



Realtek Linux Bluetooth porting guide

Realtek Confidential

Date: 2018/07/20

Version: 5.0

This document is subject to change without notice. The document contains Realtek confidential information and must not be disclosed to any third party without appropriate NDA.

目录

1. 概述	4
2. 平台信息确认	5
2.1. 模块接口及 eFuse 设定	5
2.2. 安装驱动	5
2.3. 移植 BlueZ	5
2.4. 支持其他 stack 的 driver	5
3. Kernel 设定	6
3.1. BlueZ kernel 协议栈设定	6
3.2. UART Transport layer 驱动设定	6
3.3. USB Transport Layer 驱动设定	7
3.4. AVRCP input device support	8
3.5. HOGP user-space HID support	9
4. BlueZ 移植	9
4.1. BlueZ 编译	9
4.2. UART hciattach	11
4.3. 加载固件文件	11
4.4. bluetoothd 启动	12
5. BlueZ 测试和调试	13
5.1. 查看、修改 Controller 信息	13
5.2. 搜索、配对和连接远端设备	13
5.3. 可发现和可连接	14
5.4. Test tools	14
5.5. Test scripts	16
5.6. 搜索服务	16
6. Profile 应用和开发	18
6.1. A2DP and HFP	18
6.1.1. blueALSA	19
6.1.2. PulseAudio	20
6.2. AVRCP CT	25
6.3. LE GATT Client	25
6.3.1. GATT Client API	25
6.3.2. btgatt-client	26
6.3.3. gatttool	28
6.4. LE GATT Server	31
6.4.1. GATT Server API	31
6.4.2. btgatt-server	32
6.4.3. Built-in GATT server	32
6.5. Plugin	32
7. Debug tool	34
7.1. hcidump	34
7.2. BlueZ log	34

7.3. Kernel log	35
8. List of Figures	35

Realtek Confidential

Realtek

1. 概述

本文档描述如何在 Linux 系统中支持 Realtek 蓝牙模块，主要是在系统中移植 BlueZ 以及蓝牙驱动器的说明。

2. 平台信息确认

Realtek 有不同的蓝牙模块方案，首先需要确认模块的特性是否和客户平台相匹配，主要有以下几个方面。

2.1. 模块接口及 eFuse 设定

Realtek 蓝牙模块接口有 USB/UART/PCM。对于 UART 接口，协议分为 H4/Three-wire (H5)。同时 Realtek 蓝牙可以支持单天线/双天线，这些设定都是在 eFuse 中固定的，需要确认模块的设置是否正确。另外有些配置是可以通过 driver 里面的 config 文件来修改，如串口波特率，PCM 参数，32k clock 配置等，详细设定可以参考 [UART interface BT controller initial guide-H4/-H5.pdf](#)。

2.2. 安装驱动

如果客户使用的是 Linux PC 系统或者类似的 Linux 系统，并且 kernel net/bluetooth 已选上，则可以通过 command line 方式直接安装我们的驱动，详细指令参考驱动包内的 Readme.txt，本文档 [3 Kernel 设定](#) 可以略过。如果不是此类系统，请参考 [3. Kernel 设定](#)。

2.3. 移植 BlueZ

如果客户系统并没有移植过 BlueZ 或者需要更新 BlueZ 版本，请从官网 www.bluez.org 下载需要的版本，并(交叉)编译。

2.4. 支持其他 stack 的 driver

如果客户希望使用 BlueZ 之外的 stack，则需要自行开发驱动，本文档以下说明不适用，请参考文件 [UART interface BT controller initial guide-H4/-H5.pdf](#) 或者 [USB interface BT controller initial guide.pdf](#)。

3. Kernel 设定

3.1. BlueZ kernel 协议栈设定

通过 shell 指令 `make menuconfig` 打开 `net/bluetooth` 的选项设定，也可以直接修改平台的 `.config` 文件，确认 kernel 配置中蓝牙有支持，这部分对应的 code 在 `<kernel>/net/bluetooth` 目录。

`make menuconfig (kernel 4.8) [Networking support > Bluetooth subsystem support]`

```

--- Bluetooth subsystem support
[*] Bluetooth Classic (BR/EDR) features
<M> RFCOMM protocol support
[*] RFCOMM TTY support
<M> BNEP protocol support
[ ] Multicast filter support
[ ] Protocol filter support
<M> HIDP protocol support
[ ] Bluetooth High Speed (HS) features
[*] Bluetooth Low Energy (LE) features
[ ] Enable LED triggers
[ ] Bluetooth self testing support
[*] Export Bluetooth internals in debugfs
Bluetooth device drivers --->
    
```

Figure 3-1 BlueZ kernel stack set in make menuconfig

`<kernel>/config`

```

CONFIG_BT=m
CONFIG_BT_BREDR=y
CONFIG_BT_RFCOMM=m
CONFIG_BT_RFCOMM_TTY=y
CONFIG_BT_BNEP=m
CONFIG_BT_HIDP=m
CONFIG_BT_LE=y
CONFIG_BT_DEBUGFS=y
    
```

如果没有 RFCOMM 应用，不需要选择 `CONFIG_BT_RFCOMM` 和 `CONFIG_BT_RFCOMM_TTY`。

如果没有 BNEP protocol 和 PAN 应用，不需要选择 `CONFIG_BT_BNEP`。

如果不需要支持 (BR/EDR Mouse/Keyboard)，则不需要选择 `CONFIG_BT_HIDP`。

3.2. UART Transport layer 驱动设定

这部分是 UART Transport layer 驱动，使用 USB 接口请参考 [3.3. USB Transport Layer 驱动设定](#)。

将 Realtek Bluetooth UART driver 包的 `bluetooth_uart_driver` 文件夹内的 `.h` 和 `.c` 文件都拷贝至 `<kernel>/drivers/bluetooth` 目录。

修改 `<kernel>/drivers/bluetooth` 目录下 `Kconfig` 和 `Makefile` 文件，增加对 Realtek

Three-wire (H5)的支持。

Kconfig:

```
config BT_HCIUART_RTL3WIRE
bool "Realtek Three-wire UART (H5) protocol support"
depends on BT_HCIUART
help
    Realtek Three-wire UART (H5) transport layer makes it possible
    to use Realtek Bluetooth controller with Three-wire UART.

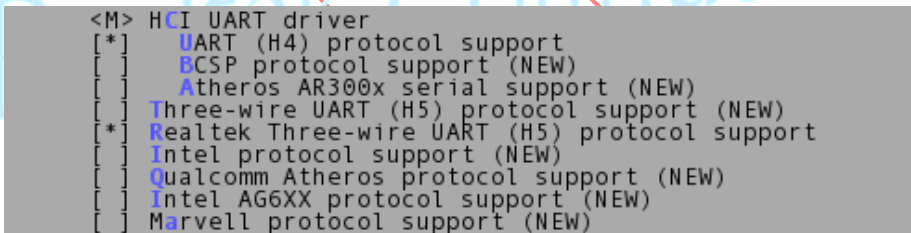
Say Y here to compile support for Realtek Three-wire UART.
```

Makefile:

```
hci_uart-y          += rtk_coex.o
hci_uart-$(CONFIG_BT_HCIUART_RTL3WIRE) += hci_rtk_h5.o
```

通过 make menuconfig 或者直接修改 kernel 的配置文件 <kernel>/.config, 确认 Realtek Three-wire (H5) driver 有支持。

make menuconfig [Networking support > Bluetooth subsystem support > Bluetooth device drivers]



```
<M> HCI UART driver
[*]   UART (H4) protocol support
[ ]   BCSP protocol support (NEW)
[ ]   Atheros AR300x serial support (NEW)
[ ]   Three-wire UART (H5) protocol support (NEW)
[*]   Realtek Three-wire UART (H5) protocol support
[ ]   Intel protocol support (NEW)
[ ]   Qualcomm Atheros protocol support (NEW)
[ ]   Intel AG6XX protocol support (NEW)
[ ]   Marvell protocol support (NEW)
```

Figure 3-2 Realtek Three-wire UART (H5) support in make menuconfig

<kernel>/.config

```
CONFIG_BT_HCIUART=m
CONFIG_BT_HCIUART_H4=y
CONFIG_BT_HCIUART_RTL3WIRE=y
```

3.3. USB Transport Layer 驱动设定

将 Realtek Bluetooth USB driver 包中 bluetooth_usb_driver 文件夹下所有 .c 和 .h 文件拷贝至 **kernel/driver/bluetooth** 目录。

修改 **kernel/driver/bluetooth** 目录下 Kconfig 和 Makefile, 增加对 Realtek Bluetooth HCI USB driver 的支持。

Kconfig:

```
config BT_HCIBTUSB_RTLBTUSB
tristate "Realtek HCI USB driver support"
```

depends on USB

help

Realtek Bluetooth HCI USB driver.

This driver is required if you want to use Realtek Bluetooth device with USB interface.

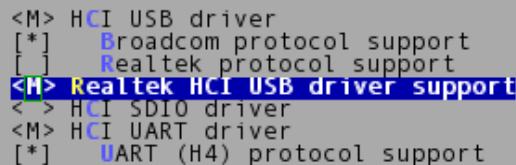
Say Y here to compile support for Bluetooth USB devices into the kernel or say M to compile it as module (rtk_btusb).

Makefile:

```
obj-$(CONFIG_BT_HCIBTUSB_RTLBTUSB) := rtk_btusb.o
rtk_btusb-objs := rtk_bt.o rtk_misc.o rtk_coex.o
```

通过 make menuconfig 或者修改 kernel 的配置文件，确认 Realtek Bluetooth HCI USB driver 有支持。

make menuconfig [Networking support > Bluetooth subsystem support > Bluetooth device drivers]



```
<M> HCI USB driver
[*]   Broadcom protocol support
[ ]   Realtek protocol support
<M> Realtek HCI USB driver support
<> HCI SDIO driver
<M> HCI UART driver
[*]   UART (H4) protocol support
```

Figure 3-3 Realtek HCI USB driver support in make menuconfig

<kernel>/config

```
CONFIG_BT_HCIBTUSB_RTLBTUSB=m
```

3.4. AVRCP input device support

如果要把 AVRCP 的 Up, Down, Left 等按键值和 press/release 信息通过 input device 送给用户态程序,则需要打开 kernel user level driver support (UINPUT)。此选项可以通过 make menuconfig 或者修改 kernel 的配置文件打开。

make menuconfig [Device Drivers > Input device support > Miscellaneous devices]


```

--- Miscellaneous devices
< > Analog Devices AD714x Capacitance Touch Sensor
< > BMA150/SMB380 acceleration sensor support
< > NI Ettus Research USRP E3xx Button support.
< > PC Speaker support
< > MMA8450 - Freescale's 3-Axis, 8/12-bit Digital Accelerometer
< > MPU3050 Triaxial gyroscope sensor
< > Fujitsu Lifebook Application Panel buttons
< > x86 Wistron laptop button interface
< > x86 Atlas button interface
< > ATI / Philips USB RF remote control
< > Keyspan DMR USB remote control
< > Kionix KXTJ9 tri-axis digital accelerometer
< > Griffin PowerMate and Contour Jog support
< > Yealink usb-plk voip phone
< > C-Media CM109 USB I/O Controller
<H> User level driver support
< > PCF8574 Keypad input device

```

Figure 3-4 User lever input device support

<kernel>/config

```

CONFIG_INPUT_MISC=y
CONFIG_INPUT_UINPUT=m

```

3.5. HOGP user-space HID support

如果要把 HoG 的 KEY_1, KEY_2, KEY_ESC 等按键值通过 input device 送给用户态程序, 则需要打开 kernel 的 user-space I/O driver support for HID subsystem(UHID)。此选项可以通过 make menuconfig 或者修改 kernel 的配置文件打开。

make menuconfig [Device Drivers > HID support]

```

-*- HID bus support
[ ] Battery level reporting for HID devices
[*] /dev/hidraw raw HID device support
<H> User-space I/O driver support for HID subsystem
<*> Generic HID driver
Special HID drivers --->
USB HID support --->
I2C HID support --->

```

Figure 3-5 UHID driver support

<kernel>/config

```

CONFIG_UHID=m

```

4. BlueZ 移植

4.1. BlueZ 编译

从 BlueZ 官网 (www.bluez.org) 下载 BlueZ 源码, 进行移植编译。编译过程中还需要

移植 BlueZ 依赖的库，如 D-Bus, GLib 等。推荐使用 **BlueZ 5.xx**，版本越新，支持的功能越多，并且代码越完善；如果需要支持 LE 功能，要求 **kernel 版本 3.5 及以上**，**BlueZ 5.0 及以上**。Realtek BT driver 支持 kernel 版本 2.6.32 以上。

BlueZ 的编译可以参考 <bluez-5.xx>/README。首先通过 configure 进行配置，然后编译。可以通过 configure 增减一些功能，例如：--enable-test 可以编译 <bluez-5.xx>/test 下的一些 test 工具；BlueZ 的 --enable-experimental 选项增加了 heartrate 等 plugins，具体可以看 Makefile.plugins 文件，含有 **if EXPERIMENTAL** 的都受其控制。另外对于 BlueZ 5.xx，需要编译出 bluetoothctl，用于测试，它的 configure 参数为 --enable-client。

BlueZ 中有两个重要的地址与 configure 配置有关：**CONFIGDIR**、**STORAGEDIR**。它们对应的 configure 参数分别是 --sysconfdir 和 --localstatedir。CONFIGDIR 是 BlueZ daemon **bluetoothd** 配置文件的存放位置。STORAGEDIR 是 BlueZ bluetoothd 存放每一个 adapter 和其相关的 devices 信息的位置。

configure:

```
if (test "$sysconfdir" = '${prefix}/etc'); then
    configdir="${prefix}/etc/bluetooth"
else
    configdir="${sysconfdir}/bluetooth"
fi

if (test "$localstatedir" = '${prefix}/var'); then
    storagedir="${prefix}/var/lib/bluetooth"
else
    storagedir="${localstatedir}/lib/bluetooth"
fi
```

若按照 README 设置:

```
./configure --prefix=/usr --mandir=/usr/share/man --sysconfdir=/etc --localstatedir=/var
--libexecdir=/lib
```

CONFIGDIR 为 /etc/bluetooth，STORAGEDIR 为 /var/lib/bluetooth。bluetoothd(后面会介绍)启动后会在 CONFIGDIR 中查找 main.conf 文件进行配置。在 main.conf 中可以设置 name, class 等信息。移植后要将 main.conf 文件放在对应的目录下。

bluetoothd 在 STORAGEDIR 的 adapter address 文件夹内会为每一个 remote device 以 address 创建文件夹，其中的 info 文件会存储该 device 的相关信息，如 Name, Class, LinkKey, LongTermKey 等。

```
[root@linux-pc /var/lib/bluetooth]# cat 00:05:44:33:22:11/33:E0:33:67:98:12/info
[LongTermKey]
Key=27708DFD10B8C6FA085C637D04AEB13E
Authenticated=0
EncSize=16
EDiv=13763
Rand=1129395808884020994
```

```
[ConnectionParameters]
MinInterval=12
MaxInterval=12
Latency=14
Timeout=100

[General]
Name=REALTEK_RCU
Appearance=0x03c0
AddressType=public
SupportedTechnologies=LE;
Trusted=false
Blocked=false
Services=00001800-0000-1000-8000-00805f9b34fb;0000180a-0000-1000-8000-00805f9b34fb;0000180f-0000-1000-8000-00805f9b34fb;00001812-0000-1000-8000-00805f9b34fb;00001813-0000-1000-8000-00805f9b34fb;0000ffd0-0000-1000-8000-00805f9b34fb;1800;180a;180f;1812;1813;ffd0;

[DeviceID]
Source=1
Vendor=93
Product=1
Version=8485
```

4.2. UART hciattach

hciattach tool 是 BlueZ 为 UART 接口蓝牙控制器提供的初始化工具，对于 USB 接口的蓝牙模块直接跳过这部分的设置。

对于 Realtek Bluetooth UART，请使用 driver 包 rtk_hciattach 目录下源码编译生成的 rtk_hciattach，不要使用 BlueZ 编出的 hciattach。初始化时通过 rtk_hciattach -n -s 115200 ttyS1 rtk_h5 或 rtk_hciattach -n -s 115200 ttyS1 rtk_h4 命令执行，其中串口设备名称在各平台会有不同。

4.3. 加载固件文件

蓝牙模块初始化过程需要加载固件文件，对于 UART 接口的可在 hciattach_rtk.c 中定义固件文件所在的目录。默认路径如下：

```
#define FIRMWARE_DIRECTORY "/lib/firmware/rtlbt/"
#define BT_CONFIG_DIRECTORY "/lib/firmware/rtlbt/"
```

对于 USB 接口的蓝牙模块, driver 通过函数 `request_firmware()` 获取固件, 这个函数查找文件的目录与平台相关, 一般情况这个目录为 `/lib/firmware`。需要客户确认所使用的平台对应的位置, 并将固件拷贝至目录。

4.4. bluetoothd 启动

`bluetoothd` 是 BlueZ 的一个守护进程, 实现了 A2DP, AVRCP, GATT, SDP 等 profiles, 并提供 D-Bus services 给外部程序使用。

在启动 `bluetoothd` 之前需要完成以下几步配置:

1、确认 `bluetooth.conf` 已放置在 `/etc/dbus-1/system.d` 目录下, 启动 bluez 进程: `bluetoothd -n -d`; 对于 Bluez 5.xx, 可以添加 `-C` 参数, 提供 deprecated command line interfaces, 如 `sdptool browse local`。

如果 `bluetoothd` 启动失败, 出现以下 log:

D-Bus setup failed: Connection ":1.12" is not allowed to own the service "org.bluez" due to security policies in the configuration file

则可能是 D-Bus 权限问题, 可以在 `/etc/dbus-1/system.d/bluetooth.conf` 中添加

```
<policy user="root">
  <allow own="org.bluez"/>
  <allow send_destination="org.bluez"/>
  <allow send_interface="org.bluez.Agent1"/>
  <allow send_interface="org.bluez.MediaEndpoint1"/>
  <allow send_interface="org.bluez.MediaPlayer1"/>
  <allow send_interface="org.bluez.ThermometerWatcher1"/>
  <allow send_interface="org.bluez.AlertAgent1"/>
  <allow send_interface="org.bluez.Profile1"/>
  <allow send_interface="org.bluez.HeartRateWatcher1"/>
  <allow send_interface="org.bluez.CyclingSpeedWatcher1"/>
  <allow send_interface="org.bluez.GattCharacteristic1"/>
  <allow send_interface="org.bluez.GattDescriptor1"/>
  <allow send_interface="org.freedesktop.DBus.ObjectManager"/>
  <allow send_interface="org.freedesktop.DBus.Properties"/>
  <!-- for bluez 4 -->
  <allow send_interface="org.bluez.Agent"/>
  <allow send_interface="org.bluez.HandsfreeAgent"/>
  <allow send_interface="org.bluez.MediaEndpoint"/>
  <allow send_interface="org.bluez.MediaPlayer"/>
  <allow send_interface="org.bluez.Watcher"/>
  <allow send_interface="org.bluez.ThermometerWatcher"/>
  <allow send_type="method_call"/>
</policy>
```

如果不是以 `root` 运行 `bluetoothd`, 则需要添加相应 user 的 policy, 和 `root policy` 类似, 复制上面的内容, 然后把第一行修改成 `<policy user="xxx">`

不过请在 **root** 权限下运行 **bluetoothd**。

2、如果使用 UART 接口的卡片，需要将#BT_DIS pin 拉 high 并通过 rtk_hciattach 完成初始化和配置：rtk_hciattach -n -s 115200 ttyUSB0 rtk_h5；

3、在 terminal 中输入 rfcill list 查看 Bluetooth 状态。如果蓝牙状态为 Blocked: yes，请使用 rfcill unblock bluetooth 修改蓝牙状态为 Blocked: no。

4、将蓝牙带起来：运行 bluetoothctl，在其提供的命令行中输入 power on 或者使用 hciconfig 工具，命令为 hciconfig hci0 up。建议使用 bluetoothctl 使能蓝牙。

5. BlueZ 测试和调试

在移植 BlueZ 之后，可以使用它提供的一些测试工具测试。

5.1. 查看、修改 Controller 信息

hciconfig -a 可以显示 Controller 的一些基本信息，也可以通过 bluetoothctl 的 show 命令查看 Controller 信息。(bluetoothctl 在 [5.2 搜索、配对和连接远端设备](#) 会说明)。

在 BlueZ 的 main.conf (/etc/bluetooth/main.conf) 中可以设置 Name、Class 等信息。如果要修改 Controller 在手机等远端设备上的显示图标，可以修改 class。例如要显示成耳机，可修改成 0x210404。(Only the major and minor device class bits are considered)。同时需要修改 plugins/hostname.c，注释掉函数 update_name() 和 update_class() 中的代码。

5.2. 搜索、配对和连接远端设备

BlueZ 提供了一个命令行工具 bluetoothctl，实现了 GAP 操作。此程序位于 <bluez-5.xx>/client/ 目录下。请在 root 权限下运行 bluetoothctl，进入内部命令行模式。

```
[bluetooth]# show //查看控制器的 Power 是否为 yes，如果 Power 为 no，则运行 power on
[bluetooth]# power on
[bluetooth]# agent NoInputNoOutput //可以设置其他 IO caps，如 KeyboardDisplay
[bluetooth]# default-agent
[bluetooth]# scan on //扫描到对应的设备后，使用 scan off 关闭 scan。
[bluetooth]# pair 00:22:48:DC:89:0F //配对远端设备。
[bluetooth]# connect 00:22:48:DC:89:0F //连接远端设备
```

配对完之后，可以使用 [bluetooth]# trust 00:22:48:DC:89:0F 指令将以配对过的设备设置为 trusted，避免以后设备回连，请求服务授权时，要求用户输入 Yes/No。如果请求服务授权时，没有 agent，则会出错，最终断开连线。

5.3. 可发现和可连接

如果需要能被其它设备搜索到，则需要打开 Inquiry Scan。如果同时可以被连接，则需要开启 Page Scan。

Inquiry Scan 和 Page Scan 可以通过 hciconfig 开启: `hciconfig hci0 piscan`。参数含义如下:

`piscan`: 将 Page Scan 和 Inquiry Scan 都打开;

`iscan`: 仅打开 Inquiry Scan;

`pscan`: 仅打开 Page Scan;

`noscan`: 关闭 Page Scan 和 Inquiry Scan。

用 `hciconfig hci0 leadv [type]` 可以控制 adapter 发 LE Advertising, `type` 参数为 advertising type (0~4); 用 `hciconfig hci0 noleadv` 停止 advertising。

使用 `bluetoothctl` 也可以打开可发现和可连接模式, 输入 `discoverable on` 可以同时打开 `discoverable` 和 `connectable`, 默认情况开启的是 `limited discoverable`, 时间为 180s。如果希望 `discoverable` 一直开着, 则设置 `main.conf` 中的 `DiscoverableTimeout` 为 0。不过使用 `discoverable on` 不会打开 LE advertising。如果想要打开广播, 可以使用上面的 `hciconfig hci0 leadv`, 也可以使用 `hcitool cmd` 发送 HCI LE commands 设置广播格式并开启广播。

5.4. Test tools

BlueZ 有提供了一些工具, 可以测试基本功能, 具体代码在 `<bluez-5.xx>/tools/` 目录下。

在测试 BlueZ 之前首先要保证 `bluetoothd` 进程已经正常运行并且卡片已经成功带起来。

可用 `hciconfig -a` 确认。

```
[alex_lu@linux-pc /tmp]$ sudo hciconfig -a
hci1:Type: BR/EDR   Bus: UART
      BD Address: 34:C3:D2:0E:AE:36   ACL MTU: 1021:8   SCO MTU: 255:16
      UP RUNNING PSCAN ISCAN
      RX bytes:11078 acl:62 sco:0 events:153 errors:0
      TX bytes:9582 acl:63 sco:0 commands:75 errors:0
      Features: 0xff 0xff 0xff 0xfe 0xdb 0xff 0x7b 0x87
      Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
      Link policy: RSWITCH HOLD SNIFF PARK
      Link mode: SLAVE ACCEPT
      Name: 'luke-INP-131LT-MMTBKCN'
      Class: 0x0c010c
      Service Classes: Rendering, Capturing
      Device Class: Computer, Laptop
      HCI Version: 4.0 (0x6)   Revision: 0x1e6c
      LMP Version: 4.0 (0x6)   Subversion: 0xa747
      Manufacturer: Realtek Semiconductor Corporation (93)
```

(a) hciconfig: HCI 配置工具

```
sudo hciconfig hci0 up /* Open and initialize HCI device */
```

```
sudo hciconfig hci0 iscan /* Visible */
```

其它 hciconfig cmd 可用 `hciconfig --help` 获得或查看 `<bluez-5.xx>/tools/hciconfig.c` 文件。

```
[alex_lu@linux-pc /tmp]$ sudo hciconfig -h
Password:
hciconfig - HCI device configuration utility
Usage:
  hciconfig
  hciconfig [-a] hciX [command ...]
Commands:
  up                Open and initialize HCI device
  down             Close HCI device
  reset            Reset HCI device
  rstat            Reset statistic counters
  auth             Enable Authentication
  noauth           Disable Authentication
  encrypt          Enable Encryption
  noencrypt        Disable Encryption
  piscan           Enable Page and Inquiry scan
  noscan           Disable scan
  iscan            Enable Inquiry scan
  pscan            Enable Page scan
  ptype [type]     Get/Set default packet type
  lm [mode]        Get/Set default link mode
  lp [policy]      Get/Set default link policy
  name [name]      Get/Set local name
  class [class]    Get/Set class of device
  voice [voice]    Get/Set voice setting
  iac [iac]        Get/Set inquiry access code
  inqtpl [level]   Get/Set inquiry transmit power level
  inqmode [mode]   Get/Set inquiry mode
  inqdata [data]   Get/Set inquiry data
  inqtype [type]   Get/Set inquiry scan type
  inqparms [win:int] Get/Set inquiry scan window and interval
  pageparms [win:int] Get/Set page scan window and interval
  ...
```

(b) hcidtool: HCI 工具

```
sudo hcidtool cmd <ogf> <ocf> <parameter1> <parameter2> ... /* Send HCI command */
```

```
sudo hcitool scan /*scan other bluetooth device*/.  
sudo hcitool lescan /*scan other le bluetooth device*/.  
其它 hcitool cmd 可用 hcitool --help 获得或查看 tools/hcitool.c 文件。
```

(c) l2ping: l2cap 测试指令

```
l2ping <address>
```

```
[alex_lu@linux-pc /tmp]$ sudo l2ping 9c:fb:d5:87:03:a3  
Ping: 9c:fb:d5:87:03:a3 from 34:C3:D2:0E:AE:36 (data size 44) ...  
44 bytes from 9c:fb:d5:87:03:a3 id 0 time 6.02ms  
44 bytes from 9c:fb:d5:87:03:a3 id 1 time 17.46ms  
44 bytes from 9c:fb:d5:87:03:a3 id 2 time 26.25ms
```

除了以上工具外，BlueZ 还提供了 `btmgmt`, `btgatt-client`, `btgatt-server`, `gatttool` 等有用的工具。

5.5. Test scripts

BlueZ 提供 D-Bus 及 socket 接口给上层应用调用。<bluez-5.xx>/test/目录下存放着脚本用例演示如何使用这些接口。D-Bus 接口的测试用例一般是 python 脚本，socket 接口是 C 代码。可以参考这些测试文件来实现上层应用开发。

5.6. 搜索服务

如果要查看本地或远端支持的 profile，可以使用 `sdptool` 实现。

查看本地支持的 profile: `sdptool browse local`

NOTE:运行 `bluetoothd` 时要加 -C 参数。

```
[alex_lu@linux-pc /tmp]$ sudo sdptool browse local  
Browsing FF:FF:FF:00:00:00 ...  
Service RecHandle: 0x10000  
Service Class ID List:  
  "PnP Information" (0x1200)  
Profile Descriptor List:  
  "PnP Information" (0x1200)  
  Version: 0x0103  
  
Browsing FF:FF:FF:00:00:00 ...  
Service Search failed: Invalid argument  
Service Name: Generic Access Profile  
Service Provider: BlueZ
```


Service RecHandle: 0x10001

Service Class ID List:

"Generic Access" (0x1800)

Protocol Descriptor List:

"L2CAP" (0x0100)

PSM: 31

"ATT" (0x0007)

uint16: 0x0001

uint16: 0x0005

Service Name: Generic Attribute Profile

Service Provider: BlueZ

Service RecHandle: 0x10002

Service Class ID List:

"Generic Attribute" (0x1801)

Protocol Descriptor List:

"L2CAP" (0x0100)

PSM: 31

"ATT" (0x0007)

uint16: 0x0006

uint16: 0x0009

Service Name: AVRCP CT

Service RecHandle: 0x10003

Service Class ID List:

"AV Remote" (0x110e)

"AV Remote Controller" (0x110f)

Protocol Descriptor List:

"L2CAP" (0x0100)

PSM: 23

"AVCTP" (0x0017)

uint16: 0x0103

Profile Descriptor List:

"AV Remote" (0x110e)

Version: 0x0106

Service Name: AVRCP TG

Service RecHandle: 0x10004

Service Class ID List:

"AV Remote Target" (0x110c)

Protocol Descriptor List:

"L2CAP" (0x0100)

PSM: 23

"AVCTP" (0x0017)

```
uint16: 0x0103
Profile Descriptor List:
  "AV Remote" (0x110e)
  Version: 0x0105

Service Name: Audio Source
Service RecHandle: 0x10005
Service Class ID List:
  "Audio Source" (0x110a)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  PSM: 25
  "AVDTP" (0x0019)
  uint16: 0x0103
Profile Descriptor List:
  "Advanced Audio" (0x110d)
```

用 `sdptool browse <address>` 可以查询远端设备支持的 profile。

6. Profile 应用和开发

BlueZ 提供 D-Bus API、socket 接口和静态、共享库给上层应用调用，用户可以在此基础上开发应用，也可以直接采用一些开源的库和程序进行整合。

一些 Linux based 系统如 Ubuntu/Debian 发行版本已经自带 BlueZ 及应用和 UI，可以在这些系统上做基本测试来了解蓝牙的使用方法和应用场景。

6.1. A2DP and HFP

BlueZ 实现了 a2dp 协议层，但客户需要开发应用来控制链路的管理及 audio data 的解码及输出等，blueALSA 和 PulseAudio 有支持这部分功能，可至 <https://github.com/Arkq/bluez-alsa> 和 <http://pulseaudio.org> 下载源码并移植到 Linux 系统。

BlueZ 提供的 a2dp audio 相关 D-Bus 接口可以查看 `doc/media-api.txt`。

在 BlueZ 5.0 之后，HFP 作为 external profile，需要由外部应用程序实现。Bluetoothd 只负责管理 rfcmm 链路。实现 HFP 的开源应用有 blueALSA 和 oFono。blueALSA 后面会介绍。oFono 虽然也实现了 HFP，但是 audio 部分由 PulseAudio 处理，所以它还需要和 PulseAudio 结合使用才能实现 HFP。

HFP 的 audio data 可以通过 HCI 或者 PCM 接口传给蓝牙控制器。Realtek 提供 Config file 决定 audio data 通过 HCI 传递，还是 PCM 接口。

6.1.1.blueALSA

对于嵌入式的应用，往往受限于 Code Size Limit。PulseAudio 是一个比较庞大的声音系统(Sound Server for Sound Routing)，对于 flash 比较小的嵌入式环境不是很好的选择。按照 bluez-alsa 的说明，This project[Bluez-alsa] is a rebirth of a direct integration between Bluez and ALSA。Bluez-4.xx 内部集成了 BT 到 ALSA 的 sound routing，但是在 Bluez-5.xx 中，这种内建的集成被移除，BT audio routing 转而寻求第三方的 audio app,比如 PulseAudio。所以目前的状态是如果使用 BT audio,就必须用 PulseAudio 或者使用 Bluez-4.xx。Bluez-4.xx 已经过时，官方也不再维护，Bluez 官方推荐使用 Bluez-5.xx。所以安装 PulseAudio 是唯一的办法。

Bluez-alsa 可以替代 PulseAudio，而且依赖更少。Bluez-alsa 注册所有的 Bluez 所支持的 audio profile。Bluez-alsa 的实现基于 ALSA PCM plugin。

1、Dependencies

- Alsa-lib
- Bluez-5+
- Glib with GIO support
- Sbc

2、编译安装

下载 bluez-alsa 源代码:

```
$ git clone https://github.com/Arkq/bluez-alsa.git
$ autoreconf --install
$ mkdir build && cd build
$ ../configure --enable-aac --enable-debug
$ make && make install
```

bluealsa daemon 类似于 bluetoothd 运行于 mainloop 中，bluetoothd 运行的时候会通过 dbus 提供一些 service: 其中一个 RegisterProfile(), bluealsa 初始化时默认注册 a2dp-source, hsp-ag, hfp-ag 3 个 profile。如果改变 profile，可以使用 -p 参数。如 sudo ./src/bluealsa -p a2dp-source -p a2dp-sink -p hfp-hf ...，也可以通过修改 src/bluealsa.c 的 struct ba_config config 来实现。

```
[alex_lu@linux-pc ~/bluez-alsa]$ sudo ./src/bluealsa
Password:
./src/bluealsa: ctl.c:548: Starting controller loop
./src/bluealsa: bluez.c:680: Registering endpoint: /A2DP/SBC/Source/1
./src/bluealsa: bluez.c:914: Registering profile: /HSP/AudioGateway
./src/bluealsa: bluez.c:914: Registering profile: /HFP/AudioGateway
./src/bluealsa: main.c:281: Starting main dispatching loop
```

3、A2DP usage

1) A2DP Sink

在 terminal 中:

```
$ sudo bluealsa-aplay 00:00:00:00:00
```

在 bluetoothctl 中

```
[bluetooth]# connect xx:xx:xx:xx:xx
```

2) A2DP Source

```
$ sudo aplay -D bluealsa:HCI=hci0,DEV=xx:xx:xx:xx:xx:xx,PROFILE=a2dp xxx.wav
```

4、HFP

1) Hands-free

在 bluetoothctl 中

```
$ connect xx:xx:xx:xx:xx
```

在手机上接通电话后，在 terminal 中运行：

```
$ sudo bluealsa-areplay --profile-sco 00:00:00:00:00
```

此时在设备上能听到对方说话的声音。

同时可以使用以下命令播放一段 **8k 16-bit wav** 文件，对方能听到声音。

```
$ sudo aplay -D bluealsa:HCI=hci0,DEV=xx:xx:xx:xx:xx:xx,PROFILE=sco xxx.wav
```

2) audio gateway

和 Hands-free 类似，但是连接的是耳机或者支持 HFP 的音箱，且不需要拨打电话。（因为设备可能没有电话功能）

在 bluetoothctl 中连接耳机或者音箱。

在 terminal 中运行：

```
$ sudo bluealsa-areplay --profile-sco 00:00:00:00:00
```

对着耳机讲话，在设备上能听到说话的声音。

同时可以使用以下命令播放一段 **8k 16-bit wav** 文件，对方能听到声音。

```
$ sudo aplay -D bluealsa:HCI=hci0,DEV=xx:xx:xx:xx:xx:xx,PROFILE=sco xxx.wav
```

注：在没有 PCM 接口的平台，如 Linux PC，请确保 Config file 中存在配置 **f4 00 01 00**。

6.1.2.PulseAudio

a. (交叉)编译

首先需要移植和 PulseAudio 相关的 intltool, libtool, json-c, libsndfile, sbc. 如果在 PC Linux(Ubuntu or Gentoo or etc)下测试，先确认这些 packages 是否已经安装。

编译 pulseaudio，参数(for example)如下：

```
./configure --with-gnu-ld \  
  --enable-shared \  
  --enable-dbus \  
  --disable-static \  
  --disable-x11 \  
  --disable-oss-output \  
  --disable-oss-wrapper \  
  --disable-coreaudio-output \  
  --disable-esound \  
  --disable-solaris \  
  --disable-waveout \  
  --enable-udev
```

```
--disable-gtk3 \  
--disable-gconf \  
--disable-jack \  
--disable-asyncns \  
--disable-tcpwrap \  
--disable-lirc \  
--disable-hal-compat \  
--disable-openssl \  
--disable-xen \  
--disable-systemd \  
--disable-systemd-journal \  
--disable-manpages \  
--disable-per-user-esound-socket \  
--disable-neon-opt \  
--disable-atomic-arm-linux-helpers \  
--disable-ipv6 \  
--disable-glib2 \  
--without-speex \  
--disable-systemd-daemon \  
--disable-systemd-login \  
--disable-systemd-journal \  
--with-system-user=root \  
--with-system-group=root \  
--with-access-group=root \  
--prefix=/usr \  
--sysconfdir=/etc \  
--localstatedir=/var \  
--libdir=/lib \  
--datarootdir=/usr/share \  
--disable-bluez4 \  
--enable-bluez5 \  
--enable-alsa \  
--enable-udev \  
--enable-bluez5-native-headset \  
--enable-bluez5-ofono-headset
```

如果平台是一个嵌入式平台，则需要根据 CPU Type 添加 **--host=xxx-linux**，如 **arm-linux**，**mips-linux** 等。

如果平台没有 udev，则把 **--enable-udev** 改成 **--disable-udev**

如果不需要支持 alsa，把 **--enable-alsa** 改成 **--disable-alsa**。一般情况，如果要支持 a2dp sink，本地都会支持 alsa。

在编译 PulseAudio 前请正确设置 LIBJSON_CFLAGS, LIBJSON_LIBS, LIBSNDFILE_CFLAGS, LIBSNDFILE_LIBS (对于 PC Linux，不需要关心这些 environments，一般系统都已设置好)

b. 运行 PulseAudio

运行 pulseaudio 前需要先把一些配置文件拷贝到系统中, 如 daemon.conf, client.conf, default.pa, system.pa。一般在 Linux PC 上, 这些配置文件被拷贝到/etc/pulse/目录。此目录由 configure 的--sysconfdir 指定。对于嵌入式系统, 可能会更改目标目录。

```
pulseaudio --start
```

pulseaudio 运行时还可以带上 --log-level=X, --dl-search-path=<modules 所在路径>, --file=<pa 后缀的配置文件>。比如 modules 所在路径为/lib/pulse-x.x/modules, pa 后缀的配置文件为/etc/pulse/default.pa 或者/etc/pulse/system.pa

```
运行命令: pulseaudio -n --log-level=3 --dl-search-path=/lib/pulse-7.1/modules
--file=/etc/pulse/default.pa 或者/usr/bin/pulseaudio -n --log-level=3
--dl-search-path=/lib/pulse-7.1/modules --file=/etc/pulse/system.pa --system
```

-n 表示不加载默认的 default.pa 或者 system.pa。

pa 配置文件的路径由 configure 的--sysconfdir 参数决定。比如--sysconfdir=/etc, 则 pa 配置文件所在路径为/etc/pulse/。

modules 路径由 configure 的--libdir 参数决定。比如--libdir=/usr/lib, 则 modules 所在路径为/usr/lib/pulse-x.y/modules/, 其中 x.y 是 pulseaudio 主副版本号, 例如 6.0, 7.1 等。

官方建议使用普通用户权限运行 pulseaudio daemon。不过一般嵌入式平台会以 root 方式运行 PulseAudio。以 root 权限运行 pulseaudio, 运行模式为 system-wide, 运行时传入参数 --system。

调试 PulseAudio 时可能需要根据调试方法设置不同的 log 等级:

- 0: error
- 1: error + warn
- 2: error + warn + info
- 3: error + warn + info + debug

