密级状态：绝密（　　）　秘密（　　）　内部（ ）　公开（ √ ）

RK3308 RTL8723DS WIFI/BT 配置说明

（技术部，系统产品一部）

文件状态： [√] 正在修改 [　] 正式发布	当前版本：	V1.0
	作　　者：	许学辉
	完成日期：	2018-08-10
	审　　核：	
	完成日期：	

福州瑞芯微电子有限公司

Fuzhou　Rockchips　Semiconductor　Co . , Ltd

版 本 历 史

版本号	作者	修改日期	修改说明	备注
V1.0	许学辉	2018-08-10	对外发布第一个版本	

目 录

1 目的 2

1.1 内核注意事项 2

1.2 BUILDROOT 注意事项 6

2 WIFI 手动连接和扫描 8

3 WIFI BLE 配网 10

4 WIFI SOFTAP 配网 13

5 BT 开关 15

6 BT A2DP SINK 16

7 BT HFP SCO、PCM 17

7.1 HFP 电话呼入接通原理简要介绍 17

7.2 RK 平台 HFP 测试 DEMO 介绍 18

8 BT AVRCP 控制 22

8.1 AVRCP 实现原理简要说明 22

8.2 LIBGDBUS-INTERNAL 库的引用 24

8.3 编译配置 24

8.4 AVRCPCTRL API 说明 24

9 A2DP SINK 音量调节 27

9.1 BLUEZ-ALSA ORG.BLUEZ.MEDIATRANSPORT1 27

9.2 直接控制喇叭音量 28

10 WIFI 无法打开问题排查 29

1 目的

明确 RK3308 linux 平台上 wifi、bt 配置，按照本文档可以进行 WIFI BT 相关开发和测试，目前 linux 平台 RTL8723DS WIFI 使用 wpa_supplicant,并配合 wpa_cli 工具进行联网，蓝牙使用 bluez + bluez-alsa 开源软件实现 A2DP sink，HFP 功能以及 BLE 蓝牙配网功能，该文档可以给客户提供 APP 开发参考。

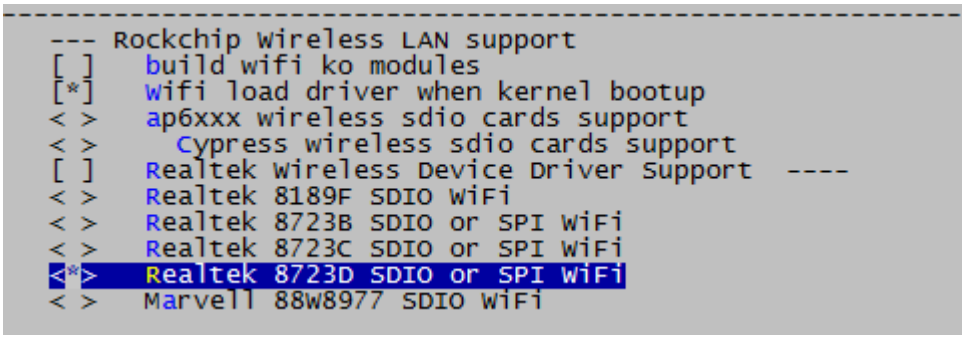
1.1 内核注意事项

Device Drivers --->

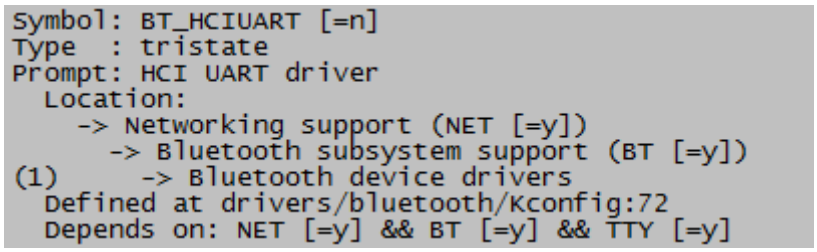
[*] Network device support --->

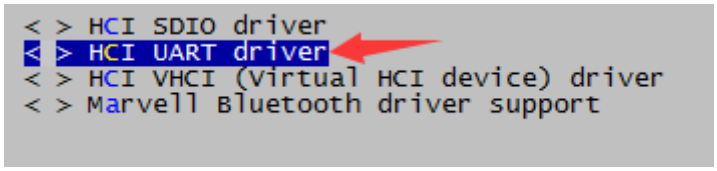
[*] Wireless LAN --->

选择 8723DS wifi 驱动，去掉默认的 CONFIG_CYW_BCMDHD 驱动



去掉默认内核默认 CONFIG_BT_HCIUART 驱动





板级 dts 先确 WIFI/BT 上电管脚和 UART 信息：

a. **reset-gpio** 为 wifi power 管脚，确认其 pintrl 设定。

b. **蓝牙使用 3308 ttys4 进行通讯，配置好 CTS 管脚，BT,power_gio 为蓝牙的上电管脚，根据实际的硬件进行修改**

```
wireless-bluetooth {
        compatible = "bluetooth-platdata";
        uart_rts_gpios = <&gpio4 RK_PA7 GPIO_ACTIVE_LOW>;
        pinctrl-names = "default", "rts_gpio";
        pinctrl-0 = <&uart4_rts>;
        pinctrl-1 = <&uart4_rts_gpio>;
        BT,power_gpio     = <&gpio4 RK_PB2 GPIO_ACTIVE_HIGH>;
        BT,wake_host_irq = <&gpio4 RK_PB4 GPIO_ACTIVE_HIGH>;
        status = "okay";
};

wireless-wlan {
        compatible = "wlan-platdata";
        rockchip,grf = <&grf>;
        pinctrl-names = "default";
        pinctrl-0 = <&wifi_wake_host>;
        wifi_chip_type = "rtl8723ds";
        WIFI,host_wake_irq = <&gpio0 RK_PA0 GPIO_ACTIVE_LOW>;
        status = "okay";
};
```

```
sdio_pwrseq: sdio-pwrseq {
        compatible = "mmc-pwrseq-simple";
        pinctrl-names = "default";
        pinctrl-0 = <&wifi_enable_h>;

        /*
         * on the module itself this is one of these (depending
         * on the actual card populated):
         * - SDIO_RESET_L_WL_REG_ON
         * - PDN (power down when low)
         */
        reset-gpios = <&gpio0 RK_PA2 GPIO_ACTIVE_LOW>;
};
```

```
};

sdio-pwrseq {
        wifi_enable_h: wifi-enable-h {
                rockchip,pins = <0 RK_PA2 RK_FUNC_GPIO &pcfg_pull_none>;
        };
};
```

c. 8723DS WIFI 芯片，如果采用 COB（realtek chip on borad）和模块（欧智通/小瑞）PCM_OUT 管脚注意事项

首先模块的 PIN25 管脚和 COB 的 PIN 22 管脚是 8723DS 内部复用管脚，一个是作为 PCM_OUT 连接 RK3308 PCM_IN，用作蓝牙 PCM 通话功能； 另外一个作为 WIFI IC LDO_SPS_SEL 功能(SPS 或者 LDO 模式选择)，当这个管脚为低电平表示 SPS 模式，如

