泰芯 Linux WiFi SMAC 驱动开发指南

泰芯半导体
TaiXin Semiconductor

珠海泰芯半导体有限公司
TaiXin Semiconductor Co., Limited

珠海市高新区港湾一号科创园港 11 栋 3 楼

修订记录

日期	版本	描　　述	修订人
2023/8/31	v1.3	修改固件下载错误说明	DY
2022/8/4	v1.2	修改蓝牙配网开发说明	DY
2022/7/27	v1.1	新增蓝牙配网自定义协议说明	DY
2022/6/23	v1.0	初始版本	DY

版权所有侵权必究
Copyright © 2022 by TaiXin Semiconductor All rights reserved

目录

版权所有侵权必究
Copyright © 2022 by TaiXin Semiconductor All rights reserved

1 概述

泰芯 SMAC 驱动需要内核支持 MAC80211/CFG80211，WiFi 协议栈运行在主控端。支持 ieee80211 bgn 协议,支持 AP/STA/P2P 模式。对应的 WiFi 固件为 vx.x.x.1 类型。

2 Linux Kernel 编译配置

SMAC 驱动支持 SDIO 和 USB 两种接口，在编译 kernel 需要打开以下功能：

1. 根据选择使用的接口（sdio/usb），打开对应的支持模块
   (1) sdio 接口：打开 *Device Driver → MMC/SD/SDIO card support*，以及对应的 mmc host driver。
   (2) usb 接口：打开 *Device Driver → USB support*，以及对应的 usb host driver。

注：编译驱动时如果出现 mmc_card_disable_cd 未定义 error，请打开 if_sdio.c 中 mmc_card_disable_cd 定义代码，如下图所示：

```
5:
6: #define FUNC_DEV(f)     (&((f)->dev))
7:
8: #define SDIO_CAP_IRQ(func)       ((func)->card->host->caps & MMC_CAP_SDIO_IRQ)
9: #define SDIO_CAP_POLL(func)      ((func)->card->host->caps & MMC_CAP_NEEDS_POLL)
0: #define HOST_SPI_CRC(func, crc) (func)->card->host->use_spi_crc=crc
1:
2: //#define mmc_card_disable_cd(c)  (1)
3: #define hgic_card_disable_cd(func)      mmc_card_disable_cd((func)->card)
4: #define hgic_card_set_highspeed(func)   mmc_card_set_highspeed((func)->card)
5: #define hgic_host_is_spi(func)          mmc_host_is_spi((func)->card->host)
6: #define hgic_card_cccr_widebus(func)    ((func)->card->cccr low_speed && !(func)->card->cc
7: #define hgic_card_cccr_highspeed(func) ((func)->card->cccr high_speed)
8: #define hgic_host_highspeed(func)       ((func)->card->host->caps & MMC_CAP_SD_HIGHSPEED)
9: #define hgic_host_supp_4bit(func)       ((func)->card->host->caps & MMC_CAP_4_BIT_DATA)
0: #define hgic_card_highspeed(func)       mmc_card_highspeed((func)->card)
1: #define hgic_func_rca(func)             ((func)->card->rca)
2: #define hgic_card_max_clock(func)       ((func)->card->cis.max_dtr)
```


