4

SCM1612
Wi-Fi 6 和 BLE 5 低功耗 SoC

ScmChannel 使用指南

文档版本 1.1
发布日期 2023-08-15

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
1.2	2024-04-22	更新 APIs
1.1	2023-08-15	格式修改
1.0	2023-08-04	1.0 发布
0.1	2023-07-11	初稿

目录

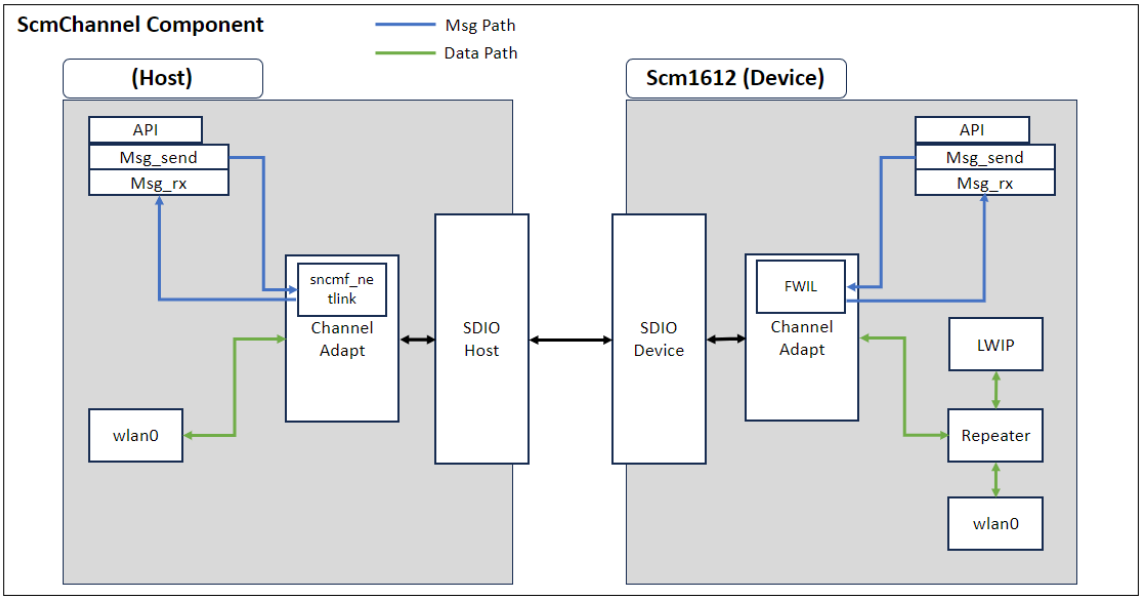
1 ScmChannel 概述

ScmChannel 组件主要提供低功耗 host-device 方案，其主要功能是搭建一个 host 芯片(如 Camera)与 Wi-Fi 芯片(device)通信的信道，该信道由两个子信道组成。

- Msg 通道: 主要用于传输与接收用户自定义之消息。
- Data 通道: 主要用于传输与接收网络数据包。

ScmChannel 软件运行架构如图 1-1 所示。

图 1-1 ScmChannel 组件基本框架图



组件说明:

- Wlan0: host 端与 device 端皆会有 wlan0 网络接口，与正常网络接口无区别，主要用于接收/发送网络数据包。
- Channel Adapt: 透过 host 端 sncmf_netlink 以及 device 端 FWIL 模块做到封装/去除 channel 通信协议。
- Repeater: Repeater 可以区分收到的网络数据包转发给 host 或是 device 端，亦可转发给两端。

