SCM1612
Wi-Fi 6 and BLE 5 Low-Power SoC

Wi-Fi Software Development Guide

Revision 1.5
Date 2024-07-22

Senscomm Confidential

Contact Information
Senscomm Semiconductor (www.senscomm.com)
Room 303, International Building, West 2 Suzhou Avenue,
SIP, Suzhou, China
For sales or technical support, please send email to
info@senscomm.com

Disclaimer and Notice

This document is provided on an “as-is” basis only. Senscomm reserves the right to make corrections, improvements and other changes to it or any specification contained herein without further notice.

All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

All third party’s information in this document is provided as is with NO warranties to its authenticity and accuracy.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners and are hereby acknowledged.

© 2024 Senscomm Semiconductor Co.,Ltd. All Rights Reserved.

Senscomm Confidential

Version History

Version	Date	Description
1.5	2024-07-22	Added new subsection 6.3.2 on channel settings in monitor mode
1.4	2024-06-25	Add Promiscuous Mode
1.3	2024-04-22	Update STA APIs
1.2	2023-08-22	Update STA APIs
1.1	2023-08-15	Format Modification
1.0	2023-08-04	Version 1.0 Release
0.1	2023-07-13	Initial Draft

Senscomm Confidential

Table of Contents

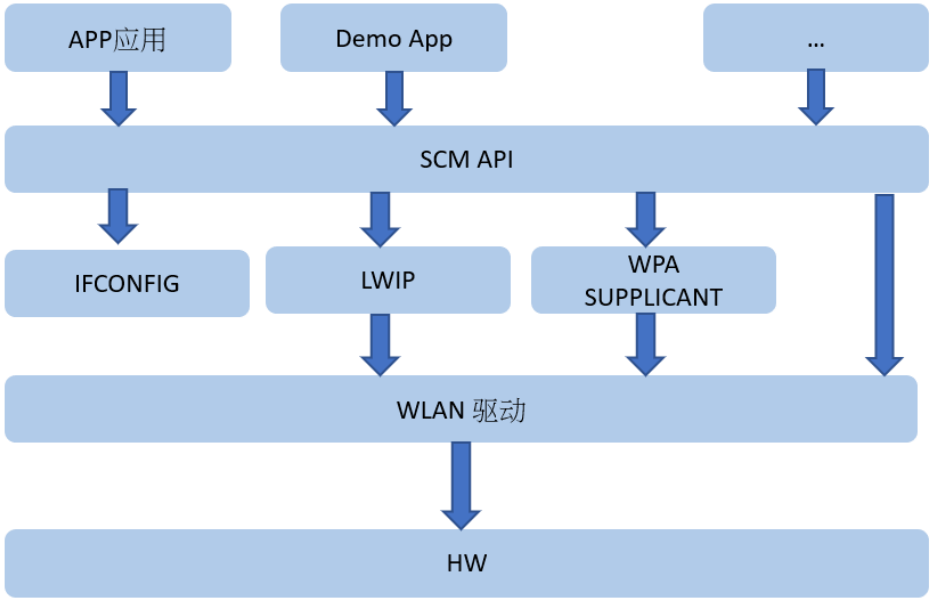
Senscomm Confidential

Senscomm Confidential

1 Overview

The SCM1612 offers a comprehensive set of SCM API interfaces for the application layer, enabling it to operate and control the WLAN driver. Through the SCM API, developers can implement Wi-Fi-related functionalities such as creating STA/SoftAP, network scanning, network configuration, association, disassociation, and status querying, as illustrated in Figure 1-1.

Figure 1-1 SCM1612 API Interface Flowchart



The descriptions for each functional module are as follows:

- APP Application: Users can perform secondary development based on the SCM API.
- Demo APP: A functional development example provided by the SDK.
- SCM API: A general interface based on the SDK.
- IFCONFIG: Used for network interface configuration, control, and querying.
- LWIP: Network protocol stack.
- WPA SUPPLICANT (including HOSTAPD): Wi-Fi management module.
- WLAN Driver: Module implementing the 802.11 network protocol.
- HW: Implementation of the WLAN hardware layer.

2 Wi-Fi Initialization

2.1 Overview

Upon powering up the chip, the WLAN driver will automatically load, initializing registers, calibrating parameters, and configuring software resources.

2.2 Development Process

Step 1: Once the chip is powered on, the WLAN will automatically complete its loading.

