# WLAN API Reference Guide

Amp’ed RF Technology, Inc.

# 1. Overview

The SDK provides programmers with the WLAN interfaces for connection, transmission, driver and configuration. The Wi-Fi stack is complete below the API layer.

The SDK is based on FreeRTOS. Programmers can use standard interfaces to realize resource management, recycling operations, execution delays, inter-task messaging and synchronization, and other task-oriented process design approaches. IAR7.2 is required to compile the SDK. The firmware download tool is term developed by Amp'ed RF. Programmers can load their customized binary to the MCU chip in the module. The memory available is 45KB flash.

# 2. Module APIs

## 2.1. Connect APIs

The following list contains all connect APIs

- wlan_sta_scan()
- wlan_sta_join()
- wlan_sta_unjoin()
- wlan_sta_connect()
- wlan_sta_disconnect()
- wlan_sta_status()

### 2.1.1. wlan_sta_scan()

Scan all available APs. This function should be called in station mode

Prototype: u8_t wlan_scan (scan_callback_t *scan_done_cb)

| Returns | Parameters |
|---|---|
| enum result_type | typedef void (*scan_callback_t)(const scan_data_t *pScandata) |

### 2.1.2. wlan_sta_join()

Join the ACC1340 station to the AP.

Prototype: u8_t wlan_sta_join(wlan_ap_t *pAP)

| Returns | Parameters |
|---|---|
| enum result_type | typedef struct wlan_ap_s |

### 2.1.3. wlan_sta_unjoin()

Exit the AP.

Prototype: void wlan_sta_unjoin(void)

### 2.1.4. wlan_sta_connect()

Connect the ACC1340 station to the remote station with connected the same AP.

Prototype: u8_t wlan_sta_connect(u8_t protocol, wlan_station_t *pSta)

| Returns | Parameters |
|---|---|
| enum result_type | u8_t protocol -- WLANCONN_UDP, WLANCONN_TCP_CLIENT, WLANCONN_TCP_SERVER<br>typedef struct wlan_station_s |

### 2.1.5. wlan_sta_disconnect()

Disconnect the local station from the remote

Prototype: void wlan_sta_disconnect(void)

### 2.1.6. wlan_sta_status ()

get the station connection status

Prototype: bool wlan_sta_status(void)

| Returns | Parameters |
|---|---|
| False-- not connected<br>True-- connected | |

## 2.2. Transmit APIs

The following list contains all transmit APIs.

- wlan_sta_send()

### 2.2.1 wlan_sta_send()

Send packet to the currently connected remote station.

Prototype: u8_t wlan_sta_send(u8_t *pBuf, u16_t len)

| Returns | Parameters |
|---|---|
| enum result_type | u8_t *pBuf<br>u16_t len |

## 2.3. Configuration APIs

The following list contains all configuration APIs.

- wlan_get_device_name()
- wlan_set_device_name()

- wlan_get_mac_address()
- wlan_set_mac_address()
- wlan_get_dhcp_mode()
- wlan_set_dhcp_mode()
- wlan_get_ip_info()
- wlan_set_ip_info()
- wlan_get_sleep_mode()
- wlan_set_sleep_mode()
- wlan_get_operation_mode()
- wlan_set_operation_mode()

### 2.3.1. wlan_get_device_name()

Get the module device name

Prototype: char* wlan_get_device_name(void)

| Returns | Parameters |
|---|---|
| device name string | |

### 2.3.2. wlan_set_device_name()

Set the module device name

Prototype: u8_t wlan_set_device_name(char* pName)

| Returns | Parameters |
|---|---|
| enum result_type | device name string pointer |

### 2.3.3. wlan_get_mac_address()

Get mac address

Prototype: void wlan_get_mac_address(u8_t* pMac)

| Returns | Parameters |
|---|---|
| | mac string pointer |

### 2.3.4. wlan_set_mac_address()

Set mac address

Prototype: void wlan_set_mac_address(u8_t* pMac)

| Returns | Parameters |
|---|---|
| | mac string pointer |

### 2.3.5. wlan_get_dhcp_mode()

Get DHCP mode

Prototype: bool wlan_get_dhcp_mode(void)

| Returns | Parameters |
|---|---|
| true or false | |

### 2.3.6. wlan_set_dhcp_mode()

Set DHCP mode true or false

Prototype: u8_t wlan_set_dhcp_mode(bool mode)

