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.

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.

Usage	API & Keyword
How does station connect to AP?	1. Use ATCMD ATW0=<ssid> ATW1=<password> ATW2=<key_id> ATWC 2. Call API in wifi_conf.c wifi_connect : use SSID to connect to AP wifi_connect_bssid : use bssid to connect to AP
How does station disconnect from AP?	1. Use ATCMD ATWD 2. Call API in wifi_conf.c wifi_disconnect
How to register wifi event callback function?	Search “wifi_reg_event_handler” as reference in wifi_conf.c
How to detect wlan condition of connect or disconnect event?	Register wifi event callback function for specific event. (Total event can reference 3.24)

	WIFI_EVENT_CONNECT : association done WIFI_EVENT_FOURWAY_HANDSHAKE_DONE : fourway handshake done WIFI_EVENT_BEACON_AFTER_DHCP : Get IP from DHCP WIFI_EVENT_DISCONNECT : wifi disconnect
How to enable/disable power saving mode in station mode?	Call API in wifi_conf.c wifi_enable_powersave wifi_disable_powersave
How to start soft AP mode?	1. Use ATCMD ATW3=<ssid> ATW4=<>password ATW5=<channel> ATWA 2. Call API in wifi_conf.c wifi_start_ap
How to start soft AP mode with hidden ssid?	Call API in wifi_conf.c wifi_start_ap_with_hidden_ssid
How to create concurrent mode?	Use ATCMD, start AP first then Station ATW3=<ssid> ATW4=<>password ATW5=<channel> ATWB ATW0=<ssid>

	ATW1=<password> ATW2=<key_id> ATWC
How to set client number in AP mode?	Call API in wifi_util.c wext_set_sta_num
How to delete station in AP mode?	Call API in wifi_util.c wext_del_station
How to get auto-scan channel ?	Call API in wifi_util.c wext_get_auto_chl
How to set partial scan channel in station mode?	Call API in wifi_conf.c wifi_set_pscan_chan
How to set auto-reconnect in station mode?	Call API in wifi_conf.c wifi_config_autoreconnect
How to get TX power?	Call API in wifi_util.c wext_get_tx_power
How to get RX RSSI?	Call API in wifi_conf.c wifi_get_rssi

3.1

The enumeration lists the return result of the function.

3.2

The enumeration lists the band type.

3.3

The enumeration lists the scan type. RTW_SCAN_TYPE_ACTIVE means actively scan a network by sending 802.11 probe(s). RTW_SCAN_TYPE_PASSIVE means passively scan a network by listening for beacons from APs. RTW_SCAN_TYPE_PROHIBITED_CHANNELS means Passively scan on channels not enabled by the country code.

3.4

The enumeration lists the bss types. RTW_BSS_TYPE_UNKNOWN denotes infrastructure network. RTW_BSS_TYPE_ADHOC denotes an 802.11 ad-hoc IBSS network. RTW_BSS_TYPE_ANY denotes either infrastructure or ad-hoc network. RTW_BSS_TYPE_UNKNOWN may be returned by scan function if BSS type is unknown.

3.5

The enumeration lists the wps type.

3.6

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

3.7

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

3.8

The enumeration lists the status to describe the connection link.

3.9

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

3.10

The enumeration lists all the network bgn mode .

3.11

The enumeration lists the interface. RTW_STA_INTERFACE means STA or client interface, RTW_AP_INTERFACE means softAP interface .

3.12

The enumeration lists the packet filter rule. RTW_POSITIVE_MATCHING means receiving the datas matching with this pattern and discard the other data. RTW_NEGATIVE_MATCHING means discard the data matching with this pattern and receive the other data.

3.13

The enumeration lists the promisc level. RTW_PROMISC_DISABLE means disable the promisc, RTW_PROMISC_ENABLE means enable the promisc. RTW_PROMISC_ENABLE_1 is used to enable the promisc special when the length is used.

3.14

This struct is used to describe the SSID.

3.15

This struct is used to describe the unique 6-byte MAC address.

3.16

This struct is used to describe the scan result of the AP.

3.17

This structure is used to describe the data needed by scan result handler function.

3.18

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.

3.19

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.

3.20

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

3.21

The struct is used to describe the setting when config the network.

3.22

The struct is used to describe the maclist. Count means number of MAC addresses in the list. Mac_list means variable length array of MAC address.

