

Realtek Bluetooth Porting Guide for Android6.x

v0.3

2017/03/01

修订历史 (Revision History)

日期	版本	修改	作者
2015/12/1	v0.1	初稿	毛为锋
2016/04/20	v0.2	v0.2	毛为锋
2017/03/02	V0.3	Move patch/code/linux to linux	毛为锋

Realtek

Realtek

目 录

修订历史 (Revision History)	2
目 录	4
表格目录	6
图表目录	7
第 1 章 文档适用范围	8
第 2 章 Release 说明	9
2.1 Release 文件说明	9
2.2 Porting Guide 说明	9
第 3 章 Porting 方法	11
3.1 Android 部分	11
3.1.1 Changelist	11
3.1.2 build	11
3.1.3 device/{vendor}/{product}/	12
3.1.4 frameworks/base	19
3.1.5 packages/app/Bluetooth	19
3.1.6 其他部分	19
3.2 Kernel 部分	20
3.2.1 Change list	20
3.2.2 rtk_btusb driver	21
3.2.3 rtk_rfkill driver	21
3.2.4 TUN driver	22
3.2.5 UINPUT driver	22
3.2.6 HID driver	22
第 4 章 BT 功能配置	23
4.1 支持的 Profile 配置	23
4.2 配置本地设备名称	23
4.3 配置本地设备 COD	23

4.4 配置 Extra Config 文件	24
第 5 章 Porting 结束后的基本测试	25
5.1 配置检查	25
5.2 BT 测试	25
5.2.1 基本功能测试	25
第 6 章 蓝牙问题报告	26
6.1 蓝牙 Log 相关的配置文件	26
6.2 Logcat -v time	27
6.3 BtSnoop	27
6.4 注意事项	27
第 7 章 常见问题分析	28
7.1 BT 打开失败(UART)	28
7.2 BT 打开失败(USB)	28

表格目录

未找到图形项目表。

Realtek

图表目录

未找到图形项目表。

Realtek

第1章 文档适用范围

本文档适用于 Realtek Bluetooth Solution for Android 6.x 产品。

Realtek

第2章 Release 说明

2.1 Release 文件说明

Release 由 linux,rtkbt,patch 目录和 Porting Guide, ReleaseNote 文件组成。

- Linux 是 Realtek 提供的 Linux Kernel 驱动文件, 仅用于 USB 接口的蓝牙卡片。
- rtkbt 是 Realtek 发布的主要驱动文件,请直接放入 device/{vendor}/{product}/rtkbt 目录即可
- patch 是 Android 补丁文件,里面包括 code 和 diff 两部分,code 包含修改后的文件,diff 目录是修改的差异部分.由于基于的原始文件会有差异,请根据实际情况合并,切勿直接覆盖.
- Porting Guide 也就是本文件, 主要说明 Realtek 驱动包的使用方法。
- ReleaseNote 列出驱动包的主要修改。

2.2 Porting Guide 说明

- 本文中所有 code 的修改都遵循如下格式：
 - 1) 所有修改的 code 都使用方框括起来。
 - 2) 方框中的 code 会使用 SDK 中没有颜色的原有 code 给出修改位置的信息。
 - 3) 所有修改或添加的 code 均使用灰色突出显示。

对 code 格式的来举例说明：

如下 code 是 SDK 中原有的 code：

```

ifeq ($(BLUETOOTH_HCI_USE_MCT),true)
LOCAL_CFLAGS := -DHCI_USE_MCT
LOCAL_SRC_FILES += \
    src/hci_mct.c \
    src/serial_mct.c
else
LOCAL_SRC_FILES += \
    src/hci_h4.c \
    src/serial.c
endif
    
```

Realtek 需要添加 H5 的支援，则在 porting guide 中会给出如下 code：

```

ifeq ($(BLUETOOTH_HCI_USE_MCT),true)
LOCAL_CFLAGS := -DHCI_USE_MCT
LOCAL_SRC_FILES += \
    src/hci_mct.c \
    src/serial_mct.c
else
ifeq ($(BLUETOOTH_HCI_USE_RTK_H5),true)
LOCAL_CFLAGS := -DHCI_USE_RTK_H5
LOCAL_SRC_FILES += \
    src/hci_h5.c \
    src/serial.c \
    src/bt_skbuff.c \
    src/bt_list.c
else
LOCAL_SRC_FILES += \
    src/hci_h4.c \
    src/serial.c
endif
endif