Step 2: Refer to "3 Wi-Fi STA Functionality" or "4 SoftAP Functionality".

--- End

2.3 Precautions

By default, the WLAN driver will automatically load the software resources required for wlan0 and wlan1.

Senscomm Confidential

3 STA Function

3.1 Overview

The STA offers the following capabilities:

- Resource configuration for STA within WPA_SUPPLICANT
- Basic and advanced network scanning
- Network association
- DHCP
- Querying the status of the associated AP
- Disassociation

3.2 Development Process

3.2.1 Application Scenario

To connect to a network and communicate with it, the STA (Station) mode must be activated.

3.2.2 STA API Functions

The API interfaces provided by the STA function are shown in Table 3-1.

Table 3-1 Description of STA Function Interface

API Name	Description
scm_wifi_sta_start	Start STA.
scm_wifi_register_event_callback	Register STA interface event callback.
scm_wifi_unregister_event	Unregister event callback.
scm_wifi_event_send	Send registered event to host (optional).
scm_wifi_sta_scan	Trigger STA to scan for APs.
scm_wifi_sta_advance_scan	Scan based on specific parameters.

Senscomm Confidential

scm_wifi_sta_scan_results	Retrieve STA scan results.
scm_wifi_sta_set_config	Configure STA Wi-Fi connection information.
scm_cli_sta_reconnect_policy	Set STA auto-reconnect configuration.
scm_wifi_sta_connect	Execute STA connection.
scm_wifi_sta_get_connect_info	Get the network status of the connected STA.
scm_wifi_sta_get_ap_rssi	Retrieve router's RSSI (returns 0xFF if not connected).
netifapi_dhcp_start	Start DHCP Client and obtain IP address.
netifapi_dhcp_stop	Stop DHCP Client.
scm_wifi_sta_disconnect	Disconnect STA.
scm_wifi_sta_fast_connect	Execute STA quick connect for WPA/WPA2 encrypted routers.
scm_cli_sta_get_psk	Retrieve PSK information for quick connect.
scm_wifi_sta_stop	Turn off STA.
scm_wifi_sta_set_ps	Turn on/off powersave mode
scm_wifi_sta_set_country_code	Set expect country code
scm_wifi_sta_get_country_code	Get current country code
scm_wifi_sta_set_keepalive	Enable/disable the keepalive function, which sends NULL frames based on the interval.
scm_wifi_wc_set_keepalive	Enable/disable the keepalive function when STA sleeping in low power mode.
wise_wifi_set_wc_bcn_loss_chk	Enable/disable beacon loss last check when STA sleeping in low power mode.
wise_wifi_set_wc_port_filter	Enable/disable TCP/UDP port number filtering when the STA is sleeping in low power mode.

Senscomm Confidential

3.2.3 Implementation Process

Implementation Steps:

Step 1: Call `scm_wifi_register_event_callback` to register the STA event callback function.

Step 2: Call `scm_wifi_sta_start` to start the STA.
Step 3: Call `scm_wifi_sta_scan` (or `scm_wifi_sta_advance_scan`) to perform the STA scan.
Step 4: Call `scm_wifi_sta_scan_results` to retrieve the scan results.
Step 5: [Optional] Call `scm_cli_sta_reconnect_policy` to set the auto-reconnect mechanism.
Step 6: Based on the scan results from Step 4, select the appropriate network and call `scm_wifi_sta_set_config` to configure the connection settings.
Step 7: Call `scm_wifi_sta_connect` to initiate the connection.
Step 8: Upon receiving `SYSTEM_EVENT_STA_CONNECTED`, indicating a successful connection, you can call `scm_wifi_sta_get_connect_info` to check the network status.
Step 9: Call `netifapi_dhcp_start` to obtain an IP address.
Step 10: Call `scm_wifi_sta_disconnect` to disconnect.
Step 11: Call `scm_wifi_sta_stop` to shut down the STA.
--- End

The return values of the API functions are shown in Table 3-2.

