Wificonf Application Programming Interface

This document intends to provide a comprehensive guide to the implemented user interfaces for Wi-Fi station and AP mode configuration base on the functionalities provided by Realtek Wi-Fi driver.

Table of Contents

1 Introduction

This document intends to provide a comprehensive guide to the implemented user interfaces for Wi-Fi station and AP mode configuration base on the functionalities provided by Realtek Wi-Fi driver.

2 API Specific Data Types

2.1 WIFI_MODE_TYPE

```
typedef enum _WIFI_MODE_TYPE
{
	WIFI_MODE_STA			= 0,
	WIFI_MODE_AP
	WIFI_MODE_STA_AP,
	WIFI_MODE_PROMISC
} WIFI_MODE_TYPE;
```

The enumeration lists the supported operation mode by WIFI driver, including station and AP mode.

2.2 WIFI_SECURITY_TYPE

```
typedef enum _WIFI_SECURITY_TYPE
{
	WIFI_SECURITY_OPEN		= 0,
	WIFI_SECURITY_WEP		,
	WIFI_SECURITY_WPA		,
	WIFI_SECURITY_WPA2
}WIFI_SECURITY_TYPE;
```

The enumeration lists the possible security type to set when connection. Station mode supports OPEN, WEP and WPA2. AP mode support OPEN and WPA2.

2.3 WIFI_LINK_STATUS

```
typedef enum _WIFI_LINK_STATUS
{
	WIFI_LINK_DISCONNECTED	= 0,
```

```
        WIFI_LINK_CONNECTED
}WIFI_LINK_STATUS;
```

The enumeration lists the status to describe the connection link.

2.4 WIFI_COUNTRY_CODE

```
typedef enum _WIFI_COUNTRY_CODE
{
        WIFI_COUNTRY_US             = 0,
        WIFI_COUNTRY_EU             ,
        WIFI_COUNTRY_JP             ,
        WIFI_COUNTRY_CN
}WIFI_COUNTRY_CODE;
```

The enumeration lists all the country codes able to set to WIFI driver.

2.5 WIFI_NETWORK

```
typedef struct _WIFI_NETWORK
{
        unsigned char *             ssid;
        WIFI_SECURITY_TYPE          security_type;
        unsigned char *             password;
        int                         ssid_len;
        int                         password_len;
}WIFI_NETWORK;
```

This structure is used to describe the station mode setting about SSID, security type and password, used when connecting to an AP. The data length of string pointed by ssid and password should not exceed 32.

2.6 WIFI_AP

```
typedef struct _WIFI_AP
{
        unsigned char *                 ssid;
        WIFI_SECURITY_TYPE              security_type;
        unsigned char *                 password;
        int                             ssid_len;
        int                             password_len;
        int                             channel;
} WIFI_AP;
```

This structure is used to describe the setting about SSID, security type, password and default channel, used to start AP mode. The data length of string pointed by ssid and password should not exceed 32.

2.7 WIFI_SETTING

```
typedef struct _WIFI_SETTING
{
        WIFI_MODE_TYPE              mode;
        unsigned char               ssid[32];
        unsigned char               channel;
        WIFI_SECURITY_TYPE          security_type;
        unsigned char               password[32];
}WIFI_SETTING;
```

This structure is used to store the WIFI setting gotten from WIFI driver.

2.8 WIFI_NETWORK_MODE

```
typedef enum _WIFI_NETWORK_MODE {
        WIFI_NETWORK_B              = 1,
        WIFI_NETWORK_BG             = 3,
        WIFI_NETWORK_BGN            = 11
} WIFI_NETWORK_MODE;
```

The enumeration lists the supported wireless network mode.

2.9 WIFI_EVENT_INDICATE

```
typedef enum _WIFI_NETWORK_MODE {
        WIFI_EVENT_CONNECT                          = 0,
        WIFI_EVENT_DISCONNECT                       = 1,
        WIFI_EVENT_FOURWAY_HANDSHAKE_DONE           = 2
} WIFI_NETWORK_MODE;
```

This enumeration is event type indicated from wlan driver

2.10 WIFI_CUSTOM_IE_TYPE

```
enum CUSTOM_IE_TYPE {
        PROBE_REQ                                   = BIT(0),
        PROBE_RSP                                   = BIT(1),
        BEACON                                      = BIT(2)
};
```

This enumeration is transmission type for wifi custom ie.

