UMTS LTE 5G Linux USB Driver 用户指导

UMTS/HSPA+/LTE/5G 模块系列

版本：1.1

日期：2024-04-19

状态：受控文件

上海移远通信技术股份有限公司（以下简称“移远通信”）始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市闵行区田林路 1016 号科技绿洲 3 期（B 区）5 号楼　邮编：200233
电话：+86 21 5108 6236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：http://www.quectel.com/cn/support/sales.htm。

如需技术支持或反馈我司技术文档中的问题，请随时登录网址：
http://www.quectel.com/cn/support/technical.htm 或发送邮件至：support@quectel.com。

前言

移远通信提供该文档内容以支持客户的产品设计。客户须按照文档中提供的规范、参数来设计产品。同时，您理解并同意，移远通信提供的参考设计仅作为示例。您同意在设计您目标产品时使用您独立的分析、评估和判断。在使用本文档所指导的任何硬软件或服务之前，请仔细阅读本声明。您在此承认并同意，尽管移远通信采取了商业范围内的合理努力来提供尽可能好的体验，但本文档和其所涉及服务是在“可用”基础上提供给您的。移远通信可在未事先通知的情况下，自行决定随时增加、修改或重述本文档。

使用和披露限制

许可协议

除非移远通信特别授权，否则我司所提供硬软件、材料和文档的接收方须对接收的内容保密，不得将其用于除本项目的实施与开展以外的任何其他目的。

版权声明

移远通信产品和本协议项下的第三方产品可能包含受移远通信或第三方材料、硬软件和文档版权保护的相关资料。除非事先得到书面同意，否则您不得获取、使用、向第三方披露我司所提供的文档和信息，或对此类受版权保护的资料进行复制、转载、抄袭、出版、展示、翻译、分发、合并、修改，或创造其衍生作品。移远通信或第三方对受版权保护的资料拥有专有权，不授予或转让任何专利、版权、商标或服务商标权的许可。为避免歧义，除了正常的非独家、免版税的产品使用许可，任何形式的购买都不可被视为授予许可。对于任何违反保密义务、未经授权使用或以其他非法形式恶意使用所述文档和信息的违法侵权行为，移远通信有权追究法律责任。

商标

除另行规定，本文档中的任何内容均不授予在广告、宣传或其他方面使用移远通信或第三方的任何商标、商号及名称，或其缩略语，或其仿冒品的权利。

第三方权利

您理解本文档可能涉及一个或多个属于第三方的硬软件和文档（“第三方材料”）。您对此类第三方材料的使用应受本文档的所有限制和义务约束。

移远通信针对第三方材料不做任何明示或暗示的保证或陈述，包括但不限于任何暗示或法定的适销性或特定用途的适用性、平静受益权、系统集成、信息准确性以及与许可技术或被许可人使用许可技术相关的不侵犯任何第三方知识产权的保证。本协议中的任何内容都不构成移远通信对任何移远通信产品或任何其他硬软件、设备、工具、信息或产品的开发、增强、修改、分销、营销、销售、提供销售或以其他方式维持生产的陈述或保证。此外，移远通信免除因交易过程、使用或贸易而产生的任何和所有保证。

隐私声明

为实现移远通信产品功能，特定设备数据将会上传至移远通信或第三方服务器（包括运营商、芯片供应商或您指定的服务器）。移远通信严格遵守相关法律法规，仅为实现产品功能之目的或在适用法律允许的情况下保留、使用、披露或以其他方式处理相关数据。当您与第三方进行数据交互前，请自行了解其隐私保护和数据安全政策。

免责声明

1) 移远通信不承担任何因未能遵守有关操作或设计规范而造成损害的责任。
2) 移远通信不承担因本文档中的任何因不准确、遗漏、或使用本文档中的信息而产生的任何责任。
3) 移远通信尽力确保开发中功能的完整性、准确性、及时性，但不排除上述功能错误或遗漏的可能。除非另有协议规定，否则移远通信对开发中功能的使用不做任何暗示或法定的保证。在适用法律允许的最大范围内，移远通信不对任何因使用开发中功能而遭受的损害承担责任，无论此类损害是否可以预见。
4) 移远通信对第三方网站及第三方资源的信息、内容、广告、商业报价、产品、服务和材料的可访问性、安全性、准确性、可用性、合法性和完整性不承担任何法律责任。

文档历史

修订记录

版本	日期	作者	变更表述
-	2022-04-20	Leif WU	文档创建
1.0	2023-02-01	Jerry MENG	受控版本
1.1	2024-04-19	Hunter LV/ Aaron LIU/ Ozzy ANG	1. 更新适用模块： ● 新增 EC100M-CN、EC600K-CN、EC800K-CN、EG800K 系列、EG810M 系列、EG950A 系列、EG800G 系列、EC300R-LA、EC200R 系列、RG525F-NA、RM521F-GL、RG620T 系列、RG255C 系列、RM255C-GL、AG18、AG215S 系列、RG650E 系列、RG650V 系列、EG915Q 系列和 EG916Q-GL。 ● 更新 EC200N-CN 为 EC200N 系列、EC600M-CN 为 EC600M 系列、EC600G-CN 为 EC600G 系列、RG200U-CN 为 RG200U 系列、RM500U-CN 为 RM500U 系列、EG120K-EA 为 EG120K 系列、EM060K-GL 为 EM060K 系列、EG800Q-EU 为 EG800Q 系列。 ● 删除 EOL 项目 EC20-CN、EG912Y-CN 和 EC100Y-EL。 2. 更新 USB 接口信息（表 1）： ● 更新 EM05 系列的 PID； ● 更新 RG500L 系列的 USB 驱动信息； ● 更新 EG800Q-EU 的 USB 驱动信息。 3. 新增有关 USBFS 驱动的备注（第 2 章）。 4. 新增使用 GobiNet 驱动的条件（第 3.3 章）。 5. 新增使用 QMI_WWAN 驱动的条件（第 3.4 章）。

目录

文档历史 3
目录 4
表格索引 6
图片索引 7

1 引言 8

2 Linux USB 驱动概述 9

3 系统设置 18
3.1. Linux USB 驱动架构 18
3.2. USB 转串口驱动 19
3.2.1. 添加 VID 和 PID 19
3.2.2. 使用 USBNet 驱动 19
3.2.3. 修改内核配置 20
3.2.4. 添加零包机制 21
3.2.5. 添加 Reset-resume 机制 22
3.2.6. 增加批量输出 URB 的数量和容量 22
3.3. GobiNet 驱动 23
3.3.1. 修改驱动源码 23
3.3.2. 修改内核配置 23
3.4. QMI_WWAN 驱动 23
3.4.1. 修改驱动源码 24
3.4.2. 修改内核配置 24
3.5. ACM/ECM/RNDIS/NCM/MBIM 驱动 24
3.6. 支持 PPP 25
3.7. 配置内核 25
3.8. 以内核模块的形式在 Linux PC 上安装和加载驱动 26

4 测试模块 28
4.1. 测试 AT 功能 28
4.2. 测试 PPP 功能 28
4.3. 测试 GobiNet/QMI_WWAN 驱动 31
4.4. 在 GobiNet/ QMI_WWAN 驱动上测试 AT$QCRMCALL 34
4.5. 在 GobiNet/QMI_WWAN 驱动上测试 QMAP 35
4.6. 测试 ECM/RNDIS/NCM/MBIM 驱动 35

5 电源管理 36
5.1. 启用 USB 自动挂起 36
5.2. 启用 USB 远程唤醒 37

6 常见问题和内核 Log 38
6.1. 如何检查设备中是否存在 USB 驱动 38
6.2. 如何检查模块与对应的 USB 驱动程序是否正常工作 39

6.3. 如何检查已安装的 USB 驱动程序 40
7 附录 参考文档及术语缩写 41

表格索引

表 1：适用模块和 USB 接口信息 9
表 2：USB 类驱动的配置项 25
表 3：参考文档 41
表 4：术语缩写 41

图片索引

图 1：Linux USB 驱动架构 18
图 2：在内核中配置 USB 串口 26
图 3：AT 命令测试结果 28
图 4：RG502Q 系列模块 USB 转串口 Option 和 GobiNet Log 39
图 5：RG502Q 系列模块 USB 转串口 Option 和 qmi_wwan_q Log 39
图 6：RG502Q 系列模块 USB 接口和驱动 40

1 引言

本文档介绍如何在 Linux 系统上移植移远通信 UTMS&LTE&5G 模块的 USB 驱动以及在成功移植 USB 驱动后如何对模块进行测试。USB 驱动包括 USB 串口驱动（option 和 ACM）和 USBNet 驱动（GobiNet、QMI_WWAN、MBIM、NCM、RNDIS 和 ECM）。

2 Linux USB 驱动概述

移远通信UMTS&LTE&5G模块是包含多个USB接口的USB复合设备。每个USB接口通过加载不同的USB接口驱动实现不同的功能。Linux系统可以通过驱动加载成功后生成的设备节点实现模块功能，如AT命令、GNSS、DIAG、Log和USB网络适配器等。

