# XR806 WLAN 应用开发指南

版本号：1.1

发布时间：2021-04-29

# 版本历史

| 版本 | 日期 | 责任人 | 版本描述 |
|---|---|---|---|
| 1.1 | 2021-04-29 | AWA 1029 | 添加 STA 模式下连接 WPA3 模式 AP 的示例。 |
| 1.0 | 2020-11-18 | AWA 1029 | 创建文档。 |

# 目录

# 表格目录

# 1 前言

## 1.1 文档简介

本文档主要介绍 WLAN 模块的基本原理和使用方法，并说明使用该模块的注意事项，以方便开发者正确使用该模块进行开发。

## 1.2 目标读者

XR806 SDK 用户、WLAN 开发使用人员。

## 1.3 适用范围

此文档适用于 XR806 SDK，支持 XR806 系列芯片产品。

## 1.4 文档约定

### 1.4.1 标志说明

本文档采用各种醒目的标志来表示在操作过程中应该特别注意的地方，这些标志的含义如下：

| 标识 | 说明 |
| --- | --- |
| 警告 | 该标志后的说明应给予格外关注，如果不遵守，可能会导致人员受伤或死亡。 |
| 注意 | 提醒操作中应注意的事项。不当的操作可能会损坏器件，影响可靠性、降低性能等。 |
| 说明 | 为准确理解文中指令、正确实施操作而提供的补充或强调信息。 |
| 窍门 | 一些容易忽视的小功能、技巧。了解这些功能或技巧能帮助解决特定问题或者节省操作时间。 |

### 1.4.2 地址与数据描述方法约定

本文档在描述地址、数据时遵循如下约定：

| 符号 | 例子 | 说明 |
| --- | --- | --- |
| 0x | 0x0200，0x79 | 地址或数据以 16 进制表示。 |
| 0b | 0b010，0b00 000 111 | 数据采用二进制表示(寄存器描述除外)。 |
| X | 00X，XX1 | 数据描述中，X 代表 0 或 1。<br>例如，00X 代表 000 或 001；XX1 代表 001，011，101 或 111。 |

### 1.4.3 数值单位约定

本文档在描述数据容量（如 NAND 容量）时，单位词头代表的是 1024 的倍数；描述频率、数据速率等时则代表的是 1000 的倍数。具体如下：

| 类型 | 符号 | 对应数值 |
| --- | --- | --- |
| 数据容量（如 NAND 容量） | 1 K | 1024 |
| | 1 M | 1 048 576 |
| | 1 G | 1 073 741 824 |
| 频率，数据速率等 | 1 k | 1000 |
| | 1 M | 1 000 000 |
| | 1 G | 1 000 000 000 |

# 2 概述

## 2.1 背景说明

XR806 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 XR806 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。

## 2.2 规格特性

XR806 WLAN 模块的规格特性如下表所示。

**表 2-1 WLAN 模块的规格特性**

| 规格类型 | 支持规格 | 规格描述 | 备注 |
|---|---|---|---|
| WLAN 模式 | STA 模式 | 设备工作在 station 模式，可以连接指定的 AP 或路由器等 | |
| | SoftAP 模式 | 设备工作在 AP 模式，支持其他 station 设备进行连接 | |
| | Monitor 模式 | 设备工作在混杂模式，可以接收空中任意包 | |
| 加密方式 | OPEN | 无加密方式 | |
| | WPA/WPA2 | 基于 WPA/WPA2 加密算法的连接方式 | |
| | WEP | 基于 WEP 加密算法的连接方式 | |
| | WPS | 基于 WPS 的连接方式 | |

## 2.3 文件位置

以 SDK 包为根目录，WLAN 模块涉及到的主要文件位置如下。

**表 2-2 WLAN 模块的文件位置**

| 组件名 | 文件分类 | 文件位置 |
|---|---|---|
| WLAN | 库文件 | ./lib/xradio_v3/libwlan.a |
| | 头文件 | ./include/net/wlan/wlan.h |
| | 示例工程 | ./project/example/wlan/ |

**说明**

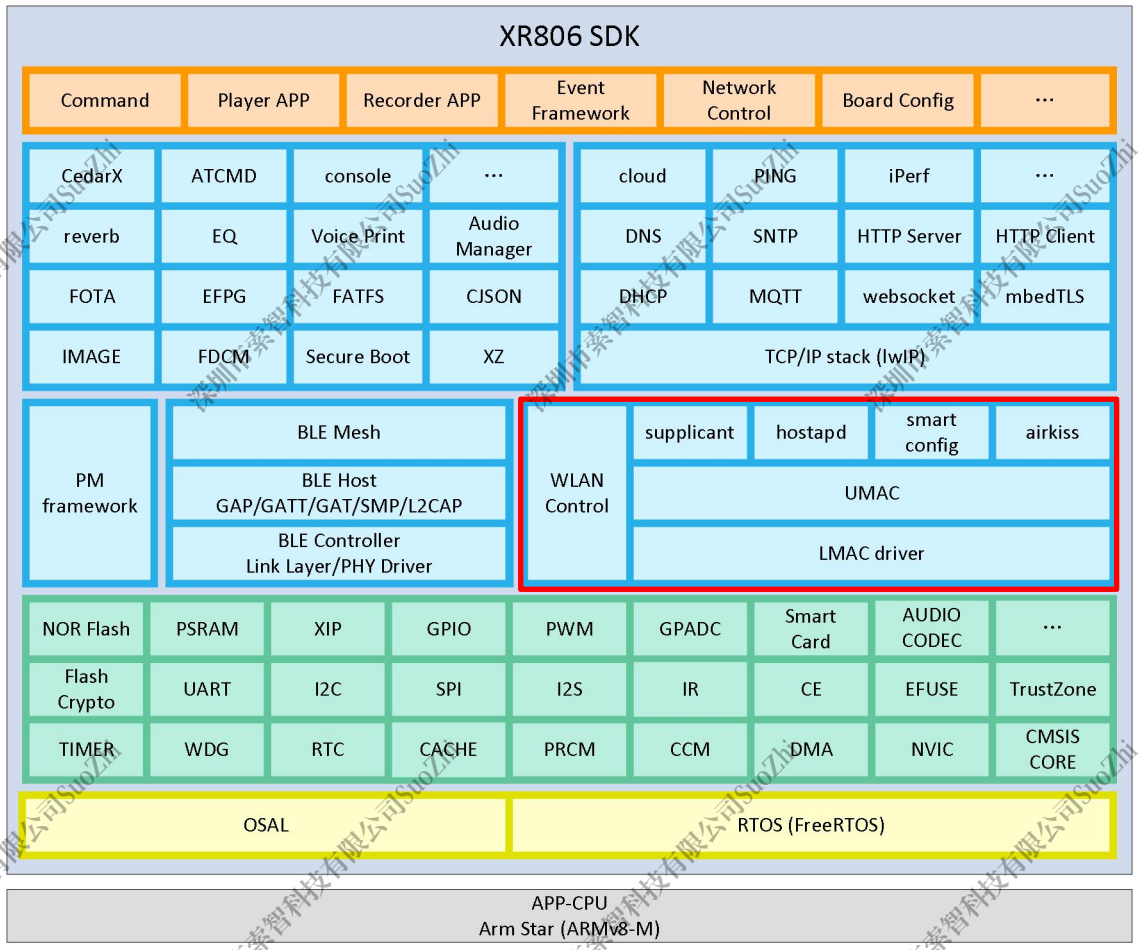
XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

