SIM767XX Series_ CMUX_USER_GUIDE

LTE Module

SIMCom Wireless Solutions Limited
SIMCom Headquarters Building, Building 3, No. 289 Linhong Road, Changning District, Shanghai P.R. China
Tel: 86-21-31575100
support@simcom.com
www.simcom.com

Document Title:	SIM767XX Series_CMUX_USER_GUIDE
Version:	1.00
Date:	2024.01.30
Status:	Released

GENERAL NOTES

SIMCOM OFFERS THIS INFORMATION AS A SERVICE TO ITS CUSTOMERS, TO SUPPORT APPLICATION AND ENGINEERING EFFORTS THAT USE THE PRODUCTS DESIGNED BY SIMCOM. THE INFORMATION PROVIDED IS BASED UPON REQUIREMENTS SPECIFICALLY PROVIDED TO SIMCOM BY THE CUSTOMERS. SIMCOM HAS NOT UNDERTAKEN ANY INDEPENDENT SEARCH FOR ADDITIONAL RELEVANT INFORMATION, INCLUDING ANY INFORMATION THAT MAY BE IN THE CUSTOMER'S POSSESSION. FURTHERMORE, SYSTEM VALIDATION OF THIS PRODUCT DESIGNED BY SIMCOM WITHIN A LARGER ELECTRONIC SYSTEM REMAINS THE RESPONSIBILITY OF THE CUSTOMER OR THE CUSTOMER'S SYSTEM INTEGRATOR. ALL SPECIFICATIONS SUPPLIED HEREIN ARE SUBJECT TO CHANGE.

SIMCom Wireless Solutions Limited
SIMCom Headquarters Building, Building 3, No. 289 Linhong Road, Changning District, Shanghai P.R. China
Tel: +86 21 31575100
Email: simcom@simcom.com

For more information, please visit:
https://www.simcom.com/technical_files.html

For technical support, or to report documentation errors, please visit:
https://www.simcom.com/online_questions.html or email to: support@simcom.com

About Document

Version History

Revision	Date	Owner	Description
V1.00	2024.01.30		New version

Scope

Based on module AT command manual, this document will introduce CMUX.
This document applies to SIM767XX Series.

Contents

1 Introduction

1.1 Purpose of the document

Based on module AT command manual, this document will introduce CMUX.

1.2 Related documents

[1] SIM767XX Series_AT Command Manual

1 PC

1.1 Software Installation

1.1.1 CMUX

If you want to virtualize the serial port, you need to use the following two software, among which cmuxapp does not need to be installed, you can decompress and use it.

cmuxapp-201803011435-f0d7c110.7z	2019/2/21 14:10	WinRAR 压缩文件
com0com-3.0.0.0-i386-and-x64-signed.zip	2019/2/21 14:06	WinRAR ZIP 压缩文件

1.1.2 com0com install

After extracting these two packages, the first thing we need to do is install com0com，Choose your own operating system to install (just click Next all the time):

Setup_com0com_v3.0.0.0_W7_x64_signed.exe	2017/7/13 14:26	应用程序
Setup_com0com_v3.0.0.0_W7_x86_signed.exe	2017/7/13 14:25	应用程序

Figure 1.1 com0com install exe

After the installation, we find this com0com folder at the bottom left of the system, that is, "Start" and so on, and run Setup.exe.

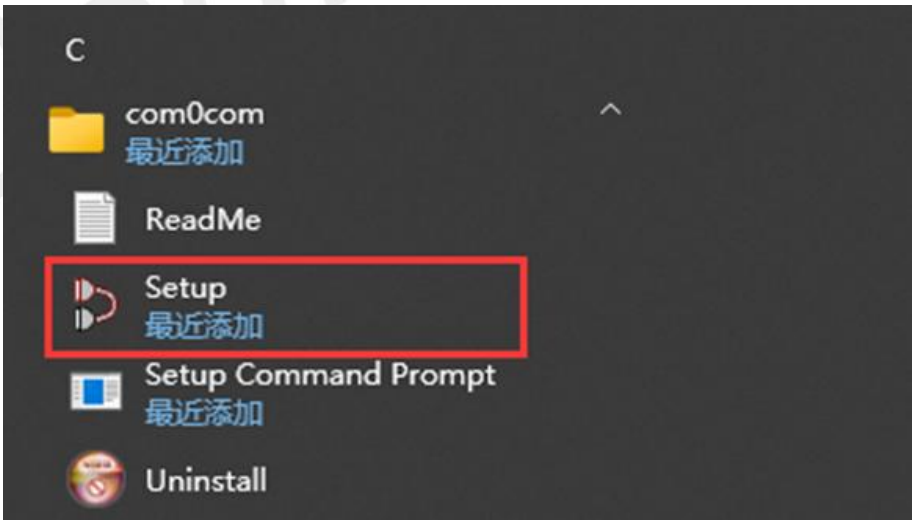


Figure 1.2 Start from the start bar com0com

You can also find the program under the installation path, the default installation path: C:\Program Files (x86)\com0com