版权所有侵权必究
Copyright © 2022 by TaiXin Semiconductor All rights reserved

3 WiFi Driver 开发使用说明

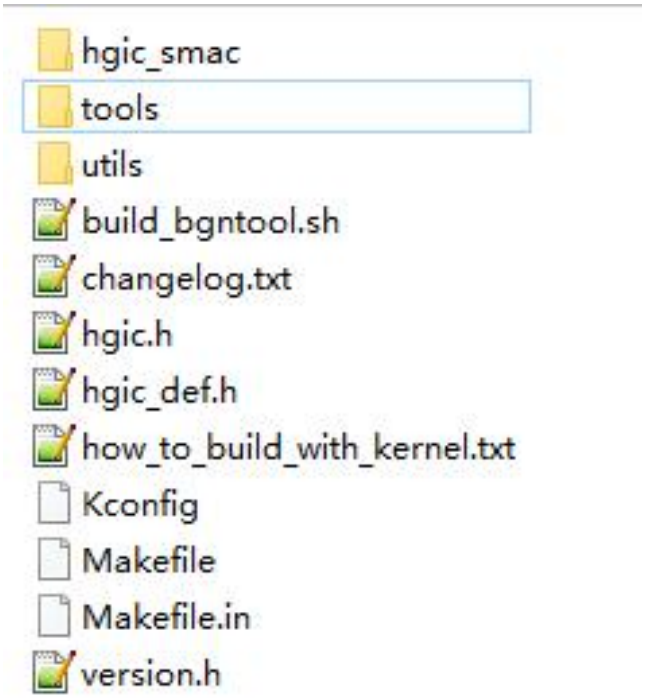
3.1 hgic_smac

SMAC 驱动需要内核支持 MAC80211/CFG80211。AP 模式需要使用 hostapd，STA 模式需要使用 wpa_supplicant。hostapd/wpa_supplicant 均可自行从 Internet 下载编译。

hgic_smac 驱动适用于 TXW801/TXW811 系列芯片，不支持低功耗模式。可编译成独立 ko 文件，也可以编译进内核。

3.1.1 hgic_smac 驱动文件

hgic_smac 驱动以源码形式发布，用户自行编译：



其中 tools 目录提供了常用工具：hostapd-2.9/wpa_supplicant-2.9/iw-5.0。smac 驱动对 hostapd/wpa_supplicant 的版本无要求，可以使用其它版本。

版权所有侵权必究
Copyright © 2022 by TaiXin Semiconductor All rights reserved

3.1.2 hgic_smac 编译说明

编译 smac 驱动需要修改 Makefile 文件添加编译工具链信息。

Makefile 文件已经添加部分主控编译参数，请自行添加方案所使用的主控的编译参数。

```
#Hi3518
#ARCH := arm
#COMPILER := arm-hisiv300-linux-uclibcgnueabi-
#LINUX_KERNEL_PATH := /home/matt/Hi3518/hi3518/linux-3.4.y

#FH8852
#ARCH := arm
#COMPILER := arm-fullhan-linux-uclibcgnueabi-
#LINUX_KERNEL_PATH := $(CURRENT_PATH)/../linux-3.0.8
#CFLAGS += -DFH8852 -DCONFIG_NO_ZERO_PACKET
```

编译 smac 驱动时可以选择支持 sdio 或者 usb 接口。执行 make 命令可以查看编译说明。

```
usage:
    make smac     : compile SMAC driver. support sdio/usb interface.  generate hgics.ko
    make smac_usb : compile SMAC driver. only support usb interface.  generate hgics.ko
    make smac_sdio: compile SMAC driver. only support sdio interface. generate hgics.ko

    make fmac     : compile FMAC driver. support sdio/usb interface.  generate hgicf.ko
    make fmac_usb : compile FMAC driver. only support usb interface.  generate hgicf.ko
    make fmac_sdio: compile FMAC driver. only support sdio interface. generate hgicf.ko

    make clean
```

3.1.3 hgic_smac 加载说明

加载 smac 驱动: insmod hgics.ko。

加载驱动时，可以根据需要指定以下参数：

- **fw_file**: 指定 WiFi 固件名称。未指定该参数时驱动默认加载 hgics.bin。
  - insmod hgics.ko fw_file=xxxx.bin　　#固件放在内核的加载目录
  - insmod hgics.ko fw_file=/usr/app/xxxx.bin　#固件放在自定义目录

  具体说明参见 3.2 固件下载章节【如果选择了自定义目录，则和 3.2 章节无关】。

版权所有侵权必究
Copyright © 2022 by TaiXin Semiconductor All rights reserved

3.1.4 hgic_smac proc fs 接口

除了可以常规使用 hostapd/wpa_supplicant 程序之外，hgic_smac 驱动也提供 proc fs 接口。应用程序通过 proc fs 接口可以与驱动/固件进行交互。

