UMTS LTE 5G Linux
USB Driver User Guide

UMTS/HSPA+/LTE/5G Module Series

Version: 3.1
Date: 2023-02-01
Status: Released
At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local offices. For more information, please visit:

For technical support, or to report documentation errors, please visit:
Or email us at: support@quectel.com.

Legal Notices
We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions
License Agreements
Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright
Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.
Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel’s or third-party’s servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.

b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.

c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.

d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

Copyright © Quectel Wireless Solutions Co., Ltd. 2023. All rights reserved.
# About the Document

## Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2015-02-27</td>
<td>Joe WANG</td>
<td>Initial</td>
</tr>
<tr>
<td>1.1</td>
<td>2015-03-25</td>
<td>Carl YIN</td>
<td>Updated supported products</td>
</tr>
<tr>
<td>1.2</td>
<td>2015-03-30</td>
<td>Kent XU</td>
<td>Added Zero Packet feature in Section 3.2.2 and 3.3.2</td>
</tr>
<tr>
<td>1.3</td>
<td>2015-06-24</td>
<td>Carl YIN</td>
<td>1. Added GobiNet and QMI WWAN description in Section 3.4 and 3.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Added building drivers as a kernel module in Section 3.2.4/3.3.4/3.4.3/3.5.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Added power management in Chapter 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4. Added FAQ and kernel log in Chapter 6</td>
</tr>
<tr>
<td>1.4</td>
<td>2015-12-16</td>
<td></td>
<td>1. Deleted Auto-Connect of GobiNet and QMI WWAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Updated the usage of quectel-CM</td>
</tr>
<tr>
<td>1.5</td>
<td>2016-05-13</td>
<td>Carl YIN/Neo HOU</td>
<td>Updated supported modules</td>
</tr>
<tr>
<td>1.6</td>
<td>2016-08-23</td>
<td>Kent XU</td>
<td>Added EC20 R2.0 in supported modules</td>
</tr>
<tr>
<td>1.7</td>
<td>2017-05-24</td>
<td>Kent XU</td>
<td>Added EG91/EG95/EG06/EP06/EM06/BG96 in supported modules</td>
</tr>
<tr>
<td>1.8</td>
<td>2017-09-01</td>
<td>Kent XU</td>
<td>Updated description of supported modules and added AG35 in supported modules</td>
</tr>
<tr>
<td>2.0</td>
<td>2019-12-11</td>
<td>Carl YIN</td>
<td>1. Added applicable modules, which can be referred in Table 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Updated USB driver interface description in Table 2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Updated description of USB serial option, GobiNet and QMI_WWAN drivers in Chapter 3.2, 3.3 and 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4. Added related content of testing command “AT$QCRMSCALL” and protocol QMAP on GobiNet or QMI_WWAN driver in Chapter 5.4 and 5.6 as well as testing MBIM driver in Chapter 5.5.</td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Author</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>2022-03-18</td>
<td>Carl YIN</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>2023-02-01</td>
<td>Jerry MENG</td>
<td></td>
</tr>
</tbody>
</table>

5. Added FAQs in Chapter 6.

1. Added applicable modules.
2. Updated the overview of Linux USB driver (Chapter 2).
3. Updated the steps of integrating USB serial option driver into the Linux operating system (Chapter 3.2).
4. Updated the kernel configuration modification of GobiNet driver (Chapter 3.3.2).
5. Updated the kernel configuration modification of QMI_WWWAN_Q (Chapter 3.4.2).
6. Added the description of ACM/ECM/RNDIS/NCM/MBIM drivers (Chapter 3.5).
7. Updated the information about how to configure the kernel to support PPP (Chapter 3.6).
8. Added the steps of how to configure the kernel (Chapter 3.7).
9. Updated AT command testing (Chapter 4.1).
10. Removed the description of quectel-CM (Chapter 4.3).
11. Added the information about how to test ECM/RNDIS/NCM/MBIM drivers (Chapter 4.6).
12. Updated the example (Chapter 6.1).

1. Updated applicable modules:
   - Updated EG915N-EU to EG915N series and updated AG550Q to AG55xQ series;

2. Updated the module functionalities implemented by Linux system (Chapter 2).
3. Updated the name of network device to usbX (Chapter 3.3).
4. Updated the commands used to install and use USB drivers (Chapter 3.8).
5. Updated data call methods for modules with ECM, RNDIS and NCM drivers (Chapter 4.6).
Contents

About the Document .................................................................................................................. 3
Contents .................................................................................................................................. 5
Table Index ............................................................................................................................... 7
Figure Index ............................................................................................................................. 8

1 Introduction .......................................................................................................................... 9

2 Overview of Linux USB Drivers ......................................................................................... 10

3 System Setup ....................................................................................................................... 16
   3.1. Linux USB Driver Structure ....................................................................................... 16
   3.2. USB Serial Option Driver ......................................................................................... 17
      3.2.1. Add VID and PID ................................................................................................. 17
      3.2.2. Use USBBNet Driver ......................................................................................... 17
      3.2.3. Modify Kernel Configuration ............................................................................ 19
      3.2.4. Add Zero Packet Mechanism ........................................................................... 19
      3.2.5. Add Reset-Resume Mechanism ....................................................................... 20
      3.2.6. Increase Quantity and Capacity of Bulk Out URBs ........................................ 21
   3.3. GobiNet Driver ........................................................................................................... 21
      3.3.1. Modify Driver Source Codes ............................................................................. 21
      3.3.2. Modify Kernel Configuration ........................................................................... 21
   3.4. QMI_WWAN Driver ................................................................................................... 22
      3.4.1. Modify Source Codes of the Driver .................................................................. 22
      3.4.2. Modify Kernel Configuration ........................................................................... 22
   3.5. ACM/ECM/RNDIS/NCM/MBIM Driver .................................................................... 23
   3.6. How to Support PPP ................................................................................................... 23
   3.7. Configure Kernel ....................................................................................................... 24
   3.8. Install and Load Driver as a Kernel Module for PC in Linux ................................... 25