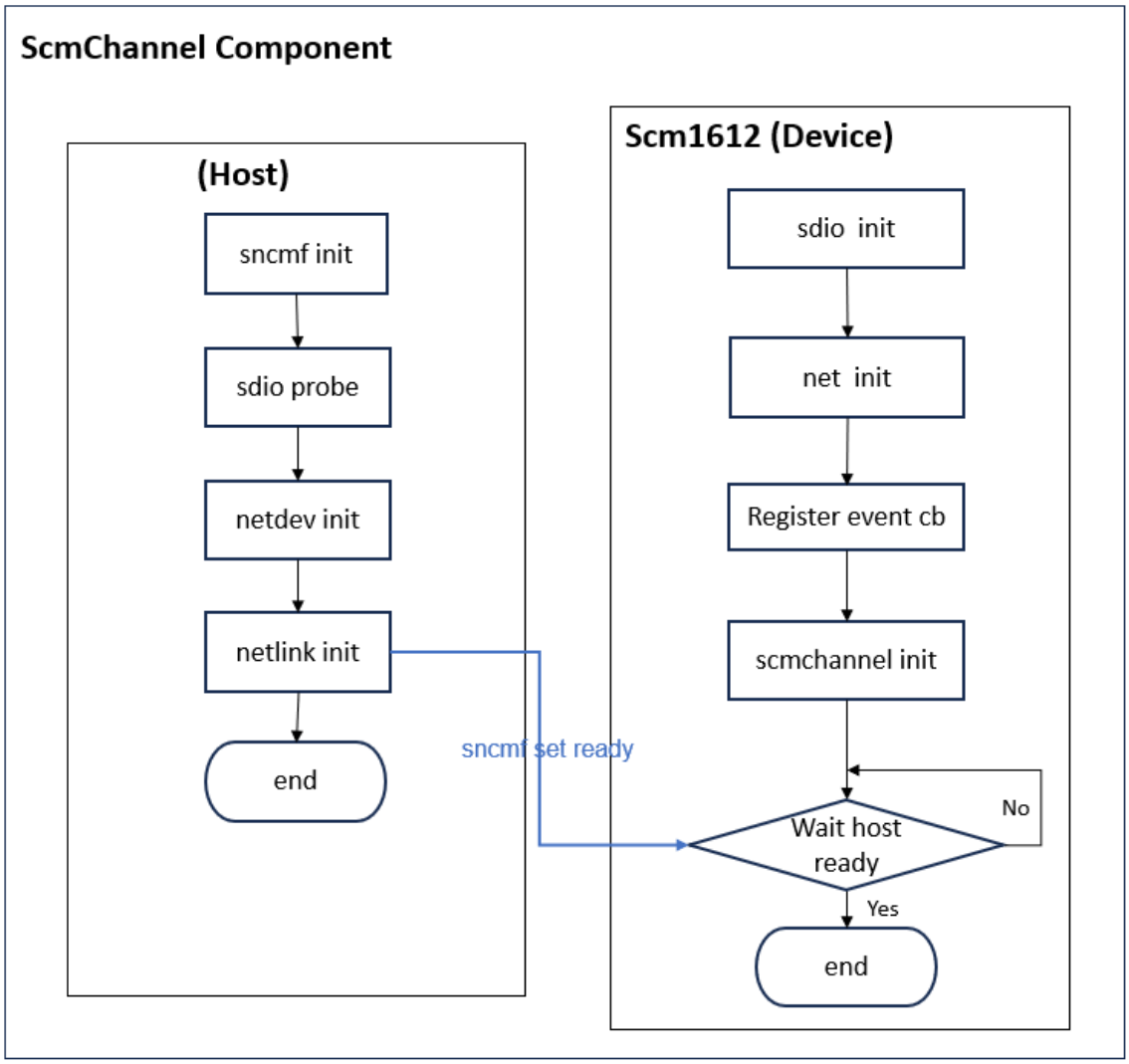
2 ScmChannel 使用指南

为了帮助用户快速将 ScmChannel 组件与自身业务对接，本章节对 ScmChannel 的基本使用规则做介绍，主要包括以下方面：

2.1 ScmChannel 组件初始化

ScmChannel 组件初始化的目的主要是建立主芯片与从芯片(Wi-Fi)的通道，保证主、从芯片可以能够进行网络数据通讯，亦能进行用户自定义消息通讯，ScmChannel 组件初始化流程如**图 2-1** 所示。