下表包含不同模块在 Linux 系统中的 USB 接口信息，包括 USB 驱动、接口号、设备节点名称和接口功能。用户可以根据模块型号查看对应的 VID、PID 和接口信息，然后移植表中对应的 USB 接口驱动。

表 1：适用模块和 USB 接口信息

模块 VID 和 PID	USB 驱动	接口号	设备节点名称	功能
EC20-CE/ EC25 系列/ EG25-G/ EG25-GL： VID: 0x2c7c PID: 0x0125	USB 转串口 option	0	/dev/ttyUSB0	DIAG
		1	/dev/ttyUSB1	GNSS
		2	/dev/ttyUSB2	AT 命令
		3	/dev/ttyUSB3	Modem
EM05 系列： VID：0x2c7c PID：0x030E **EC21 系列/ EG21-G：** VID：0x2c7c PID：0x0121	GobiNet	4	usb0 /dev/qcqmi0	USB 网络适配器 通过 **AT+QCFG="usbnet",0** 配置网卡接口类型为 RmNet
EG91 系列： VID：0x2c7c PID：0x0191 **EG95 系列：**	QMI_WWAN	4	wwan0 /dev/cdc-wdm0	USB 网络适配器 通过 **AT+QCFG="usbnet",0** 配置网卡接口类型为 RmNet

VID：0x2c7c PID：0x0195 **AG35 系列：** VID：0x2c7c PID：0x0435 **EG06 系列/** **EP06 系列/** **EM06 系列：** VID：0x2c7c PID：0x0306 **EG12 系列/** **EM12-G/** **EG18 系列：** VID：0x2c7c PID：0x0512 **EG512R-EA/** **EM160R-GL/** **EM120R-GL/** **EM121R-GL：** VID：0x2c7c PID：0x0620 **RG500Q 系列/** **RM500Q 系列/** **RG501Q-EU/** **RG502Q 系列/** **RM502Q-AE/** **RM505Q-AE/** **RM510Q-GL/** **RG500S-CE/** **RM500S-CE：** VID：0x2c7c PID：0x0800	MBIM	4 5	wwan0 /dev/cdc-wdm0	USB 网络适配器 通过 **AT+QCFG="usbnet",2** 配置网卡接口类型为 MBIM
BG95 系列/ **BG77/** **BG600L-M3：** VID：0x2c7c PID：0x0700	USB 转串口 option	0	/dev/ttyUSB0	DIAG
		1	/dev/ttyUSB1	GNSS
		2	/dev/ttyUSB2	Modem
		4	/dev/ttyUSB3	通过 **AT+QCFGEXT="usbnet","mode**

				m"将 USB 组合配置为调制解调器接口模式 对应模块 Modem USB 组合：USB DM + NEMA + Modem + Modem
	ECM	3 4	usb0	通过 **AT+QCFGEXT="usbnet","ecm"** 将 USB 组合配置为 ECM 接口模式 对应 ECM USB 组合：USB DM + NEMA + Modem + ECM
BG96： VID：0x2c7c PID：0x0296	USB 转串口 option	0	/dev/ttyUSB0	DIAG
		1	/dev/ttyUSB1	GNSS
		2	/dev/ttyUSB2	AT 命令
		3	/dev/ttyUSB3	Modem
	QMI_WWAN	4	wwan0 /dev/cdc-wdm0	RmNet
EC100N-CN/ **EC100M-CN/** **EC200S 系列/** **EC200N 系列/** **EC200M-CN/** **EC600M 系列/** **EC600N-CN/** **EC600K-CN/** **EC800M-CN/** **EC800N-CN/** **EC800K-CN/** **EG800K 系列/** **EG810M 系列/** **EG912N-EN/** **EG915N 系列：** VID：0x2c7c PID：0x6002 **EC200A 系列（RTOS）/** **EG950A 系列：** VID：0x02c7c PID：0x6005	ECM/ RNDIS	0 1	usb0	通过 **AT+QCFG="usbnet",1** 配置网卡接口类型为 ECM 通过 **AT+QCFG="usbnet",3** 配置网卡接口类型为 RNDIS
	USB 转串口 option	2	/dev/ttyUSB0	DIAG
		3	/dev/ttyUSB1	AT 命令
		4	/dev/ttyUSB2	Modem

UC200A-GL： VID：0x02c7c PID：0x6006 **EG912Y-EU/** **EC100Y-CN：** VID：0x02c7c PID：0x6001 **EC600E-CN/** **EC800E-CN：** VID：0x02c7c PID：0x0903				
EC200U 系列/ **EC600U 系列/** **EG700U-CN/** **EG500U-CN/** **EG912U-GL/** **EG915U 系列：** VID：0x2c7c PID：0x0901	ECM/ RNDIS	0	usb0	通过 **AT+QCFG="usbnet",1** 配置网卡接口类型为 ECM
		1		通过 **AT+QCFG="usbnet",3** 配置网卡接口类型为 RNDIS
	USB 转串口 option	2	/dev/ttyUSB0	AT 命令
		3	/dev/ttyUSB1	DIAG
		4	/dev/ttyUSB2	MOS
		5	/dev/ttyUSB3	CP log
		6	/dev/ttyUSB4	AP log
		7	/dev/ttyUSB5	Modem
		8	/dev/ttyUSB6	GNSS
EC200D 系列： VID：0x2c7c PID：0x0902	ECM/ RNDIS	0	usb0	通过 **AT+QCFG="usbnet",1** 配置网卡接口类型为 ECM
		1		通过 **AT+QCFG="usbnet",3** 配置网卡接口类型为 RNDIS
	USB 转串口 option	2	/dev/ttyUSB0	AT 命令
		3	/dev/ttyUSB1	DIAG
		4	/dev/ttyUSB2	WCN log
		5	/dev/ttyUSB3	AP log、Modem log
		6	/dev/ttyUSB4	Modem

		7	/dev/ttyUSB5	AT 命令
EC200G-CN/ EC600G 系列/ EG700G-CN/ EC800G-CN/ EG800G 系列： VID：0x2c7c PID：0x0904	ECM/ RNDIS	0	usb0	通过 **AT+QCFG="usbnet",1** 配置网卡接口类型为 ECM
		1		通过 **AT+QCFG="usbnet",3** 配置网卡接口类型为 RNDIS
	USB 转串口 option	2	/dev/ttyUSB0	AT 命令
		3	/dev/ttyUSB1	DIAG/AP log
		4	/dev/ttyUSB2	调试串口，目前无实际作用
		5	/dev/ttyUSB3	CP log
		7	/dev/ttyUSB4	Modem/AT 命令
		8	/dev/ttyUSB5	GNSS/AT 命令（仅后缀为 GA 的模块型号支持）
EG060V-EA/ EG060W-EA： VID：0x2c7c PID：0x6004 **EC200A 系列（Linux QuecOpen）/ EC300R-LA/ EC200R 系列：** VID：0x2c7c PID：0x6005	ECM/ RNDIS/ NCM	0	usb0	通过 **AT+QCFG="usbnet",1** 配置网卡接口类型为 ECM
		1		通过 **AT+QCFG="usbnet",3** 配置网卡接口类型为 RNDIS 通过 **AT+QCFG="usbnet",4** 配置网卡接口类型为 NCM
	USB 转串口 option	2	/dev/ttyUSB0	DIAG
		3	/dev/ttyUSB1	AT 命令
		4	/dev/ttyUSB2	Modem
RG500U 系列/ RG200U 系列/ RM500U 系列： VID：0x2c7c PID：0x0900	ECM/ RNDIS/ NCM	0	usb0	通过 **AT+QCFG="usbnet",1** 配置网卡接口类型为 ECM
		1		通过 **AT+QCFG="usbnet",3** 配置网卡接口类型为 RNDIS 通过 **AT+QCFG="usbnet",5** 配置网卡接口类型为 NCM
	USB 转串口 option	2	/dev/ttyUSB0	DIAG
		3	/dev/ttyUSB1	Log
		4	/dev/ttyUSB2	AT 命令

		5	/dev/ttyUSB3	Modem
		6	/dev/ttyUSB4	GNSS
EG065K 系列/ **EG060K 系列/** **EG120K 系列/** **EM120K-GL/** **EM060K 系列：** VID：0x2c7c PID：0x030b	USB 转串口 option	0	/dev/ttyUSB0	DIAG
		1	/dev/ttyUSB1	GNSS
		2	/dev/ttyUSB2	AT 命令
		3	/dev/ttyUSB3	Modem
	QMI_WWAN	4	wwan0 /dev/cdc-wdm0	通过 **AT+QCFG="usbnet",0** 配置网卡接口类型为 RmNet
RG520F 系列/ **RG525F-NA/** **RG520N 系列/** **RM520N 系列/** **RM521F-GL/** **RG530F 系列/** **RM530N-GL：** VID：0x2c7c PID：0x0801	MBIM	8 9	wwan0 /dev/cdc-wdm0	通过 **AT+QCFG="usbnet",2** 配置网卡接口类型为 MBIM
RM500K-CN： VID：0x2c7c PID：0x7001	RNDIS	0 1	usb0	USB 网络适配器 RNDIS
	USB 转串口 option	2	/dev/ttyUSB0	AP log
		3	/dev/ttyUSB1	AP GNSS
		4	/dev/ttyUSB2	AP Meta
	USBFS	5	-	ADB
	USB 转串口 option	6	/dev/ttyUSB3	Modem AT 命令
		7	/dev/ttyUSB4	Modem Meta
		8	/dev/ttyUSB5	Modem MIPC
		9	/dev/ttyUSB6	Modem ELT
RG500L 系列： VID：0x2c7c PID：0x7003	RNDIS	0 1	usb0	RNDIS
RG620T 系列：	USBFS	2	-	ADB

