



Perfect Wireless Experience
完美无线体验

FIBOCOM NL系列

Android RIL 适配指导手册

文档版本：V1.0.1

更新日期：2019-10-22



适用型号

序号	产品型号	说明
1	NL668-CN	NA
2	NL668-AM	NA
3	NL668-EAU	NA
4	NL668-EU	NA
5	NL668-JP	NA
6	NL668-LA	NA
7	NL678-E	NA
8	NL652-EU	NA

版权声明

版权所有©2019 深圳市广和通无线股份有限公司。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

注意

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

商标申明



为深圳市广和通无线股份有限公司的注册商标，由所有人拥有。

版本记录

文档版本	编写人	主审人	批准人	更新日期	说明
V1.0.0	李石峰			2019-03-20	初始版本
V1.0.1	王梦莹	龙奕良	程凯	2019-10-22	添加部分适用型号

目录

1	引言	5
2	Android RIL 驱动概述	5
2.1	RIL 驱动支持的功能	5
2.2	支持的 Android 版本	5
3	RIL 集成配置	6
3.1	RIL 库源码结构	6
3.2	配置 Linux 内核串口驱动	7
3.3	添加模块 PID/VID	7
3.4	NL668/NL678 模块设备加载检测	8
3.5	修改 RILD 权限	9
3.6	配置 Android 启动脚本	9
3.7	加载 RIL 库文件	10
3.7.1	手动加载 RIL 库文件	10
3.7.2	自动打包 RIL 库文件	10
4	NDIS 拨号方式	11
4.1	加载 Gobinet 驱动	11
4.2	获取 local ip,primary dns,second dns 地址	12
5	ppp 拨号方式	12
5.1	修改 ip-down ip-up 文件	12
6	APN 设置和拨号状态查询	14
6.1	手动添加 APN	14
6.2	配置 apns-conf.xml 文件	14
7	RIL 调试方法	15
8	LOG 中 TAG 标签说明	16
9	常见问题	16
9.1	无法正常的进行数据业务	16
9.2	电话打不通	16
9.3	无法发送短信	17
9.4	RIL 没有正常工作	17
10	附录 A 参考文件	17

1 引言

本文主要介绍如何将 RIL（无线接口层）驱动程序集成到客户的目标设备以及如何修改启动 RIL 服务的配置文件。

2 Android RIL 驱动概述

2.1 RIL 驱动支持的功能

表 1: RIL 支持的功能

功能	是否支持
SMS	YES
VOICE CALL	YES
DATA SERVICE	YES
SIM TOOL KIT	NO

2.2 支持的 Android 版本

目前，Fibocom RIL driver 支持以下的 Android 版本，见下表 2。

表 2: RIL 支持 android 系统版本

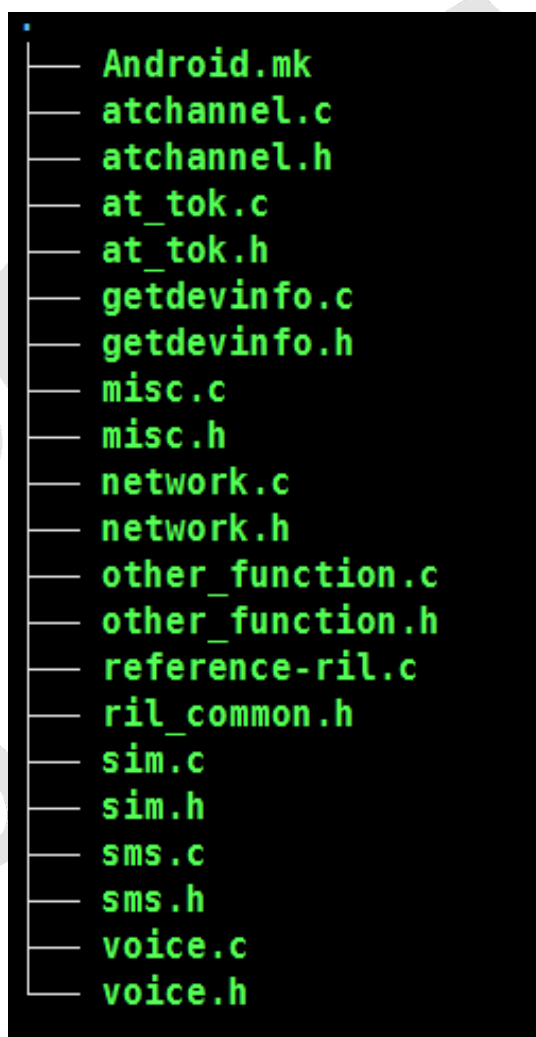
功能	是否支持
Android 2.x	NO
Android 3.x	NO
Android 4.x	YES
Android 5.x	YES
Android 6.x	YES
Android 7.x	YES
Android 8.x	YES
Android 9.x	NO

3 RIL 集成配置

本章节介绍 RIL 库集成到客户 Android 系统的操作步骤，包括 RIL 库文件结构，配置 Linux 内核驱动，添加内核支持的 USB 设备 PID/VID，加载 RIL 库文件，以及修改 Android 系统 RILD 执行权限和启动脚本。

3.1 RIL 库源码结构

Fibocom 会提供编译生成的 RIL 库文件 libreference-ril.so 和 RIL 库源码供客户集成使用。RIL 库源码包含 21 个主要文件，如下图 1 所示：



```
Android.mk
atchannel.c
atchannel.h
at_tok.c
at_tok.h
getdevinfo.c
getdevinfo.h
misc.c
misc.h
network.c
network.h
other_function.c
other_function.h
reference-ril.c
ril_common.h
sim.c
sim.h
sms.c
sms.h
voice.c
voice.h
```

图 1: RIL 库源码文件结构

3.2 配置 Linux 内核串口驱动

串口驱动需要配置 Linux 内核，配置方法如下：

- 1) cd kernel 进入到内核根目录
- 2) make menuconfig
- 3) device drivers -> usb support -> usb serial converter support

选中如下组件：

USB driver for GSM and CDMA modems 选中后保存退出。

*关于 Linux 内核编译配置方法，以客户 android 系统的配置规则为准，文档描述的方法仅供参考。

3.3 添加模块 PID/VID

- 1) 打开内核源码文件 option.c(路径一般为 drivers/usb/serial/option.c)。在源码中找到 option_ids 数组，在数组中添加模块的 PID 和 VID,以 NL668 添加 VID(0x1508)和 PID(0x1001)为例。

```
static const struct usb_device_id option_ids[] = {
.....
{ USB_DEVICE(0x1508, 0x1001) },
.....
}
```

- 2) 在 USB 串口驱动中，过滤 NDIS 接口。由于 USB 串口跟 NDIS 都属于非标准 CDC 设备，需要防止 NDIS 口被 USB 串口驱动加载而导致无法正常加载 NDIS 口驱动，并在 option_probe 函数内判断当前的 interface num 进行过滤，具体如下：

```
static int option_probe(struct usb_serial *serial, const struct usb_device_id *id)
.....
if (serial->dev->descriptor.idVendor == cpu_to_le16(0x1508) &&
serial->dev->descriptor.idProduct == cpu_to_le16(0x1001) &&
serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4) {
printk(KERN_INFO "Discover the 4th interface for fibocom\n");
return -ENODEV;
}
```



注意：

NL668/NL678 系列模块 PID/VID,可通过 android 系统命令 “lsusb” 获取当前使用模块的 PID/VID，也可以咨询 Fibocom 技术人员。

3.4 NL668/NL678 模块设备加载检测

将上述 patch 修改后的代码编译烧入目标设备，使用 adb shell:

```
root@android:/ # ls /dev/ttyUSB* //若设备正常挂载，将会有如下内容返回：
ls /dev/ttyUSB*
/dev/ttyUSB0
/dev/ttyUSB1
/dev/ttyUSB2
/dev/ttyUSB3
```

表 3: USB 端口功能说明

设备名	作用	备注
ttyUSB0	DIAG	获取 modem log 端口
ttyUSB1	MODEM	ppp 拨号端口
ttyUSB2	AT	RIL 程序发送 AT 命令请求和接收命令响应的端口。
ttyUSB3	NEMA	gps 端口（未用到）



注意:

适配 RIL 需要用 AT+GTUSBMODE?查询 USB 模式是否为 17, 如果不是, 先用 AT+GTUSBMODE=17 设置为 17。
如果是定制版本, 则不需要操作。

3.5 修改 RILD 权限

RILD 程序执行过程中需要 root 权限，可在 rild.c (<\$android dir> /hardware/ril/rild/rild.c) main 函数中，去掉 switchuser 函数的调用来达到此目的，如下图 2 所示：

```

224         arg_device[sizeof(arg_device)-1] = 0;
225         arg_overrides[1] = "-d";
226         arg_overrides[2] = arg_device;
227         done = 1;
228
229     } while (0);
230
231     if (done) {
232         argv = arg_overrides;
233         argc = 3;
234         i = 1;
235         hasLibArgs = 1;
236         rilLibPath = REFERENCE_RIL_PATH;
237
238         |   RLOGD("overriding with %s %s", arg_overrides[1], arg_overrides[2]);
239     }
240 }
241 OpenLib:
242 #endif
243 //switchUser();
244
245 dlHandle = dlopen(rilLibPath, RTLD_NOW);
246
247 if (dlHandle == NULL) {
248     RLOGE("dlopen failed: %s", dlerror());
249     exit(-1);
250 }
251

```

图 2: RILD 中 main 函数修改

3.6 配置 Android 启动脚本

配置 init.rc 文件添加 ril-daemon 进程

```

service ril-daemon /system/bin/rild -l /system/lib/libreference-ril.so -- -w 0
class main
socket rild stream 660 root radio
socket rild-debug stream 660 radio system
user root
group radio cache inet misc audio sdcard_rw log

```

- -l 指定 RIL 库加载路径
- -w 指定拨号方式，0：ppp 拨号，1：ECM 拨号，2：NDIS 拨号（需要 GobiNet 驱动支持）

也可在 Android OS 属性文件/system/build.prop 中增加属性项 ril.fibocom.dialmode

ril.fibocom.dialmode=2: NDIS 拨号

ril.fibocom.dialmode=1: ECM 拨号

ril.fibocom.dialmode=0: ppp 拨号

**注意：**

init.rc 文件中没有配置-w 参数，RIL 默认是 ppp 拨号方式，若要选择拨号方式，可以在 init.rc 文件中增加-w 参数，也可以在属性文件/system/build.prop 中增加属性项 ril.fibocom.dialmode，两种方式选其一即可。

3.7 加载 RIL 库文件

3.7.1 手动加载 RIL 库文件

将 libreference-ril.so 拷贝到/system/lib 目录, adb 终端执行操作如下：

```
adb reboot
adb remount
adb push <$android dir>/out/target/product/.../system/lib/libreference-ril.so /system/lib
```

在 android6 以后如果是 64 位系统，需要拷贝到/system/lib64

```
adb push <$android dir>/out/target/product/.../system/lib64/libreference-ril.so /system/lib64
```

3.7.2 自动打包 RIL 库文件

将 Fibocom 提供 RIL 库源码压缩包，解压到< android dir > /hardware/ril/reference-ril/目录下编译，编译系统固件的时候会自动编译 ril 库，并且会打包到 system.img 镜像文件中。

RIL 库源码可单独编译，操作如下：

```
cd <$android_dir>
source build/envsetup.sh
lunch <编译的项目>
mmm hardware/ril/reference-ril/
```

64 位系统生成 RIL 库 libreference-ril.so 文件目录

```
<$android dir>/out/target/product/.../system/lib64/libreference-ril.so
```

32 位系统生成 RIL 库 libreference-ril.so 文件目录

```
<$android dir>/out/target/product/.../system/lib/libreference-ril.so
```

编译生成的 RIL 库 libreference-ril.so，可参照章节 [3.5.1](#) 手动加载到 android 系统中。

4 NDIS 拨号方式

4.1 加载 Gobinet 驱动

NDIS 拨号需要加载 Gobinet 驱动，可参照文档《FIBOCOM NL668 应用指南_Linux 下 GobiNet 驱动加载》。

确认 GobiNet 驱动加载是否成功，可以使用如下命令查看是否有 usb0 节点

```
android:/ # lsmod
```

Module	Size	Used by
GobiNet	57152	0

```
android:/ # ifconfig -a
```

```
usb0      Link encap:Ethernet  HWaddr e6:e6:3a:93:72:2a
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 TX bytes:0
```

```
sit0      Link encap:IPv6-in-IPv4
          NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 TX bytes:0
```

