泰芯半导体
TaiXin Semiconductor

TaiXin Semiconductor Linux WiFi FMAC Driver Development Guide

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

Liability and Copyright

Limitation of Liability

THIS DOCUMENT IS INTENDED FOR REFERENCE ONLY. Zhuhai Taixin Semiconductor Co,. Ltd (hereinafter referred to as "Taixin") does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Taixin reserves the right to make corrections, enhancements, and other changes to this document without notice.

Taixin assumes no liability for applications assistance or the design of customers' products. **Customers are solely responsible for the design, validation, and testing of its applications as well as for compliance with all legal, regulatory, and safety-related requirements concerning its applications.**

Taixin shall not be responsible for any damages, costs, losses, and/or liabilities arising out of customer's non-compliance with this section；concurrently, Customer will fully indemnify Taixin against any damages, costs, losses, and/or liabilities arising out of customer's non-compliance with this section.

Copyright Notice

Without the the written consent of Taixin, no party shall modify, adapt, alter, translate, or create derivative works from this document for commercial purposes.

Without the written consent of Taixin, no party shall disclose or distribute any or parts of the source code, SDK, binaries and object code mentioned in this document to any third party.

No party shall modify, reverse engineer, disassemble, decompile or otherwise attempt to discover the source code of any non-source code parts of the SDK including, but not limited to pre-compiled binaries and object code.

Furthermore, any other actions that may infringe upon Taixin's rights or other intellectual property owners are strictly prohibited.

For the subject who commits the above infringement, Taixin has the right to take necessary legal measure in accordance with the laws of the People's Republic of China or other applicable laws and international treaties, including but not limited to filing a lawsuit or arbitration against the infringer, or applying for legal compulsory measures.

Zhuhai Taixin Semiconductor Co., Ltd.

September 24,2024

Revision History

Date	Version	Revision Notes	Reviser
2024/5/5	V2. 18	Added explanation for mode3 of DCDC13.	WY
2024/4/18	V2. 17	Added interface of get signal； Modified description of low power consumption mode for AP (PS_mode4)； Revised explanation of wake-up reasons for AP low power consumption；	WY
2024/2/8	V2. 16	Modified description of low power consumption mode for AP； Added description of relay function. ； Added note stating this document only supports SDK version 1. x. ；	WY
2023/12/8	V2. 15	Added explanation for superpwr=2；	WY
2023/11/29	V2. 14. 1	Adjusted descriptions for country_region/scan/paired_stas.	WY
2023/11/27	V2. 14	Added description for country_region and modified scan command.	ZEL
2023/10/3	V2. 13	Modified explanations for wkreason and reassoc_wkhost.	WY
2023/9/20	V2. 12	Modified explanations for ps_mode and wakeup_io.	WY
2023/9/3	V2. 11	Modified description of pa_pwrctrl_dis.	WY
2023/8/31	V2. 10	Modified explanation of firmware download error messages.	DY
2023/7/28	V2. 9	Modified description of roaming.	DY
2023/5/23	V2. 8	Modified parameters explanation for multicast mode.	WY
2023/5/17	V2. 7	Modified description of roaming interface.	DY
2023/4/20	V2. 6	Added explanation for fwinfo interface. Modified description of radio_onoff interface.	DY
2023/4/13	V2. 5	Added explanation for get conn_state	DY
2023/4/7	V2. 4	Added get bssid	DY
2023/4/6	V2. 3	Modified description of roaming. Corrected typo in superpwr.	WY
2023/3/7	V2. 2. 1	Deleted parameters for several read commands.	WY
2023/2/17	V2. 2	Fixed description of sleep and wake-up. Added description of wakeup command.	WY

2023/2/6	V2. 1	Added description of dtim and autosleep. Added hyperlink for standby parameter.	WY
2023/1/12	V2. 0	Initial version.	DY

Table of Contents

1. Overview

The TaiXin Linux WiFi FMAC driver supports running the WiFi protocol stack inside the WiFi module. The host controller does not require a WiFi stack.

The FMAC driver supports AH modules and low power modes, with the corresponding WiFi firmware being of type v1. x. x. 5.

This document is only applicable to drivers adapted for SDK version 1. x; separate documentation is available for drivers adapted for SDK version 2. x.

2. Linux Kernel Compilation Configuration

The FMAC driver supports two interfaces: SDIO and USB. When compiling the kernel, the following features need to be enabled:

Depending on the interface being used (SDIO/USB), enable the corresponding support modules:

1） For SDIO interface: Enable “Device Driver → MMC/SD/SDIO card support” and the corresponding MMC host driver.

2） For USB interface: Enable “Device Driver → USB support” and the corresponding USB host driver.

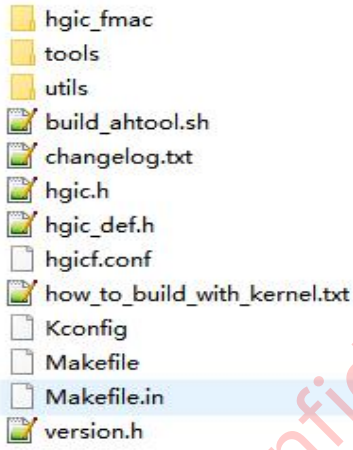
Note: If you encounter an mmc_card_disable_cd undefined reference error during driver compilation, please enable the definition of mmc_card-disable_cd in if_sdio. c as show in the following image:

3. WiFi Driver Development User Guide

3.1. hgic_fmac

3.1.1. hgic_fmac Driver Files

The hgic_fmac driver is released in source code form for users to compile on their own.



When compiling the fmac driver, you can choose to support either the sdio or usb interface. Execute the "make" command to view the compilation instructions.

```
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
```

The tools/test_app directory provides auxiliary tools and driver encapsulation APIs.

Execute build_ahtool. sh to compile the auxiliary tools, such as hgpriv and hgicf, etc.

3.1.2. hgic_fmac Loading Process

1. **Loading the fmac driver:** insmod hgicf. ko.

When loading the driver, the following parameters can be specified as needed:

- **ifname**: Specifies the network interface name. By default, the driver creates the hg0 interface (The latter defaults to hg0 as an example). For example: insmod hgicf. ko ifname="wlan%d". The driver will create the wlanx interface (x is the index obtained by %d).

- **conf_file**: Used to specify the loading parameter configuration file. The default value of this parameter is /etc/hgicf. conf. There are two ways to use it:
  1） The value of the conf_file parameter is a specific file path. For example: insmod hgicf. ko conf_file=/etc/hgicf. conf. In this case, the driver will load the specified parameter file, which is suitable for single NIC solutions.
  2） The value of the conf_file parameter is a directory path. For example: insmod hgicf. ko conf_file=/etc. In this case, the driver will read the /etc/hg0. conf parameter file in the /etc directory. This method is used for dual NIC solutions. For example, if two NICs, hg0 and hg1, are added in the same system, the driver will read /etc/hg0. conf and /etc/hg1. conf as parameters for hg0 and hg1, respectively.

- **fw_file**: Specifies the AH module firmware name. The driver defaults to loading hgicf. bin. See section 3. 3 Firmware Download for specific instructions.

insmod hgicf. ko fw_file=xxxx. bin

Note: The fw_file parameter value can only be a file name and cannot include a path.

2. **Configure the hg0 interface:** Set the IP address and bring the interface up.

3. **Use the hgpriv tool for parameter configuration (or use the encapsulated API).**

The most basic parameter settings are as follows:

- hgpriv hg0 set freq_range=9080, 9240, 8 #Set the operating frequency range and bandwidth for the AH module.
- hgpriv hg0 set bss_bw=8 #et the BSS bandwidth to 8M, which can be 1/2/4/8, consistent with the bandwidth above.
- hgpriv hg0 set tx_mcs=255 #Set the tx mcs to automatic mode.
- hgpriv hg0 set key_mgmt=NONE #Disable encryption.
- hgpriv hg0 set ssid=ah_test_ssid #Set the SSID
- hgpriv hg0 set mode=ap #Set the operating mode to ap

Parameter setting order: Set frequency/bandwidth information, set encryption method/password, set operating mode.

4. **Driver Configuration File [Optional]**

If quick loading of parameters at startup is needed, a parameter configuration file can be created for the hgicf driver. Set the conf_file parameter when loading the driver, and the driver will automatically load the set parameters.

The content of the parameter configuration file consists of parameters for the hgpriv command.

For example:

- hgpriv hg0 set mode=ap, corresponds to the content in the configuration file: mode=ap
- hgpriv hg0 set ssid=ap_ssid, corresponds to the content in the configuration file: ssid=ap_ssid

Example file:

```
freq_range=9080, 9240, 8
bss_bw=8
tx_mcs=25
acs=1, 10
key_mgmt=NONE
ssid=ah_test_ssid
mode=ap
```

3.1.3. hgpriv Configuration Guide

Note: It's recommended to use hgpriv tool only for manual input. For code integration, utilize the encapsulated APIs provided by the driver for enhanced convenience in application development.