```

灰色部分 code 是 Realtek 所添加以及修改的 code。

- 该文档中对应的厂商以及平台分别标识为{vendor}和{product}，不同的客户会对应到不同的厂商以及平台，请客户在 porting 时注意修改到对应平台的文件。例如在 1195 平台该文档{vendor}对应 realtek，{product}对应 phoenix。

第3章 Porting 方法

为了方便客户整合 Realtek 的 WIFI/BT Combo Chip 到自己的平台中, Realtek 对于如何整合 BT 的 Driver 到客户平台分成 Android 和 Linux 两部分说明.

3.1 Android 部分

3.1.1 Changelist

- 1) build
- 2) device/{vendor}/{product}/
- 3) frameworks/base
- 4) packages/app/Bluetooth
- 5) 其他部分

3.1.2 build

- 1) build\core\product.mk

```
_product_stash_var_list += \  
    BOARD_WPA_SUPPLICANT_DRIVER \  
    BOARD_WLAN_DEVICE \  
    BOARD_USES_GENERIC_AUDIO \  
    BOARD_KERNEL_CMDLINE \  
    BOARD_KERNEL_BASE \  
    BOARD_HAVE_BLUETOOTH \  
    BOARD_HAVE_BLUETOOTH_BCM \  

```

```
BOARD_HAVE_BLUETOOTH_QCOM \

BOARD_HAVE_BLUETOOTH_RTK \

BOARD_VENDOR_QCOM_AMSS_VERSION \

BOARD_VENDOR_USE_AKMD \

BOARD_EGL_CFG \

BOARD_BOOTIMAGE_PARTITION_SIZE \

...
```

添加 Realtek BT Chip 支持的宏定义。

3.1.3 device/{vendor}/{product}/

该目录主要用于设定不同硬件平台 board 配置信息。不同的子目录对应不同的硬件平台，需要根据具体硬件平台进行修改。

- 1) 将 Release 包中 rtkbt 目录拷贝到 device/{vendor}/{product}/下。

如果是 UART 接口蓝牙芯片,请修改 device/{vendor}/{product}/rtkbt/system/etc/bluetooth/rtkbt.conf

将 BtDeviceNode 设置为正确蓝牙串口设备.例如 BtDeviceNode=/dev/ttyS1

注意:Realtek 驱动默认会将 UART 波特率切换为 1.5M.如果使用的是 Amlogic 平台,建议使用 2M 波特率.使用方法:

请将 rtkbt/system/etc/firmware/UART_2M/下的文件覆盖到 rtkbt/system/etc/firmware/目录下同名文件.

如果是 USB 接口蓝牙芯片,请保持 BtDeviceNode=/dev/rtk_btusb 不变.

- 2) 添加设备权限.

以使用的是 USB 蓝牙, BtDeviceNode 设置为/dev/rtk_btusb 举例.

- 在 device 目录下有个 file_contexts 文件, 在该文件中添加

```
/dev/rtk_btusb          u:object_r:rtk_bt_device:s0
```

- 在 device.te 中添加

```
type rtk_bt_device, dev_type;
```

- device 目录下如果有 bluetooth.te 的话（没有就去 external/sepolicy 目录下去找），就在该文件中添加

```
allow bluetooth rtk_bt_device:chr_file rw_file_perms;
```

3) PRODUCT_MAKEFILES

首先请查看 **device/{vendor}/{product}/AndroidProducts.mk**，查找到正确的 PRODUCT_MAKEFILES，例如：

```
.....

PRODUCT_MAKEFILES := \

$(LOCAL_DIR)/ rtk_phoenix.mk

.....
```

