

Android 系统的通信功能 调试指导文档

Version: V1.0.3

版权声明

版权所有©深圳市广和通无线股份有限公司 2015。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标申明



为深圳市广和通无线股份有限公司的注册商标，由所有人拥有。

注意

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

版本记录

Version	Date	Remarks
V1.0.0	2013-07-23	Initial Release
V1.0.1	2014-05-21	Update to last guide
V1.0.2	2014-10-10	Added to support 4G(L810) Modules, See Chapter 2.3 Chapter 3 and Chapter 4 Deleted 2.1 Chapter
V1.0.3	2015-02-05	增加 3G 内核驱动配置以及 rild.c 的修改指引

目录

1 基本介绍	4
2 调试过程	5
2.1 系统与模块通讯端口的调试	5
2.2 添加调试 RIL	6
2.3 调试信号的显示、电话程序的功能、短信功能等	8
2.4 添加 MUX	12
2.5 调试 ppp 拨号上网功能	14
2.6 调试音频通道切换功能	17
2.7 调试音量调节功能	17
2.8 添加、定制其它功能	17
3 添加 3G 所需要内核驱动配置	18
3.1 修改内核编译配置(kernel 根目录下的.config 文件中), 确保下面的配置项已经被选定:	18
3.2 详细操作	18
4 添加 4G 所需要内核驱动配置	23
4.1 修改内核编译配置(kernel 根目录下的.config 文件中), 确保下面的配置项已经被选定:	23
4.2 详细操作	23
5 如何确认 NCM 以及 ACM 驱动已经配置入系统	26

1 基本介绍

在本文中介绍的 Android 通信功能，包括语音通话、短信收发、使用 GPRS/3G 网络上网，这些功能是基于 Fibocom 无线通信模块实现的。

Android 中的应用程序，如电话程序，其一个操作的流程大概如下：

- 1、电话程序操作界面，调用相应的处理函数；
- 2、相应的操作动作会对应调用 Android 中的电话服务的 java 文件相应的方法函数，即发送对应的 RIL 请求，这些请求都在 ril.h 里面定义了；
- 3、然后把相应的参数传给 RIL 库，并调用 RIL 库的接口函数；
- 4、RIL 库中的接口函数会发送相应的 AT 命令到配置好的端口中，同时监控端口接收模块返回的数据内容，完成一个操作处理。

其中前两步在官方的 android 代码中已经为开发者做好的很大一部分的功能，在处理一些特殊的通信功能时需要在电话服务的代码中修改添加；而后两步是需要模块厂商配合 Android 设备开发商着重修改调试的。

Android 中通信功能实现的一些要点包括：

- 1、模块开关机、休眠唤醒的调试（android 系统中关于模块驱动的调试）；
 - 2、系统与模块通讯端口的调试；
 - 3、RIL 的添加调试；
 - 4、信号的显示、电话程序的功能、短信功能等的调试；
 - 5、mux 的调试；
 - 6、ppp 拨号上网的调试；
 - 7、音频通道切换功能的调试；
 - 8、音量调节功能的调试；
 - 9、其它功能的添加、定制调试。
-

2 调试过程

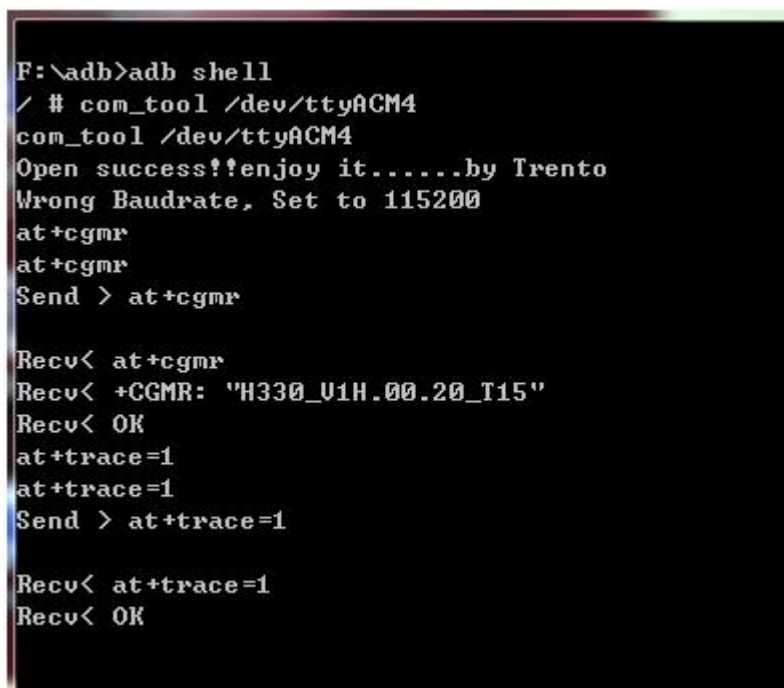
2.1 系统与模块通讯端口的调试

Android 系统的通信功能，实质上是 CPU 通过与无线通信模块进行 AT 命令的数据交互以实现的，有时会把无线通信模块称作为基带（BB，Base Band）。这要求处理器（AP，Application Processor）应具备有与外设进行数据通讯的硬件接口，如 UART、USB 等的接口，当然也需要相应的接口的软件驱动。

对于 Fibocom 的 GPRS 模块，一般使用串口与 AP 进行通讯，在 Android 系统中的内核里都会集成普通的 UART 驱动，所以不需要另外加载驱动，只需要在 Android 的系统配置中把连接到 GPRS 模块的 UART 接口做好相应的配置；

对于 Fibocom 的 WCDMA 模块，如果使用的是 USB 口进行收发 AT 命令，需要在 Android 的内核中加载对应的 USB 驱动，详细加载的方法可以参考 FIBOCOM_H330 Android 驱动程序使用手册.pdf。

在配置完成后，可以通过一些移植的串口小工具如 minicom、广和通写好的 com_tool 等或者用 echo/cat 命令来简单测试对应的模块端口生产的设备节点（如 GPRS 模块对应可能是/dev/ttyS，WCDMA 模块对应的是/dev/ttyACM3），确认通讯用的端口是否可以正常工作，如下图。在确认了端口可以正常收发 AT 命令后，就可以进行下一步的调试了。驱动的配置和编译属于内核 kernel 部分功能。



```
F:\adb>adb shell
/ # com_tool /dev/ttyACM4
com_tool /dev/ttyACM4
Open success!!enjoy it.....by Trento
Wrong Baudrate, Set to 115200
at+cgmr
at+cgmr
Send > at+cgmr

Recv< at+cgmr
Recv< +CGMR: "H330_U1H.00.20_T15"
Recv< OK
at+trace=1
at+trace=1
Send > at+trace=1

Recv< at+trace=1
Recv< OK
```

串口调试小技巧：对于串口的调试，可以通过在处理器的串口上把 txd、rxid 短接，这样，在用测试工具测试发送数据，应该可以接收到相同的数据显示。

2.2 添加调试 RIL

先配置好 init.rc 文件，在 rc 文件中修改 ril-daemon 服务，如下：

(注：如果是 L810 模块需要将红色字体 ttyACM3 替换为 ttyACM2)

#begin

```
service ril-daemon /system/bin/rild -l /system/lib/libreference-ril.so -- -d /dev/ttyACM3
```

```
    class main
```

```
        socket rild stream 660 root radio
```

```
        socket rild-debug stream 660 radio system
```

