1.首先要确定mu609模块识别到了没有，可以在shell下busybox lsusb查看usb设备，如果出现如下信息，就是已经识别到mu609了。

**[plain]** [view plaincopy](http://blog.csdn.net/suntao222/article/details/8930744)

1. shell@android:/ # busybox lsusb
2. Bus 003 Device 002: ID 12d1:1506

2.查看ttyUSB接口属性是否修改为0777

**[plain]** [view plaincopy](http://blog.csdn.net/suntao222/article/details/8930744)

1. shell@android:/ # ll /dev/ttyUSB\*
2. crwxrwxrwx root     root     188,   0 2013-05-15 12:01 ttyUSB0
3. crw------- root     root     188,   1 2013-05-15 12:00 ttyUSB1
4. crwxrwxrwx root     root     188,   2 2013-05-15 12:00 ttyUSB2

3.模块已经识别了，ttyUSB接口都有，现在手动发AT命令看mu609是否响应。

**[plain]** [view plaincopy](http://blog.csdn.net/suntao222/article/details/8930744)

1. shell@android:/ # cd /dev/
2. shell@android:/dev # cat ttyUSB0 &
3. [1] 869
4. shell@android:/dev # echo AT+CGMI > ttyUSB0
5. shell@android:/dev #
7. Huawei Technologies Co., Ltd.


11. OK

14. shell@android:/dev # echo AT+CGMM > ttyUSB0
15. shell@android:/dev #
17. MU609


21. OK

24. shell@android:/dev #

OK，mu609正常响应了。

4.现在先抛开上层RIL不管，单纯从shell手动拨号看能否正常工作。需要自己写两个脚本，拷贝到/etc/ppp目录下  
第一个脚本是/etc/ppp/chat/unicom,内容如下:

**[plain]** [view plaincopy](http://blog.csdn.net/suntao222/article/details/8930744)

1. # this is the chat script for unicom
2. ABORT "NO CARRIER"
3. ABORT "NO DIALTONE"
4. ABORT "ERROR"
5. ABORT "NO ANSWER"
6. ABORT "BUSY"
7. TIMEOUT 120
8. "" at
9. OK AT+CGDCONT=1,"IP","3gnet"
10. OK atdt\*99\*\*\*1#
11. CONNECT

第二个脚本是/etc/ppp/peers/unicom,内容如下:

**[plain]** [view plaincopy](http://blog.csdn.net/suntao222/article/details/8930744)

1. # /system/etc/ppp/peers/unicom
2. /dev/ttyUSB0
3. 115200
4. crtscts
5. connect '/system/bin/chat -s -v -f /system/etc/ppp/chat/unicom'
6. debug
7. nodetach
8. ipcp-accept-local
9. ipcp-accept-remote
10. defaultroute
11. usepeerdns
12. user card
13. password card

chat和peers两个目录没有，需要自己mkdir。还需要用到chat，就是之前在device.mk中copy过来的，android自身不带，在我的资源中有编译好的。因为rild一开始就会启动，把pppd搞起来，影响手动拨号，所以要先把rild改个名字，这样rild服务就不会起来了

**[plain]** [view plaincopy](http://blog.csdn.net/suntao222/article/details/8930744)

1. mv /system/bin/rild /system/bin/ril-d

改完后记得重启机器。好了，现在可以手动拨号了:

**[plain]** [view plaincopy](http://blog.csdn.net/suntao222/article/details/8930744)

1. shell@android:/ # pppd call unicom &
2. [1] 912
3. shell@android:/ # abort on (NO CARRIER)
4. abort on (NO DIALTONE)
5. abort on (ERROR)
6. abort on (NO ANSWER)
7. abort on (BUSY)
8. timeout set to 120 seconds
9. send (at^M)
10. expect (OK)
11. ^M
12. OK
13. -- got it
15. send (AT+CGDCONT=1,"IP","3gnet"^M)
16. expect (OK)
17. ^M
18. ^M
19. OK
20. -- got it
22. send (atdt\*99\*\*\*1#^M)
23. expect (CONNECT)
24. ^M
25. ^M
26. CONNECT
27. -- got it
29. shell@android:/ #