一开始建议打开 error + warn + info log。

如果出现类似的错误: "main.c: Failed to acquire org.pulseaudio.Server: org.freedesktop.DBus.Error.AccessDenied: Connection ":1.46" is not allowed to own the service "org.pulseaudio.Server" due to security policies in the configuration file", 需要在 /etc/dbus-1/system.d/pulseaudio-system.conf (此文件来之于 <pulseaudio-x.y>/src/daemon/pulseaudio-system.conf)中添加如下内容。如果以普通用户运行 pulseaudio, 在这个文件中为普通用户添加一个 policy, 内容和 root 用户类似。

```
<!DOCTYPE busconfig PUBLIC
"-//freedesktop//DTD D-BUS Bus Configuration 1.0//EN"
"http://www.freedesktop.org/standards/dbus/1.0/busconfig.dtd">
<busconfig>
  <policy user="root">
    <allow own="org.pulseaudio.Server"/>
    <allow send_destination="org.bluez"/>
    <allow send_interface="org.bluez.Manager"/>
  </policy>
  <policy user="pulse">
    <allow own="org.pulseaudio.Server"/>
    <allow send_destination="org.bluez"/>
  </policy>
</busconfig>
```

```
<allow send_interface="org.bluez.Manager"/>
</policy>
<policy context="default">
  <deny own="org.pulseaudio.Server"/>
  <deny send_destination="org.bluez"/>
  <deny send_interface="org.bluez.Manager"/>
</policy>
</busconfig>
```

修改配置文件 system/default.pa, 加载 module-bluetooth-policy, module-bluetooth-discover.

```
load-module module-bluetooth-policy
load-module module-bluetooth-discover
```

c. 测试 A2DP

参考 [5.2 搜索、配对和连接远端设备](#), 连接蓝牙耳机。

```
# pactl list modules //确保 module-bluez5-discover 和 module-bluetooth-policy 已加载
# pactl list cards //查看 card
```

Card #0

Name: **bluez_card.00_18_91_00_24_57**

Driver: module-bluez5-device.c

Owner Module: 12

Properties:

device.description = "WOOWI HERO"

device.string = "00:18:91:00:24:57"

device.api = "bluez"

device.class = "sound"

device.bus = "bluetooth"

device.form_factor = "headset"

bluez.path = "/org/bluez/hci0/dev_00_18_91_00_24_57"

bluez.class = "0x240404"

bluez.alias = "WOOWI HERO"

device.icon_name = "audio-headset-bluetooth"

device.intended_roles = "phone"

Profiles:

headset_head_unit: Headset Head Unit (HSP/HFP) (sinks: 1, sources: 1, priority: 20, available: yes)

a2dp_sink: High Fidelity Playback (A2DP Sink) (sinks: 1, sources: 0, priority: 10, available: no)

off: Off (sinks: 0, sources: 0, priority: 0, available: yes)

Active Profile: off

Ports:

headset-output: Headset (priority: 0, latency offset: 0 usec)

Part of profile(s): headset_head_unit, a2dp_sink

headset-input: Headset (priority: 0, latency offset: 0 usec)

```
Part of profile(s): headset_head_unit
```

```
# pactl list sinks
```

```
Sink #1
  State: SUSPENDED
  Name: bluez_sink.00_18_91_00_24_57
  Description: WOOWI HERO
  Driver: module-bluez5-device.c
  Sample Specification: s16le 1ch 8000Hz
  Channel Map: mono
  Owner Module: 20
  Mute: no
  Volume: mono: 43691 / 67%
           balance 0.00
  Base Volume: 65536 / 100%
  Monitor Source: bluez_sink.00_18_91_00_24_57.monitor
  Latency: 0 usec, configured 0 usec
  Flags: HARDWARE HW_VOLUME_CTRL LATENCY
  Properties:
    bluetooth.protocol = "headset_head_unit"
    device.intended_roles = "phone"
    device.description = "WOOWI HERO"
    device.string = "00:18:91:00:24:57"
    device.api = "bluez"
    device.class = "sound"
    device.bus = "bluetooth"
    device.form_factor = "headset"
    bluez.path = "/org/bluez/hci0/dev_00_18_91_00_24_57"
    bluez.class = "0x240404"
    bluez.alias = "WOOWI HERO"
    device.icon_name = "audio-headset-bluetooth"
  Ports:
    headset-output: Headset (priority: 0)
  Active Port: headset-output
  Formats:
    pcm
```

设置 profile 为 a2dp_sink。如果 pactl list cards 显示 BlueZ Card 的 Active Profile 为 a2dp_sink，则省略此操作。

```
# pactl set-card-profile bluez_card.00_18_91_00_24_57 a2dp_sink
```

如果出现 Failure: No such entity 查看 connection 是否断开。
加载 wav 文件

```
# pactl upload-sample /usr/share/sounds/alsa/Front_Center.wav sp1
```

播放

```
# pactl play-sample sp1 bluez_sink.00_18_91_00_24_57
```


6.2. AVRCP CT

BlueZ 提供的 AVRCP CT 相关的 D-Bus 接口可以查看 doc/media-api.txt, interface 为 org.bluez.MediaControl1 和 org.bluez.MediaPlayer1。org.bluez.MediaControl1 methods 的实现在 profile/audio/control.c 文件中。此文件的 control_play(), control_pause(),.... 是 Play, Pause 等 D-Bus API 的具体实现。用户可以根据需求添加更多操作。

6.3. LE GATT Client

6.3.1.GATT Client API

BlueZ 提供了很多有用的 GATT Client API, 方便用户开发 GATT Client application。

```
struct bt_att *bt_att_new(int fd, bool ext_signed)
```

此函数分配类型为 struct bt_att 的结构体, 此结构体包含一个 socket 描述符, 表示 L2CAP 的连接, 此连接通道用于传输 ATT packets。

```
unsigned int bt_att_register_disconnect(struct bt_att *att,  
                                     bt_att_disconnect_func_t callback,  
                                     void *user_data,  
                                     bt_att_destroy_func_t destroy)
```

此函数注册一个处理连线断开的回调函数。

```
struct gatt_db *gatt_db_new(void)
```

此函数分配一个 GATT database, 用来保存远端 GATT server 提供的 service 信息。

```
struct bt_gatt_client *bt_gatt_client_new(struct gatt_db *db,  
                                         struct bt_att *att,  
                                         uint16_t mtu)
```

此函数分配一个 GATT client 实例。

```
unsigned int gatt_db_register(struct gatt_db *db,  
                             gatt_db_attribute_cb_t service_added,  
                             gatt_db_attribute_cb_t service_removed,  
                             void *user_data,  
                             gatt_db_destroy_func_t destroy)
```

此函数注册 3 个回调函数, 分别捕获服务添加事件、服务删除事件和 Database 释放事件。

```
unsigned int bt_gatt_client_ready_register(struct bt_gatt_client *client,
                                          bt_gatt_client_callback_t callback,
                                          void *user_data,
                                          bt_gatt_client_destroy_func_t destroy)
```

此函数注册一个回调函数，捕获 GATT Client Ready 事件。当搜索服务完成时，注册的回调函数被调用。我们可以在此回调函数中打印所有服务的信息，具体参考 tools/btgatt-client.c 的 print_services() 函数。

```
unsigned int bt_gatt_client_read_value(struct bt_gatt_client *client,
                                       uint16_t value_handle,
                                       bt_gatt_client_read_callback_t callback,
                                       void *user_data,
                                       bt_gatt_client_destroy_func_t destroy)
```

此函数用于读取 handle 指向的 Attribute 的 value。

```
unsigned int bt_gatt_client_write_value(struct bt_gatt_client *client,
                                        uint16_t value_handle,
                                        const uint8_t *value, uint16_t length,
                                        bt_gatt_client_callback_t callback,
                                        void *user_data,
                                        bt_gatt_client_destroy_func_t destroy)
```