2.11 WIFI_CUSTOM_IE

```
typedef struct _cus_ie {
        __u8 *ie;
        __u8 type;
} cus_ie, *p_cus_ie;
```

This structure is used to set WIFI custom ie list, and type match WIFI_ CUSTOM_IE_TYPE. The ie will be transmitted according to the type.

ie format:

element ID	Length of Content	content in length byte

3 Application Programming Interface

3.1 Wifi enable/disable

3.1.1 wifi_on

This function uses to enable wifi .

Syntax

```
int
wifi_on(
        unsigned int mode
        )
```

Parameters

mode

Decide to enable WiFi in which mode. Such as STA mode, AP mode, STA+AP concurrent mode or Promiscuous mode.

Return Value

If the function succeeds, the return value is 0

3.1.2 wifi_off

```
int
wifi_on(
        int devnum
        )
```

Parameters

None

Return Value

If the function succeeds, the return value is 0

3.2 Station Mode Connection

3.2.1 wifi_connect

This function triggers connection to a WIFI network.

Syntax

```
int
wifi_connect(
        WIFI_NETWORK *pNetwork
        )
```

Parameters

pNetwork

Points to the WIFI_NETWORK structure including the setting used to connect to an AP.

Return Value

If the function succeeds, the return value is 0.

Remarks

If the return value is 0, it only means the operation to trigger WIFI to connect a network is done. Doing wifi_get_connect_status() will be required to check this network connection result.

3.2.2 wifi_disconnect

This function triggers disconnection from current WIFI network.

Syntax

```
int
wifi_disconnect (
        void
        )
```

Parameters

None

Return Value

If the function succeeds, the return value is 0.

Remarks

None

3.2.3 wifi_get_connect_status

This function gets the status of an existing WIFI network connection in station mode.

Syntax

```
WIFI_LINK_STATUS
wifi_get_connect_status (
        WIFI_NETWORK *pWifi
        )
```

Parameters

pWifi

Points to the WIFI_NETWORK structure used when connecting to a network by wifi_connect(). The connection based on the ssid and security type in this WIFI_NETWORK will be checked.

Return Value

If the network connected, the return value is WIFI_LINK_CONNECTED. Otherwise, the return value is WIFI_LINK_DISCONNECTED.

Remarks

None

3.3 AP Mode Startup

3.3.1 wifi_active_ap

This function triggers WIFI driver to start the AP mode.

Syntax

```
int
wifi_active_ap (
          const char *ifname
        WIFI_AP *pAP
        )
```

Parameters

Ifname

The wlan name，can use WLAN0_NAME or WLAN1_NAME.

pAP

Points to the WIFI_AP structure including the setting for AP mode startup.

Return Value

If the function succeeds, the return value is 0.

Remarks

None

3.4 Wifi Setting Information

3.4.1 wifi_get_setting

This function gets current WIFI setting from driver.

Syntax

```
int
wifi_get_setting (
        const char *ifname,
        WIFI_SETTING *pSetting
        )
```

Parameters

Ifname

The wlan name， can use WLAN0_NAME or WLAN1_NAME.

pSetting

Points to the WIFI_SETTING structure to store the WIFI setting gotten from driver.

Return Value

If the function succeeds, the return value is 0.

Remarks

None

3.4.2 wifi_show_setting

This function simply shows the information stored in a WIFI_SETTING structure.

Syntax

```
Int
```

```
wifi_show_setting (
        const char *ifname,
        WIFI_SETTING *pSetting
        )
```

Parameters

Ifname

The wlan name， can use WLAN0_NAME or WLAN1_NAME.

pSetting

Points to the WIFI_SETTING structure which information is gotten by wifi_get_setting().