好了，到这里就算是拨号成功了，然后看ppp0接口是否建立。

**[plain]** [view plaincopy](http://blog.csdn.net/suntao222/article/details/8930744)

1. shell@android:/ # netcfg
2. ip6tnl0  DOWN                                   0.0.0.0/0   0x00000080 00:00:00:00:00:00
3. lo       UP                                   127.0.0.1/8   0x00000049 00:00:00:00:00:00
4. ppp0     UP                              10.105.219.114/32  0x000010d1 00:00:00:00:00:00
5. sit0     DOWN                                   0.0.0.0/0   0x00000080 00:00:00:00:00:00
6. shell@android:/ #

好了，ppp0接口也建立了，ip地址也获得了，ping一下baidu试试看。

**[plain]** [view plaincopy](http://blog.csdn.net/suntao222/article/details/8930744)

1. shell@android:/ # ping www.baidu.com
2. PING www.a.shifen.com (220.181.112.143) 56(84) bytes of data.
3. 64 bytes from 220.181.112.143: icmp\_seq=1 ttl=48 time=164 ms
4. 64 bytes from 220.181.112.143: icmp\_seq=2 ttl=48 time=172 ms
5. 64 bytes from 220.181.112.143: icmp\_seq=3 ttl=48 time=160 ms
6. ^C64 bytes from 220.181.112.143: icmp\_seq=4 ttl=48 time=148 ms
8. --- www.a.shifen.com ping statistics ---
9. 4 packets transmitted, 4 received, 0% packet loss, time 3005ms
10. rtt min/avg/max/mdev = 148.703/161.427/172.377/8.508 ms
11. shell@android:/ #

OK，一切正常，现在可以确定拨号没有问题了，剩下验证android上层ril是否工作正常了。

5.把rild的名字改回来，进入系统设置一下APN,

**[plain]** [view plaincopy](http://blog.csdn.net/suntao222/article/details/8930744)

1. APN:3gnet
2. MCC:460
3. MNC:01

保存，然后重启，起来后就可以在状态栏看到3G的图标了，当然要打开“启动数据网络”选项。  
如果没有工作，就"logcat -b radio &"看一下ril的log，ril是否认到ttyUSB的接口了。  
  
调试时候经常用到的几个小命令。  
netcfg : 查看当前所有网络接口  
getprop net.dns1 : 查看dns(有时候dns不对也会上不了网，比如8.8.8.8，这时候需要修改pppd)  
logcat -s pppd :　查看pppd log  
logcat -b radio &: 查看radio log  
cat ttyUSB0 &然后echo AT+CMD > ttyUSB0: 对modem发命令看是否有应答。

[3g移植](http://blog.csdn.net/duloveding/article/details/8287758)

分类： [LINUX](http://blog.csdn.net/duloveding/article/category/930738) [驱动](http://blog.csdn.net/duloveding/article/category/1061402)2012-12-12 18:21 20374人阅读 [评论](http://blog.csdn.net/duloveding/article/details/8287758#comments)(2) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/duloveding/article/details/8287758#report)

目录[(?)[+]](http://blog.csdn.net/duloveding/article/details/8287758)

**一 开发环境简介**

|  |  |
| --- | --- |
| 内容 | 说明 |
| 3G模块 | 华为EM820W（WCDMA） |
| 3G SIM卡 | 中国联通3G卡 |
| 嵌入式linux版本 | Linux-2.6.37 |
| 主机开发环境 | Ubutun 10.04 |

**二 主要步骤**

|  |  |
| --- | --- |
| 1 | 向linux内核中添加3G模块驱动（USB转串口的option驱动）和ppp网络协议的支持 |
| 2 | 交叉编译ppp-2.4.4源代码  将拨号上网所需的两个应用程序pppd和chat拷贝到开发板/usr/sbin文件夹下，  编写ppp拨号脚本wcdma、wcdma-chat-connect、wcdma-chat-disconnect并存放在/etc/ppp/peers目录下。将DNS写入到/etc/resolv.conf文件中 |
| 3 | 执行“pppd call wcdma”命令进行拨号 |