Table 3-2 Explanation of STA Function Return Values

Definition	Value	Description
WISE_OK	0	Execution successful
WISE_FAIL	-1	Execution failed

3.3 Precautions

3.3.1 Connection Considerations

- Event Callback: It's essential to execute the scm_wifi_register_event_callback function to clearly understand the events occurring in STA and to take appropriate actions.
- Bandwidth Support:
  - In Wi-Fi 4 mode, this product supports BW40 and BW20.
  - In Wi-Fi 6 mode, this product only supports BW20.
- Connection Interface: The connection uses a non-blocking interface. The success of the connection can be confirmed by receiving the SYSTEM_EVENT_STA_CONNECTED event.
- Direct Connection: If the parameters of the network to be connected are known, a connection can be initiated directly without the scanning process.
- Authentication Modes:

Senscomm Confidential

- The auth parameter of the scm_wifi_sta_fast_connect interface only supports the following authentication modes:
  - SCM_WIFI_SECURITY_WPAPSK
  - SCM_WIFI_SECURITY_WPA2PSK
- Based on alliance specifications and security considerations, the following authentication modes are not supported:
  - WEP
  - WPA2PSK + TKIP

3.3.2 Scanning-related Matters

- Scanning is a non-blocking interface. After successfully issuing the scan command, there is a need to delay for a while before retrieving the scan results. For a full-channel scan, a delay of 1s is recommended. Alternatively, you can wait for `SYSTEM_EVENT_SCAN_DONE` to know when the scan is complete.
- Scans can be conducted by specifying parameters such as SSID, BSSID, Channel, etc. (refer to `scm_wifi_sta_advance_scan`).

3.4 Programming Examples

Example: Implementing STA functionality for startup, association, obtaining network information, and acquiring an IP.

Senscomm Confidential

```
#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 SoftAP Function

4.1 Overview

The SoftAP provides the following features:

- Resource configuration required for SoftAP in WPA_SUPPLICANT.
- Network configuration.
- DHCP Server
- Querying the status of associated STAs.
- Disconnecting a specified STA.

4.2 Development Process

4.2.1 Use Cases

When there's a need to create a network access point for other devices to connect and share data within the network, the SoftAP function should be activated.

4.2.2 SoftAP API Functions

The API interfaces provided by the SoftAP function are shown in Table 4-1.

Table 4-1 Description of SoftAP Function Interface

API Name	Description
scm_wifi_sap_start	Start SoftAP. You must first call `scm_wifi_sap_set_config` for network configuration.
scm_wifi_sap_stop	Stop SoftAP.
scm_wifi_register_event_callback	Register the event callback function for the interface.
scm_wifi_unregister_event	Unregister the event callback function for the interface.

Senscomm Confidential

scm_wifi_sap_set_config	Configure the information required for SoftAP Wi-Fi connection.
scm_wifi_sap_get_config	Retrieve the current configuration of SoftAP.
scm_wifi_sap_set_beacon_interval	Set the beacon interval for SoftAP.
scm_wifi_sap_set_dtim_period	Set the DTIM (Delivery Traffic Indication Message) period for SoftAP.
scm_wifi_sap_get_connected_sta	Get information about the currently connected STAs.
scm_wifi_sap_deauth_sta	Disconnect the specified STA.
scm_wifi_set_ip	Set the IP address, subnet mask, and gateway parameters for SoftAP.
scm_wifi_reset_ip	Clear the IP address, subnet mask, and gateway parameters for SoftAP.
netifapi_dhcps_start	Start the DHCP Server.
netifapi_dhcps_stop	Stop the DHCP Server.

Senscomm Confidential

4.2.3 Implementation Process

Implementation Steps:

Step 1: Call `scm_wifi_register_event_callback` to register the SoftAP event callback function.
Step 2: Call `scm_wifi_sap_set_config` to configure the network parameters for SoftAP:

- Call `scm_wifi_sap_set_beacon_interval` to set the beacon interval.
- Call `scm_wifi_sap_set_dtim_period` to set the DTIM period.
- `scm_wifi_sap_set_config` will automatically call `scm_wifi_sap_stop` and `scm_wifi_sap_start` to activate SoftAP.

Step 3: Call `scm_wifi_sap_stop` to close the previous SoftAP.
Step 4: Call `scm_wifi_sap_start` to reactivate SoftAP.
Step 5: Call `scm_wifi_set_ip` to configure the network IP.
Step 6: Call `netifapi_dhcps_start` to start the DHCP Server.
Step 7: Call `netifapi_dhcps_stop` to stop the DHCP Server.
Step 8: Call `scm_wifi_sap_stop` to close SoftAP.
--- End

The return values of the API functions are shown in Table 4-2.

Table 4-2 Explanation of SoftAP Function Return Values

Definition	Value	Description
WISE_OK	0	Execution successful
WISE_FAIL	-1	Execution failed

4.3 Precautions

- Registering the event callback function (`scm_wifi_register_event_callback`) is essential to clearly understand when a SoftAP event occurs and to take the corresponding action.
- The network parameters for SoftAP can be pre-configured with default values.
- The network parameters for SoftAP will not reset when SoftAP is closed. Restarting the board will restore the initial default values.
- SoftAP only supports OPEN and WPA2 modes.

Senscomm Confidential

- Under SoftAP mode, the maximum number of associated users is limited to no more than one.

4.4 Programming Examples

Example: Implementing SoftAP functionality for startup, obtaining network information, and setting IP.

Senscomm Confidential