Return Value

If the function succeeds, the return value is 0.

Remarks

None.

3.5 Wifi RF Control

3.5.1 wifi_rf_on

This function enables the WIFI RF.

Syntax

```
int
wifi_rf_on (
        void
        )
```

Parameters

None.

Return Value

If the function succeeds, the return value is 0.

Remarks

None.

3.5.2 wifi_rf_off

This function disables WIFI RF.

Syntax

```
int
wifi_rf_off (
        void
        )
```

Parameters

None.

Return Value

If the function succeeds, the return value is 0.

Remarks

None

3.6 Wifi RSSI Information

3.6.1 wifi_get_rssi

This function gets RSSI value from driver.

Syntax

```
int
wifi_get_rssi (
        int *pRSSI
        )
```

Parameters

pRSSI

Points to the integer to store the RSSI value gotten from driver.

Return Value

If the function succeeds, the return value is 0.

Remarks

None.

3.7 Country Code Setup

3.7.1 wifi_set_country

This function sets country code to driver.

Syntax

```
int
wifi_set_country (
        WIFI_COUNTRY_CODE country_code
        )
```

Parameters

country_code

Specifies the country code.

Return Value

If the function succeeds, the return value is 0.

Remarks

None.

3.8 Network Mode Setup

3.8.1 wifi_set_network_mode

Driver works in BGN mode in default after driver initialization. This function is used to change wireless network mode for station mode before connecting to AP

Syntax

```
int
wifi_set_network_mode(
        WIFI_NETWORK_MODE mode
        );
```

Parameters

mode

Network mode to set network to B, BG or BGN.

Return Value

If the function succeeds, the return value is 0.

3.9 Wifi Scan List

3.9.1 wifi_scan

This function is used to scan AP list.

Syntax

```
int
wifi_scan(
        char *buf,
        int buf_len
        );
```

Parameters

buf

A Pointer to the allocated buffer is used to store scanned AP's information. The information includes MAC address, RSSI and SSID.

buf_len

It indicates the length of the allocated buffer.

Return Value

If the function succeeds, the return value is the count of all scanned AP. Otherwise the return value is -1. If the return count is bigger than the count parsed from buffer, it indicated that the buffer length is not enough to store all scanned AP information.

Remarks

Return buffer format: 1st byte is the total information length(n) of first scanned AP, 2nd to 7th bytes are the MAC address, 8th to 11th bytes are the RSSI value, 12th to nth bytes are the SSID. (n+1)th byte is the length(m) of second scanned AP, (n+2)th to (n+7)th bytes are the MAC address, , (n+8)th to (n+11)th bytes are the RSSI value, (n+12)th to (n+m)th bytes are the SSID.

| 1st AP info length (1 byte) | 1st MAC address (6 bytes) | 1st RSSI (4 bytes) | 1st SSID (n-11 bytes)
| 2nd AP info length (1 byte) | 2nd MAC address (6 bytes) | 2nd RSSI (4 bytes) | 2nd SSID (m-11 bytes)
…

3.10 Wlan Driver Indication

3.10.1 wifi_indication

Wlan driver indicate event to upper layer through wifi_indication.

Syntax

```
void
wifi_indication (
        WIFI_EVENT_INDICATE event,
        char *buf,
        int buf_len
        );
```

Parameters

Event

An Event reported from driver to upper layer application.

0: WIFI_EVENT_CONNECT

For WPA/WPA2 mode, indication of connection does not mean data can be correctly transmitted or received. Data can be correctly transmitted or received only when 4-way handshake is done. Please check WIFI_EVENT_FOURWAY_HANDSHAKE_DONE event.

1: WIFI_EVENT_DISCONNECT

2: WIFI_EVENT_FOURWAY_HANDSHAKE_DONE

3: WIFI_EVENT_RECONNECTION_FAIL

This flag works with CONFIG_AUTO_RECONNECT enabled, and will be called while auto reconnection failed.

buf

If it is not NUL, buf is a Pointer to the buffer for message string.

buf_len

It indicates the length of the buffer.