| Returns | Parameters |
|---|---|
| enum result_type | ture or false |

### 2.3.7. wlan_get_ip_info()

Get the IP address of the station or AP

Prototype: void wlan_get_ip_info(ip_info_int_t* pIP_int)

| Returns | Parameters |
|---|---|
| | typedef struct ip_info_int_s |

### 2.3.8. wlan_set_ ip_info()

Set the ip address of the station or AP

Prototype: u8_t wlan_set_ip_info(ip_info_str_t ip_str)

| Returns | Parameters |
|---|---|
| | typedef struct ip_info_int_s |

### 2.3.9. wlan_get_sleep_mode()

Get the module sleep mode

Prototype: sleep_mode_t wlan_get_sleep_mode(void)

| Returns | Parameters |
|---|---|
| typedef enum<br>sleep_mode_e | |

### 2.3.10. wlan_set_sleep_mode ()

Set the module sleep mode

Prototype: u8_t wlan_set_sleep_mode(sleep_mode_t mode)

| Returns | Parameters |
|---|---|
| enum result_type | typedef enum sleep_mode_e |

### 2.3.11. wlan_get_operation_mode()

Get the current operating mode of the WiFi

Prototype: operate_mode_t wlan_get_operation_mode(void)

| Returns | Parameters |
|---|---|
| typedef enum operate_mode_e | |

### 2.3.12. wlan_set_operation_mode ()

Set the WiFi operating mode

Prototype: u8_t wlan_set_operation_mode(operate_mode_t mode)

| Returns | Parameters |
|---|---|
| enum result_type | typedef enum operate_mode_e |

## 2.4. System APIs

The following list contains partial system APIs. The operation system is FreeRTOS, so it support all FreeRTOS APIs.

- wlan_restart()
- wlan_printf()

### 2.4.1. wlan_restart()

Restart system

Prototype: void wlan_restart(void)

### 2.4.2. wlan_printf()

Print string. It is invalid in uart connect mode.

Prototype: void wlan_printf(char *fmt, ...)

| Returns | Parameters |
|---|---|
| | string |

## 2.5. Driver APIs

The following list contains all interface driver APIs.

- wlan_set_uart_connect_mode()

- wlan_uart_send()
- wlan_set_uart_baudrate()
- wlan_spi_init()
- wlan_spi_send_bytes()
- wlan_spi_sndrcv_bytes()
- wlan_spi_rcv_bytes()
- wlan_gpio_config()
- wlan_gpio_set()
- wlan_gpio_get()

### 2.5.1. wlan_set_uart_connect_mode()

The uart is used to connect to other chip or module.

Prototype: void wlan_set_uart_connect_mode(uart_callback *rx_cb)

| Returns | Parameters |
|---|---|
| | uart receive callback function |

### 2.5.2. wlan_uart_send()

Send data in connect mode.

Prototype: void wlan_uart_send(u8_t *pData, u16_t length)

| Returns | Parameters |
|---|---|
| | pointer of data and length |

### 2.5.3. wlan_set_uart_baudrate()

Set uart baudrate. The default baudrate is 115200bps.

Prototype: void wlan_set_uart_baudrate(u32_t baudrate)

| Returns | Parameters |
|---|---|
| | baudrate |

### 2.5.4. wlan_spi_init()

Initialization SPI. The default mode is master

Prototype: void wlan_spi_init(tSPIDevice* spi_device)

| Returns | Parameters |
|---|---|
| | Use tSPIDevice struct to set the spi info |

### 2.5.5. wlan_spi_send_bytes()

Send byte to the slave.

Prototype: u8_t wlan_spi_send_bytes(u8_t * sndbuf, unsigned short length)

| Returns | Parameters |
|---|---|
| | sndbuf: the buffer that want to send<br>length: the buffer length |

### 2.5.6. wlan_spi_sndrcv_bytes()

spi send and receive the bytes at the same time.

Prototype: u16_t wlan_spi_sndrcv_bytes(u8_t * sndbuf, u8_t rcvbuf, u16_t length)

| Returns | Parameters |
|---|---|
| | sndbuf: the buffer that want to send data<br>rcvbuf: the buffer that want to receive data<br>length: the buffer length |

### 2.5.7. wlan_spi_rcv_bytes()

slave mode, receive the bytes

Prototype: u16_t wlan_spi_rcv_bytes(u8_t * buf, u16_t length)

| Returns | Parameters |
|---|---|
| | rcvbuf: the buffer that want to receive data<br>length: the buffer length |

### 2.5.8. wlan_gpio_config()

Set GPIO input or output.

