REALTEK

_____

Quick Guide for Wake on WLAN

Date: 2020/11/19

Version: 1.0

1. Release note

Document Version	Note
V0.9	1. First release
V1.0	1. Add preferred network offload(PNO) function

	usage method
	2. Document reformatting

2. Support list

- USB interface
  - 8188EU, 8188CU, 8192DU, 8192EU, 8723BU, 8812AU, 8821AU, 88x2BU, 8188FU, 8723DU, 8814AU, 8821CU, 8192FU, 88x2CU, 8725AU, 8814BU.
- SDIO interface
  - 8189ES, 8189FS, 8723BS, 8703C, 8192ES, 88x2BS, 8821AS, 8703BS, 8723DS, 8723CS, 8821CS, 8192FS, 88x2CS, 8725AS...
- PCI-E interface
  - 8812AE, 8821AE, 88x2BE, 8821CE, 8723BE, 8192EE, 8723DE, 8814AE, 8192FE, 88x2CE, 8814B

3. Requirements of wakeup via in-band and out-band methods

- In-band requirements
  - SDIO Interface
    - ✓ SDIO host MUST support remote wakeup feature.
    - ✓ SDIO data1 MUST be wakeup source in the host platform.
    - ✓ The platform MUST keep power to WiFi chip in suspend state.
    - ✓ The platform MSUT work fine between suspend and resume.
    - ✓
  - USB Interface
    - ✓ USB host MUST support remote wakeup feature.
    - ✓ The platform MUST keep power to WiFi chip in suspend state.
    - ✓ The platform MSUT work fine between suspend and resume.
  - PCI Interface
    - ✓ PCI host MUST support remote wakeup feature.
    - ✓ The platform MUST keep power to WiFi chip in suspend state.
    - ✓ The platform MSUT work fine between suspend and resume.
- Out-band requirements
  - SDIO、USB and PCI Interfaces
    - ✓ The GPIO of the **PLATFORM** MUST be wakeup source.
    - ✓ The platform MUST keep power to WiFi chip in suspend state.
    - ✓ The platform MSUT work fine between suspend and resume.
    - ✓ The WIFI module MUST have the GPIO wakeup pin.

4. Driver Configuration for Wake on WLAN

4.1 In-band configuration

If using **SDIO DATA1 pin** or **USB protocol D+/D- toggle** in-band method to wakeup the host, driver need to do is only switch **CONFIG_WOWLAN** from "n" to "y" in Makefile as Figure 1.

```
###################### Wake On Lan ######################
CONFIG_WOWLAN = y
#bit3: ARP enable, bit2: deauth, bit1: unicast, bit0: magic pkt.
CONFIG_WAKEUP_TYPE = 0xf
CONFIG_WOW_LPS_MODE = default
```

<div align="center">Figure 1</div>

4.2 Out-band configuration

If using out-band method, driver need to do is modify Makefile and config GPIO. The detail is as following

- Makefile Configuration
  - Switch **CONFIG_WOWLAN** and **CONFIG_GPIO_WAKEUP** from "n" to "y" as Figure 2.

```
###################### Wake On Lan ######################
CONFIG_WOWLAN = y
#bit3: ARP enable, bit2: deauth, bit1: unicast, bit0: magic pkt.
CONFIG_WAKEUP_TYPE = 0xf
CONFIG_WOW_LPS_MODE = default
#bit0: disBBRF off, #bit1: Wireless remote controller (WRC)
CONFIG_SUSPEND_TYPE = 0
CONFIG_WOW_STA_MIX = n
CONFIG_GPIO_WAKEUP = y
```

<div align="center">Figure 2</div>

- GPIO Configuration
  - If use the module package, please use the driver default value. The default value depends on HDK document.
  - If there is any customized requirement about modify WIFI GPIO number, please modiy the value of CONFIG_WAKEUP_GPIO_IDX in Makefile and **please contact with RTK technical support team first.**
  - User could use "proc" subsystem to modify the behavior of WIFI GPIO when receive wakeup up packet in non-suspend state.
    - ✓ wowlan_gpio_info to show WIFI wakeup host GPIO number and high_active value:
      $ **cat /proc/net/rtlxxxx/wlanX/wowlan_gpio_info**
    - ✓ modify high_active form 0 to 1 in wowlan_gpio_info:

$ echo 1 > /proc/net/rtlxxxx/wlanX/wowlan_gpio_info

high_active = 0 means pull low wake. (default)

high_active = 1 means pull high wake.

```
isaac@isaac-B33E:~$ cat /proc/net/rtl8723bu/wlan50/wowlan_gpio_info
wakeup_gpio_idx: 14
high_active: 0
isaac@isaac-B33E:~$ echo 1 > /proc/net/rtl8723bu/wlan50/wowlan_gpio_info
isaac@isaac-B33E:~$ cat /proc/net/rtl8723bu/wlan50/wowlan_gpio_info
wakeup_gpio_idx: 14
high_active: 1
```

Figure 3

4.3 Wake-up condition configuration

We divided the wake-up conditions into two categories based on the STA with or without a connection.

4.3.1 STA with a connection

■ CONFIG_WAKEUP_TYPE

- If the setting of Makefile is CONFIG_WAKEUP_TYPE = 0x7, it means that WOWLAN supports "deauth wake up", "unicast wake up" and "magic packet wake up". The detail description is bit0: magic pkt, bit1: unicast and bit2: deauth.

■ Setup the wake up pattern (**ONLY** support on driver version v5.1.0 or later)

- **iwpriv**

  iwpriv wlanX wow_set_pattern pattern=[pattern]

  Examples:

  wake up on any packets sent to MAC 00:E0:4C:01:F0:EE

  $ iwpriv wlanX wow_set_pattern pattern=00:E0:4C:01:F0:EE

  wake up when receive UDP packet dst port 5353

  $ sudo iwpriv wlan0 wow_set_pattern
  pattern=-:-:-:-:-:-:-:-:-:-:-:08:00:45:-:-:-:-:-:-:-:-:11:-:-:-:-:-:-:-:-:-:-:-:-:14:e9

- **echo pattern into wow_pattern_info**

  echo [pattern] > /proc/net/rtl8xxx/wlanx/wow_pattern_info

  Examples:

  wake up on any packets sent to MAC 00:E0:4C:01:F0:EE

  $ echo 00:E0:4C:01:F0:EE > /proc/net/rtl8xxx/wlanx/wow_pattern_info

  wake up when receive UDP packet dst port 5353

  $ echo -:-:-:-:-:-:-:-:-:-:-:-:08:00:45:-:-:-:-:-:-:-:-:-:11:-:-:-:-:-:-:-:-:-:-:-:-:-:14:e9 >
  proc/net/rtl8xxx/wlanx/wow_pattern_info

- **Pattern Format**

  The pattern begins with an 802.3 (Ethernet) header with the correct src/dest MACs base on IPv4. All of the following parameters are need to use **HEX format**. The more information is as following:

AA:AA:AA:AA:AA:AA:BB:BB:BB:BB:BB:BB:CC:CC:DD:-:-:-:-:-:-:-:E

E:-:-:FF:FF:FF:FF:GG:GG:GG:GG:HH:HH:II:II

A: Ethernet destination address

B: Ethernet source address

C: Ethernet protocol type

D: IP header VER + Hlen, use: 0x45 (4-is for ver. 4, 5 is for len. 20)

E: IP protocol

F: IP source address (192.168.0.1 → C0:A8:00:01)

G: IP destination address (192.168.0.4 → C0:A8:00:04)

H: Source port (1024: 04:00)

I: Destination port (1024: 04:00)

■ Clean wake up patterns (**ONLY** support on driver version v5.1.0 or later)

- **iwpriv**

  $ iwpriv wlanX wow_set_pattern clean

- **echo clean**

  $ echo clean > /proc/net/rtl8xxx/wlanx/wow_pattern_info

4.3.2 STA without a connection (PNO)

Support list: TBD

■ Makefile configuratuon

- Switch CONFIG_PNO_SUPPORT "n" to "y" as Figure 4.
- You can also switch CONFIG_PNO_SET_DEBUG "n" to "y" to get more debug logs (as Figure 4).


```
CONFIG_HIGH_ACTIVE_HST2DEV = n
CONFIG_PNO_SUPPORT = y
CONFIG_PNO_SET_DEBUG = y
CONFIG_AP_WOWLAN = n
```