4 Test Module ......................................................................................................................... 26
   4.1. Test AT Function ....................................................................................................... 26
   4.2. Test PPP Function ...................................................................................................... 26
   4.3. Test GobiNet/QMI_WWAN Driver ......................................................................... 29
   4.4. Test "AT$QCRMSCALL" on GobiNet/QMI_WWAN Driver ...................................... 32
   4.5. Test QMAP on GobiNet/QMI_WWAN Driver ......................................................... 33
   4.6. Test ECM/RNDIS/NCM/MBIM Driver ................................................................. 33

5 Power Management ............................................................................................................ 34
   5.1. Enable USB Auto Suspend ...................................................................................... 34
   5.2. Enable USB Remote Wakeup ................................................................................... 35

6 FAQs and Kernel Log .......................................................................................................... 36
   6.1. How to Check If USB Driver Exists in Module ....................................................... 36
   6.2. How to Check If Module Works Well with Corresponding USB Driver .............. 36
6.3. How to Check Which USB Driver is Installed ................................................................. 38

7 Appendix References ........................................................................................................ 39
Table Index

Table 1: Applicable Modules and USB Interface Information ......................................................... 10
Table 2: Configuration Items for USB Class Drivers ........................................................................ 23
Table 3: Related Documents ........................................................................................................... 39
Table 4: Terms and Abbreviations .................................................................................................. 39
Figure Index

Figure 1: Linux USB Driver Structure .............................................................................................................. 16
Figure 2: Configure USB Serial Option in Kernel .............................................................................................. 24
Figure 3: AT Command Test Result .................................................................................................................. 26
Figure 4: USB Serial Option and GobiNet for RG502Q Series Module .............................................................. 37
Figure 5: USB Serial Option and QMI_WWAN for RG502Q Series Module ......................................................... 37
Figure 6: USB Interface and Driver for RG502Q Series Modules .................................................................... 38
# Introduction

This document outlines how to port the supported USB drivers for Quectel UMTS&LTE&5G modules into the Linux operating system, and how to test the module after the USB driver is integrated successfully. USB drivers include:

- USB serial drivers like option and ACM
- USBNet drivers like GobiNet, QMI_WWAN, MBIM, NCM, RNDIS and ECM
2 Overview of Linux USB Drivers

Quectel UMTS/LTE&5G modules are USB composite devices with multiple USB interfaces. Each USB interface supports different functionalities, which are implemented by loading different USB interface drivers. After a driver is loaded successfully, the corresponding device node is generated, which can be used by the Linux system to implement the module functionalities, such as AT command, GNSS, DIAG, log and USB network adapter.

The following table describes the USB interface information of different modules in the Linux system, including USB driver, interface number, device name and interface function.

You can obtain the corresponding VID, PID and interface information of the relevant model, and then port the USB interface driver listed in the following table.

Table 1: Applicable Modules and USB Interface Information

<table>
<thead>
<tr>
<th>Module VID and PID</th>
<th>USB Drivers</th>
<th>Interface Number</th>
<th>Device Names</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC20-CE/EC25 series/EG25-G/EG25-GL/EM05 series: VID: 0x2c7c PID: 0x0125</td>
<td>USB serial option</td>
<td>0</td>
<td>/dev/ttyUSB0</td>
<td>DIAG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>/dev/ttyUSB1</td>
<td>GNSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>/dev/ttyUSB2</td>
<td>AT command</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>/dev/ttyUSB3</td>
<td>Modem</td>
</tr>
<tr>
<td>EC21 series/EG21-G: VID: 0x2c7c PID: 0x0121</td>
<td>GobiNet</td>
<td>4</td>
<td>usb0 /dev/qcqmi0</td>
<td>USB network adapter</td>
</tr>
<tr>
<td>EG91 series:</td>
<td></td>
<td></td>
<td></td>
<td>Configure the type of USBnet interface as RmNet by AT+QCFG=&quot;usbnet&quot;,0.</td>
</tr>
<tr>
<td>Module Series</td>
<td>VID</td>
<td>PID</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>EG95 series</td>
<td>0x2c7c</td>
<td>0x0195</td>
<td>USB network adapter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x2c7c</td>
<td>0x0191</td>
<td>QMI_W spent on 4 wwan0 /dev/cdc-wdm0 Configure the type of USBnet interface as RmNet by <code>AT+QCFG=&quot;usbnet&quot;,0.</code></td>
<td></td>
</tr>
<tr>
<td>AG35 series</td>
<td>0x2c7c</td>
<td>0x0435</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x2c7c</td>
<td>0x0306</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EG06 series/EP06 series/EM06 series</td>
<td>0x2c7c</td>
<td>0x0512</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EG12 series/EM12-G/EG18 series</td>
<td>0x2c7c</td>
<td>0x0512</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EM121R-GL:</td>
<td>0x2c7c</td>
<td>0x0620</td>
<td>USB network adapter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x2c7c</td>
<td>0x0700</td>
<td>MBIM spent on 5 wwan0 /dev/cdc-wdm0 Configure the type of USBnet interface as MBIM by <code>AT+QCFG=&quot;usbnet&quot;,2.</code></td>
<td></td>
</tr>
<tr>
<td>RG500Q series/RM500Q series/RG501Q-EU/RG502Q series/RM502Q-AE/RM505Q-AE/RM510Q-GL:</td>
<td>0x2c7c</td>
<td>0x0800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BG95 series/BG77/BG600L-M3:</td>
<td>0x2c7c</td>
<td>0x0700</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x2c7c</td>
<td>0x0700</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### UMTS/HSPA+/LTE/5G Module Series