# 3 技术说明

## 3.1 模块结构

WLAN 模块与 SDK 中其他模块之间的关系如下图所示。在框架层 framework 和 command 中都有使用到 WLAN 相关接口，可以对 WLAN 进行指定操作以及对 WLAN 消息进行处理。

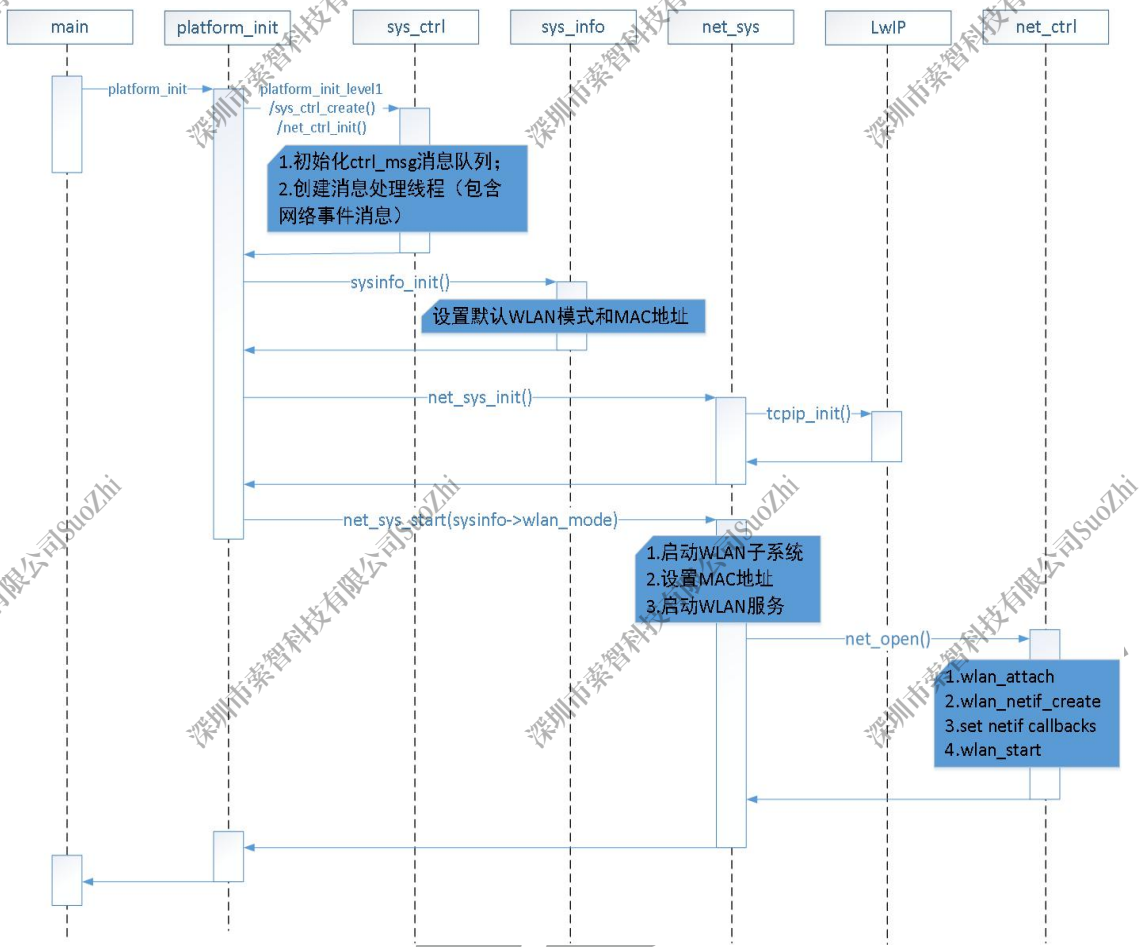
**图 3-1 XR806 SDK 框图**




## 3.2 WLAN 初始化流程

WLAN 的初始化默认会在 platform_init()中被调用，启动模式保存在 sysinfo 区域中，默认启动为 STA 模式。WLAN 模块的初始化流程框图如下所示。

图 3-2 WLAN 模块初始化流程




由上图可以看出，WLAN 模块初始化流程几个重要的过程如下：

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。

## 3.3 WLAN 消息处理

当前网络消息的传递，无论是 STA 还是 AP 模式的，都是使用 sys_ctrl 的 g_sys_publisher 来传递消息的。

当网络端产生事件时，会使用已注册的函数 net_ctrl_msg_send 就可以发送 CTRL_MSG_TYPE_NETWORK 类型的消息到 sys_ctrl 中。

