SIERRA WIRELESS IS NOW
SEMTECH®

Modules

Linux USB Driver User Guide

RC71xx

October 2024

41114799 Rev. 1

Important Notice

Information relating to this product and the application or design described herein is believed to be reliable, however such information is provided as a guide only and Semtech assumes no liability for any errors in this document, or for the application or design described herein.

Semtech reserves the right to make changes to the product or this document at any time without notice. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. Semtech warrants performance of its products to the specifications applicable at the time of sale, and all sales are made in accordance with Semtech's standard terms and conditions of sale.

SEMTECH PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS, OR IN NUCLEAR APPLICATIONS IN WHICH THE FAILURE COULD BE REASONABLY EXPECTED TO RESULT IN PERSONAL INJURY, LOSS OF LIFE OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. INCLUSION OF SEMTECH PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE UNDERTAKEN SOLELY AT THE CUSTOMER'S OWN RISK. Should a customer purchase or use Semtech products for any such unauthorized application, the customer shall indemnify and hold Semtech and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs damages and attorney fees which could arise.

Wireless Communications

Due to the nature of wireless communications, transmission and reception of data can never be guaranteed. Data may be delayed, corrupted (i.e., have errors) or be totally lost. The Semtech product should not be used in situations where failure to transmit or receive data could result in damage of any kind to the user or any other party, including but not limited to personal injury, death, or loss of property. Semtech accepts no responsibility for damages of any kind resulting from delays or errors in data transmitted or received using the Semtech product, or for failure of the Semtech product to transmit or receive such data.

Safety

Do not operate the Semtech product in areas where blasting is in progress, where explosive atmospheres may be present, near medical equipment, near life support equipment, or near any equipment which may be susceptible to any form of radio interference. In such areas, the Semtech product should be powered off.

Qualcomm licenses

Semtech's cellular modules are sold subject to certain notices and restrictions regarding patent licenses from Qualcomm Incorporated. These notices and restrictions are available at www.sierrawireless.com/qualcomm-notices.

Sierra Wireless

Semtech Corporation purchased Sierra Wireless in January 2023. The Sierra Wireless brand is gradually being phased out. During the phase-out period, references to both “Semtech” and “Sierra Wireless” may appear in product documentation.

Contact Information

Sales information and technical support, including warranty and returns	Web: sierrawireless.com/company/contact-us/ Global toll-free number: 1-877-687-7795 6:00 am to 5:00 pm PST
Corporate and product information	Web: sierrawireless.com

Revision History

Revision number	Release date	Changes
1	September 2024	Creation

Contents

1: Introduction

RC71xx modules (RC7110, RC7120) support RNDIS, ECM, and PPP network communications protocols using Linux drivers provided by Semtech.

This document describes how to configure a Linux host product's RC71xx module to use each protocol type to establish network connections.

Note: Examples shown in this document are for a host product running Ubuntu 18.04 LTS.

2: PDP Contexts

This chapter describes how to define, activate and test PDP contexts on an RC71xx module, using AT commands.

For full details of each AT command, refer to *[1] RC71xx AT Command Reference (Doc# 41114675)*.

2.1 Define a PDP Context

Use the +CGDCONT AT command to define or check the configuration of a specific PDP context. Table 2-1 describes the command arguments (parameters) required for the examples shown in this section.

Table 2-1: +CGDCONT[a] — Define PDP context

Format: AT+CGDCONT=<cid>,<PDP_type>,<APN>		
Parameter	**Value**	**Description**
<cid>	1–16	PDP context identifier
<PDP_type>	"IP"	Packet Data Protocol Type — IPv4 only
	"IPV6"	Packet Data Protocol Type — IPv6 only
	"IPV4V6"	Packet Data Protocol Type — IPv4v6
<APN>	"internet"	Access Point Name The <APN> string varies by carrier %[confirm]. "internet" is used as an example only.

a. For full command details, including additional command parameters, refer to *[1] RC71xx AT Command Reference (Doc# 41114675)*.

Example — Configure PDP context for IPv4 only

To configure PDP context 1 for IPv4 only:

1. Use <PDP_type>="IP":

```
at+cgdcont=1,"IP","internet"
OK
```

2. Optionally, confirm the context updated correctly:

```
at+cgdcont?
+CGDCONT: 1,"IP","internet","10.120.111.24",,,,,,,,,,
OK
```

Example — Configure PDP context for IPv6 only

To configure PDP context 1 for IPv6 only:

1. Use <PDP_type>="IPV6":

```
at+cgdcont=1,"IPV6","internet"
OK
```

2. Optionally, confirm the context updated correctly:

```
at+cgdcont?
+CGDCONT:
1,"IPV6","internet","36.1.225.128.136.176.181.93.167.55.51.161.231.141.19.152
",,,,,,,,,,,
```

Example — Configure PDP context for IPv4v6 (IPv4 and IPv6)

To configure PDP context 1 for both IPv4 and IPv6:

1. Use <PDP_type>="IPV4V6":

```
at+cgdcont=1,"IPV4V6","internet"
OK
```

2. Optionally, confirm the context updated correctly:

```
at+cgdcont?
+CGDCONT: 1,"IPV4V6","internet","10.144.70.73",,,,,,,,,,
OK
```

2.2 Activate PDP Context

Use the +CGACT AT command to activate a specific PDP context or check its activation status. Table 2-2 describes the command arguments (parameters) required for the examples shown in this section.

Table 2-2: +CGACT[a] — Activate/deactivate PDP context

Format: AT+CGACT=<state>,<cid>		
Parameter	**Value**	**Description**
<state>	0	Deactivated
	1	Activated
<cid>	1–16	PDP context Identifier

a. For full command details, including additional command parameters, refer to *[1] RC71xx AT Command Reference (Doc# 41114675).*

Example — Activate PDP Context

To activate PDP context 1:

1. Set the <state>=1:

```
at+cgact=1,1
OK
```

2. Optionally, confirm the context updated correctly:

```
at+cgact?
+CGACT: 1,1
OK
```

2.3 Test Network Availability

Use the +CGPADDR AT command to display the IP address assigned to the module by the operator, and use the +COPS AT command to show the network operator and RAT (radio access technology) in use. Table 2-3 and Table 2-4 describe the command arguments (parameters) required for the examples shown in this section.

