XRADIO WLAN Developer Guide

Revision 1.0

Nov 21, 2019

Declaration

THIS DOCUMENTATION IS THE ORIGINAL WORK AND COPYRIGHTED PROPERTY OF XRADIO TECHNOLOGY ("XRADIO"). REPRODUCTION IN WHOLE OR IN PART MUST OBTAIN THE WRITTEN APPROVAL OF XRADIO AND GIVE CLEAR ACKNOWLEDGEMENT TO THE COPYRIGHT OWNER.

THE PURCHASED PRODUCTS, SERVICES AND FEATURES ARE STIPULATED BY THE CONTRACT MADE BETWEEN XRADIO AND THE CUSTOMER. PLEASE READ THE TERMS AND CONDITIONS OF THE CONTRACT AND RELEVANT INSTRUCTIONS CAREFULLY BEFORE USING, AND FOLLOW THE INSTRUCTIONS IN THIS DOCUMENTATION STRICTLY. XRADIO ASSUMES NO RESPONSIBILITY FOR THE CONSEQUENCES OF IMPROPER USE (INCLUDING BUT NOT LIMITED TO OVERVOLTAGE, OVERCLOCK, OR EXCESSIVE TEMPERATURE).

THE INFORMATION FURNISHED BY XRADIO IS PROVIDED JUST AS A REFERENCE OR TYPICAL APPLICATIONS, ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT DO NOT CONSTITUTE A WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. XRADIO RESERVES THE RIGHT TO MAKE CHANGES IN CIRCUIT DESIGN AND/OR SPECIFICATIONS AT ANY TIME WITHOUT NOTICE.

NOR FOR ANY INFRINGEMENTS OF PATENTS OR OTHER RIGHTS OF THE THIRD PARTIES WHICH MAY RESULT FROM ITS USE. NO LICENSE IS GRANTED BY IMPLICATION OR OTHERWISE UNDER ANY PATENT OR PATENT RIGHTS OF XRADIO.THIRD PARTY LICENCES MAY BE REQUIRED TO IMPLEMENT THE SOLUTION/PRODUCT. CUSTOMERS SHALL BE SOLELY RESPONSIBLE TO OBTAIN ALL APPROPRIATELY REQUIRED THIRD PARTY LICENCES. XRADIO SHALL NOT BE LIABLE FOR ANY LICENCE FEE OR ROYALTY DUE IN RESPECT OF ANY REQUIRED THIRD PARTY LICENCE. XRADIO SHALL HAVE NO WARRANTY, INDEMNITY OR OTHER OBLIGATIONS WITH RESPECT TO MATTERS COVERED UNDER ANY REQUIRED THIRD PARTY LICENCE.

Revision History

Version	Date	Summary of Changes
1.0	2019-11-21	Initial Version

Table 1- 1 Revision History

Contents

Figures

1 概述

XRADIO 系列 IC 是基于 WLAN 技术的无线 MCU，其内置 WLAN 协议栈，并于 SDK 中集成 LwIP，因此可以应用于物联网中的各类 WLAN 联网设备。此文档用以说明 XRADIO 的 SDK 中的 WLAN 的功能描述和使用方法，进而指导网络应用开发者能够有效的使用正确的 API 完成应用功能实现。

此文档将指引你实现以下功能：

- 初始化 WLAN 系统进入 STA 模式或者 AP 模式
- 配置 STA 或 AP 模式下的 WLAN 系统参数
- STA 模式下扫描，连接可用的 AP 节点
- AP 模式下进行扫描操作
- Monitor 模式的使用

1.1 网络系统架构

XRADIO 软件系统架构分为四层结构：应用层，服务层，OSAL 层及驱动层，如下图所示。

- 应用层：提供应用程序实现，网络配置管理，产品集成等。
- 服务层：提供系统接口，功耗管理，网络协议实现等，WLAN 系统及网络相关内容都属于服务层。
- OSAL 层：提供操作系统抽象，对接各操作系统内核。
- 驱动层：提供芯片级功能访问 API，提供外设数据传输通道，实现完整的设备管理。