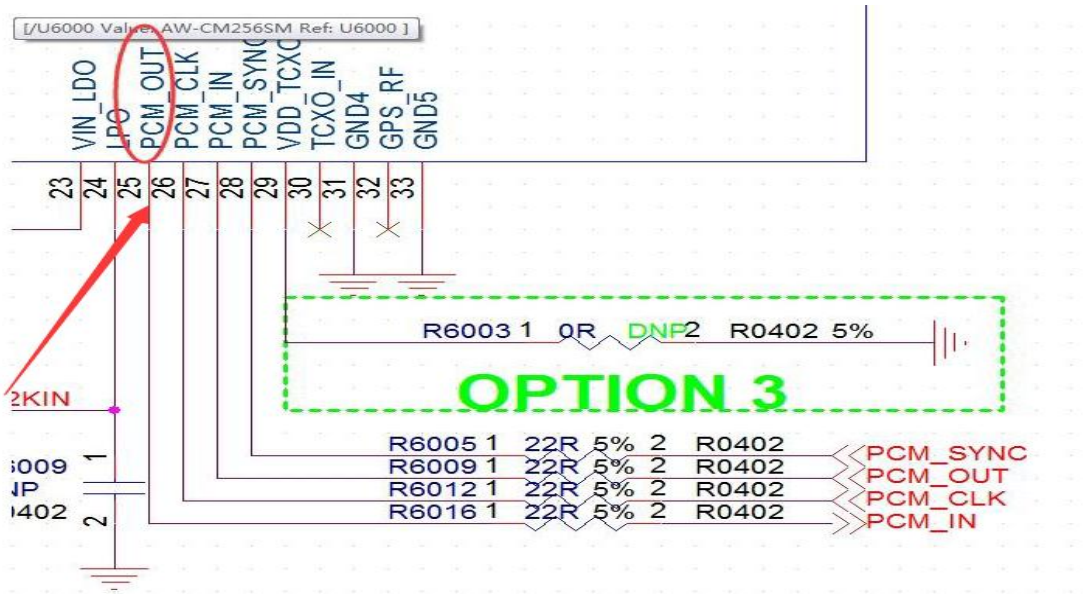
果为高电平为 LDO 模式，在跟 WIFI 模块上电的时候，LDO_SPS_SEL 会选择决定供给 WIF IC 内部的 1.2V 是何种方式，如果电平不对，会导致 WIFI 无法正常使用，比如无法扫描 SDIO 或者扫描不到 AP 问题。

对于选择 COB 和 823DS 模块，PIN 22(realtek COB)或者 PIN25(欧智通/小瑞) 脚是连接到了 RK3308 内部下拉管脚(GPIO4_C0)，

对于 COB 而言工作在 SPS 模式，低电平，但是对于模块，工作在 LDO 模式，连接到了这个 RK3308 内部下拉脚(GPIO4_C0)，就会出现半点平情况，导致 WIFI 异常，

解决办法两个：

方法 1：使用模块的版型，**硬件上将 WIFI 模块的 PIN25 脚接一个上拉电阻，直接拉高，工作在 LDO 模式**，避免跟 RK3308 内部下拉脚（GPIO4_C0）对拉出现半电平， 8723DS 模块 PCM_OUT PIN 如下图：



方法 2：软件配置在 WIFI 拉高前，将 GPIO4_C0 设置为高电平有效，但是又要考虑到 PCM RK 芯片的复用，暂时按照如下修改，**注意 COB 的不能如下修改，否则又会出现半电平对拉导致 WIFI 无法使用。**

修改 rk3308.dts 和 rk3308-evb-vxx.dtsi

```
@@ -693,7 +716,8 @@

 &pinctrl {
        pinctrl-names = "default";
-       pinctrl-0 = <&rtc_32k>;
+       pinctrl-0 = <&rtc_32k &i2s_2ch_0_sdi>;
+
```

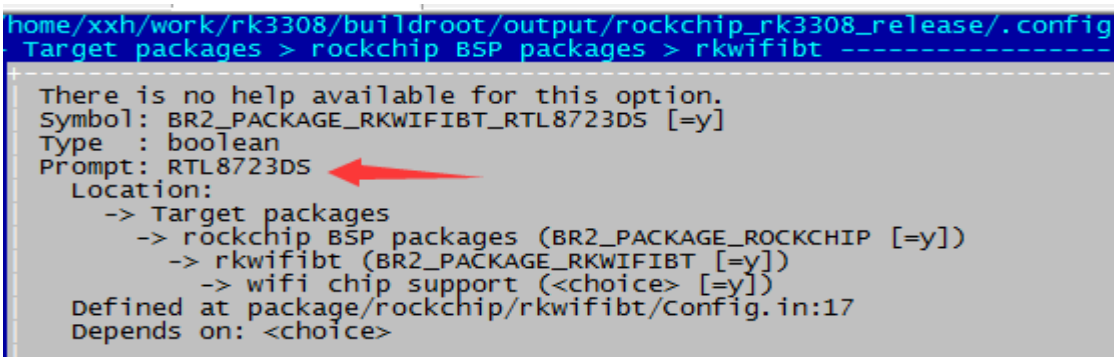
```
diff --git a/arch/arm64/boot/dts/rockchip/rk3308.dtsi b/arch/arm64/boot/dts/rockchip/rk3308.dtsi
index f628916..12f2977 100644
--- a/arch/arm64/boot/dts/rockchip/rk3308.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3308.dtsi
@@ -835,7 +835,6 @@
                pinctrl-names = "default";
                pinctrl-0 = <&i2s_2ch_0_sclk
                             &i2s_2ch_0_lrck
-                            &i2s_2ch_0_sdi
                             &i2s_2ch_0_sdo>;
                status = "disabled";
        };
@@ -1261,7 +1260,7 @@

                        i2s_2ch_0_sdi: i2s-2ch-0-sdi {
                                rockchip,pins =
-                                       <4 RK_PC0 RK_FUNC_1 &pcfg_pull_none>;
+                                       <4 RK_PC0 RK_FUNC_1 &pcfg_pull_up>;
                        };
                };
```

```
&pinctrl {
        pinctrl-names = "default";
        pinctrl-0 = <&rtc_32k &i2s_2ch_0_sdi>;

        buttons {
                pwr_key: pwr-key {
                        rockchip,pins = <0 RK_PA6 RK_FUNC_GPIO &pcfg_pull_up>;
                };
        };

        usb {
                usb_drv: usb-drv {
                        rockchip,pins = <0 RK_PC5 RK_FUNC_GPIO &pcfg_pull_none>;
                };
        };

        sdio-pwrseq {
                wifi_enable_h: wifi-enable-h {
                        rockchip,pins = <0 RK_PA2 RK_FUNC_GPIO &pcfg_pull_none>;
                };
        };

        wireless-wlan {
                wifi_wake_host: wifi-wake-host {
                        rockchip,pins = <0 RK_PA0 RK_FUNC_GPIO &pcfg_pull_up>;
```

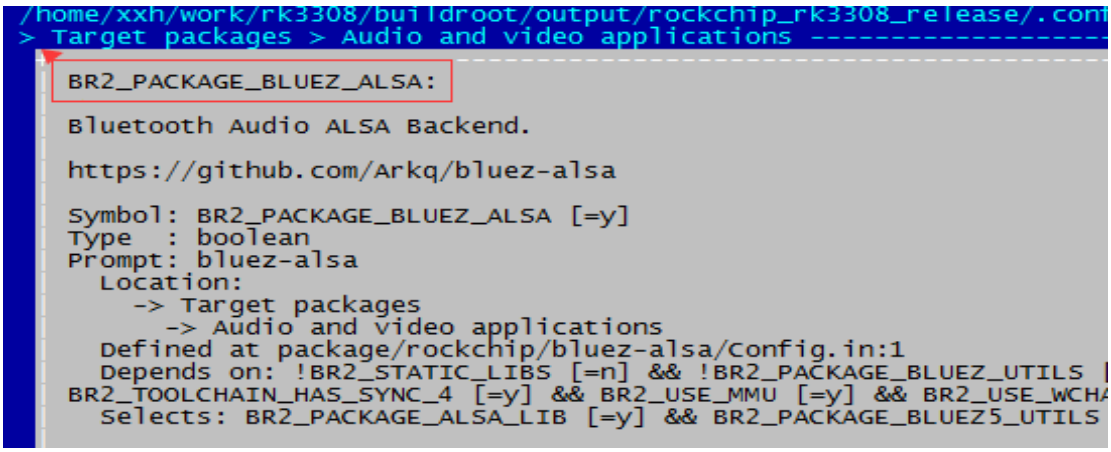
1.2 buildroot 注意事项

需要配置 RTL8723DS 蓝牙驱动，同时选择必要的 bluez 工具和 bluez-alsa 工具，buildroot 编译前 make menuconfig 选择如下 PACKAGE