- **Basic Networking Parameters**
  1. **hgpriv hg0 set mode=xx**

     Set the operating mode of the WiFi module: AP, STA, Group, or APSTA.
     - mode=ap: Operates in AP mode.
     - mode=sta : Operates in STA mode.
     - mode=group: Operates in broadcast group mode.
     - mode=apsta: Operates in relay mode. In this mode, the device functions both as an STA to connect to an upper-level AP and as an AP to provide connection services to other STAs. Use the r_ssid and r_psk commands to set the connection parameters for the upper-level AP.

     Note: Set other networking parameters first before setting the mode parameter.

  2. **hgpriv hg0 set ssid=xx**

Set the SSID of the AH module, up to a maximum of 32 characters.

3. **hgpriv hg0 set key_mgmt=xx**

   Set the encryption mode for the WiFi module:

   - NONE : Disables encryption.
   - WPA-PSK: Enables encryption. (Requires setting wpa_psk)

4. **hgpriv hg0 set wpa_psk=64_hexchar**

   Set the encryption key for the WiFi module. The key consists of 64 hex characters. This character string can be generated using the wpa_passphrase tool. For example:

   The part marked in red is the generated key.

   Note: If you don't want to use wpa_passphrase, you can generate 64 hex characters between 0 and f yourself.

5. **hgpriv hg0 set pairing=xx**

   Set the start/stop of the WiFi module's one-touch pairing function, which is used to control 2 WiFi modules for automatic pairing.

   - pairing=1 : Starts pairing
   - pairing=0 : Stops pairing

   For detailed operation instructions, refer to the one-key pairingsection.

6. **hgpriv hg0 set bss_bw=xx**

   Set the BSS bandwidth at which the WiFi module operates, valid values are: 1/2/4/8

7. **hgpriv hg0 set chan_list=freq0, freq1, . . . , freqN**

   Sets the list of operating frequencies for the WiFi module, allowing customization of the operating frequency points. Non-consecutive frequency points can be set, with units in 0. 1MHz. Up to 16 frequency points are supported.

8. **hgpriv hg0 set freq_range=start, end, bw**

   Set the operating frequency range of AH module, the unit of start and end is 0.

1MHz, the unit of bw is Mhz, for example, freq_range=9080, 9160, 8:

- 9080 : Start center frequency, 908M
- 9240 : End center frequency, 924M
- 8 : bss bandwidth, 8M [This value must match bss_bw.]

Note: The channels generated by this command are continuous without gaps. If non-consecutive channels are required, you can only use the following chan_list. If chan_list is used, the freq_range setting is ignored, as chan_list takes precedence over freq_range.

9. **hgpriv hg0 set country_region=xx**

   Set the country code for the AH module. Upon successful setting, the operating frequency points will be switched to comply with the standards of the corresponding country/region based on the bss_bw parameter. For detailed instructions, refer to the "TaiXin 802. 11AH Frequency Point Setting Guide" document.

   Currently, AH firmware supports frequency point settings for ten countries/regions: US (United States), EU (European Union), KR (South Korea), SG (Singapore), AU (Australia), NZ (New Zealand), ID (Indonesia), JP (Japan), MY (Malaysia), TH (Thailand), etc.

   Note: This command is mutually exclusive with the set chan_list and set freq_range commands. Using the latter two commands to set frequency points will clear the country code field.

10. **hgpriv hg0 set acs=1, 10**

    Enables the automatic frequency selection function for the WiFi module. When this function is enabled, the AH module will automatically select the frequency point with the least interference within the specified range as the operating frequency point. Typically used only in AP mode.

    Parameter Description:

    - 1 : Enables the function (0: disables this function).
    - 10: The frequency monitoring time of the automatic frequency selection function is 10ms, and this generally does not need to be adjusted.

- **Debugging Commands**

1. **hgpriv hg0 set loaddef=0/1**

- loaddef=1: Restores default parameters, erasing the parameter area in the WiFi module's flash memory. After successful execution of this command, the WiFi module will restart.
- loaddef=0: No reboot after executing restore parameters

2. **hgpriv hg0 set dbginfo=0/1**

   Enables or disables the output of firmware debugging information. When this function is enabled, debugging information from the firmware will be output to the WiFi driver and printed. This function is mainly used for capturing debugging information for problem analysis.

   - hgpriv hg0 set dbginfo=1 #Enable firmware debugging information output
   - hgpriv hg0 set dbginfo=0 #Disable firmware debugging information output

3. **hgpriv hg0 set sysdbg=type, 0/1**

   Enables or disables debugging information output for certain categories of firmware. Type represents the categories of debugging information as follows:

   - heap: Heap information (default = 0);
   - top: CPU usage information for each thread (default = 0).
   - wnb : WiFi protocol stack information (default = 0). ;
   - lmac: lmac layer information (default = 1).

4. **hgpriv hg0 set atcmd=at+xx**

   Sends AT commands to the module through the host controller.

5. **hgpriv hg0 scan=0/1/2**

   Executes this command in STA mode to scan for surrounding AP information.

   - scan=0: Stops scanning；
   - scan=1: Starts scanning (does not clear previously scanned AP list before startup, caches for 10 seconds). ；
   - scan=2: Starts scanning (clears previously scanned AP list before startup).

- **Status Query Commands**

1. **hgpriv hg0 get conn_state**

   Queries the connection state of the WiFi module. Returns: 0 or 9.

```
enum hgicf_hw_state {
    HGICF_HW_DISCONNECTED      = 0,
    HGICF_HW_DISABLED          = 1,
    HGICF_HW_INACTIVE          = 2,
    HGICF_HW_SCANNING          = 3,
    HGICF_HW_AUTHENTICATING    = 4,
    HGICF_HW_ASSOCIATING       = 5,
    HGICF_HW_ASSOCIATED        = 6,
    HGICF_HW_4WAY_HANDSHAKE    = 7,
    HGICF_HW_GROUP_HANDSHAKE   = 8,
    HGICF_HW_CONNECTED         = 9,
};
```

2. **hgpriv hg0 get module_type**

   Get information about the AH module type. Applications adaptively set relevant parameters based on the module type.

   utils/ah_freqinfo. c defines the frequency information/transmit power for each region, and the application can integrate this part of the code to adaptively set the frequency/transmit power parameters according to different regions.

   The command returns a 16-bit number, including two fields: Type (low 8 bits) and Saw (high 8 bits).

```
struct hgic_module_hwinfo{
    union{
        struct{
            unsigned char type;
            unsigned char saw:1, rev:7;
        };
        unsigned short v;
    };
};
```

   Type meanings:

   - 1: 700M module
   - 2: 900M module
   - 3: 860M module

   Saw meanings:

   - 0: Without saw
   - 1: With saw

3. **hgpriv hg0 get mode**

   Refer to "set mode" for the definition of mode.

4. **hgpriv hg0 get center_freq**

   Returns the center frequency of the currently used channel, in units of 100 kHz.

5. **hgpriv hg0 get sta_list**

   View currently connected sta information, aid, includes: mac address, ps_mode(module sleep mode), rssi, evm, tx_snr(rssi - bgrssi from the other device's stats, over-the-air feeds back), rx_snr(rssi - bgrssi statistics for this device)

   Executing this command on the AP side retrieves information about connected STAs.

   Executing this command on the STA side allows viewing information about the AP to which it is currently connected.

6. **hgpriv hg0 get scan_list**

   After executing the scan command, retrieves the list of scanned AP list.

7. **hgpriv hg0 get disassoc_reason**

   Get the reason for connection failure:

   - 0: Connection successful
   - 1: Incorrect password or SSID
   - 17: AP connection STA number is full
   - 93: Insufficient AP memory resources
   - 255: AP not found

8. **hgpriv hg0 get bgrssi=chan_index**

   Return the bgrssi corresponding to the specified chan_index.

   chan_index: Specifies the channel, starting from 1.

9. **hgpriv hg0 get bssid**

   In STA mode, queries the MAC address of the connected AP and returns the STA's AID. Output format: MAC, AIDFor example: 00: 11: 22: 33: 44: 55, 1

10. **hgpriv hg0 get signal**

    Gets the signal strength RSSI.

    Note: This interface can only be used by STAs; if used by an AP, the RSSI obtained will be inaccurate.

11. **hgpriv hg0 get fwinfo**

    Gets firmware information.

    Note: Manually executing this command will print garbled characters. Applications should use the encapsulated API: hgic_iwpriv_get_fwinfo to retrieve firmware information.

    Manually executing "cat /proc/hgicf/status" can display version information.

- **Networking Advanced Parameters**

1. **hgpriv hg0 set txpower=xx**

   Sets the maximum transmission power (unit: dBm, step: 1 dB), with a range of 6 to 20. Values outside this range will revert to the default value of 20.