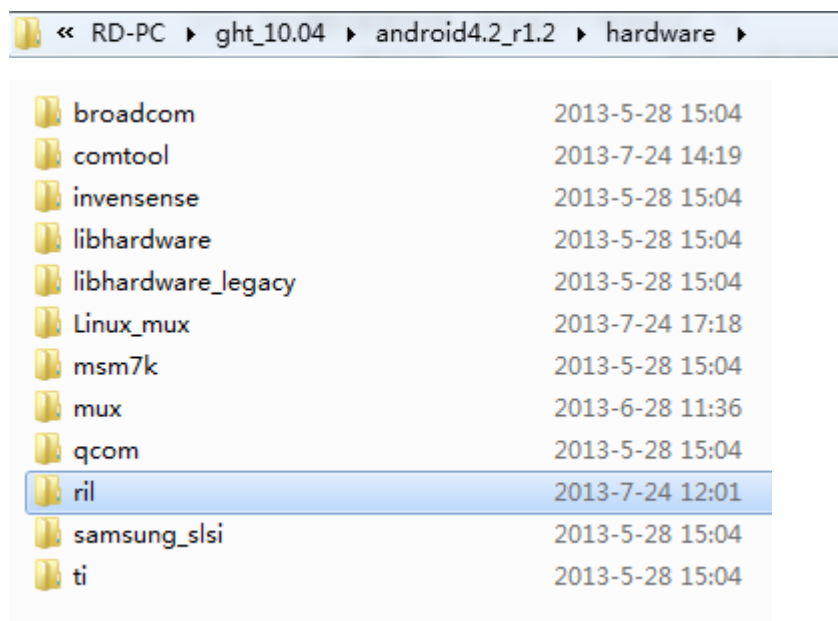
```
    user root
```

```
    group radio cache inet misc audio sdcard_rw log
```

#end

-d 后面的参数可以配置为实际的 usb 映射端口作为 AT 通讯口。H350 使用 ttyACM3 作为 AT 通讯口，使用 ttyACM0 作为数据业务的端口（即用于上网业务）。

接着把对应 android 版本的 RIL 的代码 copy 到/android/hardware/目录下，如下图：



再修改 android/hardware/ril/rild/rild.c 文件，将 switchUser()函数调用注释掉，如下图所示

```
~ ~
1: OpenLib:
2: #endif
3: // switchUser();
4:
5:     dlHandle = dlopen(rilLibPath, RTLD_NOW);
6:
7:     if (dlHandle == NULL) {
```

然后重新编译系统，以得到新的 rild（RIL 的守护进程）、libril.so(封装了与 java 层沟通的接口)、libreference-ril.so（具体的 RIL_REQUEST 的实现），更新镜像 system.img 即可。

具体操作如下：

把整个 android 系统重新 make 一次，再输入 pack 命令，把 rc 文件重新打包到系统镜像中，然后重新烧写新的系统镜像，烧写到 Android 系统中重启即可。

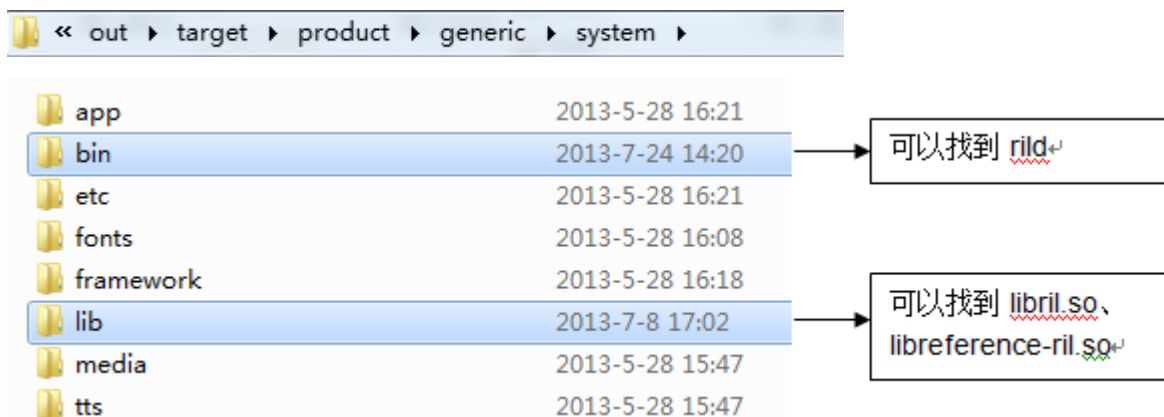
调试技巧：在配置好 init.rc 文件为前提下，为了方便、提高效率，在后面的调试中也可以只编译上述提到的 3 个文件。编译方法：

编译 rild: `mmm hardware/ril/rild/`

编译 libril.so: `mmm hardware/ril/libril/`

编译 libreference-ril.so: `mmm hardware/ril/reference-ril/`

rild 程序在 out 目录的/system/bin 下面生成，另外两个 so 文件在/system/lib 目录下。



编译完成后，通过 adb 工具把 libril.so、libreference-ril.so 两个库文件 push 到/system/lib 目录，并把 rild 可执行文件 push 到/system/bin 目录下，并修改权限使其可执行然后下然后重启：

```
F:\adb>adb push rild /system/bin
277 KB/s (9948 bytes in 0.035s)

F:\adb>adb shell
/ # chmod 777 /system/bin/rild
chmod 777 /system/bin/rild
/ #
```

2.3 调试信号的显示、电话程序的功能、短信功能等

到这一步，就可以达成基本的功能如通话功能、短信功能等，当然这需要 android 系统的界面是带通信功能的 UI，并安装有通话、短信等的 apk 应用程序。

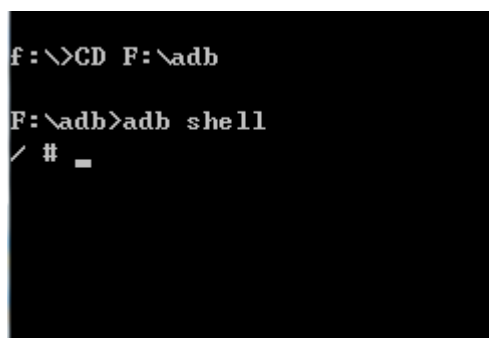
开机后，在 Android 系统中的设置里勾选上“开发人员选项”->>“usb 调试”选项，以使用 adb 调试功能。



用 usb 线把安卓设备与 PC 连接起来，并安装好 android 设备的 adb 驱动。adb 驱动可以通过常用的安卓手机助手工具安装。安装完成会在设备管理其中多出一个 adb 的设备。



在 windows 系统中打开一个 cmd 窗口，进入到 adb 的目录，然后输入“adb shell”命令可以进入到 android 设备的终端，然后如同在 linux 下面一样进行各种调试操作。



首先可以看看 RIL 是否正常加载。打开一个 adb shell，输入 logcat -b radio，查看 radio 的 log 就可以看到 AT 命令是否正常，端口时候正常，如下图为 RILD 初始化正常的一个打印：