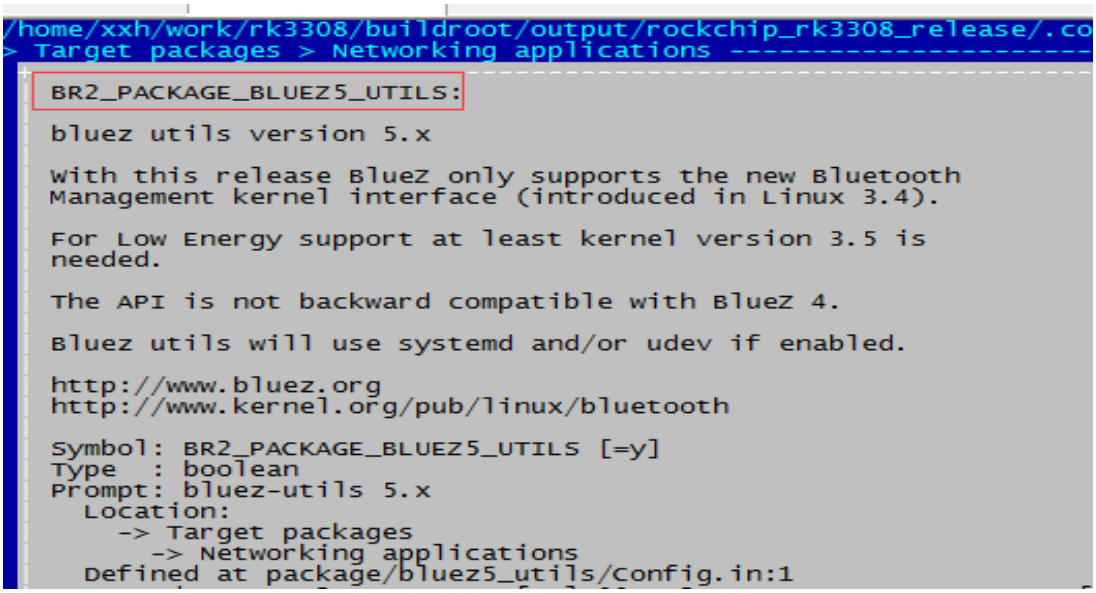
1. **选择 BR2_PACKAGE_RKWIFIBT_RTL8723DS**



2. **选择 BR2_PACKAGE_BLUEZ_ALSA，RK 蓝牙 A2DP SINK/HFP 基于 bluez-alsa 实现**



3. 选择 **BR2_PACKAGE_BLUEZ5_UTILS**，该 package 有很多蓝牙相关工具，客户可以根据需要自由选择，方便蓝牙相关调试



客户可以将如下配置默认集成到 buildrootconfigs/rockchip_rk3308_release_defconfig

BR2_PACKAGE_BLUEZ5_UTILS=y

BR2_PACKAGE_BLUEZ5_UTILS_OBEX=y

BR2_PACKAGE_BLUEZ5_UTILS_CLIENT=y

BR2_PACKAGE_BLUEZ5_UTILS_EXPERIMENTAL=y

BR2_PACKAGE_DBUS=y

BR2_PACKAGE_DBUS_GLIB=y

2 WIFI 手动连接和扫描

启动 wpa_supplicant：

wpa_supplicant -B -i wlan0 -c /data/cfg/wpa_supplicant.conf

修改如下文件：

/ # vi /data/cfg/wpa_supplicant.conf

ctrl_interface=/var/run/wpa_supplicant

ap_scan=1

#添加如下配置项

network={

ssid="WiFi-AP" // WiFi 名字

psk="12345678" // WiFi 密码

key_mgmt=WPA-PSK // 加密方式

key_mgmt=NONE // 不加密

}

重新读取上述配置：wpa_cli reconfigure

并重新连接：**wpa_cli reconnec**t，WIFI 连接后，系统即可上网，可以进行 PING 或者 iperf 相关测试，或者基于 linux 的网络应用测试。

wpa_cli 进行 wifi 扫描

wpa_cli

Selected interface 'wlan0'

Interactive mode

> **scan**

OK

<3>CTRL-EVENT-SCAN-STARTED

<3>CTRL-EVENT-SCAN-RESULTS

<3>CTRL-EVENT-NETWORK-NOT-FOUND

> **scan_results**

bssid / frequency / signal level / flags / ssid

f4:ec:38:2c:2e:0c 2412 -45 [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]

TP-LINK_FF

14:75:90:49:b4:26 2457 -49 [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS] RkMra

bc:5f:f6:68:e1:b2 2472 -52 [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS] hellowifi

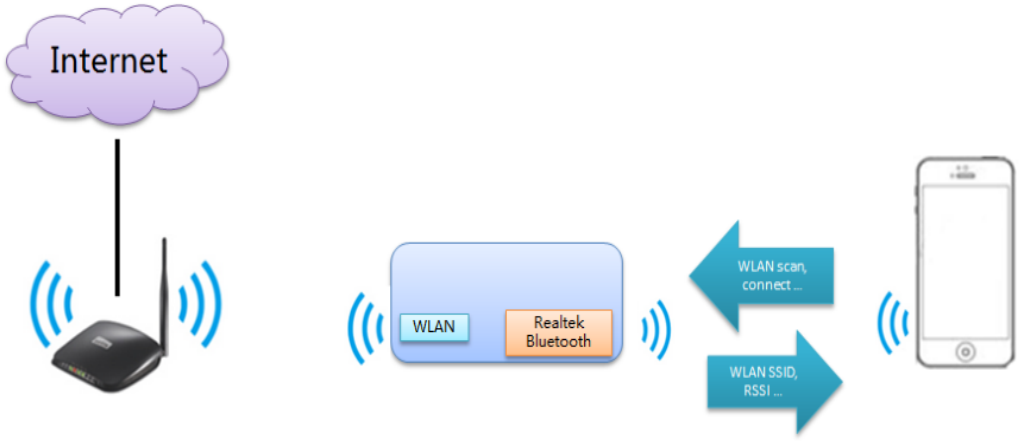
78:44:76:18:0f:44 2437 -53 [WPA2-PSK-CCMP][ESS] TP-LINK-FZK

>

3 WIFI BLE 配网

3.1 RTL8723DS 配网说明

配置 WLAN 的框架结构图如下：



手机通过蓝牙连接带 BT 和 WiFi 的设备，实现快速配置 WiFi 的功能，手机端 WLAN app 和 RK3308 端 gatt-server 数据交换基于 BLE GATT，gatt-server 是 BlueZ 5.50 及以上版本， GATT server 参考代码，位于 BlueZ source code tools/gatt-server.c， RK 提供的 gatt-server 添加了供 Client 端连接的 Service 以及相应的 characteristic，Service 和 characteristic 概念可查询 BLE spec 官方文档。每个 service 和相应 characteristic 通过 UUID 唯一标识，gatt-server 实现也是基于 Dbus 接口封装，开发 app 需要导入 Dbus-glib 标准库，在 BT AVRCP 章节有说明，gatt-server 程序定义了定义了以下接口：

```
#define GATT_MGR_IFACE                  "org.bluez.GattManager1"
#define GATT_SERVICE_IFACE              "org.bluez.GattService1"
#define GATT_CHR_IFACE                  "org.bluez.GattCharacteristic1"
#define GATT_DESCRIPTOR_IFACE           "org.bluez.GattDescriptor1"
```

其中 GATT_SERVICE_IFACE 为 org.bluez.GattService1 接口，然后定义了 WIFI_SERVICES_UUID 服务 uuid 和对应的特征值 uuid 服务

```
#define WIFI_SERVICES_UUID      "1B7E8251-2877-41C3-B46E-CF057C562023" //wifi 配网服务
```

创建和注册 WIFI_SERVICES 服务，并为该服务注册特征值，具体描述如下：

#define SECURITY_CHAR_UUID "CAC2ABA4-EDBB-4C4A-BBAF-0A84A5CD93A1" //wifi psk

#define SSID_CHAR_UUID "ACA0EF7C-EEAA-48AD-9508-19A6CEF6B356" //wifi ssid

#define PASSWORD_CHAR_UUID "40B7DE33-93E4-4C8B-A876-D833B415A6CE"//wifi 密码

#define NOTIFY_CHAR_UUID "8AC32D3f-5CB9-4D44-BEC2-EE689169F626"//wifi 连接成功通知

#define NOTIFY_DESC_UUID "00002902-0000-1000-8000-00805f9b34fb" //配置 WIFI

Characteristic 特征值 dbus 总线接口是 org.bluez.GattCharacteristic1，对应的操作函数有 ReadValue，WriteValue，StartNotify()，StopNotify()