在系统初始化时，就已经注册了回调函数 net_ctrl_msg_process，可以对 CTRL_MSG_TYPE_NETWORK 消息进行处理。用户也可以自己再注册一个回调函数，然后对需要的网络消息进行自定义处理，注册方式如下：

```
void net_msg_receiver(uint32_t event, uint32_t data, void *arg)
```

```
{
	uint16_t type = EVENT_SUBTYPE(event);

	switch (type) {
	case NET_CTRL_MSG_WLAN_CONNECTED:
		break;
	case NET_CTRL_MSG_WLAN_DISCONNECTED:
		break;
	case NET_CTRL_MSG_WLAN_SCAN_SUCCESS:
	case NET_CTRL_MSG_WLAN_SCAN_FAILED:
		break;
	case NET_CTRL_MSG_WLAN_4WAY_HANDSHAKE_FAILED:
	case NET_CTRL_MSG_WLAN_CONNECT_FAILED:
	case NET_CTRL_MSG_CONNECTION_LOSS:
		break;
	case NET_CTRL_MSG_NETWORK_UP:
		break;
	case NET_CTRL_MSG_NETWORK_DOWN:
		break;
	default:
		printf("unknown msg (%u, %u)\n", type, data);
		break;
	}
}

int main(void)
{
	platform_init();
	observer_base *ob = sys_callback_observer_create(CTRL_MSG_TYPE_NETWORK,
                                                     NET_CTRL_MSG_ALL,
                                                     net_msg_receiver,
                                                     NULL);
	if (ob == NULL)
		return -1;
	if (sys_ctrl_attach(ob) != 0)
		return -1;

	return 0;
}
```

对于具体的事件消息，存在不同的模式使用情况，其含义解释如下：

表 3-1 WLAN 消息简介

| 消息类型 | 使用模式 | 说明 |
| --- | --- | --- |
| NET_CTRL_MSG_WLAN_CONNECTED | STA/SoftAP | MAC 层 STA 连上 AP 或者 SoftAP 成功建立 |
| NET_CTRL_MSG_WLAN_DISCONNECTED | STA/SoftAP | MAC 层 STA 断开连接或者 SoftAP 停止 |
| NET_CTRL_MSG_WLAN_SCAN_SUCCESS | STA | STA 扫描成功 |
| NET_CTRL_MSG_WLAN_SCAN_FAILED | STA | STA 扫描失败 |
| NET_CTRL_MSG_WLAN_4WAY_HANDSHAKE_FAILED | STA | STA 连接 AP 时 4 次握手失败 |
| NET_CTRL_MSG_WLAN_CONNECT_FAILED | STA | STA 连接失败 |
| NET_CTRL_MSG_CONNECTION_LOSS | STA | STA 在连接状态时丢失 AP |
| NET_CTRL_MSG_NETWORK_UP | STA/SoftAP | 网络层建立连接获取到 IP |
| NET_CTRL_MSG_NETWORK_DOWN | STA/SoftAP | 网络层断开连接 |

# 4 应用说明

## 4.1 应用简述

WLAN 模块已经内嵌到 XR806 SDK 中，用户通过 WLAN 模块提供的接口即可调用 WLAN 的相关功能。

### 4.1.1 配置说明

WLAN 模块的启用需要使能 SDK 的 net 和 WLAN 相关功能，具体配置项如下表所示。

**表 4-1 XR806 WLAN 模块工程配置说明**

| 配置项 | 配置说明 |
| --- | --- |
| WLAN 功能使能 | 设置说明：<br>此项配置用于在 SDK 中启用 WLAN 功能。<br><br>设置方式：<br>1、在 prj_config.h 文件，添加或修改以下定义，其中 1 为使能 NET 功能，0 为不使能 NET 功能。<br>/* network and wlan enable/disable */<br>#define PRJCONF_NET_EN 1<br><br>2、在控制台执行命令“make menuconfig”，将“NET and WLAN”功能项打开。 |

在使用 SoftAP 模式时，可以设置启动时的默认参数，具体配置如下表所示。

**表 4-2 XR806 SoftAP 默认启动配置说明**

| 配置项 | 配置说明 |
| --- | --- |
| SoftAP 默认启动配置 | 设置说明：<br>此项配置用于在 SDK 中设置 SoftAP 的默认启动参数。<br><br>设置方式：<br>1、在用户的应用代码中增加一个 g_wlan_ap_default_conf 结构体的定义并给其默认值赋值，可以参考 platform_init.c 文件中 g_wlan_ap_default_conf 结构体的写法。需要包含头文件 net/wlan/wlan_defs.h。<br>建议用户使用此方式修改配置。<br>2、直接修改 platform_init.c 文件中 g_wlan_ap_default_conf 结构体各个成员的默认值，例如 SSID、password、加密方式等。<br>__weak const wlan_ap_default_conf_t g_wlan_ap_default_conf = {<br>.ssid = "AP-XRADIO",<br>.ssid_len = 9,<br>.psk = "123456789",<br>.hw_mode = WLAN_AP_HW_MODE_IEEE80211G,<br>.....<br>}; |

| 配置项 | 配置说明 |
| --- | --- |
| | 不建议用户使用此方式修改配置。 |

**说明**

platform_init.c 文件中的 g_wlan_ap_default_conf 结构体被定义为__weak 类型，因此用户可以通过在应用代码中定义同名结构体将其覆盖。不建议用户直接修改此处的定义。

## 4.2 接口说明

WLAN 模块总共提供 4 套接口，主要功能点包括网络子系统开关、STA 模式操作、SoftAP 模式操作以及 monitor 模式操作等。

**表 4-3 WLAN 模块接口简介**

| 接口名 | 简要介绍 |
| --- | --- |
| 网络子系统接口 | 用于开关整个网络子系统，以及控制 WLAN 的工作模式 |
| STA 模式接口 | 用于 STA 模式的操作，例如扫描连接等 |
| SoftAP 模式接口 | 用于 SoftAP 模式的操作，例如启动和关闭等 |
| monitor 模式接口 | 用于 monitor 模式的操作，例如注册 rx 回调处理等 |

