

---

# 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 MUST 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 MUST 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 MUST 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 MUST 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
```

Figure 1

### 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
```

Figure 2

#### ■ 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 modify 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

We divided the wake-up conditions into two categories based on the STA with or without 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 pkt (only TCP and ICMP/ICMPv6 unicast pkt are allowed to wake up in default setting)
  - ✓ bit2: deauth

- **iwpriv**

```
iwpriv wlanX wow_set_pattern pattern=[pattern]
```

```
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
```

[illegible]

- ```
echo [pattern] > /proc/net/r8xxx/wlanx/wow_pattern info
```

```
$ echo 00:E0:4C:01:F0:EE > /proc/net/rtnl8xxx/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
```

- 6

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:

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

```

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", 0_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.

Realtek