Wl$(comma)--build-id,)
KBUILD_LDFLAGS_MODULE += $(LDFLAGS_BUILD_ID)
```

如果注释上面的宏还不行的话，就需要按照下面的一个个修改。

```
                  -include $(srctree)/include/linux/kconfig.h

KBUILD_CPPFLAGS := -D__KERNEL__

KBUILD_CFLAGS   := -Wall -Wundef -Wstrict-prototypes -Wno-trigraphs \
                   -fno-strict-aliasing -fno-common \
                   -Wno-format-security \
                   -fno-delete-null-pointer-checks
#-Werror-implicit-function-declaration
KBUILD_CFLAGS_KERNEL :=
```

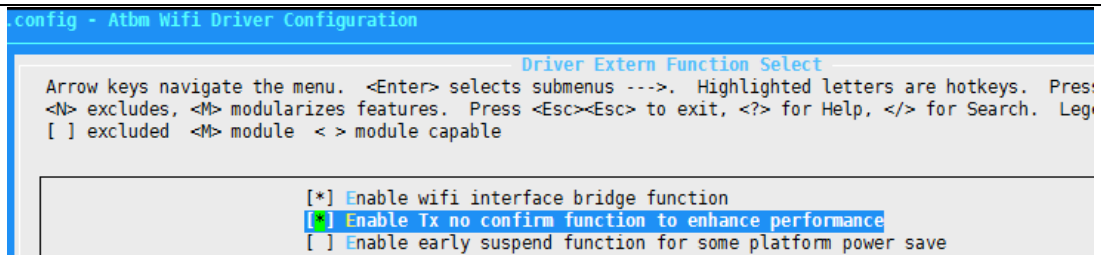
类似警告导致 error 的问题，类似修改。

3.2 加载出错

(1) NO_CONFIRM 宏没配置对导致出错

```
:CAPABILITIES_SHARE_CRYSTAL        [0]
:CAPABILITIES_HW_CHECKSUM           [0]
:CAPABILITIES_SINGLE_CHANNEL_MULRX [1]
:CAPABILITIES_CFO_DCXO_CORRECTION   [0]
:LMAC SET CAPABILITIES_NO_CONFIRM <ERROR>
-[ cut here ]------------
at bfd62b14 [verbose debug info unavailable]
ror: Oops - BUG: 0 [#1] PREEMPT SMP THUMB2
```

解决办法需要在一开始配置驱动时候打开对应的宏，如果打开了就给关闭。



3.3 扫描 AP 个数少

(1) 扫描状态返回-110

Log 如下图,这种可能是内核做了微小的修改以后没有再重新编译驱动以后导致的。

一般是修改 CONFIG_HZ 这个参数的值。

```
V380-linux# iwlist p2p0 scan | grep SSID
[   43.167634] [atbm_log]:atbm_hw_scan:if_id(1)
[   43.171134] [atbm_log]:atbm_hw_scan:scan, delay suspend
[   43.178162] [atbm_log]:scan start band(0),(14)
[   43.757991] [atbm_log]:Timeout waiting for scan complete notification.
[   43.763360] [atbm_log]:wsm_stop_scan_confirm 0 wait_complete 1
[   43.771316] [atbm_log]:atbm_scan_work:end(1)
[   43.774466] [atbm_log]:if_id = -1
[   43.774533] [atbm_log]:if_id = -1
[   43.774580] [atbm_log]:if_id = -1
[   43.777490] [atbm_log]:hw_priv->scan.status -110
                    ESSID:"B"
                    ESSID:"macro-video"
                    ESSID:"HUAWEI-10EC3B_Wi-Fi5"
                    ESSID:"YanFa-06090"
                    ESSID:"HUAWEI-10EC3B"
                    ESSID:""
                    ESSID:"YLGJ"
                    ESSID:"360A"
                    ESSID:""
                    ESSID:"Xiaomi_F2D4"
                    ESSID:"360_123"
                    ESSID:"gongcheng"
                    ESSID:"wifi"
                    ESSID:"ceshi02"
                    ESSID:"B_Wi-Fi5"
                    ESSID:"DIRECT-273F1225"
                    ESSID:"MV15106437"
V380-linux# [8373bf5a9d5b35b15c6f46963d89c75d]
```

(2) 扫描状态正常但是扫描的 AP 数量少，并且发现前几个信道的 ap 很少或者没有

出现这种问题先查一下 cfg80211 的配置，在内核的 net/wireless/scan.c 查看如下参数配置：

```
#define IEEE80211_SCAN_RESULT_EXPIRE (15 * HZ)
```

正常 IEEE80211_SCAN_RESULT_EXPIRE 配置为 15 * HZ 或者 30 * HZ,太短会导致扫描到的 ap 个数少

3.4 添加详细的反汇编信息方法

需要在 Makefile.build.kernel 里面添加上-g 编译参数