2. **hgpriv hg0 set super_pwr=xx**

   Sets whether the AH module enables the super power feature.

   Enabling this feature increases the transmission power of the AH module to a maximum of 25 dBm during long-range communication. This feature is enabled by default in normal mode and disabled by default in AH test mode.

   - hgpriv hg0 set super_pwr=1 # Enables super power
   - hgpriv hg0 set super_pwr=0 #Enables super power, up to 25 dBm, supporting lower rates.
   - hgpriv hg0 set supper_pwr=2 #Enables super power, up to 22 dBm, supporting rates greater than 25 dBm but requires slightly closer proximity.

3. **hgpriv hg0 set tx_mcs=xx**

   Sets the mcs for TX of the AH module. Valid values are: 0/1/2/3/4/5/6/7/10.

   Setting other values indicates auto, the AH module will enable the tx rate auto-adjustment function.

The default value is 255, indicating automatic adjustment, which is generally kept as default.

4. **hgpriv hg0 set agg_cnt=xx**
   Sets the maximum frame aggregation count, with a default of 16. Setting a smaller aggregation count reduces average traffic, but setting it above 16 increases traffic fluctuations. It's recommended to keep the default value.

5. **hgpriv hg0 set max_txcnt=xx**
   Sets the maximum retransmission count of WiFi frames. "0" is invalid, and "N" indicates a maximum of N transmissions, with a default of 7 times.

6. **hgpriv hg0 set ap_hide=1**
   Enables the AP hiding feature, only valid in AP mode. After enabling AP hiding, sta will be unable to discover the AP through scanning.

7. **hgpriv hg0 set bss_max_idle=xx**
   Set the BSS max idle time in unit S.

   The STA must send a packet to the AP within the max idle time to maintain the connection. If the AP does not receive any information from the STA for longer than the max idle time, it considers the STA disconnected.

   Note: This parameter affects the sleep power consumption of STA. Smaller values increase power consumption. The default is 300 seconds, corresponding to 200uA@DTIM10.

8. **hgpriv hg0 set disassoc_sta=f8: de: 09: 96: 8f: 28**
   Disconnects a specified STA. AP can use this command to actively disconnect STA. However, in normal mode, disconnected STA will automatically reconnect.

9. **hgpriv hg0 set unpair=f8: de: 09: 96: 8f: 28**
   Unpairs the specified MAC from the module. This command can be executed on both the AP and STA sides.

10. **hgpriv hg0 set conn_paironly=1/0**
    After setting conn_paironly, the AP only allows STAs in the pairing list to connect.

STAs not in the pairing list cannot connect even with the correct SSID and password. Implement APs to restrict STA connections, with effects similar to whitelisting.

- conn_paironly=1 : Only allows STAs in the pairing list to establish connections.
- conn_paironly=0 : Allows all STAs to initiate connections, requiring the correct SSID and password. The default value is 0.

11. **hgpriv hg0 set paired_stas=mac1, mac2, . . .**

During pairing, the module automatically generates STA information, forming an STA list. If the module has flash memory, the STA list is saved to flash and automatically loaded after a module reboot. Without flash memory, the STA list cannot be saved, and is lost after a reboot.

For APs without flash memory, “set paired_stas” can be used to reset the pairing list after power-on. When paired_stas is set, and conn_paironly=1 is also set, only the specified STAs can connect to the AP, even if other STAs have the correct SSID and password, which can be achieved by restricting the connection of STAs by the AP, and the effect is similar to whitelisting.

The maximum number of STAs that can be set with this command is limited by the firmware's maximum supported number of STAs. MAC addresses are separated by commas. For example:

- iwpriv hg0 set paired_stas=00: 11: 22: 33: 44: 55, 00: 12: 34: 56: 78: 99
  The example setup STA list contains 2 STAs with MAC addresses 00: 11: 22: 33: 44: 55 and 00: 12: 34: 56: 78: 99.

It needs to set conn_paironly=1, then set paired_stas, and finally set ap/sta mode.

12. **hgpriv hg0 set pair_autostop=xx**

Sets whether the AH module automatically stops pairing after successful pairing.

By default, AH firmware does not automatically stop pairing, and requires manually executing hgpriv hg0 set pairing=0 to stop pairing.

- hgpriv hg0 set pair_autostop=1 #Enables automatic pairing stop.
- hgpriv hg0 set pair_autostop=0 #Disables automatic pairing stop.

13. **hgpriv hg0 set acktmo=xx**

Set the ack timeout value of AH module's WiFi protocol parameters, measured in

microseconds, defaulting to 0. This parameter needs to be set only when conducting communication over 1km. The calculation formula is 10 * (distance in kilometers - 1), for example, for a distance of 2km:

```
acktmo=10    #Increase ack timeout by 10us
```

14. **hgpriv hg0 set channel=xx**

    Set the channel index to fix a working frequency point for an AH module, starting from 1.

- **Low Power Related Parameters**

  1. **hgpriv hg0 set sleep=1**

     Control the WiFi module to enter sleep mode.

     - hgpriv hg0 set sleep=1 //Control the WiFi module to enter sleep mode
     - hgpriv hg0 set sleep=0 //Clear the sleep flag recorded by the WiFi driver. When entering sleep, the WiFi driver will record the current sleep status of WiFi, blocking all access to the WiFi module. Clear this flag to continue accessing the WiFi module.

  2. **hgpriv hg0 set ps_mode=xx**

     Set the sleep mode of the WiFi module:

     - 0: Sleep mode not set, with the same effect as mode 3.
     - 1: Module enters sleep mode and maintains connection with the server (the module maintains connection with the server by itself, requiring secondary development).
     - 2: Module enters sleep mode and maintains connection with the server (AP replaces the module to maintain connection with the server, lower power consumption than mode 1. Only AH modules support this mode, and only UDP protocol is supported).
     - 3: Module enters sleep mode and only maintains connection with AP. Any unicast packet can wake up the module (Hibernation power consumption is the same as mode 2).
     - 4: Module enters sleep mode and maintains connection only with AP. It can only be woken up by AP executing the wakeup command (Hibernation power consumption is the same as mode 2).
     - 6: Module enters sleep mode without maintaining connection, can only be woken up by pulling wakeup_io (firmware before 1. 6. x does not support).

3. **hgpriv hg0 set ap_psmode=1**

   When using AP low power consumption, both the AP and STA sides need to set this parameter to help the STA quickly reconnect to the AP when restarted.

   Note: When using AP low power consumption, this parameter needs to be set to 1 on both the AP and STA sides.

4. **hgpriv hg0 set wakeup_io=1, 0**

   Set the host wake-up IO and trigger edge to wake up the AH module from sleep: 0: Rising edge (positive pulse), 1: Falling edge (negative pulse). If this command is not set, the AH module will be woken up by default using mclr; according to the current plan, an IO in IOB2-IOB6 is generally considered.

   The corresponding IO numbers are: IOA0~31: 0~31, IOB0~7: 32~39, mclr: 51.

   Example:
   hgpriv hg0 set wakeup_io=1, 0
   Set the wake-up IO to IOA1, waking up on the rising edge.

5. **hgpriv hg0 set wkio_mode=xx**

   Set the working mode of AH wakeup Host IO (IOB0). Default is pulse mode.
   - 1: Level mode, maintains high level when WiFi is running normally, low level during sleep.
   - 0: Pulse mode, pulls up a 2ms pulse signal when the WiFi module wakes up the host controller.

6. **hgpriv hg0 set wkhost_reason=2, 7, 8, 12, 14**

   Set which wake-up reasons need to wake up the host controller. Refer to the explanation of wake reason below.

7. **hgpriv hg0 get wkreason**

   Query the wake-up reasons for the WiFi module(valid under the connected state).
   Meaning of return value:
   STA Wake-Up Reasons:
   - 1: Timer wake-up in non-connected state (this reason is generally not used to wake up the host controller).
   - 2: Unicast TIM wake-up, active wake-up by AP or unicast wake-up by other

devices(usually need to set wake-up main control)

- 3: Broadcast TIM wake-up, awakened by broadcast data (this reason has been deleted).
- 4: Key IO wake-up (default MCLR pin, MCLR pulls about 500us) (if pulled by the main control, there is no need to wake up the main control).
- 5: Wake on beacon loss(this reason is generally not used to wake up the main control)
- 6: AP restart detection wake-up (this reason is generally not used to wake up the main control).
- 7: Heartbeat timeout wake-up (used under sleep mode 2).
- 8: Wake-up packet wake-up (used under sleep mode 2).
- 9: MCLR IO reset wake-up (reset caused by MCLR being pulled for more than 2ms in sleep state, this reason is generally not used to wake up the main control).
- 10: : LVD low voltage reset (this reason is not set to wake up the main control by default).
- 11: PIR IO wake-up (requires special hardware circuit support, otherwise no need to set).
- 12: AP API wake-up, AP executes wnb_wakeup_sta or hgic_iwpriv_wakeup_sta(usually need to set wake-up)
- 13: During STA sleep, AP detects STA offline (this reason is generally not used to wake up the main control).
- 14: Wake up when connecting to the AP in STANDBY state (set after using the STANDBY function).