proc fs 接口文件存放在/proc/hgics/目录下。

1. **/proc/hgics/fwevnt** （只读）

   该文件用于接收驱动消息。在驱动无消息时访问该接口会被阻塞 100ms。

   tools/test_app/hgics.c 提供了示例代码。

   如果应用程序未处理驱动的 event， 当 event 队列已满时则会产生 warning 打印，如下图所示。如果应用程序确定不需要处理 event，屏蔽驱动的打印即可。

```
[   16.785532] hgics_bgnops_init:193::Leave
[   16.817675] hgics_create_procfs:239::enter
[   16.821872] hgics_create_procfs:247::leave
[   16.825978] hgics_core_init:932::ok
[   16.829397] hgics_delay_init:1033::hgics core init done!
[   16.834718] hgics_delay_init:1044::Leave, ret=0, soft_fc=0
[   17.438342] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   17.469791] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   17.511987] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   17.518982] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   17.590768] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   17.719231] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   17.733918] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   17.871067] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   17.993118] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   18.027238] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   18.101576] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   18.110420] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   18.123453] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   18.130445] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   18.168367] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   18.175350] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   18.182293] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   18.204687] hgics_rx_fw_event:66::event list is full (max 16), new event:34
[   18.211700] hgics_rx_fw_event:66::event list is full (max 16), new event:34
```

2. **/proc/hgics/status** （只读）

   cat /proc/hgics/status 可以查看驱动运行状态。包括固件信息，驱动数据缓存信息。

3. **/proc/hgics/iwpriv** (读写)

   该文件是泰芯提供的 hgpriv.c/iwpriv.c 工具交互接口，用于驱动/固件参数设置。具体参见 3.1.5 章节。

版权所有侵权必究
Copyright © 2022 by TaiXin Semiconductor All rights reserved

3.1.5 hgic_smac 辅助模块

除了可以正常使用 hostapd/wpa_supplicant 应用程序外，smac 驱动提供了/proc/hgics/iwpriv 接口，通过该接口程序可以与 WiFi 驱动/固件进行交互。

为了方便使用/proc/hgics/iwpriv 接口，smac 驱动在 tools/test_app 目录下提供了辅助 API 文件：iwpriv.c/hgics.c/hgpriv.c。

应用程序可以集成这些代码，直接通过 API 方式与 WiFi 驱动/固件进行交互。

其中 hgics.c 可以作为 smac 驱动的服务程序，接收处理驱动的 event 数据，与驱动/固件进行交互。

说明：tools/test_app/iwpriv.c 封装了所有的驱动接口，但是 smac 固件只支持其中部分 API。

3.1.6 hgic_smac 调试与测试

1. **接口测试**

   hgic_smac 提供了 sdio/usb 接口测试模式，可对 sdio/usb 接口进行稳定性测试。在 insmod hgics.ko 时指定 if_test 参数可以启动接口测试。

   - insmod hgics.ko if_test=1 #启动接口单向测试，测试数据从 host 端发送给 WiFi 芯片
   - insmod hgics.ko if_test=2 #启动接口双向测试，测试数据双向传输。

2. 调试信息

   WiFi 驱动可以导出 WiFi 芯片调试信息输出到主控端打印。

   在不方便使用 WiFi 芯片串口时可以使用此方法查看 WiFi 芯片调试信息。

   - hgpriv wlan0 set dbginfo=1　　//打开 WiFi 调试信息输出
   - hgpriv wlan0 set dbginfo=0　　//关闭 WiFi 调试信息输出

3.1.7 hgic_smac BLE 配网

请参考文档《泰芯 Linux WiFi SMAC 驱动 BLE 配网开发指南》。

版权所有侵权必究
Copyright © 2022 by TaiXin Semiconductor All rights reserved

3.1.8 hgic_smac 添加接口

hgic_smac 驱动可以支持 2 个 wlan 接口，但是初始化时内核只创建了一个 wlan 接口。有 2 种方式创建 2 个 wlan 接口：

1. **修改内核代码**：这个方式更简单

   打开内核文件 net/mac80211/main.c，找到 ieee80211_register_hw 函数。该函数在初始化时只添加了一个接口，修改添加第二个接口即可。