这样可以找到 PRODUCT_MAKEFILES 为 rtk_phoenix.mk，然后在 rtk_phoenix.mk 中进行下面的修改：

```
.....

$(call inherit-product, $(SRC_TARGET_DIR)/product/aosp_base.mk)

$(call inherit-product, device/realtek/phoenix/device.mk)

#rtkbt

$(call inherit-product, device/{vendor}/{product}/rtkbt/rtkbt.mk)

PRODUCT_BRAND   := realtek

PRODUCT_NAME    := rtk_phoenix

PRODUCT_DEVICE  := phoenix

.....
```

注意: 请根据实际情况替换{vendor}与{product}。

4) 修改 device/{vendor}/{product}/rtkbt/Android.mk

由于 device/{vendor}/可能存在多个 product 使用 realtek 蓝牙芯片,为了降低不同产品之间的耦合性,我们建议在 Android.mk 增加对产品的判断.例如在编译 Android 时执行命令 `lunch` 命令结果如下:

```
$ lunch rtk_phoenix-eng

..... <省略部分>

PLATFORM_VERSION=5.1.1

TARGET_PRODUCT=rtk_phoenix

TARGET_BUILD_VARIANT=eng

.....<省略部分>
```

我们建议在 Android.mk 中增加下面的修改.

```
LOCAL_PATH := $(call my-dir)

ifeq $(TARGET_PRODUCT), rtk_phoenix)

ifeq $(BOARD_HAVE_BLUETOOTH_RTK),true)

include $(call all-subdir-makefiles)

endif

endif
```

5) device/{vendor}/{product}/init.{product}.rc

注意: 在添加过程中, 请务必注意不要有重复项, 如果原始文件中有相同内容, 请删除相关内容后再添加。

注意: 请根据实际情况替换{vendor}与{product}。

- 如果该产品采用的是 USB 蓝牙, 请将下列的所有项目添加到 device/{vendor}/{product}/init.{product}.rc。

on boot

.....

```
# bluetooth
```

```
# change back to bluetooth from system
```

```
chown bluetooth net_bt_stack /data/misc/bluetooth
```

```
mkdir /data/misc/bluedroid 0770 bluetooth net_bt_stack
```

```
# bluetooth LPM
```

```
chown bluetooth net_bt_stack /proc/bluetooth/sleep/lpm
```

```
chown bluetooth net_bt_stack /proc/bluetooth/sleep/btwrite
```

```
#USB device
```

```
insmod /system/lib/modules/rtk_btusb.ko
```

```
chmod 0660 /dev/rtk_btusb
```

```
chown bluetooth net_bt_stack /dev/rtk_btusb
```

```
# rfkill
```

```
chmod 0660 /sys/class/rfkill/rfkill0/state
```

```
chmod 0660 /sys/class/rfkill/rfkill0/type
```

```
chown bluetooth net_bt_stack /sys/class/rfkill/rfkill0/state
```

```
chown bluetooth net_bt_stack /sys/class/rfkill/rfkill0/type
```

```
# bluetooth MAC address programming
```

```
chown bluetooth net_bt_stack ro.bt.bdaddr_path
```

```
chown bluetooth net_bt_stack /system/etc/bluetooth
```

```
chown bluetooth net_bt_stack /data/misc/bluetooth
```

```
setprop ro.bt.bdaddr_path "/data/misc/bluetooth/bdaddr"
```

```
service dhcpcd_bnep0 /system/bin/dhcpcd -BKLG
```

```
disabled
```

```
oneshot
```

```
service dhcpcd_bnep1 /system/bin/dhcpcd -BKLG
```

```
disabled
```

```
oneshot
```

```
service dhcpcd_bnep2 /system/bin/dhcpcd -BKLG
```

```
disabled
```

```
oneshot
```

```
service dhcpcd_bnep3 /system/bin/dhcpcd -BKLG
```

```
disabled
```

```
oneshot
```

```
service dhcpcd_bnep4 /system/bin/dhcpcd -BKLG
```

```
disabled
```

```
oneshot
```

```
service dhcpcd_bt-pan /system/bin/dhcpcd -ABKL
```

```
class main
```

```
disabled
```



```
oneshot
```

```
service iprenw_bt-pan /system/bin/dhccpd -n
```

```
class main
```

```
disabled
```

```
oneshot
```