<table>
<thead>
<tr>
<th>Option</th>
<th>Device Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>usb0</td>
<td>Configure USB composition as ECM interface mode by AT+QCFGEXT=&quot;usbnent&quot;, &quot;ecm&quot;. Corresponds to ECM USB combination: USB DM + NEMA + Modem + ECM.</td>
</tr>
</tbody>
</table>

#### BG96:
- **VID:** 0x2c7c
- **PID:** 0x0296

<table>
<thead>
<tr>
<th>Option</th>
<th>Device Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>/dev/ttyUSB0</td>
<td>DIAG</td>
</tr>
<tr>
<td>1</td>
<td>/dev/ttyUSB1</td>
<td>GNSS</td>
</tr>
<tr>
<td>2</td>
<td>/dev/ttyUSB2</td>
<td>AT command</td>
</tr>
<tr>
<td>3</td>
<td>/dev/ttyUSB3</td>
<td>Modem</td>
</tr>
</tbody>
</table>

#### EC200S series/ECG12N-EN/ECG15N series:
- **VID:** 0x2c7c
- **PID:** 0x6002

<table>
<thead>
<tr>
<th>Option</th>
<th>Device Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>usb0</td>
<td>Configure the type of USBnet interface as ECM by AT+QCFG=&quot;usbnent&quot;, 1.</td>
</tr>
<tr>
<td>1</td>
<td>/dev/cdc-wdm0</td>
<td>RmNet</td>
</tr>
</tbody>
</table>

#### EC200A series (RTOS):
- **VID:** 0x02c7c
- **PID:** 0x6005

<table>
<thead>
<tr>
<th>Option</th>
<th>Device Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>/dev/ttyUSB0</td>
<td>DIAG</td>
</tr>
<tr>
<td>3</td>
<td>/dev/ttyUSB1</td>
<td>AT command</td>
</tr>
</tbody>
</table>

#### UC200A-GL:
- **VID:** 0x02c7c
- **PID:** 0x6006

<table>
<thead>
<tr>
<th>Option</th>
<th>Device Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>/dev/ttyUSB2</td>
<td>Modem</td>
</tr>
</tbody>
</table>

#### EG912Y series:
- **VID:** 0x02c7c
- **PID:** 0x6001

#### EC200U series:
- **ECM/serial option**
  - **USB serial option**
    - **4/6**
      - **/dev/ttyUSB2** Modem
      - **/dev/ttyUSB3** Modem

---

UMTS_LTE_5G_Linux_USB_Driver_User_Guide 12 / 40
<table>
<thead>
<tr>
<th>UMTS/HSPA+/LTE/5G Module Series</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EG912U-GL/EG915U series:</strong></td>
<td><strong>USB serial option</strong></td>
</tr>
<tr>
<td>VID: 0x2c7c</td>
<td>VID: 0x2c7c</td>
</tr>
<tr>
<td>PID: 0x0901</td>
<td>PID: 0x0901</td>
</tr>
<tr>
<td><strong>RNDIS</strong></td>
<td><strong>ECM/RNDIS/NCM</strong></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>as ECM by AT+QCFG=&quot;usbnet&quot;,1.</td>
<td>as ECM by AT+QCFG=&quot;usbnet&quot;,1.</td>
</tr>
<tr>
<td>Configure the type of USBnet interface as RNDIS by AT+QCFG=&quot;usbnet&quot;,3.</td>
<td>Configure the type of USBnet interface as RNDIS by AT+QCFG=&quot;usbnet&quot;,3.</td>
</tr>
<tr>
<td>/dev/ttyUSB0 AT command</td>
<td>/dev/ttyUSB0 AT command</td>
</tr>
<tr>
<td>/dev/ttyUSB1 DIAG</td>
<td>/dev/ttyUSB1 DIAG</td>
</tr>
<tr>
<td>/dev/ttyUSB2 MOS</td>
<td>/dev/ttyUSB2 MOS</td>
</tr>
<tr>
<td>/dev/ttyUSB3 CP log</td>
<td>/dev/ttyUSB3 CP log</td>
</tr>
<tr>
<td>/dev/ttyUSB4 AP log</td>
<td>/dev/ttyUSB4 AP log</td>
</tr>
<tr>
<td>/dev/ttyUSB5 Modem</td>
<td>/dev/ttyUSB5 Modem</td>
</tr>
<tr>
<td>/dev/ttyUSB6 GNSS</td>
<td>/dev/ttyUSB6 GNSS</td>
</tr>
</tbody>
</table>

| **EG060V-EA/EG060W-EA:**        | **USB serial option** |
| VID: 0x2c7c                     | VID: 0x2c7c                     |
| PID: 0x6004                     | PID: 0x6004                     |
| **ECM/RNDIS/NCM**               | 1                               |
| 0                               | usbo                            |
| as ECM by AT+QCFG="usbnet",1.   | as ECM by AT+QCFG="usbnet",1.   |
| Configure the type of USBnet interface as RNDIS by AT+QCFG="usbnet",3. | Configure the type of USBnet interface as RNDIS by AT+QCFG="usbnet",3. |
| /dev/ttyUSB0 DIAG                | /dev/ttyUSB0 DIAG                |
| /dev/ttyUSB1 AT command          | /dev/ttyUSB1 AT command          |
| /dev/ttyUSB2 MOS                 | /dev/ttyUSB2 MOS                 |
| /dev/ttyUSB3 CP log              | /dev/ttyUSB3 CP log              |
| /dev/ttyUSB4 AP log              | /dev/ttyUSB4 AP log              |
| /dev/ttyUSB5 Modem               | /dev/ttyUSB5 Modem               |