   如下所示，copy 添加一行代码即可创建 2 个接口：

```
958:     rtnl_lock();
959:
960:     result = ieee80211_init_rate_ctrl_alg(local,
961:                                           hw->rate_control_algorithm);
962:     if (result < 0) {
963:         wiphy_debug(local->hw.wiphy,
964:                     "Failed to initialize rate control algorithm\n");
965:         goto fail_rate;
966:     }
967:
968:     /* add one default STA interface if supported */
969:     if (local->hw.wiphy->interface_modes & BIT(NL80211_IFTYPE_STATION)) {
970:         result = ieee80211_if_add(local, "wlan%d", NULL,
971:                                   NL80211_IFTYPE_STATION, NULL);
972:         result = ieee80211_if_add(local, "wlan%d", NULL,
973:                                   NL80211_IFTYPE_STATION, NULL);
974:         if (result)
975:             wiphy_warn(local->hw.wiphy,
976:                        "Failed to add default virtual iface\n");
977:     }
978:
979:     rtnl_unlock();
980:
981:     local->network_latency_notifier.notifier_call =
982:         ieee80211_max_network_latency;
983:     result = pm_qos_add_notifier(PM_QOS_NETWORK_LATENCY,
984:                                  &local->network_latency_notifier);
985:     if (result) {
986:         rtnl_lock();
987:         goto fail_pm_qos;
988:     }
989:
990: #ifdef CONFIG_INET
991:     local->ifa_notifier.notifier_call = ieee80211_ifa_changed;
992:     result = register_inetaddr_notifier(&local->ifa_notifier);
993:     if (result)
994:         goto fail_ifa;
```

2. **使用 iw tools**：需要编译 iw tools

   执行 iw 命令：iw phy phy0 interface add wlan1 type station

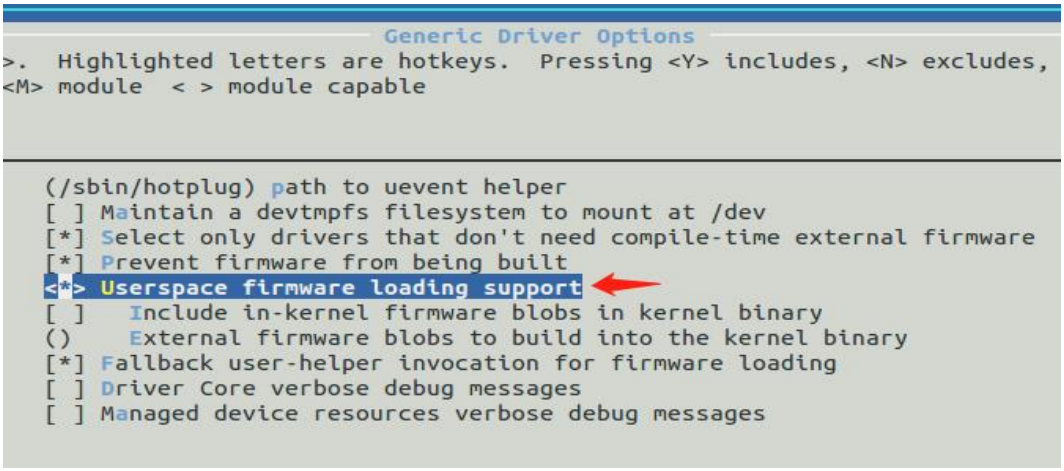
版权所有侵权必究
Copyright © 2022 by TaiXin Semiconductor All rights reserved

3.2 固件下载

WiFi 模块支持通过 SDIO/USB 接口下载固件运行。固件下载功能与 Linux 系统有关，不同版本的 Linux kernel，使用方式可能会有差异。WiFi 固件要放在系统支持的固件加载目录下，通常是/lib/firmware。如果固件加载遇到问题，请检查以下几个方面：

1. kernel 编译配置

kernel 需要支持 CONFIG_FW_LOADER（Devices Drivers->Generic Driver Options）



❖ 修改该选项后，WiFi 驱动需要重新编译。

2. 查看 kernel 支持的固件目录

查看 firmware_class.c 文件中的 fw_path 数组。

```
/* 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"
};
```

注：

1. 部分旧版本 kernel 的 firmware_class.c 可能没有 fw_path 数组，直接忽略此项。
2. Android 系统默认的固件目录是/vendor/firmware 或 /etc/firmware 或 /firmware/image

