

The background of the top half of the page features a world map composed of small grey dots. The Fibocom logo, consisting of the word "Fibocom" in blue and "广和通" in black, is positioned in the upper left area of the map.

Fibocom 广和通

完 美 无 线 体 验

RIL集成指南_Android

V2.2

版权声明

版权所有©2021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

Fibocom 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。

联系方式

公司网址：<https://www.fibocom.com/>

总部地址：深圳市南山区西丽街道西丽社区打石一路深圳国际创新谷六栋A座10-14层

总机：+86 755-26733555

目录

修订记录.....	2
1. 引言.....	3
1.1. 概述.....	3
1.2. 适用型号.....	3
2. RIL集成配置.....	4
2.1. USB接口的option驱动配置.....	4
2.2. 设备加载检测.....	6
2.3. 修改RILD权限.....	7
2.4. 配置Android启动脚本.....	7
2.5. 加载RIL库文件.....	8
2.5.1. 手动加载RIL库文件.....	8
2.5.2. 自动打包RIL库文件.....	9
3. ECM拨号预置条件.....	10
4. APN设置和拨号状态查询.....	11
5. RIL调试方法.....	12
6. RIL扩展特性.....	13
7. 常见问题.....	14
7.1. 无法正常进行数据业务.....	14
7.2. RIL没有正常工作.....	14
8. TAG标签说明.....	16
附 A: 缩略语.....	17

修订记录

V2.2 (2021-10-27)	添加适用型号
V2.1 (2021-08-09)	新增RIL扩展特性章节
V2.0 (2021-07-28)	文档的版本号从三位变更为两位 删除ppp及Rmnet拨号方式 删除GobiNet及PCIe驱动集成指导
V1.0.4 (2021-05-12)	添加RIL支持的功能和支持的Android版本
V1.0.3 (2021-05-12)	添加使用产品型号 修改格式和拼写 修改文档中图片 完善文档中的描述
V1.0.2 (2021-05-07)	添加PCIe MHI驱动集成指导
V1.0.1 (2021-02-28)	修改格式和拼写
V1.0.0 (2021-02-06)	初始版本

1. 引言

本文主要介绍如何将RIL（无线接口层）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，使客户能正常在其设备上做数据业务。

1.1. 概述

本文主要介绍如何在客户的Android系统中修改option驱动，集成RIL及设置RIL启动服务，为Fibocom模块产品在客户的Android系统上正常使用提供指导。

Fibocom RIL的支持情况如下：

- 支持的功能：短信、数据业务
- 支持的Android版本：Android 5.x/6.x/7.x/8.x/9.x/10.x

1.2. 适用型号

表 1. 适用型号

序号	产品型号	说明
1	NL95X系列	M.2接口4G通信模组。
2	FG150系列	LGA封装，5G通信模组。
3	FM150系列	M.2接口5G通信模组。
4	NL668系列	--
5	FG621系列	LGA封装，4G通信模组。
6	FM650系列	M.2接口5G通信模组。
7	FG101系列	--
8	FM160系列	--
9	FM101系列	M.2接口模组。
10	FM100系列	M.2接口模组。
11	FG160系列	LGA封装，5G通信模组。
12	FG650系列	LGA封装，5G通信模组。

2. 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，其主要步骤有：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。

Fibocom会提供编译生成的RIL库文件libreference-ril.so供客户集成使用。

2.1. USB接口的option驱动配置

1. 打开内核源码文件**option.c**(路径一般为kernel/drivers/usb/serial/option.c)。
2. 在命令行状态下查找是否存在Fibocom vendor ID宏定义。
如果不存在请参考如下代码定义Fibocom vendor ID。

```
#define FIBOCOM_VENDOR_ID 0x2CB7
```