| **EC200A series (Linux QuecOpen):** | **USB serial option** |
| VID: 0x2c7c | VID: 0x2c7c |
| PID: 0x6005 | PID: 0x6005 |
| **ECM/RNDIS/NCM** | 1 |
| 0 | usbo |
| as ECM by AT+QCFG="usbnet",1. | as ECM by AT+QCFG="usbnet",1. |
| Configure the type of USBnet interface as RNDIS by AT+QCFG="usbnet",3. | Configure the type of USBnet interface as RNDIS by AT+QCFG="usbnet",3. |
| /dev/ttyUSB0 DIAG | /dev/ttyUSB0 DIAG |
| /dev/ttyUSB1 AT command | /dev/ttyUSB1 AT command |
| /dev/ttyUSB2 Modem | /dev/ttyUSB2 Modem |

<p>| <strong>RG500U series/RG200U-CN/RM500U-CN:</strong> | <strong>USB serial option</strong> |
| VID: 0x2c7c | VID: 0x2c7c |
| PID: 0x0900 | PID: 0x0900 |
| <strong>ECM/RNDIS/NCM</strong> | 1 |
| 0 | usbo |
| as ECM by AT+QCFG=&quot;usbnet&quot;,1. | as ECM by AT+QCFG=&quot;usbnet&quot;,1. |
| Configure the type of USBnet interface as RNDIS by AT+QCFG=&quot;usbnet&quot;,3. | Configure the type of USBnet interface as RNDIS by AT+QCFG=&quot;usbnet&quot;,3. |
| /dev/ttyUSB0 DIAG | /dev/ttyUSB0 DIAG |
| /dev/ttyUSB1 Log | /dev/ttyUSB1 Log |
| /dev/ttyUSB2 AT command | /dev/ttyUSB2 AT command |
| /dev/ttyUSB3 Modem | /dev/ttyUSB3 Modem |</p>
<table>
<thead>
<tr>
<th>Device</th>
<th>VID</th>
<th>PID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EG065K series/</strong></td>
<td>0x2c7c</td>
<td>0x030b</td>
<td>Serial Port</td>
</tr>
<tr>
<td><strong>EG060K series/</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>EG120K series/</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>EM120K-GL/EM060K series</strong></td>
<td>0x2c7c</td>
<td>0x07eb</td>
<td></td>
</tr>
<tr>
<td><strong>RG520F series/RG520N series/</strong></td>
<td>0x2c7c</td>
<td>0x07eb</td>
<td></td>
</tr>
<tr>
<td><strong>RM520N series/RG530F series/RG530N-GL</strong></td>
<td>0x2c7c</td>
<td>0x07eb</td>
<td></td>
</tr>
<tr>
<td><strong>EG800Q-EU</strong></td>
<td>0x2c7c</td>
<td>0x6007</td>
<td></td>
</tr>
<tr>
<td><strong>RG500L series</strong></td>
<td>0x2c7c</td>
<td>0x7003</td>
<td></td>
</tr>
</tbody>
</table>

**USB serial option**

<table>
<thead>
<tr>
<th>Port</th>
<th>Device</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>/dev/ttyUSB0</td>
<td>DIAG</td>
</tr>
<tr>
<td>1</td>
<td>/dev/ttyUSB1</td>
<td>GNSS</td>
</tr>
<tr>
<td>2</td>
<td>/dev/ttyUSB2</td>
<td>AT command</td>
</tr>
<tr>
<td>3</td>
<td>/dev/ttyUSB3</td>
<td>Modem</td>
</tr>
<tr>
<td>4</td>
<td>wwan0</td>
<td>Configure the type of USBnet interface as RmNet by AT+QCFG=&quot;usbnet&quot;,0.</td>
</tr>
<tr>
<td>5</td>
<td>wwan0</td>
<td>Configure the type of USBnet interface as MBIM by AT+QCFG=&quot;usbnet&quot;,2.</td>
</tr>
<tr>
<td>6</td>
<td>usb0</td>
<td>Configure the type of USBnet interface as ECM by AT+QCFG=&quot;usbnet&quot;,1.</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ECM/RNDIS**

<table>
<thead>
<tr>
<th>Port</th>
<th>Device</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>/dev/ttyACM0</td>
<td>AT command</td>
</tr>
<tr>
<td>1</td>
<td>/dev/ttyACM1</td>
<td>Log</td>
</tr>
<tr>
<td>2</td>
<td>/dev/ttyACM2</td>
<td>Modem</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**RNDIS**

<table>
<thead>
<tr>
<th>Port</th>
<th>Device</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>/dev/ttyUSB0</td>
<td>RNDIS</td>
</tr>
<tr>
<td>1</td>
<td>/dev/ttyUSB1</td>
<td>RNDIS</td>
</tr>
</tbody>
</table>

**ACM**

<table>
<thead>
<tr>
<th>Port</th>
<th>Device</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>/dev/ttyACM0</td>
<td>Modem AT command</td>
</tr>
<tr>
<td>3</td>
<td>/dev/ttyACM1</td>
<td>Modem ELT</td>
</tr>
</tbody>
</table>
UMTS/HSPA+/LTE/5G Module Series