VID：0x2c7c PID：0x7006	ACM	3	/dev/ttyACM0	Modem AT 命令
		4		
		5	/dev/ttyACM1	Modem ELT
		6		
AG520R 系列： VID：0x2c7c PID：0x0452 **AG55xQ 系列：** VID：0x2c7c PID：0x0455	USB 转串口 option	0	/dev/ttyUSB0	DIAG
	QMI_WWAN	2	wwan0 /dev/cdc-wdm0	通过 **AT+QCFG="usbnet",0** 配置网卡接口类型为 RmNet
	GobiNet	2	usb0 /dev/qcqmi0	
	ECM	4	usb0	通过 **AT+QCFG="usbnet",1** 配置网卡接口类型为 ECM
		5		
	USB 转串口 option	6	/dev/ttyUSB1	GNSS
		7	/dev/ttyUSB2	AT 命令
		8	/dev/ttyUSB3	Modem
	RNDIS	9	usb0	通过 **AT+QCFG="usbnet",3** 配置网卡接口类型 RNDIS
		10		
	MBIM	11	wwan0 /dev/cdc-wdm0	通过 **AT+QCFG="usbnet",2** 配置网卡接口类型 MBIM
		12		
RG255C 系列/ **RM255C-GL：** VID：0x2c7c PID：0x0316	USB 转串口 option	0	/dev/ttyUSB0	DIAG
		1	/dev/ttyUSB1	GNSS
		2	/dev/ttyUSB2	AT 命令
	QMI_WWAN	3	wwan0 /dev/cdc-wdm0	通过 **AT+QCFG="usbnet",0** 配置网卡接口类型为 RmNet
	GobiNet	3	usb0 /dev/qcqmi0	
	USBFS	5	-	ADB
	RNDIS	0	usb0	通过 **AT+QCFG="usbnet",3** 配置网卡接口类型为 RNDIS
		1		

	MBIM	6 7	wwan0 /dev/cdc-wdm0	通过 **AT+QCFG="usbnet",2** 配置网卡接口类型为 MBIM
	ECM	8 9	usb0	通过 **AT+QCFG="usbnet",1** 配置网卡接口类型为 ECM
AG18： VID：0x2c7c PID：0x0452	USB 转串口 option	0	/dev/ttyUSB0	DIAG
		1	/dev/ttyUSB1	IPC
		2	/dev/ttyUSB2	V2X
		7	/dev/ttyUSB3	AT 命令
	RNDIS	9 10	usb0	默认网卡接口类型为 RNDIS
AG215S 系列： VID：0x2c7c PID：0x0456	USB 转串口 option	0	/dev/ttyUSB0	DIAG
	USBFS	15	-	ADB
RG650E 系列/ RG650V 系列： VID：0x2c7c PID：0x0122	USB 转串口 option	0	/dev/ttyUSB0	DIAG
		1	/dev/ttyUSB1	GNSS
		2	/dev/ttyUSB2	AT 命令
		3	/dev/ttyUSB3	Modem
	QMI_WWAN	4	wwan0 /dev/cdc-wdm0	通过 **AT+QCFG="usbnet",0** 配置网卡接口类型为 RmNet
	GobiNet	4	usb0 /dev/qcqmi0	通过 **AT+QCFG="usbnet",0** 配置网卡接口类型为 RmNet
	USBFS	5	-	ADB
	RNDIS	0 1	usb0	通过 **AT+QCFG="usbnet",3** 配置网卡接口类型为 RNDIS
	MBIM	8 9	wwan0 /dev/cdc-wdm0	通过 **AT+QCFG="usbnet",2** 配置网卡接口类型为 MBIM
	ECM	10 11	usb0	通过 **AT+QCFG="usbnet",1** 配置网卡接口类型为 ECM

	USBFS	12	-	QXDM QDSS log
		13	-	QXDM DPL log
EG800Q 系列/ EG915Q 系列/ EG916Q-GL： VID：0x2c7c PID：0x6007	ECM	4 5	usb0	通过 **AT+QCFG="USBNET",1** 配置卡接口类型为 ECM
	RNDIS	0 1	usb0	通过 **AT+QCFG="USBNET",3** 配置网卡接口类型为 RNDIS

备注

1. 可同时移植 GobiNet 和 QMI_WWAN 至 Linux 操作系统，但一次只有其中一个可以工作，即如果已加载 GobiNet，则无法加载 QMI_WWAN，反之亦然。
2. 模块的设备名称不固定。如果用户系统没有连接其他 USB 串口设备，则模块的设备名称从 /dev/ttyUSB0 开始，如上图所示；如果用户系统连接了其他 USB 串口设备，则模块的设备名称由 USB 串口设备生成的设备节点数决定。例如，如果一个 USB 串口设备连接到用户系统并生成一个设备节点，/dev/ttyUSB0 被 USB 串口设备占用，那么模块的设备名称从/dev/ttyUSB1 开始。
3. ADB 工具和 QLog 工具运行时，会动态加载 USBFS 驱动。USBFS 是内核自带驱动，用户无需移植。
4. **AT+QCFG** 的详细信息，请参考***文档 [2]***。
5. **AT+QCFGEXT** 的详细信息，请参考***文档 [3]***。

3 系统设置

本章介绍 Linux 中 USB 协议栈的一般架构以及如何使用、编译和加载 USB 驱动程序。

3.1. Linux USB 驱动架构

USB 是一种分层总线结构。USB 设备与主机之间的数据传输由 USB 控制器控制。Linux USB 驱动程序架构如下图所示。Linux USB 主机驱动包括三部分：USB 主机控制器驱动、USB 核心和 USB 设备驱动。

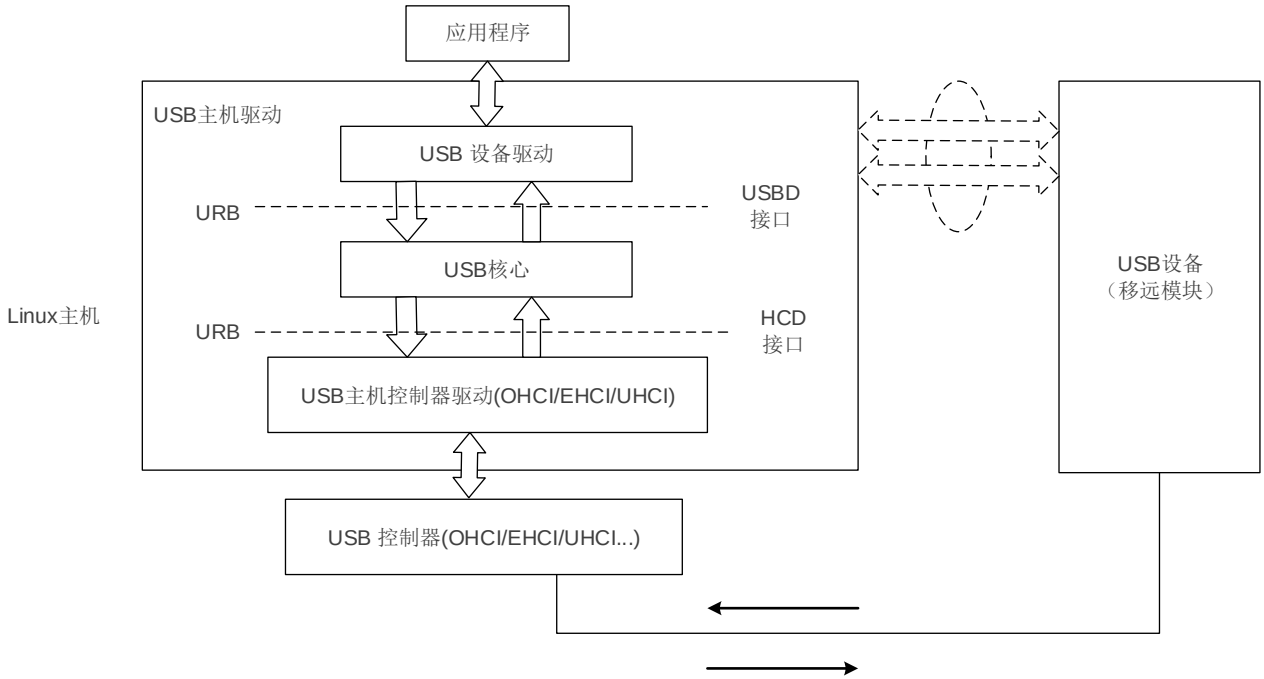

图 1：Linux USB 驱动架构

USB 主机控制器驱动在分层结构的最底层，直接与硬件交互。USB 核心是整个 USB 主机驱动的核心，用于管理 USB 总线、USB 总线设备和 USB 总线带宽；它为 USB 设备驱动程序提供接口，应用程序可以通过这些接口访问 USB 系统文件。