Return Value

None

Remarks

If upper layer application triggers additional operations on receiving of wext_wlan_indicate, please strictly check current stack size usage (by using uxTaskGetStackHighWaterMark()) , and tries not to share the same stack with wlan driver if remaining stack space is not available for the following operations. ex: using semaphore to notice another thread instead of handing event directly in wifi_indication().

3.11 Wifi Partial Channel Scan

3.11.1 wifi_set_pscan_chan

This function sets the channel used to be partial scanned.

Syntax

```
void
wifi_set_pscan_chan (
        __u8 *channel_list
        __u8 length,
        );
```

Parameters

channel_list

An array stores the channel list.

length

Indicate the length of the channel_list.

Return Value

Return 0 if success, otherwise return -1.

Remarks

This function should be used with wifi_scan function. First, use wifi_set_pscan_chan to indicate which channel will be scanned scan, and then use wifi_scan to get scanned results.

3.12 Wifi Promiscuous Mode

3.12.1 wifi_set_promisc

This function lets Wi-Fi to start or stop Promiscuous mode.

Syntax

```
void
wifi_set_promisc (
        __unsigned char enabled,
        __void (*callback)(unsigned char*, unsigned char*, unsifned int),
        __unsigned char len_used
);
```

Parameters

enabled

Enable or disable promiscuous mode.

callback

Callback function used to process packet information captured by Wi-Fi.

len_used

If len_used set to 1, packet length will be saved and transferred to callback function.

Return Value

None.

Remarks

This function can be used to implement vendor specified simple configure.

3.13 Wifi Auto Reconnection

3.13.1 wifi_set_autoreconnect

This function sets reconnection mode.

Syntax

```
int
wifi_set_autoreconnect (
        __u8 mode,
        );
```

Parameters

mode

set 1/0 to enable/disable the reconnection mode.

Return Value

Return 0 if success, otherwise return -1.

Remarks

Defining CONFIG_AUTO_RECONNECT in “autoconfig.h” needs to be done before compiling, or this API won’t be effective.

3.13.2 wifi_get_autoreconnect

This function gets the result of setting reconnection mode

Syntax

```
int
wifi_get_autoreconnect (
        __u8 *mode
        );
```

Parameters

mode

Point to the result of setting reconnection mode.

Return Value

Return 0 if success, otherwise return -1.

Remarks

Defining CONFIG_AUTO_RECONNECT in “autoconfig.h” needs to be done before compiling, or this API won’t be effective.

3.14 Wifi Custom IE

3.14.1 wifi_add_custom_ie

This function setups custom ie list according to WIFI_ CUSTOM_IE_TYPE.

Syntax

```
int
wifi_add_custom_ie (
        void *cus_ie,
        int ie_num,
        );
```

Parameters

cus_ie

pointer to WIFI CUSTOM IE list.

ie_num

It indicate the number of WIFI CUSTOM IE list.

Return Value

Return 0 if success, otherwise return -1.

Remarks

Defining CONFIG_CUSTOM_IE in “autoconfig.h” needs to be done before compiling, or this API won’t be effective. This API can’t be executed twice before deleting the previous custom ie list.

3.14.2 wifi_update_custom_ie

This function updates the item in WIFI CUSTOM IE list.

Syntax

```
int
wifi_update_custom_ie (
        void *cus_ie,
        int ie_index
        );
```

Parameters

cus_ie

pointer to WIFI CUSTOM IE address.

ie_index

index of WIFI CUSTOM IE list.

Return Value

Return 0 if success, otherwise return -1.

Remarks

Defining CONFIG_CUSTOM_IE in “autoconfig.h” needs to be done before compiling, or this API won’t be effective.

3.14.3 wifi_del_custom_ie

This function sets connection mode to reconnection mode.

Syntax

```
int
wifi_del_custom_ie ();
```

Parameters

None

Return Value

Return 0 if success, otherwise return -1.

Remarks

Defining CONFIG_CUSTOM_IE in “autoconfig.h” needs to be done before compiling, or this API won’t be effective.