图 2-1 ScmChannel 组件初始化流程



- Host 侧:
  - sncmf init: 初始化整个 sncmf 模块。
  - sdio init: 初始化 sdio 相关资源。
  - netdev init: 初始化网络节点，用于接收传送网络数据包。
  - netlink init: netlink 初始化，主要用于建立应用层与内核信息传递，初始化完成会传送消息 sncmf set ready 到 SCM1612。
- device 侧:
  - sdio init: 使 sdio device 处于等卡状态，以便 sdio 主设备与其建立连接，此流程在 device 启动流程便会自动执行。

- register event cb: 建立网络节点网络属性变化的回调函数，并且启动 ScmChannel 传送信息的功能。

- ScmChannel init: 主要功能为设置 Repeater 模块以及注册 SCM1612 侧消息回调函数(Msg rx cb)。
  - 设置 Repeater: 此模块控制网络数据包转发规则，详细过滤规则请参见 2.3 Repeater 转发规则开发。
  - 注册回调函数: 注册 SCM1612 侧接收消息回调函数，主要用于用户专属的业务开发。

- wait host ready: SCM1612 侧会收到 host ready 通知，表示 ScmChannel 通道已开通。

ScmChannel 初始化过程 Demo 如下:

步骤 1: 注册网络节点网络属性变化的回调函数，并且启动 ScmChannel 传送讯息的功能，对应 API 为 scm_wifi_register_event_callback。
步骤 2: 初始化 scmchannel，对应 API 为 scm_channel_init。
步骤 3: 重置 Repeater 规则，对应 API 为 scm_vlwip_netif_reset。
步骤 4: 设置 Repeater 模块转发规则，对应 Demo code 函数为 scm_channel_set_default_filter。
步骤 5: 注册 SCM1612 侧接收消息回调函数，对应 API 为 scm_channel_register_rx_cb。
--- 结束

```
int scm_channel_init(void)
{

        if (scm_vlwip_netif_reset(WIFI_FILTER_TYPE_IPV4) != WISE_OK) {
                printf("%s: netif reset failed\n", __func__);
                return WISE_FAIL;
        }

        if (scm_channel_set_default_filter() != WISE_OK) {
                printf("%s: set_default_filter failed\n", __func__);
                return WISE_FAIL;
        }

        scm_channel_dump_filter();
        scm_channel_register_rx_cb(scm_channel_rx_callback);

        printf("ScmChannel    init OK\n");
        return WISE_OK;
}

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

        scm_wifi_register_event_callback(event_handler, NULL);

        scm_wifi_sta_start(ifname, &len);

#ifdef CONFIG_API_SCMCHANNEL
        scm_channel_init();
#endif
        ret = scm_wifi_start_connect();

        return ret;
}
```

2.2 待机唤醒配置 (待定)

2.3 Repeater 转发规则开发

ScmChannel 组件里面的 Repeater 模块提供了网络转发功能。可以通过调用 Repeater 模块 API 来达到特定网络封包的转发方向。host 侧可以利用 device 侧的 Repeater 模块来过滤需要的网络封包，可以将不需要的网络封包交由 device 侧处理。

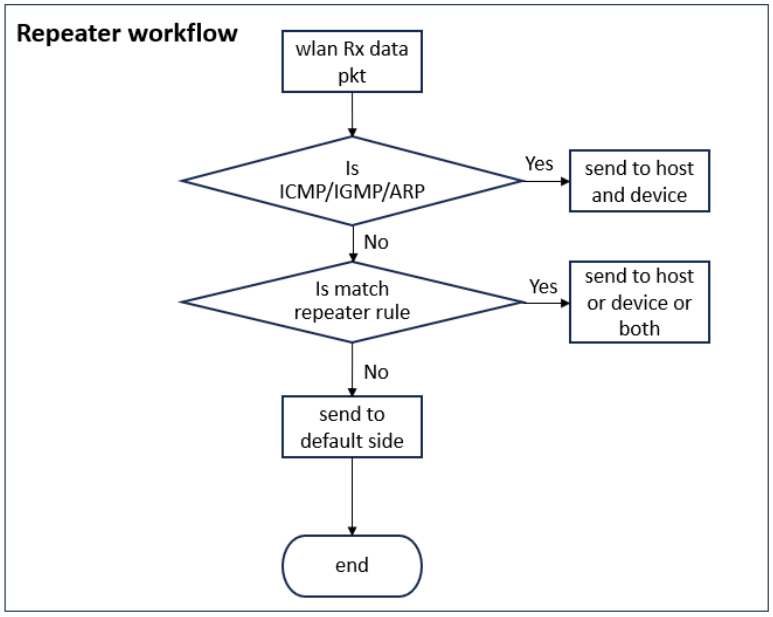
由于 host 与 device 共享相同的网络配置(MAC、IP 地址、netmask、getway)，ScmChannel 组件一旦建立成功后，用户可以在 host 侧直接使用 device 侧网络接口向外网发送网络包。