3.2 RTL8723DS BLE 配网命令

蓝牙成功开启后，RK3308 端执行 gatt-service 命令：

gatt-service //启动配网服务

手机端 apk：

1、点击选择设备（select a device），即可扫描出设备蓝牙，名字默认为 bluez-5.50

2、点击右上角的连接按钮，界面显示连接成功与否

3、连接成功后，填写要连接的 wifi ssid 和 password，点击下面的 Connect 即可

3.3 GATT-SERVICE API 介绍

1、创建服务：service_path = register_service(WIFI_SERVICES_UUID);

2、添加 characteristic descriptor

```
static gboolean register_characteristic(const char *chr_uuid,          //characteristic  uuid
                                        const uint8_t *value, int vlen,          //data
                                        const char **props,          //characteristic props
                                        const char *desc_uuid,          //descriptor uuid
                                        const char **desc_props,          //descriptor props
                                        const char *service_path)          //services_path
```

characteristic 读写方法位于 chr_methods 中：

```
static DBusMessage *chr_read_value()
```

static DBusMessage *chr_write_value()

descriptor 读写方法位于 desc_methods 中

static DBusMessage *desc_read_value()

static DBusMessage *desc_write_value()

3、发送广播接口：static void send_advertise()

（1）发送随机地址，自动生成

//LE Set Random Address Command

execute(CMD_RA, buff);

（2）发送广播数据，数据格式可按需修改

// LE Set Advertising Data Command

execute(CMD_ADV_DATA, buff);

数据格式：由 flag+uuid+name 组成，uuid 和 name 可修改

SERVICES_UUID（注意格式）

#define SERVICES_UUID "23 20 56 7c 05 cf 6e b4 c3 41 77 28 51 82 7e 1b"

BT_NAME 自定义不超过 8 字符

#define BT_NAME "GATT"

（3）使能广播

// LE Set Advertise Enable Command

execute(CMD_EN, buff);

4、配网

接收到 APP 下发的 SSID 和 PASSWORD 后，调用以下接口进行连网：

static gboolean config_wifi()

static bool saveWifiConfig()

wifi 的加密方式默认为 WPA-PSK，若 APP 下发加密方式，则保存在 SECURITY_CHAR_UUID 的 data 中，获取后加入到 config_wifi()中即可。

5、发送通告消息。

配网后，检测 WIFI 是否连接成功，并将状态写到 NOTIFY_CHAR_UUID 的 data 数据中，BLUEZ 会自动发送通告消息。

static gboolean check_wifi_isconnected();

chr_write(temp_chr, &level, sizeof(level));

4 WIFI SOFTAP 配网

简介：首先，用 SDK 板的 WiFi 创建一个 AP 热点，在手机端连接该 AP 热点；其次，通过手机端 apk 获取 SDK 板的当前扫描到的热点列表，在手机端填入要连接 AP 的密码，apk 会把 AP 的 ssid 和密码发到 SDK 端；最后，SDK 端会根据收到的信息连接 WiFi。

Buildroot 配置：

```
There is no help available for this option.
Symbol: BR2_PACKAGE_SOFTAPSERVER [=y]
Type  : boolean
Prompt: socket server based on softap
  Location:
    -> Target packages
      -> rockchip BSP packages (BR2_PACKAGE_ROCKCHIP [=y])
  Defined at package/rockchip/softapServer/Config.in:1
  Depends on: BR2_PACKAGE_ROCKCHIP [=y]
  Selects: BR2_PACKAGE_SOFTAP [=y]
```

源码开发目录：

/external/softapServer/ -- WIFI 与 APK 端相关操作

/external/softapDemo/ -- WiFi 相关操作

external/app/RkEcho.apk --softap 配网 apk

第一步：RK3308 端的执行如下命令：

softapServer Rockchip-Echo-123 (wifi 热点的名字，**前缀必须为 Rockchip-Echo-xxx**)

```
/ # softapServer Rockchip-Echo-123
DEBUG 263: check_wifi_chip_type_string: AP6255DEBUG 274:
wifi type: AP6255
DEBUG 297: start softap with name: Rockchip-Echo-123---DEBUG 30: cmdline = killall dnsmasq
killall: dnsmasq: no process killed
DEBUG 30: cmdline = killall hostapd
killall: hostapd: no process killed
DEBUG 30: cmdline = ifconfig wlan1 down
DEBUG 30: cmdline = rm -rf /data/bin/wlan1
DEBUG 30: cmdline = iw dev wlan1 del
DEBUG 30: cmdline = ifconfig wlan0 up
DEBUG 30: cmdline = iw phy0 interface add wlan1 type managed
```

第二步：打开手机的 wifi setting 界面：找到 Rockchip-Echo-123，点击连接

第三步：打开手机 RkEcho.apk：点击 wifi setup->CONFIRM->确认->wifi 列表->点击你要连接的网络名字->输入密码->点击确认

注意要点：

1、softspServer Rockchip-Echo-123 执行后命令行是无法退出的，直到配网完成；

2、名字千万不要写错，否则 apk 无法进入确认界面（Rockchip-Echo-xxx）

5 BT 开关

键入如下命令，打开蓝牙并设置蓝牙名字，如下命令客户可以自行集成到脚本直接开机运行即可。

echo 0 > sys/class/rfkill/rfkill0/state　//蓝牙 powe 下电

echo 1 > sys/class/rfkill/rfkill0/state　//蓝牙 power 上电

insmod usr/lib/modules/hci_uart.ko //加载 8723ds 蓝牙 uart 驱动

rtk_hciattach -n -s 115200 /dev/ttyS4 rtk_h5 & //　下载蓝牙 FW 和 config,并指定使用的协议 H5

usr/libexec/bluetooth/bluetoothd --compat -n &　//启动蓝牙协议栈，如果要查看 debug log，添加-d

hciconfig hci0 up　//蓝牙 hci0 设备 up

hciconfig hci0 piscan //蓝牙支持 PSCAN　ISCAN

hciconfig hci0 class 0x240404　//设置蓝牙 COD 为蓝牙音箱或者外联设备

hciconfig hci0 name 'rk-bt-12345' //设置蓝牙设备名字

命令执行成功键入 hciconfig –a 查看蓝牙 info，正常开启蓝牙后会有如下信息

```
/ #
/ # hciconfig -a
hci0:   Type: BR/EDR  Bus: UART
        BD Address: 00:E0:4C:23:99:87  ACL MTU: 1021:8  SCO MTU: 255:12
        UP RUNNING PSCAN ISCAN
        RX bytes:34684 acl:175 sco:0 events:1190 errors:0
        TX bytes:31582 acl:144 sco:0 commands:591 errors:0
        Features: 0xff 0xff 0xff 0xfe 0xdb 0xfd 0x7b 0x87
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH HOLD SNIFF PARK
        Link mode: SLAVE ACCEPT
        Name: 'rk-bt-12345'
        Class: 0x240404
        Service Classes: Rendering, Audio
        Device Class: Audio/Video, Device conforms to the Headset profile
        HCI Version: 4.1 (0x7)  Revision: 0xaa7a
        LMP Version: 4.1 (0x7)  Subversion: 0x8cb2
        Manufacturer: Realtek Semiconductor Corporation (93)
```

RK3308 平台已经集成了如上命令，运行 **bt_realtek_start** 命令后，即可开启，用户可以根据自己需要，集成这些命令，详见 **usr/bin/bt_realtek_start**