USB 设备驱动程序与应用程序交互并提供用于访问特定 USB 设备的接口。

3.2. USB 转串口驱动

模块加载 USB 转串口 option 驱动程序后，在*/dev* 目录下创建 *ttyUSB0*、*ttyUSB1* 和 *ttyUSB2* 等设备文件。以下章节介绍如何将 USB 转串口 option 驱动程序移植到 Linux 操作系统中。

3.2.1. 添加 VID 和 PID

为了识别模块，需将模块的 VID 和 PID 信息添加到*[KERNEL]/drivers/usb/serial/option.c* 文件中，对应的 VID 和 PID 如***表 1*** 所示。

以 EC25 系列模块为例：

```
static const struct usb_device_id option_ids[] = {
#if 1 //Added by Quectel
    { USB_DEVICE(0x2C7C, 0x0125) },
#endif
```

3.2.2. 使用 USBNet 驱动

第 3.2.1 章中的配置使模块的所有 USB 接口均绑定 USB 转串口 option 驱动程序，导致 USBNet 驱动程序接口无法工作。用户可以添加以下语句以防止 USBNet 驱动程序接口绑定 USB 转串口 option 驱动程序。

- 高于 2.6.30 的 Linux 内核版本，用户可以在*[KERNEL]/drivers/usb/serial/option.c* 文件中添加以下语句：

```
static int option_probe(struct usb_serial *serial, const struct usb_device_id *id) {
    struct usb_wwan_intf_private *data;
    ……
#if 1   //Added by Quectel
        if (serial->dev->descriptor.idVendor == cpu_to_le16(0x2C7C)) {
                __u16 idProduct = le16_to_cpu(serial->dev->descriptor.idProduct);
                struct usb_interface_descriptor *intf = &serial->interface->cur_altsetting->desc;

                if (intf->bInterfaceClass != 0xFF || intf->bInterfaceSubClass == 0x42) {
                        //ECM, RNDIS, NCM, MBIM, ACM, UAC, ADB
                        return -ENODEV;
                }

                if ((idProduct&0xF000) == 0x0000) {
                        //MDM interface 4 is QMI
                        if (intf->bInterfaceNumber == 4 && intf->bNumEndpoints == 3
                                        &&      intf->bInterfaceSubClass      ==      0xFF      &&
```

```
intf->bInterfaceProtocol == 0xFF)
                                        return -ENODEV;
                        }
                }
#endif
        /* Store device id so we can use it during attach. */
        usb_set_serial_data(serial, (void *)id);
        return 0;
}
```

- 低于 2.6.31 的 Linux 内核版本，用户可以在*[KERNEL]/drivers/usb/serial/option.c* 文件中添加以下语句：

```
static int option_startup(struct usb_serial *serial)
{
……
        dbg("%s", __func__);
#if 1   //Added by Quectel
          if (serial->dev->descriptor.idVendor == cpu_to_le16(0x2C7C)) {
                        __u16 idProduct = le16_to_cpu(serial->dev->descriptor.idProduct);
                        struct usb_interface_descriptor *intf = &serial->interface->cur_altsetting->desc;

                        if (intf->bInterfaceClass != 0xFF || intf->bInterfaceSubClass == 0x42) {
                                //ECM, RNDIS, NCM, MBIM, ACM, UAC, ADB
                                return -ENODEV;
                        }

                        if ((idProduct&0xF000) == 0x0000) {
                                //MDM interface 4 is QMI
                                if (intf->bInterfaceNumber == 4 && intf->bNumEndpoints == 3
                                        &&      intf->bInterfaceSubClass      ==      0xFF      &&
intf->bInterfaceProtocol == 0xFF)
                                        return -ENODEV;
                        }
                }
#endif
……
}
```

3.2.3. 修改内核配置

用户需启用以下配置项。内核配置相关信息，请参考***第 3.7 章***。

```
CONFIG_USB_SERIAL
```

```
CONFIG_USB_SERIAL_WWAN
CONFIG_USB_SERIAL_OPTION
```

3.2.4. 添加零包机制

根据 USB 协议的要求，通过添加如下语句在 bulk-out 传输过程中添加处理零包的机制：

- 高于 2.6.34 的 Linux 内核版本，需在*[KERNEL]/drivers/usb/serial/usb_wwan.c* 文件中添加以下语句。

```
static struct urb *usb_wwan_setup_urb(struct usb_serial *serial, int endpoint,
                                      int dir, void *ctx, char *buf, int len,void (*callback) (struct urb *))
{
……
    usb_fill_bulk_urb(urb, serial->dev,
                      usb_sndbulkpipe(serial->dev, endpoint) | dir,
                      buf, len, callback, ctx);
    #if 1     //Added by Quectel for zero packet
    if (dir == USB_DIR_OUT) {
        struct usb_device_descriptor *desc = &serial->dev->descriptor;

        if (desc->idVendor == cpu_to_le16(0x2C7C))
            urb->transfer_flags |= URB_ZERO_PACKET;
    }
#endif
    return urb;
}
```

- 低于 2.6.35 的 Linux 内核版本，需在*[KERNEL]/drivers/usb/serial/option.c* 文件中添加以下语句。

```
/* Helper functions used by option_setup_urbs */
static struct urb *option_setup_urb(struct usb_serial *serial, int endpoint,
        int dir, void *ctx, char *buf, int len,
        void (*callback)(struct urb *))
{
……
    usb_fill_bulk_urb(urb, serial->dev,
                      usb_sndbulkpipe(serial->dev, endpoint) | dir,
                      buf, len, callback, ctx);
    #if 1     //Added by Quectel for zero packet
    if (dir == USB_DIR_OUT) {
        struct usb_device_descriptor *desc = &serial->dev->descriptor;

        if (desc->idVendor == cpu_to_le16(0x2C7C))
```

```
                urb->transfer_flags |= URB_ZERO_PACKET;
#endif
        return urb;
}
```

3.2.5. 添加 Reset-resume 机制

部分 USB 主机控制器或 USB hub 在 MCU 进入 Suspend/Sleep（挂起/睡眠）模式时会发生掉电或复位，并且在 MCU 退出 Suspend/Sleep 模式后不能使模块恢复。需要通过添加以下语句来启用 reset-resume 机制。

- 高于 3.4 的 Linux 内核版本，需在*[KERNEL]/drivers/usb/serial/option.c* 文件中添加以下语句。

```
static struct usb_serial_driver option_1port_device = {
……
#ifdef CONFIG_PM
        .suspend                = usb_wwan_suspend,
        .resume                 = usb_wwan_resume,
#if 1    //Added by Quectel
        .reset_resume      = usb_wwan_resume,
#endif
#endif
};
```

- 低于 3.5 的 Linux 内核版本，需在*[KERNEL]/drivers/usb/serial/usb-serial.c* 文件中添加以下语句。

```
/* Driver structure we register with the USB core */
static struct usb_driver usb_serial_driver = {
                .name =              "usbserial",
                .probe =             usb_serial_probe,
                .disconnect =      usb_serial_disconnect,
                .suspend =           usb_serial_suspend,
                .resume =            usb_serial_resume,
#if 1 //Added by Quectel
                .reset_resume = usb_serial_resume,
#endif
                .no_dynamic_id =             1,
                .supports_autosuspend = 1,
};
```

3.2.6. 增加批量输出 URB 的数量和容量

低于 2.6.29 的 Linux 内核版本，需在*[KERNEL]/drivers/usb/serial/option.c* 文件中添加如下语句以增加 Bulk Out URB 的数量和容量，从而获得更快的上行速度。

```
#define N_IN_URB 4
#define N_OUT_URB 4             //Increase the quantity of the bulk out URBs to 4.
#define IN_BUFLEN 4096
#define OUT_BUFLEN 4096     //Increase the capacity of the bulk out URBs to 4096.
```

3.3. GobiNet 驱动

当模块成功加载 GobiNet 驱动程序后，将创建一个网络设备和一个 QMI 设备节点。网络设备名称为“usbX”，QMI 设备节点名称为“/dev/qcqmiX”。网络设备用于数据传输，QMI 设备节点用于 QMI 消息交互。

该驱动仅适用于支持 RmNet 接口的模块。使用 GobiNet 驱动前，请通过 **AT+QCFG="usbnet",0** 配置网卡接口类型为 RmNet。

以下章节介绍如何将 GobiNet 驱动程序移植到 Linux 操作系统中。

3.3.1. 修改驱动源码

GobiNet 驱动程序由移远通信以源文件的形式提供。源文件应复制到*[KERNEL]/drivers/net/usb/*（若内核版本低于 2.6.22，则应复制到*[KERNEL]/drivers/usb/net/*）。