3.23

The struct is used to describe the bss info of the network. It include the version, BSSID, beacon_period, capability, SSID, channel, atm_window, dtim_period, RSSI e.g.

3.24

This enumeration is event type indicated from wlan driver.

3.25

This enumeration is transmission type for wifi custom ie.

3.26

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

ie format:

element ID	Length of Content	content in length byte

3.27

This structure is used to set WIFI packet filter pattern. Offset in bytes is used to specify where to start filtering. Mask_size is the size of the mask in bytes. Mask means filter mask. Pattern is the bytes used to filter.

3.28

This enumeration is adaptivity type. RTW_ADAPTIVITY_NORMAL is for CE and RTW_ADAPTIVITY_CARRIER_SENSE is for MKK.

4.1

This function uses to enable wifi .

Syntax

mode

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

If the function succeeds, the return value is 0

None

If the function succeeds, the return value is 0

This function checked if the interface specified is up.

interface

The interface can be set RTW_AP_INTERFACE or RTW_MODE_STA_AP.

If the function succeeds, the return value is 0

This function checked if the interface specified is ready to transceiver Ethernet packets.

interface

The interface can be set RTW_AP_INTERFACE or RTW_MODE_STA_AP.

If the function succeeds, the return value is 0

4.2

This function triggers connection to a WIFI network.

Syntax

ssid

A null terminated string containing the SSID name of the network to join.

Security_type

The security type of the AP to connect.

password

A byte array containing the security_key.

ssid_len

The length of the SSID in bytes.

password_len

The length of the security_key in bytes.

key_id

The index of the wep key.

semaphore

A user provided semaphore that is flagged when the join is complete.

If the function succeeds, the return value is RTW_SUCCESS.

None.

This function triggers disconnection from current WIFI network.

None

If the function succeeds, the return value is RTW_SUCCESS.

None

4.3

This function triggers WIFI driver to start the AP mode.

ssid

A null terminated string containing the SSID name of the AP.

security_type

The security type of the AP to start.

password

A byte array containing the security key for the AP.

ssid_len

The length of the SSID in bytes.

password_len

The length of the security_key in bytes.

channel

802.11 channel number.

If the function succeeds, the return value is RTW_SUCCESS.

None

This function triggers WIFI driver to restart an infrastructure WiFi network.

ssid

A null terminated string containing the SSID name of the network to join.

security_type

Authentication type.

password

A byte array containing the security key for the network.

ssid_len

The length of the SSID in bytes.

password_len

The length of the security_key in bytes.

channel

802.11 channel number.

If the function succeeds, the return value is RTW_SUCCESS.

None

This function gets the SoftAP information.

```

```

ap_info

The location where the AP info will be stored.

security

The security type.

If the result was successfully get return RTW_SUCCESS, else return RTW_ERROR.

None

This function gets the associated clients with SoftAP.

client_list_buffer

The location where the client list will be stored.

buffer_length

The buffer length.

If the result was successfully get return RTW_SUCCESS, else return RTW_ERROR.

None

4.4

This function gets current WIFI setting from driver.

Ifname

The wlan name， can use WLAN0_NAME or WLAN1_NAME.

pSetting

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

If the function succeeds, the return value is RTW_SUCCESS.

None

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

Ifname

The wlan name， can use WLAN0_NAME or WLAN1_NAME.

pSetting

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

If the function succeeds, the return value is RTW_SUCCESS.

None.

4.5

This function enables the WIFI RF.

None.

If the function succeeds, the return value is 0.

None.

This function disables WIFI RF.

None.

If the function succeeds, the return value is 0.

None

4.6

This function gets RSSI value from driver.

pRSSI

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

If the function succeeds, the return value is 0.

None.

4.7

This function sets country code to driver.

country_code

Specifies the country code.

If the function succeeds, the return value is 0.

None.

4.8

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

mode

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

If the function succeeds, the return value is 0.

4.9

This function is used to scan AP list.

results_handler

The callback function which will receive and process the result data.

user_data

user specific data that will be passed directly to the callback function.

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.

Callback must not use blocking functions, since it is called from the context of the RTW thread. The callback, user_data variables will be referenced after the function returns. Those variable must remain valid until the scan is complete.