AP Wake-Up Reasons:

- 20: AP failed to enter sleep mode resulting in wake-up.
- 21: STA disconnection causing AP wake-up.
- 22: AP wake-up due to receiving STA data.
- 23: Perform a paired wake-up call

8. **hgpriv hg0 set dcdc13=xx**

Setting the AH module to use external 1. 3V DCDC supply or internal LDO supply.

- hgpriv hg0 set dcdc13=0 #Using internal LDO, no external DCDC supply is required. Normal power consumption and sleep power consumption are relatively high, suitable for scenarios where power saving is not needed. This

is usually the firmware default setting. When power saving is required, configure dcdc13 to one of the following modes via the interface.

- hgpriv hg0 set dcdc13=1 #Using external DCDC (hardware circuit must have 1. 3V DCDC) to power VDD13A and VDD13D. Normal power consumption and sleep power consumption are low (about 40% lower than LDO power supply). However, there may be RF performance risks (EVM issues) due to VDD13A being powered by an external DCDC. Consider using dcdc13=3. ；
- hgpriv hg0 set dcdc13=2 # Using external DCDC (hardware circuit must have 1. 3V DCDC) to power VDD13A and VDD13D during sleep. Low sleep power consumption (consistent with dcdc13=1). During normal operation, using internal LDO to power VDD13A and VDD13D (software increases the LDO output voltage), resulting in high normal power consumption, similar to LDO power supply. Due to the high normal power consumption, this mode is not recommended.
- hgpriv hg0 set dcdc13=3 #Using external DCDC (hardware circuit must have 1. 3V DCDC) to power VDD13D. Low sleep power consumption (consistent with dcdc13=1). In normal mode, VDD13A is powered by internal LDO, while VDD13D is still powered by external DCDC. Power consumption is lower than mode 2, but higher than mode 1 (about 20% lower than LDO power supply). This option can ensure RF performance while saving power to some extent. If using DCDC leads to poor RF EVM, consider using this option.

This parameter must be set according to the actual hardware circuit. Otherwise, if set to a non-zero value when there is no external DCDC, the device may fail to boot. If set to 0 but there is an external DCDC, power saving may not be achieved.

9. **hgpriv hg0 set pa_pwrctl_dis=xx**

Set the module's PA power control logic switch during sleep.

- hgpriv hg0 set pa_pwrctl_dis=0(default value), Does not disable PA power control logic. PA is not powered during sleep. This needs to be used in conjunction with hardware, meaning an external circuit controlled by IOA30 is added to power off the RF during sleep.
- hgpriv hg0 set pa_pwrctl_dis=1, Disables PA power control logic. PA is powered constantly.

The actual default value for this parameter is fine.

10. **hgpriv hg0 set dtim_period=xx**

    Set the DTIM period under WiFi module sleep mode, in milliseconds. WiFi will regularly wake up at the specified DTIM period to check the connection with the AP and receive AP wake-ups. DTIM is set on the AP side.

    DTIM10 is set to 1000 (default value), DTIM20 is set to 2000, and so on.

    This parameter affects SLEEP power consumption and wake-up time. A larger DTIM value reduces SLEEP power consumption but increases wake-up time. A smaller DTIM value increases SLEEP power consumption but decreases wake-up time.

    If you want to use a value less than 1000, meaning a keep-alive period of less than 1 second, besides setting dtim_period, you also need to set beacon_int reasonably. For specifics, consult with the FAE.

11. **hgpriv hg0 set autosleep_time=xx**

    Set the auto sleep time of the module, in seconds, defaulting to 10 seconds, with a maximum value of 32 seconds. Setting it to -1 turns off auto sleep. Setting it to 0 also represents the default value of 10 seconds.

    Auto sleep refers to the module entering sleep automatically if no commands are received from the host within the specified time after booting.

    Example:

        hgpriv hg0 set autosleep_time=20

        Sets the auto sleep time to 20 seconds.

12. **hgpriv hg0 set wait_psmode=0/1**

    Set the sleep mode under non-connected status, with 0 being ps connect (default) and 1 being standby mode. This is set on the STA side.

13. **hgpriv hg0 set ps_connect=60, 4**

    The STA's WiFi module will wake up to reconnect to the AP after disconnecting while in sleep mode. If the connection fails, the WiFi module will enter PS Connect mode, cycling through sleep/wake/reconnect phases. The intermediate sleep period helps prevent excessive power consumption due to continuous

reconnection attempts.

The default PS Connect behavior in the firmware is as follows: the sleep interval is 1 minute, incrementing by 1 minute for each subsequent failure up to n times.

- 1st connection failure: sleep for 1 minute
- 2nd connection failure: sleep for 2 minutes
- 3rd connection failure: sleep for 3 minutes
- 4th connection failure: sleep for 4 minutes

After incrementing the sleep time 4 times, it cycles back to the interval of the 1st attempt and repeats this pattern.

Example:

hgpriv hg0 set ps_connect=30, 4

Sets the PS Connect sleep interval to 30 seconds, with a maximum increment of 4 times.

14. **hgpriv hg0 set dis_psconnect=1**

    If you do not want the STA to automatically enter sleep mode when not connected to an AP, you can disable PS Connect mode. For example, in certain test scenarios, you may want the device to remain awake when not connected.

    Since the current sleep STA does not report information to the host after disconnection, it is recommended not to disable PS Connect mode for low-power devices under normal circumstances. Otherwise, if disconnection occurs, the device will continuously try to reconnect to the AP, and if it fails to reconnect, it will lead to rapid battery depletion.

15. **hgpriv hg0 set standby=chn_idx, wakeuptime**

    Sets the frequency channel (starting from 1) and wake-up interval (in milliseconds) for standby mode. This should be set on the STA.

16. **hgpriv hg0 set aplost_time=xx**

    Set the time (in seconds) for detecting AP loss when the module sleeps to stay alive. If no beacon is received from the AP within the specified time, the AP is considered lost. The default value is 10 seconds.

17. **hgpriv hg0 set wkdata_save=1**

    Set whether the module should save wake-up data sent by the server. After saving the wake-up data, the host can read the cached wake-up data from the module via

the /proc/hgic/wkdata_buff interface after booting up. The module can cache up to 4 wake-up data packets.

18. **hgpriv hg0 set reassoc_wkhost=1**

    When the module detects an AP anomaly while in sleep mode, it will restart scanning and connecting to the AP. By default, the module will not notify the host of the reconnection to the AP. Setting reassoc_wkhost=1 allows the module to notify the host after reconnecting to the AP. (This interface is currently deprecated; to set host wake-up, use wkhost_reason)

19. **hgpriv hg0 set heartbeat=ip_str, port, hb_interval, hb_tmo**

    Set the heartbeat server's IP address/port number/heartbeat interval/heartbeat timeout.

    Example:

    hgpriv hg0 set heartbeat=192. 168. 1. 1, 6000, 60, 300

    Sets the heartbeat server's IP address to 192. 168. 1. 1, port to 6000, heartbeat interval to 60 seconds, and heartbeat timeout to 300 seconds.

    This command can only be used to set parameters on the AH module when ps_mode=2. When ps_mode=2, before entering low-power sleep, the heartbeat application must send at least one heartbeat packet to the server. The WiFi module will capture the content of the heartbeat packet during transmission, and after entering SLEEP mode, the WiFi module will automatically send heartbeat packets to the server for keep-alive.

    When using this command on the TXW80x module, only the ip_str parameter is valid; other parameters will be ignored. For secondary development of the WiFi module, refer to the "TaiXin TXW80x Low Power Development Guide"

20. **hgpriv hg0 set heartbeat_resp**

    Set the content of the server's heartbeat response. This is used for response matching by the WiFi module to determine if the server's response has been received, indicating whether the heartbeat has been lost.

    This command can only be set using the API: hgic_iwpriv_set_heartbeat_resp_data.

This parameter only needs to be set when the AH module is in ps_mode=2.
The TXW80x module does not support this parameter.

21. **hgpriv hg0 set wakeup_data**

    Set the content of the wake-up packet sent by the server to wake up the device. This is used for wake-up packet content matching by the WiFi module to determine if it needs to wake up.

    This command can only be set using the API: hgic_iwpriv_set_wakeup_data

    This parameter only needs to be set when the AH module is in ps_mode=2.
    Use this command to set the wake-up data content for the TXW80x module. For secondary development of the WiFi module, refer to the "TaiXin TXW80x Low Power Development Guide".

22. **hgpriv hg0 set wkdata_mask**

    Set the wake-up packet content matching mask for when the server wakes up the device.