对于发送到外部的网络包，host 与 device 互不影响，统一都是通过 device 向外发送，发送行为不涉及转发规则。

对于网络收包行为，device 侧收到网络封包会依据 Repeater 规则做转发到 host 或是 device。

device 侧收到报文后的转发处理如下图 2-2:

图 2-2 Repeater 转发处理流程



- 来自外部的 ICMP、IGMP、ARP 都会给 host 与 device 均转发一份，无需特别配置转发规则，Repeater 预设以实现。
- 收到外部封包，但未匹配上任何转发规则，按默认规则进行转发 (预设 ScmChannel 已经设置为转发到 host 侧，用户端亦可透过 API scm_wifi_set_default_filter 作修改) 。
- 接收到外部的封包，若匹配上转发规则 (IP、协议类型、端口、端口范围)，按照配置的转发方向转给对应的目标 (host 或 device 或两侧皆转) 。

在 Repeater 初始化过程便会根据报文属性设置特定转发规则，转发配置以 IPV4 举例，配置转发规则的具体方法如下:

使用者依据业务需求的报文属性修改参数 ipv4_filter_def_setting[]，依据需求填写一至多组的 filter 数组参数。透过 scm_wifi_add_filter 函数会将每组 filter 新增至 Repeater 模块，其中需注意预设做多可以设定 15 组 filter，若需要更多组数可更改 config 参数。

```
static struct wifi_ipv4_filter ipv4_filter_def_setting[] = {
        /* UDP */
        {
                0,          /* 未设定 */
                7002,       /* 设置对端端口号，即报文携带的目的端口号需要为 7002 */
                0,          /* 未设定 */
                0,          /* 未设定 */
                0,          /* 未设定 */
                0,          /* 未设定 */
                0,          /* 未设定 */
                17,         /* 设置报文类型需要为 UDP 报文， UDP 为 17， TCP 为 6 */
                WIFI_FILTER_TO_LWIP,        /* 配置匹配本转发规则的报文转发到 device 侧 */
                WIFI_FILTER_MASK_REMOTE_PORT | WIFI_FILTER_MASK_PROTOCOL,
                /* 掩码字段配置需要同时匹配对端端口号、协议类型 */
        },
};
```

struct wifi_ipv4_filter 结构体定义如下:

```
/* Data Filter Structure */
struct wifi_ipv4_filter {
        unsigned int remote_ip; /*  选设字段，指定接收报文携带的对端 IP 地址。即报文中的源 IP 地址*/

        unsigned short local_port; /*  选设字段，指定接收报文对应的本端端口号。由于设备是接收方，即为对应报文携带的目的端口号*/

        unsigned short localp_min; /*  选设字段，指定接收报文对应的本端端口范围最小值。由于设备是接收方，即为对应报文的目的端口
                                        范围最小值*/

        unsigned short localp_max; /*  选设字段，指定接收报文对应的本端端口范围最大值。由于设备是接收方，即为对应报文的目的端口号
                                        范围最大值，最大值需大于最小值*/

        unsigned short remote_port; /*  选设字段，指定接收报文对应的对端端口号。由于设备是接收方，即为对应报文携带的源端口号*/

        unsigned short remotep_min; /*  选设字段，指定接收报文对应的对端端口范围最小值。由于设备是接收方，即为对应报文的源端口号
                                        范围最小值*/

        unsigned short remotep_max; /*  选设字段，指定接收报文对应的对端端口范围最大值。由于设备是接收方，即为对应报文的源端口号
                                        范围最大值，最大值需大于最小值*/

        unsigned char packet_type; /*  选设字段，指定接收报文采用的传输层协议，一般指定为 TCP（6）或者 UDP（17）
*/

        unsigned char config_type; /*  必设字段，设置匹配本转发规则的报文的转发方向， 设置为 WIFI_FILTER_LWIP 转发到 device 侧，设
                                        置为 WIFI_FILTER_VLWIP 转发到主控侧， WIFI_FILTER_BOTH 则两边都转发*/

        unsigned char match_mask; /*  必设字段，指定上述哪些选设字段是有效值，每个选设字段通过对应 Bit 掩码置位。置位之后代表需要
                                        同时满足匹配该字段，例如 WIFI_FILTER_MASK_IP |
                                        WIFI_FILTER_MASK_PROTOCOL 值为 0x1 | 0x2 = 0x3,表示报文需要同时满足匹配源 IP 地址和协议类型（TCP 或 UDP）才匹配本规则，匹配之后按照 config_type 指定的方向转发本报文 */
        unsigned char   resv;   /*  保留字段 */
};
```