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope: Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 TX bytes:0
```

4.2 获取 local ip,primary dns,second dns 地址

NDIS 拨号成功后,local ip,primary dns,second dns 会被写入到 Android OS 属性项 net.usb0.local-ip, net.usb0.dns1, net.usb0.dns2 中, 执行如下操作可获取:

- 1) 获取 local ip, 执行 Android OS 命令:

```
getprop net.usb0.local-ip
```

- 2) 获取 primary dns, second dns: 执行 Android OS 命令:

```
getprop net.usb0.dns1
```

```
getprop net.usb0.dns2
```



注意:

NDIS 拨号需要 linux 内核支持 DHCP 协议 (linux 内核默认支持)。

5 ppp 拨号方式

5.1 修改 ip-down ip-up 文件

修改 out/target/product/...../system/etc/ppp/目录下的 ip-up,ip-down 文件。

ip-up 文件修改如下:

```
#!/system/bin/sh

/system/bin/setprop "net.interfaces.defaultroute" "ppp0"
/system/bin/setprop "net.ppp0.dns1" "$DNS1"
/system/bin/setprop "net.ppp0.dns2" "$DNS2"
/system/bin/setprop "net.ppp0.local-ip" "$IPLOCAL"
/system/bin/setprop "net.ppp0.remote-ip" "$IPREMOTE"

exit 0
```

ip-down 文件修改如下：

```
#!/system/bin/sh
case $1 in
ppp1)
echo 0 > /proc/sys/net/ipv4/ip_forward;
;;
esac
rm /etc/ppp/ppp*.pid
# Use interface name if linkname is not available
NAME=${LINKNAME:-"$1"}
/system/bin/setprop "net.$NAME.local-ip" ""
/system/bin/setprop "net.$NAME.remote-ip" ""

/system/bin/setprop "net.interfaces.defaultroute" ""
/system/bin/setprop "net.ppp0.dns1" ""
/system/bin/setprop "net.ppp0.dns2" ""
/system/bin/setprop "net.ppp0.local-ip" ""
/system/bin/setprop "net.ppp0.remote-ip" ""
```

若安卓属性项不是 net.ppp0.xxx，则需要修改为对应 ppp 拨号端口，如 net.modem.xxx。

ppp 拨号成功后，IP 地址，primary dns，second dns 被写入到属性 net.ppp0.local-ip, net.ppp0.dns1, net.ppp0.dns2 中，执行如下操作可获取：

1) 获取 local ip: 执行 Android OS 命令”

```
getprop net.ppp0.local-ip
```

2) 获取 primary dns，second dns: 执行 Android OS 命令

```
getprop net.ppp0.dns1
```

```
getprop net.ppp0.dns2
```



注意：

ip-up,ip-down 文件需要具有执行权限 555。

6 APN 设置和拨号状态查询

在 APN 为空的情况下，需要设置 APN 才能拨号，有两种方式。

6.1 手动添加 APN

请执行如下操作(仅适用于调试)：

- ◆ 将 NL668/NL678 模块安装在开发板上
- ◆ 在开发板上插入 SIM 卡
- ◆ 打开电源，等待开机。
- ◆ 设置 **ppp** 拨号或者 **NDIS** 拨号用到的 APN：
 - 设置->无线和网线（更多）->移动网络->接入点名称（APN）
 - 按菜单键选择“新建 APN”，输入 Name，APN，MCC，MNC，按菜单键保存。
如果运营商有提供特殊 APN 需要鉴权用户名、密码时，请使用运营商提供的 APN、鉴权用户名、鉴权密码。
- ◆ 选用设置的 APN 拨号，可以使用以下命令确认 RIL 是否拨号成功：
 - 1) 查看 **usb0** 或者 **ppp0** 网络接口状态，执行 Android OS 命令

```
ifconfig
```

- 2) 获取 **ppp** 拨号或者 **NDIS** 拨号成功后的 local ip，执行 Android OS 命令

```
getprop net.ppp0.local-ip
```

```
getprop net.usb0.local-ip
```

6.2 配置 apns-conf.xml 文件

Android设备路径：/system/etc/apns-conf.xml

修改 apns-conf.xml 文件时，根据 SIM 卡的实际信息进行添加，添加的主要信息有：
apn,mcc,mnc,user,password,type,protocol 等。

以下是列举的某个 APN 的信息添加：