UMTS_LTE_5G_Linux_USB_Driver_User_Guide  15 / 40

### AG520R series:
- **VID:** 0x2c7c
- **PID:** 0x0452

#### USB serial option
- 0 /dev/ttyUSB0 DIAG
- 2 /dev/cdc-wdm0 wwan0
- **2 /dev/qcumi0 usb0**

#### AG55xQ series:
- **VID:** 0x2c7c
- **PID:** 0x0455

#### USB serial option
- 1 /dev/ttyUSB1 GobiNet
- 1 /dev/ttyUSB2 QMI_WWAN
- **2 /dev/ttyUSB3 usb0**
- **3 /dev/ttyUSB4 usbn0**
- **5 /dev/ttyUSB5 usbn0**
- 6 /dev/ttyUSB1 GNSS
- 7 /dev/ttyUSB2 AT command
- 8 /dev/ttyUSB3 Modem
- 9 /dev/ttyUSB4 RNDIS
- 10 /dev/ttyUSB5 RNDIS
- **11 /dev/ttyUSB1 MBIM**
- **12 /dev/ttyUSB2 MBIM**

#### NOTE

1. **GobiNet and QMI_WWAN can be ported simultaneously for Linux operating system, but only one of them can work at a time.**

2. **The device names of the modules are not fixed. If no other USB serial device is connected to user’s system, the device names of the modules start from /dev/ttyUSB0 as shown above; If another USB serial device is connected to the user’s system, the device names of the modules are determined by the number of device nodes generated by the USB serial device. For example, if a USB serial device is connected to user’s system and generates one device node, /dev/ttyUSB0 is occupied by USB serial device, then the device name of current module starts from /dev/ttyUSB1.**

3. **For more details about AT+QCFG, see document [2].**

4. **For more details about AT+QCFGEXT, see document [3].**
# 3 System Setup

This chapter describes the general structure of the USB stack in Linux, as well as how to use, compile and load USB drivers.

## 3.1. Linux USB Driver Structure

USB is a kind of hierarchical bus structure. The data transmission between USB devices and the host is realized by the USB controller. The following figure illustrates the structure of the Linux USB driver. Linux USB host driver comprises three parts: USB host controller driver, USB core and USB device drivers.

![Linux USB Driver Structure](image)

*Figure 1: Linux USB Driver Structure*

USB host controller, at the bottom of the hierarchical structure, is a USB driver which directly interacts with the hardware.
USB core, the core of the whole USB host driver, is used for managing USB bus, USB bus devices and USB bus bandwidth; it provides the interfaces for USB device drivers, through which the applications can access the USB system files.

USB device drivers interact with the applications and provide the interfaces for accessing specific USB devices.

### 3.2. USB Serial Option Driver

Once the module loads the USB serial option driver successfully, the device files named `ttyUSB0`, `ttyUSB1`, `ttyUSB2` and so on are created in directory `/dev`.

The following chapters show how to port USB serial option driver into the Linux operating system.

#### 3.2.1. Add VID and PID

To identify the module, add the module VID and PID information to the `[KERNEL]/drivers/usb/serial/option.c` file and the corresponding VID and PID can be obtained in Table 1.

**EC25 series Example:**

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

#### 3.2.2. Use USBNet Driver

The configuration in Chapter 3.2.1 makes all USB interfaces of the module attach to USB serial option driver, which prevents USBNet driver interfaces from working. You can add the following statements to prevent USBNet driver interfaces from attaching to USB serial option driver.

- For Linux kernel versions beyond 2.6.30, you can add the following statements to the `[KERNEL]/drivers/usb/serial/option.c` file.

```c
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);
    }
```

---
For Linux kernel versions before 2.6.31, you can add the following statements to the [KERNEL]/drivers/usb/serial/option.c file.

```c
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
    /* Store device id so we can use it during attach. */
    usb_set_serial_data(serial, (void *)id);
    return 0;
}
```
3.2.3. Modify Kernel Configuration

You need to enable the following configuration items. See Chapter 3.7 for the explanation how to configure the kernel.

- CONFIG_USB_SERIAL
- CONFIG_USB_SERIAL_WWAN
- CONFIG_USB_SERIAL_OPTION

3.2.4. Add Zero Packet Mechanism

As required by the USB protocol, the mechanism for processing zero packets needs to be added during bulk-out transmission by adding the following statements.

- For Linux kernel version beyond 2.6.34, you need to add the following statements to the [KERNEL]/drivers/usb/serial/usb_wwan.c file.

```c
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;
```

- For Linux kernel version below 2.6.35, you need to add the following statements to the [KERNEL]/drivers/usb/serial/option.c file.

```c
```
3.2.5. Add Reset-Resume Mechanism

When MCU enters the Suspend/Sleep mode, some USB host controllers/USB hubs will lose power or reset and cannot be used for module resume after MCU exits from the Suspend/Sleep mode. The reset-resume mechanism needs to be enabled by adding the following statements.

- For Linux kernel version beyond 3.4, you need to add the following statements to the [KERNEL]/drivers/usb/serial/option.c file.

```c
static struct usb_serial_driver option_1port_device = {
    .suspend = usb_wwan_suspend,
    .resume = usb_wwanResume,
    #if 1  // Added by Quectel
    .reset_resume  = usb_wwanResume,
    #endif
};
```

- For Linux kernel version below 3.5, you need to add the following statements to the [KERNEL]/drivers/usb/serial/usb-seria/serial.c file.

```c
/* Driver structure we register with the USB core */
static struct usb_driver usb_serial_driver = {
```
3.2.6. Increase Quantity and Capacity of Bulk Out URBs

For Linux kernel version below 2.6.29, the quantity and capacity of the bulk out URBs need to be increased to get faster uplink speed by adding the following statements to the [KERNEL]/drivers/usb/serial/option.c file.

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

3.3. GobiNet Driver

After the module loads the GobiNet driver successfully, a network device and a QMI device node are created. The network device is named `usbX` and the QMI device node is `/dev/qcmiX`. The network device is used for data transmission, and QMI device node is used for QMI message interaction.

The following chapters explain how to port the GobiNet driver into the Linux operating system.