3.3.2. 修改内核配置

用户需先启用以下配置项。内核配置相关信息，请参考***第 3.7 章***。

```
CONFIG_USB_NET_DRIVERS
CONFIG_USB_USBNET
```

然后添加以下语句至*[KERNEL]/drivers/net/usb/Makefile*（若内核版本低于 2.6.22，则添加至*[KERNEL]/drivers/usb/net/Makefile*）。

```
obj-y += GobiNet.o
GobiNet-objs := GobiUSBNet.o QMIDevice.o QMI.o
```

3.4. QMI_WWAN 驱动

当模块成功加载 QMI_WWA 驱动程序后，会创建一个网络设备和一个 QMI 设备节点。网络设备名称为“wwanX”，QMI 设备节点名称为“/dev/cdc-wdmX”。网络设备用于数据传输，QMI 设备节点用于 QMI 消息交互。

该驱动仅适用于支持 RmNet 接口的模块。使用 QMI_WWAN 驱动前，请通过 **AT+QCFG="usbnet",0** 配置网卡接口类型为 RmNet。

以下章节介绍如何将 QMI_WWAN 驱动程序移植到 Linux 操作系统中。

3.4.1. 修改驱动源码

移远通信模块使用 qmi_wwan 驱动，需修改驱动程序源文件*[KERNEL]/drivers/net/usb/qmi_wwan.c*。为简化移植步骤，移远通信提供 QMI_WWAN 驱动。QMI_WWAN 驱动源文件 *qmi_wwan_q.c* 可与 *qmi_wwan.c* 共存，且仅用于移远通信模块。移植时，将 *qmi_wwan_q.c* 复制到*[KERNEL]/drivers/net/usb/* 目录下。

3.4.2. 修改内核配置

用户需先启用以下配置项。内核配置相关信息，请参考***第 3.7 章***。

```
CONFIG_USB_NET_DRIVERS
CONFIG_USB_USBNET
CONFIG_USB_NET_QMI_WWAN
CONFIG_USB_WDM
```

然后添加如下语句至*[KERNEL]/drivers/net/usb/Makefile*。

```
# must insert qmi_wwan_q.o before qmi_wwan.o
obj-${CONFIG_USB_NET_QMI_WWAN} += qmi_wwan_q.o
obj-${CONFIG_USB_NET_QMI_WWAN} += qmi_wwan.o
```

3.5. ACM/ECM/RNDIS/NCM/MBIM 驱动

ACM、ECM、RNDIS、NCM 和 MBIM 驱动程序是 USB 接口类驱动程序，即 Linux 系统根据接口的类、子类和协议自动加载相应的驱动。若用户使用 Ubuntu 和 Fedora 等 Linux 发行版，这些驱动程序在上游 Linux 版本中可用。当模块通过 USB 接口连接到 Linux PC 时，会自动加载驱动程序。如果使用嵌入式系统，只需开启相应的配置项即可。

每个驱动需启用的配置项如下：

表 2：USB 类驱动的配置项

USB 驱动	配置项	源文件
ACM	CONFIG_USB_ACM	*[KERNEL]/drivers/net/usb/cdc-acm.c*
ECM	CONFIG_USB_NET_DRIVERS CONFIG_USB_USBNET CONFIG_USB_NET_CDCETHER	*[KERNEL]/drivers/net/usb/cdc_ether.c*
RNDIS	CONFIG_USB_NET_DRIVERS CONFIG_USB_USBNET CONFIG_USB_NET_RNDIS_HOST	*[KERNEL]/drivers/net/usb/rndis_host.c*
NCM	CONFIG_USB_NET_DRIVERS CONFIG_USB_USBNET CONFIG_USB_NET_CDC_NCM	*[KERNEL]/drivers/net/usb/cdc_ncm.c*
MBIM	CONFIG_USB_NET_DRIVERS CONFIG_USB_USBNET CONFIG_USB_NET_CDC_MBIM	*[KERNEL]/drivers/net/usb/cdc_mbim.c*

3.6. 支持 PPP

若使用 PPP 功能，用户需先启用以下内核配置项来支持 PPP。内核配置相关信息，请参考***第 3.7 章***。

```
CONFIG_PPP
CONFIG_PPP_ASYNC
CONFIG_PPP_SYNC_TTY
CONFIG_PPP_DEFLATE
```

3.7. 配置内核

请按照以下步骤和相应命令配置内核。

步骤一：执行以下命令切换到内核目录：

```
cd <用户内核目录>
```

步骤二：执行以下命令设置环境变量并导入开发板的“defconfig”文件（以树莓派板子为例）。

```
export ARCH=arm
export CROSS_COMPILE=arm-none-linux-gnueabi-
make bcmrpi_defconfig
```

步骤三：执行以下命令编译内核。

```
make menuconfig
```

步骤四：启用配置项。

选择<*>表示将驱动程序编译到内核映像。
选择<M>表示将驱动程序编译成模块。

以 USB 转串口 option 驱动为例，用户可以通过以下选项启用 CONFIG_USB_SERIAL_OPTION，将 USB 转串口 option 驱动编译到内核镜像。

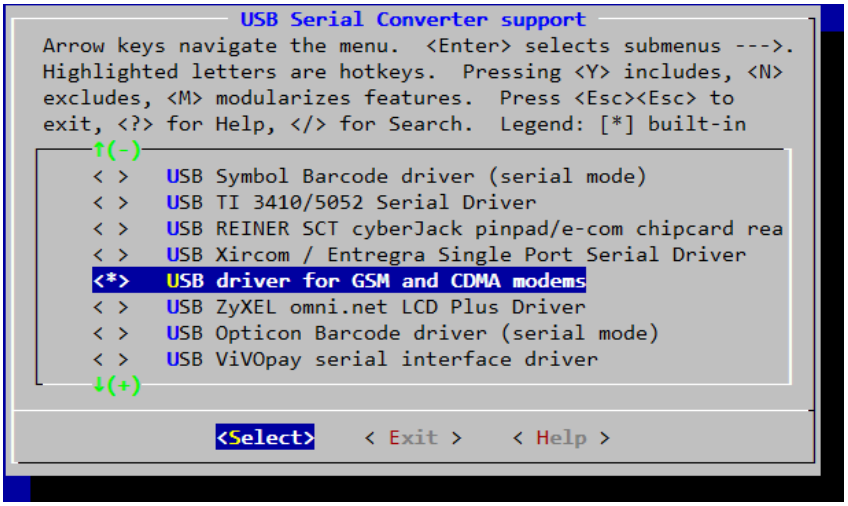

图 2：在内核中配置 USB 串口

3.8. 以内核模块的形式在 Linux PC 上安装和加载驱动

对于需要在 Ubuntu 等 Linux PC 上测试移远通信模块的开发人员，移远通信可以提供 USB 转串口 option、GobiNet 和 QMI_WWAN 驱动程序的源文件。开发人员可以使用以下命令安装驱动，驱动安装成功后重启 PC 就可以使用驱动。

- 安装 QMI_WWAN 驱动。

```
q@q-OptiPlex-7050:~/quectel/qmi_wwan$ sudo make install
```

- 安装 GobiNet 驱动。

```
q@q-OptiPlex-7050:~/quectel/ GobiNet$ sudo make install
```

- 安装 USB 转串口 option 驱动。

```
# First use command `uanme –r` to query the current using kernel version
q@q-OptiPlex-7050:~/quectel/usb-serial-option$ uname    -r
4.4.0-31-generic
# Switch to the correspond kernel source directory
q@q-OptiPlex-7050:~/quectel/usb-serial-option$ cd 4.4.0/
q@q-OptiPlex-7050:~/quectel/usb-serial-option/4.4.0$ cp ../Makefile    ./
q@q-OptiPlex-7050:~/quectel/usb-serial-option/4.4.0$ sudo make install
```

4 测试模块

模块支持 AT 和 PPP 功能。如果已经安装了 USBNet 驱动程序，也可以使用模块的 USB 网络适配器功能。以下章节介绍如何测试这些功能。

4.1. 测试 AT 功能

模块连接主机并成功加载 USB 驱动后，会在*/dev* 目录下创建多个设备文件。

通常 AT 端口是由 USB 转串口 option 驱动程序创建的 ttyUSB 端口。可在***表 1*** 中查看 AT 或 Modem 功能对应的设备名称。