**三 配置3G模块驱动和ppp网络协议栈**

首先进入make menuconfig界面

**1 配置3G模块驱动**

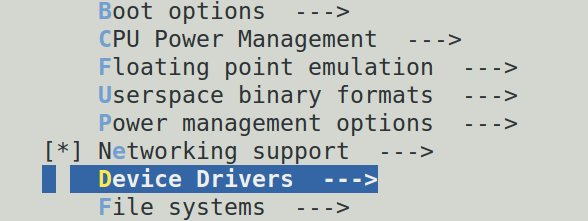


                     图 1

如图1首先选择Device Drivers

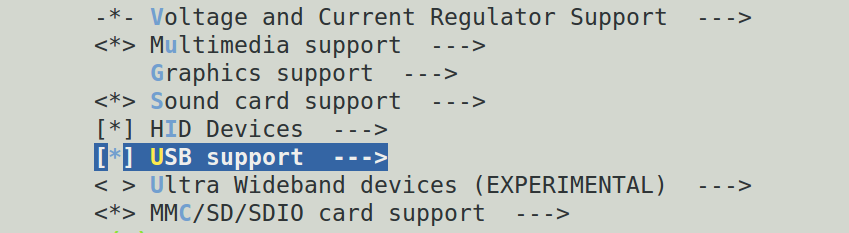


                     图2

如图2进入USB support

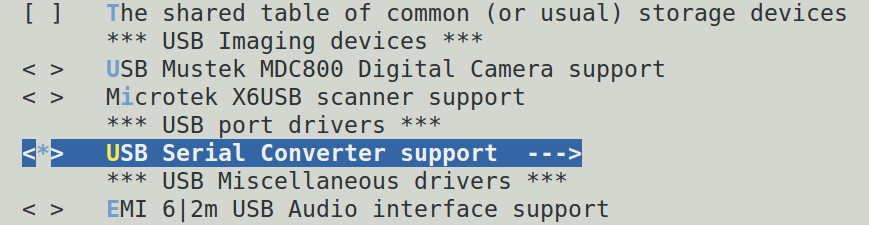


                      图 3

进入USB support之后一直往下直到找到图3并选择USB转串口

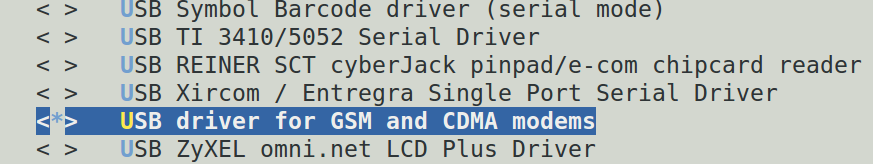


                      图4

最后选择USB driver for GSM and CDMA modems到此3G卡驱动已经添加完毕

主要是添加了driver/usb/serial/option.c和driver/usb/serial/usb\_wwan.c

两个文件，尽管驱动添加了，但是这仅仅只是一个usb转串口驱动，并不能作为网络设备直接使用，此时就需要ppp网络协议的支持，他将ttyUSB\*等设备模拟成网络设备使用也即(/dev/ttyUSB0<---->ppp0)