6 BT A2DP SINK

基于以上步骤，蓝牙已经正常打开，并且手机能够扫描到蓝牙设备，键入如下命令测试蓝牙 A2DP sink 测试蓝牙音频播放功能，首先配对手机配对蓝牙成功后，执行如下命令进行蓝牙播放。

bluealsa --profile=a2dp-sink & // 启动 bluealsa

bluealsa-aplay --profile-a2dp D8:1D:72:1C:26:DC //手机蓝牙 mac

bluealsa-aplay --profile-a2dp 00:00:00:00:00:00 //或者 00：xx：xx

执行成功后，会有如下正常打印， 表明系统已经开启了 bluealsa BT A2DP Profile

```
/ #  bluealsa --profile=a2dp-sink
bluealsa: ctl.c:548: Starting controller loop
bluealsa: bluez.c:677: Registering endpoint: /A2DP/SBC/Sink/1
bluealsa: main.c:281: Starting main dispatching loop
bluealsa: ctl.c:609: New client accepted: 10
bluealsa: ctl.c:630: +-+-
bluealsa: ctl.c:630: +-+-
bluealsa: ctl.c:630: +-+-
bluealsa: ctl.c:580: Client closed connection: 10
bluealsa: ctl.c:630: +-+-
bluealsa: ctl.c:609: New client accepted: 10
```

bluealsa-aplay --profile-a2dp D8:1D:72:1C:26:DC –>手机 BT MAC

或者 bluealsa-aplay --profile-a2dp 00:00:00:00:00:00 ->也可设置为 00

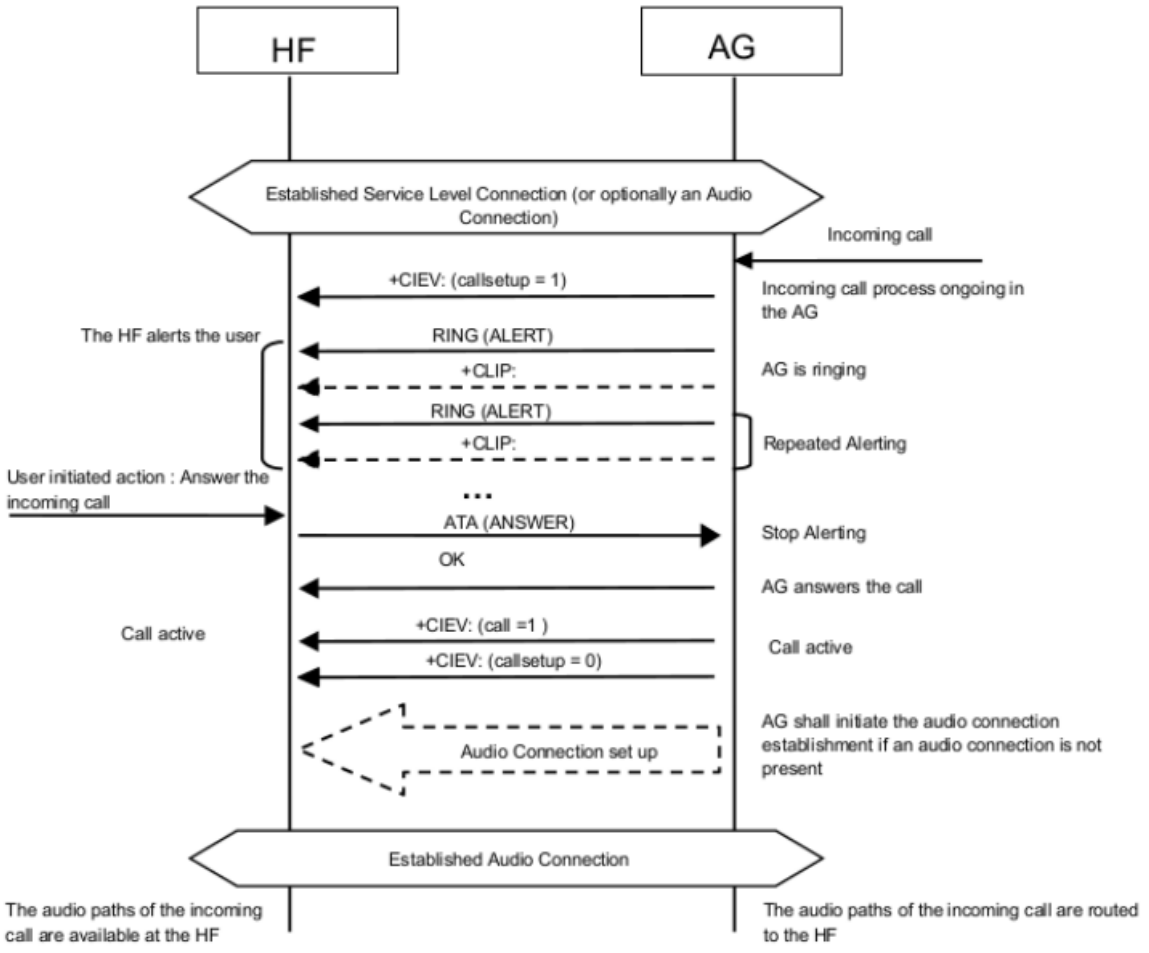
RTL8723DS a2dp sink，如果蓝牙 RF 指标和天线指标没有问题，蓝牙接收的数据包都会稳定在 6X/7X 水平，如果低于这个水平，会出现卡顿和声音不连续问题，请先调整蓝牙指标再进行测试。

```
rtk_btcoex: count_a2dp_packet_timeout: a2dp_packet_count 70
rtk_btcoex: count_a2dp_packet_timeout: a2dp_packet_count 69
rtk_btcoex: count_a2dp_packet_timeout: a2dp_packet_count 69
rtk_btcoex: count_a2dp_packet_timeout: a2dp_packet_count 69
rtk_btcoex: count_a2dp_packet_timeout: a2dp_packet_count 69
```

7 BT HFP SCO、PCM

7.1 HFP 电话呼入接通原理简要介绍

电话的呼入、接通的消息序列图如下：(来自于 HFP 文档)



当有电话呼入时，HF 端收到+CIEV: (setup = 1)命令，rfcomm_handler_ciev_resp_cb()被调用，indication 为 HFP_IND_CALLSETUP, value 为 1。

电话呼入时，HF 端还会收到 RING (ALERT)命令。目前 bluealsa 不处理这个 Command。如果希望在收到 RING (ALERT)时接听电话，则需要在 rfcomm_get_callback()的 handers 末尾添加 &rfcomm_handler_resp_ok，同时修改 rfcomm_handler_resp_ok_cb()，添加处理 RING 的代码，如：

```
if (strcmp(at->value, "RING") == 0) {
        debug("received RING");
```

return 0;

}

当电话接通时，HF 会接连收到+CIEV: (call = 1)和+CIEV: (callsetup = 0)，处理函数依然是

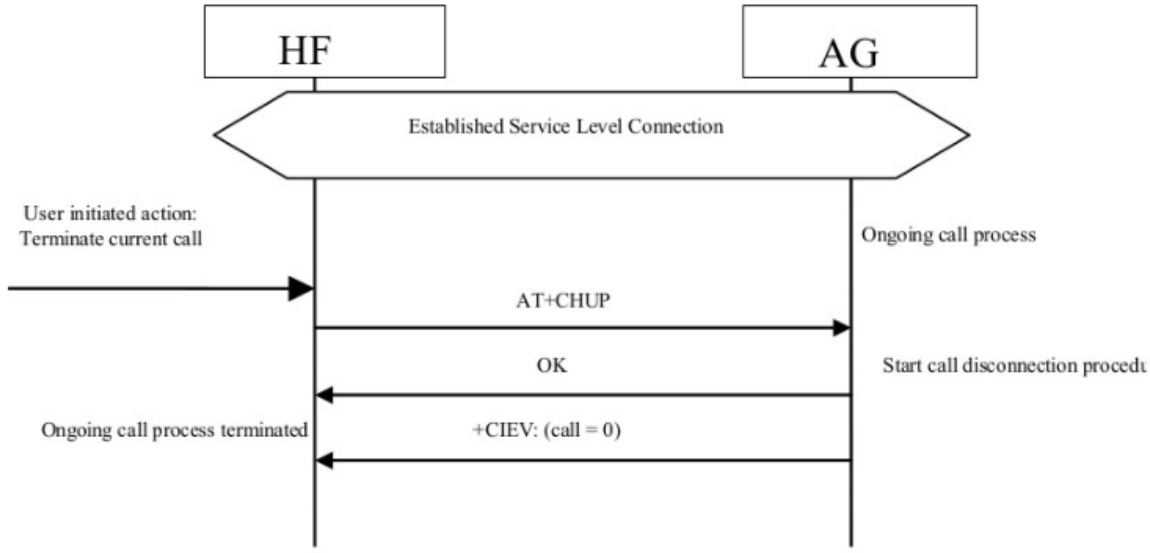
rfcomm_handler_ciev_resp_cb()

当电话挂断时，HF 会收到+CIEV: (callsetup = 0)或者+CIEV: (call = 0)。处理函数也是

rfcomm_handler_ciev_resp_cb()

接听电话如前面的消息序列图所示，HF 端发送 ATA Command 即可，

来电时挂断的消息序列图如下，HF 使用的 Command 是 AT+CHUP (REJECT)