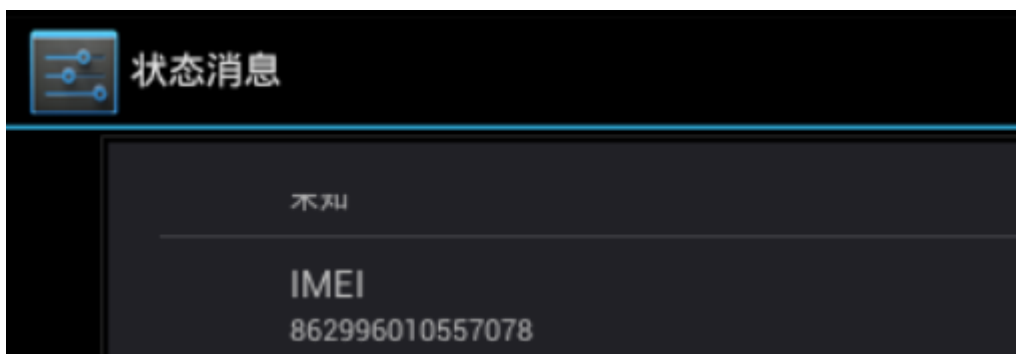
```
F:\adb>adb shell
/ # logcat -b radio
logcat -b radio
E/RILD < 1484>: ***---Colin--111--- hardware ril rild
D/RILD < 1484>: *****--//switchUser()---*****
I/RIL < 1484>: Opening tty device /dev/ttyUSB0
E/RILC < 1484>: RIL_register: RIL version 6
D/RIL < 1484>: setRadioState(0)
D/AT < 1484>: AT> ATE0Q0U1
D/AT < 1484>: AT< ATE0Q0U1
D/AT < 1484>: AT< OK
D/AT < 1484>: AT> ATE0Q0U1
D/AT < 1484>: AT< OK
D/AT < 1484>: AT> ATS0=0
D/AT < 1484>: AT< OK
D/AT < 1484>: AT> AT+CME=1
D/AT < 1484>: AT< OK
D/AT < 1484>: AT> AT+CREG=2
D/AT < 1484>: AT< OK
D/AT < 1484>: AT> AT+CGREG=1
D/AT < 1484>: AT< OK
D/AT < 1484>: AT> AT+CCWA=1
D/AT < 1484>: AT< OK
D/AT < 1484>: AT> AT+CMOD=0
```

一般在初始化完成后，上层应用会查询模块的 IMEI 号，并会有不断查询信号值的操作，可以根据这些来判断 RIL 是否已经正常加载运行了。

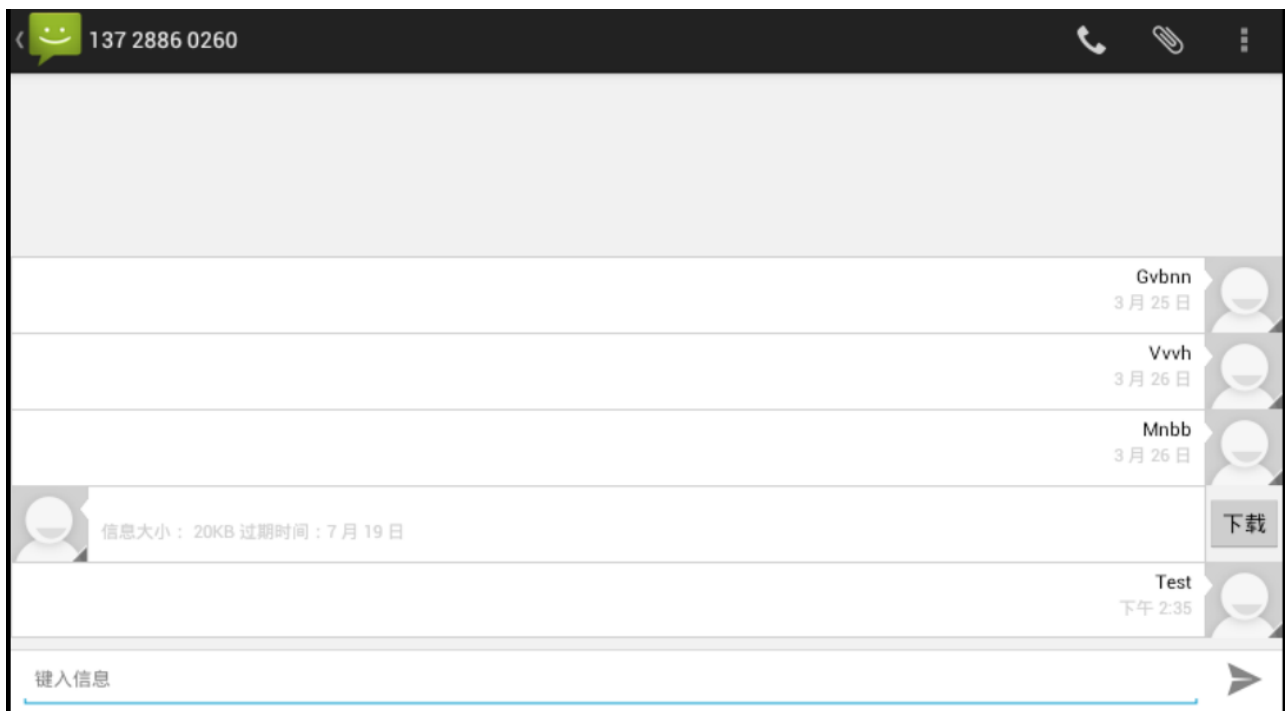
IMEI 号：

```
D/RILB < 1821>: getLteOnCdmaMode=0 curVal=-1 product_type='' lteOnCdmaProduct
Type=''
D/GSM < 1821>: [IccCard] Broadcasting intent ACTION_SIM_STATE_CHANGED READY
reason null
D/AT < 1484>: AT< +CGSN: "862996010557078"
U/GSM < 1821>: SIMRecords:fetchSimRecords 0
D/RILJ < 1821>: [0022]> GET_IMSI
D/AT < 1484>: AT< OK
```

进入“设置”->“平板信息”->“状态信息”，可以看到有 IMEI 的信息显示，否则为“Unknown”：



CSQ 信号值查询：



RIL 的相对应的 log:

```
D/RILJ    < 2033>: [0058] > SEND_SMS
D/RIL     < 1484>: onRequest: SEND_SMS
D/AT      < 1484>: AT> AT+CMGS=17
D/AT      < 1484>: AT< >
D/AT      < 1484>: AT> 0001000b813127880662f00000004d4f29c0e^Z
D/AT      < 1484>: AT< +CMGS: 24
D/AT      < 1484>: AT< OK
D/RILJ    < 2033>: [0058] < SEND_SMS { messageRef = 0, errorCode = 0, ackPdu = nu
11>
```

2.4 添加 MUX

MUX 功能即使端口可以多路复用的功能，针对只能用一个串口时，却要实现 AT 控制、ppp 上网两种功能的这种情况而添加的，如使用 G600、G610、G510、G520 等 GPRS 模块与 AP 通讯时要实现上述功能时就需要 MUX 了。如果使用 H330 系列的 WCDMA 模块，使用 USB 接口通讯，本身在系统中就会生成 ttyACM0~ttyACM6 多个设备节点以供通讯，这时候是没有必要加载添加 MUX 功能的

像添加 RIL 一样，同样先配置好 init.rc 文件，在 rc 文件中添加 gsmmux 服务，如下：

#begin

```
service gsmmux /system/bin/gsmmux -p /dev/ttyS1 -b 115200 -s /dev/mux -w /dev/ptmx /dev/ptmx
```

```
class main
```

```
user root
```

```
group radio cache inet misc
```

```
oneshot
```

#end

并把 ril-daemon 服务的配置的端口修改为/dev/mux1:

#begin

```
service ril-daemon /system/bin/rild -l /system/lib/libreference-ril.so -- -d /dev/mux1
```

```
class main
```

```
socket rild stream 660 root radio
```