- 如果该产品采用的是 UART 蓝牙，将下面的所有项目添加到 device/{vendor}/{product}/init.{product}.rc。

注意: {UART 设备节点} 代表着蓝牙接口的设备节点,例如/dev/ttyS1, /dev/ttyS2...等

chmod 0660 {UART 设备节点} 真正应用场合可以修改为 chmod 0660 /dev/ttyS1

```
on boot
```

```
.....
```

```
# bluetooth
```

```
# change back to bluetooth from system
```

```
chown bluetooth net_bt_stack /data/misc/bluetooth
```

```
mkdir /data/misc/bluedroid 0770 bluetooth net_bt_stack
```

```
# bluetooth LPM
```

```
chown bluetooth net_bt_stack /proc/bluetooth/sleep/lpm
```

```
chown bluetooth net_bt_stack /proc/bluetooth/sleep/btwrite
```

```
#UART device
```

```
chmod 0660 {UART 设备节点}
```

```
chown bluetooth net_bt_stack {UART 设备节点}
```

```
# rfkill
```

```
chmod 0660 /sys/class/rfkill/rfkill0/state
```

```
chmod 0660 /sys/class/rfkill/rfkill0/type

chown bluetooth net_bt_stack /sys/class/rfkill/rfkill0/state

chown bluetooth net_bt_stack /sys/class/rfkill/rfkill0/type

write /sys/class/rfkill/rfkill0/state 0


# bluetooth MAC address programming

chown bluetooth net_bt_stack ro.bt.bdaddr_path

chown bluetooth net_bt_stack /system/etc/bluetooth

chown bluetooth net_bt_stack /data/misc/bluetooth

# setprop ro.bt.bdaddr_path "/data/misc/bluetooth/bdaddr"


service dhcpcd_bnep0 /system/bin/dhcpcd -BKLG
disabled
oneshot


service dhcpcd_bnep1 /system/bin/dhcpcd -BKLG
disabled
oneshot


service dhcpcd_bnep2 /system/bin/dhcpcd -BKLG
disabled
oneshot


service dhcpcd_bnep3 /system/bin/dhcpcd -BKLG
disabled
oneshot


service dhcpcd_bnep4 /system/bin/dhcpcd -BKLG
```

```
disabled
```

```
oneshot
```

```
service dhcpcd_bt-pan /system/bin/dhcpcd -ABKL
```

```
class main
```

```
disabled
```

```
oneshot
```

```
service iprenw_bt-pan /system/bin/dhcpcd -n
```

```
class main
```

```
disabled
```

```
oneshot
```

6) `device/{vendor}/{product}/ueventd.{product}.rc`

注意：这个文件只会在采用 USB 蓝牙时才会使用到。请在文件末尾添加下列行：

```
/dev/rtk_btusb 0660 bluetooth net_bt_stack
```

3.1.4 frameworks/base

3.1.5 packages/app/Bluetooth

3.1.6 其他部分

由于很多平台已经集成了其他的蓝牙方案,所以移植的第一件事可能是先将其他方案的蓝牙相关文件去除,最简单的方法如下:

- 1) 将 BoardConfig 中的 BOARD_HAVE_BLUETOOTH_XXX 全部设置为 false 或者删除.

注意:正常情况下这样配置后,其他厂家相关的蓝牙代码都不应该继续编译,但是不排除某些蓝牙厂家代码处理不干净,于是关闭这个配置还是照样会编译该厂家独有的文件,所以请仔细检查后面编译生成的文件是否存在冗余.

- 2) 将 通 过 命 令 ” `rm -rf out/target/product/{product}/obj/*/*bt*`

out/target/product/{product}/obj/*/*lueetooth*

out/target/product/{product}/obj/*/*hci*

out/target/product/{product}/obj/*/*a2dp* out/target/product/{product}/system/* ”删除掉之前编译出来的文件.