然后可使用下图所示的“minicom”或“busybox microcom”等 UART 工具来测试 AT 功能。

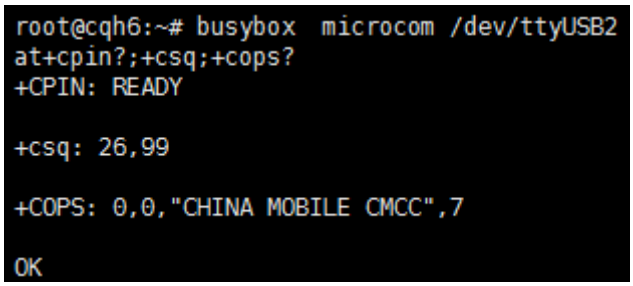

图 3：AT 命令测试结果

4.2. 测试 PPP 功能

若模块支持 USBNet 驱动，推荐使用 USBNet 驱动。

PPP 拨号比网卡拨号复杂，对 CPU 的电流消耗较大，因此不推荐使用 PPP 拨号。

PPP 拨号需要以下文件。用户需检查产品中是否存在以下文件：

1. PPPD 和 chat 程序。若这两个程序不存在，用户可从 https://ppp.samba.org/download.html 下载这两个程序的源代码并将其移植至产品中。

2. PPP 脚本文件*/etc/ppp/ip-up*，用于设置 DNS。若无该文件，请使用移远通信提供的 *linux-ppp-scripts\ip-up*。

3. *quectel-ppp*、*quectel-chat-connect* 和 *quectel-chat-disconnect* 脚本。均由移远通信提供，位于 *linux-ppp-scripts* 目录下。用户可能需要根据不同的模块进行相应修改。更多信息，请参考 *linux-ppp-scripts\readme*。

将 *quectel-ppp*、*quectel-chat-connect* 和 *quectel-chat-disconnect* 复制到*/etc/ppp/peers* 目录下，然后通过以下命令进行 PPP 拨号。

```
# pppd call quectel-ppp &
```

PPP 拨号建立过程如下图所示：

```
abort on (BUSY)
abort on (NO CARRIER)
abort on (NO DIALTONE)
abort on (ERROR)
abort on (NO ANSWER)
timeout set to 30 seconds
send (AT^M)
expect (OK)
AT^M^M
OK
 -- got it

send (ATD*99#^M)
expect (CONNECT)
^M
ATD*99#^M^M
CONNECT
 -- got it

Script chat -s -v -f /etc/ppp/peers/quectel-chat-connect finished (pid 2912), status = 0x0
Serial connection established.
using channel 1
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB3
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x588fbf7f> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x0 <asyncmap 0x0> <auth chap MD5> <magic 0xea02c208> <pcomp>
<accomp>]
sent [LCP ConfAck id=0x0 <asyncmap 0x0> <auth chap MD5> <magic 0xea02c208> <pcomp>
<accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x588fbf7f> <pcomp> <accomp>]
```

```
sent [LCP EchoReq id=0x0 magic=0x588fbf7f]
rcvd [LCP DiscReq id=0x1 magic=0xea02c208]
rcvd [CHAP Challenge id=0x1 <86b3d5669380a4bcfa502b8e92a4cc93>, name = "UMTS_CHAP_SRVR"]
sent [CHAP Response id=0x1 <9efc37eaf3dd8d819ac3e452d242e026>, name = "test"]
rcvd [LCP EchoRep id=0x0 magic=0xea02c208 58 8f bf 7f]
rcvd [CHAP Success id=0x1 ""]
CHAP authentication succeeded
CHAP authentication succeeded
sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [IPCP ConfReq id=0x0]
sent [IPCP ConfNak id=0x0 <addr 0.0.0.0>]
rcvd [IPCP ConfNak id=0x1 <addr 10.187.151.143> <ms-dns1 202.102.213.68> <ms-dns2 61.132.163.68>]
sent [IPCP ConfReq id=0x2 <addr 10.187.151.143> <ms-dns1 202.102.213.68> <ms-dns2 61.132.163.68>]
rcvd [IPCP ConfReq id=0x1]
sent [IPCP ConfAck id=0x1]
rcvd [IPCP ConfAck id=0x2 <addr 10.187.151.143> <ms-dns1 202.102.213.68> <ms-dns2 61.132.163.68>]
Could not determine remote IP address: defaulting to 10.64.64.64
not replacing default route to eth0 [172.18.112.1]
local  IP address 10.187.151.143
remote IP address 10.64.64.64
primary     DNS address 202.102.213.68
secondary DNS address 61.132.163.68
Script /etc/ppp/ip-up started (pid 2924)
Script /etc/ppp/ip-up finished (pid 2924), status = 0x0
```

此时 PPP 拨号成功。用户使用以下命令检查系统中的 IP、DNS 及路由信息是否属于移远通信模块。

```
# ifconfig ppp0
ppp0          Link encap:Point-to-Point Protocol
              inet addr: 10.187.151.143  P-t-P:10.64.64.64  Mask:255.255.255.255
              UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
              RX packets:15 errors:0 dropped:0 overruns:0 frame:0
              TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:3
              RX bytes:1057 (1.0 KiB)  TX bytes:1228 (1.1 KiB)

# cat /etc/resolv.conf
nameserver 61.132.163.68
nameserver 202.102.213.68
```

```
# route    -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.64.64.64     0.0.0.0         255.255.255.255 UH    0      0        0 ppp0
0.0.0.0         0.0.0.0         0.0.0.0         U     0      0        0 ppp0

# ping www.baidu.com
PING www.a.shifen.com (115.239.211.112) 56(84) bytes of data.
64 bytes from 115.239.211.112: icmp_seq=1 ttl=54 time=46.4ms
```

可执行如下命令终止 PPPD 进程以断开 PPP 拨号。

```
# killall pppd
Terminating on signal 15
Connect time 0.4 minutes.
Sent 0 bytes, received 0 bytes.
```

备注

数据速率高于Cat 4的移远通信5G模块系列和LTE模块系列不支持PPP拨号。

4.3. 测试 GobiNet/QMI_WWAN 驱动

请按照以下步骤测试 GobiNet 或 QMI_WWAN 驱动：

步骤一：使用以下命令编译 Connect Manager 程序。移远通信提供 Connect Manager 程序（“quectel-CM”）供用户手动建立数据连接。Connect Manager 在*/quectel-CM/*目录下以源码的形式提供。

- PC Linux：

```
# make
```

- 嵌入式 Linux：

```
# make CROSS-COMPILE=arm-none-linux-gnueabi-
```

用户需用实际使用的交叉编译器替换 *arm-none-linux-gnueabi-*。此步骤将输出名称为“quectel-CM”

的可执行程序。

步骤二：准备 busybox udhcpc 工具。

quectel-CM 调用 busybox udhpc 获取 IP 和 DNS，busybox udhpc 调用脚本文件 */usr/share/udhcpc/default.script* 为 Linux 开发板设置 IP、DNS 和路由表。

用户可以从 https://busybox.net 下载 busybox udhpc 工具的源代码，然后使用以下命令启用 CONFIG_UDHCPC 并将脚本文件*[BUSYBOX]/examples/udhcp/simple.script* 复制到 Linux 开发板（重命名为*/usr/share/udhcpc/default.script*）。

```
busybox menuconfig
```

步骤三：使用 quectel-CM 拨号。

模块连接主机并成功加载 GobiNet 或 QMI_WWAN 驱动后，会创建 USB 网卡和 QMI 设备节点。GobiNet 驱动创建的 USB 网卡名称为“usbX”，QMI 设备节点名称为“/dev/qcqmiX”。QMI_WWAN 驱动创建的 USB 网卡名称为“wwanX”，QMI 设备节点名称为“/dev/cdc-wdmX”。

quectel-CM 通过 QMI 设备节点向模块发送 QMI 消息，建立数据连接。quectel-CM 的用法，请参考***文档 [1]***。

quectel-CM 的工作流程（以 EM12 运行 QMI_WWAN 驱动为例）如下图所示：

```
root@cqh6:~# ./quectel-CM/quectel-CM &
[07-03_06:56:28:172] WCDMA&LTE_QConnectManager_Linux&Android_V1.3.4
[07-03_06:56:28:172] ./quectel-CM/quectel-CM profile[1] = (null)/(null)/(null)/0, pincode = (null)
[07-03_06:56:28:174] Find /sys/bus/usb/devices/2-1.2 idVendor=2c7c idProduct=0512
[07-03_06:56:28:174] Find /sys/bus/usb/devices/2-1.2:1.4/net/wwan0
[07-03_06:56:28:174] Find usbnet_adapter = wwan0
[07-03_06:56:28:175] Find /sys/bus/usb/devices/2-1.2:1.4/usbmisc/cdc-wdm0
[07-03_06:56:28:175] Find qmichannel = /dev/cdc-wdm0
[07-03_06:56:28:197] cdc_wdm_fd = 7
[07-03_06:56:28:381] Get clientWDS = 18
[07-03_06:56:28:445] Get clientDMS = 1
[07-03_06:56:28:509] Get clientNAS = 2
[07-03_06:56:28:573] Get clientUIM = 2
[07-03_06:56:28:637] Get clientWDA = 1
[07-03_06:56:28:701] requestBaseBandVersion EM12GPAR01A06M4G
[07-03_06:56:28:957] requestGetSIMStatus SIMStatus: SIM_READY
[07-03_06:56:29:021] requestGetProfile[1] cmnet///0
[07-03_06:56:29:085] requestRegistrationState2 MCC: 460, MNC: 0, PS: Attached, DataCap: LTE
[07-03_06:56:29:149] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[07-03_06:56:29:277] requestRegistrationState2 MCC: 460, MNC: 0, PS: Attached, DataCap: LTE
```

```
[07-03_06:56:29:341] requestSetupDataCall WdsConnectionIPv4Handle: 0x127b42c0
[07-03_06:56:29:469] requestQueryDataCall IPv4ConnectionStatus: CONNECTED
[07-03_06:56:29:533] ifconfig wwan0 up
[07-03_06:56:29:543] busybox udhcpc -f -n -q -t 5 -i wwan0
udhcpc: started, v1.27.2
udhcpc: sending discover
udhcpc: sending select for 10.170.235.201
udhcpc: lease of 10.170.235.201 obtained, lease time 7200
[07-03_06:56:29:924] /etc/udhcpc/default.script: Resetting default routes
[07-03_06:56:29:936] /etc/udhcpc/default.script: Adding DNS 211.138.180.2
[07-03_06:56:29:936] /etc/udhcpc/default.script: Adding DNS 211.138.180.3
```