Figure 4

- Fill wpa_supplicant.conf file path in driver code
  - In order for the driver to get the security protocols used by SSIDs, the path to wpa_supplicant.conf must be filled in in rtw_dev_nlo_info_set(). (As shown in Figure 5)

```
int rtw_dev_nlo_info_set(struct pno_nlo_info *nlo_info, pno_ssid_t *ssid,
                int num, int pno_time, int pno_repeat, int pno_freq_expo_max)
{

        int i = 0;
        struct file *fp;
        mm_segment_t fs;
        loff_t pos = 0;
        u8 *source = NULL;
        long len = 0;

        RTW_INFO("+%s+\n", __func__);

        nlo_info->fast_scan_period = pno_time;
        nlo_info->ssid_num = num & BIT_LEN_MASK_32(8);
        nlo_info->hidden_ssid_num = num & BIT_LEN_MASK_32(8);
        nlo_info->slow_scan_period = (pno_time * 2);
        nlo_info->fast_scan_iterations = 5;

        if (nlo_info->hidden_ssid_num > 8)
                nlo_info->hidden_ssid_num = 8;

        /* TODO: channel list and probe index is all empty. */
        for (i = 0 ; i < num ; i++) {
                nlo_info->ssid_length[i]
                        = ssid[i].SSID_len;
        }

        /* cipher array */
        fp = filp_open("/data/misc/wifi/wpa_supplicant.conf", O_RDONLY,  0644);
        if (IS_ERR(fp)) {
                RTW_INFO("Error, wpa_supplicant.conf doesn't exist.\n");
                RTW_INFO("Error, cipher array using default value.\n");
                return 0;
        } else {
                RTW_INFO("Open wpa_supplicant.conf successfully.\n");
        }
```

Figure 5.

- Usage method
  - Before the platform enters suspend state
    - ✓ Use wpa_cli enter cmd to enable pno
      $ sudo wpa_cli -iwlan0 set pno 1
  - After the platform wakes up from suspend state
    - ✓ Use wpa_cli enter cmd to disable pno
      $ sudo wpa_cli -iwlan0 set pno 0

5. The wake up reason table

The DUT could be waked up by the WIFI chip with the following reasons:

Reason Value	Description	Note
0x01	Receive pairwise key change packet.	
0x02	Receive group key change packet.	
0x04	Receive disassociate packet.	
0x08	Receive de-auth. Packet.	
0x10	AP power off, or could not receive AP's beacon in a period time	
0x21	Receive magic packet.	
0x22	Receive unicast packet.	The unicast packet included IP level.
0x23	Pattern Match	The device could be waked up by specific pattern.
0x55	PNO SSID Match	The device has PNO enabled.

6. wpa_supplicant Configuration for Wake on WLAN

- **The configuration file of wpa_supplicant should add "wowlan_triggers=any" when the driver adopt CFG80211 interface in the linux kernel. If there is no "wowlan_triggers=any", the CFG80211 module will send disconnect command to wifi driver and the wake on WLAN function will fail.**

Example:

```
ctrl_interface=/var/run/wpa_supplicant
#update_config=1
wowlan_triggers=any

#connect to open network
network={
  ssid="SSID"
  psk="12345678"
}
```

- **The wpa_supplicant need to restart again after the wifi device/driver remove and insert.**
  The below command can check the wowlan status of CFG80211:

```
//get the phy number mapping to wlan interface
$ iw dev
phy#135
     Interface wlan1
          ifindex 138
          type managed
phy#134
     Interface wlan20
          ifindex 137
          type managed

//get the wowlan status, below is correct
$ iw phy134 wowlan show
WoWLAN is enabled:
 * wake up on special any trigger
```

//below wowlan status is wrong
$ iw phy134 wowlan show
WoWLAN is disabled.