

GobiNet驱动集成及RmNet拨号指南\_Linux（桌面）

V2.3

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

G:\repos\cbb\zh\static\images\logo.png 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 3](#_Toc86747335)

[1 引言 5](#_Toc86747336)

[1.1 目的 5](#_Toc86747337)

[1.2 适用范围 5](#_Toc86747338)

[1.3 预置条件 5](#_Toc86747339)

[1.4 已适配的系统及内核版本 6](#_Toc86747340)

[2 Ubuntu环境下Host侧驱动编译环境搭建 8](#_Toc86747341)

[3 Fedora环境下Host侧驱动编译环境搭建 10](#_Toc86747342)

[3.1 Fedora系统信息 10](#_Toc86747343)

[3.2 Fedora 29/30下编译环境搭建 10](#_Toc86747344)

[4 OpenSUSE环境下Host侧驱动编译环境搭建 13](#_Toc86747345)

[4.1 OpenSUSE维护支持系统信息 13](#_Toc86747346)

[4.2 OpenSUSE Leap 15.0/15.1下编译环境搭建 13](#_Toc86747347)

[5 集成Fibocom模块option驱动信息 15](#_Toc86747348)

[5.1 配置Linux内核option串口驱动 15](#_Toc86747349)

[5.2 添加Fibocom模块VID/PID信息 15](#_Toc86747350)

[6 集成Fibocom模块GobiNet驱动信息 19](#_Toc86747351)

[6.1 GobiNet驱动代码结构 19](#_Toc86747352)

[6.2 GobiNet驱动中添加Fibocom模块VID/PID信息 19](#_Toc86747353)

[6.3 GobiNet驱动配置说明 20](#_Toc86747354)

[6.3.1 GobiNet驱动2.0.0.x以上版本 20](#_Toc86747355)

[6.3.1.1 AT模式 20](#_Toc86747356)

[6.3.1.2 QMI模式 21](#_Toc86747357)

[6.4 GobiNet驱动log打印部分说明 22](#_Toc86747358)

[7 Fibocom模块Host侧驱动编译 24](#_Toc86747359)

[7.1 usb\_driver\_option驱动编译 24](#_Toc86747360)

[7.2 GobiNet驱动编译 24](#_Toc86747361)

[8 Fibocom模块设备加载检测 26](#_Toc86747362)

[9 Fibocom模块拨号 27](#_Toc86747363)

[9.1 RMNET拨号参考流程图 27](#_Toc86747364)

[9.2 关键AT命令说明 29](#_Toc86747365)

[9.3 相关AT log和说明 30](#_Toc86747366)

[10 常见问题 33](#_Toc86747367)

[10.1 IP获取失败 33](#_Toc86747368)

[10.2 ping域名失败 33](#_Toc86747369)

[10.3 IP地址刷新失败 34](#_Toc86747370)

[10.4 网卡名称被修改 34](#_Toc86747371)

# 修订记录

|  |  |  |
| --- | --- | --- |
| V2.3（2021-11-02） | 增加适配模块型号，FM101系列，FM130系列和FM160系列 |  |
| V2.2（2021-10-09） | 修改“GobiNet驱动2.0.0.x以上版”章节中“切换AT模式与QMI模式”的描述  GobiNet驱动log等级改变，修改6.4节GobiNet驱动log打印部分说明 |  |
| V2.1（2021-08-25） | 完善5.2节添加模块pid/vid信息步骤。完善6.2节gobinet驱动添加VID/PID步骤  修改表8<umts\_profile>参数的说明，补充驱动代码结构文件,统一4.6节中粘贴代码与库上一致  增加章节10.常见问题 |
| V2.0（2021-06-26） | 修改书写有误的log打印命令（ehco改为echo）  增加“网卡down掉后再up起来无法获取到IP地址”的解决方法  对外发布文档新版本的版本号从三位数修改为两位数 |
| V1.0.9（2021-05-11） | 新增GobiNet2.1.0.x版本驱动配置 |
| V1.0.8（2021-02-20） | 新增GobiNet2.0.0.x版本驱动配置 |
| V1.0.7（2020-10-21） | 修改产品适用型号  新增所支持的系统及内核版本列表 |
| V1.0.6（2020-09-25） | 增加在网络断开的情况下，ip地址刷新不及时问题解决方案  新增模块dump和download模式时的适配 |
| V1.0.5（2020-09-22） | 增加log打印部分的说明 |
| V1.0.4（2020-05-12） | 修改格式和拼写  完善表格  文档review问题修改  修改图片格式 |
| V 1.0.3（2020-02-27） | 增加支持的产品信息  增加支持USB mode信息  文档review问题修改 |
| V 1.0.2（2020-01-09） | 增加支持的产品信息 |
| V 1.0.1（2019-09-09） | 根据评审建议修改文档 |
| V 1.0.0（2019-08-29） | 初始版本 |

# 引言

桌面Linux发行版本众多，Fibocom模块Host侧驱动基于Ubuntu、OpenSUSE、Fedora三大主流系统进行编译使用。

## 目的

本文主要介绍桌面Linux下Host侧option/GobiNet驱动在Ubuntu、OpenSUSE、Fedora编译环境搭建方法，为Fibocom模块产品在桌面Linux上正常使用option和GobiNet驱动提供指导。

## 适用范围

适用于测试部/市场导入人员参考文档或指导客户搭建嵌入式Linux Host驱动编译环境使用。

GobiNet驱动目前适用的产品型号如下表所示：

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 | NA |
| 5 | FM100系列 | M.2接口4G通信模组 |
| 6 | FG101系列 | LGA封装，5G通信模组。 |
| 7 | FM101系列 | M.2接口4G通信模组。 |
| 8 | FM130系列 | M.2接口5G通信模组。 |
| 9 | FM160系列 | M.2接口5G通信模组。 |

## 预置条件

1. PC需要安装下列的一种桌面Linux系统：
   * Ubuntu Desktop 16.04 LTS
   * Ubuntu Desktop 18.04 LTS
   * Ubuntu Desktop 19.04 LTS
   * OpenSUSE Leap 15.0
   * OpenSUSE Leap 15.1
   * Fedora 29 Workstation
   * Fedora 30 Workstation
2. PC已预置Fibocom模块。
3. 用户具有root操作权限。