3.3.1. Modify Driver Source Codes

The GobiNet driver is provided by Quectel in the form of the source file with source codes. The source file should be copied to [KERNEL]/drivers/net/usb/ (or [KERNEL]/drivers/usb/net/ if the kernel version is below 2.6.22).

3.3.2. Modify Kernel Configuration

You need to enable the following configuration item first. See Chapter 3.7 for the explanation how to configure the kernel.
3.4. QMI_WWAN Driver

After the module loads the QMI_WWAN driver successfully, a network device and a QMI device node are created. The network device is named `wwanX` and the QMI device node is `/dev/cdc-wdmX`. The network device is used for data transmission, and QMI device node is used for QMI message interaction.

The following chapters explain how to port the QMI_WWAN driver into the Linux operating system.

3.4.1. Modify Source Codes of the Driver

The source file containing source codes of QMI_WWAN driver is `[KERNEL]/drivers/net/usb/qmi_wwan.c`. To use the QMI_WWAN driver along with the Quectel module, the source file needs to be modified.

To simplify works, Quectel provides the source file `qmi_wwan_q.c`, which can coexist with `qmi_wwan.c` and only be used for Quectel's modules. The source file `qmi_wwan_q.c` should be copied to `[KERNEL]/drivers/net/usb/`.

3.4.2. Modify Kernel Configuration

Enable the following configuration items first. See Chapter 3.7 for the explanation how to configure the kernel.

Then add the following statements to `[KERNEL]/drivers/net/usb/Makefile`.

```bash
# 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 Driver

ACM, ECM, RNDIS, NCM and MBIM drivers are USB interface class drivers. The Linux system automatically attaches to the corresponding driver according to the interface class, sub-class and protocol. These drivers are available in the upstream Linux releases if you use Linux distribution such as Ubuntu and Fedora. The drivers are automatically loaded when the module is connected to the Linux PC through the USB interface. If you use an embedded system, you just need to enable the corresponding configuration items.

The configuration items to be enabled for each driver:

Table 2: Configuration Items for USB Class Drivers

<table>
<thead>
<tr>
<th>USB Driver</th>
<th>Configuration Item</th>
<th>Source File</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM</td>
<td>CONFIG_USB_ACM</td>
<td>[KERNEL]/drivers/net/usb/cdc-acm.c</td>
</tr>
</tbody>
</table>
| 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_MBIN | [KERNEL]/drivers/net/usb/cdc_mbin.c |

3.6. How to Support PPP

Before using PPP function, you need to enable the following configuration items to configure the kernel to support PPP. See Chapter 3.7 for the explanation how to configure the kernel.

CONFIG_PPP
CONFIG_PPP_ASYNC
CONFIG_PPP_SYNC_TTY
CONFIG_PPP_DEFLATE
3.7. Configure Kernel

To configure the kernel follow the steps and the corresponding commands below.

**Step 1:** Execute the following command to switch to kernel directory:

```
cd <your kernel directory>
```

**Step 2:** Execute the following command to set environment variables and import the board’s “defconfig” file (Raspberry Pi board example is given below).

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

**Step 3:** Execute the following command to compile the kernel.

```
make menuconfig
```

**Step 4:** Enable the configuration item.

Selecting `<*>` means to compile the driver to kernel image.

Selecting `<M>` means to compile the driver as module.

Taking USB serial option as an example, you can enable `CONFIG_USB_SERIAL_OPTION` with the options below to compile the USB serial option driver into the kernel image.

```
Figure 2: Configure USB Serial Option in Kernel
```
3.8. Install and Load Driver as a Kernel Module for PC in Linux

For developers who are required to test Quectel modules on PC with Linux operating system like Ubuntu, Quectel can provide source files of USB serial option/GobiNet/qmi_wwan_q drivers. These USB drivers can be installed and used by executing the following commands and then rebooting the PC.

- Install QMI_WWAN driver.

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

- Install GobiNet driver.

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

- Install USB serial option driver.

```bash
# First use command `uname -r` to query the current 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 Test Module

Generally, AT and PPP functions are supported. If a USBNet driver has been installed, the USB network adapter function can also be used on the module. The following chapters explain how to test these functions.

4.1. Test AT Function

After the module is connected and the USB driver is loaded successfully, several device files are created in the /dev directory.

The AT port is the ttyUSB port created by the USB serial option driver. See Table 1 to view the device name corresponding to AT Command or Modem function.

UART port tools such as “minicom” or “busybox microcom” shown below can be used to test AT function.

![AT Command Test Result](image)

Figure 3: AT Command Test Result

4.2. Test PPP Function

If the module supports USBNet drivers, it is recommended to use USBNet driver interface.

PPP dial-up is more complex than network card dial-up, and it causes higher current consumption of CPU, so it is not recommended to use PPP dial-up.

To set up a PPP call, the following files are required. Check if the following files exist in your product:
1. PPPD and chat programs. If the two programs do not exist, you can download their source codes from https://ppp.samba.org/download.html and port them to the module.

2. One PPP script file named /etc/ppp/ip-up, which is used to set DNS. If there is no such file, use linux-ppp-scripts/ip-up provided by Quectel.

3. Three scripts named quectel-ppp, quectel-chat-connect and quectel-chat-disconnect. They are provided by Quectel in the /linux-ppp-scripts/ directory. You may need to make corresponding changes based on different modules. For more information, refer to linux-ppp-scripts/readme.

Copy quectel-ppp, quectel-chat-connect and quectel-chat-disconnect to the directory /etc/ppp/peers, then set up a PPP call by executing the following command:

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

The process of PPP call setup is shown below:

```
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 0x588fb7f> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x0 <asyncmap 0x0> <auth chap MD5> <magic 0xea02c208> <pcomp> <accomp>]
```
Now a PPP call is set up successfully.