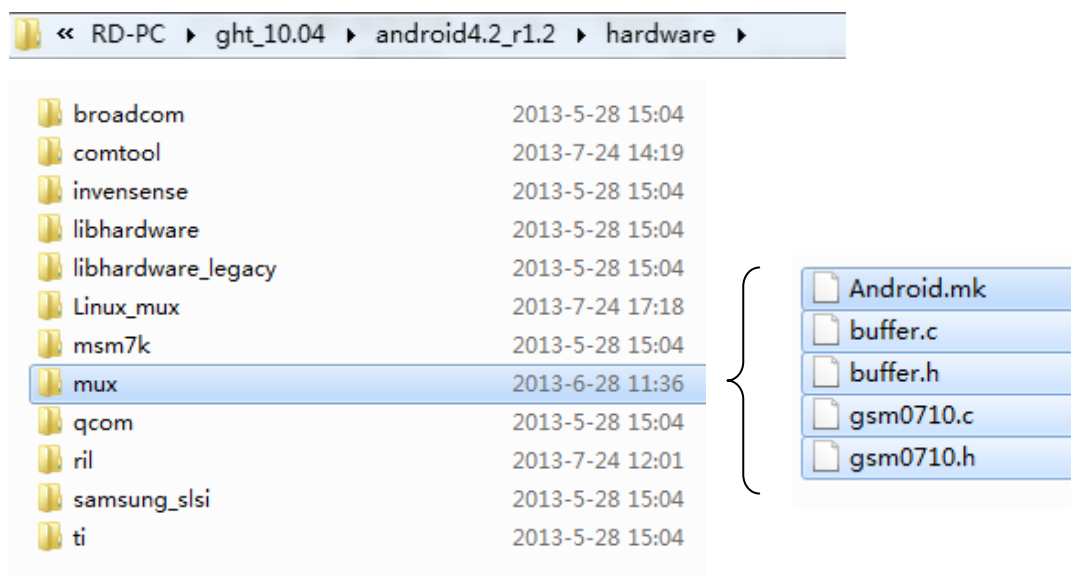
```
socket rild-debug stream 660 radio system
```

```
user root
```

```
group radio cache inet misc audio sdcard_rw log
```

#end

接着把 MUX 的代码 copy 到/android/hardware/目录下，如下图:



把整个 android 系统重新 make 一次，再输入 pack 命令，把 rc 文件重新打包到系统镜像中，然后重新烧写新的系统镜像，重启就可以了。

重启后，在/dev 目录下会生成 mux0、mux1 两个设备节点，这两个就是由 ttyS1 生成的虚拟串口：

可以从 RIL 的 log 里面看到使用的端口变为/dev/mux1,这时候初始化成功，就表示 MUX 添加成功了，如下图：

```
F:\adb>adb shell
/ # ls /dev/mux*
ls /dev/mux*
/dev/mux0
/dev/mux1
/ # ls -l /dev/mux*
ls -l /dev/mux*
lrwxrwxrwx root      radio      2013-07-25 15:11 mux0 -> /dev/pts/0
lrwxrwxrwx root      radio      2013-07-25 15:11 mux1 -> /dev/pts/1
/ #
```

添加 MUX 后重复一遍 2.4 节的调试，确保在添加了 MUX 功能后，ril 使用 mux1 端口也是可以正常工作的。

2.5 调试 ppp 拨号上网功能

前面的工作确认正常后，就可以进行 ppp 拨号上网的调试了。GPRS 模块使用了 MUX，可以在 ppp 拨号脚本中配置为拨号使用的端口为/dev/mux0，而 WCDMA 模块 H330 可以使用 ttyACM0。

Mark: 在广和通发布的 RIL_V42.00.03 以后的 RIL code 已经集成了 ppp 拨号上网的脚本，估不需要在另外配置 pppd 的脚本也可以上网了。下面的步骤可以用作调试的参考。

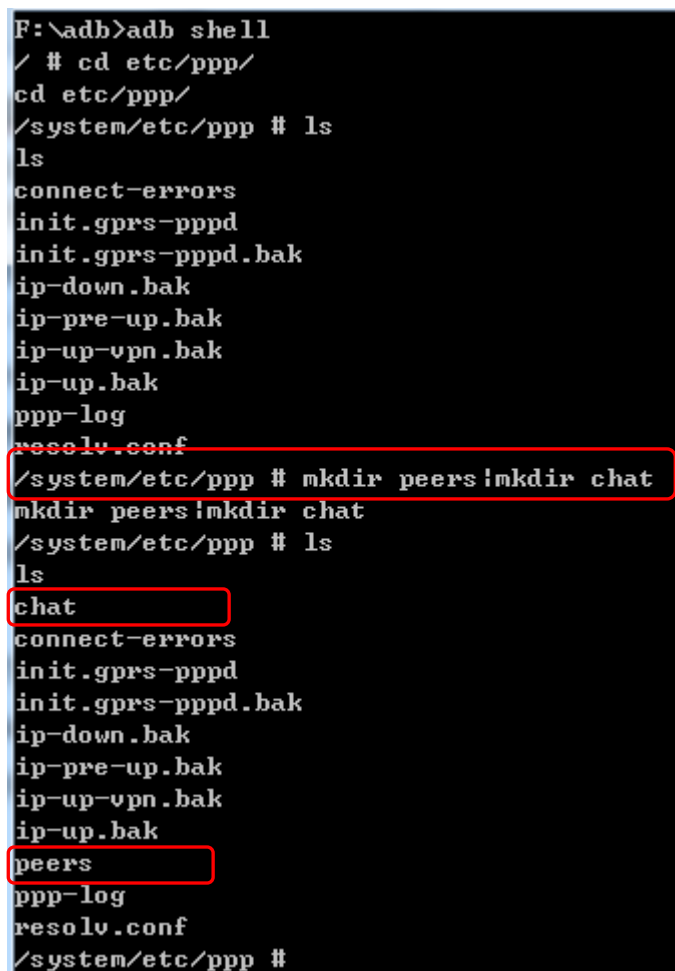
pppd 拨号脚本包括 ppp-dial 文件、cmtc-isp 文件等（如果用到认证功能，pap-secrets、chap-secrets 文件也是需要的）。配置方法：

1、建立 peers 和 chat 目录

peers 目录路径为/etc/ppp/peers/;

chat 目录路径为/etc/ppp/chat/;

调试过程中，可以进入 adb shell 后使用 mkdir 命令可以建立目录。



```
F:\adb>adb shell
/ # cd etc/ppp/
cd etc/ppp/
/system/etc/ppp # ls
ls
connect-errors
init.gprs-pppd
init.gprs-pppd.bak
ip-down.bak
ip-pre-up.bak
ip-up-vpn.bak
ip-up.bak
ppp-log
resolv.conf
/system/etc/ppp # mkdir peers!mkdir chat
mkdir peers!mkdir chat
/system/etc/ppp # ls
ls
chat
connect-errors
init.gprs-pppd
init.gprs-pppd.bak
ip-down.bak
ip-pre-up.bak
ip-up-vpn.bak
ip-up.bak
peers
ppp-log
resolv.conf
/system/etc/ppp #
```

2、拷贝脚本文件

把 ppp-dial 文件拷贝到 peers 目录下，把 cmtc-isp 文件拷贝到 chat 目录下即可。

调试过程中，可以通过 adb 工具的 adb push 命令拷贝到 android 系统中。

```

F:\adb>adb push ppp-dial /etc/ppp/peers
14 KB/s <271 bytes in 0.018s>

F:\adb>adb push cmtc-isp /etc/ppp/chat
10 KB/s <192 bytes in 0.018s>

F:\adb>adb shell
/ # ls -l /etc/ppp/peers
ls -l /etc/ppp/peers
-rw-rw-rw- root      root          271 2013-07-25 15:30 ppp-dial
/ # ls -l /etc/ppp/chat
ls -l /etc/ppp/chat
-rw-rw-rw- root      root          192 2013-06-14 12:24 cmtc-isp
/ # _

```

到这一步，重启 Android 设备后，就可以通过 android 系统中的“设置”->“更多”->“移动网络”里面的“启用网络数据流量”来控制 ppp 拨号上网的连接与断开了。



通过在 adb shell 中敲入“ifconfig ppp0”，可以查看是否获取到了 IP 地址，如果成功获取到 IP 地址，就可以在 android 中上网了。

```

F:\adb>adb shell
/ # ifconfig ppp0
ifconfig ppp0
ppp0: No such device
/ # ifconfig ppp0
ifconfig ppp0
ppp0: ip 10.30.164.218 mask 255.255.255.255 flags [up point-to-point running mul
ticast]
/ #