Table 2-3: +CGPADDR[a] — Display module's PDP context addresses

Format: AT+CGPADDR=<cid>		
Parameter	**Value**	**Description**
<cid>	1–15	PDP context Identifier

a. For full command details, refer to *[1] RC71xx AT Command Reference (Doc# 41114675).*

Table 2-4: +COPS[a] — Operator selection

Format: AT+COPS? Response format: +COPS: <mode>[,<format>,<oper>][,<AcT>]		
Parameter	**Value**	**Description**
<mode>	0	Automatic
	1	Manual
	2	Deregister from network
	3	Set only
	4	Manual / automatic
<format>	0	Long format alphanumeric
	1	Short format alphanumeric
	2	Numeric
<oper>	string	Operator
<AcT>	7	RAT selected (E-UTRAN, e.g., LTE)

a. For full command details, refer to *[2] AT Command Set for User Equipment (UE) (Release 6) (Doc# 3GPP TS 27.007).*

Example — Display PDP Context and Network Operator Details

To display network details for PDP context 1:

1. Display the IP address assigned to the PDP context by the network operator:

```
AT+CGPADDR=1
+CGPADDR:
1,"10.24.53.159","36.1.225.128.136.64.69.63.250.146.175.234.253.246.6.223"
OK
```

2. Display the network operator's information (including the access technology):

```
AT+COPS?
+COPS: 0,2,"46601",7
OK
```

3: RNDIS Network Interface

This chapter describes how to configure the module's USB network interface to use the RNDIS (Remote Network Driver Interface Specification) protocol.

3.1 RNDIS Configuration via Menu

Make sure relevant Linux kernel configurations are set correctly:

1. Make sure the following RNDIS drivers are defined as shown:

```
CONFIG_USB_USBNET=m
CONFIG_USB_NET_RNDIS_HOST=m
```

2. Make sure the USB cdc_acm driver configuration is defined as shown:

```
CONFIG_USB_ACM=m
```

3.2 RNDIS USB Network Interface Configuration

Use the !USBCOMP AT command to configure the USB network interface to use RNDIS.

Table 3-1: !USBCOMP — Set/report USB interface configuration[a]

Format: AT!USBCOMP=<Config Index>,<Config Type>,<Interface bitmask>		
Parameter	**Value**	**Description**
<Config Index>	1	Configuration index to which the composition applies, should be 1
<Config Type>	1	Generic
	2	Network Interface (Note: <Interface bitmask> must include RNDIS or ECM, but not both.)
<Interface bitmask>	00000001	Log DIAG
	00000008	AT (Note: This interface cannot be disabled. (The command returns ERROR if not selected.)
	00000010	AT_PPP
	00001000	**RNDIS**
	00002000	ECM

a. For full command details, refer to *[1] RC71xx AT Command Reference (Doc# 41114675).*

To configure the module to use RNDIS:

1. Set the USB configuration to include RNDIS (e.g., "1019" enables the Log, AT, AT_PPP and RNDIS interfaces):

```
AT!USBCOMP=1,2,1019
OK
```

2. Reboot the module to make the configuration change take effect:

```
AT!RESET
```

3.3 Enable the Network Connection

The !NETDEVCTL AT command is used to bind a PDP context to the host to establish a network connection, or to unbind the context to deactivate a connection.

Table 3-2: !NETDEVCTL[a] — Bind/Unbind IP address with host for NIC USB interface

Format: AT!NETDEVCTL=<op>,cid>[<urc_en>]		
Parameter	**Value**	**Description**
<op>	0	Unbind IP address of PDP context <cid> from the host
	1	Bind the IP address of PDP context <cid> to the host one time only. If the context is deactivated and then reactivated, do not automatically rebind.
	2	Bind the IP address of PDP context <cid> to the host. If the context is deactivated and then reactivated, automatically rebind.
	3	Auto-connect and bind <cid> when the module powers on. This setting is persistent.
<cid>	1–15	PDP Context ID
<urc_en>	0	Disable URC (unsolicited response code)
	1	Enable URC (unsolicited response code)

a. For full command details, refer to *[1] RC71xx AT Command Reference (Doc# 41114675).*

To enable automatic network connection for PDP context 1, which automatically establishes a network connection at power on and automatically re-establishes a network connection if the connection drops unexpectedly, make sure to set the <op> parameter to 3:

1. Use <op>=3 to enable automatic network connection:

```
AT!NETDEVCTL=3,1,1
OK
```

3.4 Check RNDIS Network Connection

To check whether the RNDIS-based network connection established correctly:

1. List the enumerated USB devices and confirm the rndis_host driver is present (highlighted **in red** below):

```
#lsusb -t
/:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/3p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/8p, 480M
        |__ Port 3: Dev 3, If 0, Class=Vendor Specific Class,
Driver=i2400m_usb, 480M
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/3p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/6p, 480M
        |__ Port 1: Dev 3, If 0, Class=Human Interface Device, Driver=usbhid,
1.5M
        |__ Port 2: Dev 8, If 3, Class=CDC Data, Driver=cdc_acm, 480M
        |__ Port 2: Dev 8, If 1, Class=CDC Data, Driver=rndis_host, 480M
        |__ Port 2: Dev 8, If 6, Class=Communications, Driver=cdc_acm, 480M
```

If rndis_host did not enumerate, redo 3.1 RNDIS Configuration via Menu, 3.2 RNDIS USB Network Interface Configuration, and 3.3 Enable the Network Connection.

2. Find the network adapter interface name for the RNDIS port in the kernel messages (highlighted **in red** below):

```
#dmesg
[30570.814042] usb 1-1.2: new high-speed USB device number 15 using ehci-pci
[30570.924725] usb 1-1.2: New USB device found, idVendor=1199,
idProduct=9114, bcdDevice= 2.00
[30570.924729] usb 1-1.2: New USB device strings: Mfr=1, Product=2,
SerialNumber=3
[30570.924731] usb 1-1.2: Product: RC7110
[30570.924734] usb 1-1.2: Manufacturer: Sierra Wireless, Incorporated
[30570.924736] usb 1-1.2: SerialNumber: 000000000001
[30570.929418] rndis_host 1-1.2:1.0 eth0: register 'rndis_host' at usb-
0000:00:1a.0-1.2, RNDIS device, 20:89:84:6a:96:ab
[30570.930654] cdc_acm 1-1.2:1.2: ttyACM0: USB ACM device
[30570.931820] cdc_acm 1-1.2:1.4: ttyACM1: USB ACM device
[30570.933095] cdc_acm 1-1.2:1.6: ttyACM2: USB ACM device
[30571.180775] rndis_host 1-1.2:1.0 enx2089846a96ab: renamed from eth0
```

For the example above, **enx2089846a96ab** is the network adapter interface name for the module's RNDIS port.