以下仅列举常用的一些接口，完整接口以及说明请参考 WLAN 头文件内容。

文件路径为：./include/net/wlan/wlan.h

### 4.2.1 网络子系统接口

#### 4.2.1.1 net_sys_start

**表 4-4 net_sys_start 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int net_sys_start(enum wlan_mode mode); |
| 功能 | 打开 WLAN 的指定模式，如果该模式已经打开，会直接返回 |
| 参数 | enum wlan_mode mode<br>含义解释：需要启动的 WLAN 模式<br>使用说明：取值参考表 4-4 wlan_mode 枚举类型说明 |
| 返回值 | 0：成功<br>-1：失败 |

表 4-5 wlan_mode 枚举类型说明

| 成员项 | 说明 |
| --- | --- |
| WLAN_MODE_STA | STA 模式 |
| WLAN_MODE_HOSTAP | SoftAP 模式 |
| WLAN_MODE_MONITOR | Monitor 模式，即混杂模式 |
| WLAN_MODE_NUM | 模式枚举最大值，不可被使用 |
| WLAN_MODE_INVALID | 等同于 WLAN_MODE_NUM，不可被使用 |

#### 4.2.1.2 net_sys_stop

表 4-6 net_sys_stop 接口函数说明

| 信息项 | 说明 |
| --- | --- |
| 原型 | int net_sys_stop(enum wlan_mode mode); |
| 功能 | 关闭 WLAN 的指定模式，如果该模式已经关闭，会直接返回 |
| 参数 | enum wlan_mode mode<br>含义解释：需要关闭的 WLAN 模式<br>使用说明：取值参考表 4-4 wlan_mode 枚举类型说明 |
| 返回值 | 0：成功<br>-1：失败 |

#### 4.2.1.3 net_switch_mode

表 4-7 net_switch_mode 接口函数说明

| 信息项 | 说明 |
| --- | --- |
| 原型 | int net_switch_mode (enum wlan_mode mode); |
| 功能 | 切换 WLAN 的指定模式，如果该模式已经打开，会直接返回 |
| 参数 | enum wlan_mode mode<br>含义解释：需要切换的 WLAN 模式<br>使用说明：取值参考表 4-4 wlan_mode 枚举类型说明 |
| 返回值 | 0：成功<br>-1：失败 |

### 4.2.2 STA 模式接口

#### 4.2.2.1 wlan_sta_set

**表 4-8 wlan_sta_set 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_sta_set (uint8_t *ssid, uint8_t ssid_len, uint8_t *psk); |
| 功能 | STA 模式下设置需要连接的 AP 的 ssid 和 psk，支持以下连接模式：<br>OPEN，WPA/WPA2/WPA3-PSK |
| 参数 | uint8_t *ssid<br>含义解释：设置需要连接的 ssid 字符串<br><br>uint8_t ssid_len<br>含义解释：ssid 字符串的长度<br><br>uint8_t *psk<br>含义解释：非 open 模式时的密码字符串，若为 open 模式，则设置为 NULL 或者 “” |
| 返回值 | 0：成功<br>-1：失败 |

#### 4.2.2.2 wlan_sta_config

**表 4-9 wlan_sta_config 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_sta_config(uint8_t *ssid, uint8_t ssid_len, uint8_t *psk, uint32_t flag); |
| 功能 | STA 模式下设置需要连接的 AP 的 ssid 和 psk，支持以下连接模式：<br>OPEN，WPA/WPA2/WPA3-PSK<br>此接口可以通过参数 flag 指定是否支持 WPA3 模式连接 |
| 参数 | uint8_t *ssid<br>含义解释：设置需要连接的 ssid 字符串<br><br>uint8_t ssid_len<br>含义解释：ssid 字符串的长度<br><br>uint8_t *psk<br>含义解释：非 open 模式时的密码字符串，若为 open 模式，则设置为 NULL 或者 “”<br><br>uint32_t flag<br>含义解释：若使用 WPA3 模式或兼容模式进行连接，则设置为 WLAN_STA_CONF_FLAG_WPA3，若仅使用 WPA/WPA2 模式进行连接，则设置为 0 |
| 返回值 | 0：成功 |

| 信息项 | 说明 |
| --- | --- |
| | -1：失败 |

#### 4.2.2.3 wlan_sta_set_config

**表 4-10 wlan_sta_set_config 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_sta_set_config (wlan_sta_config_t *config); |
| 功能 | STA 模式下设置需要的 wlan 参数字段信息，包括 ssid/password/加密方式 等 |
| 参数 | wlan_sta_config_t *config<br>含义解释：需要设置的 wlan 参数 |
| 返回值 | 0：成功<br>-1：失败 |

#### 4.2.2.4 wlan_sta_get_config

**表 4-11 wlan_sta_get_config 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_sta_get_config (wlan_sta_config_t *config); |
| 功能 | STA 模式下获取需要的 wlan 参数字段信息，包括 ssid/password/加密方式 等 |
| 参数 | wlan_sta_config_t *config<br>含义解释：需要获取的 wlan 参数 |
| 返回值 | 0：成功<br>-1：失败 |

#### 4.2.2.5 wlan_sta_enable

**表 4-12 wlan_sta_enable 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_sta_enable (void); |
| 功能 | STA 模式使能，supplicant 会自动进行扫描和连接的操作 |
| 参数 | 无 |
| 返回值 | 0：成功<br>-1：失败 |

#### 4.2.2.6 wlan_sta_disable

**表 4-13 wlan_sta_disable 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_sta_disable (void); |
| 功能 | STA 模式关闭 |
| 参数 | 无 |
| 返回值 | 0：成功<br>-1：失败 |