## 已适配的系统及内核版本

当前已适配的系统及内核版本如下表所示，对于未列出的系统及内核版本，并非代表不支持，用户可直接进行集成验证。

表2. 已适配的系统及内核版本

|  |  |  |
| --- | --- | --- |
| 平台架构 | 操作系统 | 内核版本 |
| MT7621 | Openwrt | 4.14.180 |
| MT7621 | Openwrt | 3.10.108 |
| MT7621 | Openwrt | 3.10.14 |
| RTD1296 | Openwrt | 4.1.7 |
| INTEL X86\_64 | Centos 8.0 | 4.18.0 |
| INTEL X86\_64 | Centos 7.6 | 3.10.108 |
| INTEL X86\_64 | Ubuntu 12.04 | 3.13.0 |
| INTEL X86\_64 | Ubuntu 14.04 | 3.16.54 |
| INTEL X86\_64 | Ubuntu 16.04 | 4.15.0 |
| INTEL X86\_64 | Ubuntu 18.04 | 5.3.0 |
| NVIDIA AGX | Ubuntu 18.04 | 4.9.140 |
| INTEL X86\_64 | Ubuntu 20.04 | 5.4.0 |

# Ubuntu环境下Host侧驱动编译环境搭建

Ubuntu当前支持系统如下表所示，其中Ubuntu 16.04/19.04缺省情况下，默认安装了c/c++的编译环境，所以Ubuntu 16.04/19.04系统上，不需要安装额外编译环境，可以直接编译Fibocom模块Host侧驱动。

表3. Ubuntu系统版本信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ubuntu版本 | 发布日期 | 桌面版本支持结束时间 | 内核版本 | Make版本 | GCC版本 |
| 16.04 LTS | 2016-04-21 | 2021-04 | 4.4 | 4.1 | 5.4.0 |
| 18.04 LTS | 2018-04-26 | 2023-04 | 4.15 | NA | NA |
| 19.04 | 2019-04-18 | 2020-01 | 5.0 | 4.2.1 | 8.3.0 |

Ubuntu 18.04缺省情况下，未提供c/c++的编译环境，如下图示：

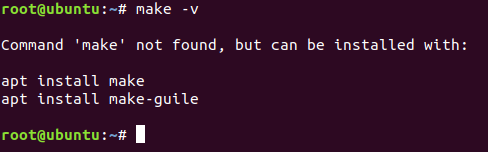


图1. 未提供make的编译环境

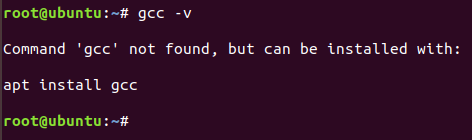


图2. 未提供gcc的编译环境

安装make：

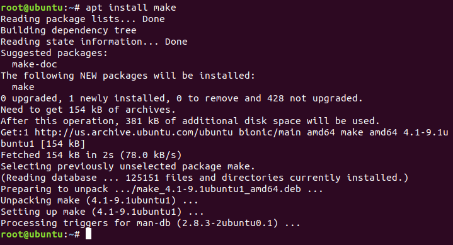


图3. 安装make

安装gcc工具：

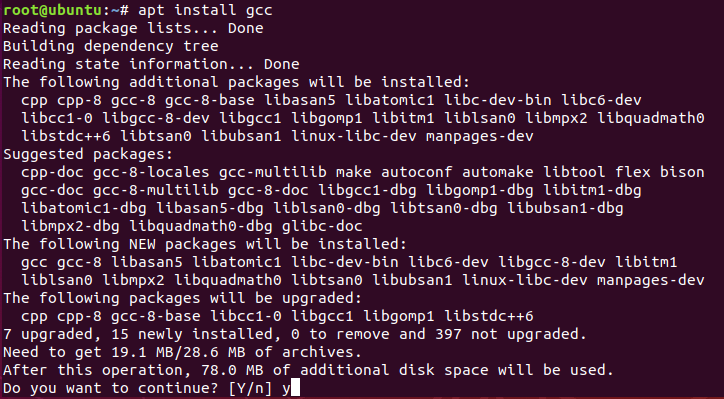


图4. 安装gcc工具

在安装gcc时，出现“Do you want to continue”提示后，输入**y**继续安装即可。

# Fedora环境下Host侧驱动编译环境搭建

## Fedora系统信息

Fedora仍被支持系统如下表所示，缺省情况下，未安装c/c++的编译环境，所以在Fedora系统上，需要安装额外编译环境后，再进行Fibocom模块Host侧驱动编译工作。

表4. Fedora系统版本信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fedora版本 | 发布日期 | 内核版本 | Make版本 | GCC版本 |
| Fedora 29 | 2018-10-30 | 4.19 | 8.2.1 | NA |
| Fedora 30 | 2019-04-29 | 5.0 | NA | NA |

## Fedora 29/30下编译环境搭建

**安装kernel开发包**：

1. 输入“**sudo -s**”切换到root模式。
2. 输入“**yum install kernel-devel-$(uname -r)**”命令，安装kernel开发包，如下图示：

出现“**Is this ok[y/N]**”时，输入“**y**”回车继续。

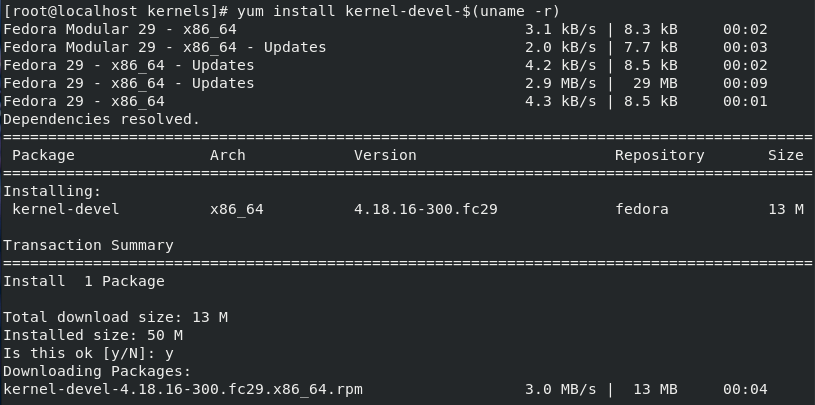


图5. 安装kernel开发包

|  |  |  |  |
| --- | --- | --- | --- |
|  | C:\Users\Administrator\Desktop\图层 4.png |  | 本文档基于Fedora 29搭建，用户请参照实际运行Fedora系统环境提示进行。 |