3. The interface's IP address is assigned automatically based on the PDP context's configured PDP type (i.e., IP, IPV6, IPV4V6).

Using the example interface above (**enx2089846a96ab**), test the network connection:

- IPV4 connection expected:

i. Display the interface configurations to confirm the IPv4 RNDIS connection was established:

```
#ifconfig
enx2089846a96ab: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.31.117.95  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::2e2d:adc3:c608:7d9b  prefixlen 64  scopeid
0x20<link>
        ether 20:89:84:6a:96:ab  txqueuelen 1000  (Ethernet)
        RX packets 19  bytes 2940 (2.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 54  bytes 9638 (9.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

ii. Ping the google domain name server (8.8.8.8) to test IPv4, and ping www.google.com to test IPv6:

```
#ping 8.8.8.8

PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=54 time=116 ms
...
64 bytes from 8.8.8.8: icmp_seq=9 ttl=54 time=42.8 ms
--- 8.8.8.8 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8009ms
rtt min/avg/max/mdev = 26.045/43.315/116.211/26.581 ms

#ping -6 www.google.com
connect: Network is unreachable
```

This example confirms the RNDIS connection was established for IPv4 communication only (ping 8 .8.8.8 succeeded, and ping www.google.com failed).

- IPV6 connection expected:

  i. Display the interface configurations to confirm the IPv6 RNDIS connection was established:

```
#ifconfig
enx2089846a96ab: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
       inet6 2401:e180:8844:a56d:156b:4f37:e7da:808  prefixlen 64  scopeid
0x0<global>
        inet6 fe80::2e2d:adc3:c608:7d9b  prefixlen 64  scopeid 0x20<link>
        inet6 2401:e180:8844:a56d:6c19:cc0b:43a1:5f21  prefixlen 64
scopeid 0x0<global>
        ether 20:89:84:6a:96:ab  txqueuelen 1000  (Ethernet)
        RX packets 13  bytes 1629 (1.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 52  bytes 11060 (11.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

  ii. Ping the google domain name server (8.8.8.8) to test IPv4, and ping www.google.com to test IPv6:

```
#ping 8.8.8.8
connect: Network is unreachable


#ping www.google.com

PING www.google.com(tsa03s06-in-x04.1e100.net (2404:6800:4012:1::2004))
56 data bytes
64 bytes from tsa03s06-in-x04.1e100.net (2404:6800:4012:1::2004):
icmp_seq=1 ttl=55 time=29.1 ms...
64 bytes from tsa03s06-in-x04.1e100.net (2404:6800:4012:1::2004):
icmp_seq=12 ttl=55 time=30.9 ms

--- www.google.com ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11017ms
rtt min/avg/max/mdev = 25.889/33.677/45.820/6.150 ms
```

  This example confirms the RNDIS connection was established for IPv6 communication only (ping 8.8.8.8 failed and ping www.google.com succeeded).

- IPV4V6 connection expected:

  i. Display the interface configurations to confirm the IPv4v6 RNDIS connection was established:

```
#ifconfig
enx2089846a96ab: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.144.99.6  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 2401:e180:8813:7006:cde5:d823:7927:aacf  prefixlen 64
scopeid 0x0<global>
        inet6 fe80::2e2d:adc3:c608:7d9b  prefixlen 64  scopeid
0x20<link>
        inet6 2401:e180:8813:7006:51b6:4179:2959:e322  prefixlen 64
scopeid 0x0<global>
        ether 20:89:84:6a:96:ab  txqueuelen 1000  (Ethernet)
        RX packets 20  bytes 3185 (3.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 73  bytes 14425 (14.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

ii. Ping the google domain name server (8.8.8.8) to test IPv4, and ping www.google.com to test IPv6:

```
#ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=55 time=113 ms
...
64 bytes from 8.8.8.8: icmp_seq=10 ttl=55 time=54.9 ms

--- 8.8.8.8 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms
rtt min/avg/max/mdev = 25.585/45.139/113.699/24.145 ms

#ping www.google.com
PING www.google.com(tsa01s11-in-x04.1e100.net (2404:6800:4012:2::2004))
56 data bytes
...
64 bytes from tsa01s11-in-x04.1e100.net (2404:6800:4012:2::2004):
icmp_seq=7 ttl=55 time=36.1 ms

--- www.google.com ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6010ms
rtt min/avg/max/mdev = 24.857/31.487/38.641/4.836 ms
```

This example confirms the RNDIS connection was established for IPv4v6 communication (both ping 8.8.8.8 and ping www.google.com succeeded).

3.5 RNDIS Network Reconnect

The !NETDEVCTL <op> used in 3.3 Enable the Network Connection determines how the context reconnects to a network after being deactivated (e.g., when network service is lost and then regained).

- If <op> = 3 (as suggested in 3.3 Enable the Network Connection), the context automatically binds and connects when the module powers on, and automatically reconnects after network service is lost and regained.
- If <op> = 0 or 1 was used, the context does not automatically reconnect.
- If <op> = 2 was used, the context automatically reconnects after it was deactivated.

3.6 RNDIS Disconnect

To disconnect (unbind) the host from a PDP context (e.g., context 1):

1. Use <op>=0 to unbind the host:

```
AT!NETDEVCTL=0,1
OK
```

2. Deactivate the PDP context on the module.