```

调试技巧：调试的时候可以先在 ppp-dial 脚本中加入 “nodetach” 和 “debug” 选项，如下图，成功后再去掉这两个选项：

```

/dev/mux0
115200
nocrtscts
nocdtrcts
#nobsdcomp
#+pap
modem
#noauth
#auth
#user card
nodetach
debug
#logfile /etc/ppp/ppp-log

```

然后把 ppp-dial 文件重新 push 到 /etc/ppp/peers 目录下，这样 pppd 将会不后台运行，并打开调试模式，然后直接在 adb shell 中敲入命令 “pppd call ppp-dial”，就可以手动拨号上网，以测试能否正常调用 pppd 并获取 IP 地址，如下图：

```

F:\adb>adb shell
/ # pppd call ppp-dial
pppd call ppp-dial
abort on <BUSY>
abort on <NO CARRIER>
abort on <ERROR>
abort on <+CME ERROR:100>
send <AT^M>
expect <OK>
AT^M^M
OK
-- got it

send <AT+CME=2^M>
expect <OK>
^M
AT+CME=2^M^M
OK
-- got it

```


2.6 调试音频通道切换功能

需要 Android 工程师结合相对应的平台，在电话服务层、HAL 层或驱动层做相应的添加处理，可以通过提供 AT 命令配合调试实现。

2.7 调试音量调节功能

需要 Android 工程师结合相对应的平台，在电话服务层、HAL 层或驱动层做相应的添加处理，可以通过提供 AT 命令配合调试实现。

2.8 添加、定制其它功能

需要 Android 工程师结合相对应的平台，在电话服务层、HAL 层或驱动层做相应的添加处理，可以通过提供 AT 命令配合调试实现。

3 添加 3G 所需要内核驱动配置

3.1 修改内核编译配置(kernel 根目录下的.config 文件中)，确保下面的配置项已经被选定：

– PPP 拨号的相关配置项：

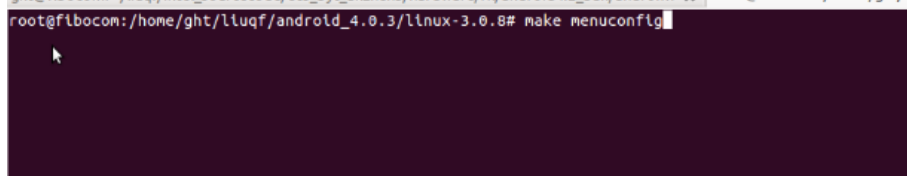
```
CONFIG_PPP=y  
CONFIG_PPP_MULTILINK=y  
CONFIG_PPP_FILTER=y  
CONFIG_PPP_ASYNC=y  
CONFIG_PPP_SYNC_TTY=y  
CONFIG_PPP_DEFLATE=y  
CONFIG_PPP_BSDCOMP=y
```

– USB ACM 相关配置项：

```
CONFIG_USB_ANNOUNCE_NEW_DEVICES=y (此选项存在的情况建议配置一下，没有请忽略)  
CONFIG_USB_ACM=y
```

3.2 详细操作

打开 Terminal 工具，进入 kernel 目录(假定为：，/home/ght/liuqf/android-4.0.3/linux-3.0.8/)，然后执行 make <configuration>命令，在本文中，假定使用标准 make menuconfig)



按照下列图指引完成配置

– PPP 拨号的相关配置项：

Linux Kernel Configuration

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [] excluded <M> module < > module capable

```
General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
System Type --->
Bus support --->
Kernel Features --->
Boot options --->
CPU Power Management --->
Floating point emulation --->
Userspace binary formats --->
Power management options --->
[*] Networking support --->
  Device Drivers --->
  File systems --->
  Kernel hacking --->
  Security options --->
  < > Cryptographic API --->
  Library routines --->
---
Load an Alternate Configuration File
```

v(+)

<Select> < Exit > < Help >

Device Drivers

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [] excluded <M> module < > module capable

```
^(-)
< > Parallel port support --->
[ ] Block devices --->
[*] Misc devices --->
< > ATA/ATAPI/MFM/RLL support --->
  SCSI device support --->
< > Serial ATA (prod) and Parallel ATA (experimental) drivers --->
[ ] Multiple devices driver support (RAID and LVM) --->
[ ] Fusion MPT device support --->
  IEEE 1394 (FireWire) support --->
< > I2O device support --->
[*] Network device support --->
  ISDN support --->
< > Telephony support --->
  Input device support --->
  Character devices --->
<*> I2C support --->
[*] SPI support --->
  PPS support --->
  *- GPIO Support --->
< > Dallas's 1-wire support --->
```

v(+)