3. busybox 支持 mdev

查看是否存在/sbin/mdev 文件。

4. 查看/proc/sys/kernel/hotplug 事件处理程序

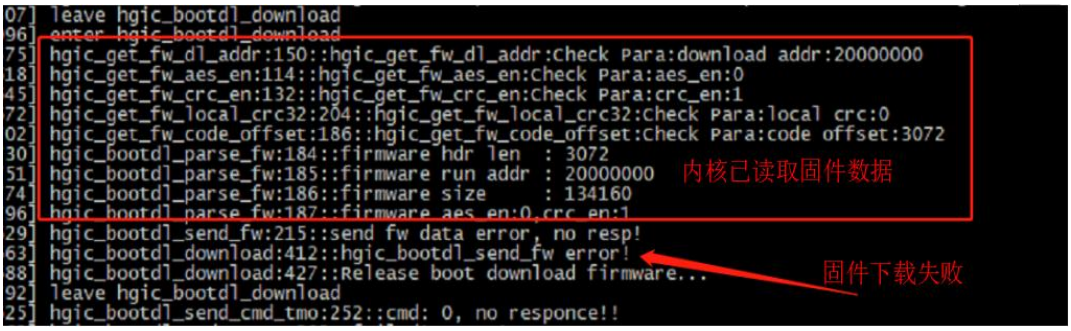
cat /proc/sys/kernel/hotplug 查看是否指定事件处理程序，如果未指定则需要在系统初始化时

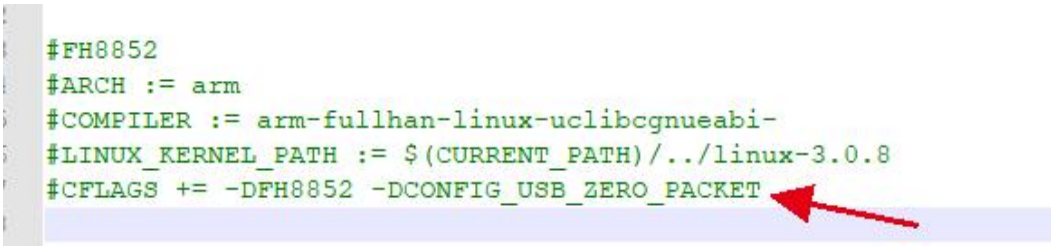
版权所有侵权必究
Copyright © 2022 by TaiXin Semiconductor All rights reserved

执行：echo /sbin/mdev > /proc/sys/kernel/hotplug

5. USB 接口发送空包

部分主控的 USB Host 不支持发送空包，会造成固件下载失败。如果遇到如下类似现象，请确认 USB Host 是否支持发送空包，或者尝试在 Makefile 添加 -DCONFIG_USB_ZERO_PACKET





6. fw_file 参数不能包含路径

7. 固件下载常见错误说明

(1) 提示找不到固件：请按上面的步骤进行确认，多数情况是因为固件位置不对，或者内核未打开支持固件加载功能。

(2) 固件数据下载返回的错误码

- 1：非法命令
- 2：非法地址（读写地址错误）
- 3：非法长度
- 4：没有权限（boot enter 密钥不对）
- 5：命令校验失败
- 6：数据失败（crc 校验出错）
- 7：通信超时

版权所有侵权必究
Copyright © 2022 by TaiXin Semiconductor All rights reserved

8. SDIO 接口最大包长

在固件下载阶段，WiFi 驱动默认设置的最大包长为 32704 bytes，部分主控的 SD Host 可能不支持发送这么长的数据。如果固件下载遇到问题，可以尝试修改最大包长，代码需要修改的位置如下图所示（**if_sdio.c**）：

```
    sdiodev->trans_cnt_addr = SDIO_TRANS_COUNT_ADDR;
    sdiodev->int_status_addr = SDIO_INIT_STATUS_ADDR;
    //sdiodev->status = SDIO_STATUS_STOP;
    sdiodev->bus.type = HGIC_BUS_SDIO;
    sdiodev->bus.drv_tx_headroom = SDIO_TX_HEADROM;
    sdiodev->bus.tx_packet = hgic_sdio_tx_packet;
    sdiodev->bus.tx_ctrl = hgic_sdio_tx_ctrl;
    sdiodev->bus.is_busy = hgic_sdio_check_busy;
#ifdef CONFIG_SDIO_REINIT
    sdiodev->bus.reinit = hgic_sdio_reinit;
#endif
    sdiodev->bus.bootdl_pktlen = 32704;
    sdiodev->bus.bootdl_cksum = HGIC_BUS_BOOTDL_CHECK_0XFD;
    sdiodev->bus.probe = probe_hdl;
    sdiodev->bus.dev_id = DEVID_ID(id);
    sdiodev->bus.blk_size = SDIO_BLOCK_SIZE;
    sdiodev->rx_retry = SDIO_CAP_IRQ(func) ? 0 : (max_pkt_len > 5 * HGIC_PKT_
    init_completion(&sdiodev->busy);

    if (!SDIO_CAP_POLL(func)) {
        set_bit(HGIC_BUS_FLAGS_NOPOLL, &sdiodev->bus.flags);
    }
```