4.10

Wlan driver indicate event to upper layer through wifi_indication.

```
rtw_event_indicate_t
```

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.

4: WIFI_EVENT_SCAN_DONE

5: WIFI_EVENT_RECONNECTION_FAIL

6: WIFI_EVENT_SEND_ACTION_DONE

7: WIFI_EVENT_RX_MGNT

8: WIFI_EVENT_STA_ASSOC

9: WIFI_EVENT_STA_DISASSOC

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.

flag

It indicates some extra information, some times it is zero.

None

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().

4.11

This function sets the channel used to be partial scanned.

channel_list

An array stores the channel list.

length

Indicate the length of the channel_list.

Return 0 if success, otherwise return -1.

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.

4.12

This function is used to init packet filter related data.

None.

None.

This function is used to add packet filter, and now the maximum number of filter is 5 .

filter_id

The filter id.

patt

Point to the filter pattern.

rule

Point to the filter rule.

Return 0 if success, otherwise return -1.

This function is used to enable packet filter. The filter can be used only if it has been enabled.

filter_id

The filter id.

Return 0 if success, otherwise return -1.

This function is used to disable the packet filter.

filter_id

The filter id.

Return 0 if success, otherwise return -1.

This function is used to remove the packet filter.

filter_id

The filter id.

Return 0 if success, otherwise return -1.

4.13

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

enabled

Enable or disable promiscuous mode. 0 means disable the promiscuous mode, 1 means enable the promiscuous, 2 is used special for the len_used.

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.

customer

4.14

This function sets reconnection mode.

mode

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

Return 0 if success, otherwise return -1.

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

This function gets the result of setting reconnection mode

mode

Point to the result of setting reconnection mode.

Return 0 if success, otherwise return -1.

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

4.15

pointer to WIFI CUSTOM IE address.

index of WIFI CUSTOM IE list.

Return 0 if success, otherwise return -1.

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

This function sets connection mode to reconnection mode.

None

Return 0 if success, otherwise return -1.

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

4.16

This function sets mac address of the 802.11 device.

mac

Pointer to a variable that the current MAC address will be written to.

Return RTW_SUCCESS if success, otherwise return RTW_ERROR.

NONE.

This function gets the mac address of the 802.11 device.

mac

Point to the result of the mac address will be get.

Return RTW_SUCCESS if success, otherwise return RTW_ERROR.

NONE.

4.17

This function enable wifi powersave mode.

void

Return RTW_SUCCESS if success, otherwise return RTW_ERROR.

NONE.

This function disable wifi powersave mode.

void

Return RTW_SUCCESS if success, otherwise return RTW_ERROR.

NONE.

4.18

This function set the tx power in index units.

poweridx

The desired tx power in index.

Return RTW_SUCCESS if success, otherwise return RTW_ERROR.

NONE.

This function gets the tx power in index units.

poweridx

The variable to receive the tx power in index.

Return RTW_SUCCESS if success, otherwise return RTW_ERROR.

NONE.

4.19

This function set the current channel on STA interface.

channel

Set the current channel on STA interface.

Return RTW_SUCCESS if success, otherwise return RTW_ERROR.

NONE.

This function gets the current channel on STA interface.

poweridx

A pointer to the variable where the channel value will be written..

Return RTW_SUCCESS if success, otherwise return RTW_ERROR.

NONE.

4.20

This function registers interest in a multicast address.

mac

Ethernet MAC address.

Return RTW_SUCCESS if success, otherwise return RTW_ERROR.

NONE.

This function unregisters interest in a multicast address.

mac

Ethernet MAC address.

Return RTW_SUCCESS if success, otherwise return RTW_ERROR.

NONE.

4.21

This function sets wps phase.

```
wifi_set_wps_phase
```

is_trigger_wps

set 1/0 to enable/disable the wps phase.

Return 0 if is_trigger_wps is 1 or 0, otherwise return -1.

NONE.

4.22

This function can call wext_set_adaptivity to setup adaptivity.

```
wext_set_adaptivity
```

Adaptivity mode

set adpativity mode. RTW_ADAPTIVITY_DISABLE is disable, RTW_ADAPTIVITY_NORMAL is for CE and RTW_ADAPTIVITY_CARRIER_SENSE is for MKK.

Return 0 if enable is 1 or 0, otherwise return -1.

NONE.