3. 在源码中找到option_ids数组，在数组中添加Fibocom产品的VID和PID，不同产品的VID和PID不同，需要根据实际情况添加下代码。

```
static const struct usb_device_id option_ids[] = {  
    /* Fibocom begin */  
    { USB_DEVICE(FIBOCOM_VENDOR_ID, 0x0001) }, /* L710 */  
    { USB_DEVICE(FIBOCOM_VENDOR_ID, 0x0103) }, /* NL678 */  
    { USB_DEVICE(FIBOCOM_VENDOR_ID, 0x0104) },  
    { USB_DEVICE(FIBOCOM_VENDOR_ID, 0x0105) },  
    { USB_DEVICE(FIBOCOM_VENDOR_ID, 0x0106) }, /* NL678 */  
    { USB_DEVICE(FIBOCOM_VENDOR_ID, 0x0107) },  
    { USB_DEVICE(FIBOCOM_VENDOR_ID, 0x0108) },  
    { USB_DEVICE(FIBOCOM_VENDOR_ID, 0x0109) },  
    { USB_DEVICE(FIBOCOM_VENDOR_ID, 0x0a04) }, /* Fibocom FG650 */  
    { USB_DEVICE(FIBOCOM_VENDOR_ID, 0x0a05) }, /* Fibocom FG650 */  
    { USB_DEVICE(0x1508, 0x1001) }, /* Fibocom NL668 */  
}
```

4. 在USB串口驱动中过滤adb接口。
请在option_probe函数中添加如下代码段对adb端口进行过滤。

```

if(serial->dev->descriptor.idVendor == FIBOCOM_VENDOR_ID &&
    ((serial->dev->descriptor.idProduct == cpu_to_le16(0x0104) &&
        serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4) ||
    (serial->dev->descriptor.idProduct == cpu_to_le16(0x0105) &&
        serial->interface->cur_altsetting->desc.bInterfaceNumber >= 5) ||
    (serial->dev->descriptor.idProduct == cpu_to_le16(0x0109) &&
        serial->interface->cur_altsetting->desc.bInterfaceNumber >= 2))
)
{
    printk(KERN_INFO "Discovery the interface for Fibocom .");
    return -ENODEV;
}

if(serial->dev->descriptor.idVendor == 0x1508 &&
    ((serial->dev->descriptor.idProduct == cpu_to_le16(0x1001) &&
        serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4))
)
{
    printk(KERN_INFO "Discovery the interface for Fibocom .");
    return -ENODEV;
}

```

5. 配置串口驱动的Linux内核。

- a. 执行`cd kernel`命令，进入到内核根目录。
- b. 执行`make menuconfig`命令。
- c. 在弹出的界面中依次选择：Device Drivers > USB support > USB Serial Converter support。

```

x x      Power management options  --->
x x      [*] Networking support  --->
x x      Device Drivers  --->
x x      File systems  --->

```

```

x x      <*> Sound card support  --->
x x      HID support  --->
x x      [*] USB support  --->
x x      <*> MMC/SD/SDIO card support  --->

```

```
x x          *** USB port drivers ***
x x          < * > USB Serial Converter support --->
x x          *** USB Miscellaneous drivers ***
```

d. 选中如下USB driver for GSM and CDMA modems组件后，保存退出。

```
x x          < > USB Xircom / Entegra Single Port Serial Driver
x x          < * > USB driver for GSM and CDMA modems
x x          < > USB ZyXEL omni.net LCD Plus Driver
```



关于Linux 内核配置以客户系统配置规则为准，本文描述的方法仅供参考。

2.2. 设备加载检测

USB模式下的端口

将上述patch修改后的代码编译烧录到目标设备，模块正常枚举后可通过lsusb命令查看对应的设备。



不同的产品型号PID/VID不同，以实际情况为准。

使用adb shell执行ls /dev/ttyUSB*命令，若设备正常挂载，将会有如下内容返回。

```
/dev/ttyUSB0
/dev/ttyUSB1
/dev/ttyUSB2
```

表 2. USB端口功能说明

设备名	作用	备注
ttyUSB0	DIAG	获取modem Log端口
ttyUSB1	MODEM	modem端口
ttyUSB2	AT	RIL程序发送AT命令请求和接收命令响应的端口

上述USB端口功能与AT+GTUSBMODE?的值有关。使用ECM方式需要使用AT+GTUSBMODE命令将usb端口模式切换到ECM。不同的模块需要查看对应模块的AT手册。如FG650模块USB端口组合如下：

```
mode: 34 0x2CB7 / 0x0A04 ECM+AT+DIAG+LOG
mode: 35 0x2CB7 / 0x0A04 ECM+AT+DIAG+LOG+ADB
mode: 36 0x2CB7 / 0x0A05 NCM+AT+MODEM+DIAG+LOG
mode: 37 0x2CB7 / 0x0A05 NCM+AT+MODEM+DIAG+LOG+ADB
mode: 38 0x2CB7 / 0x0A06 RNDIS+AT+MODEM+DIAG+LOG
mode: 39 0x2CB7 / 0x0A06 RNDIS+AT+MODEM+DIAG+LOG+ADB
```