    This command can only be set using the API: hgic_iwpriv_set_wkdata_mask

    int hgic_iwpriv_set_wkdata_mask(char *ifname, int offset/*from IP hdr*/, char *mask, int mask_len)

    The mask is a bitmap that supports up to 8 bytes, allowing for wake-up data matching up to 64 bytes in length, starting from the IP header offset.
    For secondary development of the WiFi module, refer to the "TaiXin TXW80x Low Power Development Guide".

23. **hgpriv hg0 set wakeup=mac_addr**

    The host executes the wake-up command to wake up a specified MAC address of a STA.

    For example:

        hgpriv hg0 set wakeup=00: 11: 22: 33: 44: 55
        If the STA does not exist or is not in sleep mode, the return value will be -1.

- **Multicast Mode Parameters**

1. **hgpriv hg0 set join_group=group_addr, aid**

   When the WiFi module's operating mode is set to group, this command can be used to join a specific certain network. Once joined, the WiFi module will only receive data from that multicast network. All data communication will occur using the multicast address.

   If the operating mode is set to group but no multicast network is joined, all data communication will be broadcast.

   Note: The JOINGROUP command can only be set after the GROUP mode has been configured.

   Parameters:

   - group_addr: The address of the multicast network to join.
   - aid: The AID for this device in the multicast network. Valid AID values: 1~N (N being the maximum number of STAs supported by the firmware). Each device in the network should have a unique AID.
     - Setting a valid AID: The WiFi module will periodically send a heartbeat in the multicast network to announce its presence to other WiFi modules.
     - Setting an invalid AID: The WiFi module will not send heartbeats or notify other WiFi modules. If all devices have their AID set to 0, they will not be limited by the firmware's maximum STA count
     .

   Example:

   hgpriv hg0 set join_group=11: 22: 33: 44: 55: 66, 3

2. **hgpriv hg0 set mcast_txparam=dupcnt, tx_bw, tx_mcs, clearch**

   Set the TX parameters for multicast communication.

   - dupcnt: The number of times multicast data is retransmitted n. By default, it is sent only once. Setting this parameter will cause each multicast data packet to be retransmitted `n` times.
   - tx_bw: Set the transmission bandwidth for multicast data. :
   - tx_mcs: Set the mcs for multicast data.
   - Clearch: Set whether to clear the channel before sending multicast data. This parameter is currently ineffective and should be set to 0.

- **Relay Parameters**
  1. **hgpriv hg0 set r_ssid=xx**
     Set the SSID of the upper-level AP to which the relay mode will connect. The SSID can be up to 32 characters long. The usage requirements are the same as those for the `ssid` parameter.

  2. **hgpriv hg0 set r_psk=64_hexchar**
     Set the password for connecting to the upper-level AP in relay mode. The key value should be 64 hexadecimal characters. The usage requirements are the same as those for the `wpa_psk` parameter.

- **Roaming Parameters**
  1. **hgpriv hg0 set roaming=onoff, 0, threshold, rssi_diff, rssi_int**
     Enable or disable the roaming function. This command includes three parameters:

     - Parameter 1 onoff: Takes the value 0 (off) or 1 (on), indicating whether to enable the roaming function. It is disabled by default.
     - Parameter 2: Takes the value 0. This parameter is deprecated.
     - Parameter 3 threshold: Sets the signal strength threshold for roaming switching. The default is -60, meaning that the STA starts looking for a stronger AP when the AP's signal strength is below -60 dBm. This can generally be increased to -50.
     - Parameter 4 rssi_diff: Roaming detects the signal difference and reaches the difference value before the AP is considered suitable for switching. The default difference is 12db, which means that the signal strength of the newly discovered AP is 12db greater than the signal of the current AP before switching is considered necessary.
     - Parameter 5 rssi_interval: The roaming RSSI monitoring period, default is 5, representing 5 beacon intervals. This value affects the speed of roaming switching.

     The STA needs to set this parameter, while the AP does not.

- **Dual Antenna Parameters**
  1. **hgpriv hg0 set ant_auto=1**
     Enable dual-antenna automatic selection mode.

2. **hgpriv hg0 set ant_sel=0/1**

   Manually select either antenna 0 or antenna 1.

   When manually selecting, the automatic selection mode needs to be turned off.

- **Other Parameters**

1. **hgpriv hg0 set auto_chswitch=xx**

   Enable or disable the module's automatic channel switching feature (enabled by default). .

   - auto_chswitch=1 Enable
   - auto_chswitch=0 Disable

   The automatic channel switching feature means that when AP detects interference on the current channel during operation, it will automatically switches to another relatively clean channel. When the AP switches channels, it notifies the STA to switch as well. However, it cannot notify STAs that are in sleep mode. When the AP switches to another channel, sleeping STAs will detect the AP timeout, restart, and scan to reconnect to the AP. Once reconnected, they will return to sleep.

2. **hgpriv hg0 set auto_save=xx**

   Set whether the AH module automatically saves parameters (if the AH module is configured with nor flash).

   With auto-save turned off, only hgpriv hg0 save will save parameters (the parameter values for this command are saved immediately).

   - hgpriv hg0 set auto_save=0 #Disable auto-save, which needs to be used with the save command below.
   - hgpriv hg0 set auto_save=1 #Enable auto-save (default value).

3. **hgpriv hg0 save**

   This command has no parameters. The feature is to set the AH module save parameters(if the AH module is configured with nor flash).

   AH firmware automatically saves parameters by default; it detects parameter changes and saves them to flash. If auto_save is set to 0, you need to call this save

command to save parameters.

4. **hgpriv hg0 set dhcpc=1**
   Enable the module's internal DHCP Client function. The module will obtain an IP address upon booting, allowing the host controller to use the IP address directly when it starts, saving time for the host controller to apply for an IP address.

5. **hgpriv hg0 set reset_sta=mac_addr**
   Reset the remote AH module with the specified MAC address.

6. **hgpriv hg0 set radio_onoff=x**
   Control the wifi radio to turn it on or off.
   - hgpriv hg0 set radio_onoff=0 Turn off wifi radio.
   - hgpriv hg0 set radio_onoff=1 Turn on wifi radio.
   - hgpriv hg0 set radio_onoff=2 Turn on wifi radio RX, Turn off TX.

3.1.4. Proc fs Interfaces

The hgic_fmac driver provides proc fs interfaces, allowing applications to interact with the driver through proc fs.

The tools/test_app directory provides iwpriv. c, which offers a wrapped API that applications can integrate to directly interact with the driver using the API.

- **/proc/hgicf/status(Read-Only)**
  Running `cat /proc/hgicf/status` allows you to check the driver's operational status, including firmware information and driver data cache information.

  The interface returns version information as a string in the following format:

  1. 2. 0. 5-9566

  Meaning of each field:
  - 1: Major version number
  - 2: Minor version number
  - 3: Patch version number
  - 9556: Patch code version number (this version number increments and is unique)

- **/proc/hgicf/ota(Write-Only)**
  This is the module firmware OTA interface, allowing firmware upgrades for the module.

If the module does not have attached Flash and the firmware is downloaded for operation, this interface is invalid.

Usage:

1) Place the firmware firmware. bin in the /lib/firmware directory.
2) Execute echo -n firmware. bin > /proc/hgicf/ot to start the OTA upgrade process (alternatively, you can use the API: hgic_proc_ota). The entire process may take about 24 seconds (tested with firmware size of 320K).

If the module has attached Flash but it is empty, you can also use the OTA function to write the firmware. First, download the firmware through the interface to run it, and then use OTA to write it to Flash.

- **/proc/hgicf/fwevnt(Read-Only)**
  The driver event read interface, which can receive messages from the driver and firmware by cyclically reading the interface.

  Demo code is provided in the `tools/test_app/hgicf. c` file.

- **/proc/hgicf/iwpriv(Read-Only)**
  This is the driver parameter setting interface. The hgpriv. c and iwpriv. c files in the tools/test_app directory use this interface.

  The tools/test_app/iwpriv. c file provides a wrapped API for driver commands, allowing applications to directly invoke these commands via the API.

  The tools/test_app/hgpriv. c file is a manual command execution tool, which can be used to manually execute driver parameter commands.

3.1.5. Driver Event Messages

The WiFi module generates various event notifications to the host during its operation. By reading the /proc/hgicf/fwevnt interface, you can receive event messages generated by the firmware. The tools/test_app/hgicf. c file provides demo code for this. For detailed parsing of each event, please refer to hgicf. c.

Common event descriptions:

- HGIC_EVENT_SCANNING = 5
  Generated when the WiFi module enters scanning state.

- HGIC_EVENT_SCAN_DONE = 6
  Generated when the WiFi module completes scanning.

- HGIC_EVENT_TX_BITRATE = 7
  Generated every 5 seconds to report the current maximum transmission capability based on current wireless conditions. Applications can use this feedback to make adjustments, such as changing video bitrate.

