Realtek Bluetooth Porting Guide for Android 6.0

Date : 2015/12/29

Version : 3.0

日期	版本	说明
2015/12/29	V3.0	1. The First version for android 6.0

目　录

1 开始 Porting Code

1. 本文中所有 code 的修改都遵循如下格式：

a) 所有修改的 code 都使用方框括起来。

b) 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。

c) 所有修改或添加的 code 均使用灰色突出显示。

2. 对 code 格式的举例说明：

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/userial_mct.c
else
LOCAL_SRC_FILES += \
    src/hci_h4.c \
    src/userial.c
endif
```

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

```
ifeq ($(BLUETOOTH_HCI_USE_MCT),true)
LOCAL_CFLAGS := -DHCI_USE_MCT
LOCAL_SRC_FILES += \
        src/hci_mct.c \
        src/userial_mct.c
else
ifeq ($(BLUETOOTH_HCI_USE_RTK_H5),true)
LOCAL_CFLAGS := -DHCI_USE_RTK_H5
LOCAL_SRC_FILES += \
    src/hci_h5.c \
     src/userial.c \
    src/bt_skbuff.c \
    src/bt_list.c
```

```
else
LOCAL_SRC_FILES += \
        src/hci_h4.c \
        src/userial.c
endif
endif
```

- 灰色部分 code 是 Realtek 所添加以及修改的 code。

3. porting 说明

该文档 porting 说明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，该文档{vendor}对应 realtek，{product}对应 phoenix。

1.1 SDK 修改说明

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 到客户平台分成了 3 个部分：平台相关修改说明，5.0_bluetooth_patches 以及 Kernel 设置。下面会分别说明。

1.1.1 Change list

1) build

Chg build/core/product.mk

2) device

New device/{**vendor**}/{**common**}/**rtkbt**

New device/**{vendor}/{product}/rtkbt**

Chg device/**{vendor}/{product}/{**PRODUCT_MAKEFILE**}**

Chg device/**{vendor}/{product}**/init.**{**product**}**.rc

Chg device/**{vendor}/{product}**/uevent.**{**product**}**.rc

3) system

New system/**bt/hci/include/bt_list.h bt_skbuff.h rtk_parse.h**

New system/**bt/hci/include/bt_list.c bt_skbuff.c hci_h5.c rtk_parse.c**

Chg system/bt/{**others**}

1.1.2 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， 需要注意其他厂家，例如 BOARD_HAVE_BLUETOOTH_BCM, BLUETOOTH_HCI_USE_MCT 等宏定义可能引入冲突，请注释掉。

1.1.3 Device

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 realtek 提供的 device/{**vendor}/common/rtkbt** 目录拷贝到 device/{vendor}/{common}/下。
2) 将 realtek 提供的 device/**{vendor}/{product}/rtkbt** 目录拷贝到 device/{vendor}/{product}/下。

修改 device/vendor/product /rtkbt/conf/rtkbt.conf 文件，其中的{vendor}与{product}为对应的平台和产品, rtkbt.conf 文件 BtDeviceNode 默认为/dev/ttyS1（Uart 接口），可以修改为 dev/rtk_btusb（Usb 接口）：

```
……..
    #Indicate USB or UART driver bluetooth default uart ttyS1
    #For usb /dev/rtk_btusb   //BT USB 设备节点
    #For uart /dev/ttyS1   //修改为平台实际 BT UART ttySx 节点
    BtDeviceNode=/dev/ttyS1   //修改为平台实际 BT UART   ttySx 节点
……..
```

实际使用中，adb pull system/etc/bluetooth/rtkbt.conf 文件，根据上面来修改 push 到平台，重新开关蓝牙，即可以动态切换 USB/UART 接口的蓝牙芯片。

3) PRODUCT_MAKEFILES

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

```
.....

PRODUCT_MAKEFILES := \

    $(LOCAL_DIR)/ aosp_flo.mk

.....
```

这样可以找到 PRODUCT_MAKEFILES 为 aosp_flo.mk，然后在 aosp_flo.mk 中进行下面的修改（参考 code/device/vendor/product/rtkbt/aosp_flo.mk）:

```
.....

# Inherit from the common Open Source product configuration

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

#rtkbt

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

PRODUCT_NAME := aosp_flo

PRODUCT_DEVICE := flo

.....
```


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

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。

需要注意:

/dev/ttyS1 根据平台蓝牙实际 Uart 接口进行配置。

sys/class/rfkill/rfkill0 根据平台蓝牙实际蓝牙 rfkill 节点进行配置。

A. 如果该产品同时支持 USB/UART 接口蓝牙，请参考 code/device/vendor/product/rtkbt

/init.flo.rc，将如下部分添加到 device/**{vendor}**/**{product}**/init.**{product}**.rc。

```
on boot

.....