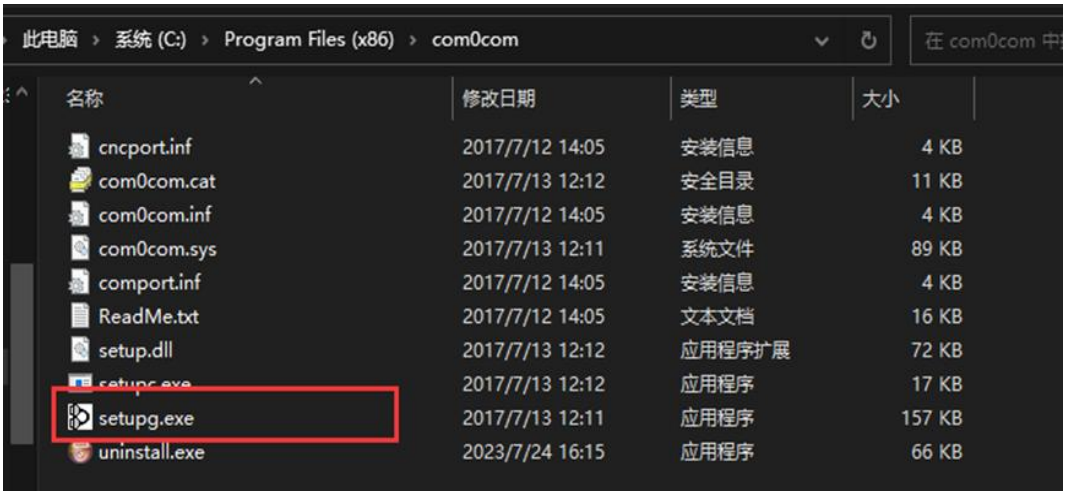


Figure 1.3 Launch from the installation path com0com

When you open setup.exe, you may encounter the following prompts, at this time, you can install the corresponding version of .NetFramework according to the prompts. Once the installation is complete, restart your computer to use com0com.

Figure 1.4 You are prompted to install .NET Framework

1.2 Use com0com Create a virtual serial port

- First, add a pair of COMs by clicking Add Pair in the lower left corner of the software (the name of the COM needs to not conflict with the actual COM name).
- After the addition is complete, click Apply in the lower right corner.

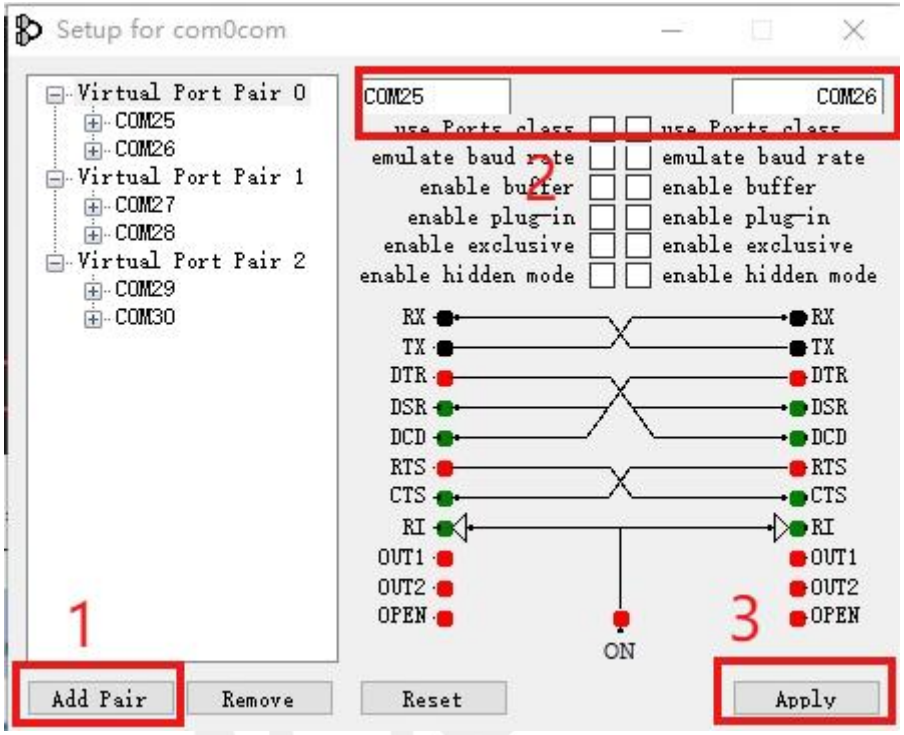


Figure 1.5 Create a virtual serial port

- After adding the port, you can open Device Manager to check whether the COM port exists
- If you do not see the virtual serial port in the device management, you need to disable the driver forced signing (the steps are: Settings - > recovery - > restart immediately under advanced boot - > in the advanced options on the selection page - > startup settings - > F7 (disable driver forced signing), and you can restart it)