- HGIC_EVENT_PAIR_START = 8
  Generated when the WiFi module enters pairing mode.

- HGIC_EVENT_PAIR_SUCCESS = 9
  Generated when the WiFi module successfully pairs. This event continues until pairing stops and reports the MAC address of the paired STA.

- HGIC_EVENT_PAIR_DONE = 10
  Generated when the WiFi module stops pairing. This event reports the MAC list of all currently paired STAs.

- HGIC_EVENT_CONECT_START = 11
  Generated when the WiFi module starts connecting.

- HGIC_EVENT_CONECTED = 12
  Generated when the WiFi module successfully connects.

- HGIC_EVENT_DISCONECTED = 13
  Generated when the WiFi module disconnects.

- HGIC_EVENT_SIGNAL = 14
  Generated when the WiFi module's wireless signal changes. This event reports the current RSSI and EVM information.

- HGIC_EVENT_REQUEST_PARAM = 16
  Generated when the WiFi module, acting as a STA, fails to connect to an AP. The

application can reset parameters in response to this event.

- HGIC_EVENT_CUSTOMER_MGMT = 19,
  Generated when the WiFi module receives a custom management frame. This event reports the content of the custom management frame.

- HGIC_EVENT_DHCPC_DONE = 21
  Generated when the WiFi module successfully executes a DHCP request to obtain an IP address.

- HGIC_EVENT_CONNECT_FAIL = 22
  Generated when the WiFi module fails to connect. This event reports the reason for the connection failure.

- HGIC_EVENT_CUST_DRIVER_DATA = 23
  Generated when the WiFi module sends custom driver data to the host.

- HGIC_EVENT_UNPAIR_STA = 24
  Generated when the WiFi module receives an unpairing request from the other side.

- HGIC_EVENT_EXCEPTION_INFO = 27
  Generated when the WiFi module reports an abnormal condition.

3.1.6. One-Key Pairing

The TaiXin FMAC driver supports one-key pairing for networking, simplifying the configuration of AP and STA network parameters.

When initiating pairing, the AP typically needs to set parameters such as SSID/password first. If no password is set, the AP will automatically generate a random password for each STA during pairing.

During the network configuration process, the STA will receive the SSID and password information from the AP. Additionally, both the AP and STA will record each other's MAC addresses, establishing a paired STA list.

If the module has flash memory, this information will be automatically saved to the flash.

If the module lacks flash memory, this information will be lost upon reboot. In this case, the

host can read and save the SSID/password/MAC address information once pairing is complete.

Notes:

1. After completing the one-key pairing, execute `set pairing=0` to stop pairing and enter the connection state.
2. One-key pairing supports pairing with only two devices simultaneously. For one-to-many network pairing, pair each STA device with the AP one by one. For example, to set up a network with one AP and four STAs, pair each of the four STA devices sequentially with the AP; do not attempt to pair all four STA devices simultaneously with the AP.
3. When the number of paired STA devices reaches the maximum, the AP will choose to overwrite the information of a disconnected STA. If no such STA is found, pairing will fail.

You can control the pairing process using the hgpriv command or the API: hgic_iwpriv_set_pairing.

- hgpriv hg0 set pairing=1 #Start pairing. Another device participating in the pairing must also enter pairing mode.
- hgpriv hg0 set pairing=0 #Stop pairing and exit pairing mode. If pairing is successful, the WiFi module will automatically establish a connection.

During the pairing process, the firmware will generate various events, which hgicf. c will read. Applications can handle these events, for example, by reading and saving SSID, password, and other information on the host side upon successful pairing.

By default, pairing does not automatically exit the pairing state after success. To configure automatic exit after successful pairing, set the pair_autostop parameter (see advanced networking parameters in hgpriv).

If pairing is unsuccessful, it will not automatically time out, so you need to manually execute hgpriv hg0 set pairing=0 to stop pairing.

3.1.7. STA Low Power Process

The TaiXin AH module supports a low-power SLEEP mode. Once in SLEEP mode, the AH module will close the interface with the host controller, resulting in access failures if the host attempts to communicate with the AH module.

There are two ways to use the SLEEP mode:

1. **Keep-Alive Only with AP During SLEEP**

When the SLEEP device does not need to maintain a connection with a remote server, it only needs to keep alive with the AP. In this case, the SLEEP process is simpler and follows these steps:

1) Enter Low Power Mode: hgpriv hg0 set sleep=1 execute the API: hgic_iwpriv_sleep

2) Wireless Wake-Up: Execute the API on the AP side to wake up the STA: hgic_iwpriv_wakeup_sta

3) Local Wake-Up: The host controller (or a button press) can wake up the AH module by pulling the MCLR pin of the AH module low. Ensure that the wake-up pulse is not less than 100uS and not more than 1ms, with the recommended value being 500uS.

4) WiFi Module Wakes Up the Host: Use the IOB0 pin to wake up the host. IOB0 has two working modes, with the default being pulse mode. In this mode, IOB0 generates a pulse signal by pulling high for 2ms to wake up the host. You can change the working mode of IOB0 using the hgpriv wkio_mode command.

2. Keeping Alive with Remote Server During SLEEP

When a device in SLEEP mode needs to keep alive with a remote server, the AH module will maintain heartbeat communication with the server. Follow these steps:

1) Set Heartbeat Server's IP Address, Port, Heartbeat Interval, and Timeout

Configure the AH module with the IP address and port number of the heartbeat server. The AH module will automatically capture the heartbeat packets during communication and, once in SLEEP mode, will maintain the heartbeat with the server on behalf of the device. If the heartbeat times out, the WiFi module will wake up and also wake up the host via IOB0.

Use the API : hgic_iwpriv_set_heartbeat to configure these settings. Example:
hgic_iwpriv_set_heartbeat("hg0", inet_addr("192. 168. 0. 1"), 60002, 60, 300);

This sets the heartbeat server IP to 192. 168. 0. 1, port number to 60002, heartbeat interval to 60 seconds, and heartbeat timeout to 300 seconds.

2) Set Heartbeat Response Content

Configure the AH module with the content of the heartbeat response from the server. During SLEEP mode, the AH module will identify the heartbeat response based on this content. If the AH module does not receive a response after sending a heartbeat, it will attempt again on the next wake-up. After seven consecutive missed responses, the module will notify the host of a network anomaly via IO.

Use the API : hgic_iwpriv_set_heartbeat_resp_data to set the heartbeat response content.

3) Set Wake-Up Packet Content from Heartbeat Server
Configure the AH module with the wake-up packet content from the heartbeat server. Upon receiving and recognizing this packet, the AH module will wake up the host via IO and wake up itself.

Use the API : hgic_iwpriv_set_wakeup_data to set the wake-up packet content.

4) Ensure Heartbeat Communication Before Entering SLEEP
The AH module captures heartbeat packets during communication. Therefore, ensure that at least one heartbeat packet is sent before entering SLEEP so that the AH module can capture it. If the AH module does not capture a heartbeat packet, it will not maintain the heartbeat with the remote server during SLEEP.

5) Remote Wake-Up
The remote server sends a wake-up packet to the device. The WiFi module receives and recognizes this packet, waking up the device.

6) Local Wake-Up
The host controller (or a button press) can wake up the WiFi module by pulling the MCLR pin of the WiFi module low. Ensure that the wake-up pulse is not less than 100uS and not more than 1ms, with the recommended value being 500uS.

7) WiFi Module Wakes Up the Host
The WiFi module uses IOB0 to wake up the host. IOB0 is in pulse mode by default and generates a 2ms high-level pulse signal to wake up the host. You can modify the working mode of IOB0 using the hgpriv wkio_mode command. After the host powers on, it can query the wake-up reason via the API hgic_iwpriv_get_wkreason.

3. Exception Handling Process

1) AP Abnormal Reboot:
When the WiFi module in Sleep mode detects that the AP has rebooted abnormally, the WiFi module will restart and reconnect to the AP. After a successful connection, it will automatically enter sleep mode after 10 seconds. During this process, the

module will not wake up the host. The module assumes the host is in sleep mode if no interaction commands are detected during this 10-second period. The auto-sleep time can be modified using the command: iwpriv hg0 set autosleep_time

2) AP Shutdown:
When the WiFi module in sleep mode loses connection due to the AP shutting down, it will restart and attempt to reconnect to the AP. If the reconnection fails, the WiFi module will enter either PS Connect mode or standby mode, which involves cyclic sleep/wake/reconnect processes.

Select whether to enter PS Connect mode (wait_psmode=0, default) or Standby mode (wait_psmode=1) by setting wait_psmode.