上述 wifi_ipv4_filter 结构中 match_mask 字段，通过掩码枚举值对应的组合来定义。例如 “WIFI_FILTER_MASK_REMOTE_PORT | WIFI_FILTER_MASK_PROTOCOL”表示报文需要同时匹配上指定的 IP 地址和对端端口号才算匹配本条规则。

针对 wifi_filter_field_enum 定义中源端口和目的端口的比对规则加以说明：

- 如果源端口号不匹配，则再去匹配指定的源端口号范围。
- 如果源端口号可以匹配，则认为源端口号范围也已匹配。

对于同时匹配报文的目的端口号和目的端口号范围的场景也类似。

```
/* Data Filter Item */
typedef enum {
        WIFI_FILTER_MASK_IP = 0x01, /*  掩码枚举：表示源 IP 地址，对应 remote_ip 字段  */

        WIFI_FILTER_MASK_PROTOCOL = 0x02, /*掩码枚举：协议类型（  TCP 或 UDP)，对应 packet_type 字段，不指
                                            则表示该字段不作为匹配条件  */

        WIFI_FILTER_MASK_LOCAL_PORT = 0x04, /*  掩码枚举：接收到报文的目的端口号，对应 local_port 字段，不指
                                            定则表示该字段不作为匹配条件  */

        WIFI_FILTER_MASK_LOCAL_PORT_RANGE = 0x08, /*  掩码枚举：接收到报文的目的端口号范围，对应
                                            localp_min 和 localp_max 字段，不指定则表示该字段不作为
                                            配条件  */

        WIFI_FILTER_MASK_REMOTE_PORT = 0x10, /*  掩码枚举：接收到报文的源端口号，对应 remote_port 字段，不指
                                            定则表示该字段不作为匹配条件*/

        WIFI_FILTER_MASK_REMOTE_PORT_RANGE = 0x20, /*掩码枚举：接收到报文的源端口号范围，对应
                                        remotep_min 和 remotep_max 字段，不指定则表示该字段不作为匹配条  */

        WIFI_FILTER_MASK_BUTT
} wifi_filter_field_enum;
```

在范例应用中，预设 DHCP 封包传送至 lwip 端，并且通过网络事件变化接口更新 DHCP 资讯至 host 端，如果服务器提供独立端口专供 device 侧进行业务交互，则可利用远端端口号作为转发规则识别并转发来自该服务器的报文到 device 侧。用户可以利用 Repeater 规则将适当的任务分配至 host 或 device 侧，从而实现省电效果。