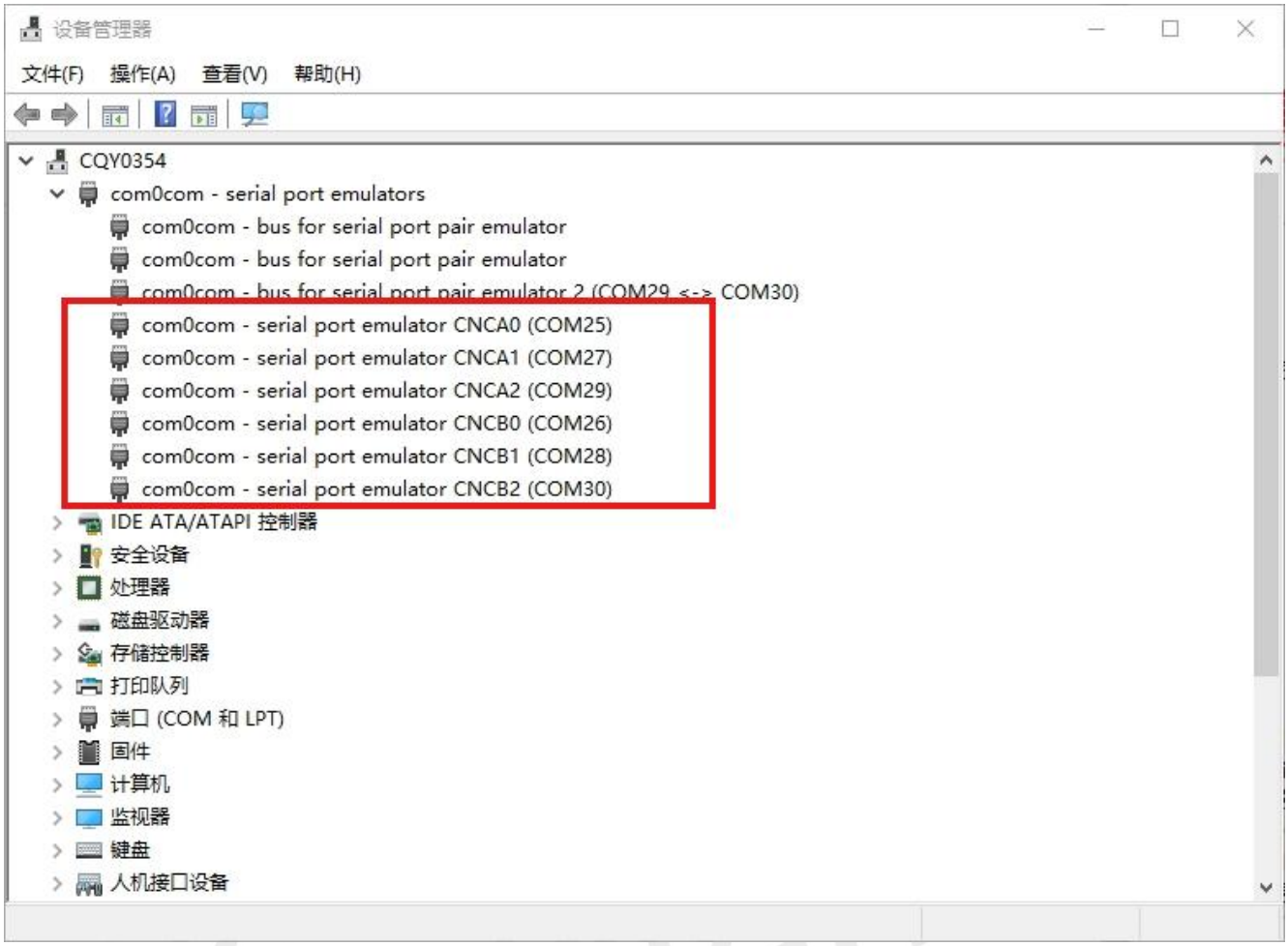


Figure 1.6 Observe the virtual serial port from the Device Manager

1.3 cmuxapp description

- Extract the compressed package and click cmuxapp.exe to run

名称	修改日期	类型	大小
doc	2023/6/1 4:35	文件夹	
imageformats	2023/6/1 4:35	文件夹	
platforms	2023/6/1 4:35	文件夹	
cmuxapp.exe	2018/3/1 13:50	应用程序	85 KB
Qt5Core.dll	2017/12/8 22:41	应用程序扩展	4,533 KB
Qt5Gui.dll	2017/12/8 22:42	应用程序扩展	4,902 KB
Qt5SerialPort.dll	2017/12/8 22:44	应用程序扩展	61 KB
Qt5Widgets.dll	2017/12/8 22:43	应用程序扩展	4,330 KB

Figure 1.7 run cmuxapp

- **Physical COM:** It is the actual COM port, and you can select an Enhanced port here.
- **Virtual COM pair:** Virtual port selection. When using the serial port debugging assistant to open the

virtual serial port later, remember to select the right one in a pair of virtual ports, for example, the first pair of virtual ports COM25 <-> COM26, you should use the serial port assistant to open COM26 for testing, COM25 cannot be opened with the serial port assistant. At present, each serial port can only have a maximum of three virtual ports.

- **CMUX：Check the Send AT+CMUX option, generally set the frame to 1500).**
- **To exit, just click STOP.**