步骤四：使用以下命令查看 IP、DNS 和路由信息。

```
root@cqh6:~# ifconfig wwan0
wwan0: flags=4291<UP,BROADCAST,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.170.235.201  netmask 255.255.255.252  broadcast 10.170.235.203

root@cqh6:~# cat /etc/resolv.conf
nameserver 211.138.180.2
nameserver 211.138.180.3

root@cqh6:~# ip route show
default via 10.170.235.202 dev wwan0
10.170.235.200/30 dev wwan0 proto kernel scope link src 10.170.235.201
172.18.112.0/23 dev eth0 proto kernel scope link src 172.18.112.13

# ping www.baidu.com
PING www.a.shifen.com (115.239.211.112) 56(84) bytes of data.
64 bytes from 115.239.211.112: icmp_seq=1 ttl=53 time=24.8 ms
```

步骤五：使用以下命令终止 quectel-CM 进程以断开数据连接。

```
root@cqh6:~# killall quectel-CM
[07-03_07:00:10:145] requestDeactivateDefaultPDP err = 0
[07-03_07:00:10:145] ifconfig wwan0 down
[07-03_07:00:10:152] ifconfig wwan0 0.0.0.0
[07-03_07:00:10:553] QmiWwanThread exit
[07-03_07:00:10:554] main exit
```

4.4. 在 GobiNet/ QMI_WWAN 驱动上测试 AT$QCRMCALL

本章主要介绍如何使用 **AT$QCRMCALL** 拨号。

虽然推荐使用 quectel-CM、libqmi 和 uqmi 等 QMI 工具拨号，若用户 MCU 的 USB 主机控制器不完全支持 USB 中断类型端点，用户需使用 **AT$QCRMCALL** 而不是 QMI 工具进行拨号。

对于 GobiNet 驱动，为使用 **AT$QCRMCALL**，需将 *GobiUSBNet.c* 中的 qcrmcall_mode 变量修改为 1；而对于 QMI_WWAN 驱动程序，无需额外修改。

如下日志显示如何使用 **AT$QCRMCALL** 进行拨号。详情请联系移远通信技术支持。

```
root@imx6qdlsabresd:~# busybox microcom /dev/ttyUSB2
at+csq;+cgreg?;+cops?
+CSQ: 27,99
+CGREG: 0,1
+COPS: 0,0,"CHINA MOBILE",7
OK

AT$QCRMCALL=1,1
$QCRMCALL: 1,V4
OK

AT+QNETDEVSTATUS?
+QNETDEVSTATUS: 0,1,4,1
OK

root@imx6qdlsabresd:~# busybox udhcpc -fnq -i wwan0
udhcpc (v1.24.1) started
Sending discover...
Sending select for 10.166.47.120...
Lease of 10.166.47.120 obtained, lease time 7200
/etc/udhcpc.d/50default: Adding DNS 211.138.180.2
/etc/udhcpc.d/50default: Adding DNS 211.138.180.3
root@imx6qdlsabresd:~#
```

备注

仅 EC25 系列、EG25-G、EC21 系列、EG91 系列、EG21-G、EC20-CE、EG95 系列和 EM05 系列模块支持 **AT$QCRMCALL**。

4.5. 在 GobiNet/QMI_WWAN 驱动上测试 QMAP

本章介绍如何在 GobiNet 或 QMI_WWAN 驱动上测试 QMAP（Qualcomm Multiplexing and Aggregation Protocol 高通复用和聚合协议），特别适合使用 GobiNet 或 QMI_WWAN 驱动且需要 QMAP 的开发者。

使用 GobiNet 或 QMI_WWAN 驱动时，默认只能创建一个物理网卡，因此只能进行一个 PDN 拨号。但是通过复用协议，可以在一张物理网卡上创建多个虚拟网卡，从而可以进行多个 PDN 拨号。

使用 GobiNet 或 QMI_WWAN 驱动时，一个 URB 中只能传输一个 IP 数据包，所以如果出现高吞吐量和频繁 URB 中断的情况，主机 CPU 就会过载。但是聚合协议可在一个 URB 中传输多个 IP 数据包，通过减少 URB 中断的数量来提高吞吐量。

如需使用多路复用或聚合协议，请参考***文档 [1]***。

4.6. 测试 ECM/RNDIS/NCM/MBIM 驱动

部分使用 ECM、RNDIS 和 NCM 驱动的模块，也可通过移远通信提供的 quectel-CM 执行 **AT+QNETDEVCTL** 进行拨号，详见***文档 [1]***。详情请联系移远通信技术支持。

对于 MBIM 模式，可以使用 mbimcli 和 umbim 等 MBIM 工具进行拨号，也可使用移远通信提供的 quectel-CM 进行拨号，详见***文档 [1]***。

5 电源管理

Linux 中的 USB 系统提供了两个高级电源管理功能：USB 自动挂起和 USB 远程唤醒。本章针对有需要的开发人员介绍如何启用该功能。

当 USB 主机和 USB 设备之间的 USB 通信空闲一段时间（例如 3 秒）时，USB 主机可以使 USB 设备自动进入挂起模式。此功能称为 USB 自动挂起。

USB 远程唤醒允许挂起的 USB 设备通过 USB 远程唤醒 USB 主机，该 USB 也可能被挂起（例如深度睡眠模式）。具有远程唤醒能力的 USB 设备执行唤醒 USB 主机的活动，然后 USB 主机被远程活动唤醒。

除 USB 转串口 option 驱动程序外，本文档描述的驱动程序的 USB 自动挂起和 USB 远程唤醒功能均默认开启。

5.1. 启用 USB 自动挂起

对于 USB 转串口 option 驱动程序，请在*[KERNEL]/drivers/usb/serial/option.c* 文件中的 *option_probe()* 中添加以下语句以启用 USB 自动挂起功能。

```
static int option_probe(struct usb_serial *serial, const struct usb_device_id *id) {
    struct usb_wwan_intf_private *data;
    ……
#if 1 //Added by Quectel
    //For USB Auto Suspend
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x2C7C)) {
        pm_runtime_set_autosuspend_delay(&serial->dev->dev, 3000);
        usb_enable_autosuspend(serial->dev);
    }
#endif
    /* Store device id so we can use it during attach. */
    usb_set_serial_data(serial, (void *)id);
    return 0;
}
```

5.2. 启用 USB 远程唤醒

对于 USB 转串口 option 驱动程序，请在*[KERNEL]/drivers/usb/serial/option.c* 文件中的 *option_probe()* 中添加以下语句以启用 USB 远程唤醒功能。

```
static int option_probe(struct usb_serial *serial, const struct usb_device_id *id) {
    struct usb_wwan_intf_private *data;
    ……
#if 1 //Added by Quectel
    //For USB Remote Wakeup
    if (serial->dev->descriptor.idVendor == cpu_to_le16(0x2C7C)) {
        device_init_wakeup(&serial->dev->dev, 1); //usb remote wakeup
    }
#endif
    /* Store device id so we can use it during attach. */
    usb_set_serial_data(serial, (void *)id);
    return 0;
}
```

6 常见问题和内核 Log

6.1. 如何检查设备中是否存在 USB 驱动

可通过检查*/sys/bus/usb/drivers* 的目录确认是否存在 USB 驱动程序。例如：

```
root@OpenWrt:~# ls /sys/bus/usb/drivers
GobiNet          cdc_wdm          rndis_host       usbfs
cdc_ether        hub              uas              usbserial
cdc_mbim         option           usb              usbserial_generic
cdc_ncm          qmi_wwan_q       usb-storage
```

若需要 USB 转串口 option 驱动程序，请确保*/sys/bus/usb/drivers* 目录下存在 *option*。同理，若需要 GobiNet 驱动程序，请确保 *GobiNet* 存在。若需要 QMI_WWAN 驱动程序，请确保 *qmi_wwan_q* 存在，依此类推。

6.2. 如何检查模块与对应的 USB 驱动程序是否正常工作

本章显示的内核 log 是由连接了模块并安装相应 USB 驱动程序的 Linux 系统打印。如果模块不能正常工作，用户可将自己的内核 log 与本章中的内核 log 进行比较，以帮助排除故障。