DHCP 转发规则配置如下

```
static struct wifi_ipv4_filter ipv4_filter_def_setting[] = {
        /* DHCP */
        {
                0,                                                          /* remote ip */
                68,                                                         /* local port */
                0,                                                          /* localp_min */
                0,                                                          /* localp_max */
                0,                                                          /* remote_port */
                0,                                                          /* remotep_min */
                0,                                                          /* remotep_max */
                17,                                                         /* packet type */
                WIFI_FILTER_TO_LWIP,                                        /* config_type */
                WIFI_FILTER_MASK_LOCAL_PORT | WIFI_FILTER_MASK_PROTOCOL, /* match_mask */
        },
        /* DHCP */
        {
                0,                                                          /* remote ip */
                67,                                                         /* local port */
                0,                                                          /* localp_min */
                0,                                                          /* localp_max */
                0,                                                          /* remote_port */
                0,                                                          /* remotep_min */
                0,                                                          /* remotep_max */
                17,                                                         /* packet type */
                WIFI_FILTER_TO_LWIP,                                        /* config_type */
                WIFI_FILTER_MASK_LOCAL_PORT | WIFI_FILTER_MASK_PROTOCOL, /* match_mask */
        }
}
```

ScmChannel 组件 Repeater 相关 API 如下:

- scm_wifi_set_default_filter
- scm_wifi_add_filter
- scm_wifi_del_filter
- scm_wifi_query_filter

2.4 API 使用指南

ScmChannel 提供的 API 如表 2-1 所示。

Name	Description
scm_channel_set_default_filter	设置 Repeater 默认的过滤转发方向(预设为转发到 host 侧)。
scm_channel_add_filter	添加过滤规则到 Repeater 的转发表。
scm_channel_del_filter	从 Repeater 的转发表删除过滤规则。
scm_channel_query_filter	查询 Repeater 的转发表内容。

scm_channel_reset_filter	将 filter 配置重置，在 bootup 过程中 Repeater 会有一组预设 filter 配置，用户更新自定义 filter 前，须先执行此 API。
scm_channel_send_to_host	Device 传送消息到 host，每次发送最大字节为 1500 bytes。
scm_channel_register_rx_cb	注册接收处理回调函数，用于处理从 host 接收到的消息数据。
scm_channel_host_ready	此 API 用来判断 host 是否处于可以接收消息状态。

注意事项:

- 预设的 Repeater 组件过滤规则数为 15 组，如果想要修改，请调整 CONFIG_SUPPORT_WIFI_REPEATER_IPV4_CNT 宏。
- 在 Scm1612 bootup 过程中，预设配置了多组 filter。用户可以通过 repeater filter show 来查询：

```
$ repeater filter show
[001539.161481]
 Al$ low filter count 15 default_dir 2
[001539.161692]
----- Total Filter Count 6 -----
[001539.161880] [0] protocol(17) dest port(68) config_type(1) match_mask(0x6)
[001539.162169] [1] protocol(6) dest port(5201) config_type(2) match_mask(0x6)
[001539.163224] [2] protocol(6) dest port(0) config_type(2) match_mask(0x12)
[001539.163506] [3] protocol(17) dest port(5201) config_type(2) match_mask(0x6)
[001539.163805] [4] protocol(17) dest port(0) config_type(2) match_mask(0x12)
[001539.164095] [5] protocol(6) dest port(1234) config_type(2) match_mask(0x26)
```

建议用户在配置自定义 filter 之前执行 scm_vlwip_netif_reset 来进行清除动作。

2.5 芯片间心跳机制 (待定)

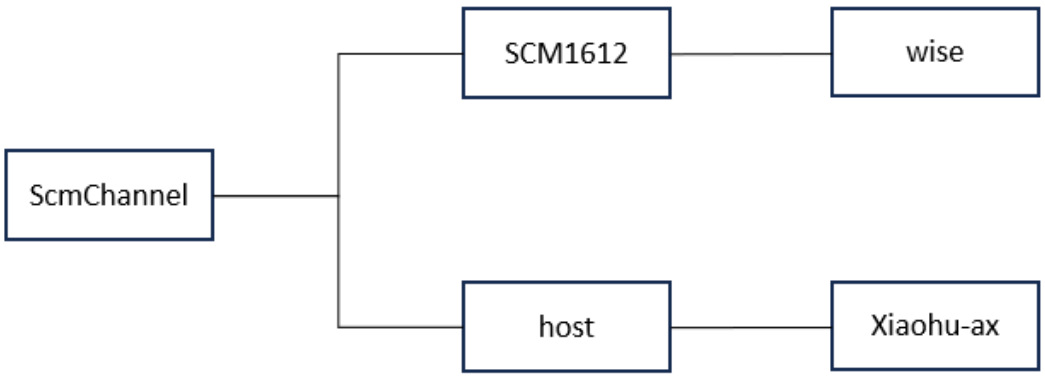
3 ScmChannel 组件使用指南

ScmChannel 组件作为 SCM1612 系统的组成成份，本章节对如何将 ScmChannel 软件包集成到 SCM1612 软件包中做了详细描述，方便用户能够快速将 ScmChannel 组件集成到 SCM1612 版本中。