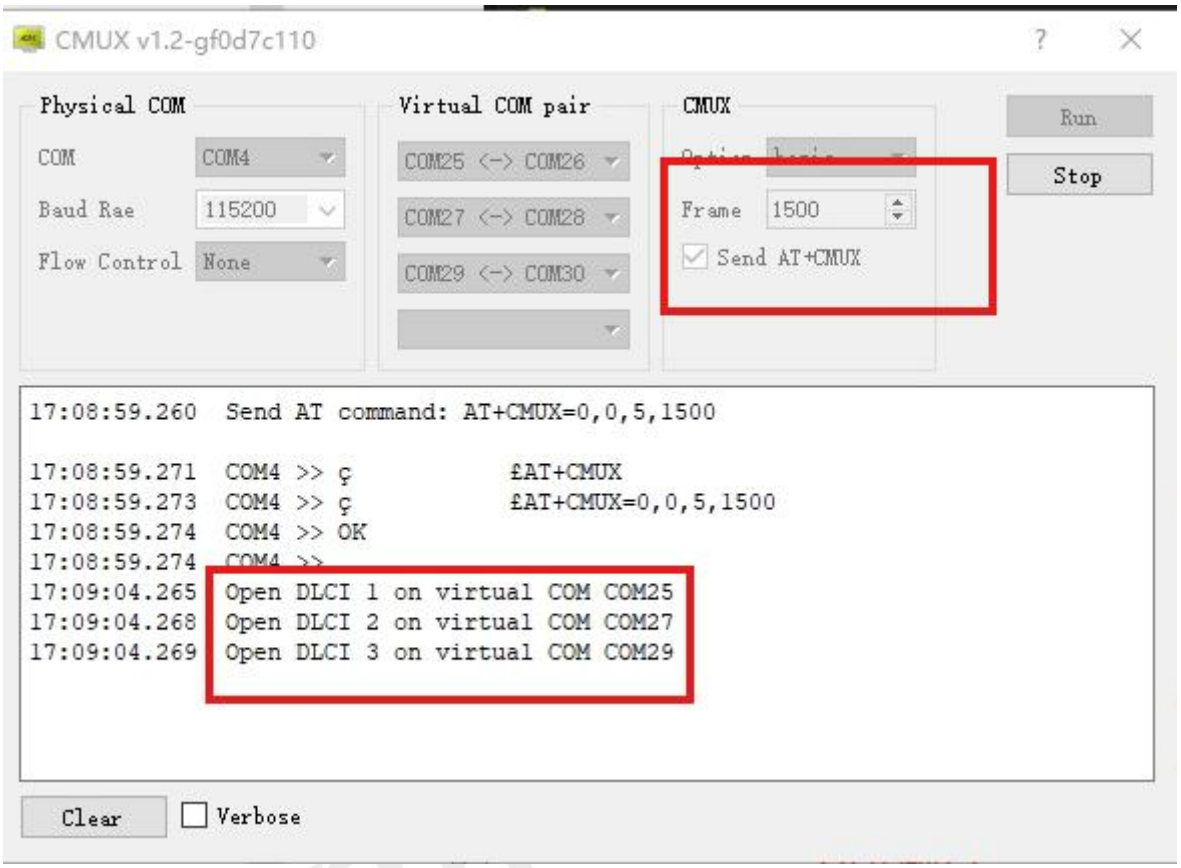


Figure 1.8 cmuxapp illustrate

1.4 Test CMUX

Open CMUXAPP， Set as follows and click Run to run (**remember to close the serial port assistant before Run, otherwise it will be occupied**)

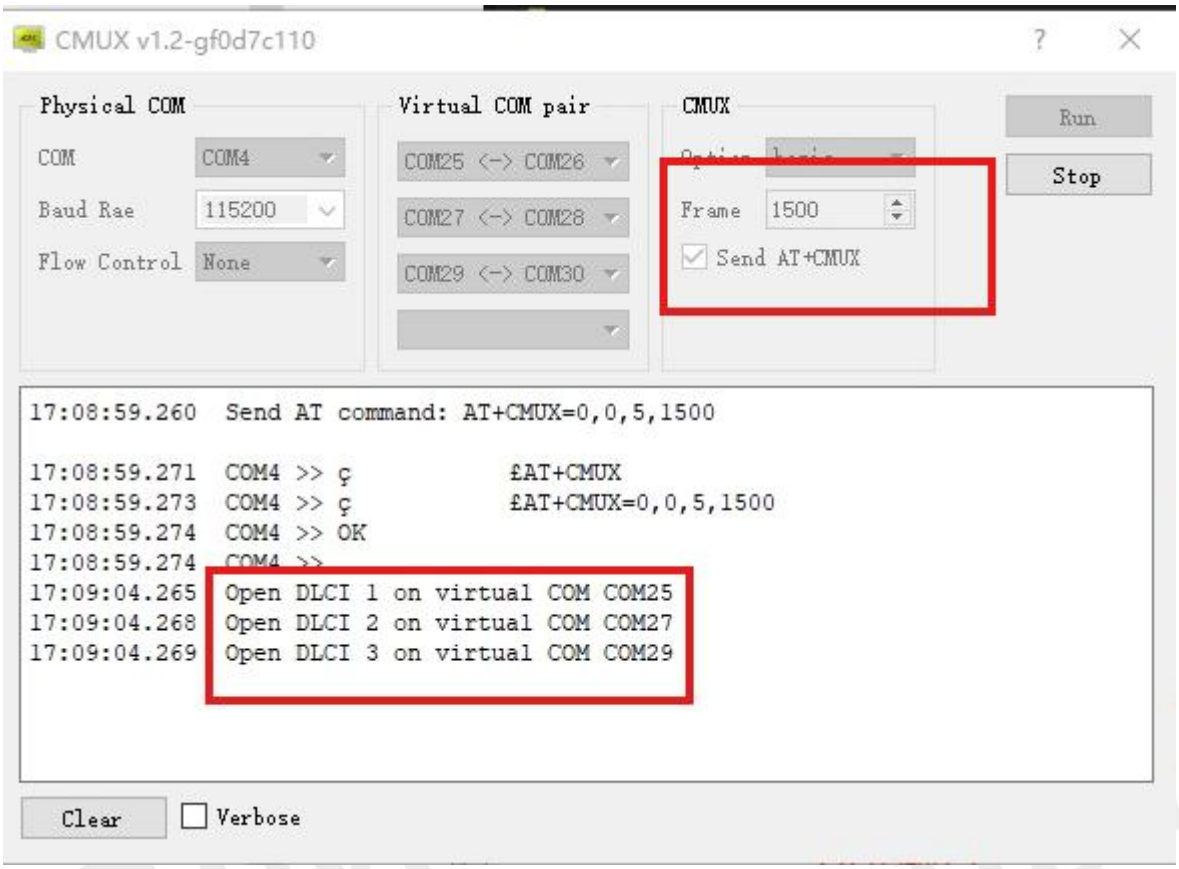


Figure 1.9 Generate a virtual serial port

Then use the serial port assistant to select the corresponding serial port to send and receive data normally.

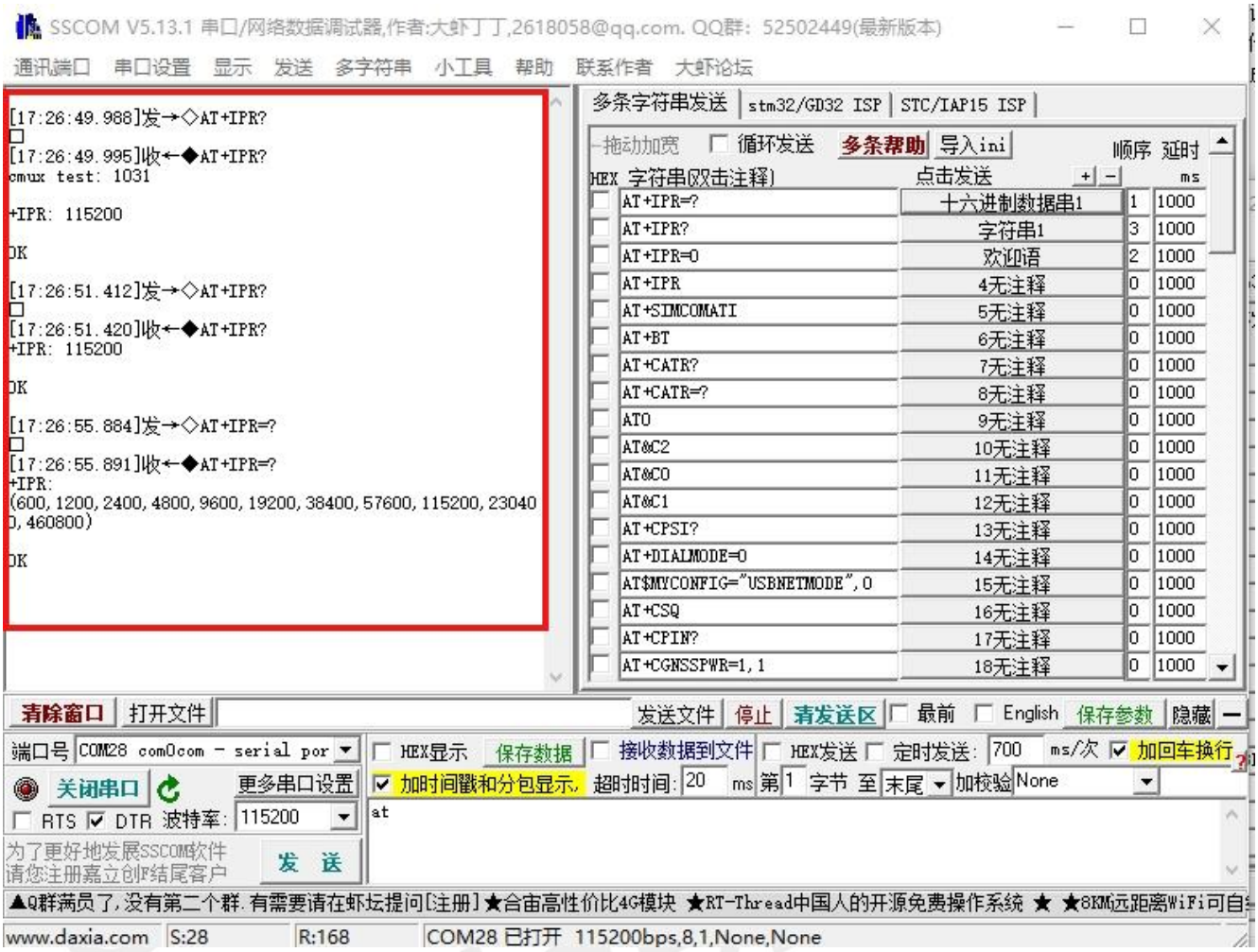


Figure 1.10 AT command

1.5 Problems that may be encountered during use

Why can't i run cmuxapp?

- Please check whether the serial port tool SSMCOM has opened the UART port, please close it first.

Why am I unable to send data and fail when I have succeeded in the past?

- The virtual ports are based on the right, for example, if you have a virtual pair of COM, please select COM23 to send and receive AT commands.

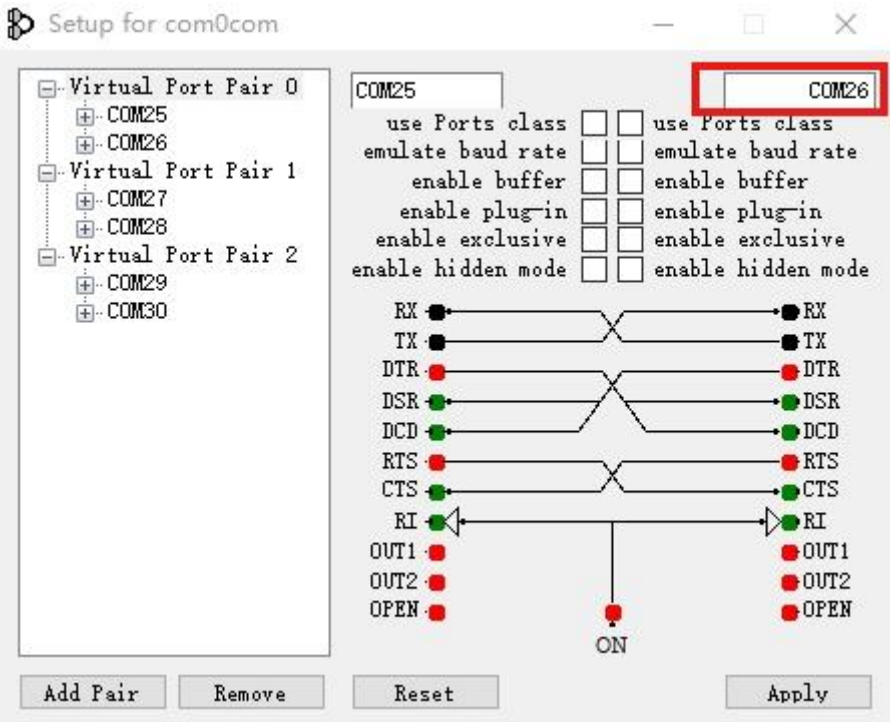


Figure 1.11 sscom test cmux, open the right port (COM2)

2 Linux

2.1 CMUX description

2.1.1 CMUX framework

The multiplexing protocol provides the ability to virtualize multiple parallel logical communication channels on top of a single physical communication channel, which is generally applied between TE (Terminal Equipment) and MS (Mobile Station), TE is equivalent to the AP end of the smartphone, and MS is equivalent to the MODEM end of the smartphone：

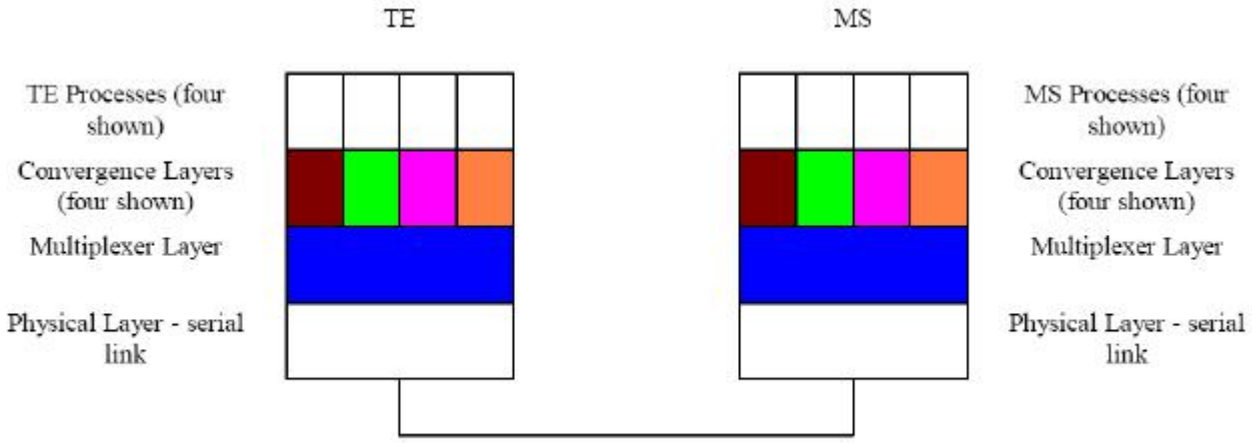


Figure 1: Protocol Stacks

The blue part is the MUX multiplexing layer, which uses the underlying physical serial port to link to send and receive data, and at the same time provides several logically independent transceiver channels for the upper layer (four logical channels are provided in the figure above, which are represented by different colors). Each logical channel is created independently and can have software flow control. In actual use, the MUX on the TE side initiates a channel establishment request to the MUX on the MS side, sets channel parameters, etc., and is the active party. The MUX on the MS side waits for the service request of the TE side and provides the corresponding service according to its own capabilities. That is, the roles of the two are asymmetrical.

2.2 TCP CMUX driver

Simcom provides the source code for CMUX that can be compiled under linux, which can be obtained from Simcom's R&D or technical support colleagues.The source directory name is **gsmMuxLinux**

2.2.1 Driver compilation

Copy the above driver directory to any directory on the Linux system, and use the makefile in the directory to compile (if the compilation fails, you can check the detailed reason for the compilation failure in the **/var/log/syslog file**):

```
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$ sudo make clean
rm -f gsm0710.o buffer.o gsmMuxd
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$ sudo make
gcc -Wall -DDEBUG -c -o gsm0710.o gsm0710.c
gcc -Wall -DDEBUG -c -o buffer.o buffer.c
gcc -lm -o gsmMuxd gsm0710.o buffer.o
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$ ls
buffer.c  buffer.o   gsm0710.h  gsmMuxd   mux.d    readme       run.sh
buffer.h  gsm0710.c  gsm0710.o  Makefile  ppp_fc9  README.orig  vssver.scc
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
```

compile

driver of cmux

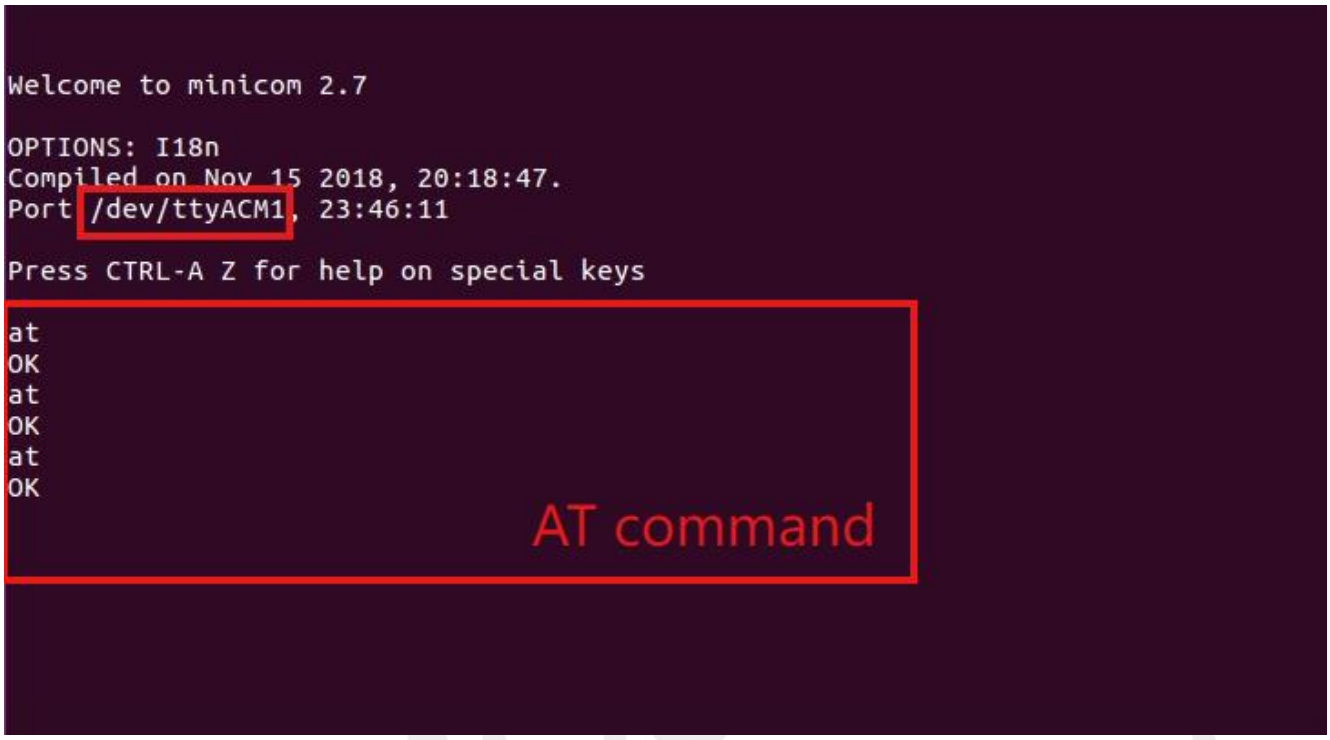
make is for compilation, and the **gsmMuxd** is the Executable program.