```
AT+CGACT=0,1
OK
```

4: ECM Network Interface

This chapter describes how to configure the module's USB network interface to use the ECM (Ethernet Control Model) protocol.

4.1 ECM Configuration via Menu

Make sure relevant Linux kernel configurations are set correctly:

1. Make sure the following ECM drivers are defined as shown:

```
CONFIG_USB_USBNET=m
CONFIG_USB_NET_CDCETHER=m
```

2. Make sure the USB cdc_acm driver configuration is defined as shown:

```
CONFIG_USB_ACM=m
```

4.2 ECM USB Network Interface Configuration

Use the !USBCOMP AT command to configure the USB network interface to use ECM.

Table 4-1: !USBCOMP — Set/report USB interface configuration[a]

Format: AT!USBCOMP=<Config Index>,<Config Type>,<Interface bitmask>		
Parameter	**Value**	**Description**
<Config Index>	1	Configuration index to which the composition applies, should be 1
<Config Type>	1	Generic
	2	Network Interface (Note: <Interface bitmask> must include RNDIS or ECM, but not both.)
<Interface bitmask>	00000001	Log DIAG
	00000008	AT (Note: This interface cannot be disabled. (The command returns ERROR if not selected.)
	00000010	AT_PPP
	00001000	RNDIS
	00002000	**ECM**

a. For full command details, refer to *[1] RC71xx AT Command Reference (Doc# 41114675).*

To configure the module to use RNDIS:

1. Set the USB configuration to include ECM (e.g., "2019" enables the Log, AT, AT_PPP and ECM interfaces):

```
AT!USBCOMP=1,2,2019
OK
```

2. Reboot the module to make the configuration change take effect:

```
AT!RESET
```

4.3 Enable the Network Connection

The !NETDEVCTL AT command is used to bind a PDP context to the host to establish a network connection, or to unbind the context to deactivate a connection.

Table 4-2: !NETDEVCTL[a] — Bind/Unbind IP address with host for NIC USB interface

Format: AT!NETDEVCTL=<op>,cid>[<urc_en>]		
Parameter	**Value**	**Description**
<op>	0	Unbind IP address of PDP context <cid> from the host
	1	Bind the IP address of PDP context <cid> to the host one time only. If the context is deactivated and then reactivated, do not automatically rebind.
	2	Bind the IP address of PDP context <cid> to the host. If the context is deactivated and then reactivated, automatically rebind.
	3	Auto-connect and bind <cid> when the module powers on. This setting is persistent.
<cid>	1–15	PDP Context ID
<urc_en>	0	Disable URC (unsolicited response code)
	1	Enable URC (unsolicited response code)

a. For full command details, refer to *[1] RC71xx AT Command Reference (Doc# 41114675).*

To enable automatic network connection for PDP context 1, which automatically establishes a network connection at power on and automatically re-establishes a network connection if the connection drops unexpectedly, make sure to set the <op> parameter to 3:

1. Use <op>=3 to enable automatic network connection:

```
AT!NETDEVCTL=3,1,1
OK
```

4.4 Check ECM Network Connection

To check whether the ECM-based network connection established correctly:

1. List the enumerated USB devices and confirm the cdc_ether driver is present (highlighted **in red** below):

```
#lsusb -t
/:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/3p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/8p, 480M
        |__ Port 3: Dev 3, If 0, Class=Vendor Specific Class,
Driver=i2400m_usb, 480M
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/3p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/6p, 480M
        |__ Port 1: Dev 3, If 0, Class=Human Interface Device, Driver=usbhid,
1.5M
        |   Port 2: Dev 11, If 3, Class=CDC Data, Driver=cdc acm, 480M
        |__ Port 2: Dev 11, If 1, Class=CDC Data, Driver=cdc_ether, 480M
        |__ Port 2: Dev 11, If 6, Class=Communications, Driver=cdc_acm, 480M
```

If cdc_ether did not enumerate, redo 4.1 ECM Configuration via Menu, 4.2 ECM USB Network Interface Configuration, and 4.3 Enable the Network Connection.

2. Find the network adapter interface name for the ECM port in the kernel messages (highlighted **in red** below):

```
#dmesg
[30886.320306] usb 1-1.2: new high-speed USB device number 16 using ehci-pci
[30886.436657] usb 1-1.2: New USB device found, idVendor=1199,
idProduct=9114, bcdDevice= 2.00
[30886.436662] usb 1-1.2: New USB device strings: Mfr=1, Product=2,
SerialNumber=3
[30886.436665] usb 1-1.2: Product: RC7110
[30886.436668] usb 1-1.2: Manufacturer: Sierra Wireless, Incorporated
[30886.436671] usb 1-1.2: SerialNumber: 000000000001
[30886.438887] cdc_ether 1-1.2:1.0 eth0: register 'cdc_ether' at usb-
0000:00:1a.0-1.2, CDC Ethernet Device, 20:89:84:6a:96:aa
[30886.439910] cdc_acm 1-1.2:1.2: ttyACM0: USB ACM device
[30886.444425] cdc_acm 1-1.2:1.4: ttyACM1: USB ACM device
[30886.445314] cdc_acm 1-1.2:1.6: ttyACM2: USB ACM device
[30886.468058] cdc_ether 1-1.2:1.0 enx2089846a96aa: renamed from eth0
```

For the example above, **enx2089846a96aa** is the network adapter interface name for the module's ECM port.

3. The interface's IP address is assigned automatically based on the PDP context's configured PDP type (i.e., IP, IPV6, IPV4V6).

Using the example interface above (**enx2089846a96aa**), test the network connection:

- IPV4 connection expected:

i. Display the interface configurations to confirm the IPv4 ECM connection was established:

```
#ifconfig
enx2089846a96aa: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.81.50.111  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::c67a:1940:b950:cc42  prefixlen 64  scopeid
0x20<link>
        ether 20:89:84:6a:96:aa  txqueuelen 1000  (Ethernet)
        RX packets 30  bytes 3688 (3.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 68  bytes 8242 (8.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

ii. Ping the google domain name server (8.8.8.8) to test IPv4, and ping www.google.com to test IPv6:

```
#ping 8.8.8.8

PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=52 time=142 ms
...
64 bytes from 8.8.8.8: icmp_seq=5 ttl=52 time=39.8 ms

--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 39.898/63.291/142.758/39.846 ms

#ping -6 www.google.com
connect: Network is unreachable
```

This example confirms the ECM connection was established for IPv4 communication only (ping 8.8.8.8 succeeded and ping www.google.com failed).

- IPV6 connection expected:
  i. Display the interface configurations to confirm the IPv6 ECM connection was established:

```
#ifconfig
enx2089846a96aa: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 2401:e180:8880:b3ff:ac6:25d7:cb2c:7c76  prefixlen 64
scopeid 0x0<global>
        inet6 fe80::c67a:1940:b950:cc42  prefixlen 64  scopeid
0x20<link>
        inet6 2401:e180:8880:b3ff:3568:4819:9d89:1d19  prefixlen 64
scopeid 0x0<global>
        ether 20:89:84:6a:96:aa  txqueuelen 1000  (Ethernet)
        RX packets 52  bytes 6206 (6.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 118  bytes 15808 (15.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

  ii. Ping the google domain name server (8.8.8.8) to test IPv4, and ping www.google.com to test IPv6:

```
#ping 8.8.8.8
connect: Network is unreachable

#ping www.google.com
PING www.google.com(tsa01s11-in-x04.1e100.net (2404:6800:4012:2::2004))
56 data bytes
64 bytes from tsa01s11-in-x04.1e100.net (2404:6800:4012:2::2004):
icmp_seq=1 ttl=56 time=32.3 ms
...
64 bytes from tsa01s11-in-x04.1e100.net (2404:6800:4012:2::2004):
icmp_seq=5 ttl=56 time=33.8 ms

--- www.google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 29.918/32.631/36.746/2.505 ms
```

  This example confirms the ECM connection was established for IPv6 communication only (ping 8.8.8.8 failed and ping www.google.com succeeded).

- IPV4V6 connection expected:
  i. Display the interface configurations to confirm the IPv4v6 RNDIS connection was established:

```
#ifconfig
enx2089846a96aa: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.5.57.134  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 2401:e180:8890:e734:3568:4819:9d89:1d19  prefixlen 64
scopeid 0x0<global>
        inet6 2401:e180:8890:e734:1ece:a87a:58b7:560e  prefixlen 64
scopeid 0x0<global>
        inet6 fe80::c67a:1940:b950:cc42  prefixlen 64  scopeid
0x20<link>
        ether 20:89:84:6a:96:aa  txqueuelen 1000  (Ethernet)
        RX packets 88  bytes 11227 (11.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 212  bytes 29548 (29.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

ii. Ping the google domain name server (8.8.8.8) to test IPv4, and ping www.google.com to test IPv6:

```
#ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=54 time=111 ms
...
64 bytes from 8.8.8.8: icmp_seq=6 ttl=54 time=35.7 ms

--- 8.8.8.8 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5005ms
rtt min/avg/max/mdev = 28.839/45.007/111.876/29.980 ms

#ping www.google.com
PING www.google.com(tsa01s11-in-x04.1e100.net (2404:6800:4012:2::2004))
56 data bytes
64 bytes from tsa01s11-in-x04.1e100.net (2404:6800:4012:2::2004):
icmp_seq=1 ttl=54 time=36.6 ms
...
64 bytes from tsa01s11-in-x04.1e100.net (2404:6800:4012:2::2004):
icmp_seq=5 ttl=55 time=36.8 ms
--- www.google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 28.794/33.103/36.803/3.176 ms
```

This example confirms the ECM connection was established for IPv4v6 communication (both ping 8.8.8.8 and ping www.google.com succeeded).

4.5 ECM Network Reconnect

The !NETDEVCTL <op> used in 4.3 Enable the Network Connection determines how the context reconnects to a network after being deactivated (e.g., when network service is lost and then regained).

- If <op> = 3 (as suggested in 4.3 Enable the Network Connection), the context automatically binds and connects when the module powers on, and automatically reconnects after network service is lost and regained.
- If <op> = 0 or 1 was used, the context does not automatically reconnect.
- If <op> = 2 was used, the context automatically reconnects after it was deactivated.

4.6 ECM Disconnect

To disconnect (unbind) the host from a PDP context (e.g., context 1):

1. Use <op>=0 to unbind the host:

```
AT!NETDEVCTL=0,1
OK
```

2. Deactivate the PDP context on the module.

```
AT+CGACT=0,1
OK
```

5: PPP Network Interface

This chapter describes how to configure the module's USB network interface to use PPP (Point-to-Point Protocol).

5.1 PPP Configuration via Menu

Make sure relevant Linux kernel configurations are set correctly:

1. Make sure the following PPP settings are defined as shown (suggested configuration for Ubuntu 18.04 LTS kernel):

```
CONFIG_PPP=y
CONFIG_PPP_BSDCOMP=m
CONFIG_PPP_DEFLATE=m
CONFIG_PPP_FILTER=y
CONFIG_PPP_MPPE=m
CONFIG_PPP_MULTILINK=y
CONFIG_PPPOATM=m
CONFIG_PPPOE=m
CONFIG_PPTP=m
CONFIG_PPPOL2TP=m
CONFIG_PPP_ASYNC=m
CONFIG_PPP_SYNC_TTY=m
```

2. Make sure the USB cdc_acm driver configuration is defined as shown:

```
CONFIG_USB_ACM=m
```

5.2 PPP USB Network Interface Configuration

Use the !USBCOMP AT command to configure the USB network interface to use PPP.

Table 5-1: !USBCOMP—Set/report USB interface configuration[a]

Format: AT!USBCOMP=<Config Index>,<Config Type>,<Interface bitmask>		
Parameter	**Value**	**Description**
<Config Index>	1	Configuration index to which the composition applies, should be 1
<Config Type>	1	Generic
	2	Network Interface (<interface bitmask>)
<Interface bitmask>	00000001	Log DIAG
	00000008	AT (Note: This interface cannot be disabled. (The command returns ERROR if not selected.)
	00000010	**AT_PPP**
	00001000	RNDIS
	00002000	ECM

a. For full command details, refer to *[1] RC71xx AT Command Reference (Doc# 41114675).*

To configure the module to use PPP:

1. Set the USB configuration to include PPP (e.g., "19" enables the Log, AT, and AT_PPP interfaces):

```
AT!USBCOMP=1,1,19
OK
```

Reboot the module to make the configuration change take effect:

```
AT!RESET
```

5.3 Set up and test PPP Network Connection with WvDial (PPP Dialer)

WvDial is a PPP dialer used to dial a modem, start PPP and connect a host product to the Internet.

To set up and test PPP network connections:

Note: In the example used in this procedure, the module has enumerated an AT port (/dev/ttyACM0) and a PPP port (/dev/ttyACM2).

1. Install WvDial on the host product:

```
# sudo apt-get install -y wvdial
```

2. Customize WvDial:
   a. Edit the WvDial configuration file:

```
# sudo vi /etc/wvdial.conf
```

   b. Set the "[Dialer Defaults]" values as appropriate to connect to your Internet Service Provider (ISP). For WvDial configuration details, refer to linux.die.net/man/5/wvdial.conf.

   For the example used in this procedure, the configuration file has been edited as follows:

```
# sudo vi /etc/wvdial.conf

[Dialer Defaults]
Modem = /dev/ttyACM0
Baud = 115200
Init1 = AT+CGDCONT=1, "IPV4V6", "internet"
Phone = *99#
Username = test
Password = test
Dial Command = ATD
Stupid Mode = 1
```

   where:
   - Modem — An enumerated serial tty COM port (i.e., an AT port or PPP port). In this example, the AT port enumerated on /dev/ttyACM0 will be used.
   - Baud — Modem connection speed (related to AT+IPR set value), which matches the module's baud rate. In this example, the connection speed is set to 115200.
   - Init1...Init9 — Initialization strings (i.e., AT commands) to send, in numerical order, to the modem before dialing. In this example, only one initialization string is specified — the command AT+CGDCONT=1,"IPV4V6", "internet" will be sent. No other initialization strings will be sent.
   - Phone — Phone number that wvdial will dial.
   - Username — Username used to access your ISP.
   - Password — Password used to access your ISP.
   - Stupid Mode — Starts pppd immediately after the modem connects, without any prompts.

3. Execute wvdial to establish and test a PPP network connection:

```
# sudo [wvdial path]/wvdial

--> WvDial: Internet dialer version 1.61
--> Initializing modem.
--> Sending: AT+CGDCONT=1, "IPV4V6", "internet"
AT+CGDCONT=1, "IPV4V6", "internet"
OK
--> Modem initialized.
--> Sending: ATD*99#
--> Waiting for carrier.
ATD*99#
CONNECT
--> Carrier detected.  Starting PPP immediately.
--> Starting pppd at Wed Aug 30 11:14:58 2023
--> Pid of pppd: 4410
--> Using interface ppp0
--> local  IP address 10.154.41.228
--> remote IP address 10.64.64.64
--> primary   DNS address 210.241.208.1
--> secondary DNS address 139.175.1.2
```

If a ppd Pid (highlighted **in red** above) is displayed, the network connection was established.

4. Test the network connection:

   a. Display the interface configurations to confirm an IPv4 PPP connection was established:

```
# ifconfig ppp0
ppp0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.154.41.228  netmask 255.255.255.255  destination
10.64.64.64
        ppp  txqueuelen 3  (Point-to-Point Protocol)
        RX packets 7  bytes 250 (250.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 225 (225.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

   b. Ping the Google domain name server (8.8.8.8) to test IPv4:

```
# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) from 10.154.41.228 ppp0: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=116 ms
...
64 bytes from 8.8.8.8: icmp_seq=5 ttl=113 time=27.6 ms

--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 27.627/50.885/116.593/33.323 ms
```

   This example confirms the PPP connection was established for IPv4 communication (the ping 8 .8.8.8 succeeded.

5.3.1 PPP Disconnect

To disconnect the PPP network connection:

1. Stop the PPP process using “kill <pid>)” (e.g., where the pppd PID is shown above in 5.3 Set up and test PPP Network Connection with WvDial (PPP Dialer)):

```
# kill 4410
```

6: CMUX Over UART1

RC71xx modules support CMUX (Connection Multiplexer), as described in 3GPP TS 27.010. CMUX enables the establishment of a multiplexing layer that carries multiple data link connection (DLC) virtual connection channels over a single serial link (physical serial interface).

For RC71xx modules, CMUX is used over the UART1 interface, and the maximum number of DLCs that can be opened is 4, per 3GPP TS 27.010 modes of operation.

All of the channels support AT commands. If a PPP data channel is needed, AT commands can be used to switch one of the channels from AT to PPP.

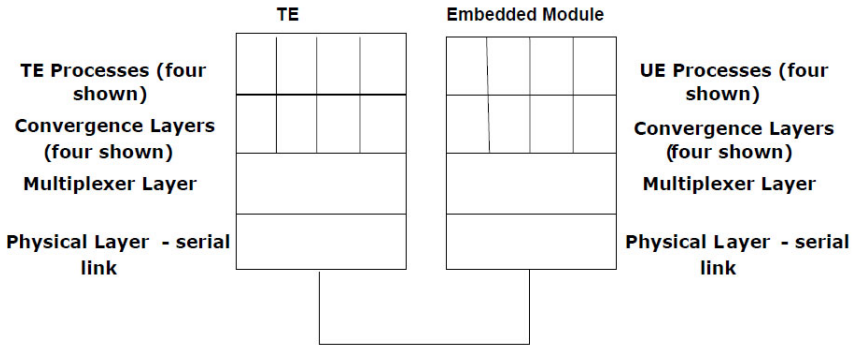


Figure 6-1: CMUX illustration (single physical UART divided into four virtual UARTs)

The examples shown in this chapter are for a Linux build using the kernel module 'n_gsm' to implement CMUX.

For additional details, refer to github.com/Rtone/cmux for cmux open source code, and www.kernel.org/doc/html/v5.9/driver-api/serial/n_gsm.html for n_gsm information.

6.1 Set Up CMUX over UART

To set up the host product to use CMUX over UART:

1. Insert the n_gsm kernel module:

```
# sudo modprobe n_gsm
OK
```

2. Patch the cmux source code:
   a. If you do not have the cmux source code, download it from github.com/Rtone/cmux.
   b. Apply the required changes from the patch file (included in the package) to the cmux source code.

      The following is an example patch file showing the 'diff' results between the cmux source code (a/cmux.c) and the Semtech patch code (b/cmux.c).

> **Important:** *Make sure to use the actual patch file from the package. Do not use the example shown below in Figure 6-2.*

```
diff --git a/cmux.c b/cmux.c
index 1af0f50..c917120 100644
--- a/cmux.c
+++ b/cmux.c
@@ -50,7 +50,7 @@
 #endif

 /* serial port of the modem */
-#define SERIAL_PORT    "/dev/ttyS1"
+#define SERIAL_PORT    "/dev/ttyUSB0"  // The UART serial tty COM device