Please use the following commands to check whether the information of IP, DNS and route in your system belongs to Quectel modules.

```
# 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
```
The following commands can be used to terminate the PPPD process to disconnect a PPP call:

```
# killall pppd
```

Terminating on signal 15
Connect time 0.4 minutes.
Sent 0 bytes, received 0 bytes.

**NOTE**

PPP call is not supported on Quectel 5G module series and LTE module series with data rates higher than Cat 4.

### 4.3. Test GobiNet/QMI_WWAN Driver

To test the GobiNet or QMI_WWAN driver, follow the steps below:

**Step 1:** Compile the Connect Manager program with the following commands. Quectel provides a Connect Manager program (“quectel-CM”) in the form of source code in the directory `/quectel-CM/` for you to set up data connection manually.

- For PC Linux:

  ```
  # make
  ```
For embedded Linux:

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

Replace `arm-none-linux-gnueabi` with the cross compiler on the module.

Program "quectel-CM" will be output in this step.

**Step 2:** Prepare “busybox udhcpc” tool.

quectel-CM calls “busybox udhcpc” to obtain IP and DNS, and “busybox udhcpc” calls script file `/usr/share/udhcpc/default.script` to set IP, DNS and routing table for Linux board.

You can download the source codes of “busybox udhcpc” tool from [https://busybox.net](https://busybox.net), then enable `CONFIG_UDHCPC` with the command below and copy the `[BUSYBOX]/examples/udhcp/simple.script` script file to Linux board (renamed as `/usr/share/udhcpc/default.script`).

```bash
busybox menuconfig
```

**Step 3:** Use “quectel-CM” to set up a data call.

After the module is connected and the GobiNet or QMI_WWAN driver is installed successfully, a USB network adapter and a QMI device node are created. The USB network adapter of the GobiNet driver is named `usbX`, and the QMI device node is `/dev/qcqmIX`. The USB network adapter of the QMI_WWAN driver is named `wwanX`, and the QMI channel is `/dev/cdc-wdmX`.

quectel-CM sends QMI messages to the module through QMI device node to set up a data connection. See document [1] for the usage of quectel-CM.

The working process of quectel-CM is shown below (EM12-G running QMI_WWAN driver example):

```bash
root@cqh6:~# ./quectel-CM/quectel-CM &
[07-03_06:56:28:172] WCDMA&LTE_QConnectManager_Linux&Android_V1.3.4
[07-03_06:56:28:174] Find /sys/bus/usb/devices/2-1.2 idVendor=2c7c idProduct=0512
[07-03_06:56:28:174] Find usbnadpater = 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
```
Step 4: Use the following commands to check the information about IP, DNS and route.

```bash
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
```

Step 5: Use the following command to terminate the quectel-CM process to disconnect the data connection:

```bash
root@cqh6:~# killall quectel-CM
[07-03_07:00:10:145] requestDeactivateDefaultPDP err = 0
[07-03_07:00:10:145] ifconfig wwan0 down
```
4.4. Test "AT$QCRMCALL" on GobiNet/QMI_WWAN Driver

This chapter explains how to use AT$QCRMCALL to set up a data call.

Although it is recommended to use QMI tools like quectel-CM/libqmi/uqmi to set up a data call, if your MCU’s USB Host Controller does not fully support USB interrupt type endpoint, you need to use AT$QCRMCALL instead of QMI tools.

For GobiNet driver, to use AT$QCRMCALL, “qcrmcall_mode” in GobiUSBNet.c needs to be modified to “1”. While for QMI_WWAN driver, no extra modification is required.

The following logs show how to use AT$QCRMCALL to set up a data call. For details, contact Quectel Technical Support.

```
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:~#
```
4.5. Test QMAP on GobiNet/QMI_WWAN Driver

This chapter explains how to test the QMAP (Qualcomm Multiplexing and Aggregation Protocol) on GobiNet or QMI_WWAN driver, especially for developers using GobiNet or QMI_WWAN driver and requiring QMAP.

When GobiNet or QMI_WWAN driver are used, only one physical network card can be created by default, so only one PDN data call can be set up. However, by using the multiplexing protocol, multiple virtual network cards can be created over one physical network card, so multiple PDN data calls can be set up.

When GobiNet or QMI_WWAN driver are used, only one IP Packet in one URB can be transferred, which can lead to Host CPU overload in case of high throughput and frequent URB interrupts. However, the aggregation protocol can be used to transfer multiple IP Packs in one URB with increased throughput by reducing the number of URB interrupts.

If multiplexing or aggregation protocol is needed, contact Quectel Technical Support support@quectel.com.

4.6. Test ECM/RNDIS/NCM/MBIM Driver

On modules using ECM, RNDIS and NCM drivers, you can also perform a call by executing quectel-CM provided by Quectel (see document [1] for details). You can contact Quectel Technical Support for details if needed.

In MBIM mode, MBIM tools like “mbimcli” and “umbim” can be used to set up a data call. quectel-CM, which is provided by Quectel, can also be used to set up a data call.
5 Power Management

The USB system in Linux has two advanced power management features: USB Auto Suspend and USB Remote Wakeup. This chapter explains how to enable these features, particularly for developers in need.

If USB communication between the USB host and the USB devices has been idle for some time (for example 3 seconds), the USB host can make the USB devices enter Suspend mode automatically. This feature is called USB Auto Suspend.

USB Remote Wakeup allows a suspended USB device to remotely wake up the USB host over the USB, which may also be suspended (for example, in Deep Sleep mode). The USB device, which has a remote wakeup capability, performs an activity to wake up the USB host, and then the USB host is woken up by the remote activity.

USB Auto Suspend and USB Remote Wakeup features of the drivers described in this document except USB serial option driver are enabled by default.