Prototype: void wlan_gpio_config(wlan_gpio_t gpio, gpio_direction_t dir)

| Returns | Parameters |
|---|---|
| | typedef enum gpio_e<br>typedef enum gpio_direction_e |

### 2.5.9. wlan_gpio_set()

Set GPIO value

Prototype: void wlan_gpio_set(wlan_gpio_t gpio, u8_t value)

| Returns | Parameters |
|---|---|
| | typedef enum gpio_e<br>0 or 1 |

#### 2.5.10. wlan_gpio_get()

Get GPIO value.

Prototype: u8_t wlan_gpio_get(wlan_gpio_t gpio)

| Returns | Parameters |
|---|---|
| 0 or 1 | typedef enum gpio_e |

# 3. Demo

## 3.1. Connect to AP

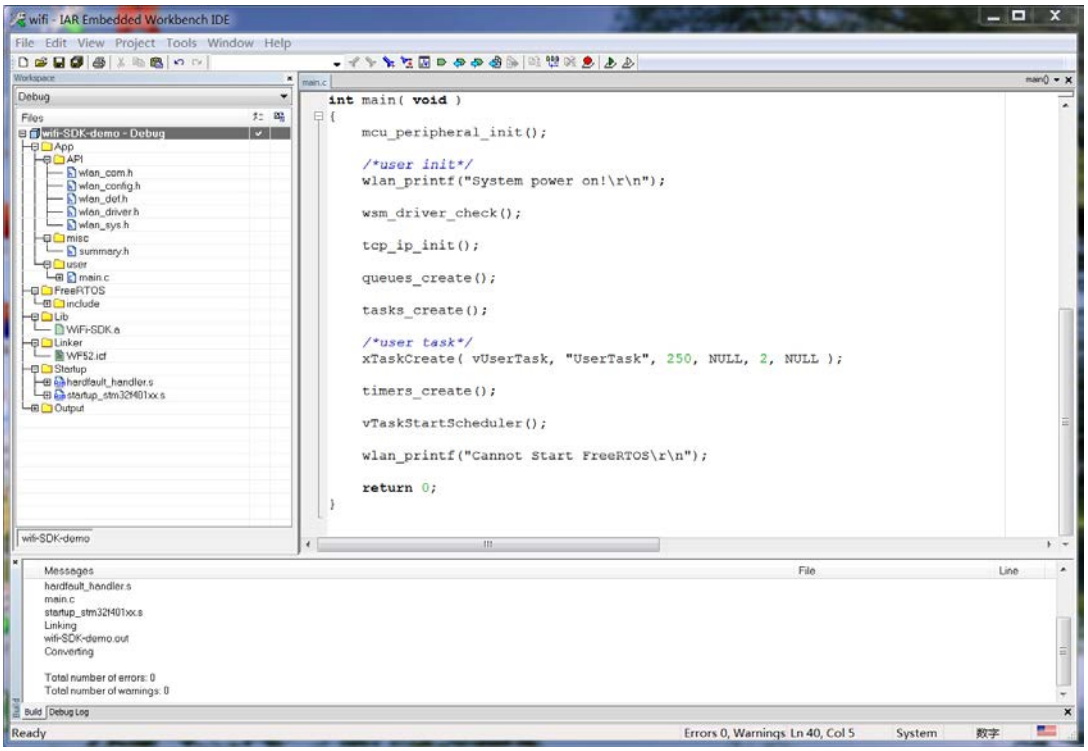
### 3.1.1. The ACC1340 default mode is station. wlan_ap_t is used to set the AP information. Create the vUserTask() after tasks_create() in main().

```
void vUserTask(void *arg)
{
    wlan_ap_t ap;

    //AP configurations
    strcpy((char*)ap.ssid, "liuting");
    strcpy((char*)ap.password, "12345678");

    for(;;)
    {
        vTaskDelay(1000);
        if(NO_ERR == wlan_sta_join(&ap))
        {
            vTaskDelete(NULL);
        }
    }
}
```

3.1.2. Build the project.




3.1.3. Download the binary to the mcu. Result: