SCM1612
Wi-Fi 6 和 BLE 5 低功耗 SoC

Wi-Fi 软件开发指南

文档版本 1.5
发布日期 2024-07-22

联系方式
速通半导体科技有限公司 (www.senscomm.com)
江苏省苏州市工业园区苏州大道西 2 号国际大厦 303 室
销售或技术支持，请发送电子邮件至
support@senscomm.com

免责声明和注意事项

本文档仅按"现状"提供。速通半导体有限公司保留在无需另行通知的情况下对其或本文档中包含的任何规格进行更正、改进和其他变更的权利。

与使用本文档中的信息有关的一切责任，包括侵犯任何专有权利的责任，均不予承认。此处不授予任何明示或暗示、通过禁止或其他方式对任何知识产权的许可。

本文档中的所有第三方信息均按"现状"提供，不对其真实性和准确性提供任何保证。

本文档中提及的所有商标、商号和注册商标均为其各自所有者的财产，特此确认。

版本历史

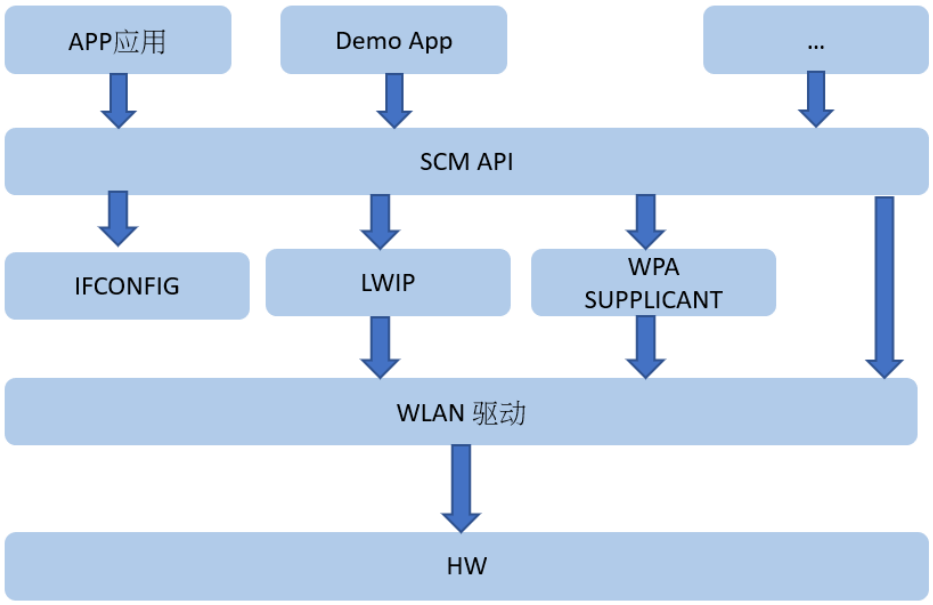
版本	日期	描述
1.5	2024-07-22	新增关于监控模式下信道设置的 6.3.2 小节
1.4	2024-06-25	添加混杂模式
1.3	2024-04-22	更新 STA APIs
1.2	2023-08-22	更新 STA APIs
1.1	2023-08-15	格式修改
1.0	2023-08-04	1.0 发布
0.1	2023-07-13	初稿

目 录

1 概述

SCM1612为应用层提供了丰富的SCM API接口，使其能够操作和控制WLAN驱动。通过 SCM API，开发者可以实现与 Wi-Fi 相关的功能，例如创建 STA/SoftAP、网络扫描、网络配置、关联、解除关联以及状态查询，如图 1-1 所示。

图 1-1 SCM1612 API 接口流程图



以下是各功能模块的说明：

- APP 应用: 用户可基于 SCM API 进行二次开发。
- Demo APP: SDK 提供的功能开发示例。
- SCM API: 基于 SDK 提供的通用接口。
- IFCONFIG: 用于配置、控制和查询网络接口。
- LWIP: 网络协议栈。
- WPA SUPPLICANT(含 HOSTAPD): Wi-Fi 管理模块。
- WLAN 驱动: 实现 802.11 网络协议的模块。
- HW: WLAN 的硬件层实现。

2 Wi-Fi 加载

2.1 概述

当芯片上电后，WLAN 驱动将自动完成加载，并进行寄存器的初始配置、参数校准以及软件资源的申请和配置。

2.2 开发流程

步骤 1：芯片上电后，WLAN 自动加载成功。

步骤 2：请参考“3 Wi-Fi STA 功能”或“4 SoftAP 功能”。

--- 结束

2.3 注意事项

WLAN 驱动默认会自动加载 wlan0 和 wlan1 所需的软件资源。

3 Wi-Fi STA 功能

3.1 概述

STA 提供以下功能

- WPA_SUPPLICANT 中为 STA 配置所需资源
- 基础和进阶的网络扫描
- 网络关联
- DHCP 功能
- 查询关联 AP 的状态
- 去关联

3.2 开发流程

3.2.1 使用场景

当需要连接到某个网络并与其通信时，应启动 STA 功能。

3.2.2 STA API 功能

STA 功能提供的 API 接口如表 3-1 所示。

表 3-1 驱动 STA 功能接口描述

API 名称	描述
scm_wifi_sta_start	启动 STA。
scm_wifi_register_event_callback	注册 STA 接口的事件回调函数。
scm_wifi_unregister_event	取消注册接口的事件回调函数。
scm_wifi_event_send	传送注册 event 到 host(非必要函数) 。
scm_wifi_sta_scan	触发 STA 扫描 AP。

scm_wifi_sta_advance_scan	依据特定参数值型扫描。
scm_wifi_sta_scan_results	获取 STA 扫描结果。
scm_wifi_sta_set_config	配置 STA 模式下的 Wi-Fi 连接信息。
scm_cli_sta_reconnect_policy	设置 STA 模式下的自动重连配置。
scm_wifi_sta_connect	执行 STA 连线。
scm_wifi_sta_get_connect_info	获取已连接 STA 的网络状态。
scm_wifi_sta_get_ap_rssi	获取路由器的接收信号强度指示（RSSI）值。如果未连接，则返回 0xFF。
netifapi_dhcp_start	启动 DHCP Client，并且获取 IP 地址。
netifapi_dhcp_stop	停止 DHCP Client。
scm_wifi_sta_disconnect	执行 STA 断线。
scm_wifi_sta_fast_connect	执行 STA 快速连接，用于 WPA/WPA2 加密的路由器。
scm_cli_sta_get_psk	获取预共享密钥（PSK）信息以快速连接。
scm_wifi_sta_stop	关闭 STA。
scm_wifi_sta_set_ps	设置 STA 模式下的省电模式（Power Save）配置。
scm_wifi_sta_set_country_code	设置期望的国家代码。
scm_wifi_sta_get_country_code	获取当前设置的国家代码。
scm_wifi_sta_set_keepalive	启用/禁用保持活动功能，根据间隔发送空帧。
scm_wifi_wc_set_keepalive	在 STA 进入低功耗模式时，启用/禁用保持活动功能。
wise_wifi_set_wc_bcn_loss_chk	在 STA 进入低功耗模式时，启用/禁用信标丢失的最后检查。
wise_wifi_set_wc_port_filter	在 STA 进入低功耗模式时，启用/禁用 TCP/UDP 端口号过滤。

3.2.3 实现流程

实现步骤如下：

步骤 1: 调用`scm_wifi_register_event_callback` 注册 STA 事件回调函数。
步骤 2: 调用`scm_wifi_sta_start` 启动 STA。

步骤 3: 执行 STA 扫描，可以选择调用`scm_wifi_sta_scan` 或 `scm_wifi_sta_advance_scan`。
步骤 4: 通过`scm_wifi_sta_scan_results`获取扫描结果。
步骤 5: [可选]调用`scm_cli_sta_reconnect_policy`设定自动重连策略。
步骤 6: 依据第 4 步的扫描结果，选择适当的网络并使用`scm_wifi_sta_set_config `配置连线设置。
步骤 7: 调用`scm_wifi_sta_connect`进行网络连接。
步骤 8: 在收到`SYSTEM_EVENT_STA_CONNECTED`事件后，可以调用 `scm_wifi_sta_get_connect_info`查询网络状态。
步骤 9: 调用`netifapi_dhcp_start`获取 IP 地址。
步骤 10: 调用`scm_wifi_sta_disconnect`断开连接。
步骤 11: 调用 scm_wifi_sta_stop 关闭 STA。
---结束

API 函数返回值如表 3-2 所示。

表 3-2 STA 函数返回值说明

定义	数值	描述
WISE_OK	0	**执行成功**
WISE_FAIL	-1	**执行失败**

3.3 注意事项

3.3.1 连线相关事项

- 事件回调：执行 scm_wifi_register_event_callback 函数是必要的，以便清晰地了解 STA 发生的事件并执行相应的动作。
- 带宽支持：
  - 在 Wi-Fi 4 模式下，本产品支持 BW40 和 BW20。
  - 在 Wi-Fi 6 模式下，本产品仅支持 BW20 。
- 连线接口：连线采用非阻塞式接口。可以通过接收 SYSTEM_EVENT_STA_CONNECTED 事件来确认是否成功连接。
- 直接连接：如果已知待连接网络的参数，可以直接发起连线，无需进行扫描过程。
- 认证模式：
  - scm_wifi_sta_fast_connect 接口的 auth 参数仅支持以下认证模式：
    - SCM_WIFI_SECURITY_WPAPSK

- SCM_WIFI_SECURITY_WPA2PSK

■ 基于联盟规范和安全性的考虑，以下认证模式不受支持：
- WEP
- WPA2PSK + TKIP

3.3.2 扫描相关事项

- 扫描采用非阻塞式接口，当扫描命令下发成功后，建议延迟 1 秒后再获取扫描结果，或等待 SYSTEM_EVENT_SCAN_DONE 事件来确认扫描已结束。
- 可通过指定 SSID、BSSID、Channel 等参数进行特定的扫描（参考 scm_wifi_sta_advance_scan）。

3.4 编程实例

示例: 实现 STA 功能的启动、关联、获取网络信息和 IP 地址。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <hal/unaligned.h>
#include <hal/kernel.h>
#include <hal/wlan.h>
#include <hal/kmem.h>
#include "kernel.h"
#include "compat_if.h"
#include "if_media.h"
#include "sncmu_dll.h"
#include "fwil_types.h"
#include "fweh.h"
#include "scdc.h"
#include "task.h"
#include "FreeRTOS.h"
#include <net80211/ieee80211_var.h>
#include <wise_err.h>
#include <wise_log.h>
#include <wise_wifi.h>
#include <wise_event_loop.h>
#include <common.h>
#include <scm_wifi.h>

#define SCM_NEED_DHCP_START(event) ((event)->event_info.connected.not_en_dhcp == false)

/* 待连接的网络资讯，可编译其间选定或是执行时间修改 */
scm_wifi_assoc_request g_assoc_req = {
        .ssid = "Xiaomi_7AB6", /* 网络名称 */
        .auth = SCM_WIFI_SECURITY_WPA2PSK, /* 认证模式 */
        .key = "12345678",  /* 认证密码 */
};

int scm_wifi_register_event_callback(system_event_cb_t event_cb, void *priv)
{
        wise_event_loop_init(event_cb, priv);

        return WISE_OK;
}

/* 此函数可以接收 Wi-Fi 相关必要事件 */
wise_err_t event_handler(void *ctx, system_event_t * event)
{

        switch (event->event_id) {
        case SYSTEM_EVENT_STA_START:
                break;
        case SYSTEM_EVENT_STA_STOP:
                break;
        case SYSTEM_EVENT_STA_GOT_IP:
                printf("\r\n WIFI GOT IP indicate\r\n");
                break;
        case SYSTEM_EVENT_AP_START:
                break;
        case SYSTEM_EVENT_AP_STOP:
                break;
```

```
        case SYSTEM_EVENT_AP_STADISCONNECTED:
                break;
        case SYSTEM_EVENT_STA_CONNECTED:
                printf("\r\nWIFI CONNECTED indicate\r\n");
                /* 成功连线，启动 DHCP client */
                if (SCM_NEED_DHCP_START(event)) {
                        scm_wifi_status connect_status;
                        netifapi_dhcp_start(scm_wifi_get_netif(WISE_IF_WIFI_STA));
                        scm_wifi_sta_get_connect_info(&connect_status);
                        scm_wifi_sta_dump_ap_info(&connect_status);
                }
                break;
        case SYSTEM_EVENT_STA_DISCONNECTED:
                printf("\r\nWIFI DISCONNECT\r\n");
                break;
        case SYSTEM_EVENT_SCAN_DONE:
                printf("WiFi: Scan results available\n");

                break;
        case SYSTEM_EVENT_SCM_CHANNEL:
                printf("WiFi: Scm channel send msg\n");
                scm_wifi_event_send(event, sizeof(system_event_t));
                break;
        default:
                break;
        }
        return WISE_OK;
}

int scm_wifi_start_connect(void)
{
        scm_wifi_assoc_request *assoc_req = &g_assoc_req;

        /* 配置连线资讯 */
        if (scm_wifi_sta_set_config(assoc_req, NULL))
                return WISE_FAIL;

        return scm_wifi_sta_connect();
}
```

```
int main(void)
{
        int ret = WISE_OK;
        char ifname[WIFI_IFNAME_MAX_SIZE + 1] = {0};
        int len = sizeof(ifname);

        printf("Sta Hello world!\n");

        /* 注册事件回调函数 */
        scm_wifi_register_event_callback(event_handler, NULL);

        /* 启动 STA 功能 */
        scm_wifi_sta_start(ifname, &len);

        /* 启动 STA 连线 */
        ret = scm_wifi_start_connect();

        /* ret 为 0 表示执行成功 */
        return ret;
}
```

4 Wi-Fi SoftAP 功能

4.1 概述

SoftAP 功能主要包括

- 在 WPA_SUPPLICANT 中为 SoftAP 配置所需资源
- 网络配置
- DHCP 服务器功能
- 查询关联的 STA 状态
- 断开指定的 STA

4.2 开发流程

4.2.1 使用场景

当需要创建一网络接入点，供其他设备接入并共享网络资源时，应启动 SoftAP 功能。

4.2.2 SoftAP API 功能

SoftAP 功能提供的 API 接口如表 4-1 所示。

表 4-1 驱动 SoftAP 功能接口描述

API 名称	描述
scm_wifi_sap_start	启动 SoftAP。需先调用 scm_wifi_sap_set_config 配网。
scm_wifi_sap_stop	关闭 SoftAP。
scm_wifi_register_event_callback	注册接口的事件回调函数。
scm_wifi_unregister_event	取消注册接口的事件回调函数。
scm_wifi_sap_set_config	配置 SoftAP Wi-Fi 连线所需信息。

scm_wifi_sap_get_config	获取 SoftAP 当前配置
scm_wifi_sap_set_beacon_interval	设置 SoftAP 的 beacon 间距
scm_wifi_sap_set_dtim_period	设置 SoftAP 的 dtim 周期
scm_wifi_sap_get_connected_sta	获取当前接入的 STA 信息
scm_wifi_sap_deauth_sta	断开指定 STA 的连信息
scm_wifi_set_ip	设置 SoftAP 的 IP 地址、子网掩码和网关参数。
scm_wifi_reset_ip	清除 SoftAP 的 IP 地址、子网掩码和网关参数。
netifapi_dhcps_start	启动 DHCP Server。
netifapi_dhcps_stop	停止 DHCP Server。

4.2.3 实现流程

实现步骤如下：

步骤 1: 调用`scm_wifi_register_event_callback`注册 SoftAP 事件回传函数。
步骤 2: 调用`scm_wifi_sap_set_config`配置 SoftAP 的网络参数。

- 调用`scm_wifi_sap_set_beacon_interval`设置 beacon 间距。
- 调用`scm_wifi_sap_set_dtim_period`设置 dtim 周期。
- `scm_wifi_sap_set_config`将自行调用`scm_wifi_sap_stop`, `scm_wifi_sap_star`t 以启用 SoftAP。

步骤 3: 调用`scm_wifi_sap_stop`,关闭之前的 SoftAP。
步骤 4: 调用`scm_wifi_sap_start`,重新启动 SoftAP。
步骤 5: 调用`scm_wifi_set_ip`配置网络 IP。
步骤 6: 调用`netifapi_dhcps_start`启动 DHCP 服务器。
步骤 7: 调用`netifapi_dhcps_stop`停止 DHCP 服务器。
步骤 8: 调用`scm_wifi_sap_stop`关闭 SoftAP。
---结束

API 函数返回值如表 4-2 所示。

表 4-2 SoftAP 函数返回值说明

定义	数值	描述
WISE_OK	0	执行成功
WISE_FAIL	-1	执行失败

4.3 注意事项

- 为了清晰地了解 SoftAP 发生的事件并执行相应的动作，执行 scm_wifi_register_event_callback 函数是必要的。
- SoftAP 的网络参数可以预先设置为默认值。
- 关闭 SoftAP 时，其网络参数不会被重置，但重启设备后会恢复到初始默认值。
- SoftAP 只支持 OPEN 和 WPA2 模式。
- SoftAP 模式下，最大关联用户数不超过 1 个。

4.4 编程实例

示例: 实现 SoftAP 功能启动、获取网络信息、设置 IP。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <hal/unaligned.h>
#include <hal/kernel.h>
#include <hal/wlan.h>
#include <hal/kmem.h>
#include "kernel.h"
#include "compat_if.h"
#include "if_media.h"
#include "sncmu_dll.h"
#include "fwil_types.h"
#include "fweh.h"
#include "scdc.h"
#include "task.h"
#include "FreeRTOS.h"
#include <net80211/ieee80211_var.h>
#include <wise_err.h>
#include <wise_log.h>
#include <wise_wifi.h>
#include <wise_event_loop.h>
#include <common.h>
#include "dhcps.h"
#include <scm_wifi.h>

/* 设置的网络资讯，可编译其间选定或是执行时间修改 */
scm_wifi_softap_config g_sap_cfg = {
        .ssid = "sap_test",
        .key = "12345678",
        .channel_num = 6,
        .authmode = SCM_WIFI_SECURITY_WPA2PSK,
        .pairwise = SCM_WIFI_PAIRWISE_AES,
};

wise_err_t event_handler(void *ctx, system_event_t * event)
{
        switch (event->event_id) {
        case SYSTEM_EVENT_AP_START:
                printf("\r\nSYSTEM_EVENT_AP_START\r\n");
                /* 设置网络 IP */
                scm_wifi_set_ip("wlan1", "192.168.200.1", "255.255.255.0", NULL);
                /* 启动 DHCP Server */
                netifapi_dhcps_start(scm_wifi_get_netif(WISE_IF_WIFI_AP));
                break;
        case SYSTEM_EVENT_AP_STOP:
                printf("\r\nSYSTEM_EVENT_AP_STOP\r\n");
                break;
        case SYSTEM_EVENT_AP_STACONNECTED:
                printf("\r\nSYSTEM_EVENT_AP_STACONNECTED\r\n");
                printf("Connected STA:" MACSTR "\r\n",
                        MAC2STR(event->event_info.sta_connected.mac));
                break;
```

```
        case SYSTEM_EVENT_AP_STADISCONNECTED:
                printf("\r\nSYSTEM_EVENT_AP_STADISCONNECTED\r\n");
                printf("Disconnected STA:" MACSTR "\r\n",
                        MAC2STR(event->event_info.sta_disconnected.mac));
                break;

        default:
                break;
        }
        return WISE_OK;
}

int main(void)
{
        int ret = WISE_OK;
        char ifname[WIFI_IFNAME_MAX_SIZE + 1] = {0};
        int len = sizeof(ifname);

        scm_wifi_softap_config *sap = &g_sap_cfg;

        printf("SoftAP Hello world!\n");

        /* 注册事件回调函数 */
        scm_wifi_register_event_callback(event_handler, NULL);

        /* 设置 SoftAP */
        scm_wifi_sap_set_config(sap);

        /* 启动 SoftAP */
        ret = scm_wifi_sap_start(ifname, &len);

        /* ret 为 0 表示执行成功 */
        return ret;
}
```

5 Wi-Fi STA/SoftAP 共存

5.1 概述

STA 与 SoftAP 共存意味着 STA 功能和 SoftAP 功能可以同时运行。根据 STA 和 SoftAP 的启动顺序，可以分为以下几种场景：

场景	描述
同频同带共存	全时共存
同频异带共存	全时共存
异频同带共存	平均分时共存
异频异带共存	平均分时共存

全时共存: STA 和 SoftAP 可以同时工作。
分时共存: STA 和 SoftAP 会在各自的时间段内工作。

5.2 开发流程

5.2.1 使用场景

在网络配置时，产品首先启动 SoftAP。当手机连接到 SoftAP 后，通过手机 APP 发送家庭网络信息（如 SSID 和密码）给产品。当产品接收到这些信息后，它会启动 STA 并连接到家庭网络。一旦产品成功连接，它会关闭 SoftAP，并仅保持 STA 与家庭网络的连接。其他共存场景可以根据产品的具体需求来决定。

5.2.2 实现流程

步骤 1: 创建 SoftAP 网络接口（详见 “4 Wi-Fi SoftAP 功能”）。
步骤 2: 手机连接到 SoftAP 网络，并通过手机 APP 发送家庭网络信息。
步骤 3: [非必要]为了避免分时共存，建议重新启动 SoftAP 至家居网络的信道（详见 “4 Wi-Fi SoftAP 功能”）。

步骤 4: 创建 STA，并根据家居网络信息(SSID 和密码)，完成关联(详见“3 Wi-Fi STA 功能”)。
步骤 5: 关闭 SoftAP(详见“4 Wi-Fi SoftAP 功能”)。
--- 结束

返回值
请参考对应模块功能的返回值说明

5.3 注意事项

- 在分时共存模式下，由于 STA 和 SoftAP 需要轮流使用，性能可能会受到影响。为了获得更好的性能，建议将 SoftAP 设置在 STA 的工作信道上。

5.4 编程实例

请参考”3 Wi-Fi STA 功能”及”4 Wi-Fi SoftAP 功能”编程实例。

6 混杂模式

6.1 概述

混杂模式允许 Wi-Fi 硬件捕捉指定频道上的所有 Wi-Fi 帧，并可以发送原始的 802.11 帧。此模式对于高级网络监控和问题诊断是极为关键的。

6.2 应用场景

6.2.1 应用场景

通过使用通用套接字接口来捕获由硬件接收的 Wi-Fi 原始帧，相比于使用回调注册，此方法在数据路径应用中提供了更高的一致性和可靠性。

6.2.2 混杂模式 API 功能

以下是混杂模式支持的 API 功能列表：

表 6-1：混杂模式功能接口描述

API 名称	描述
scm_wifi_set_promiscuous	激活混杂模式，用于监控和捕获流量。
scm_wifi_get_promiscuous	查询当前是否启用了混杂模式。
scm_wifi_80211_tx	在指定频道上发送原始的 IEEE 802.11 数据帧。
scm_wifi_set_channel	配置设备的主/副频道。
scm_wifi_get_channel	获取当前设备设置的主/副频道信息。

6.2.3 实施步骤

按以下步骤实现混杂模式：

1. 通过 scm_wifi_set_promiscuous 启用混杂模式。
2. 使用 scm_wifi_set_channel 配置频道。
3. 通过 scm_wifi_80211_tx 发送原始 802.11 数据帧。
4. 可选：利用 RAW 套接字从频道捕获 802.11 数据帧，进行进一步分析。

表 6-2：API 功能返回值说明

定义	值	描述
WISE_OK	0	执行成功。
WISE_FAIL	-1	执行失败，需检查配置及硬件状态。

6.3 注意事项

6.3.1 运行模式注意

在启用混杂模式前，确保停用 STA 和 SoftAP 模式，可以通过执行 `scm_wifi_sta_sto`p 和`scm_wifi_sap_sto`p 命令来实现。仅在混杂模式激活时，`scm_wifi_80211_tx`和`scm_wifi_set_channel`等 API 才会正常工作。

6.3.2 监控模式下的信道设置

在每次启用混杂模式时，必须设置 RF 信道。例如，以下操作顺序是无效的：

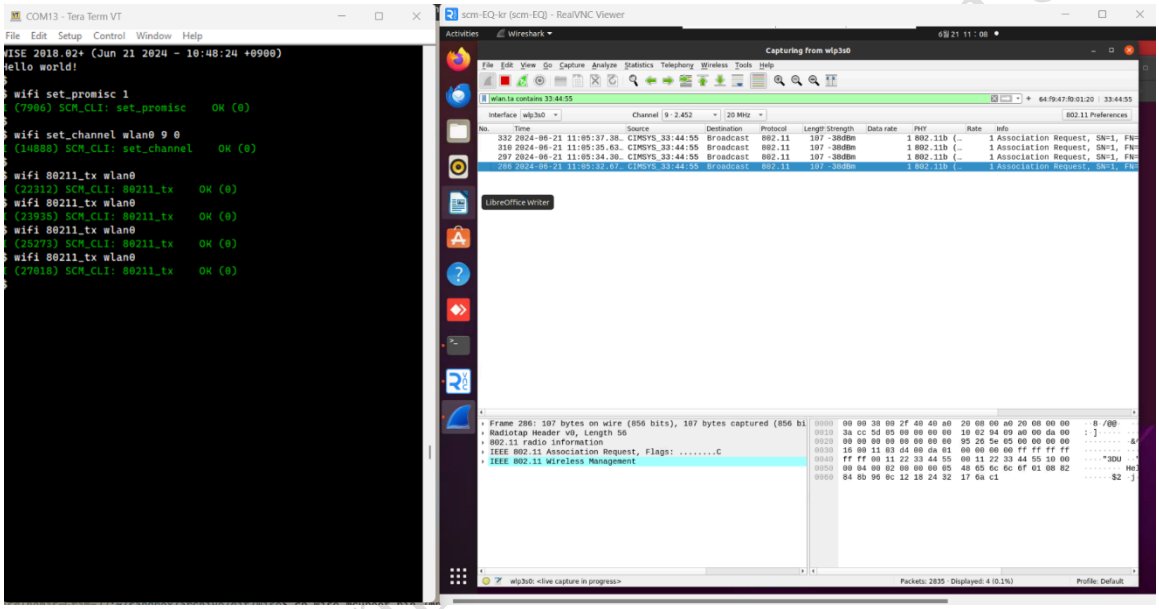
1. 在`wlan0`上启用混杂模式。
2. 将`wlan0`的信道设置为 1。
3. 在`wlan1`上启用混杂模式。
4. 将`wlan1`的信道设置为 6。
5. 在 wlan0 上发送一个原始的 802.11 帧，期望它会在信道 1 上发送。

由于 SCM1612s 只有一个 RF 链路，第 4 步中的设置会覆盖之前的信道设置，因此在第 5 步中 RF 信道将会被设置为 6。

6.4 编程示例

示例 1：

向 6 频道发送一个原始 802.11 数据帧，具体实现方法可参考 api/scm_cli_wifi.c 文件。确保已启用 CONFIG_CLI_WIFI_CHANNEL 和 CONFIG_CLI_WIFI_MONITOR 以使用相关 CLI 命令。



示例 2：

使用‘wshark’命令显示接收到的 802.11 原始数据帧。演示此功能前需要启用 CONFIG_CMD_WSHARK。

```
WISE 2018.02+ (Jul 17 2024 - 15:16:59 -0700)
Hello world!
$ wifi set_promisc wlan0 1
I (13371) SCM_CLI: set_promisc    OK (0)
$ wifi set_channel wlan0 9 0
I (20317) SCM_CLI: set_channel    OK (0)
$ wifi 80211_tx wlan0
I (26885) SCM_CLI: 80211_tx    OK (0)
$ wifi 80211_tx wlan0
I (28864) SCM_CLI: 80211_tx    OK (0)
$ wifi 80211_tx wlan0
I (30247) SCM_CLI: 80211_tx    OK (0)
$ wifi 80211_tx wlan0
I (32239) SCM_CLI: 80211_tx    OK (0)
$ wifi 80211_tx wlan0
I (34882) SCM_CLI: 80211_tx    OK (0)
$
$
$ wshark -i wlan0
```

No.	Time	RA	TA	Rate	BW	RSSI	LEN	SEQ	FREQ	Flags	Info
252	1640995507.173565	ff:ff:ff:ff:ff:ff	50:6a:03:c4:17:a6	1.0	20	-76	274	1494.0	2452	.....	BEACON (ssid=)
253	1640995507.183540	ff:ff:ff:ff:ff:ff	98:25:4a:66:2d:7a	1.0	20	-72	301	2590.0	2452	.....	BEACON (ssid=StarwayInsurance)
254	1640995507.196108	ff:ff:ff:ff:ff:ff	e6:bf:fa:04:b9:8a	1.0	20	-70	218	791.0	2452	.....	BEACON (ssid=CoxWiFi)
255	1640995507.207375	ff:ff:ff:ff:ff:ff	a0:36:bc:70:55:a0	1.0	20	-40	438	212.0	2452	.....	BEACON (ssid=senscomm_int2g)
256	1640995507.218668	ff:ff:ff:ff:ff:ff	60:38:e0:12:73:1a	1.0	20	-47	355	2107.0	2452	.....	BEACON (ssid=BBWF)
257	1640995507.230048	ff:ff:ff:ff:ff:ff	88:96:4e:98:2c:40	1.0	20	-50	295	3773.0	2452	.....	BEACON (ssid=ATTSkbI5S2)
258	1640995507.241182	01:80:c2:00:00:00	88:96:4e:98:2c:40	1.0	20	-50	105	3774.0	2452	.p....F..	DATA
259	1640995507.251026	ff:ff:ff:ff:ff:ff	f8:f5:32:b9:ee:00	1.0	20	-75	295	1454.0	2452	.....	BEACON (ssid=ATTu25kXi2)
260	1640995507.262306	ff:ff:ff:ff:ff:ff	6c:4b:b4:f0:57:d4	1.0	20	-58	324	3801.0	2452	.....	BEACON (ssid=ATTeNxGmW7)
261	1640995507.273583	ff:ff:ff:ff:ff:ff	50:eb:f8:19:88:a0	1.0	20	-45	458	2561.0	2452	.....	BEACON (ssid=Xiaohu_ASUS)
262	1640995507.284866	ff:ff:ff:ff:ff:ff	98:25:4a:66:2d:7a	1.0	20	-73	301	2591.0	2452	.....	BEACON (ssid=StarwayInsurance)
263	1640995507.297496	33:33:00:00:00:0c	4c:12:65:9d:c9:40	1.0	20	-59	771	2886.0	2452	.p....F..	DATA
264	1640995507.307313	ff:ff:ff:ff:ff:ff	a0:36:bc:70:55:a0	1.0	20	-40	438	213.0	2452	.....	BEACON (ssid=senscomm_int2g)
265	1640995507.318676	ff:ff:ff:ff:ff:ff	60:38:e0:12:73:1a	1.0	20	-47	355	2108.0	2452	.....	BEACON (ssid=BBWF)
266	1640995507.329853	ff:ff:ff:ff:ff:ff	c4:41:1e:20:58:70	1.0	20	-84	304	1475.0	2452	.....	BEACON (ssid=)
267	1640995507.339848	ff:ff:ff:ff:ff:ff	88:96:4e:98:2c:40	1.0	20	-50	295	3775.0	2452	.....	BEACON (ssid=ATTSkbI5S2)
268	1640995507.351048	ff:ff:ff:ff:ff:ff	f8:f5:32:b9:ee:00	1.0	20	-74	295	1458.0	2452	.....	BEACON (ssid=ATTu25kXi2)
269	1640995507.363647	ff:ff:ff:ff:ff:ff	6c:4b:b4:f0:57:d4	1.0	20	-59	324	3802.0	2452	.....	BEACON (ssid=ATTeNxGmW7)
270	1640995507.374910	ff:ff:ff:ff:ff:ff	6c:4b:b4:f0:57:d4	1.0	20	-59	105	3812.0	2452	.p...MF..	DATA
271	1640995507.384745	ff:ff:ff:ff:ff:ff	50:eb:f8:19:88:a0	1.0	20	-45	458	2562.0	2452	.....	BEACON (ssid=Xiaohu_ASUS)
272	1640995507.396047	ff:ff:ff:ff:ff:ff	6c:4b:b4:f0:57:d4	1.0	20	-58	105	3828.0	2452	.p...MF..	DATA
273	1640995507.405883	ff:ff:ff:ff:ff:ff	6c:4b:b4:f0:57:d4	1.0	20	-58	105	3837.0	2452	.p...MF..	DATA
274	1640995507.415734	ff:ff:ff:ff:ff:ff	6c:4b:b4:f0:57:d4	1.0	20	-58	105	3848.0	2452	.p...MF..	DATA
275	1640995507.424257	ff:ff:ff:ff:ff:ff	a4:11:62:6a:e7:a6	2.0	20	-75	266	1595.0	2452	.....	BEACON (ssid=)
276	1640995507.435482	ff:ff:ff:ff:ff:ff	88:96:4e:98:2c:40	1.0	20	-50	295	3776.0	2452	.....	BEACON (ssid=ATTSkbI5S2)
277	1640995507.446750	ff:ff:ff:ff:ff:ff	f8:f5:32:b9:ee:00	1.0	20	-75	295	1463.0	2452	.....	BEACON (ssid=ATTu25kXi2)
278	1640995507.458032	ff:ff:ff:ff:ff:ff	6c:4b:b4:f0:57:d4	1.0	20	-59	324	3852.0	2452	.....	BEACON (ssid=ATTeNxGmW7)
279	1640995507.469307	ff:ff:ff:ff:ff:ff	50:eb:f8:19:88:a0	1.0	20	-45	458	2563.0	2452	.....	BEACON (ssid=Xiaohu_ASUS)
280	1640995507.480582	ff:ff:ff:ff:ff:ff	98:25:4a:66:2d:7a	1.0	20	-74	301	2593.0	2452	.....	BEACON (ssid=StarwayInsurance)
281	1640995507.493210	ff:ff:ff:ff:ff:ff	a0:36:bc:70:55:a0	1.0	20	-40	438	215.0	2452	.....	BEACON (ssid=senscomm_int2g)
282	1640995507.504515	ff:ff:ff:ff:ff:ff	60:38:e0:12:73:1a	1.0	20	-48	355	2110.0	2452	.....	BEACON (ssid=BBWF)
283	1640995507.515751	8c:d9:d6:d6:57:2c	60:38:e0:12:73:1a	1.0	20	-61	25	0.0	2452	.....	ACK
284	1640995507.525580	ff:ff:ff:ff:ff:ff	88:96:4e:98:2c:40	1.0	20	-49	295	3777.0	2452	.....	BEACON (ssid=ATTSkbI5S2)
285	1640995507.536867	ff:ff:ff:ff:ff:ff	f8:f5:32:b9:ee:00	1.0	20	-74	295	1465.0	2452	.....	BEACON (ssid=ATTu25kXi2)
286	1640995507.548138	ff:ff:ff:ff:ff:ff	6c:4b:b4:f0:57:d4	1.0	20	-60	324	3853.0	2452	.....	BEACON (ssid=ATTeNxGmW7)
287	1640995507.559400	ff:ff:ff:ff:ff:ff	4c:12:65:9d:c9:40	1.0	20	-59	303	2890.0	2452	.....	BEACON (ssid=CA Mission Hospice)
288	1640995507.572052	ff:ff:ff:ff:ff:ff	4c:12:65:9d:c9:40	1.0	20	-59	145	2891.0	2452	.p....F..	DATA
289	1640995507.581893	33:33:00:00:00:fb	4c:12:65:9d:c9:40	1.0	20	-58	154	2896.0	2452	.p....F..	DATA
290	1640995507.591739	ff:ff:ff:ff:ff:ff	50:6a:03:c4:17:a6	1.0	20	-77	274	1501.0	2452	.....	BEACON (ssid=NETGEAR22)
291	1640995507.603023	26:00:24:68:34:83	50:eb:f8:19:88:a0	1.0	20	-44	557	2223.0	2452	.....	PROBE_RESP
292	1640995507.612885	26:00:24:68:34:83	60:38:e0:12:73:1a	1.0	20	-49	470	2111.0	2452	.....	PROBE_RESP
293	1640995507.622756	ff:ff:ff:ff:ff:ff	60:38:e0:12:73:1a	1.0	20	-48	355	2112.0	2452	.....	BEACON (ssid=BBWF)
294	1640995507.634025	26:00:24:68:34:83	82:e8:2c:a4:25:c4	1.0	20	-77	485	2187.0	2452	....R....	PROBE_RESP
295	1640995507.643876	26:00:24:68:34:83	82:e8:2c:a4:25:c4	1.0	20	-77	485	2188.0	2452	....R....	PROBE_RESP
296	1640995507.653766	ff:ff:ff:ff:ff:ff	f8:f5:32:b9:ee:00	1.0	20	-74	295	1467.0	2452	.....	BEACON (ssid=ATTu25kXi2)
297	1640995507.666370	ff:ff:ff:ff:ff:ff	6c:4b:b4:f0:57:d4	1.0	20	-59	324	3854.0	2452	.....	BEACON (ssid=ATTeNxGmW7)
298	1640995507.677635	ff:									

7 Wi-Fi 常见问题 FAQ