<Select> < Exit > < Help >

```
Network device support
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing
<Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for
Search. Legend: [*] built-in [ ] excluded <M> module <> module capable

^(-)
*** Enable WiMAX (Networking options) to see the WiMAX drivers ***
USB Network Adapters --->
[ ] Wan interfaces support --->
<> FDDI driver support
[ ] HIPPI driver support (EXPERIMENTAL)
<*> PPP (point-to-point protocol) support
[*] PPP multilink support (EXPERIMENTAL)
[*] PPP filtering
<*> PPP support for async serial ports
<*> PPP support for sync tty ports
<*> PPP Deflate compression
<*> PPP BSD-Compress compression
<> PPP MPPE compression (encryption) (EXPERIMENTAL) (NEW)
<> PPP over Ethernet (EXPERIMENTAL) (NEW)
<> PPP over L2TP (EXPERIMENTAL) (NEW)
<> PPP on L2TP Access Concentrator (NEW)
<> PPP on PPTP Network Server (NEW)
<> SLIP (serial line) support
[ ] Fibre Channel driver support
<> Network console logging support (EXPERIMENTAL)

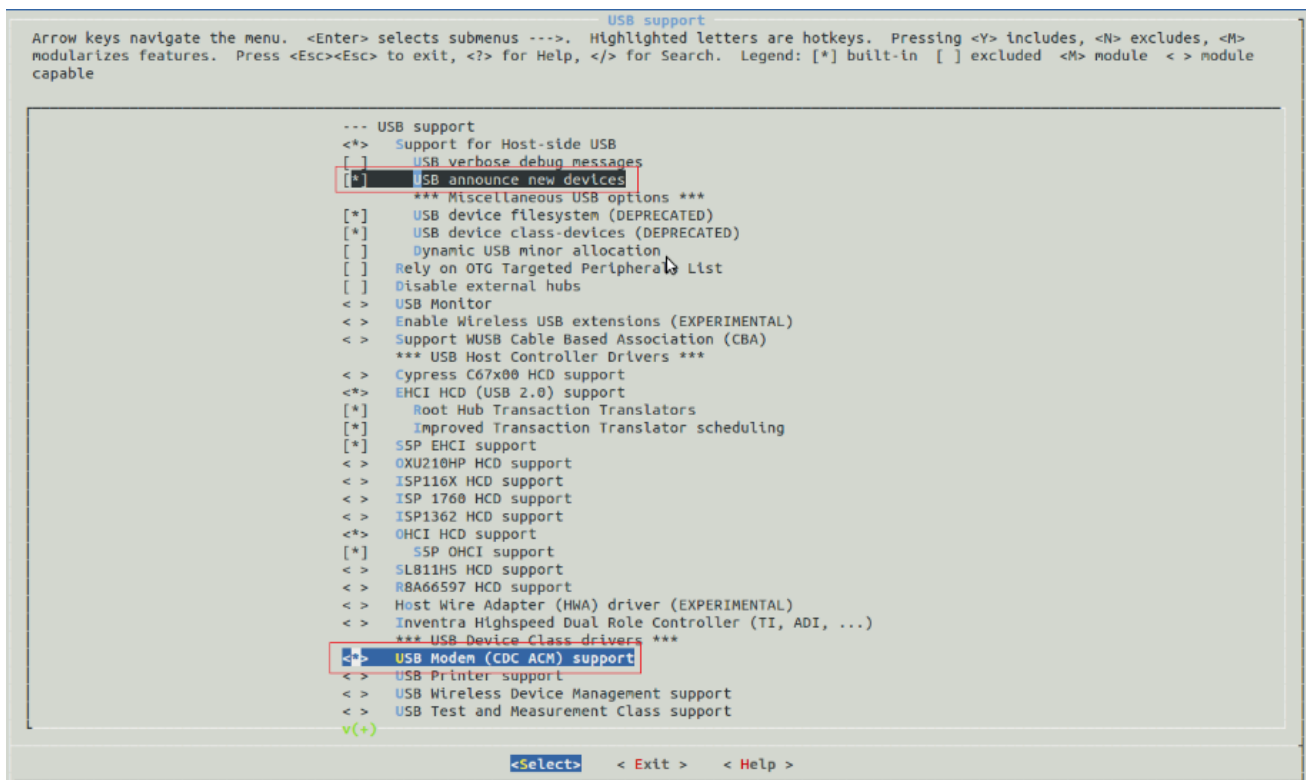
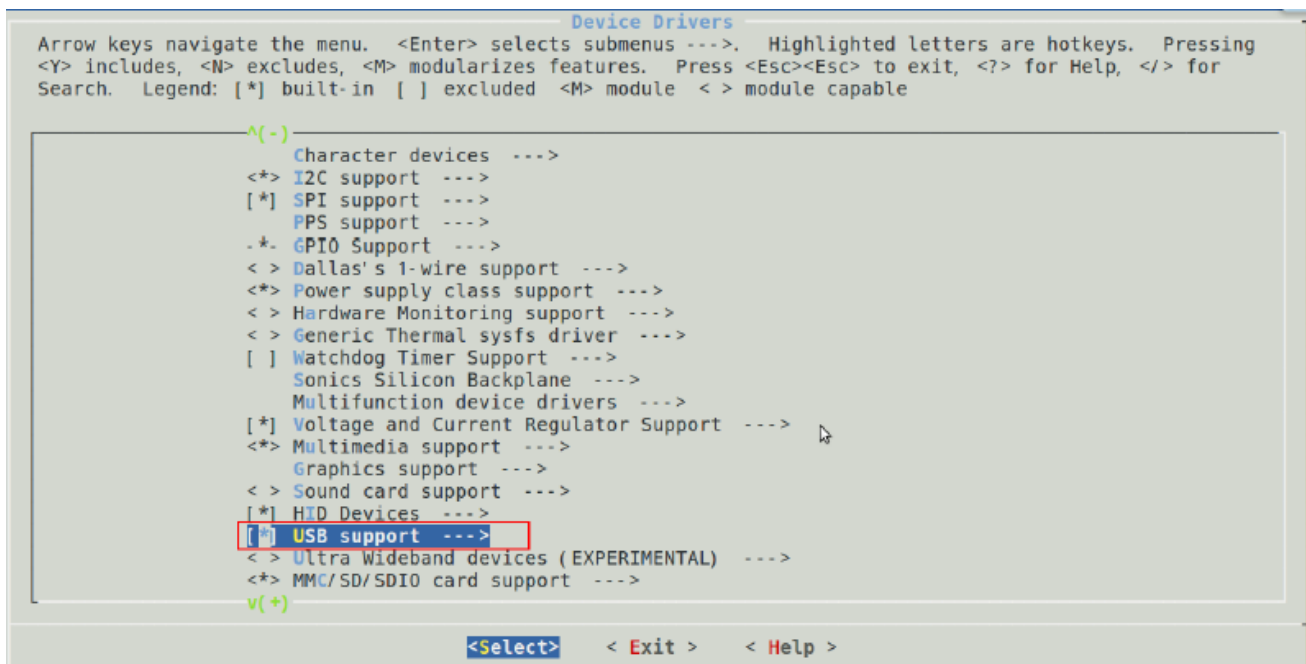
<Select> < Exit > < Help >
```

- USB ACM 相关配置项:

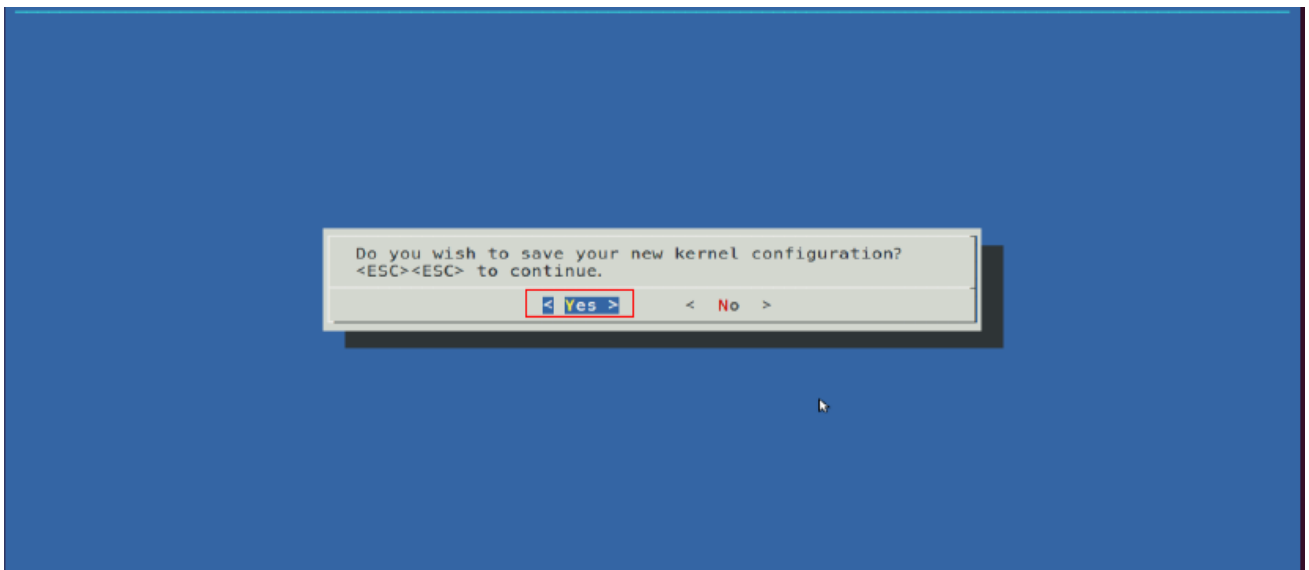
```
Linux Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing
<Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for
Search. Legend: [*] built-in [ ] excluded <M> module <> module capable

General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
System Type --->
Bus support --->
Kernel Features --->
Boot options --->
CPU Power Management --->
Floating point emulation --->
Userspace binary formats --->
Power management options --->
[*] Networking support --->
[*] Device Drivers --->
File systems --->
Kernel hacking --->
Security options --->
<> Cryptographic API --->
Library routines --->
---
Load an Alternate Configuration File
v(+)

<Select> < Exit > < Help >
```