**安装elfutils-libelf-devel工具**：

1. 输入“**sudo -s**”切换到root模式。
2. 输入“**yum install elfutils-libelf-devel**”命令，安装elfutils-libelf-devel，如下图示：

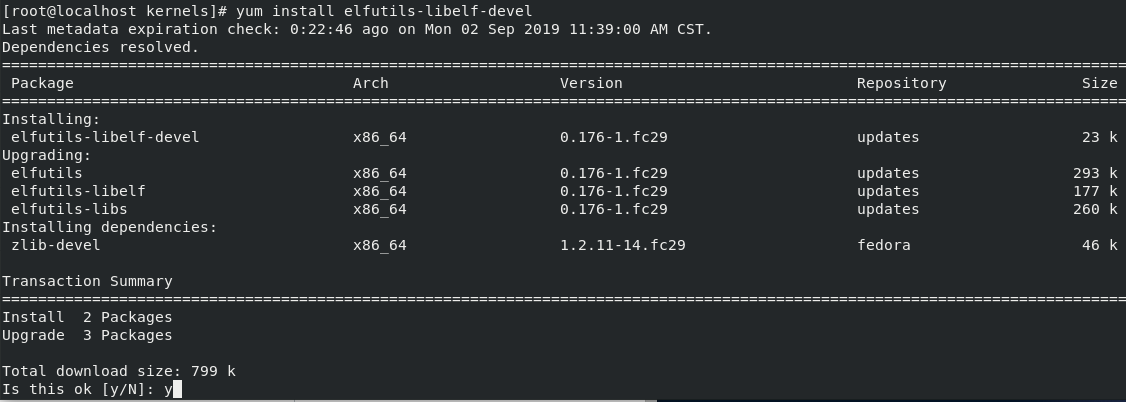


图6. 安装elfutils-libelf-devel

出现“**Is this ok[y/N]**”时，输入“**y**”回车继续。

**安装make工具**：

输入“**yum install make**”命令，安装make工具，如下图示：

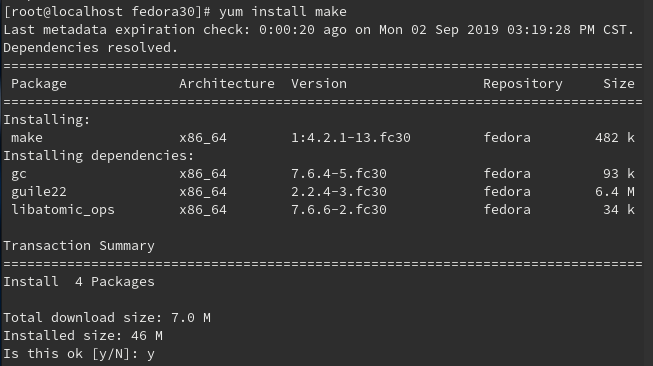


图7. 安装make工具

在出现“**Is this ok[y/N]**”时，输入“**y**”继续make工具的安装。

|  |  |  |  |
| --- | --- | --- | --- |
|  | C:\Users\Administrator\Desktop\图层 4.png |  | 上述工具安装完成后，请重启系统。 |

# OpenSUSE环境下Host侧驱动编译环境搭建

## OpenSUSE维护支持系统信息

OpenSUSE仍被维护支持的系统如表5所示，缺省情况下，未安装c/c++的编译环境，在OpenSUSE系统上，也需要安装额外编译环境后，才能进行Fibocom模块Host侧驱动编译。

表5. OpenSUSE维护支持系统版本信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OpenSUSE版本 | 发布日期 | 内核版本 | Make版本 | GCC版本 |
| OpenSUSE Leap 15.0 | 2018-05-25 | 4.12 | 4.2.1 | NA |
| OpenSUSE Leap 15.1 | 2019-05-22 | 4.12 | 4.2.1 | NA |

## OpenSUSE Leap 15.0/15.1下编译环境搭建

**安装kernel开发包**：

1. 输入“**sudo -s**”切换到root模式。
2. 输入“**zypper install kernel-default-devel**”命令，安装kernel开发包，如下图示：

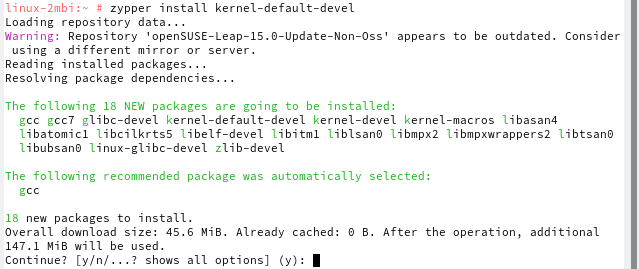


图8. 安装kernel开发包

出现“**Continue?[y/n/…]**”后输入“**y**”继续安装kernel module。

OpenSUSE Leap 15.0下安装完kernel开发包后，请确认/usr/src下安装的kernel开发包版本与系统内核版本是否一致。

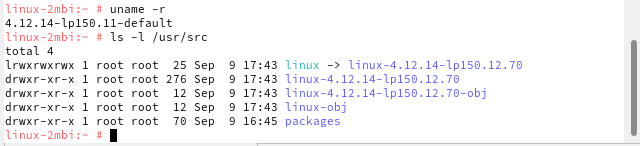


图9. 确认版本一致

如果不一致请使用“**zypper update**”命令同步进行系统更新，避免出现安装的kernel开发包与内核默认版本不匹配而出现编译失败问题。

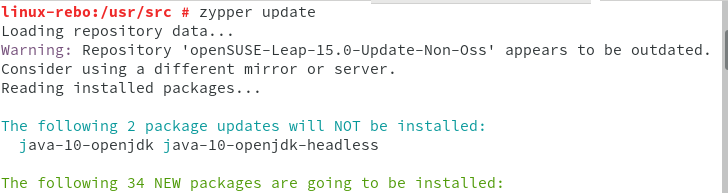


图10. 同步更新系统

|  |  |  |  |
| --- | --- | --- | --- |
|  | C:\Users\Administrator\Desktop\图层 4.png |  | 上述工具安装完成后，请重启系统。 |