- 3) 再次编译 Android, 编译通过后, 确定 out/target/product/{product}/system/lib/hw/ 没有 bluetooth.default.so.如果要是存在 bluetooth.default.so,需要找到对应的 bluedroid源码所在位置,然后按照下面的方法修改 bluedroid 的 Android.mk.

```
ifneq ($(BOARD_HAVE_BLUETOOTH_RTK),true)

.....

#original Makefile

.....

endif
```

- 4) 然后回头执行 2,3 步骤.直到编译后不再产生 bluetooth.default.so
- 5) 完成第 4 步后开始移植我们的蓝牙驱动.
- 6) 在移植完后请留意一下编译出来的/system/app 目录是否有 BCM 或者其他厂家的蓝牙应用.如果在运行中没有问题,则不予处理,如果运行有问题,请将这些 APK 从产品中移除.

注意: 上述这些动作根据平台需要灵活处理,请移植工作负责人留意处理.

3.2 Kernel 部分

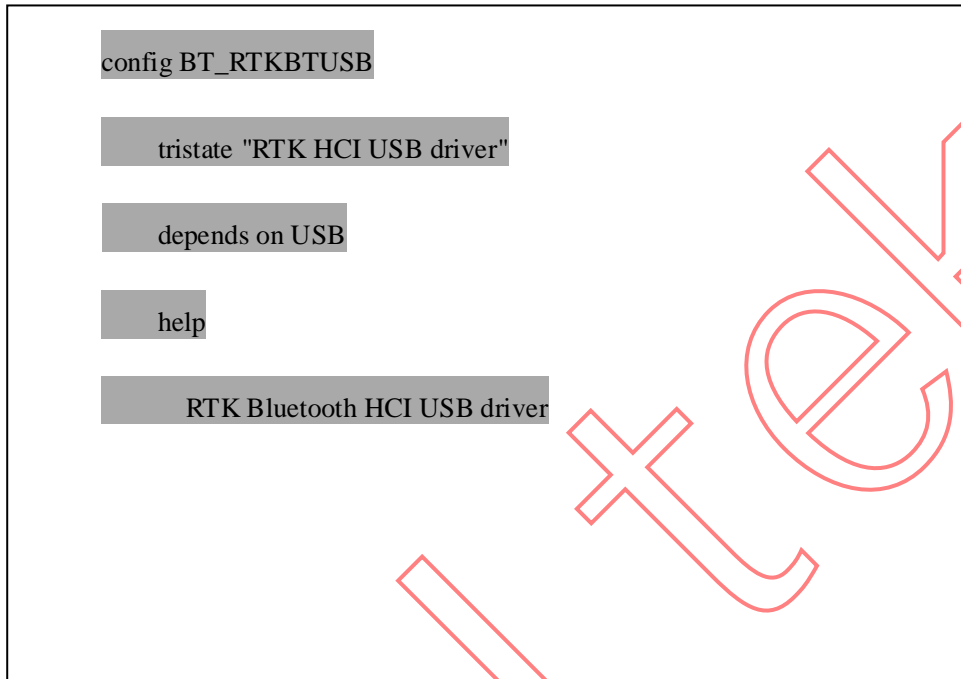
3.2.1 Change list

Chg	linux /driver/bluetooth/Kconfig
Chg	linux /driver/bluetooth/Makefile
New	linux /driver/bluetooth/rtk_btusb.c
New	linux /driver/bluetooth/tk_btusb.h

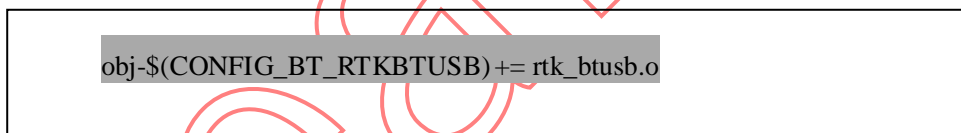
3.2.2 rtk_btusb driver

- 1) 将 Realtek 提供的文件 rtk_btusb.h 和 rtk_btusb.c 拷贝到 linux kernel 的/drivers/bluetooth/目录下;
- 2) 修改 linux kernel 的/kernel/drivers/bluetooth/目录下的文件“Kconfig” and “Makefile”