```
<apn carrier="China Mobile"  
    apn="cmnet"  
    mcc="460"  
    mnc="04"  
    user=""
```

```
server=""
password=""
proxy=""
port=""
mmsproxy=""
mmsport=""
mmsc=""
type="default,net,supl"
preferred="true"
localized_name="APN_NAME_CMNET"
protocol="IPV4V6"
roaming_protocol="IPV4V6"

/>
```

更新APN配置列表后，由于Android设备启动很多次，telephony.db已经初始化完成，因此需要先删除数据库文件，然后再次启动，才会加载新的配置文件。

Android设备上telephony.db的路径：

/data/data/com.android.providers.telephony/databases/telephony.db

7 RIL 调试方法

1) 获取 RIL log 的方法通过在 windows 操作系统的 cmd 窗口，输入如下命令：

```
adb logcat -b radio -v time
```

2) 获取 Android OS 运行 log，输入如下命令：

```
adb logcat -v time
```

3) 获取 ppp 拨号 pppd log，输入如下命令：

```
adb logcat -s pppd
```

4) 在一些特殊使用场景，如在许多设备上或很长一段时间内执行测试，可执行以下命令将 log 保存在 Android 系统指定的目录下：

```
adb shell
```

```
logcat -b radio -v time > <$android_dir>/<filename>
```

5) 提取设备中保存的 log 文件，执行如下命令：

```
adb pull <filename> <$local_directory>
```

8 LOG 中 TAG 标签说明

RIL log 中各个进程的 log 都在一个文件中，可以通过 tag 标签来区分 log 来自于那个文件，tag 标签与对应的文件，见表 4。

表 4，RIL log 中 TAG 标签说明

TAG	文件
RLOG-RIL	hardware/ril/reference-ril/reference-ril.c
RLOG-AT	hardware/ril/reference-ril/atchannel.c
RILC	hardware/ril/libril/ril.cpp
RILD	hardware/ril/rild/rild.c
RILJ	frameworks/opt/telephony/src/java/com/android/internal/telephony/RIL.java

9 常见问题

9.1 无法正常的进行数据业务

- 1、查询拨号状态，是否获取的 local ip , primary dns , second dns。
- 2、如果设备不拨号，请检测 Android 设置是否设置了 APN, APN 的设置可以通过 Andorid UI 界面设置，一般流程如下：

Setings -> Network& Internet -> More -> Mobile network -> Advanced -> Access Point Names (见章节 [6](#))。

9.2 电话打不通

如果电话打不通，确认 SIM 卡是否支持语音功能，如果支持需要向 Fibocom 技术支持人员确认模块在当前网络制式下是否支持电话功能。

9.3 无法发送短信

如果无法发送短信，确认 SIM 卡是否支持短信功能，如果支持需要向 Fibocom 技术支持人员确认模块在当前网络制式下是否支持短信功能。

9.4 RIL 没有正常工作

有很多原因导致 Android 设备无法启动 RIL，如果出现这类现象请按照以下流程进程排查：

- 1) RIL 库没有被正确的加载
- 进入 Android shell 终端：输入 `cat init.rc | grep ril` ；期待的结果"`service ril-daemon /vendor/bin/hw/rild -l /system/lib64/libreference-ril.so -- -w 0`" 的出现，如果没有请修改对应项目的 init.rc 以便加载正确的库文件。
- 2) RILD 是否正在运行
- 进入 Android shell 终端：输入 `getprop | grep init.svc.ril-daemon`；检测 RIL 的运行状态，期待的结果：`[init.svc.ril-daemon]: [running]`
- 3) USB 串口是否被枚举
- 进入 Android shell 终端：输入 `ls /dev/ttyUSB* -l` 以检测 usb 串口是否被枚举。如果没有枚举端口，RIL 也无法正常工作。

10 附录 A 参考文件

表 5，关联文件

文件名	备注
《FIBOCOM NL668 应用指南_Linux 下 GobiNet 驱动加载》	NDIS 拨号加载 GobiNet 驱动可做参考。

表 6，术语和缩写

缩写词	描述
GSM	全球移动通信系统
VID	供应商 ID
PID	产品识别码
RIL	无线接口层

本文件版权属深圳市广和通无线股份有限公司所有，未经批准，不得复制。

缩写词	描述
RILD	无线接口层守护进程
APN	接入点名称

FIBOCOM
Confidential