5.1. Enable USB Auto Suspend

For USB serial option driver, add the following statements to `option_probe()` in the `[KERNEL]/drivers/usb/serial/option.c` file to enable USB Auto Suspend feature.

```c
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. Enable USB Remote Wakeup

For USB serial option driver, add the following statements to option_probe() in the [KERNEL]/drivers/usb/serial/option.c file to enable USB Remote Wakeup feature.

```c
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); //usb remote wakeup
    }
    #endif
    /* Store device id so we can use it during attach. */
    usb_set_serial_data(serial, (void *)id);
    return 0;
}
```
6 FAQs and Kernel Log

6.1. How to Check If USB Driver Exists in Module

The existence of the USB driver can be checked from the content of the /sys/bus/usb/drivers directory.

Example:

<table>
<thead>
<tr>
<th>Driver</th>
<th>Subdirectory</th>
<th>Subdirectory</th>
<th>Subdirectory</th>
<th>Subdirectory</th>
</tr>
</thead>
<tbody>
<tr>
<td>GobiNet</td>
<td>cdc_wdm</td>
<td>rndis_host</td>
<td>usbf</td>
<td></td>
</tr>
<tr>
<td>cdc_ether</td>
<td>hub</td>
<td>uas</td>
<td>usbserial</td>
<td></td>
</tr>
<tr>
<td>cdc_mbim</td>
<td>option</td>
<td>usb</td>
<td>usbselasticGeneric</td>
<td></td>
</tr>
<tr>
<td>cdc_ncm</td>
<td>qmi_wwan_q</td>
<td>usb-storage</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If the USB serial option driver is required, please make sure option exists in the content of the /sys/bus/usb/drivers directory.

Similarly, if GobiNet driver is required, make sure GobiNet exists.

If QMI_WWAN driver is required, make sure qmi_wwan_q exists, and so forth.

6.2. How to Check If Module Works Well with Corresponding USB Driver

This chapter shows the module kernel log with the corresponding installed USB driver in the Linux operating system. If the module does not work well, compare the module kernel log with that in this chapter to help you with troubleshooting.

- For USB serial option and GobiNet driver: Kernel logs of different modules are almost the same except for the VID&PID information (marked in red in the following figure). USB serial option and GobiNet for RG502Q series module example is as follows:
For USB serial option and QMI_WWAN driver: kernel logs of different modules are almost the same except for the VID&PID information (framed in red in the following figure). USB serial option and QMI_WWAN for RG502Q series module example is as follows:
6.3. How to Check Which USB Driver is Installed

This chapter explains how to query the name of the USB driver for the Quectel module USB interface. The USB driver name is identified by the keyword “Driver=”. If “Driver=None”, the reason may be that the corresponding configuration item is not enabled in your kernel configuration, or the VID and PID of Quectel modules are not inserted into the corresponding USB driver source files. In such a case, follow the steps referred to in Chapter 2 to check again.

USB interface and driver for RG502Q series modules example is as follows:

```
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=0000 Rev= 4.14
S: Manufacturer=Quectel
S: Product=RG502Q-EA
S: SerialNumber=39249701
C:* #Ifs= 5 Cfg# 1 Atr=0 MxPwr=886mA
I:* If# 0 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=00 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= 18 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= 18 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= 18 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=8a(I) Atr=02(Bulk) MxPS=1024 Ivl=0ms
E: Ad=0f(O) Atr=02(Bulk) MxPS=1024 Ivl=0ms
```

Figure 6: USB Interface and Driver for RG502Q Series Modules
7 Appendix References

Table 3: Related Documents

<table>
<thead>
<tr>
<th>Document Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1] Quectel_QConnectManager_Linux_User_Guide</td>
</tr>
<tr>
<td>[2] Quectel_EC2x&amp;EG2x&amp;EG9x&amp;EM05_Series_QCFG_AT_Commands_Manual</td>
</tr>
</tbody>
</table>

Table 4: Terms and Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM</td>
<td>Abstract Control Model</td>
</tr>
<tr>
<td>AP</td>
<td>Application Processor</td>
</tr>
<tr>
<td>APN</td>
<td>Access Point Name</td>
</tr>
<tr>
<td>CP</td>
<td>Coprocessor</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>ECM</td>
<td>Ethernet Control Model</td>
</tr>
<tr>
<td>EHCI</td>
<td>Enhanced Host Controller Interface</td>
</tr>
<tr>
<td>GNSS</td>
<td>Global Navigation Satellite System</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>HCD</td>
<td>Host Controller Driver</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>MBIM</td>
<td>Mobile Interface Broadband Model</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>MCU</td>
<td>Microcontroller Unit</td>
</tr>
<tr>
<td>NCM</td>
<td>Network Control Model</td>
</tr>
<tr>
<td>NDIS</td>
<td>Network Driver Interface Specification</td>
</tr>
<tr>
<td>NMEA</td>
<td>National Marine Electronics Association</td>
</tr>
<tr>
<td>OHCI</td>
<td>Open Host Controller Interface</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PDN</td>
<td>Packet Data Network</td>
</tr>
<tr>
<td>PID</td>
<td>Product ID</td>
</tr>
<tr>
<td>PPP</td>
<td>Point to Point Protocol</td>
</tr>
<tr>
<td>QMAP</td>
<td>Qualcomm Multiplexing and Aggregation Protocol</td>
</tr>
<tr>
<td>QMI</td>
<td>Qualcomm Messaging Interface</td>
</tr>
<tr>
<td>UHCI</td>
<td>Universal Host Controller Interface</td>
</tr>
<tr>
<td>URB</td>
<td>USB Request Block</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>VID</td>
<td>Vendor ID</td>
</tr>
</tbody>
</table>