3.1 ScmChannel 软件包目录结构 (Need Confirm Later)

ScmChannel 软件包目录如图 3-1 所示

图 3-1 ScmChannel 软件包目录结构



3.2 ScmChannel 组件编译

步骤 1: SCM1612 侧编译

在 SCM1612 下的 wise-sdk 目录执行

```
make distclean
make scm1612s_sdio_defconfig
```

```
make
```

在 wise-sdk 目录下生成 wise.mcuboot.bin

步骤 2: Host 侧编译

i.
```
cd xiaohu-ax/
cp configs/cfg_xxx.mk cfg.mk
```
由于支持多种平台和配置模式，如 SDIO OOB 模式、SDIO INT 模式、USB 模式等。您需要依据平台与需求首先选择一个配置文件进行编译。系统默认提供配置文件保存在 configs/目录下：

配置文件	平台/模式
cfg_sdio_normal_fullhan.mk	fullhan, 4 bit mode interrupt
cfg_sdio_normal_goke.mk	goke, 4 bit mode interrupt
cfg_sdio_normal.mk	Linux PC, 4 bit mode interrupt
cfg_sdio_polling.mk	Linux PC, polling mode
cfg_sdio_oob_int_goke.mk	goke, OOB interrupt mode
cfg_usb.mk	Linux PC

ii. 编译驱动 sncmfmac.ko

虽然 ARCH 和 CROSS_COMPILE 在文件中已经配置好，但参数'KDIR'需要根据个人环境进行明确指定。

```
make KDIR=/home/apache/page/linux-4.9
```

如果没有指定'KDIR'，系统则会使用默认的内核 KDIR，这将编译出的驱动将适用于 LinuxPC 上运行。

```
make
```

iii. 编译应用程序

```
make KDIR=/home/apache/page/linux-4.9 apps
```

目前有两种类型的应用程序：

'sncm_cmd'：通过主机命令进行 Wi-Fi 配置和控制。

'sncm_chn'（也称为 ScmChannel）：Wi-Fi 配置和控制在 SCM1612 侧完成，通过 ScmChannel 获取 Wi-Fi 连线信息。'sample_link'主要用于同步 SCM1612 侧网络节点的 Mac 地址、IP 地址等信息。

'sample_cli':主要用于发送客户自定义的信息

3.3 ScmChannel 使用示例

3.3.1 配置网络信息并连接

首先在 1612 端，通过 SCM1612_Wi-Fi 软件开发指南.docx 的第三章节:Wi-Fi STA 功能里的 api 函数或者 cli 命令行来配置网络：

```
$ wifi reg_evt_cb
reg_evt_cb OK (0)
$ wifi sta_start
ifname: wlan0
sta_start OK (0)
$ wifi sta_cfg TPTest 0 0 00:00:00:00:00:00 0
sta_cfg OK (0)
$ wifi sta_connect
sta_connect OK (0)
$
STA_CONNECTED
AP SSID: TPTest
AP BSSID: ec:60:73:8:0:a8
AP CH: 11
AP RSSI: -8
Status: CONNECTED

WIFI CONNECTED indicate

WIFI GOT IP

WIFI GOT IP wait host app

$ ifconfig wlan0
wlan0: flags=104451<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  metric 0
        inet 192.168.1.101  netmask 255.255.255.0  broadcast 192.168.1.255
        ether 64:f9:47:f0:01:20  txqueuelen 0
        RX packets 14 dropped 0
        TX packets 3
```