# 集成Fibocom模块option驱动信息

## 配置Linux内核option串口驱动

Fibocom模块Host侧option驱动源码共包含3个文件，如下图所示，其中模块相关信息修改主要在option.c中完成。

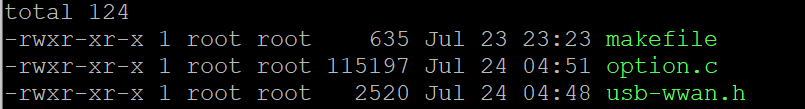


图11. option驱动源码

## 添加Fibocom模块VID/PID信息

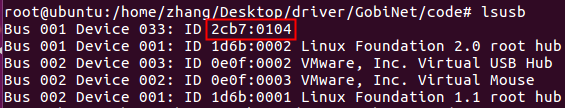
1. 打开option.c，在命令行状态下查找是否存在FIBOCOM\_VENDOR\_ID宏定义，如果不存在根据以下步骤进行添加。
2. 首先在命令行输入**lsusb**查询模块VID和PID信息，如下所示，模块信息为2CB7：0104

图12. lsusb查询模块信息

1. 参考如下代码定义Fibocom Vendor ID宏：#define FIBOCOM\_VENDOR\_ID 0x2CB7

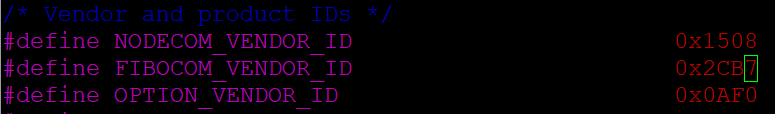


图13. 宏定义FIBOCOM\_VENDOR\_ID 0x2CB7

1. 在源码中找到option\_ids数组，在数组中添加Fibocom模块的VID(0x2CB7)和PID(0x0104)，如下代码示例：

static const struct usb\_device\_id option\_ids[ ] = {

//Fibocom begin

{ USB\_DEVICE(FIBOCOM\_VENDOR\_ID, 0x0104) },

{ USB\_DEVICE(FIBOCOM\_VENDOR\_ID, 0x0105) },

{ USB\_DEVICE(FIBOCOM\_VENDOR\_ID, 0x0106) },

{ USB\_DEVICE(FIBOCOM\_VENDOR\_ID, 0x0107) },

{ USB\_DEVICE(FIBOCOM\_VENDOR\_ID, 0x0108) },

{ USB\_DEVICE(FIBOCOM\_VENDOR\_ID, 0x0109) },

{ USB\_DEVICE(FIBOCOM\_VENDOR\_ID, 0x010A) },

{ USB\_DEVICE(FIBOCOM\_VENDOR\_ID, 0x010B) },

{ USB\_DEVICE(NODECOM\_VENDOR\_ID, 0x1000) },

{ USB\_DEVICE(NODECOM\_VENDOR\_ID, 0x1001) },

{ USB\_DEVICE(NODECOM\_VENDOR\_ID, 0x0110) },

{ USB\_DEVICE(NODECOM\_VENDOR\_ID, 0x0111) },

{ USB\_DEVICE(QUALCOMM\_VENDOR\_ID, 0x9025) },

{ USB\_DEVICE(QUALCOMM\_VENDOR\_ID, 0x90DB) },

{ USB\_DEVICE(0x2C7C, 0x0104) },

//FIBOCOM DUMP

{ USB\_DEVICE(QUALCOMM\_VENDOR\_ID, 0x0104) },

{ USB\_DEVICE(QUALCOMM\_VENDOR\_ID, 0x0105) },

{ USB\_DEVICE(QUALCOMM\_VENDOR\_ID, 0x0106) },

{ USB\_DEVICE(QUALCOMM\_VENDOR\_ID, 0x0107) },

{ USB\_DEVICE(QUALCOMM\_VENDOR\_ID, 0x0108) },

{ USB\_DEVICE(QUALCOMM\_VENDOR\_ID, 0x0109) },

{ USB\_DEVICE(QUALCOMM\_VENDOR\_ID, 0x010A) },

{ USB\_DEVICE(QUALCOMM\_VENDOR\_ID, 0x010B) },

{ USB\_DEVICE(QUALCOMM\_VENDOR\_ID, 0x900E) },

{ USB\_DEVICE(QUALCOMM\_VENDOR\_ID, 0x0110) },

{ USB\_DEVICE(QUALCOMM\_VENDOR\_ID, 0x0111) },

//FIBOCOM DOWNLOAD

{ USB\_DEVICE(QUALCOMM\_VENDOR\_ID, 0x9008) },

//Fibocom end

1. 在USB串口驱动中，需要过滤RMNET、ADB、ECM等接口，操作步骤如下。
2. 查询模块对应AT手册的+GTUSBMODE命令部分，支持RMNET拨号的模式有几种，分别对应那个端口，如下所示，参考FG150系列AT手册，支持RMNET拨号的模式有17,22,32，分别对应从0开始的4号端口，2号端口和4端口。

<mode>: integer type and supported mode depends on the target device and they may be as below, default value is 32:

17 Diag+Modem+AT+Pipe+RMNET+ADB

18 Diag+Modem+AT+Pipe+ECM+ADB

20 Modem

21 Modem+AT

22 Modem+AT+RMNET

23 Modem+AT+ECM

24 RNDIS+Modem+Diag+ADB

28 MBIM

29 MBIM+AT+Diag

30 MBIM+Modem+Diag+AT

31 Diag+Modem+RMNET+DPL+QDSS+ADB (Reserved QCOM for internal debug only)

32 Diag+Modem+AT+Pipe+RMNET

33 Diag+Modem+AT+Pipe+ECM

34 Modem+AT+Pipe+RMNET(reserve)

35 Modem+AT+Pipe+ECM(reserve)

1. 在option\_probe函数添加如下代码对上述interface number进行过滤，参考如下：

if(serial->dev->descriptor.idVendor == FIBOCOM\_VENDOR\_ID &&

(((serial->dev->descriptor.idProduct == cpu\_to\_le16(0x0104) ||

serial->dev->descriptor.idProduct == cpu\_to\_le16(0x0105)) &&

serial->interface->cur\_altsetting->desc.bInterfaceNumber >= 4)||

((serial->dev->descriptor.idProduct == cpu\_to\_le16(0x0109) ||

serial->dev->descriptor.idProduct == cpu\_to\_le16(0x010A)) &&

serial->interface->cur\_altsetting->desc.bInterfaceNumber >= 2)))

{

printk(KERN\_INFO "Discovery the interface for FIBOCOM .");

return -ENODEV;

}

if(((serial->dev->descriptor.idProduct == cpu\_to\_le16(0x0110) ||

serial->dev->descriptor.idProduct == cpu\_to\_le16(0x0111)) &&

serial->interface->cur\_altsetting->desc.bInterfaceNumber < 2))

{

printk(KERN\_INFO "Discovery the interface for FIBOCOM .");

return -ENODEV;

}

# 集成Fibocom模块GobiNet驱动信息

GobiNet驱动是高通产品的QUALCOMM CDC/ECM NDIS网卡驱动（也叫RMNET驱动）。

## GobiNet驱动代码结构

GobiNet驱动代码结构如下所示：

GobiNetDriver/

├──driverLoader.sh

├── GobiUSBNet.c

├── makefile

├── QMI.c

├── QMI.h

├── QMIDevice.c

├── QMIDevice.h

├── Structs.h

## GobiNet驱动中添加Fibocom模块VID/PID信息

1. 打开GobiUSBNet.c文件，在命令行状态下查找是否存在FIBOCOM\_VENDOR\_ID宏定义，如果不存在参考5.2章节并按照以下代码定义FIBOCOM\_VENDOR\_ID宏。

#define FIBOCOM\_VENDOR\_ID 0x2CB7

1. 在源码GobiUSBNet.c文件中找到GobiVIDPIDTable函数，在函数中添加Fibocom模块的VID(0x2CB7)和PID(0x0104)。

{

USB\_DEVICE( 0x2CB7, 0x0104 ),

.driver\_info = (unsigned long)&GobiNetInfo

},

{

USB\_DEVICE( 0x2CB7, 0x0109 ),

.driver\_info = (unsigned long)&GobiNetInfo

},

1. 在GobiUSBNet.c文件GobiNetDriverBind函数中，添加模块VID/PID以及过滤非NDIS interface接口信息。

else if (pDev->udev->descriptor.idVendor == FIBOCOM\_VENDOR\_ID

&& pDev->udev->descriptor.idProduct == cpu\_to\_le16(0x0104))

{

if (pIntf->cur\_altsetting->desc.bInterfaceNumber != 4)

return -ENODEV;

}

else if (pDev->udev->descriptor.idVendor == FIBOCOM\_VENDOR\_ID

&& pDev->udev->descriptor.idProduct == cpu\_to\_le16(0x0109))

{

if (pIntf->cur\_altsetting->desc.bInterfaceNumber != 2)

return -ENODEV;

}

## GobiNet驱动配置说明

### GobiNet驱动2.0.0.x以上版本

在GobiNet驱动2.0.0版本之后，为了优化驱动性能，增强驱动稳定性，我们将AT拨号与QMI的多路拨号分开了。增加了驱动参数用于在驱动加载阶段选择当前的驱动模式，下文描述如何使用及两种模式的建议使用场景。

#### AT模式

AT模式主要用于本文档第9章Fibocom模块拨号中的AT命令拨号。有两种方式可以用于设置该模式：

1. 修改源码

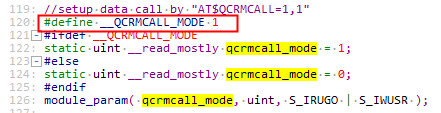
在编译驱动前，用户应该检查GobiNet驱动源码中GobiUSBNet.c文件，确保在代码中添加宏定义“\_\_QCRMCALL\_MODE”，如下图120行添加的代码所示：

图14. 添加宏定义“\_\_QCRMCALL\_MODE”

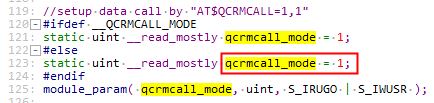
如果不添加宏定义，也可以考虑将123行qcrmcall\_mode设置为1，保证qcrmcall\_mode为1即可，如下图所示：

图15.设置qcrmcall\_mode为1

1. 修改驱动加载参数

如果在驱动编译前未修改GobiUSBNet.c文件，也可以在驱动加载时使用特定的驱动参数来设置驱动模式，作用是一样的。

insmod GobiNet.ko qcrmcall\_mode=1

#### QMI模式

QMI模式主要用于使用IP聚合和多路拨号下使用，在该模式下不支持AT命名拨号，拨号上网需要使用Fibocom-dial工具。具体使用可以参考文档《Fibocom QMI拨号工具使用指南\_Linux》。

使用IP聚合可以明显降低上位机的CPU占用率，提高数据收发效率，建议客户在上位机CPU或内存较小的情况下考虑使用。

有两种方式可以用于设置该模式：

1. 修改源码

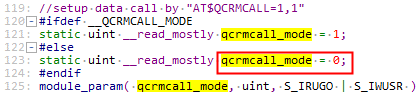
在编译驱动前，用户应该检查GobiNet驱动源码中GobiUSBNet.c文件，确保将qcrmcall\_mode设置为0，源码默认是支持QMI模式的，所以这里不需要修改源码，检查此处与源码保持一致即可。