在 Kconfig 文件中增加 BT_RTKBTUSB 的选项:



在 Makefile 文件中添加目标文件 rtk_btusb.o:



- 3) 在 kernel 中 make menuconfig 选中 rtk_btusb driver;

3.2.3 rtk_rfkill driver

在需要用到 BT_DIS pin 来控制蓝牙芯片的场合, 则需要一个 rfkill 驱动.

rfkill 驱动的具体实现与平台严重相关, 所以如果有需要, 请根据实际情况进行开发. RTK 可以提供 reference code.

注意: 大多数情况下, 客户平台之前搭配过其他家的蓝牙, 所以一般都已有 rfkill 驱动, 只是需要将 GPIO pin 脚接到之前蓝牙使用过的 pin 就可以, 如果不一样, 修改一下 rfkill 驱动中 GPIO 脚即可.

确认当前是否有支持蓝牙的 rfkill 驱动方法如下:

```
cat /sys/class/rfkill/rfkill*/type
```

检查是否有 bluetooth 字样输出,如果有,则已有 rfkill 驱动,不需要重新开发.

3.2.4 TUN driver

如果需要支持 Bluetooth PAN, 确保 TUN Driver 已经编译到 kernel,

```
CONFIG_TUN=y
```

3.2.5 UINPUT driver

如果需要支持 AVRCP 功能, 确认以下打开配置,

```
CONFIG_INPUT_UINPUT=y    # User level driver support
CONFIG_INPUT_MISC=y
```

3.2.6 HID driver

如果需要使用 Bluetooth HID, 必须支持 uhid driver; 确认打开需要支持的 HID 配置

Kernel 对一些 HID 有支持, 请尽可能全部打开。

```
CONFIG_UHID=y
CONFIG_HID_xxx=y
```

注意:经常碰到某些 HID 设备可以连接,但是不能正常使用,大多都是因为这个配置不全导致的兼容性问题, 请确保所有的 CONFIG_HID_xxx 都被设置为 y。

第4章 BT 功能配置

4.1 支持的 Profile 配置

对于有些平板应用不需要支持 PBAP, HFP 以及 HSP, 可以按照下面的配置来关闭这几个 Profile, 如果用户需要支持, 只需把对应值设置为 true 即可。

packages/apps/Bluetooth/res/values/config.xml 文件如下

```
<resources>
    <bool name="profile_supported_a2dp">true</bool>
    <bool name="profile_supported_a2dp_sink">false</bool>
    <bool name="profile_supported_hdp"> false </bool>
    <bool name="profile_supported_hs_hfp"> false </bool>
    <bool name="profile_supported_hfpclient">false</bool>
    <bool name="profile_supported_hid">true</bool>
    <bool name="profile_supported_opp">true</bool>
    <bool name="profile_supported_pan"> false </bool>
    <bool name="profile_supported_pbap"> false </bool>
    <bool name="profile_supported_gatt">true</bool>
    <bool name="pbap_include_photos_in_vcard"> false </bool>
    <bool name="pbap_use_profile_for_owner_vcard"> false </bool>
    <bool name="profile_supported_map"> false </bool>
    <bool name="profile_supported_avrcp_controller">false</bool>
</resources>
```

4.2 配置本地设备名称

修改 device/{vendor}/{product}/rtkbt/bluetooth/bdroid_buildcfg.h

BTM_DEF_LOCAL_NAME 配置项用于配置设备的名字.如果没有这个配置项,将采用 ro.product.model property 设置的名称作为设备名.