如第一组USB端口的组合中：**34**表示mode模式，**0x2CB7**表示供应商ID，**0x0A04**表示产品识别码，**ECM+AT+DIAG+LOG**表示该模式下的端口组合形态。

FG650模块就可以使用AT+GTUSBMODE=35或AT+GTUSBMODE=34设置USB模式为ECM。其它模块产品请参考对应的AT手册。

2.3. 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
```

2.4. 配置Android启动脚本

为了让系统启动时启动rild进程，需要在Android启动脚本init.rc (<\$android dir> /system/core/rootdir/init.rc) 文件中添加ril-daemon进程。如下内容定义了rild启动时对应的选项。

```
service ril-daemon /system/bin/rild -l /vendor/lib/libreference-ril.so -- -w 1
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
```

Android 7.0之前的版本，该服务的内容是被定义在init.rc中的。到了Android 7.0之后，init.rc文件中的许多内容被移出，添加到各个进程中，rild服务被定义在rild.rc中，具体内容如下所示：

```
service ril-daemon /vendor/bin/hw/rild -l /vendor/lib/libreference-ril.so -- -w 1
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
```

Android 6.0以后的版本，若是64位系统，需要链接库/vendor/lib64/libreference-ril.so，其他语句不变。

```
service ril-daemon /system/bin/rild -l /vendor/lib64/libreference-ril.so -- -w 1
```

查看android cpu是32位还是64位可以使用如下语句：

```
adb shell getprop ro.product.cpu.abi
```

参数说明

- -l
指定RIL库加载路径。
- -w
指定拨号方式，1：ECM拨号。

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

```
ril.fibocom.dialmode=1 /* ECM拨号 */
```

2.5. 加载RIL库文件

2.5.1. 手动加载RIL库文件

将Fibocom提供RIL库libreference-ril.so拷贝到/vendor/lib目录，adb终端执行操作如下：

```
adb root
adb remount
adb push <$android_dir>/out/target/product/.../system/lib/libreference-ril.so
/vendor/lib
adb reboot
```

在Android 6.0以后如果是64位系统，需要拷贝到/vendor/lib64。

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

2.5.2. 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/目录下完成替换，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，adb终端执如下操作：

```
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[手动加载RIL库文件](#)章节手动加载到android系统中。

3. ECM拨号预置条件

根据[配置Android启动脚本](#)章节配置Android启动脚本或修改Android OS属性文件/system/build.prop中增加属性项ril.fibocom.dialmode。

Fibocom产品适配RIL ECM拨号需要使用AT+GTUSBMODE命令切换USB端口到ECM，参考[设备加载检测](#)章节。

4. APN设置和拨号状态查询

在特殊情况下，需要设置APN才能拨号，请执行如下操作：

1. 将Fibocom产品连接到Android设备上。
2. 在Android设备上插入SIM卡。
3. 打开电源，等待开机。
4. 设置拨号使用的APN。
 - a. 依次选择设置 > 无线和网线（更多） > 移动网络 > 接入点名称（APN）。
 - b. 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。
5. 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：
 - a. 执行如下Android OS命令，查看usb0网络接口状态。

```
netcfg
```

- b. 执行如下Android OS命令，获取拨号成功后的local ip。

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

5. RIL调试方法

1. 通过在Windows操作系统的cmd窗口，执行如下命令获取RIL Log。

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

2. 执行如下命令，获取android os运行Log。

```
adb logcat -v time
```

3. 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>
```

4. 执行如下命令，提取设备中保存的Log文件。

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

6. RIL扩展特性

1. 执行如下命令，将`ril.network.always5G`属性设置为`true`，可解决在低版本的Android系统没有优先注册5G网络的问题。

```
adb shell
setprop ril.network.always5G true
```

若不设置`ril.network.always5G`属性，则按照系统上层默认机制进行。若选择4G优先，实际下发4G优先命令。

2. 设置`ril.fibocom.netadapter`属性，适配自定义网卡名称。若用户对网卡名称进行修改，如修改为`modem`，可以执行如下命令进行适配。