v2.x

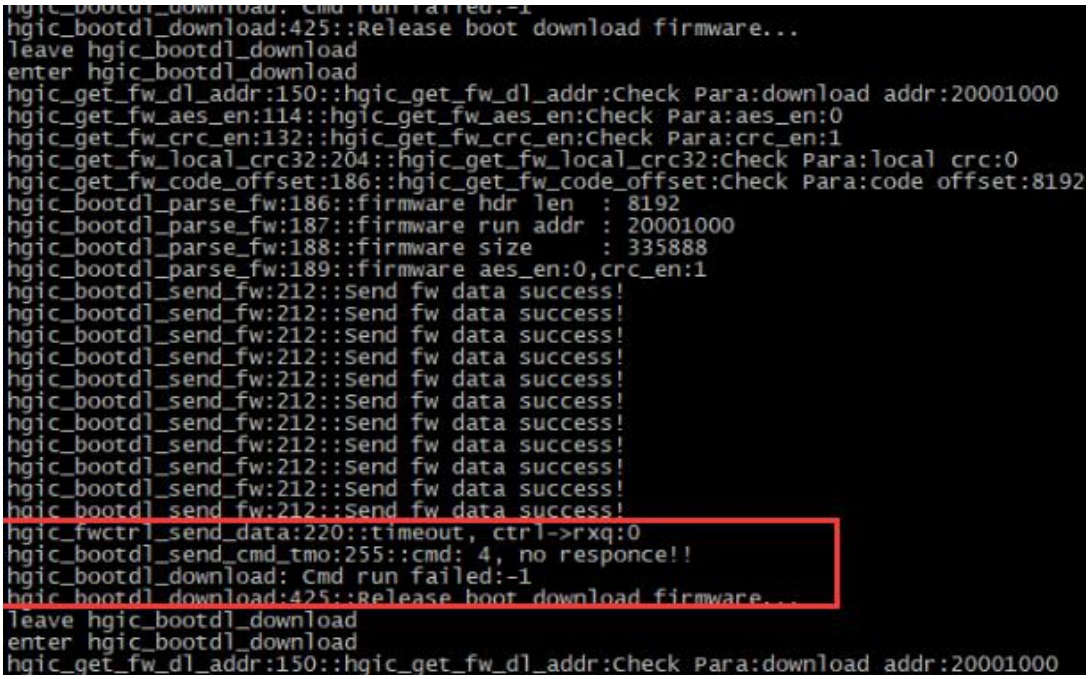
```
static struct hgic_bus hgic_ifbus_sdio = {
    .type = HGIC_BUS_SDIO,
    .drv_tx_headroom = SDIO_TX_HEADROM,
    .tx_packet = hgic_sdio_tx_packet,
    .reinit = hgic_sdio_reinit,
    .bootdl_pktlen = 32704,
    .bootdl_cksum = HGIC_BUS_BOOTDL_CHECK_0XFD,
};

static int hgic_sdio_enable(struct sdio_func_t *func)
```

v1.x

在固件下载时如果遇到如下图所示的错误：WiFi 驱动提示固件数据下载成功，但是 cmd 4 fail。可以尝试修改一下 sdio 最大包长。

版权所有侵权必究
Copyright © 2022 by TaiXin Semiconductor All rights reserved



产生此问题的原因可能和 sd host 所支持的最大 block count 有关，WiFi 驱动默认设置的 block size 是 64byte，最大包长也就是 512 个 block。所以加大 block size 应该也可以解决此问题。

版权所有侵权必究
Copyright © 2022 by TaiXin Semiconductor All rights reserved