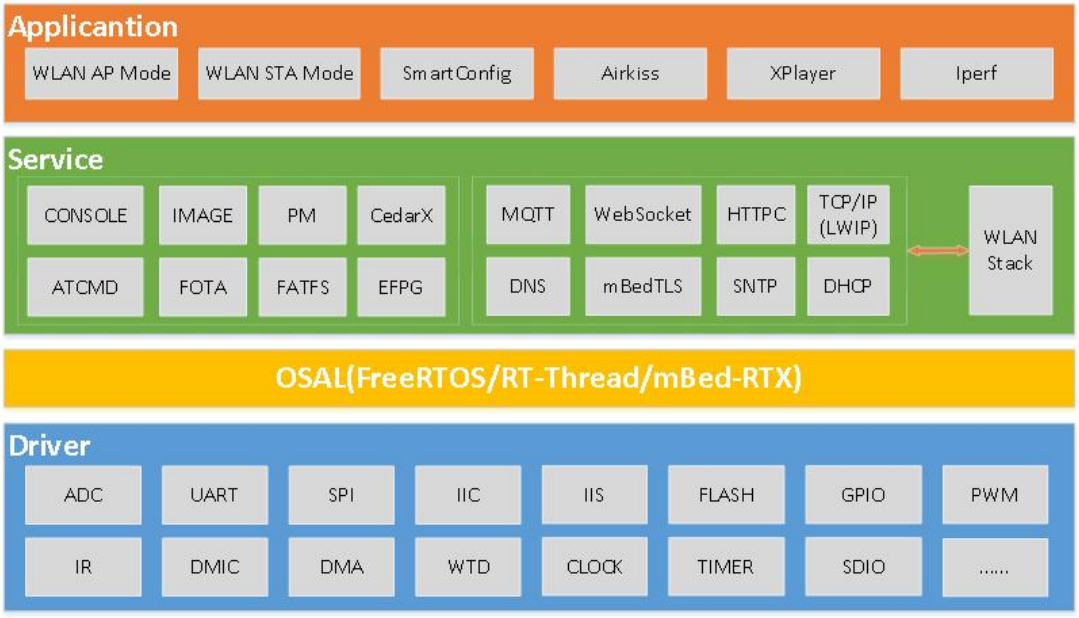

图 1-1 软件架构框图

2 WLAN 功能描述

2.1 示例工程

可以参考以下工程代码：

sdk-code/project/demo/wlan_demo

sdk-code/project/example/wlan

2.2 WLAN 系统启动

在 wlan_demo 工程中，网络系统加载流程如下：

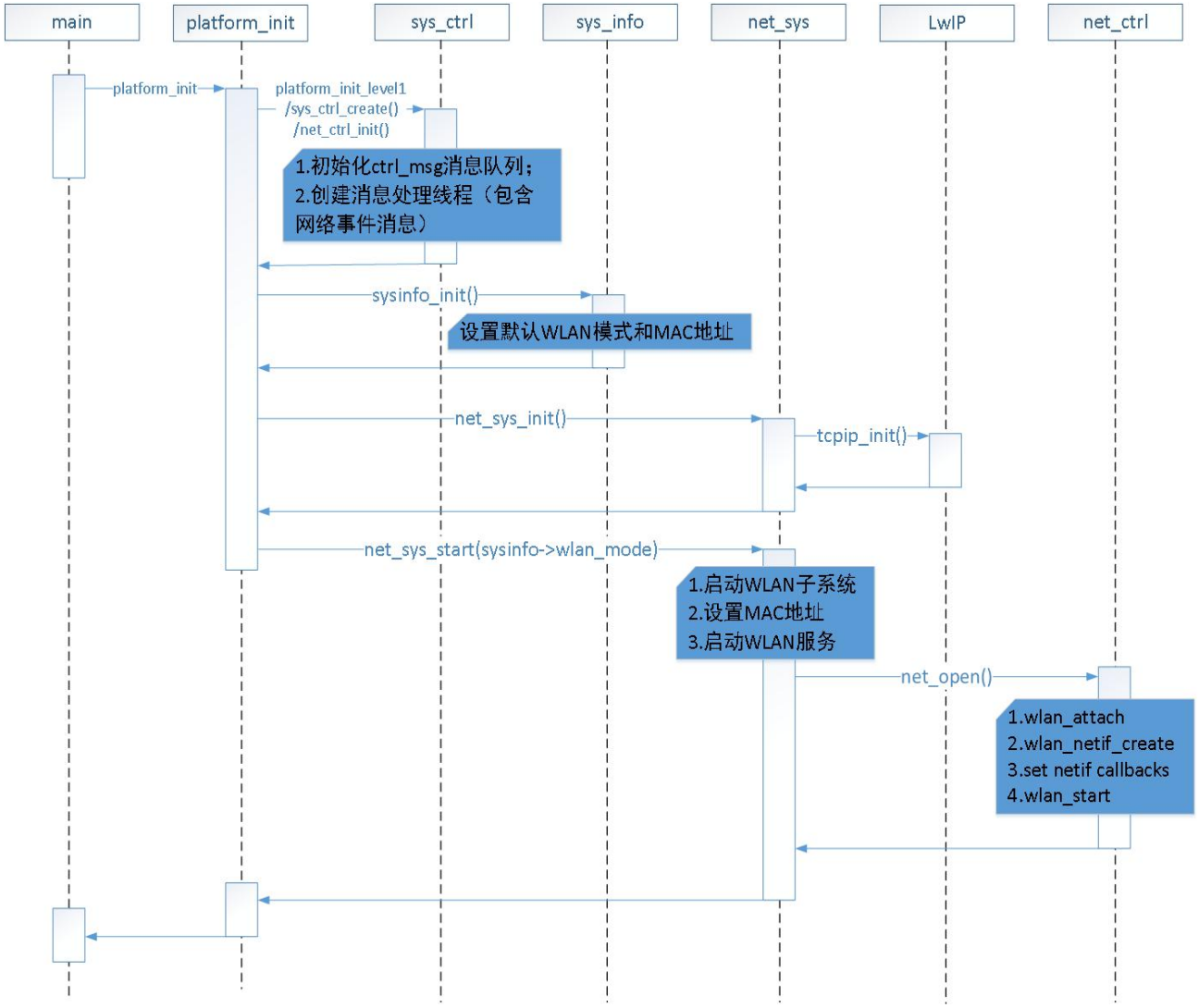

图 2-1 网络系统启动流程

网络系统加载过程中依次完成了以下几个重要的过程：

1. 初始化系统消息队列，创建系统消息处理线程；
2. 初始化 TCP/IP 协议栈；
3. 配置网络子系统硬件；
4. 创建 netif 网络接口并启动 WLAN 子系统任务；

2.3 Station 模式操作

2.3.1 初始化 STA 模式

上一章节已在网络启动流程说明通过调用 net_sys_start(mode)的方式来启动网络子系统，并初始化到相关的模式，下面通过示例说明配置 Station 模式下的网络参数。

1. 连接一个工作在开放模式的 AP，其 SSID 为 TEST_AP

   - 方式一，通过傻瓜匹配

```
uint8_t ssid[] = "TEST_AP";
uint8_t ssid_len = strlen((char *)ssid);

wlan_sta_set(ssid, ssid_len, NULL);
wlan_sta_enable();
```

   - 方式二，通过精确配置

```
uint8_t ssid[] = "TEST_AP";
uint8_t ssid_len = strlen((char *)ssid);

wlan_sta_config_t config;
memset(&config, 0, sizeof(config));

/* ssid */
config.field = WLAN_STA_FIELD_SSID;
memcpy(config.u.ssid.ssid, ssid, ssid_len);
config.u.ssid.ssid_len = ssid_len;
if (wlan_sta_set_config(&config) != 0)
    return -1;

/* auth_alg: OPEN */
config.field = WLAN_STA_FIELD_AUTH_ALG;
config.u.auth_alg = WPA_AUTH_ALG_OPEN;
if (wlan_sta_set_config(&config) != 0)
    return -1;

/* key_mgmt: NONE */
config.field = WLAN_STA_FIELD_KEY_MGMT;
config.u.key_mgmt = WPA_KEY_MGMT_NONE;
if (wpa_ctrl_request(WPA_CTRL_CMD_STA_SET, &config) != 0)
    return -1;
```

```
    if (wlan_sta_enable()!= 0)
        return -1;
```