```
adb shell
setprop ril.fibocom.netadapter modem
```

若不设置`ril.fibocom.netadapter`属性，RIL会根据模块VID在系统`/sys/bus/usb/devices`路径中读取网卡名称，若未读取到，则网卡名称为默认值`usb0`。

3. 设置`ril.fibocom.usbmode`属性，切换模块的USB端口模式。若用户需要的USB端口模式为18，可以执行如下命令进行适配。

```
adb shell
setprop ril.fibocom.usbmode 18
```

- 设置了该属性后，RIL流程中会对模块当前端口模式和此属性设置的端口模式进行比较，若一样则不做切换，不一样则切换为用户此属性设置的端口模式。
- 若不设置此属性，RIL会默认使用模块当前的端口模式（`AT+GTUSBMODE`）。



通过`setprop`命令设置的属性掉电不保存，若需要永久生效，需要将属性添加到`/system/build.prop`文件中，本章节中的属性可分别执行如下命令：

```
ril.network.always5G=true
ril.fibocom.netadapter=modem
ril.fibocom.usbmode=18
```

7. 常见问题

7.1. 无法正常进行数据业务

1. 使用adb shell登录到Android设备查询拨号状态，执行ifconfig命令查看是否获取local ip，若查询到usb0的网卡，且usb0网卡获取到ip地址，说明拨号成功。

```
rk3399pro:/ # ifconfig
usb0      Link encap:Ethernet  HWaddr 4e:6e:d0:51:86:2f  Driver cdc_ether
          inet addr:10.33.156.79  Bcast:10.33.156.255  Mask:255.255.255.0
          inet6 addr: fe80::4c6e:d0ff:fe51:862f/64 Scope: Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10 errors:0 dropped:0 overruns:0 frame:0
          TX packets:51 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1111 TX bytes:8309
```

2. 若设备拨号成功，但还是无法进行数据业务，请使用busybox route命令检查网关和ip是否在同一网段，使用getprop | grep dns命令检查dns是否设置正确。

```
rk3399_firefly_mid:/ # getprop | grep dns
getprop | grep dns
[net.dns1]: [61.134.1.6]
[net.dns2]: [218.30.19.40]
[net.usb0.dns1]: [61.134.1.6]
[net.usb0.dns2]: [218.30.19.40]
rk3399_firefly_mid:/ # busybox route
busybox route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
default          192.168.225.1   0.0.0.0          UG    0      0      0 usb0
192.168.225.0    *                255.255.255.0    U     0      0      0 usb0
```

3. 若设备不拨号，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：
依次选择Setings > Network&Internet > More > Mobile network > Advanced > Access Point Names。
4. 若APN已经设置，但拨号不成功，则需要执行如下命令设置mtu。

```
adb shell
ip link set dev usb0 mtu 1500
```

7.2. RIL没有正常工作

有很多原因导致Android设备无法启动RIL，请进入Android shell终端，按照以下流程进行排查。

1. RIL库没有被正确的加载。

执行`cat init.rc | grep ril`命令，期待的结果为“`service ril-daemon /vendor/bin/hw/rild -l /vendor/lib64/libreference-ril.so -- -w 1`”。若没有出现期待的结果，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。

执行`getprop | grep init.svc.ril-daemon`命令，检测RIL的运行状态，期待的结果为“`[init.svc.ril-daemon]: [running]`”。

3. USB串口是否被枚举。

执行`ls /dev/ttyUSB* -l`命令，检测usb串口是否被枚举。若没有枚举端口，则RIL也无法正常工作。

8. TAG标签说明

RIL各个进程的Log都在同一份Log文件中，可以通过tag标签来区分Log来自于哪个文件，tag标签与对应的文件如下表所示。

表 3. 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

附 A: 缩略语

表 4. 缩略语

缩写词	描述
GSM	Global System for Mobile Communications, 全球移动通信系统
VID	Vendor ID, 供应商ID
PID	Product ID, 产品识别码
RIL	Radio Interface Layer, 无线接口层
RILD	Radio Interface Layer Daemon, 无线接口层守护进程
APN	Access Point Name, 接入点名称