此函数用于改变 handle 指向的 Attribute 的 value。

请参考 <bluez-5.xx>/tools/btgatt-client.c 了解更多 GATT Client API 的使用。

6.3.2.btgatt-client

btgatt-client 可以连接远端 LE device，搜索服务，读写 Characteristics。

```
[alex_lu@linux-pc ~/bluez/bluez-5.41]$ sudo ./tools/btgatt-client -d 00:E0:00:34:56:78
Connecting to device... Done
[GATT client]# Service Added - UUID: 00001800-0000-1000-8000-00805f9b34fb start: 0x0001
end: 0x0007
[GATT client]# Service Added - UUID: 00001812-0000-1000-8000-00805f9b34fb start: 0x0008
end: 0x0026
[GATT client]# Service Added - UUID: 00001801-0000-1000-8000-00805f9b34fb start: 0x0027
end: 0x002a
...
[GATT client]# GATT discovery procedures complete
[GATT client]#
service - start: 0x0001, end: 0x0007, type: primary, uuid:
00001800-0000-1000-8000-00805f9b34fb
      charac - start: 0x0002, value: 0x0003, props: 0x02, ext_props: 0x0000, uuid:
00002a00-0000-1000-8000-00805f9b34fb
```

```
charac - start: 0x0004, value: 0x0005, props: 0x02, ext_props: 0x0000, uuid:
00002a01-0000-1000-8000-00805f9b34fb
charac - start: 0x0006, value: 0x0007, props: 0x02, ext_props: 0x0000, uuid:
00002a04-0000-1000-8000-00805f9b34fb

service - start: 0x0008, end: 0x0026, type: primary, uuid:
00001812-0000-1000-8000-00805f9b34fb
charac - start: 0x0009, value: 0x000a, props: 0x02, ext_props: 0x0000, uuid:
00002a4a-0000-1000-8000-00805f9b34fb
charac - start: 0x000b, value: 0x000c, props: 0x04, ext_props: 0x0000, uuid:
00002a4c-0000-1000-8000-00805f9b34fb
charac - start: 0x000d, value: 0x000e, props: 0x06, ext_props: 0x0000, uuid:
00002a4e-0000-1000-8000-00805f9b34fb
charac - start: 0x000f, value: 0x0010, props: 0x02, ext_props: 0x0000, uuid:
00002a4b-0000-1000-8000-00805f9b34fb
charac - start: 0x0011, value: 0x0012, props: 0x1a, ext_props: 0x0000, uuid:
00002a4d-0000-1000-8000-00805f9b34fb
descr - handle: 0x0013, uuid: 00002902-0000-1000-8000-00805f9b34fb
descr - handle: 0x0014, uuid: 00002908-0000-1000-8000-00805f9b34fb
charac - start: 0x0015, value: 0x0016, props: 0x1a, ext_props: 0x0000, uuid:
00002a4d-0000-1000-8000-00805f9b34fb
descr - handle: 0x0017, uuid: 00002902-0000-1000-8000-00805f9b34fb
descr - handle: 0x0018, uuid: 00002908-0000-1000-8000-00805f9b34fb
charac - start: 0x0019, value: 0x001a, props: 0x0e, ext_props: 0x0000, uuid:
00002a4d-0000-1000-8000-00805f9b34fb
descr - handle: 0x001b, uuid: 00002908-0000-1000-8000-00805f9b34fb
charac - start: 0x001c, value: 0x001d, props: 0x1a, ext_props: 0x0000, uuid:
00002a4d-0000-1000-8000-00805f9b34fb
descr - handle: 0x001e, uuid: 00002902-0000-1000-8000-00805f9b34fb
descr - handle: 0x001f, uuid: 00002908-0000-1000-8000-00805f9b34fb
charac - start: 0x0020, value: 0x0021, props: 0x1a, ext_props: 0x0000, uuid:
00002a4d-0000-1000-8000-00805f9b34fb
descr - handle: 0x0022, uuid: 00002902-0000-1000-8000-00805f9b34fb
descr - handle: 0x0023, uuid: 00002908-0000-1000-8000-00805f9b34fb
charac - start: 0x0024, value: 0x0025, props: 0x0e, ext_props: 0x0000, uuid:
00002a4d-0000-1000-8000-00805f9b34fb
descr - handle: 0x0026, uuid: 00002908-0000-1000-8000-00805f9b34fb

service - start: 0x0027, end: 0x002a, type: primary, uuid:
00001801-0000-1000-8000-00805f9b34fb
charac - start: 0x0028, value: 0x0029, props: 0x20, ext_props: 0x0000, uuid:
00002a05-0000-1000-8000-00805f9b34fb
descr - handle: 0x002a, uuid: 00002902-0000-1000-8000-00805f9b34fb
```

```

...

[GATT client]# help
Commands:
  help                Display help message
  services            Show discovered services
  read-value          Read a characteristic or descriptor value
  read-long-value     Read a long characteristic or descriptor value
  read-multiple       Read Multiple
  write-value         Write a characteristic or descriptor value
  write-long-value    Write long characteristic or descriptor value
  write-prepare       Write prepare characteristic or descriptor value
  write-execute      Execute already prepared write
  register-notify     Subscribe to not/ind from a characteristic
  unregister-notify   Unregister a not/ind session
  set-security        Set security level on le connection
  get-security        Get security level on le connection
  set-sign-key        Set signing key for signed write command

[GATT client]# read-value 0x0003
Read value (11 bytes): 4d 65 64 69 61 51 20 52 43 55 00

```

6.3.3.gatttool

除了 `btgatt-client` 之外，BlueZ 还提供了另外一个有用的工具 `gatttool`，此工具位于 `<bluez-5.xx>/attrib/` 目录，也提供了搜索服务，读写 Characteristics 等功能。

```

[alex_lu@linux-pc ~/bluez/bluez-5.47]$ sudo ./attrib/gatttool --help-all
Password:
Usage:
  gatttool [OPTION...]

Help Options:
  -h, --help                Show help options
  --help-all                Show all help options
  --help-gatt                Show all GATT commands
  --help-params              Show all Primary Services/Characteristics
arguments
  --help-char-read-write    Show all Characteristics Value/Descriptor
Read/Write arguments

GATT commands
  --primary                  Primary Service Discovery

```

--characteristics	Characteristics Discovery
--char-read	Characteristics Value/Descriptor Read
--char-write	Characteristics Value Write Without Response
(Write Command)	
--char-write-req	Characteristics Value Write (Write Request)
--char-desc	Characteristics Descriptor Discovery
--listen	Listen for notifications and indications
Primary Services/Characteristics arguments	
-s, --start=0x0001	Starting handle(optional)
-e, --end=0xffff	Ending handle(optional)
-u, --uuid=0x1801	UUID16 or UUID128(optional)
Characteristics Value/Descriptor Read/Write arguments	
-a, --handle=0x0001	Read/Write characteristic by handle(required)
-n, --value=0x0001	Write characteristic value (required for write operation)
Application Options:	
-i, --adapter=hciX	Specify local adapter interface
-b, --device=MAC	Specify remote Bluetooth address
-t, --addr-type=[public random]	Set LE address type. Default: public
-m, --mtu=MTU	Specify the MTU size
-p, --psm=PSM	Specify the PSM for GATT/ATT over BR/EDR
-l, --sec-level=[low medium high]	Set security level. Default: low
-I, --interactive	Use interactive mode

搜索 primary services

```
gatttool sudo ./attrib/gatttool -b 00:E0:00:34:56:78 --primary
```

```
[alex_lu@linux-pc ~/bluez/bluez-5.47]$ sudo ./attrib/gatttool -b 00:E0:00:34:56:78 --primary
Password:
attr handle = 0x0001, end grp handle = 0x0007 uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle = 0x0008, end grp handle = 0x0026 uuid: 00001812-0000-1000-8000-00805f9b34fb
attr handle = 0x0027, end grp handle = 0x002a uuid: 00001801-0000-1000-8000-00805f9b34fb
...
```