2. 连接工作在 WEP 模式的 AP，SSID 为 TEST_AP，WEP_KEY0 为 1234567890

```
    uint8_t ssid[] = "TEST_AP";
    uint8_t ssid_len = strlen((char *)ssid);
    uint8_t psk[] = "1234567890";

    wlan_sta_config_t config;
    memset(&config, 0, sizeof(config));

    /* ssid */
    config.field = WLAN_STA_FIELD_SSID;
    memcpy(config.u.ssid.ssid, ssid, ssid_len);
    config.u.ssid.ssid_len = ssid_len;
    if (wlan_sta_set_config(&config) != 0)
        return -1;

    /* WEP key0 */
    config.field = WLAN_STA_FIELD_WEP_KEY0;
    strlcpy((char *)config.u.wep_key, psk, sizeof(config.u.wep_key));
    if (wlan_sta_set_config(&config) != 0)
        return -1;

    /* WEP key index */
    config.field = WLAN_STA_FIELD_WEP_KEY_INDEX;
    config.u.wep_tx_keyidx = 0;
    if (wlan_sta_set_config(&config) != 0)
        return -1;

    /* auth_alg: SHARED */
    config.field = WLAN_STA_FIELD_AUTH_ALG;
    config.u.auth_alg = WPA_AUTH_ALG_SHARED;
    if (wlan_sta_set_config(&config) != 0)
        return -1;

    /* key_mgmt: NONE */
    config.field = WLAN_STA_FIELD_KEY_MGMT;
    config.u.key_mgmt = WPA_KEY_MGMT_NONE;
    if (wlan_sta_set_config(&config) != 0)
        return -1;

    if (wlan_sta_enable()!= 0)
        return -1;
```

3. 连接工作在 WPA | WPA2 模式的 AP，SSID 为 TEST_AP，PSK 为 12345678

- 方式一，通过傻瓜匹配

```
    uint8_t ssid[] = "TEST_AP";
    uint8_t ssid_len = strlen((char *)ssid);
    uint8_t psk[] = "12345678";

    wlan_sta_set(ssid, ssid_len, psk);
    wlan_sta_enable();
```

- 方式二，通过精确配置

```
    uint8_t ssid[] = "TEST_AP";
```

```
    uint8_t ssid_len = strlen((char *)ssid);
    uint8_t psk[] = "12345678";

    wlan_sta_config_t config;
    memset(&config, 0, sizeof(config));

    /* ssid */
    config.field = WLAN_STA_FIELD_SSID;
    memcpy(config.u.ssid.ssid, ssid, ssid_len);
    config.u.ssid.ssid_len = ssid_len;
    if (wlan_sta_set_config(&config) != 0)
        return -1;

    /* psk */
    config.field = WLAN_STA_FIELD_PSK;
    strlcpy((char *)config.u.psk, (char *)psk, sizeof(config.u.psk));
    if (wlan_sta_set_config(&config) != 0)
        return -1;

    /* auth_alg: OPEN */
    config.field = WLAN_STA_FIELD_AUTH_ALG;
    config.u.auth_alg = WPA_AUTH_ALG_OPEN;
    if (wlan_sta_set_config(&config) != 0)
        return -1;

    /* proto: WPA | RSN */
    config.field = WLAN_STA_FIELD_PROTO;
    config.u.proto = WPA_PROTO_WPA | WPA_PROTO_RSN;
    if (wlan_sta_set_config(&config) != 0)
        return -1;

    /* key_mgmt: PSK */
    config.field = WLAN_STA_FIELD_KEY_MGMT;
    config.u.key_mgmt = WPA_KEY_MGMT_PSK;
    if (wlan_sta_set_config(&config) != 0)
        return -1;

    /* pairwise: CCMP | TKIP */
    config.field = WLAN_STA_FIELD_PAIRWISE_CIPHER;
    config.u.pairwise_cipher = WPA_CIPHER_CCMP | WPA_CIPHER_TKIP;
    if (wlan_sta_set_config(&config) != 0)
        return -1;

    /* group: CCMP | TKIP | WEP40 | WEP104 */
    config.field = WLAN_STA_FIELD_GROUP_CIPHER;
    config.u.pairwise_cipher = WPA_CIPHER_CCMP | WPA_CIPHER_TKIP
                | WPA_CIPHER_WEP40 | WPA_CIPHER_WEP104;
    if (wlan_sta_set_config(&config) != 0)
        return -1;

    if (wlan_sta_enable()!= 0)
        return -1;
```