- PS Connect Mode: The default PS Connect behavior is as follows: the sleep interval starts at 1 minute and increases incrementally with each failure.
  - 1st failure: sleep for 1 minute
  - 2nd failure: sleep for 2 minutes
  - 3rd failure: sleep for 3 minutes
  - 4th failure: sleep for 4 minutes

  After four failures, the sleep interval cycles back to 1 minute and repeats to avoid excessive power consumption from frequent reconnections.

  After a successful connection, the module will automatically enter sleep mode after 10 seconds without waking up the host.

  To modify the PS Connect parameters, use: iwpriv ps_connect

  To disable PS Connect mode, use: iwpriv dis_psconnect

  Note: In PS Connect mode, the module restarts cyclically, leading to relatively high power consumption.

- Standby Mode: Standby mode is a new unconnected sleep mode that offers lower power consumption than PS Connect mode and allows the STA to reconnect to the AP more quickly once it powers on.

  The STA needs to configure the standby parameters: the sleep channel index

(starting from 1) and the wake interval (in milliseconds).

Below is the relationship between wake time and standby sleep current:

Wake Time (S)	1	3	5	10
Standby Sleep Current(uA)	500	300	250	200

The AP must specify the same sleep channel as the STA when powered on (using the set channelcommand).

It is advisable to set the standby channel to the same one used by the AP before it shut down (using the get center freq command).

3) Network Disconnection:
When the WiFi module or AP in Sleep mode detects a heartbeat timeout, it indicates a network anomaly. In this situation, the WiFi module will restart and wake up the host. Once the host is powered on, it should attempt to reconnect to the server to verify network connectivity. Control is then handed over to the host, which will decide the subsequent actions.

3.1.8. AP Low Power Mode Process

The Tai Xin AH module supports AP low power mode, which can be set by the host controller. After entering low power mode, the host can choose to stand by or power down to reduce overall power consumption.

In AP low power mode, the AH module does not need to be awakened through IO and can be awakened by reinitializing the interface.

To use AP low power mode, both the AP and STA need to be configured with hgpriv hg0 set ap_psmode=1, and the AP must be set to PS_MODE=4.

1. **General Process:**
   - **Steps to enter AP low power mode:**
     1) Close Data Communication: Before entering AP low power mode, close all data communication.
     2) Enter Low Power Mode: Execute the API: hgic_iwpriv_sleep("hg0", 1) to notify

the AH module to enter low power mode.

- **There are two ways to trigger the exit from AP low power mode:**

  1) Host Trigger: The host wakes up the module by sending a command through the interface.

  2) STA Trigger: When the STA sends data to the AP, it will trigger the AP to exit low power mode. Alternatively, the STA can use the hgic_iwpriv_wakeup_sta API to wake up the AP. After the AP receives the wake-up packet, the AH module is not fully awake yet; it will use IOB0 to wake up the host, which then fully wakes up the module through the interface. IOB0 is in pulse mode by default, generating a 2ms high pulse signal to wake up the host. The working mode of IOB0 can be modified using the `hgpriv wkio_mode` command. Refer to the instructions for using this command. Note that at this point, the module is not fully awake; the host needs to initialize the SDIO interface to fully wake up the module. After the host is powered on, it can use the API `hgic_iwpriv_get_wkreason` to query the wake-up reason and execute the corresponding process based on different reasons. For detailed explanations of wake-up reasons, refer to the get wkreason instructions.

2. **Exception Handling**

   - **STA Loses Connection after AP Enters Low Power Mode**

     In AP low power mode, the AP does not guarantee timely detection of the STA's disconnection status.

   - **AP Fails to Enter Low Power Mode**

     When entering low power mode, the AP will notify all connected STAs. If a STA is powered off or the signal is lost during this process, causing the AP to be unable to notify the STA, the AP will fail to enter low power mode. Upon failure, the AP will wake up the host. After the host is awakened, it can query the wake-up reason and notify the AH module to forcefully enter low power mode again.

   - **Currently, Simultaneous Low Power Sleep Mode for AP and STA is Not Supported**

3.1.9. Relay Function Instructions

Refer to the relevant interface settings for the relay function.

3.1.10. Roaming Function Instructions

The AH - STA mode supports roaming between multiple APs. The roaming function can be used as follows:

1. Enable Roaming: STA uses the command hgpriv hg0 set roaming to enable roaming. AP does not need to set roaming enable.
2. Set Roaming AP Parameters:
    1) Set the same SSID for each AP, or differentiate the last 3 characters of the SSID, e. g. , roaming_ssid_001, roaming_ssid_002.
    2) Set the same password for each AP.
3. If communication between roaming APs is required, they need to be interconnected using a wired network.

3.1.11. Driver Auxiliary Module

To simplify the application use of the fmac driver, the fmac driver package provides a driver auxiliary module: tools/test_app/iwpriv. c. This file implements encapsulated APIs for the hgpriv commands and proc interfaces, which can be directly integrated into applications. By accessing the fmac driver in the form of APIs, application development can be simplified. These encapsulated APIs parse the output information of the hgpriv/proc interfaces.

- **hgpriv Command Encapsulation API**

  The hgpriv command encapsulation API is shown below. These APIs are used in the same way as the corresponding hgpriv commands, but there are two differences:
  - A new ifname parameter is added: this parameter is the AH network interface name, such as "hg0".
  - The mac parameter is a 6byte char array, while the hgpriv command uses a MAC address string.

3.1.12. Firmware Exception Information

This section introduces debugging information for exceptions in EVENTS, which can be read via /proc/hgicf/fwevnt. The hgicf. c file contains sample code:

```
        case HGIC_EVENT_EXCEPTION_INFO:
            exp = (struct hgic_exception_info *)data;
            switch(exp->num){
                case HGIC_EXCEPTION_TX_BLOCKED:
                    printf("*wireless tx blocked, maybe need reset wifi module*\r\n");
                    break;
                case HGIC_EXCEPTION_TXDELAY_TOOLONG:
                    printf("*wireless txdelay too loog, %d:%d:%d *\r\n",
                            exp->info.txdelay.max, exp->info.txdelay.min, exp->info.txdelay.avg);
                    break;
                case HGIC_EXCEPTION_STRONG_BGRSSI:
                    printf("*detect strong backgroud noise. %d:%d:%d *\r\n",
                            exp->info.bgrssi.max, exp->info.bgrssi.min, exp->info.bgrssi.avg);
                    break;
                case HGIC_EXCEPTION_TEMPERATURE_OVERTOP:
                    printf("*chip temperature too overtop: %d *\r\n", exp->info.temperature.temp);
                    break;
                case HGIC_EXCEPTION_WRONG_PASSWORD:
                    printf("*password maybe is wrong *\r\n");
                    break;
            }
            break;
```

1. TXDELAY_TOOLONG
   - Print: *wireless txdelay too long, max : min : avg *
   - Description: tx delay is too long, reported when it exceeds 500ms.
2. STRONG_BGRSSI
   - Print: *detect strong backgroud noise. max : min : avg *
   - Description: Background noise is poor, exceeding -85dBm.
3. TEMPERATURE_OVERTOP
   - Print: chip temperature too overtop
   - Description: Overheating, exceeding 85 degrees Celsius.
4. WRONG_PASSWORD
   - Print: password maybe is wrong
   - Description: Disconnection due to encryption error on the other side, the host has an error prompt, and encryption-related parameters need to be checked.
5. TX_BLOCKED
   - Print: wireless tx blocked
   - Description: Transmission queue is blocked.

3.1.13. Interface Test Mode

hgic_fmac provides sdio/usbinterface test modes to perform stability tests on sdio/usbinterfaces. By specifying the `if_test` parameter when loading the `hgicf. ko` module, you can start interface testing.

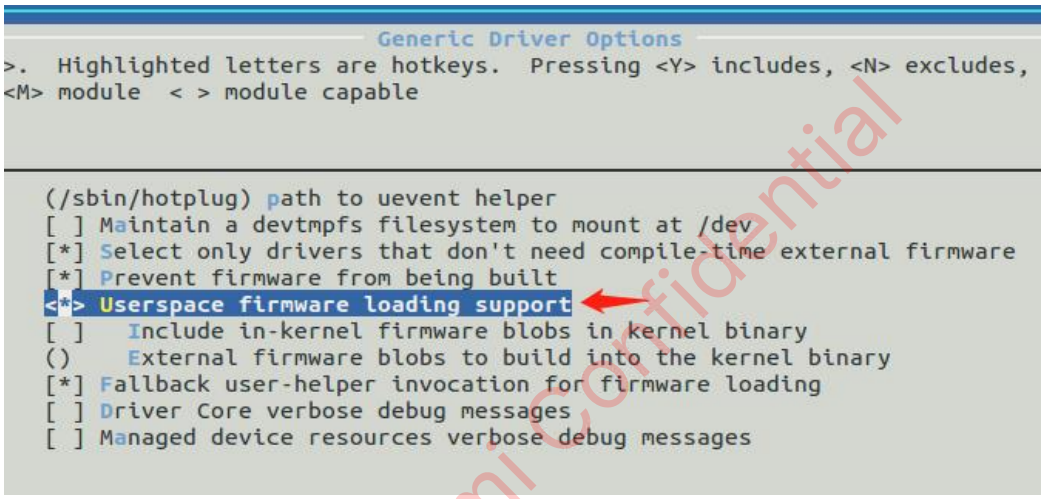
- insmod hgicf. ko if_test=1 # To start unidirectional interface testing, where data is sent from the host to the AH module
- insmod hgicf. ko if_test=2 #To start bidirectional interface testing, where data is transmitted in both directions

3.2. Firmware Download

The WiFi module supports firmware downloads via the SDIO/USB interface. The process for downloading firmware is dependent on the Linux system in use, and it may vary between different versions of the Linux kernel. The WiFi firmware should be placed in a directory supported by the system for firmware loading, typically /lib/firmware. If there are issues with loading the firmware, check the following aspects:

1. **Kernel Compilation Configuration**

   Ensure that the kernel supports CONFIG_FW_LOADER(Devices Drivers->Generic Driver Options)



   After modifying this option, the WiFi driver needs to be recompiled.