#### 4.2.2.7 wlan_sta_scan_once

**表 4-14 wlan_sta_scan_once 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_sta_scan_once (void); |
| 功能 | STA 模式进行一次扫描，若需要扫描指定的 ssid，使用 wlan_sta_config()配置 ssid 即可 |
| 参数 | 无 |
| 返回值 | 0：成功<br>-1：失败 |

#### 4.2.2.8 wlan_sta_scan_result

**表 4-15 wlan_sta_scan_result 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_sta_scan_result (wlan_sta_scan_results_t *results); |
| 功能 | STA 模式获取上一次的扫描结果，results 中的 size 可以指定需要获取的 ap 个数 |
| 参数 | wlan_sta_scan_results_t *results<br>含义解释：用于获取扫描结果的结构体指针 |
| 返回值 | 0：成功<br>-1：失败 |

### 4.2.3 SoftAp 模式接口

#### 4.2.3.1 wlan_ap_set

**表 4-16 wlan_ap_set 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_ap_set (uint8_t *ssid, uint8_t ssid_len, uint8_t *psk); |

| 信息项 | 说明 |
| --- | --- |
| 功能 | SoftAP 模式下设置自身的 ssid 和 password，此配置需要将 SoftAP disable 后再 enable 才会生效 |
| 参数 | uint8_t *ssid<br>含义解释：设置自身的 ssid 字符串<br><br>uint8_t ssid_len<br>含义解释：ssid 字符串的长度<br><br>uint8_t *psk<br>含义解释：非 open 模式时的密码字符串，若为 open 模式，则设置为 NULL 或者 “” |
| 返回值 | 0：成功<br>-1：失败 |

#### 4.2.3.2 wlan_ap_set_config

**表 4-17 wlan_ap_set_config 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_ap_set_config (wlan_ap_config_t *config); |
| 功能 | SoftAP 模式下设置需要的 wlan 参数字段信息，包括 ssid/password/加密方式 等，此配置需要将 SoftAP disable 后再 enable 才会生效 |
| 参数 | wlan_ap_config_t *config<br>含义解释：需要设置的 wlan 参数 |
| 返回值 | 0：成功<br>-1：失败 |

#### 4.2.3.3 wlan_ap_get_config

**表 4-18 wlan_ap_get_config 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_ap_get_config (wlan_ap_config_t *config); |
| 功能 | STA 模式下获取需要的 wlan 参数字段信息，包括 ssid/password/加密方式 等 |
| 参数 | wlan_ap_config_t *config<br>含义解释：需要获取的 wlan 参数 |
| 返回值 | 0：成功<br>-1：失败 |

#### 4.2.3.4 wlan_ap_enable

**表 4-19 wlan_ap_enable 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_ap_enable (void); |
| 功能 | SoftAP 模式使能，默认切换到 SoftAP 模式时已经是使能状态了 |
| 参数 | 无 |
| 返回值 | 0：成功<br>-1：失败 |

#### 4.2.3.5 wlan_ap_disable

**表 4-20 wlan_ap_disable 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_ap_disable (void); |
| 功能 | SoftAP 模式关闭 |
| 参数 | 无 |
| 返回值 | 0：成功<br>-1：失败 |

#### 4.2.3.6 wlan_ap_scan_once

**表 4-21 wlan_ap_scan_once 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_ap_scan_once (void); |
| 功能 | SoftAP 模式进行一次扫描 |
| 参数 | 无 |
| 返回值 | 0：成功<br>-1：失败 |

#### 4.2.3.7 wlan_ap_scan_result

**表 4-22 wlan_ap_scan_result 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_ap_scan_result (wlan_sta_scan_results_t *results); |
| 功能 | SoftAP 模式获取上一次的扫描结果，results 中的 size 可以指定需要获取的 ap 个数 |
| 参数 | wlan_sta_scan_results_t *results<br>含义解释：用于获取扫描结果的结构体指针，该结构体与 STA 模式下使用的相 |

| 信息项 | 说明 |
| --- | --- |
| | 同 |
| 返回值 | 0：成功<br>-1：失败 |

### 4.2.4 Monitor 模式接口

#### 4.2.4.1 wlan_monitor_set_rx_cb

**表 4-23 wlan_monitor_set_rx_cb 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_monitor_set_rx_cb(struct netif *nif, wlan_monitor_rx_cb cb); |
| 功能 | Monitor 模式设置接收包的回调处理函数 |
| 参数 | struct netif *nif<br>含义解释：网络接口<br><br>wlan_monitor_rx_cb cb<br>含义解释：回调处理函数指针 |
| 返回值 | 0：成功<br>-1：失败 |

#### 4.2.4.2 wlan_monitor_set_channel

**表 4-24 wlan_monitor_set_channel 接口函数说明**

| 信息项 | 说明 |
| --- | --- |
| 原型 | int wlan_monitor_set_channel(struct netif *nif, int16_t channel); |
| 功能 | Monitor 模式切换接收信道 |
| 参数 | struct netif *nif<br>含义解释：网络接口<br><br>int16_t channel<br>含义解释：需要切换的信道值 |
| 返回值 | 0：成功<br>-1：失败 |

# 5 示例说明

本章节以启动 STA、SoftAP 以及 monitor 模式为例，简要说明 WLAN 模块接口的使用。

## 5.1 STA 模式示例

### 5.1.1 示例简介

本示例通过演示 WLAN 的 STA 模式连接指定的路由器之后再断开连接的过程，简要介绍对应接口的使用方法。

### 5.1.2 关键实现

系统平台的初始化在 XR806 SDK 的 platform_init()函数中完成，具体请阅读此函数。之后会切换到 STA 模式，然后设置需要连接的 SSID 和 password，使能 STA 模式后会自动进行连接。等待 60 秒时间后，程序会断开 STA 连接，示例结束。