```
#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;
}
```

Senscomm

5 STA/SoftAP Coexistence

5.1 Overview

The coexistence of STA & SoftAP means that the STA and SoftAP functions work simultaneously. Depending on the startup order of STA & SoftAP, the following scenarios can be distinguished:

Scenario	Description
Coexistence on the same frequency and band	Full-time coexistence
Coexistence on the same frequency but different bands	Full-time coexistence
Coexistence on different frequencies but the same band	Time-shared coexistence
Coexistence on different frequencies and bands	Time-shared coexistence

Full-time coexistence: STA & SoftAP work simultaneously.
Time-shared coexistence: STA & SoftAP operate in their respective time slots.

5.2 Development Process

5.2.1 Use Cases

During network configuration, the product first starts SoftAP. After the phone connects to SoftAP, it sends home network information (SSID & password) to the product. Once the product receives this information, it starts the STA to connect to the home network. After the product successfully connects, it shuts down SoftAP, retaining only the STA to maintain a long-term connection with the home network. Other coexistence scenarios can be used according to product form and requirements.

Senscomm Confidential

5.2.2 Implementation Process

Step 1: Create a SoftAP network interface (for details, refer to "4 Wi-Fi SoftAP Function").
Step 2: The phone connects to the SoftAP network interface and sends home network information via the mobile app.
Step 3: [Optional] To avoid time-shared coexistence, it's recommended to restart SoftAP to the home network channel (for details, refer to "4 Wi-Fi SoftAP Function").
Step 4: Create a STA and, based on the home network information (SSID & password), complete the association (for details, see "3 Wi-Fi STA Function").
Step 5: Shut down SoftAP (for details, refer to "4 Wi-Fi SoftAP Function").

--- End

Return Value:
Please refer to the return value description of the corresponding module function.

5.3 Precautions

- Under time-shared coexistence, since STA & SoftAP take turns using time slots, performance may be suboptimal. It's recommended to start SoftAP on the channel where STA operates.

5.4 Programming Examples

Please refer to the programming examples in "3 Wi-Fi STA Function" and "4 Wi-Fi SoftAP Function".

Senscomm Confidential

Senscomm Confidential

6 Promiscuous Mode

6.1 Overview

Promiscuous Mode enables the Wi-Fi hardware to report all Wi-Fi frames on a specific channel and allows for sending raw 802.11 frames. This mode is essential for advanced network debugging and monitoring.

6.2 Development Process

6.2.1 Application Scenario