2. **Check the Kernel Supported Firmware Directory**

   Inspect the fw_path array in the firmware_class. c file.

```
/* 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"
};
```

   Notes:

   1) Some older kernel versions might not have the fw_path array in firmware_class. c.

   2) The default firmware directory for Android is /vendor/firmware or /etc/firmware or /firmware/image

3. **busybox Support for mdev**

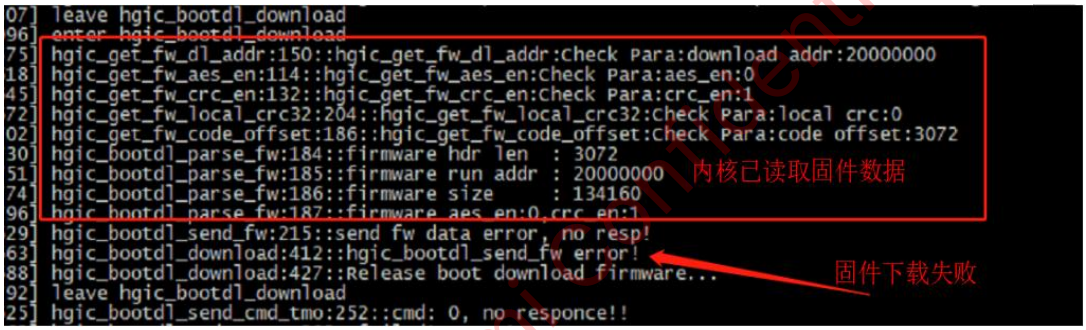
   Ensure the existence of the /sbin/mdev file.

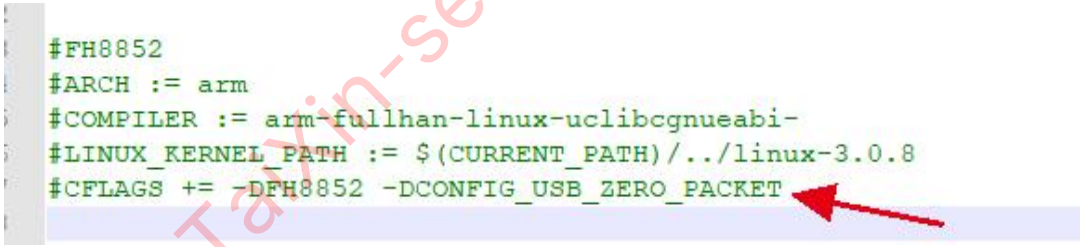
4. **Verify Event Handler for /proc/sys/kernel/hotplug**

   Check the event handler: cat /proc/sys/kernel/hotplug. If no handler is specified, set it during system initialization: echo /sbin/mdev > /proc/sys/kernel/hotplug

5. **USB Interface Sending Empty Packets**

   Some USB hosts do not support sending empty packets, which can cause firmware download failures. If you encounter issues similar to the ones described, verify USB Host support for empty packets or add -DCONFIG_USB_ZERO_PACKET to the Makefile.





6. **The fw_file parameter must not contain a path.**

7. **Firmware Download Common Errors Instructions**

   1) Firmware Not Found: f the system reports missing firmware, verify the steps outlined above. This issue is usually due to incorrect firmware placement or the kernel not turned on to support firmware loading.

   2) Error Code Returned by Firmware Data Download:

   - 1: Invalid command
   - 2: Invalid address (read/write error)
   - 3: Invalid length

- 4: No permission (incorrect boot enter key)
- 5: Command verification failure
- 6: Data failure (CRC check error)
- 7: Communication timeout

8. SDIO Interface Maximum Packet Length

During the firmware download phase, the WiFi driver sets the default maximum packet length to 32704 bytes. Some SD Hosts may not support such long packets. If you encounter issues, you can modify the maximum packet length in the `if_sdio. c` file. The code needs to be changed in the following locations:

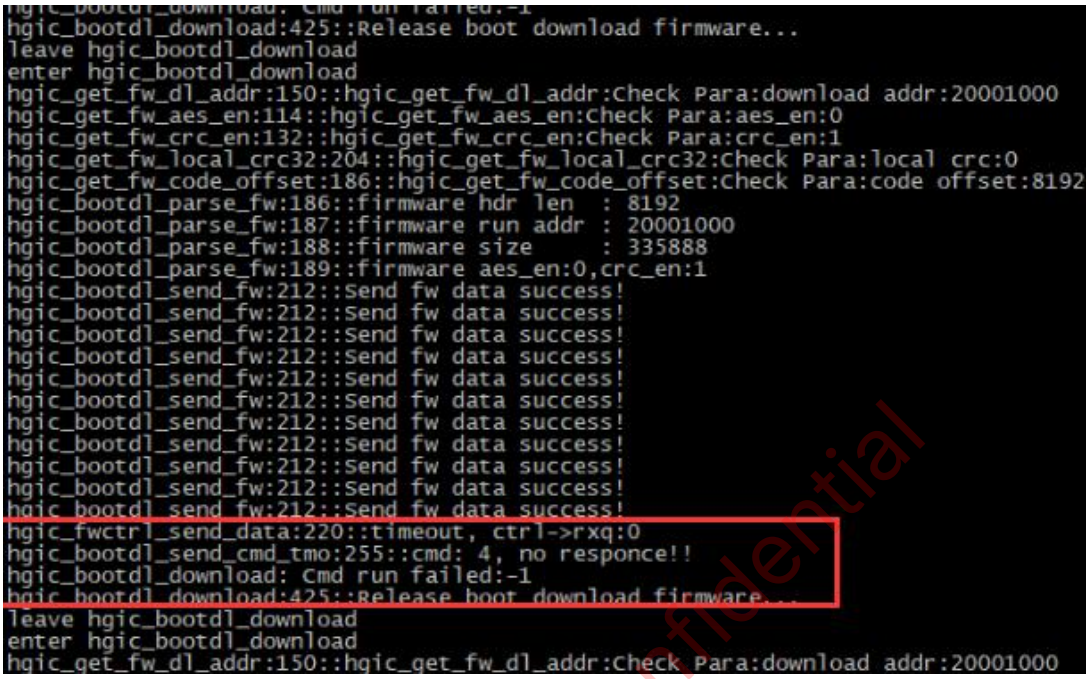
```
    sdiodev->trans_cnt_addr = SDIO_TRANS_COUNT_ADDR;
    sdiodev->int_status_addr = SDIO_INIT_STATUS_ADDR;
    //sdiodev->status = SDIO_STATUS_STOP;
    sdiodev->bus.type = HGIC_BUS_SDIO;
    sdiodev->bus.drv_tx_headroom = SDIO_TX_HEADROOM;
    sdiodev->bus.tx_packet = hgic_sdio_tx_packet;
    sdiodev->bus.tx_ctrl = 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

```
48:
49: static struct hgic_bus hgic_ifbus_sdio = {
50:     .type = HGIC_BUS_SDIO,
51:     .drv_tx_headroom = SDIO_TX_HEADROOM,
52:     .tx_packet = hgic_sdio_tx_packet,
53:     .reinit = hgic_sdio_reinit,
54:     .bootdl_pktlen = 32704,
55:     .bootdl_cksum = HGIC_BUS_BOOTDL_CHECK_0XFD,
56: };
57:
58: static int hgic_sdio_enable(struct sdio_func_t *func)
59: {
```

v1.x

If the firmware download encounters an error as shown in the following figure: WiFi driver prompts firmware data download successfully, but cmd 4 fails. it can try to modify the sdio max packet length.



The reason for this problem may be related to the maximum block count supported by the sd host, the default block size set by the WiFi driver is 64 bytes, and the maximum packet length is 512 blocks, so increasing the block size should also be able to solve this problem.