#RTK BLUETOOTH START

    # 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 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"


    # UART device

    chmod 0660 /dev/ttyS1

    chown bluetooth net_bt_stack /dev/ttyS1


    # RFKILL

    wait /sys/class/rfkill/rfkill0/state

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

```
    wait /sys/class/rfkill/rfkill0/type
    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


    # USB device
    insmod /system/lib/modules/rtk_btusb.ko
    wait /dev/rtk_btusb
    chmod 0660 /dev/rtk_btusb
    chown bluetooth net_bt_stack /dev/rtk_btusb
#RTK BLUETOOTH END
```


B. 如果该产品仅支持 USB 蓝牙，请参考 code/device/vendor/product/rtkbt /init.flo.rc，将如下部分添加到 device/**{vendor}**/**{product}**/init.**{product}**.rc。

```
on boot

.....

#RTK BLUETOOTH START
    # 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 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"

    # USB device
    insmod /system/lib/modules/rtk_btusb.ko
    wait /dev/rtk_btusb
    chmod 0660 /dev/rtk_btusb
    chown bluetooth net_bt_stack /dev/rtk_btusb
#RTK BLUETOOTH END
```


C. 如果该产品仅支持 UART 蓝牙，请参考 code/device/vendor/product/rtkbt /init.flo.rc，将如下部分添加到 device/**{vendor}**/**{product}**/init.**{product}**.rc。

```
on boot
.....
#RTK BLUETOOTH START
    # 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 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"

    # UART device
    chmod 0660 /dev/ttyS1
    chown bluetooth net_bt_stack /dev/ttyS1

    # RFKILL
    wait /sys/class/rfkill/rfkill0/state
    chmod 0660 /sys/class/rfkill/rfkill0/state
    wait /sys/class/rfkill/rfkill0/type
    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
#RTK BLUETOOTH END
```


5) 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。请参考 code/device/vendor/product/rtkbt /ueventd.flo.rc 在文件末尾添加下列行：

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

1.2 Bluetooth patches

为了使 Realtek BT chip 在 android6.0 上能够正常工作，还需要合入下面的 patch 文件。需要合入的 bluetooth patches 文件都在 system_bt_patches 文件夹下。其中文件夹的命令方式是以目录的方式来命名的，比如：**system_bt_patches** 表示这里面的 patch 都是合入到 system/bt 目录，一般情况下这些目录都会有 git project，这时客户只需要将该目录下的 patch 文件全部合入（git am *.patch）到客户的 SDK 中即可。

1.2.1 如何合入 patch

客户需要首先拷贝 patch 文件到指定的目录，比如对于 **system_bt_patches** 目录下的 patch 文件，首先需要将这下面的 patch 文件全部拷贝到 system/bt 目录，然后在 system/bt 目录执行下面的命令打入所有的 patch。

```
git am *.patch
```

当然，合入 patch 的方法还有很多种，客户也可以采用自己熟悉的方式。合入 patch 时有可能会出现冲突，如果出现冲突了，就需要客户手动来修改解决，这时需要注意： system/**bt/hci/include/bt_list.h bt_skbuff.h rtk_parse.h** 以及 system/**bt/hci/include/bt_list.c bt_skbuff.c hci_h5.c rtk_parse.c** 为新添加的文件，而 system/bt/ 的其他目录中其他的文件均为修改的文件（使用 BLUETOOTH_RTK 以及 BLUETOOTH_RTK_COEX 宏兼容 Android6.0 代码）。

1.2.2 Patch 文件说明

1）packages_apps_Bluetooth_patches

该目录下的 patch 用于修正 bluetooth java 层的一些 bug。

2）packages_apps_Settings_patches

该目录下的 patch 用于修正 bluetooth java 层的一些 bug。

……

1.3 Kernel 设置

1.3.1 Change list

Chg	kernel/arch/arm/configs/xxx_defconfig
Chg	kernel/driver/bluetooth/Kconfig
Chg	kernel/driver/bluetooth/Makefile
New	kernel/driver/bluetooth/rtk_btusb.c
New	kernel/driver/bluetooth/tk_btusb.h

备注:

Chg： 表示 Realtek 在原始的 SDK 文件上做了修改。

New： 表示原始的 SDK 没有该文件，Realtek 添加了该文件。

1.3.2 Rtk_btusb driver

1. 将 Realtek 提供的文件 **rtk_btusb.h 和 rtk_btusb.c** 拷贝到目录 **kernel/drivers/bluetooth/**下；
2. 修改目录/**kernel/drivers/bluetooth/**下的文件“Kconfig” and “Makefile”

在 Kconfig 文件中增加 BT_RTKBTUSB 的选项：

```
config BT_RTKBTUSB
	tristate "RTK HCI USB driver"
	depends on USB
	help
	  RTK Bluetooth HCI USB driver
```


在 Makefile 文件中添加目标文件 rtk_btusb.o:

```
obj-$(CONFIG_BT_RTKBTUSB)+= rtk_btusb.o
```

3. 在 kernel 中 make menuconfig 选中 rtk_btusb driver；

1.3.3 Rtk_rfkill driver

注意事项：

一般平台均已实现 Rfkill 驱动，即申请 RFKILL_TYPE_BLUETOOTH 类型的 rfkill 节点，并且映射 GPIO 为 BT 芯片 RESET_PIN。如有疑问，可以联系厂内进行协助。