Wi-Fi raw frames are retrieved through the generic socket interface, which offers a more consistent and reliable data path application than callback registration methods.

6.2.2 Promiscuous API Functions

The following API functions are available for managing the promiscuous mode:

Table 6-1 Description of Promiscuous Function Interface

API Name	Description
scm_wifi_set_promiscuous	Enable the promiscuous mode for monitoring traffic.
scm_wifi_get_promiscuous	Queries the current state of promiscuous mode.
scm_wifi_80211_tx	Sends raw IEEE 802.11 data on the specified channel.
scm_wifi_set_channel	Sets the primary or secondary channel of the device.
scm_wifi_get_channel	Retrieves the currently set primary or secondary channel.

Senscomm Confidential

6.2.3 Implementation Process

Implementation involves the following clearly defined steps:

1. Enable promiscuous mode with `scm_wifi_set_promiscuous`.
2. Set the channel configuration with `scm_wifi_set_channel`.
3. Send a raw 802.11 frame using `scm_wifi_80211_tx`.
4. Optionally, retrieve 802.11 frames using a RAW socket for further analysis.

Table 6-2 Explanation of STA Function Return Values

Definition	Value	Description
WISE_OK	0	Indicates successful execution.
WISE_FAIL	-1	Indicates an error during execution.

6.3 Precautions

6.3.1 Operating Mode Considerations

To enable promiscuous mode effectively, ensure that Station (STA) and SoftAP operations are disabled. This can be done by invoking `scm_wifi_sta_stop` and `scm_wifi_sap_stop`.

Note: Functions such as `scm_wifi_80211_tx` and `scm_wifi_set_channel` are operational only when the device is in promiscuous mode.

6.3.2 Channel in monitor mode

The RF channel must be set each time promiscuous mode is enabled. For example, the following sequence is not valid:

1. Enable promiscuous mode on `wlan0`.
2. Set channel to 1 on `wlan0`.
3. Enable promiscuous mode on `wlan1`.

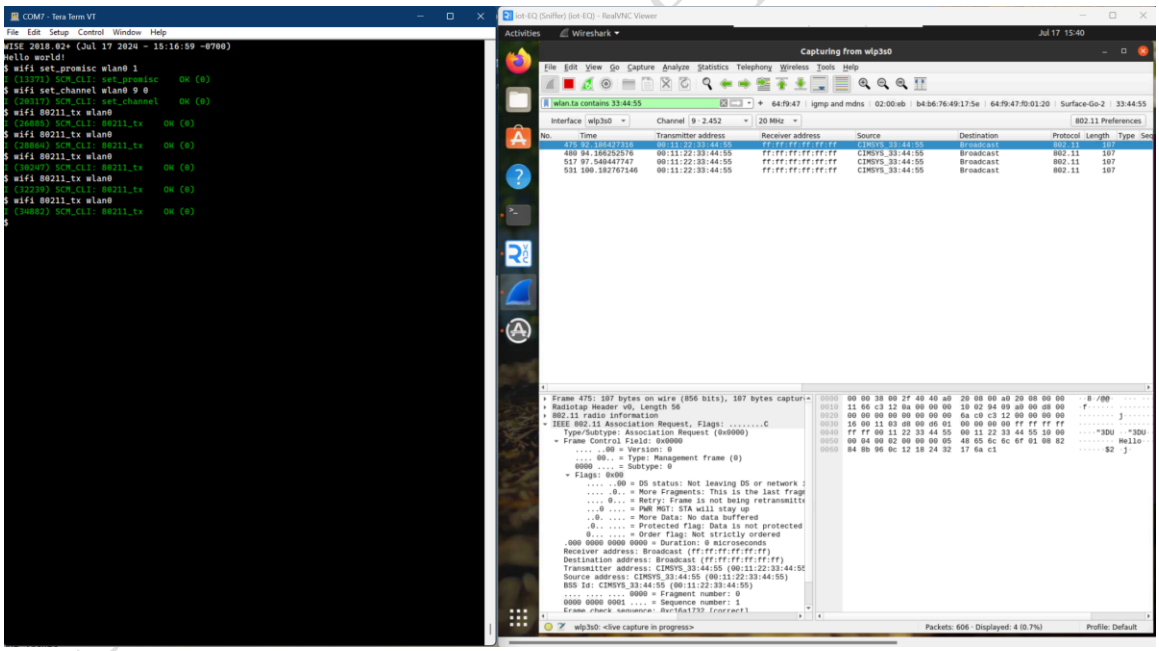
Senscomm Confidential

4. Set channel to 6 on `wlan1`.
5. Transmit a raw 802.11 frame on `wlan0`, expecting it will be sent out on channel 1.

Because the SCM1612s has only one RF chain, Step 4 above will overwrite the channel setting, resulting in the RF channel being set to 6 in Step 5.

6.4 Programming Examples

Example 1:
Send a raw 802.11 frame to channel 6 using the `scm_wifi_80211_tx` function. For implementation details, refer to `api/scm_cli_wifi.c`. Ensure that `CONFIG_CLI_WIFI_CHANNEL` and `CONFIG_CLI_WIFI_MONITOR` are enabled to utilize CLI commands effectively.



Example 2:
Display received 802.11 raw frames using the ‘wshark’ CLI command. This demonstration requires `CONFIG_CMD_WSHARK` to be enabled.

```
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 FAQs

Senscomm Confidential