3.3.2 确认 SDIO 连接

在 Host 端，首先确认 SDIO 已连接：

```
/tmp/0721_scmchannel # insmod sncmfmac.ko debug=0x001fffdf
sncmfmac: loading out-of-tree module taints kernel.
sncmfmac: sncmf_c_preinit_dcmds: Chip     : 0x0
sncmfmac: sncmf_c_preinit_dcmds: Chip  Rev: 0x0
sncmfmac: sncmf_c_preinit_dcmds: Board Rev: 0x0
sncmfmac: sncmf_c_preinit_dcmds: Ucode Rev: 0x4bc054bb
/tmp/0721_scmchannel # random: crng init done
```

3.3.3 建立 ScmChannel 连接

a. 1612 端，初始化 ScmChannel

```
$ wifi scm_ch_init
set all net packets foward to camera default.
add filter failed: duplicate element
add filter failed: duplicate element
scm_ch_init OK (0)
$ WiFi: Scm channel send msg
WiFi: Scm channel send msg
```

b. Host 端通过 sample_link 自动获取 wlan0 IP 和 mac 地址。
成功后，可以通过 ifconfig wlan0 确认信息已经同步。

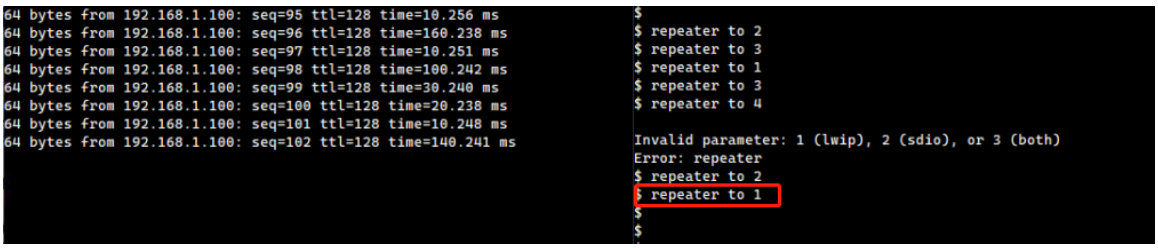
```
/tmp/0721_scmchannel # ./sample_link &
/tmp/0721_scmchannel # [sncmchannel_main.c][sample_wlan_init_up][l:487],net device up success

[sncmchannel_host.c][sncm_channel_host_thread][l:77],type 28, data len 7, recv len 24, hdr len 16

[sncmchannel_main.c][sample_link_rx_cb][l:254],index:0

[sncmchannel_main.c][sample_link_rx_cb][l:256],msg[7] 0:64:f9:47

[sncmchannel_main.c][sample_wlan_init_mac][l:239],net device set mac success

[sncmchannel_host.c][sncm_channel_host_thread][l:77],type 28, data len 14, recv len 32, hdr len 16

[sncmchannel_main.c][sample_link_rx_cb][l:254],index:1

[sncmchannel_main.c][sample_link_rx_cb][l:256],msg[14] 1:c0:a8:1

[sncmchannel_main.c][sample_wlan_init_ip][l:178],net device set ip success


/tmp/0721_scmchannel # ifconfig
wlan0     Link encap:Ethernet  HWaddr 64:F9:47:F0:01:20
          inet addr:192.168.1.101  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:146 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:37264 (36.3 KiB)  TX bytes:1180 (1.1 KiB)
```

3.3.4 更改转发规则

a. Host ping 通局域网里的 PC1

```
/tmp/0721_scmchannel # ping 192.168.1.100
PING 192.168.1.100 (192.168.1.100): 56 data bytes
64 bytes from 192.168.1.100: seq=0 ttl=128 time=118.575 ms
64 bytes from 192.168.1.100: seq=1 ttl=128 time=170.243 ms
64 bytes from 192.168.1.100: seq=2 ttl=128 time=60.243 ms
64 bytes from 192.168.1.100: seq=3 ttl=128 time=110.241 ms
```

b. 修改转发规则，只转发到 1612 端，不转发到 Host 端。Ping 中断



c. 修改转发规则，同时转发到 Host 端和 1612 端，ping 恢复