图16. 设置qcrmcall\_mode为0

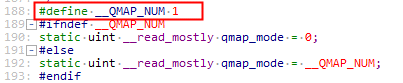
若使用IP聚合，将下图中188行的宏定义\_\_QMAP\_NUM设置为1（多路则设置为对应的路数，目前支持的最大路数为5路）。驱动源码此处的\_\_QMAP\_NUM为0，表示不适用IP聚合。

图17. 设置宏定义\_\_QMAP\_NUM

1. 修改驱动加载参数

如果在驱动编译前未修改GobiUSBNet.c文件，也可以在驱动加载时使用特定的驱动参数来设置驱动模式，作用是一样的。

insmod GobiNet.ko qcrmcall\_mode=0 qmap\_mode=1 // IP聚合

insmod GobiNet.ko qcrmcall\_mode=0 qmap\_mode=5 // 5路拨号

## 6.4 GobiNet驱动log打印部分说明

当前GobiNet驱动代码中，针对不同场景下的log打印设置了不同的打印级别，如QMI.h文件中所示，打印级别范围为1 ~ 3。以fibo\_debug参数值作为判断标准，只有小于或者等于fibo\_debug值的log打印会随驱动打印出来。

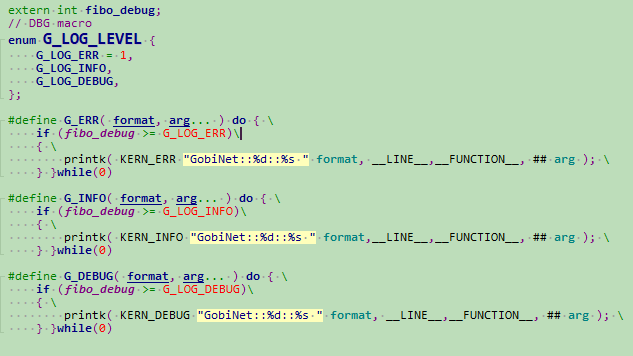


图18. Log打印设置

在GobiUSBNet.c文件中，fibo\_debug是作为模块参数，并默认初始化为G\_LOG\_ERR，即默认在驱动加载时G\_ERR的相关log会打印出来。

int fibo\_debug = G\_LOG\_ERR;

module\_param( fibo\_debug, int, S\_IRUGO | S\_IWUSR );

MODULE\_PARM\_DESC( fibo\_debug, "Debuging enabled or not" );