 /* line speed */
 #define LINE_SPEED B115200
@@ -294,7 +294,7 @@ int main(void) {
   tio.c_iflag = 0;
   tio.c_oflag = 0;
   tio.c_cflag = CS8 | CREAD | CLOCAL;
-  tio.c_cflag |= CRTSCTS;
+  tio.c_cflag  &= ~ CRTSCTS;//related to AT+IFC configure in module setting
   tio.c_lflag = 0;
   tio.c_cc[VMIN] = 1;
   tio.c_cc[VTIME] = 0;
@@ -313,18 +313,12 @@ int main(void) {
   *   to fit your modem needs.
   *   The following matches Quectel M95.
   */
+  // suggest change for AT+IPR, AT+IFC, AT+ICF configure in mod-ule setting.
-  if (send_at_command(serial_fd, "AT+IFC=2,2\r") == -1)
-     errx(EXIT_FAILURE, "AT+IFC=2,2: bad response");
-  if (send_at_command(serial_fd, "AT+GMM\r") == -1)
-     warnx("AT+GMM: bad response");
   if (send_at_command(serial_fd, "AT\r") == -1)
     warnx("AT: bad response");
-  if (send_at_command(serial_fd, "AT+IPR=115200&w\r") == -1)
-     errx(EXIT_FAILURE, "AT+IPR=115200&w: bad response");
-  sprintf(atcommand, "AT+CMUX=0,0,5,%d,10,3,30,10,2\r", MTU);
-  if (send_at_command(serial_fd, atcommand) == -1)
+  // to active +CMUX feature by AT+cMUX=1
+  if (send_at_command(serial_fd, "AT+CMUX=1\r") == -1)
     errx(EXIT_FAILURE, "Cannot enable modem CMUX");

+
   /* use n_gsm line discipline */
   sleep(2);
   if (ioctl(serial_fd, TIOCSETD, &ldisc) < 0)
```

Figure 6-2: Sample cmux patch file

3. Build and execute CMUX:

```
# make
$ sudo./cmux
SERIAL_PORT = /dev/ttyUSB0
AT      : AT  OK
AT+CMUX=: AT+CMUX=1  OK
Line dicipline set
Created /dev/ttyGSM1
Created /dev/ttyGSM2
Created /dev/ttyGSM3
Created /dev/ttyGSM4
Going to background
```

4. Check the ttyGSM node to make sure all of the nodes were enumerated:

```
$ ls /dev/ttyGSM*
/dev/ttyGSM1  /dev/ttyGSM2  /dev/ttyGSM3  /dev/ttyGSM4
```

5. Edit the swi-ppp script and change the port to one of the enumerated ttyGSMx nodes (e.g., /dev/ttyGSM1) to simulate GSM port for PPP connect.

```
...
debug               # (from /etc/ppp/peers/swi-ppp)
nodetach            # (from /etc/ppp/peers/swi-ppp)
dump                # (from /etc/ppp/peers/swi-ppp)
noauth              # (from /etc/ppp/peers/swi-ppp)
user test           # (from /etc/ppp/peers/swi-ppp)
password ??????     # (from /etc/ppp/peers/swi-ppp)
remotename 3gppp    # (from /etc/ppp/peers/swi-ppp)
/dev/ttyGSM1        # (from /etc/ppp/peers/swi-ppp)
115200              # (from /etc/ppp/peers/swi-ppp)
...
```

6. Execute swi-ppp. (Note: For assistance with the swi-ppp dialer script, contact your Sierra Wireless technical support representative.)

```
/etc/ppp/peers$ sudo pppd call swi-ppp
pppd options in effect:
debug               # (from /etc/ppp/peers/swi-ppp)
nodetach            # (from /etc/ppp/peers/swi-ppp)
dump                # (from /etc/ppp/peers/swi-ppp)
noauth              # (from /etc/ppp/peers/swi-ppp)
user test           # (from /etc/ppp/peers/swi-ppp)
password ??????     # (from /etc/ppp/peers/swi-ppp)
remotename 3gppp    # (from /etc/ppp/peers/swi-ppp)
/dev/ttyGSM1        # (from /etc/ppp/peers/swi-ppp)
115200              # (from /etc/ppp/peers/swi-ppp)
lock                # (from /etc/ppp/peers/swi-ppp)
connect chat -s -v -f /etc/ppp/peers/swi-chat-connect # (from /etc/ppp/
peers/swi-ppp)
disconnect chat -s -v -f /etc/ppp/peers/swi-chat-disconnect # (from /etc/
ppp/peers/swi-ppp)
nocrtscts           # (from /etc/ppp/peers/swi-ppp)
modem               # (from /etc/ppp/peers/swi-ppp)
asyncmap 0          # (from /etc/ppp/options)
lcp-echo-failure 4 # (from /etc/ppp/options)
```

```
lcp-echo-interval 30 # (from /etc/ppp/options)
hide-password      # (from /etc/ppp/peers/swi-ppp)
novj               # (from /etc/ppp/peers/swi-ppp)
novjccomp          # (from /etc/ppp/peers/swi-ppp)
ipcp-accept-local  # (from /etc/ppp/peers/swi-ppp)
ipcp-accept-remote # (from /etc/ppp/peers/swi-ppp)
ipparam 3gppp      # (from /etc/ppp/peers/swi-ppp)
noipdefault        # (from /etc/ppp/peers/swi-ppp)
ipcp-max-failure 30# (from /etc/ppp/peers/swi-ppp)
defaultroute       # (from /etc/ppp/peers/swi-ppp)
usepeerdns         # (from /etc/ppp/peers/swi-ppp)
noccp              # (from /etc/ppp/peers/swi-ppp)
noipx              # (from /etc/ppp/options)
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)
^M
QCRDY^M
AT^M
OK
 -- got it

send (ATE0^M)
expect (OK)
^M
ATE0^M
OK
 -- got it

send (ATI;+CSQ;+CPIN?;+COPS?;+CEREG?;&D2^M)
expect (OK)
^M
^M
Manufacturer: Sierra Wireless ^M
Model: RC7120^M
Revision: SWI216C_01.02.05.03 2023/11/10 17:31:57^M
SVN: 01^M
IMEI: 359112220001572^M
FSN: EU3314011101GB^M
TS.25: 2023.10.23^M
+GCAP: +CGSM^M
^M
+CSQ: 13,0^M
^M
+CPIN: READY^M
^M
+COPS: 0,2,"46601",7^M
```