7.2 RK 平台 HFP 测试 DEMO 介绍

HFP 功能测试，需要先开启 HFP profile，注意 HFP 和上面提到的 A2DP sink 不能同时使用，使用其他 profile 需要关闭先前开启的 profile ，否则，蓝牙工作在错误的 profile 模式。

Ps 查看，busysbox killall bluealsa 即可关闭先前开启的 profile

执行 **bluealsa -p hfp-hf & 开启 hfp profile** ,正常开启后会有如下 LOG

```
/ # ./data/bluealsa -p hfp-hf
./data/bluealsa: ctl.c:548: Starting controller loop
./data/bluealsa: bluez.c:915: Registering profile: /HFP/HandsFree
./data/bluealsa: main.c:281: Starting main dispatching loop
./data/bluealsa: bluez.c:862: Profile method call: org.bluez.Profile1.NewConnection()
./data/bluealsa: transport.c:690: State transition: 0 -> 2
./data/bluealsa: io.c:1173: Starting IO loop: HFP Hands-Free
./data/bluealsa: transport.c:104: Created new IO thread: HFP Hands-Free
./data/bluealsa: bluez.c:802: HFP Hands-Free configured for device D8:1D:72:1C:26:DC
./data/bluealsa: transport.c:690: State transition: 0 -> 2
./data/bluealsa: ctl.c:630: +-+-
```

开启 HFP profile 后，即可进行测试了，目前 8723DS HFP 支持 SCO over HCI UART 和 SCO over PCM 两种方式。

说明：客户 APP 只需按照 bluealsa-rfcomm bin 文件中的 bluealsa_send_rfcomm_command 标准 API 既可以实现 AT 命令的下发，具体测试见如下描：

使用 bluealsa /utils 目录下的 bluealsa-rfcomm 工具测试挂断电话：

echo AT+CHUP | bluealsa-rfcomm --hci=hci0 $(hcitool con | sed -n 2p |busybox tr -d "\n" | busybox cut -b 7-24)

说明：hcitool con | sed -n 2p |busybox tr -d "\n" | busybox cut -b 7-24 获取手机蓝牙 MAC 地址

echo **AT+CHUP** | bluealsa-rfcomm --hci=hci0 $(hcitool con | sed -n 2p |busybox tr -d "\n" | busybox cut -b 8-25)

测试接通电话：

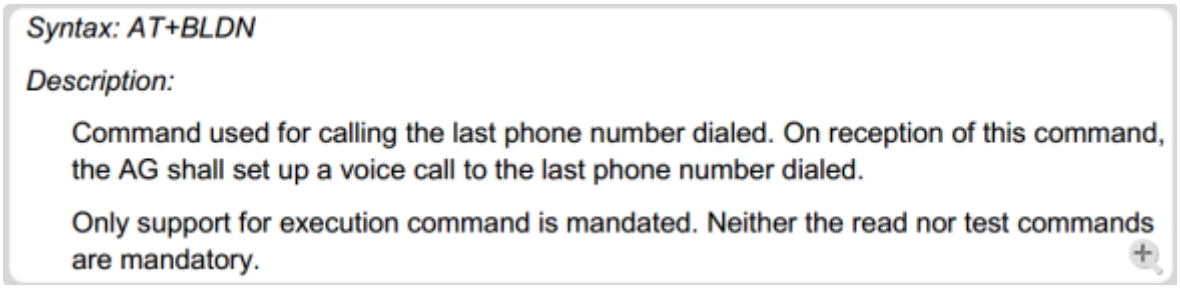
echo **ATA** | bluealsa-rfcomm --hci=hci0 $(hcitool con | sed -n 2p |busybox tr -d "\n" | busybox cut -b 8-25)

测试拨打电话：ATD 指令+需要拨打的手机号码，格式为： ATD156xxxxxxx

echo **ATD156xxxxxxx** | bluealsa-rfcomm --hci=hci0 $(hcitool con | sed -n 2p |busybox tr -d "\n" | busybox cut -b 8-25)

拨打手机通讯记录里面最近一次通话的电话号码：

echo **AT+BLDN** | bluealsa-rfcomm --hci=hci0 $(hcitool con | sed -n 2p |busybox tr -d "\n" | busybox cut -b 8-25)



注意，根据 HFP spec 来看，如下红色方框内的 3 个打电话命令 AG 都要支持，但是有的手机可能是不支持的，实测 iphone 6 不支持 place a call with a phone number，也就是不支持通过 HF 打电话功能，关于 SIG 发布的 HFP 文档客户自行网上下载即可。

	Feature	Support in HF	Support in AG
1.	Connection management	M	M
2.	Phone status information	M(note 1)	M
3.	Audio Connection handling	M	M
4	Accept an incoming voice call	M	M
5.	Reject an incoming voice call	M	O
6.	Terminate a call	M	M
7.	Audio Connection transfer during an ongoing call	M	M
8.	Place a call with a phone number supplied by the HF	O	M
9.	Place a call using memory dialing	O	M
10.	Place a call to the last number dialed	O	M

bluealsa-rfcomm 核心 API，int bluealsa_send_rfcomm_command(int fd, bdaddr_t addr, const char *command)，该 API 通过 rfcomm 下发 AT common 统一接口，处理 AT commd 的最终函数为 static int rfcomm_write_at(int fd, enum bt_at_type type, const char *command,

const char *value)

```
/**
 * Write AT message.
 *
 * @param fd RFCOMM socket file descriptor.
 * @param type Type of the AT message.
 * @param command AT command or response code.
 * @param value AT value or NULL if not applicable.
 * @return On success this function returns 0. Otherwise, -1 is returned and
 *   errno is set to indicate the error. */
static int rfcomm_write_at(int fd, enum bt_at_type type, const char *command,
                const char *value) {

        char msg[256];
        size_t len;

        debug("Sending AT message: %s: command:%s, value:%s",
                        at_type2str(type), command, value);

        at_build(msg, type, command, value);
        len = strlen(msg);

retry:
        if (write(fd, msg, len) == -1) {
                if (errno == EINTR)
                        goto retry;
                return -1;
        }
```

接听电话的 Command 是 ATA，挂断电话的 Command 是 AT+CHUP，Command 通过 rfcomm_write_at()发送给 AG，具体实现函数 bluealsa/src/rfcomm.c，发送 ATA Command 的代码是 **rfcomm_write_at(fd, AT_TYPE_CMD, “ATA”, NULL)**

发送 AT+CHUP 的代码是 **rfcomm_write_at(fd, AT_TYPE_CMD, “+CHUP”, NULL)**

其中 fd 为 ba_fd = bluealsa_open(ba_interface)) ba_interface 就是 hci0

说明：原生态 rfcomm.c 没有添加 AT+CHUP 命令，开发 APP 的时候可以自行添加，添加后执行效果跟上面提到的 echo 命令实现功能一样。比如客户需要执行按键接听或者语音控制进行接听，实现 API 均可基于这个 API。电话接通后，要测试是否能听到音响和手机端的声音，执行如下操作即可，这部分内容需要整合到系统 alsa 音频通路框架里面(蓝牙已经可以传输 PCM 音频数据)，alsa config 文件需要根据实际进行调整音频通路。

1. **SCO over HCI UART**

   a) bluealsa-aplay -v --profile-sco bt_mac（配对手机的蓝牙 MAC 地址，格式为 D8:1D:72:1C:26:DC） // 音箱端播放远程电话的声音

   b) arecord -r 8000 -f S16_LE -c 2|aplay -D bluealsa:HCI=hci0,DEV= bt mac,PROFILE=sco -t raw -r 8000 -f S16_LE -c 2 // 将音箱端的录音推送到远程电话

   // 如果 b)不通，可以尝试 c)将音箱本地的 wav 推送到远程电话测试：

   c) aplay -D bluealsa:HCI=hci0,DEV=bt_mac,PROFILE=sco /data/test /Front_Right.wav

如果觉得输入 MAC 地址比较麻烦，可以通过如下命令直接嵌入到命令，如下：

获取配对设备的 MAC 地址：键入如下命令：

hcitool con | sed -n 2p |busybox tr -d "\n" | busybox cut -b 7-24

bluealsa-aplay -v --profile-sco $(hcitool con | sed -n 2p |busybox tr -d "\n" | busybox cut -b 7-24)