**2 添加ppp协议**

同样也是进入make menuconfig界面

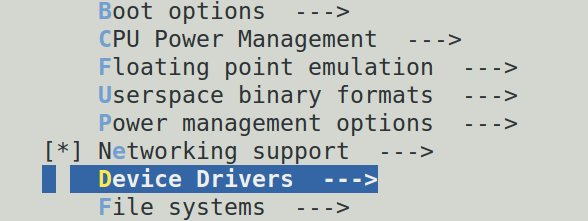


                         图5

首先同样也是进入驱动配置界面Device Drivers

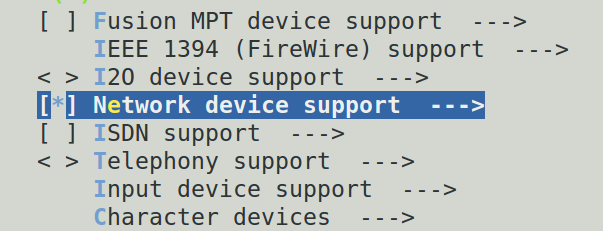


                         图6

如图6选择网络设备支持

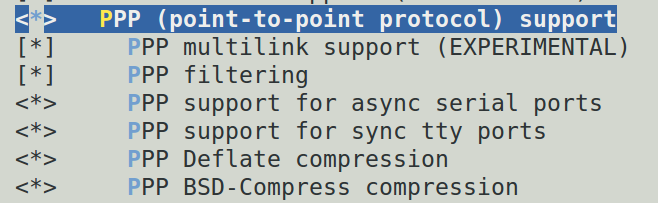


                         图7

如图7选择ppp

到这一步驱动和协议就都已经配置成功了接下来就是编译下载到板子

登入到linux, dmesg查看内核打印信息

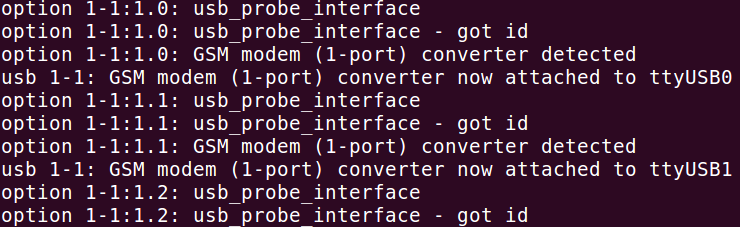


                         图8

同时查看/dev/目录下已经创建了

/dev/ttyUSB0,/dev/ttyUSB1,/dev/ttyUSB2,/dev/ttyUSB3,/dev/ttyUSB4,/dev/ttyUSB5

/dev/ppp等7个设备文件表明内核正确识别了EM820W3G模块

**四 编写ppp拨号脚本**

由于该拨号脚本需要使用chat,pppd,等命令，所以需要下载ppp-2.4.4

下载成功之后首先便是编译

**1 编译 ppp-2.4.4**

./configure

Make CC=arm-none-linux-gnueabi-gcc

编译成功后将编译生成的chat,pppd,pppdump等命令拷贝到板子文件系统的/usr/sbin目录下边去

**2 编写拨号脚本**

在板子文件系统/etc/ppp/peers目录下边创建三个文件如下

wcdma,chat-wcdma-connect,chat-wcdma-disconnect

wcdma配置文件内容

debug

nodetach

/dev/ttyUSB0

115200

usepeerdns

noauth

noipdefault

novj

novjccomp

noccp

defaultroute

ipcp-accept-local

ipcp-accept-remote

connect '/usr/sbin/chat -s -v -f /etc/ppp/peers/chat-wcdma-connect'

disconnect '/usr/sbin/chat -s -v -f/etc/ppp/peers/chat-wcdma-disconnect'

wcdma-chat-connect配置文件内容

TIMEOUT 5

ABORT 'NO CARRIER'

ABORT 'ERROR'

ABORT 'NODIALTONE'

ABORT 'BUSY'

ABORT 'NO ANSWER'

''  \rAT

OK  \rATZ

OK \rAT+CGDCONT=1,"IP","3GNET",,0,0

OK-AT-OK ATDT\*99#

CONNECT \d\c

wcdma-chat-disconnect配置文件内容

ABORT"ERROR"

ABORT "NODIALTONE"

SAY"\nSending break to the modem\n"

'' "\K"

''"+++ATH"

SAY"\nGoodbay\n"

创建好拨号脚本之后就是正式拨号了

**pppd call wcdma&**

 (加上&符号主要是因为拨号成功后，程序会一直执行不退出，所以采用后台运行的方式执行,假如有多个3G卡可能就会产生多个ttyUSB\*的设备文件，同时会需要多个wcdma文件此时可以采用pppd call wcdma1来为3G模块1进行ppp拨号)，pppd call wcdma是作用是建立协议，并生成网络接口ppp0

拨号成功会出现如下图所示打印

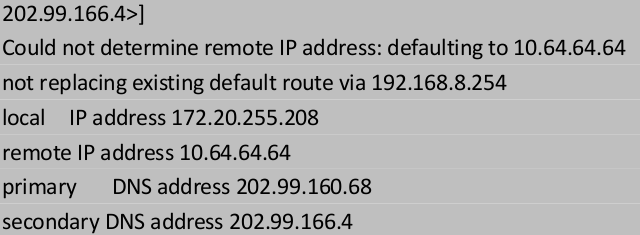


                                                        图9

出现上图9的打印后在/etc/resocv.conf中间添加DNS地址vi /etc/resolv.conf,加入内容：

nameserver 202.99.160.68

nameserver 202.99.166.4

使用ifconfig –a查看是否有ppp0的网络设备

最后测试ppp0网络设备是否能够使用

Ping –I ppp0 [www.baidu.com](http://www.baidu.com/)

能够ping通就表示3G卡移植驱动成功

到此阶段3G卡驱动已经移植完毕

**五 调试**

在移植该驱动时候出现过诸多问题

**1 pppd call wcdma调用失败**

root@cz:~# pppd call wcdma&

timeout set to 5 seconds

abort on (NO CARRIER)

abort on (ERROR)

abort on (NO DIALTONE)

abort on (BUSY)

abort on (NO ANSWER)

send (^MAT^M)

expect (OK)