4. 连接隐藏 SSID 的 AP，其工作在 WPA | WPA2 模式，SSID 为 TEST_AP，PSK 为 12345678

当 AP 的 SSID 隐藏时，在配置完相关参数需要明确的指明扫描此 SSID，如下面代码所示：

- 方式一，通过傻瓜匹配

```
uint8_t ssid[] = "TEST_AP";
uint8_t ssid_len = strlen((char *)ssid);
uint8_t psk[] = "12345678";

wlan_sta_set(ssid, ssid_len, psk);
wlan_sta_enable();
```

- 方式二，通过精确配置

```
uint8_t ssid[] = "TEST_AP";
uint8_t ssid_len = strlen((char *)ssid);
uint8_t psk[] = "12345678";

wlan_sta_config_t config;
memset(&config, 0, sizeof(config));

/* ssid */
……

/* psk */
……

/* auth_alg: OPEN */
……

/* proto: WPA | RSN */
……

/* key_mgmt: PSK */
……

/* pairwise: CCMP | TKIP */
……

/* group: CCMP | TKIP | WEP40 | WEP104 */
……

/* scan_ssid: 1 */
config.field = WLAN_STA_FIELD_SCAN_SSID;
config.u.scan_ssid = 1;
if (wlan_sta_set_config(&config) != 0)
    return -1;

if (wlan_sta_enable()!= 0)
    return -1;
```

2.3.2 获取已设置的配置参数

当设置完 WLAN 相关参数后，可以通过 wlan_sta_get_config(&config)函数来获取所对应的配置项，在 get 配置项之前需要设置所要获取的配置项的类别，示例如下：

```
wlan_sta_config_t config;
memset(&config, 0, sizeof(config));
```

```
    config.field = WLAN_STA_FIELD_PSK;
    if (wlan_sta_get_config(&config) != 0)
        return -1;

    printf("psk: %s\n", config.u.psk);
```

2.3.3 扫描可用的 AP 列表

扫描 AP 列表并获取结果可以分别使用 wlan_sta_scan_once()和 wlan_sta_scan_result(&result)来实现，在获取结果之前需要为扫描结果分配空间，并指定希望得到的 AP 数目，那么返回结果将优先提供信号最好的 AP 节点信息，AP 节点信息结构 wlan_sta_scan_ap_t 定义如下：

```
typedef struct wlan_sta_ap {
    wlan_ssid_t   ssid;
    uint8_t       bssid[6];
    uint8_t       channel;
    uint16_t  beacon_int;
    int    freq;
    int    rssi;
    int    level;
    int    wpa_flags;
    int    wpa_cipher;
    int    wpa_key_mgmt;
    int    wpa2_cipher;
    int    wpa2_key_mgmt;
}wlan_sta_ap_t;
```

SCAN 操作示例如下：

- 设置一次 scan

```
    wlan_sta_scan_once();
```

- 获取 scan 结果，以最多获取 10 个 AP 节点为例

```
    #include "net/wlan/wlan.h"

    wlan_sta_scan_results_t results;
    results.size = 10;      //最多获取 10 个 AP 节点信息
    results.ap = malloc(results.size * sizeof(wlan_sta_ap_t));
    if (results.ap == NULL) {
        ……                  //出错处理
    }
    if (wlan_sta_scan_result(&results) == 0) {
        ……                  //AP 信息处理，实际获取 AP 节点数为 results.num
    }
    cmd_free(results.ap);
```

2.3.4 设置扫描间隔

正确配置目标 AP 的信息，并调用 wlan_sta_enable()后，在连接到目标 AP 前，Station 将以固定时间间隔进行扫描。以设置扫描间隔为 10 秒为例：

```
    wlan_sta_scan_interval(10);
```

2.3.5 设置最大扫描 AP 个数

若周围环境中 AP 较多，由于设备默认只保存 20 个扫描结果，其他 AP 信息会被丢弃，设备可能无法扫描到用户想要的 AP。通过设置最大扫描 AP 个数可以解决上述问题，但同时也会占用更多内存。