同理，有用到蓝牙 MAC 的地方，都可以通过这个方式快速的进行测试和使用

2. **SCO over PCM**

   arecord -Dhw:1,0 -r 8000 -c 2 -f S16_LE |aplay -Dhw:0,0 -r 8000 -c 2 -f S16_LE // 音箱端播放远程电话的声音

   arecord -Dhw:0,0 -r 8000 -f S16_LE -c 2 |aplay -D hw:0,1 -r 8000 -c 2 -f S16_LE // 将音箱端的录音推送到远程电话

3. RK3308 平台已经集成了这些命令，详见 usr/bin/bt_realtek_hfp_start 和 usr/bin/bt_realtek_hfp_hanup，直接运行/bt_realtek_hfp_start 和 usr/bin/bt_realtek_hfp_hanup 即可接通和挂断电话

8 BT AVRCP 控制

8.1 AVRCP 实现原理简要说明

RK 蓝牙 A2DP sink 音频播放是基于 bluez5.x 协议栈，bluez5.x 协议栈中提供的 AVRCP CT 相关的 dbus 接口可以查看 doc/media-api.txt，interface 为 org.bluez.MediaControl1。下面简要介绍 bluez5 协议中 AVRCP 操作（包括 play，pause 等），实现在 profile/audio/control.c 文件中。 同理，根据 ATRCP 协议，在 TG 端也可以实现同样功能，build/bluez5_utils-5.39/tools/bluetooth-player.c 提供 AVRCP TG 暂停，播放，下一曲等测试功能，提供测试的功能如下：

```
cmd_table[] = {
        { "list",           NULL,       cmd_list, "List available players" },
        { "show",           "[player]", cmd_show, "Player information" },
        { "select",         "<player>", cmd_select, "Select default player" },
        { "play",           "[item]",   cmd_play, "Start playback" },
        { "pause",          NULL,       cmd_pause, "Pause playback" },
        { "stop",           NULL,       cmd_stop, "Stop playback" },
        { "next",           NULL,       cmd_next, "Jump to next item" },
        { "previous",       NULL,       cmd_previous, "Jump to previous item" },
        { "fast-forward",   NULL,       cmd_fast_forward,
                                        "Fast forward playback" },
        { "rewind",         NULL,       cmd_rewind, "Rewind playback" },
        { "equalizer",      "<on/off>", cmd_equalizer,
                                        "Enable/Disable equalizer"},
        { "repeat",         "<singletrack/alltrack/group/off>", cmd_repeat,
                                        "Set repeat mode"},
        { "shuffle",        "<alltracks/group/off>", cmd_shuffle,
                                        "Set shuffle mode"},
        { "scan",           "<alltracks/group/off>", cmd_scan,
                                        "Set scan mode"},
        { "change-folder", "<item>",  cmd_change_folder,
                                        "Change current folder" },
        { "list-items", "[start] [end]",  cmd_list_items,
                                        "List items of current folder" },
        { "search",     "string",     cmd_search,
                                        "Search items containing string" },
        { "queue",        "<item>",     cmd_queue, "Add item to playlist queue" },
        { "show-item",    "<item>",     cmd_show_item, "Show item information" },
        { "quit",           NULL,       cmd_quit, "Quit program" },
        { "exit",           NULL,       cmd_quit },
        { "help" },
        {}
```

这里只罗列关键 API，原生态 bluez 协议栈提供的 bluetooth-player 工具是基于标准的 dbug 接口（g_dbus_proxy_method_call）进行命令的处理。bluetooth-player 需要 buildroot 打开如下宏才会编译生成 bin 文件，如下所示：

```
BR2_PACKAGE_BLUEZ5_UTILS_CLIENT:

Enable the Bluez 5.x command line client.

Symbol: BR2_PACKAGE_BLUEZ5_UTILS_CLIENT [=y]
Type  : boolean
Prompt: build CLI client
  Location:
    -> Target packages
      -> Networking applications
        -> bluez-utils 5.x (BR2_PACKAGE_BLUEZ5_UTILS [=y])
  Defined at package/bluez5_utils/Config.in:42
  Depends on: BR2_PACKAGE_BLUEZ5_UTILS [=y]
  Selects: BR2_PACKAGE_READLINE [=y]
```

sbluetooth-player bin 文件位于如下目录：

buildroot/output/rockchip_rk3308_release/build/bluez5_utils-5.39/tools/ bluetooth-player

bluetooth-player demo 程序运行如下所示：

```
/userdata # ./bluetooth-player
[NEW] Player /org/bluez/hci0/dev_D8_1D_72_1C_26_DC/player2 [default]
[bluetooth]# play
Attempting to play
Play successful
[CHG] Player /org/bluez/hci0/dev_D8_1D_72_1C_26_DC/player2 Status: playing
[CHG] Player /org/bluez/hci0/dev_D8_1D_72_1C_26_DC/player2 Position: 0x0219d7
[bluetooth]#
```

AVRCP 控制，需要先获取到 GDBusProxy 代理的播放器接口，再调用标准 dbus 接口函数(g_dbus_proxy_method_call(default_player, "Pause", NULL, pause_reply, NULL, NULL)，比如要控制蓝牙 A2DP sink 音乐暂停，首先获取到当前蓝牙播放 default_player，调用 dbus 接口 g_dbus_proxy_method_call 函数实现蓝牙 A2DP 暂停功能，如下所示：（其他功能类似）

```
static void player_added(GDBusProxy *proxy)
{
        players = g_slist_append(players, proxy);
        printf("xxh3 player add\n");
        if (default_player == NULL) {
                rl_printf("xxh4 ayer_added4\n");
                default_player = proxy;
        }

        if (default_player ==NULL)
                printf("no player \n");
        if (g_dbus_proxy_method_call(default_player, "Pause", NULL,
                                pause_reply, NULL, NULL) == FALSE) {
                printf("Failed to play\n");
                //dbus_error_free(&buserror);

        }
}
```

```
static void cmd_pause(int argc, char *argv[])
{
        if (!check_default_player())
                return;

        if (g_dbus_proxy_method_call(default_player, "Pause", NULL,
                                pause_reply, NULL, NULL) == FALSE) {
                rl_printf("Failed to play\n");
                return;
        }

        rl_printf("Attempting to pause\n");
}
```

说明：由于 bluetooth-player.c 代码是集成在 bluez 协议栈里面，依赖 bus 相关库，所以，可以在 bluetooth-player.c 源码基础上重新封装改文件，便于系统其它地方使用或者 app 开发使用，本文 6.3 章节罗列了必要的 dbus 相关依赖库和 glib 依赖库，app 开发需要添加这些配置才能直接使用 dbug 接口进行控制，客户开发 APP 的时候可以自行封装或者参考 RK 提供的 DEMO 进行封装即可。RK 提供基于 Bluez5 AVRCP 封装，具体使用见本文 6.4 章节

8.2 libgdbus-internal 库的引用

在 lib 目录中放入 libgdbus-internal.a， **APP 应用需要引用** libgdbus-internal.a **静态库**

8.3 编译配置

在 Makefile 加入库的引用

-lglib-2.0 -ldbus-1 -ldbus-glib-1 -lreadline

CFLAGS 加入

$(shell pkg-config --libs dbus-glib-1) $(shell pkg-config --libs glib-2.0) $(shell pkg-config --cflags dbus-1)

8.4 Avrcpctrl API 说明

AVRCP 接口的封装详见 avrcpctrl.c 文件，APP 开发可以直接调用 avrcpctrl.c 中接口，RK 将该封装放到了 SDK external/rkwifibt/realtek/demo 目录，主要接口描述如下，：

```
/**
* 初始化 蓝牙音频反向控制模块
* 应用主程序起来需要先初始反向控制模块,以便于监听相关状态的变化
*/
int init_avrcp_ctrl();
/**
```