4.3 配置本地设备 COD

修改 device/{vendor}/{product}/rtkbt/bluetooth/bdroid_buildcfg.h

```
#define BTA_DM_COD {0x1A, 0x01, 0x1C}
```

三个直接分别表示 DevClassServiceClass, DevClassMajorClass, DevClassMinorClass

默认配置如下:

DevClassServiceClass=0x1A

DevClassMajorClass=0x01

DevClassMinorClass=0x1C

表示设备 COD 为{0x1A,0x01,0x1C}

4.4 配置 Extra Config 文件

对于新卡片支持 Extra Config 文件配置 config, 如需使用 Extra Config 文件配置 config, 可在 /data/misc/bluetooth/目录下新建 rtk_btconfig.txt 文档, 修改文件权限为 644。具体配置方式以 8723bs 为例说明:

```
rtl8723bs_config
0x5b 0x01 0x04 0x21 0x22 0x22 0x21
0xe6 0x01 0x01 0x20
#0xbb 0x01 0x01 0x3c
#0xed 0x00 0x01 0x00
~
```

- 1) 文件首行必须为需要配置的蓝牙芯片的 config 文件名, 如这里对 8723bs config 进行配置, 则首行内容必须与 8723bs config 文件名一致, 即“rtl8723bs_config”
- 2) 从文件次行开始进行 config offset 及 value 值配置。格式为两字节 offset + 一字节 length + length 字节 value 值, 小端模式, 以 16 进制设置, 以空格分隔每个字节, 每行设置一个 offset 及 value。
如图中次行设置即为: offset:0x015b、length:0x04 value:0x21222221
- 3) 支持用“#”进行单行注释
- 4) 不支持对 MAC address 配置

注: 请不要随意使用 Extra Config 文件进行 config 内容配置, 如有需要, 请一定请 FAE 对配置内容进行 review, 否则容易由于异常配置导致各种问题。

第5章 Porting 结束后的基本测试

5.1 配置检查

为了进一步确保 porting 没有问题，在测试之前先确认 fw 以及 config 文件是否存在。

adb shell 到测试平台的根目录，检查测试平台的 system/etc/firmware/ 目录中 rtlxxxx_fw 以及 rtlxxxx_config 文件是否存在(xxxx 为 BT Chip 型号)。检查 Release 包中的 rtkbt/ 下的文件是否都已经安装到对应的目录。

5.2 BT 测试

注意: 本测试是 porting 结束后对 BT 基本和常用功能的一个快速测试，旨在快速验证一些基本问题，不代表 BT 完整的测试，测试结果也非正式 test report。如果使用的是非 Realtek BT chip，该项测试可能没有意义。

5.2.1 基本功能测试

- 1) 打开/关闭 BT 无失败现象。
- 2) 能够搜索到近处 BT 设备。
- 3) 和搜索到的蓝牙耳机或其他设备配对。
- 4) 连接上蓝牙耳机，使用 BT A2DP 听音乐(sdc card 确保存在)。
- 5) 连接上蓝牙耳机，使用 BT HFP/HSP 打电话(确保用蓝牙时能够正常通话)。
- 6) 传输文件到远端支持蓝牙 OPP Server 的设备，从远端支持蓝牙 OPP client 的设备传送文件到本地(sdc card 确保存在)。
- 7) 连接上蓝牙键盘，打开需要输入的应用，通过蓝牙键盘输入。

第6章 蓝牙问题报告

当发现蓝牙有问题的时候,需要同时提供下面的 log,否则可能会因为 log 信息不足无法定位问题.所以请务必学会抓取下列蓝牙 log 的方法.