调用以下接口进行最大扫描 AP 个数设置，以设置最大个数为 50 为例：

```
    wlan_sta_bss_max_count(50);
```

2.3.6 移除设定时长内未更新的 AP 节点

每次扫描都会更新扫描到的 AP 列表，调用 wlan_sta_bss_flush()可以从 AP 列表中移除设定时长内未更新的 AP 节点，以移除 30s 内未更新的 AP 节点为例：

```
    wlan_sta_bss_flush(30);
```

2.3.7 连接、断开连接以及获取连接状态

Station 连接上目标 AP 后，可通过以下 API 断开与 AP 的连接：

```
    wlan_sta_disconnect();
```

断开连接后，可调用以下 API 重新连接目标 AP：

```
    wlan_sta_connect();
```

此外，Station 还可以获取当前连接状态，连接状态在代码中定义如下：

```
    typedef enum wlan_sta_states {
    WLAN_STA_STATE_DISCONNECTED = 0,
    WLAN_STA_STATE_CONNECTED = 1,
    } wlan_sta_states_t;
```

获取连接状态的方式为：

```
    wlan_sta_states_t state;
    wlan_sta_state(&state);
```

2.3.8 获取已连接 AP 节点的信息

Station 连接上目标 AP 后，可通过以下 API 获得已连接 AP 的信息：

```
    wlan_sta_ap_t *ap = malloc(sizeof(wlan_sta_ap_t));
    if (ap == NULL) {
        ……//出错处理
    }

    wlan_sta_ap_info(ap);
```

2.3.9 通过 WPS 连接 AP

该系统支持通过 WPS 方式连接目标 AP。启动 PBC（BUTTON）模式 WPS 连接方式为：

```
int wlan_sta_wps_pbc(void);
```

启动 PIN 码模式 WPS 连接方式为：

```
int wlan_sta_wps_pin_set(wlan_sta_wps_pin_t *wps);
```

此外，还可以获取一个随机合法的 PIN 码。

```
int wlan_sta_wps_pin_get(wlan_sta_wps_pin_t *wps);
```

2.4 AP 模式操作

2.4.1 启动一个 AP 节点

上一章节讲述了通过调用 net_sys_start(mode)的方式来启动工作在 Station 模式下的网络子系统，同样的使用 net_sys_start(mode)并设置 mode 为 WLAN_MODE_HOSTAP 即可将 WLAN 设置为 AP 模式，如果系统先启动在其他模式，可以直接调用切换模式的接口，如下：

```
net_switch_mode(WLAN_MODE_HOSTAP);
```

AP 模式启动之后需要通过一系列操作配置 AP 模式的网络参数，下面一一说明。

1. 配置工作在 OPEN 模式的 AP，其 SSID 为 TEST_AP，注意系统默认 AP 为 AP-XRADIO，修改时先 disable 再 enable 完成配置更新。

```
#include <string.h>
#include "net/wlan/wlan.h"

uint8_t ssid[] = "TEST_AP";
uint8_t ssid_len = strlen((char *)ssid);

wlan_ap_set(ssid, ssid_len, NULL);
wlan_ap_disable();
wlan_ap_enable();
```

2. 设置工作在加密模式的 AP，其 SSID 为 TEST_AP，PSK 为 12345678（此处只选择设置最通用的模式，WEP 模式不推荐不做描述说明）

```
#include <string.h>
#include "net/wlan/wlan.h"

uint8_t ssid[] = "TEST_AP";
uint8_t ssid_len = strlen((char *)ssid);
uint8_t psk[] = "12345678";

wlan_ap_set(ssid, ssid_len, psk);
wlan_ap_disable();
wlan_ap_enable();
```

2.4.2 Set/Get AP 节点配置

可以通过 wlan_ap_set_config()接口和 wlan_ap_get_config()来设置或获取相关参数，在 SET/GET 的调用中，通过制定 wlan_ap_config 的 field 参数来指定所要设置或者获取的参数值，目前能够配置的参数类型如下：

```
typedef enum wlan_ap_field {
```

```
    WLAN_AP_FIELD_SSID = 0,
    WLAN_AP_FIELD_PSK,
    WLAN_AP_FIELD_KEY_MGMT,
    WLAN_AP_FIELD_WPA_CIPHER,
    WLAN_AP_FIELD_RSN_CIPHER,
    WLAN_AP_FIELD_PROTO,
    WLAN_AP_FIELD_AUTH_ALG,
    WLAN_AP_FIELD_GROUP_REKEY,
    WLAN_AP_FIELD_STRICT_REKEY,
    WLAN_AP_FIELD_GMK_REKEY,
    WLAN_AP_FIELD_PTK_REKEY,
    WLAN_AP_FIELD_HW_MODE,
    WLAN_AP_FIELD_IEEE80211N,
    WLAN_AP_FIELD_CHANNEL,
    WLAN_AP_FIELD_BEACON_INT,
    WLAN_AP_FIELD_DTIM,
    WLAN_AP_FIELD_MAX_NUM_STA,

    WLAN_AP_FIELD_NUM,
};
```