搜索 characteristics:

gatttool -b <address> --characteristics, 也可加上 start handle 和 end handle, gatttool -b <address> --characteristics -s <start handle> -e <end handle>

```
[alex_lu@linux-pc ~/bluez/bluez-5.47]$ sudo ./attrib/gatttool -b 00:E0:00:34:56:78 --char-desc
handle = 0x0001, uuid = 00002800-0000-1000-8000-00805f9b34fb
handle = 0x0002, uuid = 00002803-0000-1000-8000-00805f9b34fb
handle = 0x0003, uuid = 00002a00-0000-1000-8000-00805f9b34fb
handle = 0x0004, uuid = 00002803-0000-1000-8000-00805f9b34fb
```

```
handle = 0x0005, uuid = 00002a01-0000-1000-8000-00805f9b34fb
handle = 0x0006, uuid = 00002803-0000-1000-8000-00805f9b34fb
handle = 0x0007, uuid = 00002a04-0000-1000-8000-00805f9b34fb
handle = 0x0008, uuid = 00002800-0000-1000-8000-00805f9b34fb
handle = 0x0009, uuid = 00002803-0000-1000-8000-00805f9b34fb
...

[alex_lu@linux-pc ~/bluez/bluez-5.47]$ sudo ./attrib/gatttool -b 00:E0:00:34:56:78
--characteristics
handle = 0x0002, char properties = 0x02, char value handle = 0x0003, uuid =
00002a00-0000-1000-8000-00805f9b34fb
handle = 0x0004, char properties = 0x02, char value handle = 0x0005, uuid =
00002a01-0000-1000-8000-00805f9b34fb
handle = 0x0006, char properties = 0x02, char value handle = 0x0007, uuid =
00002a04-0000-1000-8000-00805f9b34fb
handle = 0x0009, char properties = 0x02, char value handle = 0x000a, uuid =
00002a4a-0000-1000-8000-00805f9b34fb
handle = 0x000b, char properties = 0x04, char value handle = 0x000c, uuid =
00002a4c-0000-1000-8000-00805f9b34fb
handle = 0x000d, char properties = 0x06, char value handle = 0x000e, uuid =
00002a4e-0000-1000-8000-00805f9b34fb
handle = 0x000f, char properties = 0x02, char value handle = 0x0010, uuid =
00002a4b-0000-1000-8000-00805f9b34fb
handle = 0x0011, char properties = 0x1a, char value handle = 0x0012, uuid =
00002a4d-0000-1000-8000-00805f9b34fb
handle = 0x0015, char properties = 0x1a, char value handle = 0x0016, uuid =
00002a4d-0000-1000-8000-00805f9b34fb
...

[alex_lu@linux-pc ~/bluez/bluez-5.47]$ sudo ./attrib/gatttool -b 00:E0:00:34:56:78
--characteristics -s 0x0001 -e 0x0010
handle = 0x0002, char properties = 0x02, char value handle = 0x0003, uuid =
00002a00-0000-1000-8000-00805f9b34fb
handle = 0x0004, char properties = 0x02, char value handle = 0x0005, uuid =
00002a01-0000-1000-8000-00805f9b34fb
handle = 0x0006, char properties = 0x02, char value handle = 0x0007, uuid =
00002a04-0000-1000-8000-00805f9b34fb
handle = 0x0009, char properties = 0x02, char value handle = 0x000a, uuid =
00002a4a-0000-1000-8000-00805f9b34fb
handle = 0x000b, char properties = 0x04, char value handle = 0x000c, uuid =
00002a4c-0000-1000-8000-00805f9b34fb
handle = 0x000d, char properties = 0x06, char value handle = 0x000e, uuid =
00002a4e-0000-1000-8000-00805f9b34fb
handle = 0x000f, char properties = 0x02, char value handle = 0x0010, uuid =
```

6.4. LE GATT Server

6.4.1. GATT Server API

BlueZ 提供了一些 GATT Server API，方便用户开发 GATT Server application。

```
struct bt_gatt_server *bt_gatt_server_new(struct gatt_db *db,  
                                          struct bt_att *att, uint16_t mtu)
```

此函数分配类型为 struct bt_gatt_server 的结构体，此函数还注册了 exchange mtu, read by group type, read by type 等 ATT request 的处理函数。

```
struct gatt_db_attribute *gatt_db_add_service(struct gatt_db *db,  
                                              const bt_uuid_t *uuid,  
                                              bool primary,  
                                              uint16_t num_handles)
```

添加一个 service，在它之后需要调用其他函数添加 characteristics 和 descriptors。

```
struct gatt_db_attribute *  
gatt_db_service_add_characteristic(struct gatt_db_attribute *attrib,  
                                  const bt_uuid_t *uuid,  
                                  uint32_t permissions,  
                                  uint8_t properties,  
                                  gatt_db_read_t read_func,  
                                  gatt_db_write_t write_func,  
                                  void *user_data)
```

添加一个 characteristic 到 attrib 指向的 service。

```
struct gatt_db_attribute *  
gatt_db_service_add_descriptor(struct gatt_db_attribute *attrib,  
                              const bt_uuid_t *uuid,  
                              uint32_t permissions,  
                              gatt_db_read_t read_func,  
                              gatt_db_write_t write_func,  
                              void *user_data)
```

添加一个 descriptor 到 attrib 指向的 service。

```
bool gatt_db_attribute_write(struct gatt_db_attribute *attrib, uint16_t offset,  
                             const uint8_t *value, size_t len,  
                             uint8_t opcode, struct bt_att *att,
```

```
gatt_db_attribute_write_t func,  
void *user_data)
```

设置 attribute 的 value。此函数一般在没有设置 attribute write callback 的情况下调用。

```
bool gatt_db_service_set_active(struct gatt_db_attribute *attrib, bool active)
```

激活 service。

6.4.2. btgatt-server

GATT server 的开发可以参考 <bluez-5.xx>/tools/btgatt-server.c。

6.4.3. Built-in GATT server

内嵌的 GATT server 可以参考 rtk_gatt_server.c



rtk_gatt_server.c

6.5. Plugin

开发新的 profile，需要注册 plugin，首先在源码文件中添加如下内容：

```
BLUETOOTH_PLUGIN_DEFINE(name, VERSION, BLUETOOTH_PLUGIN_PRIORITY_DEFAULT, init,  
exit)
```

在 init 函数中将该 profile 添加到 profiles 链表上，注册 profile 的函数是 btd_profile_register(&profile1)。在 exit 时调用 btd_profile_unregister(&profile1)。

然后修改 Makefile.plugins，添加：

```
builtin_modules += name  
builtin_sources += <src file>
```

修改 Makefile.plugins 之后需要在 <bluez-5.xx> 源码目录下运行一个脚本 bootstrap (可以自己命令)，脚本内容为：

```
#!/bin/sh  
  
aclocal && \  
  autoheader && \  
  libtoolize --automake --copy --force && \  
  automake --add-missing --copy && \  
  autoconf
```

参考 [4.1 BlueZ 编译](#) 生成 Makefile 并编译。

Realtek Confidential

REALTEK

7. Debug tool

7.1. hcidump

BlueZ 有提供抓取 btsnoop log 的工具 hcidump。此工具可以录下 host stack 与 Bluetooth controller 之间交互的蓝牙数据。此工具位于<bluez-5.xx>/tools/。

hcidump 工具可以将 log 打印在 terminal，也可以以 raw data 的方式保存到文件，保存的文件可以用 Frontline 的 Capture File Viewer 解析，也可以用 Ellisys Bluetooth Analyzer 解析。

抓取 btsnoop log 的命令

直接打印到 terminal:

```
$ sudo hcidump -t
```

把文本 log 保存到 txt 文件:

```
$ sudo hcidump -t > xxx.txt
```

保存 raw data:

```
$ sudo hcidump -w xxx.cfa
```

7.2. BlueZ log

运行 bluetoothd 时加 **-d -n** 参数打开 BlueZ debug log (源码中使用 DBG()函数打印的 log 都将输出)。

BlueZ 的 log 同时保存到 syslog，所以要确定客户平台是否支持，如 Ubuntu 系统 log 会存储在 /var/log/syslog 文件中。客户可以移植 busybox 将 syslogd 打开，启动后在后台运行 syslogd，log 将会保存在 /var/log/message 文件中。

```
void btd_debug(const char *format, ...)
{
    va_list ap;

    va_start(ap, format);
    vsyslog(LOG_DEBUG, format, ap);
    va_end(ap);

    ...
}
```

7.3. Kernel log

通过 `echo 7 > /proc/sys/kernel/printk` 修改 `printk` 级别, 将 `net/bluetooth` 和 `Realtek Bluetooth driver` 的 log 打印出来。

8. List of Figures

Figure 3-1 BlueZ kernel stack set in make menuconfig	6
Figure 3-2 Realtek Three-wire UART (H5) support in make menuconfig	7
Figure 3-3 Realtek HCI USB driver support in make menuconfig	8
Figure 3-4 User lever input device support	9
Figure 3-5 UHID driver support	9

Realtek Confidential

READ