6.1 蓝牙 Log 相关的配置文件

蓝牙相关的配置文件存放在设备的/system/etc/bluetooth/bt_stack.conf.可以通过 adb 方式进行修改.默认的配置文件如下:

```
# Enable BtSnoop logging function
# valid value : true, false
BtSnoopLogOutput=false

# BtSnoop log output file
BtSnoopFileName=/sdcard/btsnoop_hci.log

# Preserve existing BtSnoop log before overwriting
BtSnoopSaveLog=false

# Enable trace level reconfiguration function
# Must be present before any TRC_ trace level settings
TraceConf=true

# configuration for uart card to save HCI log for slave
H5LogOutput=1

# Enable Coex log
BtCoexLogOutput=0

# Trace level configuration
#   BT_TRACE_LEVEL_NONE      0    ( No trace messages to be generated )
#   BT_TRACE_LEVEL_ERROR     1    ( Error condition trace messages )
#   BT_TRACE_LEVEL_WARNING   2    ( Warning condition trace messages )
#   BT_TRACE_LEVEL_API       3    ( API traces )
#   BT_TRACE_LEVEL_EVENT     4    ( Debug messages for events )
#   BT_TRACE_LEVEL_DEBUG     5    ( Full debug messages )
#   BT_TRACE_LEVEL_VERBOSE   6    ( Verbose messages ) - Currently supported for
TRC_BTAPP only.
```

```

TRC_BTM=2
TRC_HCI=2
TRC_L2CAP=2
TRC_RFCOMM=2
TRC_OBEX=2
TRC_AVCT=2
TRC_AVDT=2
TRC_AVRC=2
TRC_AVDT_SCB=2
TRC_AVDT_CCB=2
TRC_A2D=2
TRC_SDP=2
TRC_GATT=2
TRC_SMP=2
TRC_BTAPP=2
TRC_BTIF=3
TRC_GAP=2
TRC_HID=2

```

6.2 Logcat -v time

在抓取 log 的时候,必须添加“-v time”选项.否则没法将 log 和出现问题的时间点对应起来,给分析 log 会带来很大的麻烦.

测试前,可以打开 log 并将 log 导入到设备的某个分区,待测试完成后再通过 adb, sdcard 等方式导出 log.

6.3 BtSnoop

BtSnoopLogOutput 设置为 TRUE,打开之后,btsnoop 会存放在 BtSnoopFileName 指明的路径下.在蓝牙打开时,这个文件会自动被覆盖一次,所以重现到问题后,需要尽快采用 adb 或者 sdcard 将 btsoop 文件导出.切勿重新开关蓝牙或者重启设备.

6.4 注意事项

报告蓝牙问题至少需要提供下列信息:

- **logcat -v time**
- **btsnoop**
- **出现问题的时间点(以设备时间为准)**
- **重现步骤.**

第7章 常见问题分析

7.1 BT 打开失败(UART)

打开 H5 UART Driver Log, 使用 logcat 抓取 log, 看 H5 SYNC 过程时候成功, 如果 H5 SYNC 失败, 那么需要首先检查硬件电路是否正确 (Power Supply, BT Reset PIN, UART TX/RX, CTS/RTS), 然后检查卡片 efuse, 用示波器量测 UART 波形, 看 Host 是否把数据正确的发送到 Controller。

如果 H5 SYNC 成功, 那么下一步就是 Change Baudrate, 判断 Change Baudrate 是否成功。如果 Change Baudrate 失败, 那么需要确定 Host 是否支持该波特率, config 文件是否正确设定了波特率。

如果 Change Baudrate 成功, 下一步是下载 fw 以及 config 文件, 如果下载完毕之后, 收不到 Controller 回复的 Command Complete Event, 那么需要检查 fw 以及 config 文件是否正确, BT Reset PIN 是否为高电平。

如果下载 fw 以及 config 文件成功, 那么下一步就是根据 config 文件的设定修改 HW Flowcontrol 的设置。设置成功之后, bluebird stack 会下第一个 HCI Comamnd。

如果第一个 HCI Command 一直 H5 重传, 那么说明可能 HW flowcontrol 有问题, 需要检查 Host 的 UART driver 是否支持 HW Flowcontrol。

7.2 BT 打开失败(USB)

用 logcat 抓取打开蓝牙打开的 log, 搜索 “dev/bus/usb” 字样看是否有这样的 log: Added device UsbDevice[mName=/dev/bus/usb/002/002,mVendorId=3034,mProductId=46880,mClass=239,mSubclass=2. 如果有, 检查下 mVendorId 和 mProductId 是不是对应当前使用的蓝牙芯片。如果没有则是没有识别蓝牙卡片, 需要首先检查硬件电路是否正确。

查看 USB 的驱动是否正常加载。登陆到平台里(adb shell), 然后使用命令 lsmod 来查看是否有 rtk_btusb.ko 的存在。