此处以 SET/GET channel 为例说明如下：

1. Set Channel

```
wlan_ap_config_t config = {0};
config.field = WLAN_AP_FIELD_CHANNEL;
config.u.channel = 6;
wlan_ap_set_config(&config);
```

2. Get Channel

```
wlan_ap_config_t config = {0};
config.field = WLAN_AP_FIELD_CHANNEL;
wlan_ap_get_config(&config);
```

2.4.3 获取已连接 Station 的数量和信息

AP 可通过以下方式获得当前已连接 Station 的数量。

```
int num;
wlan_ap_sta_num(&num);
```

此外，AP 还可以获取当前已连接 Station 的信息。以最多获取 5 个已连接 Station 信息为例：

```
    wlan_ap_stas_t stas;
    stas.size = 5;          //最多获取 5 个已连接 Station 信息
    stas.sta = malloc(stas.size * sizeof(wlan_ap_sta_t));
    if (stas.sta == NULL) {
        ……                  //出错处理
    }

    if (wlan_ap_sta_info(&stas) == 0) {
        ……                  //处理获取的 Station 信息，实际获取 Station 个数为 stas.num
    }
    free(stas.sta);
```

2.4.4 AP 模式下扫描

AP 模式下进行扫描并获取结果可以分别使用 wlan_ap_scan_once()和 wlan_ap_scan_result(&result)来实现，在获取结果之前需要为扫描结果分配空间，并指定希望得到的 AP 数目，那么返回结果将优先提供信号最好的 AP 节点信息，AP 节点信息结构与 STA 模式下使用的结构相同，均为 wlan_sta_scan_ap_t

SCAN 操作示例如下：

- 设置一次 scan

```
wlan_ap_scan_once();
```

- 获取 scan 结果，以最多获取 10 个 AP 节点为例

```
#include "net/wlan/wlan.h"

wlan_sta_scan_results_t results;
results.size = 10;      //最多获取 10 个 AP 节点信息
results.ap = malloc(results.size * sizeof(wlan_sta_ap_t));
if (results.ap == NULL) {
    ……                  //出错处理
}
if (wlan_ap_scan_result(&results) == 0) {
    ……                  //AP 信息处理，实际获取 AP 节点数为 results.num
}
cmd_free(results.ap);
```

2.5 Monitor 模式操作

用户基于某些特殊的应用场景，需要用到捕获空气中任意包的模式，该模式即为 monitor 模式。

使用 net_sys_start(mode)并设置 mode 为 WLAN_MODE_MONITOR 即可将 WLAN 设置为 monitor 模式，如果系统先启动在其他模式，可以直接调用切换模式的接口，如下：

```
net_switch_mode(WLAN_MODE_MONITOR);
```

在 Monitor 模式下，用户需要注册一个回调函数对捕获的包进行处理，调用以下接口进行回调函数的注册，以回调函数名为 rx_cb 为例：

```
wlan_monitor_set_rx_cb(g_wlan_netif, rx_cb);
```

当退出 monitor 模式时，需要将注册的回调函数清除，方法如下：

```
wlan_monitor_set_rx_cb(g_wlan_netif, NULL);
```

2.6 配网功能的使用

配网是设备在没有输入接口的情况下得以连接上无线网络的一种功能。已连接的设备（通常是手机）通过广播所要连接的 AP 的加密的 SSID 和 Password，而设备端通过监听并解密信息来获得 SSID 和 Password，从而发起连接请求给指定的 AP，解密使用的 key 可以通过 API 调用进行设置。

具体配网功能的实现以及使用，可以参考文档《XRADIO_WLAN_Config_Developer_Guide-CN.pdf》。