```
#################################################################################
#               Makefile For Kernel
#################################################################################

NOSTDINC_FLAGS := -I$(src)/include/ \
        -include $(src)/include/linux/compat-2.6.h \
        -DCOMPAT_STATIC


KBUILD_CFLAGS += -Wno-error \
        -Wno-error=vla \
        -Wno-error=unused-function -g
```

这样子编译出来的驱动，执行：

```
objdump -S atbm603x_wifi.ko > atbm603x_wifi.s
```

信息就会很详细

```
Disassembly of section .text:

00000000 <atbm_timer_handle>:
#if (LINUX_VERSION_CODE >= KERNEL_VERSION(4, 14, 0))
static inline void atbm_timer_handle(struct timer_list *in_timer)
#else
static inline void atbm_timer_handle(unsigned long data)
#endif
{
       0:       e1a0c00d        mov     ip, sp
       4:       e92dd800        push    {fp, ip, lr, pc}
       8:       e24cb004        sub     fp, ip, #4
#if (LINUX_VERSION_CODE >= KERNEL_VERSION(4, 14, 0))
        struct atbm_timer_list *atbm_timer = from_timer(atbm_timer, in_timer, timer);
#else
        struct atbm_timer_list *atbm_timer = (struct atbm_timer_list *)data;
#endif
        BUG_ON(atbm_timer->function == NULL);
       c:       e590301c        ldr     r3, [r0, #28]
      10:       e3530000        cmp     r3, #0
      14:       0a000002        beq     24 <atbm_timer_handle+0x24>
        atbm_timer->function(atbm_timer->data);
      18:       e5900020        ldr     r0, [r0, #32]
      1c:       e12fff33        blx     r3
      20:       e89da800        ldm     sp, {fp, sp, pc}
      24:       e7f001f2        .word   0xe7f001f2

00000028 <ieee80211_tasklet_handler>:
              BSS_CHANGED_ERP_PREAMBLE |
              BSS_CHANGED_ERP_SLOT;
}

static void ieee80211_tasklet_handler(unsigned long data)
```

如果编译的时候没有添加-g 参数或者驱动经过了 strip 那么执行：

```
objdump -S atbm603x_wifi.ko > atbm603x_wifi.s
```

可读性就比较差。

```
Disassembly of section .text:

00000000 <atbm_timer_handle>:
       0:       e1a0c00d        mov     ip, sp
       4:       e92dd800        push    {fp, ip, lr, pc}
       8:       e24cb004        sub     fp, ip, #4
       c:       e590301c        ldr     r3, [r0, #28]
      10:       e3530000        cmp     r3, #0
      14:       0a000002        beq     24 <atbm_timer_handle+0x24>
      18:       e5900020        ldr     r0, [r0, #32]
      1c:       e12fff33        blx     r3
      20:       e89da800        ldm     sp, {fp, sp, pc}
      24:       e7f001f2        .word   0xe7f001f2

00000028 <ieee80211_tasklet_handler>:
      28:       e1a0c00d        mov     ip, sp
      2c:       e92ddff0        push    {r4, r5, r6, r7, r8, r9, sl, fp, ip, lr, pc}
      30:       e24cb004        sub     fp, ip, #4
      34:       e24dd024        sub     sp, sp, #36     ; 0x24
      38:       e3a03000        mov     r3, #0
      3c:       e24b5038        sub     r5, fp, #56     ; 0x38
      40:       e1a07000        mov     r7, r0
      44:       e50b3030        str     r3, [fp, #-48]  ; 0xfffffffd0
      48:       e50b5038        str     r5, [fp, #-56]  ; 0xfffffffc8
      4c:       e50b5034        str     r5, [fp, #-52]  ; 0xfffffffcc
      50:       e10f0000        mrs     r0, CPSR
      54:       e3802080        orr     r2, r0, #128    ; 0x80
      58:       e121f002        msr     CPSR_c, r2
      5c:       e3a02001        mov     r2, #1
      60:       e50b3044        str     r3, [fp, #-68]  ; 0xffffffbc
      64:       e1a06003        mov     r6, r3
      68:       e2873e1a        add     r3, r7, #416    ; 0x1a0
      6c:       e50b3048        str     r3, [fp, #-72]  ; 0xffffffb8
```

3.5 编译的时候显示详细的编译信息

```
make V=1
```

CONTACT INFORMATION

AltoBeam (China) Inc.
Address: B808, Tsinghua Tongfang Hi-Tech Plaza, Haidian, Beijing, China 100083
Tel: (8610) 6270 1811
Fax: (8610) 6270 1830
Website: www.altobeam.com
Email: support@altobeam.com

DISCLAIMER