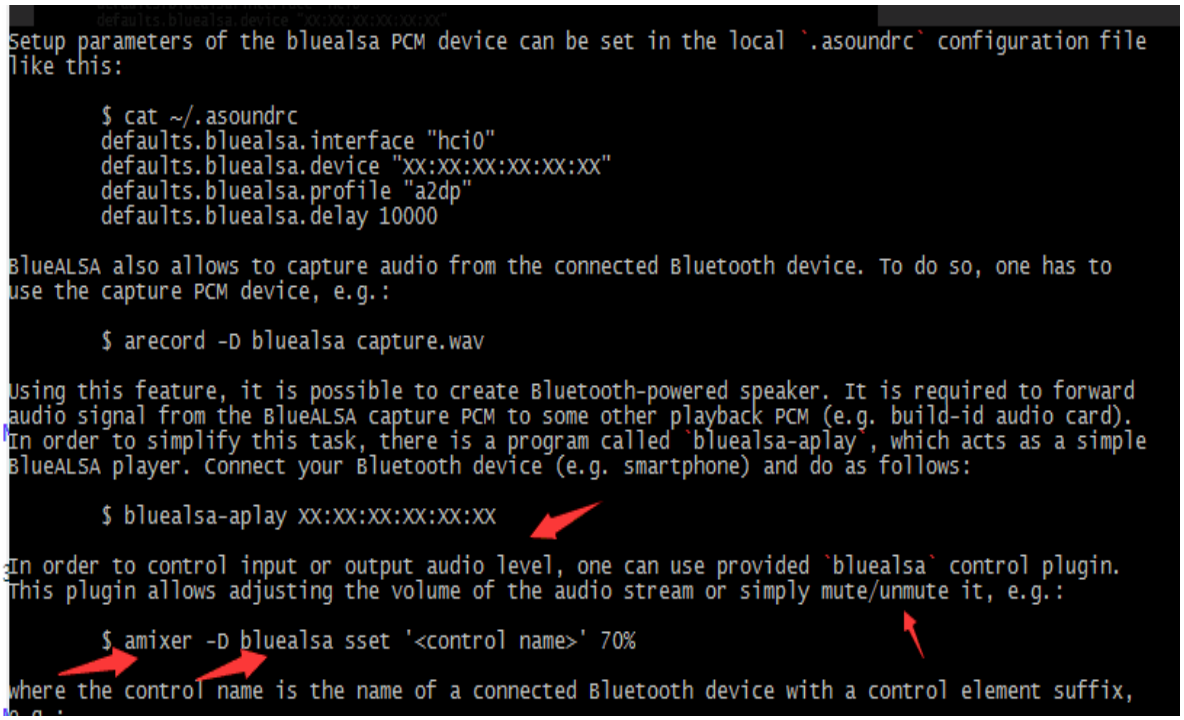
```
* 释放蓝牙音频反向控制相关资源
*/
int release_avrcp_ctrl();
/**
* 播放
*/
void play_avrcp();
 /**
* 暂停播放
*/
void pause_avrcp();
/**
* 停止播放
*/
void stop_avrcp();
/**
* 下一首
*/
void next_avrcp();
/**
* 上一首
*/
void previous_avrcp();
/**
* 快进
*/
```

```
void fast_forward_avrcp();

/**

*  重新播放

*/

void rewind_avrcp();

/**

*  获取当前蓝牙音频状态  A2DP sink  播放放状态定义

      //play status

      #define AVRCP_PLAY_STATUS_STOPPED      0x00 //  停止

      #define AVRCP_PLAY_STATUS_PLAYING      0x01 //正在播放

      #define AVRCP_PLAY_STATUS_PAUSED 0x02 //暂停播放

      #define AVRCP_PLAY_STATUS_FWD_SEEK   0x03 //快进

      #define AVRCP_PLAY_STATUS_REV_SEEK    0x04 //重播

      #define AVRCP_PLAY_STATUS_ERROR        0xFF //错误状态

*/

int getstatus_avrcp();
```

9 A2DP sink 音量调节

9.1 bluez-alsa org.bluez.MediaTransport1

目前原生态的协议栈没有封装音量控制接口，根据 bluez-alsa 官方说明文档，音量控制可以通过 amixer 工具来进行控制



命令格式如下：

其中 xxh 为连接手机的蓝牙名称，80%是控制音量大小(最大音量 100%，最小音量 0)

amixer -D bluealsa sset 'xxh - A2DP' 80%

脚本 Demo 如下：

```
/userdata # cat vomlum_up.sh
btname=$(hcitool name  $(hcitool con | sed -n 2p |busybox tr -d "\n" | busybox cut -b 7-24))
echo "btname is : $btname"
sleep 1
str2=" - A2DP"
str3=${btname}${str2}
echo "str3 is : $str3"
amixer -D bluealsa sset "$str3" 100%
```

amixer 控制 sink 端音量是通过如下流程最终使用 bluealsa_set_transport_volume()给 bluealsa 发消息，详细可见 bluealsa 如下代码：

io_thread_scale_pcm() –> bluealsa_write_integer-> bluealsa_set_transport_volume()

org.bluez.MediaTransport1 interface 提供 set_volum 方法，参考 transport_set_volume()函数

```
bluealsa: ctl.c:630: +-+-
bluealsa: transport.c:651: Setting volume for D8:1D:72:1C:26:DC profile 2: 102<>127 [OO]
bluealsa: ctl.c:630: +-+-
bluealsa: transport.c:651: Setting volume for D8:1D:72:1C:26:DC profile 2: 102<>127 [OO]
bluealsa: bluez.c:1037: Signal: PropertiesChanged: org.bluez.MediaTransport1
bluealsa: ctl.c:630: +-+-
bluealsa: transport.c:651: Setting volume for D8:1D:72:1C:26:DC profile 2: 102<>102 [OO]
bluealsa: ctl.c:630: +-+-
bluealsa: transport.c:651: Setting volume for D8:1D:72:1C:26:DC profile 2: 102<>102 [OO]
bluealsa: ctl.c:630: +-+-
bluealsa: ctl.c:580: Client closed connection: 15
bluealsa: ctl.c:630: +-+-
bluealsa: ctl.c:609: New client accepted: 15
bluealsa: ctl.c:630: +-+-
bluealsa: ctl.c:630: +-+-
bluealsa: ctl.c:630: +-+-
bluealsa: ctl.c:630: +-+-
bluealsa: ctl.c:630: +-+-
bluealsa: ctl.c:630: +-+-
bluealsa: transport.c:651: Setting volume for D8:1D:72:1C:26:DC profile 2: 127<>102 [OO]
bluealsa: ctl.c:630: +-+-
bluealsa: transport.c:651: Setting volume for D8:1D:72:1C:26:DC profile 2: 127<>102 [OO]
bluealsa: ctl.c:630: +-+-
bluealsa: transport.c:651: Setting volume for D8:1D:72:1C:26:DC profile 2: 127<>127 [OO]
bluealsa: ctl.c:630: +-+-
bluealsa: bluez.c:1037: Signal: PropertiesChanged: org.bluez.MediaTransport1
bluealsa: transport.c:651: Setting volume for D8:1D:72:1C:26:DC profile 2: 127<>127 [OO]
bluealsa: ctl.c:630: +-+-
```

9.2 直接控制喇叭音量

使用 amixer controls 工具查看系统所有的 mixer simple controls，建议采用这个方式直接控制 sperker 音量，7.1 章节目前看看 bluez-alse 提供的 io_thread_scale_pcm 算法是直接调整蓝牙 A2DP SINK 音频的参数，实际应用场景不多，**建议直接使用 amixer 调解喇叭音量**

```
/userdata # amixer controls
numid=47,iface=MIXER,name='Master Playback Volume'
numid=33,iface=MIXER,name='ADC ALC Group 0 Left Volume'
numid=34,iface=MIXER,name='ADC ALC Group 0 Right Volume'
numid=35,iface=MIXER,name='ADC ALC Group 1 Left Volume'
numid=36,iface=MIXER,name='ADC ALC Group 1 Right Volume'
numid=37,iface=MIXER,name='ADC ALC Group 2 Left Volume'
numid=38,iface=MIXER,name='ADC ALC Group 2 Right Volume'
numid=39,iface=MIXER,name='ADC ALC Group 3 Left Volume'
numid=40,iface=MIXER,name='ADC ALC Group 3 Right Volume'
```

amixer cset numid=47,iface=MIXER,name='Master Playback Volume' 40 调节喇叭声音

10 Wifi 无法打开问题排查

如果发现 WIFI 无法扫描到 SDIO 或者无法扫描 AP，请先确认 WIFI_PWR 脚配置和第一章节提到的 LDO_SPS_SEL 章节描述