具体操作见下面的示例代码。

```
char * sta_ssid = "your_ap_name";//修改为实际使用的 SSID
char * sta_psk = "your_ap_password";//修改为实际使用的 password
void sta_test(void)
{
    /* switch to sta mode */
    net_switch_mode(WLAN_MODE_STA);

    /* set ssid and password to wlan */
    wlan_sta_set((uint8_t *)sta_ssid, strlen(sta_ssid), (uint8_t *)sta_psk);

    /* start scan and connect to ap automatically */
    wlan_sta_enable();

    OS_Sleep(60);

    /* After one minute, disconnect from AP */
    wlan_sta_disable();
}
```

#### 5.1.2.1 获取方式

WLAN 模块的 STA 模式示例有示例工程代码，位于 XR806 SDK 的/project/example/wlan 目录，以下此示例工程简称为 WLAN 示例工程。

**说明**

XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

#### 5.1.2.2 准备工作

WLAN 示例工程的硬件准备如下：

1. 评估板：运行示例工程代码。
2. 串口线：连接评估板的 Uart0 插针，用于 console 控制台的输入输出。
3. PC 机：用于镜像烧录和 console 控制的输入输出。

WLAN 示例工程的软件准备，包括烧写工具、代码编译和烧写操作，请参见《XR806_SDK_快速入门指南》。

#### 5.1.2.3 操作步骤

WLAN 示例工程无需控制命令，需要先将代码中的 SSID 和 password 修改为实际使用的值，然后进行编译烧写，完成后，复位即可，示例代码自动运行。

### 5.1.3 效果展示

在评估板中运行 WLAN 示例程序后，在控制台中会打印如下所示。

```
[net INF] no need to switch wlan mode 0
en1: Trying to associate with 80:37:73:f7:dc:f4 (SSID='AW-ANC-NETGEAR-JNR3300#24' freq=2412 MHz)
[net INF] msg <wlan scan success>
en1: Associated with 80:37:73:f7:dc:f4
en1: WPA: Key negotiation completed with 80:37:73:f7:dc:f4 [PTK=CCMP GTK=CCMP]
en1: CTRL-EVENT-CONNECTED - Connection to 80:37:73:f7:dc:f4 completed [id=0 id_str=]
[net INF] msg <wlan connected>
[net INF] netif is link up
[net INF] start DHCP...
[net INF] IPv6 addr state change: 0x0 --> 0x1
[net INF] msg <network IPv6 state>
[net INF] netif (IPv4) is up
[net INF] address: 192.168.47.25
[net INF] gateway: 192.168.47.1
[net INF] netmask: 255.255.255.0
[net INF] msg <network up>
//等待 60 秒
en1: CTRL-EVENT-DISCONNECTED bssid=80:37:73:f7:dc:f4 reason=3 locally_generated=1
[net INF] msg <wlan disconnected>
[net INF] netif is link down
[net INF] bring down netif
WAR Issue unjoin command(TX) by self.
```

```
[net INF] netif (IPv4) is down
Issue unjoin.
[net INF] msg <network down>
[net INF] stop DHCP
en1: CTRL-EVENT-TERMINATING
WAR join_status:0
```

### 5.1.4 接口使用示例

以下介绍其他常用的接口的使用示例。

#### 5.1.4.1 连接OPEN模式的AP

以 AP 的 SSID 为 TEST_AP 举例如下：

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

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

#### 5.1.4.2 连接WEP模式的AP

以 AP 的 SSID 为 TEST_AP，WEP_KEY0 为 1234567890 举例如下：

```
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;
```

#### 5.1.4.3 连接WPA/WPA2/WPA3模式的AP

以 AP 的 SSID 为 TEST_AP，password 为 12345678 举例如下：

1. 默认以兼容模式连接 AP，此时 AP 若支持 WPA3 模式，则会以 WPA3 模式进行连接

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

wlan_sta_config(ssid, ssid_len, psk, WLAN_STA_CONF_FLAG_WPA3);
wlan_sta_enable();
```

**说明**

上述示例中，使用 wlan_sta_set(ssid, ssid_len, psk)接口进行配置同样会以兼容模式连接 AP。

2. 仅以 WPA/WPA2 模式连接 AP

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

```
    wlan_sta_config(ssid, ssid_len, psk, 0);
    wlan_sta_enable();
```

#### 5.1.4.4 连接隐藏SSID的AP

当 AP 的 SSID 隐藏时，以兼容模式的接口配置 SSID 和密码即可完成连接。

以 AP 工作在 WPA/WPA2 模式，其 SSID 为 TEST_AP，password 为 12345678 举例如下：

```
    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();
```

#### 5.1.4.5 获取已配置的参数

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。

可配置和获取的参数类型列举如下：

```
typedef enum wlan_sta_field {
    WLAN_STA_FIELD_SSID = 0,
    WLAN_STA_FIELD_PSK,
    WLAN_STA_FIELD_WEP_KEY0,
    WLAN_STA_FIELD_WEP_KEY1,
    WLAN_STA_FIELD_WEP_KEY2,
    WLAN_STA_FIELD_WEP_KEY3,
    WLAN_STA_FIELD_WEP_KEY_INDEX,
    WLAN_STA_FIELD_KEY_MGMT,
    WLAN_STA_FIELD_PAIRWISE_CIPHER,
    WLAN_STA_FIELD_GROUP_CIPHER,
    WLAN_STA_FIELD_PROTO,
    WLAN_STA_FIELD_AUTH_ALG,
    WLAN_STA_FIELD_WPA_PTK_REKEY,
    WLAN_STA_FIELD_SCAN_SSID,

    WLAN_STA_FIELD_NUM,
} wlan_sta_field_t;
```

以获取设置好的 password 参数为例，举例如下：

```
    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);
```

#### 5.1.4.6 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;
```

扫描操作示例如下：

1. 进行单次扫描

```
wlan_sta_scan_once();
```

2. 获取扫描结果，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);
```

#### 5.1.4.7 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，STA 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);
```

#### 5.1.4.8 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);
```

#### 5.1.4.9 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);
```

#### 5.1.4.10 连接、断开连接以及获取连接状态

STA 连接上目标 AP 后，可通过以下接口断开与 AP 的连接：

```
wlan_sta_disconnect();
```

断开连接后，可调用以下接口重新连接目标 AP：

```
wlan_sta_connect();
```

此外，STA 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;
```

可以调用以下接口获取连接状态：

```
wlan_sta_states_t state;
```

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

#### 5.1.4.11 获取已连接AP节点的信息

STA 连接上目标 AP 后，可通过以下接口获得已连接 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);
```

#### 5.1.4.12 通过WPS连接AP

WLAN 模块支持通过 WPS 方式连接目标 AP。

调用以下接口启动 PBC（BUTTON）模式 WPS 连接方式：

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

调用以下接口启动 PIN 码模式 WPS 连接方式：

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

调用以下接口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);
```

## 5.2 SoftAP 模式示例

### 5.2.1 示例简介

本示例通过演示 WLAN 的 SoftAP 模式启动后修改 SSID 和 password 的过程，简要介绍对应接口的使用方法。

#### 5.2.1.1 获取方式

WLAN 模块的 SoftAP 模式示例有示例工程代码，位于 XR806 SDK 的/project/example/wlan 目录，以下此示例工程简称为 WLAN 示例工程。

---

**说明**

XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

---

#### 5.2.1.2 准备工作

WLAN 示例工程的硬件准备如下：

1. 评估板：运行示例工程代码。
2. 串口线：连接评估板的 Uart0 插针，用于 console 控制台的输入输出。
3. PC 机：用于镜像烧录和 console 控制的输入输出。

WLAN 示例工程的软件准备，包括烧写工具、代码编译和烧写操作，请参见《XR806_SDK_快速入门指南》。

#### 5.2.1.3 操作步骤

WLAN 示例工程无需控制命令，直接编译烧写，完成后，复位即可，示例代码自动运行。

### 5.2.2 关键实现

系统平台的初始化在 XR806 SDK 的 platform_init()函数中完成，具体请阅读此函数。之后会切换到 SoftAP 模式，然后会先关闭 SoftAP，设置需要使用的 SSID 和 password，再次使能 SoftAP 模式。等待 60 秒时间后，程序会关闭 SoftAP，示例结束。

具体操作见下面的示例代码。

```
char * ap_ssid = "XRADIO_AP";//修改为实际使用的 SSID
char * ap_psk = "12345678";//修改为实际使用的 password
void ap_test(void)
{
    /* switch to ap mode */
    net_switch_mode(WLAN_MODE_HOSTAP);

    /* disable AP to set params*/
    wlan_ap_disable();

    /* set ap's ssid and password */
    wlan_ap_set((uint8_t *)ap_ssid, strlen(ap_ssid), (uint8_t *)ap_psk);

    /* enable ap mode again */
    wlan_ap_enable();

    OS_Sleep(60);

    /* After one minute, disable AP mode */
    wlan_ap_disable();
}
```

### 5.2.3 效果展示

在评估板中运行 WLAN 示例程序后，在控制台中会打印如下所示。

```
interface name: en1
Using interface en1 with hwaddr 34:0c:b7:d1:b0:0a and ssid "AP-XRADIO"
en1: interface state UNINITIALIZED->ENABLED
en1: AP-ENABLED
en1: AP-DISABLED
[net INF] msg <wlan connected>
[net INF] netif is link up
[net INF] bring up netif
[net INF] netif (IPv4) is up
[net INF] address: 192.168.51.1
[net INF] gateway: 192.168.51.1
vif0, AP/GO mode THROTTLE=38
[net INF] netmask: 255.255.255.0
[net INF] msg <network up>
 en1: interface state ENABLED->DISABLED
[net INF] msg <wlan disconnected>
[net INF] netif is link down
Using interface en1 with hwaddr 34:0c:b7:d1:b0:0a and ssid "XRADIO_AP"
en1: interface state DISABLED->ENABLED
en1: AP-ENABLED
[net INF] msg <wlan connected>
[net INF] netif is link up
[net INF] netif is already up
vif0, AP/GO mode THROTTLE=38
//等待 60 秒
en1: AP-DISABLED
en1: interface state ENABLED->DISABLED
[net INF] msg <wlan disconnected>
[net INF] bring down netif
[net INF] netif is link down
[net INF] netif (IPv4) is down
[net INF] msg <network down>
en1: interface state DISABLED->DISABLED
en1: AP-DISABLED
hostapd_free_hapd_data: Interface en1 wasn't started
ELOOP: remaining timeout: 8.388000 eloop_data=0x223590 user_data=0 handler=0x42409d
```

### 5.2.4 接口使用示例

以下介绍其他常用的接口和配置的使用示例。

#### 5.2.4.1 修改SoftAP默认启动配置

如果需要修改 SoftAP 的默认启动配置，只需要在应用代码中增加一个 g_wlan_ap_default_conf 结构体并给其默认值赋值即可。需要包含头文件 net/wlan/wlan_defs.h。

以配置 SoftAP 启动的默认 SSID 为 TEST_AP，password 为 12345678，信道为 6，举例如下：

```
#include "net/wlan/wlan_defs.h"
const wlan_ap_default_conf_t g_wlan_ap_default_conf = {
    .ssid = "TEST_AP",
    .ssid_len = 7,
    .psk = "12345678",
    .hw_mode = WLAN_AP_HW_MODE_IEEE80211G,
    .ieee80211n = 0,
    .key_mgmt = WPA_KEY_MGMT_PSK,
    .wpa_pairwise_cipher = WPA_CIPHER_TKIP,
    .rsn_pairwise_cipher = WPA_CIPHER_CCMP,
    .proto = WPA_PROTO_WPA | WPA_PROTO_RSN,
    .auth_alg = WPA_AUTH_ALG_OPEN,
    .group_rekey = 3600,
    .gmk_rekey = 86400,
    .ptk_rekey = 0,
    .strict_rekey = 1,
    .channel = 6,
    .beacon_int = 100,
    .dtim = 1,
    .max_num_sta = 4,
    .country = {'C', 'N', ' '},
};
```

#### 5.2.4.2 启动OPEN模式的SoftAP

以 SoftAP 的 SSID 为 TEST_AP 举例如下：