- 对于 USB 转串口 option 和 GobiNet 驱动：不同模块的内核 log 除了 VID&PID 信息（下图红框中所示信息）外几乎相同。RG502Q 系列模块的 USB 转串口 option 和 GobiNet log 示例如下：

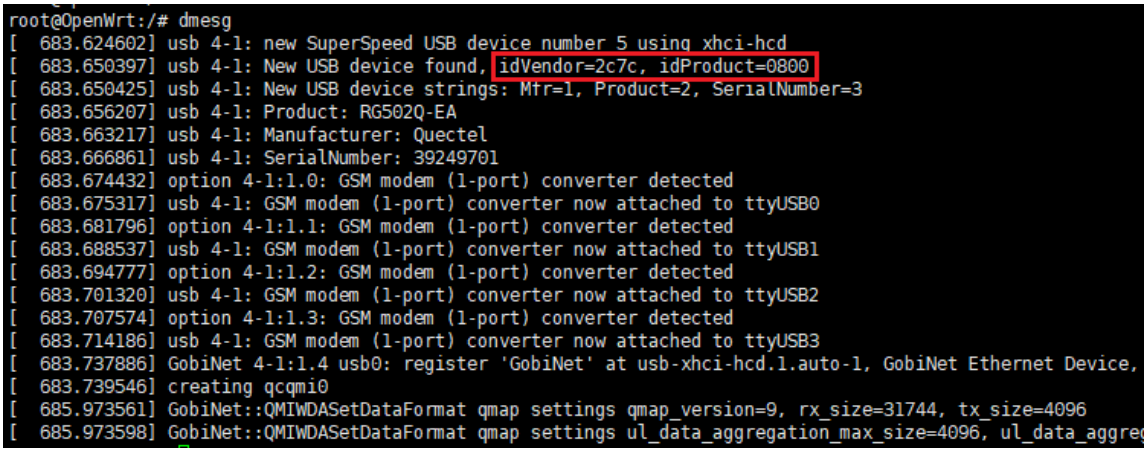

图 4：RG502Q 系列模块 USB 转串口 Option 和 GobiNet Log

- 对于 USB 转串口 option 和 QMI_WWAN 驱动：不同模块的内核 log 除了 VID&PID 信息（下图红框中所示信息）外几乎相同。RG502Q 系列模块的 USB 转串口 option 和 qmi_wwan_q log 示例如下：

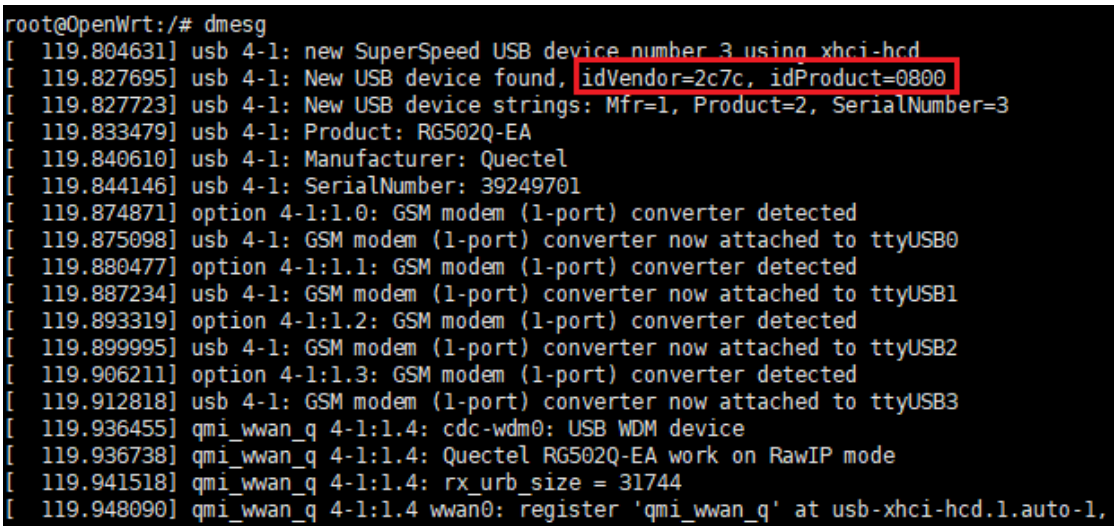

图 5：RG502Q 系列模块 USB 转串口 Option 和 qmi_wwan_q Log

6.3. 如何检查已安装的 USB 驱动程序

本章介绍如何查询移远通信模块的 USB 接口连接至哪个 USB 驱动程序。USB 驱动程序的名称由关键字“Driver=”标识。若显示“Driver=none”，可能是内核配置中没有启用相应的配置项，或者移远通信模块的 VID 和 PID 没有添加到相应的 USB 驱动源文件中。这种情况下，请按照*第 2 章*中的步骤再次检查。

RG502Q 系列模块 USB 接口及驱动示例如下：

```
root@OpenWrt:/# mount -t debugfs none /sys/kernel/debug/
root@OpenWrt:/# cat /sys/kernel/debug/usb/devices
T:  Bus=04 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#=  3 Spd=5000 MxCh= 0
D:  Ver= 3.20 Cls=00(>ifc ) Sub=00 Prot=00 MxPS= 9 #Cfgs=  1
P:  Vendor=2c7c ProdID=0800 Rev= 4.14
S:  Manufacturer=Quectel
S:  Product=RG502Q-EA
S:  SerialNumber=39249701
C:* #Ifs= 5 Cfg#= 1 Atr=a0 MxPwr=896mA
I:* If#= 0 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=30 Driver=option
E:  Ad=81(I) Atr=02(Bulk) MxPS=1024 Ivl=0ms
E:  Ad=01(O) Atr=02(Bulk) MxPS=1024 Ivl=0ms
I:* If#= 1 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=option
E:  Ad=83(I) Atr=03(Int.) MxPS=  10 Ivl=32ms
E:  Ad=82(I) Atr=02(Bulk) MxPS=1024 Ivl=0ms
E:  Ad=02(O) Atr=02(Bulk) MxPS=1024 Ivl=0ms
I:* If#= 2 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=option
E:  Ad=85(I) Atr=03(Int.) MxPS=  10 Ivl=32ms
E:  Ad=84(I) Atr=02(Bulk) MxPS=1024 Ivl=0ms
E:  Ad=03(O) Atr=02(Bulk) MxPS=1024 Ivl=0ms
I:* If#= 3 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=option
E:  Ad=87(I) Atr=03(Int.) MxPS=  10 Ivl=32ms
E:  Ad=86(I) Atr=02(Bulk) MxPS=1024 Ivl=0ms
E:  Ad=04(O) Atr=02(Bulk) MxPS=1024 Ivl=0ms
I:* If#= 4 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=ff Prot=ff Driver=qmi_wwan_q
E:  Ad=88(I) Atr=03(Int.) MxPS=   8 Ivl=32ms
E:  Ad=8e(I) Atr=02(Bulk) MxPS=1024 Ivl=0ms
E:  Ad=0f(O) Atr=02(Bulk) MxPS=1024 Ivl=0ms
```

图 6：RG502Q 系列模块 USB 接口和驱动

7 附录 参考文档及术语缩写

表 3：参考文档

文档名称
[1] Quectel_QConnectManager_Linux_用户指导
[2] Quectel_EC2x&EG2x&EG9x&EM05_Series_QCFG_AT_Commands_Manual
[3] Quectel_BG95&BG77&BG600L_Series_QCFGEXT_AT_Commands_Manual

表 4：术语缩写

缩写	英文全称	中文描述
ACM	Abstract Control Model	抽象控制模型
AP	Application Processor	应用处理器
APN	Access Point Name	接入点名称
CP	Coprocessor	协处理器
CPU	Central Processing Unit	中央处理器
DNS	Domain Name System	域名系统
ECM	Ethernet Control Model	以太网控制模型
EHCI	Enhanced Host Controller Interface	增强型主机控制器接口
GNSS	Global Navigation Satellite System	全球导航卫星系统
GPS	Global Positioning System	全球定位系统
HCD	Host Controller Driver	主机控制器驱动
IP	Internet Protocol	互联网协议
MBIM	Mobile Interface Broadband Model	移动接口宽带模型

MCU	Microcontroller Unit	微控制器单元
NCM	Network Control Model	网络控制模型
NDIS	Network Driver Interface Specification	网络驱动接口规范
NMEA	National Marine Electronics Association	国家海洋电子协会
OHCI	Open Host Controller Interface	开放式主机控制接口
OS	Operating System	操作系统
PC	Personal Computer	个人电脑
PDN	Packet Data Network	分组数据网络
PID	Product ID	产品编号
PPP	Point to Point Protocol	点对点协议
QMAP	Qualcomm Multiplexing and Aggregation Protocol	高通复用和聚合协议
QMI	Qualcomm Messaging Interface	高通消息接口
UHCI	Universal Host Controller Interface	通用主机控制器接口
URB	USB Request Block	USB 请求块
USB	Universal Serial Bus	通用串行总线
VID	Vendor ID	厂商编号