2.2.2 Physical serial port

For example, when I send AT on my virtual machine, I can send it through **/dev/ttyACM1**:

execute：**sudo minicom -D /dev/ttyACM1/dev/**

The instructions are as follows:



/dev/ttyACM1 node is our actual physical serial port node.

2.3 Test CMUX

2.3.1 Virtual serial port node

Execute this command:
sudo ./gsmMuxd -p /dev/ttyACM1 -b 115200 -s /dev/mux -w /dev/ptmx /dev/ptmx /dev/ptmx

The following /dev/ptmx is a fixed value, which is a pseudo-port used for mapping, and requires several serial port nodes of the virtual machine to have several /dev/ptmx.

Parameter description:
When using gsmmuxd, the main parameters are described as follows:

- **-p: Actual physical serial device nodes.**
- **-b: The baud rate of communication with the module is generally 115200.**
- **-s: the prefix from the device symbol (e.g. /dev/mux).**
- **-w : Wait for deamon to start successfully/failing.**

Enable CMUX mode and virtualize two virtual serial ports :

```
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$ sudo make
gcc -Wall -DDEBUG -c -o gsm0710.o gsm0710.c
gcc -Wall -DDEBUG -c -o buffer.o buffer.c
gcc -lm -o gsmMuxd gsm0710.o buffer.o
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$ ls
buffer.c  buffer.o   gsm0710.h  gsmMuxd   mux.d    readme       run.sh
buffer.h  gsm0710.c  gsm0710.o  Makefile  ppp_fc9  README.orig  vssver.scc
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$ sudo gsmMuxd -p /dev/ttyACM1 -b
115200 -s /dev/mux -w /dev/ptmx /dev/ptmx /dev/ptmx
MUX started
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
fushilin@ubuntu:~/workstation/cmux/gsmMuxLinux$
```

will generate 3 nodes,"MUX started" is success

Use **ls /dev** to check that there are three new device nodes in the device directory, mux0, mux1 and mux2, which are virtual device nodes.

```
fushilin@ubuntu:~$
fushilin@ubuntu:~$ ls /dev/
agpgart          loop4            snapshot  tty34  tty9       ttyS5
autofs           loop5            snd       tty35  ttyACM0    ttyS6
block            loop6            sr0       tty36  ttyACM1    ttyS7
bsg              loop7            stderr    tty37  ttyACM2    ttyS8
btrfs-control    loop-control     stdin     tty38  ttyACM3    ttyS9
bus              mapper           stdout    tty39  ttyACM4    uhid
cdrom            mcelog           tty       tty4   ttyACM5    uinput
cdrw             mem              tty0      tty40  ttyprintk  urandom
char             memory_bandwidth tty1      tty41  ttyS0      userio
console          midi             tty10     tty42  ttyS1      vcs
core             mqueue           tty11     tty43  ttyS10     vcs1
cpu_dma_latency  mux0             tty12     tty44  ttyS11     vcs2
cuse             mux1             tty13     tty45  ttyS12     vcs3
disk             mux2             tty14     tty46  ttyS13     vcs4
dmmidi           net              tty15     tty47  ttyS14     vcs5
dri              network_latency  tty16     tty48  ttyS15     vcs6
dvd              network_throughput tty17   tty49  ttyS16     vcs7
ecryptfs         null             tty18     tty5   ttyS17     vcsa
fb0              port             tty19     tty50  ttyS18     vcsa1
fd               ppp              tty2      tty51  ttyS19     vcsa2
full             psaux            tty20     tty52  ttyS2      vcsa3
fuse             ptmx             tty21     tty53  ttyS20     vcsa4
```

Note that if the above run fails, it is not MUX started in the image above. Then you can check whether the use of **AT+CMUX** is correct in the implementation of the initGeneric function in the **gsm0710.c** source file:

```
int initGeneric()
{
        //char mux_command[40] = "AT+CMUX=0,0,5,255,10,3,30,10,2\r\n";
        char mux_command[40] = "AT+CMUX=0,0,5,1500\r\n";
    unsigned char close_mux[2] = { C_CLD | CR, 1 };
    /*
    int baud = indexOfBaud(baudrate);
```

2.3.2 AT Command test virtual serial port

Just like using the physical serial port node, use minicom to open MUX0 for command testing.

Example this:

sudo minicom -D /dev/mux0

```
Welcome to minicom 2.7

OPTIONS: I18n
Compiled on Nov 15 2018, 20:18:47.
Port /dev/mux0  23:54:24

Press CTRL-A Z for help on special keys

at
OK
at+ipr=?
+IPR: (600,1200,2400,4800,9600,19200,38400,57600,115200,230400,460800)

OK
at+ipr?
+IPR: 115200

OK
```

For mux1 and mux2, it is the same, both can be turned on to send AT commands.