```
    #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();
```

#### 5.2.4.3 启动加密模式的SoftAP

设置工作在加密模式的 SoftAP，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。

以 SoftAP 的 SSID 为 TEST_AP，password 为 12345678，举例如下：

```
#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();
```

#### 5.2.4.4 设置和获取SoftAP的参数

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。

可配置的参数类型列举如下：

```
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,
};
```

以设置 SoftAP 的 channel 参数为 6 信道为例：

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

以获取 SoftAP 的 channel 参数为例：

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

#### 5.2.4.5 获取已连接STA的数量和信息

SoftAP 可通过以下方式获得当前已连接 STA 的数量。

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

此外，SoftAP 还可以获取当前已连接 STA 的信息。

以最多获取 5 个已连接 STA 信息为例：

```
    wlan_ap_stas_t stas;
    stas.size = 5;                  //最多获取 5 个已连接 STA 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 STA 信息，实际获取 STA 个数为 stas.num
    }
    free(stas.sta);
```

#### 5.2.4.6 SoftAP模式下扫描

SoftAP 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，可参考 5.1.4.6 章节介绍。

SoftAP 模式下扫描操作示例如下：

1. 进行单次扫描：

```
wlan_ap_scan_once();
```

2. 获取扫描结果，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);
```

## 5.3 Monitor 模式示例

### 5.3.1 示例简介

本示例通过演示 WLAN 的 monitor 模式启动后接收帧的过程，简要介绍对应接口的使用方法。

#### 5.3.1.1 获取方式

WLAN 模块的 monitor 模式示例有示例工程代码，位于 XR806 SDK 的/project/example/wlan 目录，以下此示例工程简称为 WLAN 示例工程。

**说明**

XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

#### 5.3.1.2 准备工作

WLAN 示例工程的硬件准备如下：

1. 评估板：运行示例工程代码。
2. 串口线：连接评估板的 Uart0 插针，用于 console 控制台的输入输出。
3. PC 机：用于镜像烧录和 console 控制的输入输出。

WLAN 示例工程的软件准备，包括烧写工具、代码编译和烧写操作，请参见《XR806_SDK_快速入门指南》。

#### 5.3.1.3 操作步骤

WLAN 示例工程无需控制命令，直接编译烧写，完成后，复位即可，示例代码自动运行。

### 5.3.2 关键实现

系统平台的初始化在 XR806 SDK 的 platform_init()函数中完成，具体请阅读此函数。之后会先注册 monitor 的接收回调处理函数，然后再切换到 monitor 模式进行收包处理。等待 60 秒时间后，程序会关闭 monitor，示例结束。

具体操作见下面的示例代码。

```
void rx_cb(uint8_t *data, uint32_t len, void *info)
{
    if (!info) {
        printf("%s(), info NULL\n", __func__);
        return;
    }

    struct ieee80211_frame *wh;
    char * str_frm_type;
    wh = (struct ieee80211_frame *)data;

    switch (wh->i_fc[0] & IEEE80211_FC0_TYPE_MASK)
    {
    case IEEE80211_FC0_TYPE_MGT:
        switch (wh->i_fc[0] & IEEE80211_FC0_SUBTYPE_MASK)
        {
        case IEEE80211_FC_STYPE_ASSOC_REQ:
            str_frm_type = "association req";
            break;
        case IEEE80211_FC_STYPE_ASSOC_RESP:
            str_frm_type = "association resp";
            break;
        case IEEE80211_FC_STYPE_REASSOC_REQ:
            str_frm_type = "reassociation req";
            break;
        case IEEE80211_FC_STYPE_REASSOC_RESP:
            str_frm_type = "reassociation resp";
            break;
        case IEEE80211_FC_STYPE_PROBE_REQ:
            str_frm_type = "probe req";
            break;
        case IEEE80211_FC_STYPE_PROBE_RESP:
            str_frm_type = "probe resp";
            break;
        case IEEE80211_FC_STYPE_BEACON:
            str_frm_type = "beacon";
            break;
```

```
            case IEEE80211_FC_STYPE_ATIM:
                str_frm_type = "atim";
                break;
            case IEEE80211_FC_STYPE_DISASSOC:
                str_frm_type = "disassociation";
                break;
            case IEEE80211_FC_STYPE_AUTH:
                str_frm_type = "authentication";
                break;
            case IEEE80211_FC_STYPE_DEAUTH:
                str_frm_type = "deauthentication";
                break;
            case IEEE80211_FC_STYPE_ACTION:
                str_frm_type = "action";
                break;
            default:
                str_frm_type = "unknown mgmt";
                break;
            }
            break;
        case IEEE80211_FC0_TYPE_CTL:
            str_frm_type = "control";
            break;
        case IEEE80211_FC0_TYPE_DATA:
            str_frm_type = "data";
            break;
        default:
            str_frm_type = "unknown";
            break;
        }
        printf("recv pack type:%s\n", str_frm_type);
}

void monitor_test(void)
{
    /* register rx callback first */
    wlan_monitor_set_rx_cb(g_wlan_netif, rx_cb);

    /* switch to monitor mode */
    net_switch_mode(WLAN_MODE_MONITOR);

    OS_Sleep(60);

    /* unregister rx callback */
```

```
    wlan_monitor_set_rx_cb(g_wlan_netif, NULL);

    /* After one minute, switch back to sta mode */
    net_switch_mode(WLAN_MODE_STA);
}
```

### 5.3.3 效果展示

在评估板中运行 WLAN 示例程序后，在控制台中会打印如下所示。

```
recv pack type:data
recv pack type:control
recv pack type:data
recv pack type:control
recv pack type:control
recv pack type:data
recv pack type:data
recv pack type:data
recv pack type:data
recv pack type:control
recv pack type:data
recv pack type:data
recv pack type:data
recv pack type:data
recv pack type:control
recv pack type:data
...
```

**著作权声明**



**商标声明**



**免责声明**