如上操作选完所需选项后，通过选择<Exit>按钮，逐层退出各个配置界面。最后在保存配置界面中，选择<Yes>选项并退出。



完成配置后，即可运行 **make** 命令，开始编译修改后的内核。

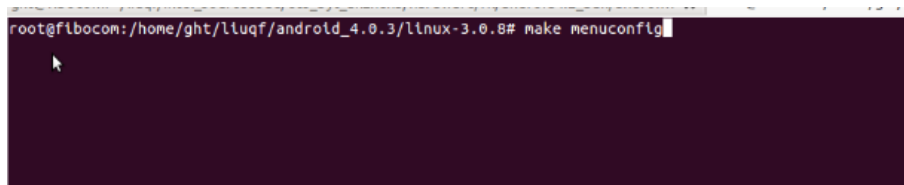
4 添加 4G 所需要内核驱动配置

4.1 修改内核编译配置(kernel 根目录下的.config 文件中)，确保下面的配置项已经被选定：

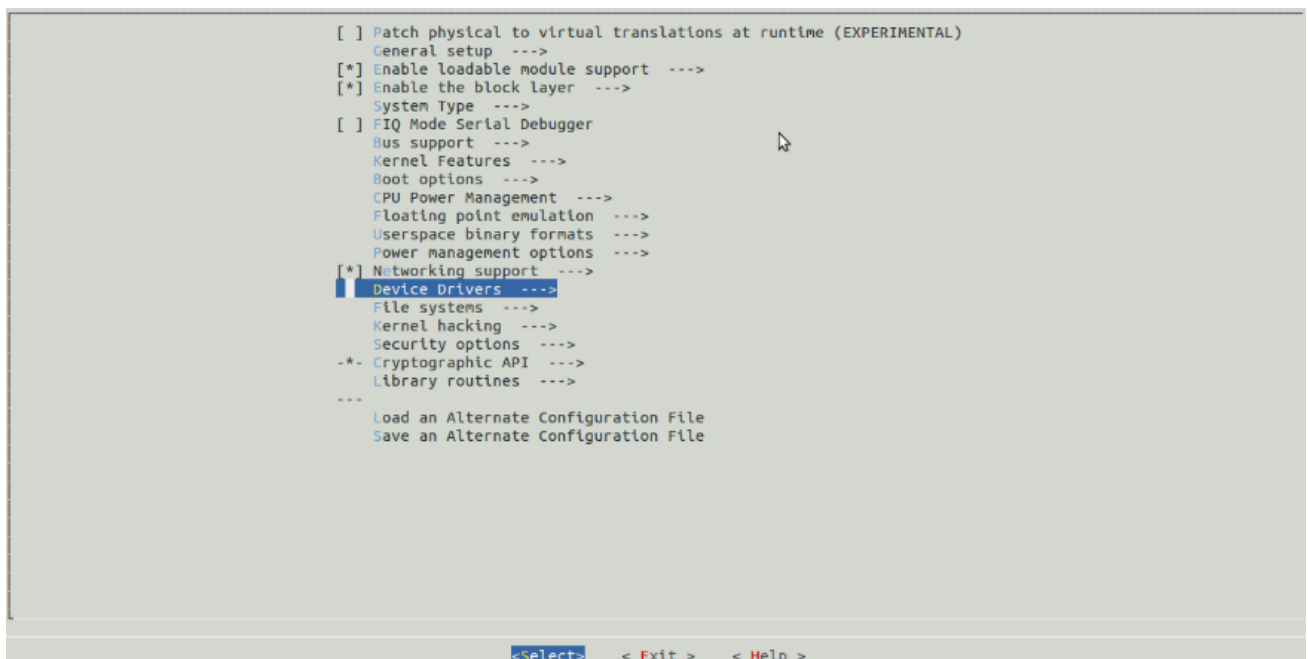
```
CONFIG_USB_USBNET=y
CONFIG_NETDEVICES=y
CONFIG_USB_NET_CDC_NCM=y
```

4.2 详细操作

打开 Terminal 工具，进入 kernel 目录(假定为:， /home/ght/liuqf/android-4.0.3/linux-3.0.8/)，然后执行 make <configuration>命令，在本文中，假定使用标准 make menuconfig)



CDC ECM 驱动配置项，按照下列图指引完成配置:



```

Generic Driver Options --->
< > Connector - unified userspace <-> kernelspace linker --->
< * > Memory Technology Device (MTD) support --->
< > Parallel port support --->
[ * ] Block devices --->
[ * ] Misc devices --->
< > ATA/ATAPI/MFM/RLI support (DEPRECATED) --->
SCSI device support --->
< > Serial ATA and Parallel ATA drivers --->
[ * ] Multiple devices driver support (RAID and LVM) --->
< > Generic Target Core Mod (TCM) and ConfigFS Infrastructure --->
[ * ] Network device support --->
[ ] ISDN support --->
< > Telephony support --->
Input device support --->
Character devices --->
< * > I2C support --->
[ * ] SPI support --->
PPS support --->
PTP clock support --->
- * - GPIO Support --->
< > Dallas's 1-wire support --->
< * > Power supply class support --->
< * > Hardware Monitoring support --->
< > Generic Thermal sysfs driver --->
[ * ] Watchdog Timer Support --->
Sonics Silicon Backplane --->
Broadcom specific AMBA --->
[ * ] Multifunction device drivers --->
[ * ] Voltage and Current Regulator Support --->
< * > Multimedia support --->
Graphics support --->
v(+)

```

<Select> < Exit > < Help >

```

--- Network device support
< * > Intermediate Functional Block support
< > Dummy net driver support
< > Bonding driver support
< > MAC-VLAN support (EXPERIMENTAL)
< > EQL (serial line load balancing) support
< * > Universal TUN/TAP device driver support
< > Virtual ethernet pair device
- * - Generic Media Independent Interface device support
< * > PHY Device support and infrastructure --->
[ * ] Ethernet (10 or 100Mbit) --->
[ ] Ethernet (1000 Mbit) --->
[ ] Ethernet (10000 Mbit) --->
[ * ] Wireless LAN --->
*** Enable WiMAX (Networking options) to see the WiMAX drivers ***
[ * ] USB Network Adapters --->
[ ] Wan interfaces support --->
*** CAIF transport drivers ***
< * > PPP (point-to-point protocol) support
[ ] PPP multilink support (EXPERIMENTAL)
[ ] PPP filtering
< * > PPP support for async serial ports
< > PPP support for sync tty ports
< * > PPP Deflate compression
< * > PPP BSD-Compress compression
< * > PPP MPPE compression (encryption) (EXPERIMENTAL)
< > PPP over Ethernet (EXPERIMENTAL)
< * > PPP on L2TP Access Concentrator
< * > PPP on PPTP Network Server
< > SLIP (serial line) support
< > Network console logging support

```

<Select> < Exit > < Help >


```

< > USB CATC NetMate-based Ethernet device support (EXPERIMENTAL)
< > USB KLSI KLSUSB101-based ethernet device support
< > USB Pegasus/Pegasus-II based ethernet device support
< > USB RTL8150 based ethernet device support (EXPERIMENTAL)
<+> Multi-purpose USB Networking Framework
<+> ASIX AX88xxx Based USB 2.0 Ethernet Adapters
-+> CDC Ethernet support (smart devices such as cable modems)
< > CDC EEM support
<+> CDC NCM support
< > Davicom DM9601 based USB 1.1 10/100 ethernet devices
< > SMSC LAN75XX based USB 2.0 gigabit ethernet devices
< > SMSC LAN95XX based USB 2.0 10/100 ethernet devices
< > GeneSys GL620USB-A based cables
<+> NetChip 1080 based cables (Laplink, ...)
< > Prolific PL-2301/2302/25A1 based cables
< > MosChip MCS7830 based Ethernet adapters
< > Host for RNDIS and ActiveSync devices (EXPERIMENTAL)
<+> Simple USB Network Links (CDC Ethernet subset)
[ ] ALi M5632 based 'USB 2.0 Data Link' cables
[ ] AnchorChips 2720 based cables (Xircom PGUNET, ...)
[*] eTEK based host-to-host cables (Advance, Belkin, ...)
[*] Embedded ARM Linux links (iPaq, ...)
[ ] Epson 2888 based firmware (DEVELOPMENT)
[ ] KT Technology KC2190 based cables (InstaNet)
<+> Sharp Zaurus (stock ROMs) and compatible
< > Conexant CX82310 USB ethernet port
< > Samsung Kalmia based LTE USB modem
< > Option USB High Speed Mobile Devices
< > Intellon PLC based usb adapter
< > CDC Phonet support
< > Apple iPhone USB Ethernet driver
< > USB-to-WWAN Driver for Sierra Wireless modems
v(+)