In this case, the physical serial port cannot be used, because this serial port has become a virtual serial port. If you are not using cmux, then you need to run **sudo killall gsmMuxd** to kill the virtual serial port process, and the corresponding device node will be automatically deleted, and the physical serial port can be used normally.

2.4 Test Virtual serial port

1. Use the node mux0 for PPP dial-up.

You need to change the device node of simcom-pppd to the device node that was virtualized earlier, for example, use **mux1**:

```
#named simcom-pppd and place in /etc/ppp/peers
#/dev/ttyUSB2 115200
/dev/mux1 115200
#Insert the username and password for authentication,default user and password a
re test
user "test" password "test"
#The chat script,customize your APN in this file
connect 'chat -s -v -f /etc/ppp/peers/simcom-connect-chat'
#The close script
disconnect 'chat -s -v -f/etc/ppp/peers/simcom-disconnect-chat'
#Hide password in debug messages
hide-password
#The phone is not required to authenticate
noauth
#Debug info from pppd
debug
#if you want to use the HSDPA link as your geteway
defaultroute
#pppd must not propose any IP address to the peer
noipdefault
#No ppp compression
novj
novjccomp
"simcom-pppd" 37L, 935C                                    1,1           Top
```

which virtual serial port to use

Run the sudo pppd call simcom-pppd command to dial the number (if it fails, you can check the reason in the /var/log/syslog file):

```
fushilin@ubuntu:/etc/ppp/peers$ sudo pppd call simcom-pppd
[sudo] password for fushilin:
pppd options in effect:
debug           # (from /etc/ppp/peers/simcom-pppd)
nodetach                # (from /etc/ppp/peers/simcom-pppd)
dump            # (from /etc/ppp/peers/simcom-pppd)
noauth          # (from /etc/ppp/peers/simcom-pppd)
user test               # (from /etc/ppp/peers/simcom-pppd)
password ??????         # (from /etc/ppp/peers/simcom-pppd)
remotename 3gppp                # (from /etc/ppp/peers/simcom-pppd)
/dev/mux0               # (from /etc/ppp/peers/simcom-pppd)
115200          # (from /etc/ppp/peers/simcom-pppd)
lock            # (from /etc/ppp/peers/simcom-pppd)
connect chat -s -v -f /etc/ppp/peers/simcom-connect-chat                # (from /etc/ppp/peers/s
disconnect chat -s -v -f/etc/ppp/peers/simcom-disconnect-chat           # (from /etc/ppp/peers/s
```

```
rcvd [LCP ProtRej id=0x2 80 fd 01 01 00 0f 1a 04 78 00 18 04 78 00 15 03 2f]
Protocol-Reject for 'Compression Control Protocol' (0x80fd) received
rcvd [IPCP ConfReq id=0x1]
sent [IPCP ConfNak id=0x1 <addr 0.0.0.0>]
rcvd [IPCP ConfNak id=0x1 <addr 10.63.222.167> <ms-dns1 183.230.126.224> <ms-dns
2 183.230.126.225>]
sent [IPCP ConfReq id=0x2 <addr 10.63.222.167> <ms-dns1 183.230.126.224> <ms-dns
2 183.230.126.225>]
rcvd [IPCP ConfReq id=0x2]
sent [IPCP ConfAck id=0x2]
rcvd [IPCP ConfAck id=0x2 <addr 10.63.222.167> <ms-dns1 183.230.126.224> <ms-dns
2 183.230.126.225>]
Could not determine remote IP address: defaulting to 10.64.64.64
local  IP address 10.63.222.167
remote IP address 10.64.64.64
primary   DNS address 183.230.126.224
secondary DNS address 183.230.126.225
Script /etc/ppp/ip-up started (pid 2246)
Script /etc/ppp/ip-up finished (pid 2246), status = 0x0
```

status 0 is success

```
fushilin@ubuntu:/etc/ppp/peers$ ifconfig
ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.121.99.208  P-t-P:10.64.64.64  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:1048 (1.0 KB)  TX bytes:794 (794.0 B)

fushilin@ubuntu:/etc/ppp/peers$
```

After the PPP dial-up is successful, you will see the virtual PPP0 NIC in **ifconfig**:

```
fushilin@ubuntu:~$ ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1227 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1227 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:122803 (122.8 KB)  TX bytes:122803 (122.8 KB)

ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.63.222.167  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:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:1340 (1.3 KB)  TX bytes:854 (854.0 B)

fushilin@ubuntu:~$
fushilin@ubuntu:~$
fushilin@ubuntu:~$
fushilin@ubuntu:~$ cd /etc/ppp/peers/
fushilin@ubuntu:/etc/ppp/peers$ vi simcom-pppd
fushilin@ubuntu:/etc/ppp/peers$
```

2. Use another mux0 port for the at command function, use sudo minicom -D /dev/mux0 to open and

3. Use mux2 to connect to TCP and transparently transmit data. You need to use at+netclose to return OK successfully, and then set AT+CIPmode=1 (set to 1 to pass through data, and set to 0 to indicate fixed-length sending):

```
Compiled on Nov 15 2018, 20:18:47.
Port /dev/mux2, 00:01:46

Press CTRL-A Z for help on special keys

at
OK
at+netclose
+NETCLOSE: 2

ERROR
at+netopen
OK
at+netclose
OK
at+cipmode=1
OK
at+cipopen=0,"TCP","183.230.174.137",6044
+CIPOPEN: 0,2

ERROR
at+netopen
OK
at+cipopen=0,"TCP","183.230.174.137",6044
CONNECT 115200
```

set the mode

tcp connect