```
^M
+CEREG: 0,1^M
^M
OK
 -- got it

send (AT+CGDCONT=1,"IPV4V6","internet",,0,0^M)
expect (OK)
^M
^M
OK
 -- got it

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

Script chat -s -v -f /etc/ppp/peers/swi-chat-connect finished (pid 3512),
status = 0x0
Serial connection established.
using channel 1
Using interface ppp0
Connect: ppp0 <--> /dev/ttyGSM1
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0xde11a684> <pcomp>
<accomp>]
rcvd [LCP ConfReq id=0x1 <asyncmap 0x0> <auth chap MD5> <magic 0x594f00c3>
<pcomp> <accomp>]
sent [LCP ConfAck id=0x1 <asyncmap 0x0> <auth chap MD5> <magic 0x594f00c3>
<pcomp> <accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0xde11a684> <pcomp>
<accomp>]
sent [LCP EchoReq id=0x0 magic=0xde11a684]
rcvd [CHAP Challenge id=0xa9 <38cb40864dfaaa3160a11819f5409abe1e0d>, name =
"lwpppos"]
sent [CHAP Response id=0xa9 <eeea3c7af0087fafd3203e8bcd501951>, name =
"test"]
rcvd [LCP EchoRep id=0x0 magic=0x594f00c3]
rcvd [CHAP Success id=0xa9 "Login ok"]
CHAP authentication succeeded: Login ok
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>]
rcvd [IPCP ConfReq id=0x1]
sent [IPCP ConfNak id=0x1 <addr 0.0.0.0>]
rcvd [IPCP ConfNak id=0x1 <addr 10.43.77.190> <ms-dns1 210.241.208.1> <ms-
dns2 139.175.1.2>]
sent [IPCP ConfReq id=0x2 <addr 10.43.77.190> <ms-dns1 210.241.208.1> <ms-
dns2 139.175.1.2>]
rcvd [IPCP ConfReq id=0x2]
```

```
sent [IPCP ConfAck id=0x2]
rcvd [IPCP ConfAck id=0x2 <addr 10.43.77.190> <ms-dns1 210.241.208.1> <ms-
dns2 139.175.1.2>]
Could not determine remote IP address: defaulting to 10.64.64.64
local  IP address 10.43.77.190
remote IP address 10.64.64.64
primary   DNS address 210.241.208.1
secondary DNS address 139.175.1.2
Script /etc/ppp/ip-up started (pid 3523)
Script /etc/ppp/ip-up finished (pid 3523), status = 0x0
```

7. Display the interface configurations to confirm the IPv4 PPP connection was established:

```
/etc/ppp/peers$ ifconfig

ppp0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.43.77.190  netmask 255.255.255.255  destination 10.64.64.64
        ppp  txqueuelen 3  (Point-to-Point Protocol)
        RX packets 53  bytes 28268 (28.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 56  bytes 7083 (7.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

$ ping -I ppp0 8.8.8.8
PING 8.8.8.8 (8.8.8.8) from 10.43.77.190 ppp0: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=51 time=240 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=51 time=57.3 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=51 time=59.9 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=51 time=79.6 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=51 time=81.1 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=51 time=78.9 ms
^C
--- 8.8.8.8 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5008ms
rtt min/avg/max/mdev = 57.304/99.582/240.513/63.748 ms
```

8. Ping the Google domain name server 8.8.8.8) to test IPv4:

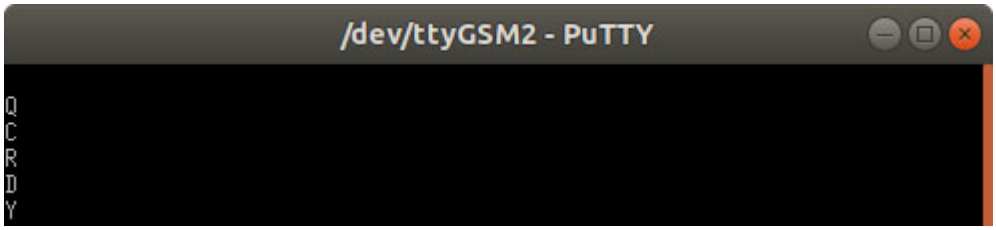
```
$ ping -I ppp0 8.8.8.8
PING 8.8.8.8 (8.8.8.8) from 10.43.77.190 ppp0: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=51 time=240 ms
...
64 bytes from 8.8.8.8: icmp_seq=6 ttl=51 time=78.9 ms
^C
--- 8.8.8.8 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5008ms
rtt min/avg/max/mdev = 57.304/99.582/240.513/63.748 ms
```

This example confirms the data connection was established for IPv4 communication (the ping 8.8.8.8 succeeded).

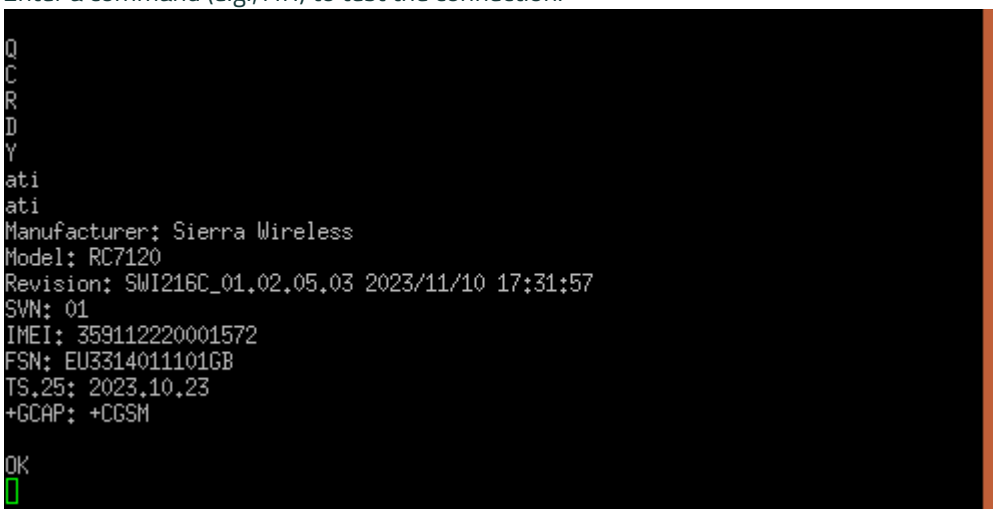
Using Other DLCs

To demonstrate that the module can use different DLCs (channels) over the shared serial link, set up a second DLC to use for AT commands:

1. Use a terminal emulator to open one of the unused DLCs (e.g., /dev/ttyGSM2).



2. Enter a command (e.g., ATI) to test the connection:



7: References

7.1 Sierra Wireless Documents

Sierra Wireless documents are available from source.sierrawireless.com, or on request (subject to license agreements or NDAs) from your Sierra Wireless representative.

Sierra Wireless Documents on the Source

The following documents are available from source.sierrawireless.com:

[1] RC71xx AT Command Reference (Doc# 41114675)

7.2 Industry / Other Documents

The following referenced document are not provided by Sierra Wireless:

[2] AT Command Set for User Equipment (UE) (Release 6) (Doc# 3GPP TS 27.007)