1.3.4 TUN driver

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

kernel\arch\arm\configs\XXX_defconfig

```
CONFIG_TUN=y
```

1.3.5 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
```

1.3.6 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 UHID 有支持，请尽可能全部打开。

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

2 BT 功能配置

2.1 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">fasle</bool>
    <bool name="profile_supported_hdp"> false </bool>
    <bool name="profile_supported_hs_hfp"> false </bool>
    <bool name="profile_supported_hfpclient">fasle</bool>
    <bool name="profile_supported_hid">true</bool>
    <bool name="profile_supported_opp">true</bool>
    <bool name="profile_supported_pan">true</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>
```

2.2 本地设备名称，COD 以及 HFP 支持设定

修改 BTM_DEF_LOCAL_NAME 为平台需要的名称。

修改 BTA_DM_COD 为平台需要的 COD（具体请参考 SIG 官方网站 COD 部分 https://www.bluetooth.org/zh-cn/specification/assigned-numbers/baseband），可以根据网站上 COD 的值来确定您需要的 COD。Realtek 默认的值是 #define BTA_DM_COD {0x5A, 0x01, 0x1C}。

Bluedroid 可以设定 HSP/HFP 的不同支持，客户可根据需求设定：

1.同时支持 HSP/HFP，bluedroid 会默认优先使用 HFP, 则在 bdroid_buildcfg.h 中定义

#define BTIF_HF_SERVICES (BTA_HSP_SERVICE_MASK| BTA_HFP_SERVICE_MASK)

#define BTIF_HF_SERVICE_NAMES { BTIF_HSAG_SERVICE_NAME,BTIF_HFAG_SERVICE_NAME}

2. 只支持 HSP, 则在 bdroid_buildcfg.h 中定义

#define BTIF_HF_SERVICES (BTA_HSP_SERVICE_MASK)

#define BTIF_HF_SERVICE_NAMES { BTIF_HSAG_SERVICE_NAME, NULL }

```
#ifndef _BDROID_BUILDCFG_H
#define _BDROID_BUILDCFG_H

#define BTM_DEF_LOCAL_NAME "Realtek Bluetooth"
// SERVICE_CLASS:0x5A (Bit17 -Networking,Bit19 - Capturing,Bit20 -Object Transfer,Bit22
-Telephony)
// MAJOR CLASS: COMPUTER
// MINOR CLASS: TABLET
#define BTA_DM_COD {0x5A, 0x01, 0x1C}

#define BTIF_HF_SERVICES (BTA_HSP_SERVICE_MASK)
#define BTIF_HF_SERVICE_NAMES { BTIF_HSAG_SERVICE_NAME }

#endif
```

2.3 在 host 端配置蓝牙地址

Bluetolth controller 内部有默认设置蓝牙地址，host 也可以修改通过以下方法修改蓝牙地址

1）在 hardware/realtek/bt/libbt/include/vnd_xxx.txt 文件中添加定义

USE_CONTROLLER_BDADDR = FALSE

2）在 init.xxx.rc 中修改指定 bt address 的文件路径

默认设置为：setprop ro.bt.bdaddr_path "/data/misc/bluetooth/bdaddr"

stack 中获取 address 的地方在 btif_core 中 btif_fetch_local_bdaddr 函数中，

获取位置优先级："ro.bt.bdaddr_path" > "persist.service.bdroid.bdaddr" > rand

所以需要修改"ro.bt.bdaddr_path"为指定 bt address 文件路径

3）关于 bdaddr 文件中 BT address 格式应该为 00:00:00:AA:BB:CC

(中间必须以非字符隔开）

3 Porting 结束后的基本测试

3.1 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。

1) 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)。

3.2 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。

3.2.1 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(sdcard 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(sdcard 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。

3.2.2 打开 bluedroid Stack Log 打印方法

修改 system/etc/Bluetooth/bt_stack.conf，把 Debug Level 从 2 调成 6， 并打开 BtSnoop 的 LOG，增加 H5 log 输出选项 H5LogOutput，如果需要测试 H5 数据发送/接收，设置 H5LogOutput 为 ture.

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

# valid value : true, false
H5LogOutput= false

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

# Preserve existing BtSnoop log before overwriting
BtSnoopSaveLog=false

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

# 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=6
TRC_HCI=6
TRC_L5CAP=6
TRC_RFCOMM=6
TRC_OBEX=6
TRC_AVCT=6
TRC_AVDT=6
TRC_AVRC=6
TRC_AVDT_SCB=6
TRC_AVDT_CCB=6
TRC_A5D=6
TRC_SDP=6
TRC_GATT=6
```

```
TRC_SMP=6
TRC_BTAPP=6
TRC_BTIF=6
```

3.3 常见问题分析

3.3.1 BT 打开失败(UART)

打开 H5 UART Driver Log， 使用 logcat 抓取 log，看 H5 SYNC 过程是否成功，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，bluedroid stack 会下第一个 HCI Command。

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。

3.3.2 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。

3.3.3 HFP 没有声音或者都是噪音

首先需要检查，Audio 模块是否把声音切换到 BT，其次需要检查 BT config 文件中的 PCM 设定是否与平台的 PCM 设定匹配。