在驱动调试过程中，如果需要某一场景下的log，而其又不是默认能打出的情况下，可以发命令动态设置fibo\_debug参数进行处理：

禁用log打印：cho 0 > /sys/module/GobiNet/parameters/fibo\_debug

动态设置log打印（n范围为1 ~ 3）：echo n > /sys/module/GobiNet/parameters/fibo\_debug

# Fibocom模块Host侧驱动编译

## usb\_driver\_option驱动编译

进入usb\_driver\_option驱动源码目录，输入“**make install**”命令编译/安装option驱动。

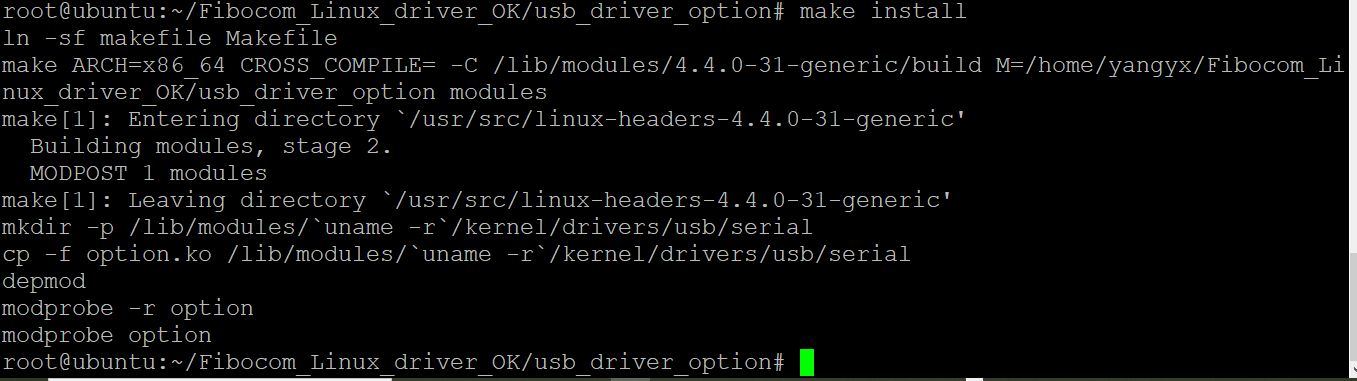


图19. 安装option驱动

输入“**lsmod | grep option**”，查看option驱动加载，驱动加载成功如下图示。

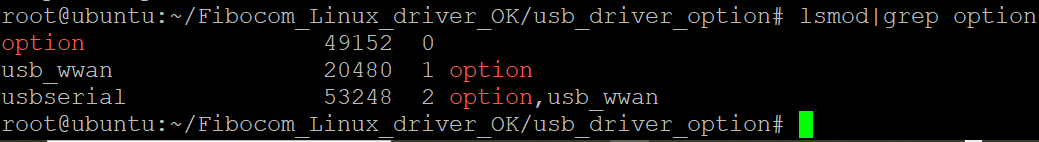


图20. 查看option驱动加载

## GobiNet驱动编译

进入Fibocom\_GobiNet\_LinuxDriver驱动目录，输入“**make install**”命令编译/安装GobiNet驱动，如下图示。

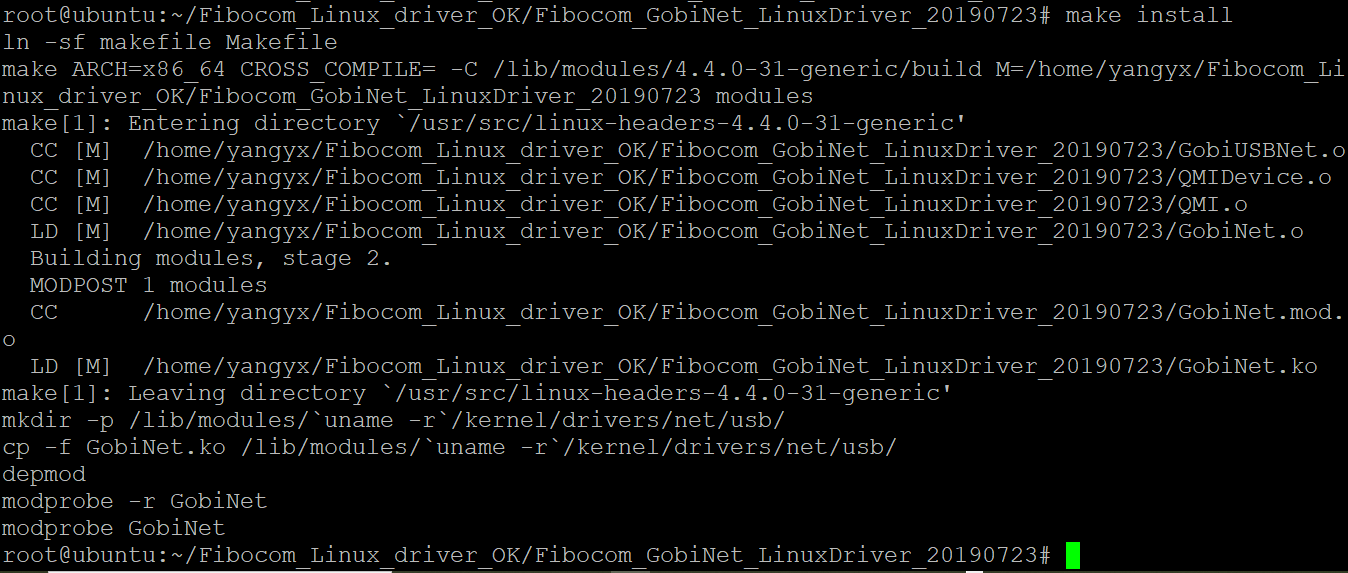


图21. 编译/安装GobiNet驱动

输入“**lsmod | grep GobiNet**”，查看GobiNet驱动加载，驱动加载成功如下图示。

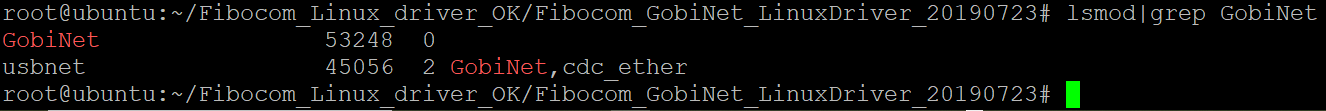


图22. 查看GobiNet驱动加载

# Fibocom模块设备加载检测

驱动加载成功后，插入模块，使用“**ls -l /dev/ttyUSB\***”命令查看模块端口信息，若设备正常挂载，将会有如下内容返回：

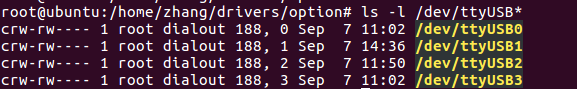


图23. 查看模块端口信息

表6. USB默认端口功能说明

|  |  |  |
| --- | --- | --- |
| 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 | RmNet | qmi拨号端口 |

# Fibocom模块拨号

## RMNET拨号参考流程图

模块开机

模块开机： 模块的供电稳定后，mini PCIe模块上电后开机。

模块关机：mini PCIe断电。推荐断电和再次上电间隔12秒以上

使用lsusb查询模块是否初始化完成。

查询模块是否初始化完成

最少等90秒。仍然没有端口则认为本次开机失败。重新进入模块开机动作。

或者延时5秒后连续给模块发AT，等模块回复OK。最长等90秒。

模块枚举出ttyUSB\*、网口

AT+CGSN? 查询IMEI，每个模块唯一

AT+CFSN? 查询批次号SN

AT+CFUN? 回复+CFUN: 1 确认模块正常工作模式。若不是1则设置成1

AT+CGMR? 查询模块当前的固件版本号

查询模块的基本信息

回复+CPIN: READY说明识别到SIM卡。

假如最少连续查询90秒仍然找不到SIM卡，复位模块。

如果需要PIN码则先用AT+CPIN按AT手册设置PIN码。

用AT+CPIN?命令确认SIM卡

用AT+CSQ命令确认信号指标正常

返回值+CSQ: <rssi>,<ber> 第一个参数 <rssi>大于0并且不是99。

假如连续查90秒返回值不正确，则复位模块。

查询结果是1或者5表示注册上PS域，连续查询超过90秒仍然注册不上PS域，复位模块。

用AT+CGREG? /AT+CEREG?确认已经注册到PS域/4G

APN是模块即将选用的运营商网络的标识。建议联系SIM供应商确认设置对应的APN。如果有多种SIM卡，可以在注册PS域后，设置AT+COPS=3,2 后根据AT+COPS? 查询结果设置。

AT+CGDCONT设置APN

用AT$QCRMCALL=1,1,3 激活。使用AT$QCRMCALL?命令确认是否激活，激活时返回结果：$QCRMCALL: 1,“IP类型”