```

<Select> < Exit > < Help >

```

< > USB CATC NetMate-based Ethernet device support (EXPERIMENTAL)
< > USB KLSI KLSUSB101-based ethernet device support
< > USB Pegasus/Pegasus-II based ethernet device support
< > USB RTL8150 based ethernet device support (EXPERIMENTAL)
<+> Multi-purpose USB Networking Framework
<+> ASIX AX88xxx Based USB 2.0 Ethernet Adapters
-+> CDC Ethernet support (smart devices such as cable modems)
< > CDC EEM support
<+> CDC NCM support
< > Davicom DM9601 based USB 1.1 10/100 ethernet devices
< > SMSC LAN75XX based USB 2.0 gigabit ethernet devices
< > SMSC LAN95XX based USB 2.0 10/100 ethernet devices
< > GeneSys GL620USB-A based cables
<+> NetChip 1080 based cables (Laplink, ...)
< > Prolific PL-2301/2302/25A1 based cables
< > MosChip MCS7830 based Ethernet adapters
< > Host for RNDIS and ActiveSync devices (EXPERIMENTAL)
<+> Simple USB Network Links (CDC Ethernet subset)
[ ] ALi M5632 based 'USB 2.0 Data Link' cables
[ ] AnchorChips 2720 based cables (Xircom PGUNET, ...)
[*] eTEK based host-to-host cables (Advance, Belkin, ...)
[*] Embedded ARM Linux links (iPaq, ...)
[ ] Epson 2888 based firmware (DEVELOPMENT)
[ ] KT Technology KC2190 based cables (InstaNet)
<+> Sharp Zaurus (stock ROMs) and compatible
< > Conexant CX82310 USB ethernet port
< > Samsung Kalmia based LTE USB modem
< > Option USB High Speed Mobile Devices
< > Intellon PLC based usb adapter
< > CDC Phonet support
< > Apple iPhone USB Ethernet driver
< > USB-to-WWAN Driver for Sierra Wireless modems
v(+)

```

<Select> < Exit > < Help >

如上操作选完所需选项后，通过选择<Exit>按钮，逐层退出各个配置界面。最后在保存配置界面中，选择<Yes>并退出。

Do you wish to save your new configuration ? <ESC><ESC>
to continue.

<Yes> < No >

5 如何确认 NCM/ACM 驱动已经配置入系统

开机启动时，执行 `dmesg` 命令，查看内核 LOG，发现红框信息即说明 NCM 驱动已经配置入系统

```
<6>[ 1.492528] eth0: dm9000a at e0838000,e083c00c IRQ 39 MAC: 08:90:00:a0:02:10 (platform data)
<6>[ 1.500005] usbcore: registered new interface driver asix
<6>[ 1.505217] usbcore: registered new interface driver cdc_ether
<6>[ 1.511027] usbcore: registered new interface driver net1080
<6>[ 1.516654] usbcore: registered new interface driver cdc_subset
<6>[ 1.522542] usbcore: registered new interface driver zaurus
<6>[ 1.527994] cdc_ncm: 04-Aug-2011
<6>[ 1.531260] usbcore: registered new interface driver cdc_ncm
<6>[ 1.536918] sdhci: Secure Digital Host Controller Interface driver
<6>[ 1.542075] sdhci: Copyright(c) Pierre Ossman
```

系统启动完全后，L810 模块上电开机，再执行 `dmesg` 命令，查看内核 LOG，发现红框信息说明 L810 NCM 驱动已经加载 OK，并且生成 `usb0 usb1 usb2 usb3` 等 NCM 网口。

```
<6>[71820.808950] usb 1-1.2: new high speed USB device number 8 using s5p-ehci
<6>[71820.928605] usb 1-1.2: New USB device found, idVendor=1519, idProduct=0443
<6>[71820.928669] usb 1-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
<6>[71820.928731] usb 1-1.2: Product: 3 CDC-ACM + 4 CDC-NCM
<6>[71820.928775] usb 1-1.2: Manufacturer: Comneon
<6>[71820.929750] usb 1-1.2: SerialNumber: 865204020007441
<3>[71820.960903] cdc_acm 1-1.2:1.0: This device cannot do calls on its own. It is not a modem.
<6>[71820.961505] cdc_acm 1-1.2:1.0: ttyACM0: USB ACM device
<3>[71820.967194] cdc_acm 1-1.2:1.2: This device cannot do calls on its own. It is not a modem.
<6>[71820.967753] cdc_acm 1-1.2:1.2: ttyACM1: USB ACM device
<3>[71820.972342] cdc_acm 1-1.2:1.4: This device cannot do calls on its own. It is not a modem.
<6>[71820.973884] cdc_acm 1-1.2:1.4: ttyACM2: USB ACM device
<6>[71820.994237] usb 1-1.2: MAC-Address: 0x00:0x00:0x11:0x12:0x13:0x14
<6>[71821.001998] cdc_ncm 1-1.2:1.6: usb0: register 'cdc_ncm' at usb-s5p-ehci-1.2, CDC NCM, 00:00:11:12:13:14
<6>[71821.020104] usb 1-1.2: MAC-Address: 0x00:0x00:0x11:0x12:0x13:0x16
<6>[71821.025785] cdc_ncm 1-1.2:1.8: usb1: register 'cdc_ncm' at usb-s5p-ehci-1.2, CDC NCM, 00:00:11:12:13:16
<6>[71821.040494] usb 1-1.2: MAC-Address: 0x00:0x00:0x11:0x12:0x13:0x18
<6>[71821.048300] cdc_ncm 1-1.2:1.10: usb2: register 'cdc_ncm' at usb-s5p-ehci-1.2, CDC NCM, 00:00:11:12:13:18
<6>[71821.066349] usb 1-1.2: MAC-Address: 0x00:0x00:0x11:0x12:0x13:0x1a
<6>[71821.072507] cdc_ncm 1-1.2:1.12: usb3: register 'cdc_ncm' at usb-s5p-ehci-1.2, CDC NCM, 00:00:11:12:13:1a
<6>[71821.072507] cdc_ncm 1-1.2:1.12: usb3: register 'cdc_ncm' at usb-s5p-ehci-1.2, CDC NCM, 00:00:11:12:13:1a
```

执行 `netcfg` 命令可以查询到有 `usb0 usb1 usb2 usb3` 等网口

```
/ # netcfg
lo UP 127.0.0.1/8 0x00000049 00:00:00:00:00:00
ifb0 DOWN 0.0.0.0/0 0x00000082 ca:98:28:4b:1d:a8
ifb1 DOWN 0.0.0.0/0 0x00000082 72:1a:49:e8:ec:14
eth0 UP 0.0.0.0/0 0x00001003 08:90:00:a0:02:10
sit0 DOWN 0.0.0.0/0 0x00000080 00:00:00:00:00:00
ip6tnl0 DOWN 0.0.0.0/0 0x00000080 00:00:00:00:00:00
usb0 DOWN 0.0.0.0/0 0x00001002 00:00:11:12:13:14
usb1 DOWN 0.0.0.0/0 0x00001002 00:00:11:12:13:16
usb2 DOWN 0.0.0.0/0 0x00001002 00:00:11:12:13:18
usb3 DOWN 0.0.0.0/0 0x00001002 00:00:11:12:13:1a
/ #
```

执行 `ls /dev/ttyACM*` 命令可以查询到有 3 个通信端口，`ttyACM0 ttyACM1 ttyACM2`

```
/ # ls /dev/ttyACM*
/dev/ttyACM0
/dev/ttyACM1
/dev/ttyACM2
```