AT^M^M

OK

 -- got it

send (^MATZ^M)

expect (OK)

^M

ATZ^M^M

OK

 -- got it

send(^MAT+CGDCONT=1,"IP","3GNET",,0,0^M)

expect (OK)

^M

AT+CGDCONT=1,"IP","3GNET",,0,0^M^M

ERROR

 -- failed

Failed (ERROR)

Connect script failed

调用AT+CGDCONT=1,"IP","3GNET",,0,0^M^M

这个指令失败，为了测试串口(命令发送通道)能否正常使用

在chat-wcdma-connect中间添加几个其他的AT命令

比如 OK \rAT+CGMI (厂家认证请求,返回模块厂家信息)

     OK \rAT+CGSN (查看产品 IMEI 序列号)

然后在调用pppd call wcdma&发现加上去的那几条AT命令都能正常执行

这说明很可能是EM820W模块没有正确识别SIM卡

后来我又添加了OK \rAT+CSQ指令(查看网络信号质量)发现会调用失败

得出初步结论凡是涉及到SIM卡相关的AT指令都会调用失败，相反其他都能成功执行，这说明板子与EM820W模块之间是能够正确交互数据的，

同时由于EM820W没能识别SIM卡导致相关指令执行失败

后来请教了相关人员，说我的测试方法不正确，我是利用pppd程序来判断某些命令是否能够正常执行，这样就不能排除pppd程序是否能够解析该命令所带来的干扰，所以改为使用一个更为纯粹的办法来进行测试了

Cat/dev/ttyUSB0&

echo “AT+CGSN”> /dev/ttyUSB0

echo “AT+CGMI”> /dev/ttyUSB0

echo “AT+CSQ”  > /dev/ttyUSB0

观察串口得到的response与pppd call wcdma一样

所以还是觉得EM820W没能正确识别SIM卡(不过以后测试还是需要选用正确的方法)

 测试SIM卡电路的5根信号线不管是否有SIM卡，5根线各自表现都是一样的，这说明却是没有识别SIM卡，而且SIM\_VCC和SIM\_RESET的时序一样，这和网上贴出的时序不一样如下图所示，很明显复位信号要比VCC慢半拍

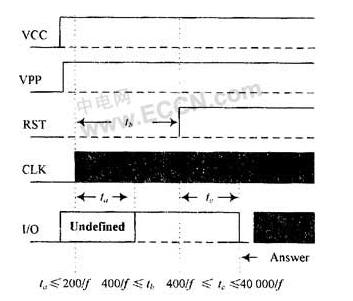


                                   图10

                            图10

硬件工程师此时修改了电路，在SIM\_RESET线上加了一个大电容，增加了些许延时

然后调用ppd call wcdma&

发现拨号成功了，同时使用ifconfig –a 命令出现了一个ppp0的网络设备

接着使用ping –I  ppp0 [www.baidu.com](http://www.baidu.com/)进行测试，发现网络不通

哎 问题挺多

**2 网络不通**

网络不通的问题从一开始就觉得肯定与卡没有关系了，很有可能是网络本身的问题，为了排除以太网口eth0的干扰我进行了如下测试

Ifconfig eth0 down

pppd call wcdma&

Ifconfig eth0 up

Ping –I [www.baidu.com](http://www.baidu.com/)

经过上面的折腾居然网络也通了，神奇。

上网查询说是双网卡造成的路由问题

果然在ppp0不通的时候route命令发现路由表如下

 # route  
     Kernel IP routing table  
        Destination    Gateway        Genmask           Flags Metric Ref    UseIface  
        10.64.64.64    \*                255.255.255.255  UH     0     0        0 ppp0  
        192.168.7.0    \*                255.255.255.0     U       0     0        0 eth0  
        default        192.168.7.1    0.0.0.0              UG     0     0        0 eth0

ppp0通的时候路由表如下

 # route  
       Kernel IProuting table  
      Destination    Gateway        Genmask           Flags Metric Ref    Use Iface  
      10.64.64.64      \*                255.255.255.255   UH   0      0        0ppp0  
      192.168.7.0      \*                255.255.255.0      U      0     0        0 eth0  
      default             \*                0.0.0.0             U      0     0        0 ppp0

由上面可以知道ppp0不通因为默认的路由是eth0的，这肯定不通啊

**六 注释**