连续尝试5次AT$QCRMCALL?后仍然拨号失败则回到AT+CPIN？

连续尝试5次拨号失败则复位一次。

AT$QCRMCALL=1,1,3

拨号失败

拨号成功

MCU侧DHCP实现IP、DNS 、路由配置

数据中断时尝试重发90秒仍无法通信。

用AT$QCRMCALL=0,1,3 释放。使用AT$QCRMCALL?命令确认是否释放，期望结果：OK

连续尝试5次AT$QCRMCALL?仍然失败则认为无法释放，复位。

MCU和网络数据收发

## 关键AT命令说明

AT$QCRMCALL命令格式如下表：

表7. AT$QCRMCALL命令格式

| 命令 | 结果 |
| --- | --- |
| $QCRMCALL=<Action>,<Instance>[,<IP Type> [,<Tech Pref >[,<umts profile number> [,<cdma profile number > [,<APN> ]]]]] | $QCRMCALL: <Instance>,<IP Type>  OK  or  ERROR/+CMS ERROR: <err>  or  NO CARRIER |
| $QCRMCALL? | $QCRMCALL: <Instance>,<IP Type>  OK  or  ERROR/+CMS ERROR: <err> |
| $QCRMCALL=? | $QCRMCALL: (list of supported <Action>s),(list of supported <Instance>s),(list of supported <IP Type>s),(list of supported <Tech Pref>s),(list of supported <umts profile number>s), (list of supported <cdma profile number>s),(max length of supported <APN>)  OK |

表8. 命令参数说明

| 参数 | 说明 |
| --- | --- |
| <Action> | 整型；值可能是：   * 0：Stop * 1：Start |
| <Instance> | 整型；值是1。 |
| <IP Type> | 整型；值可能是：   * 1：Ipv4 * 2：Ipv6 * 3：Ipv4v6 |
| <Tech Pref> | 整型；值可能是：   * 1：3GPP2 * 2：3GPP |
| <umts\_profile> | 整型；指定特定的 PDP 上下文定义（请参阅 +CGDCONT 命令）。 |
| <cdma profile number> | 整型；值的范围从100到179 |
| <APN> | 字符串类型；最大长度为100 |

## 相关AT log和说明

如果需要上网，拨号流程推荐（以电信卡为例），具体如下：

AT+CPIN?

+CPIN: READY //确保SIM卡就绪

OK

AT+CSQ

+CSQ: 21,99 //确保模块能接收到信号

OK

AT+COPS?

+COPS: 0,0,"CHN-TELECOM",7 //确保模块注册上运营商网络

OK

AT+CGREG?

+CGREG: 0,1 //模块附着数据网络

OK

AT+CEREG? //模块附着LTE网络

+CEREG: 0,1

OK

AT+CGDCONT=1,"IPV4V6","CTLTE" //设置接入点信息

OK

AT+CGDCONT?

+CGDCONT: 1,"IPV4V6","CTLTE","0.0.0.0",0,0,0,0 //查询是否设置成功

OK

AT$QCRMCALL=1,1,3 //发起拨号

$QCRMCALL: 1, V4

$QCRMCALL: 1, V6

OK

……

AT$QCRMCALL?

$QCRMCALL: 1, V4

$QCRMCALL: 1, V6 //定时查询拨号状态，若已断开，则重新发起拨号

OK

……

拨号后，在终端窗口使用ifconfig查看可知，网卡已获取到IP地址，并可通过该网卡进行数据业务，故拨号成功：

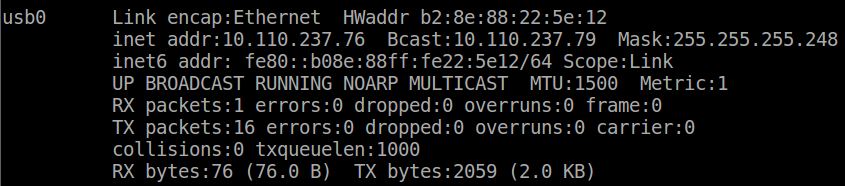


图24. 查看网卡信息

如果嵌入式Linux 未能获取IP，请发送“udhcpc -i usb0” 为系统配置IP地址：

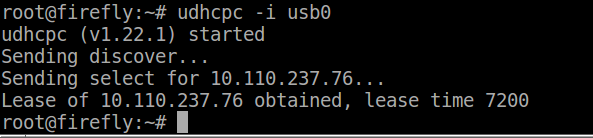


图25. 手动配置IP地址

如需断开网络，发送如下命令：

AT$QCRMCALL=0,1,3

OK

AT$QCRMCALL ?

OK

# 常见问题

## IP获取失败

如果拨号成功后手动禁用再启用网卡，无法获取到IP地址，请在AT口发送如下命令，重启模块：

AT+CFUN=15

然后手动关闭系统网络服务：

service network-manager stop

然后使用AT方式进行rmnet拨号，在拨号成功后，手动申请IP地址：

udhcpc -i 网卡名

进行ping网测试：

ping [www.baidu.com](http://www.baidu.com)

ping 8.8.8.8

手动禁用再启用网卡：

ifconfig 网卡名 down

ifconfig 网卡名 up

可以成功申请对应网卡ip地址并且可以ping通。

## ping域名失败

继10.1章节的问题，如果成功获取IP地址，ping 8.8.8.8可以成功，但是无法ping [www.baidu.com](http://www.baidu.com)并且无法上网。请执行下列操作：

chmod 777 /etc/resolv.conf

vi /etc/resolv.conf

进入/etc/resolv.conf文件之后，输入“i”，进入insert模式，添加如下代码：

nameserver 8.8.8.8

依次输入“Esc”，“:”，“wq”，“回车”即可，重新再次ping域名即可成功，并且可以上网。

## IP地址刷新失败

如果在断开网络连接的时候ip地址未及时刷新，需要使用如下命令进行刷新：

ip -4 addr flush dev xxx（xxx为网卡接口名称）

ip -6 addr flush dev xxx（xxx为网卡接口名称）

## 网卡名称被修改

在较新的linux内核中，host会自动修改网口名称，但是这种自动修改又是不确定的，这样会导致拨号成功后网卡可能操作失败，无法分配ip等信息，处于不可控状态。通过下面方法可防止网卡自动修改。

1. 执行sudo vim /etc/default/grub
2. 找到GRUB\_CMDLINE\_LINUX，增加两个参数net.ifnames=0 biosdevname=0

GRUB\_CMDLINE\_LINUX=”net.ifnames=0 biosdevname=0”

1. 执行sudo grub-mkconfig -o /boot/grub/grub.cfg 重启后网卡名称就不会自动修改了。