MEIG
股票代码:002881

全球领先的物联网终端及无线数据方案提供商

美格智能模块 Linux 适配指导

受控版本：V1.8

发布时间：2023 年 06 月 17 日

美格智能技术股份有限公司
MEIG SMART TECHNOLOGY CO.,LTD
地址：中国·深圳·宝安区福永街道凤凰第四工业区岭下路5号方格科技园
电话：86-755-61163458
传真：86-755-61163452
邮箱：m2msales@meigsmart.com
网址：www.meigsmart.com

重要声明

修订记录

版本号	日期	修订内容
V1.0	2020-08-13	初次建立
V1.1	2021-01-13	1. 添加 PCIe 拨号支持 2. 添加 GobiNet+AT IPv6 拨号支持
V1.3	2021-01-13	1. 添加 qmiwwan 拨号
V1.4	2021-11-11	1. 添加模块 subcls,protocol 方式适配说明 2. 添加最新所有模块信息 3. 更新 GobiNet 多路拨号 4. 添加 qmiwwan 多路拨号
V1.5	2022-03-08	添加展锐系列模组各种拨号支持
V1.6	2022-06-02	适配展锐 SRM811 NCM 拨号
V1.7	2022-06-23	1. 所有”美格模块“修改为“美格智能模块” 2. 删除表 1 中“SIM 卡热插拔”那一列
V1.8	2023-06-17	1. 添加 SLM770A option 驱动适配 2. 修改模块基本信息章节模块端口定义和子端口说明信息

目 录

表格索引

图片索引

1 引言

1.1 文档目的

本文档主要介绍针对美格智能模块基于 Linux 系统的适配指导说明。主要面向集成美格智能模块的相关开发调试人员，引导其快速适配美格智能模块到设备上，以设备提供数据，语音，短信等电信业务。

1.2 内容一览

本文共分为以下几部分：

第 1 章，主要介绍文档目的、章节描述等；

第 2 章，描述模块基本信息；

第 3 章，适配文件列表；

第 4 章，描述如何适配 usb 转串口驱动

第 5 章，描述如何使用 PPP 拨号

第 6 章，描述如何使用 ECM 拨号

第 7 章，描述如何使用 NCM 拨号

第 8 章，描述如何使用 Gobinet 单路拨号

第 9 章，描述如何使用 Gobinet 多路拨号

第 10 章，描述如何使用 Qmi_wwan 拨号

第 11 章，描述如何使用 MBIM 拨号

第 12 章，描述如何使用 RNDIS 拨号

第 13 章，描述如何启用 SIM 卡热插拔功能

第 14 章，描述 IPv6 的验证方法

第 15 章，描述常见问题的处理方法

第 16 章，附录,摘录常用 AT 指令说明

2 模块基本信息

本文介绍的模块都是通过 usb 与 Linux 上位机进行通信的， 并且使用复合设备驱动虚拟出多个子端口，各个端口实现不同的子功能。

2.1 模块端口定义

此文档适用于如下表格中所列出的模块，部分模块会有多种不同的 PID：

表 1 美格智能模块产品端口组合信息

美格智能模块产品端口组合信息				
VID	**PID**	**端口组合**	**系列**	**模块列表**
05C6	F601	DIAG, MODEM, AT, NMEA, ADB, [RMNET/ECM]		LTE: SLM750x/SLM730x
2DEE	4D22	DIAG, MODEM, AT, NMEA, ADB, RMNET	Q	LTE: SLM868x/SLM820x/MA800x 5G: SRM815x/SRM825x/SLM826x
2DEE	4D23	DIAG, MODEM, AT, NMEA, ADB, ECM		
2DEE	4D38	RNDIS, DIAG, MODEM, AT, NMEA, ADB		
2DEE	4D50	ECM, DIAG, MODEM, AT, LOG, ADB	U	5G: SRM811x/SRM821x/SRM810x
2DEE	4d51	RNDIS, DIAG, MODEM, AT, LOG, ADB		
2DEE	4d52	NCM, DIAG, MODEM, AT, LOG, ADB		
2DEE	4D57	RNDIS, DIAG, MODEM, AT, NMEA, [UAC]	A	LTE: SLM770x
2DEE	4D58	ECM, DIAG, MODEM, AT, NMEA, [UAC]		
2DEE	4D20	NCM, AT, DIAG, 3G DIAG, MODEM	H	LTE: SLM790x
		DIAG, AT, 3G DIAG, MODEM, ECM		

- 表中 VID(Vendor ID)、PID(Product ID)以及端口组合的顺序信息，在适配 usb 驱动时会用到。
- 表中“[]”括起来的部分表示可以通过 AT 命令动态开启或关闭。
- 一般情况下，除个别模块外，模块的 PID 与 USB 端口组合信息一一对应。如: 知道模块的 VID:2DEE, PID:4D22，就可以确认端口顺序是“DIAG, MODEM, AT, NMEA, ADB, RMNET”。

- 我们将模块分为 Q、U、A、H 四个系列，每个系列在适配时候的主要特性是一致的，后文中会有提到。

2.2 子端口说明

虚拟出来的各个子端口主要用来实现 AT 命令收发、网络通信、GPS、诊断等功能，详细见下表:

表 2 子端口说明

端口	功能说明
MODEM	用于 PPP 拨号
AT	用于收发 AT 命令
NMEA	上报 nmea 数据，用于 gps 功能
ADB	adb 调试端口,功能默认被禁用
RMNET	● 网口，仅在高通方案的模块上支持; ● 拨号后获取的是从运营商处分配的公网 IP;
ECM	● 网口，Linux 下免驱，Windows 不支持。 ● 一般获取的是局域网 IP，如 192.168.200.3； ● 占用 2 个端口,数据+控制；
NCM	● 网口 **Linux 平台上:** H 系列模块需要用美格提供的专用 ncm 驱动， 其他模块直接使用内核自带的 ncm 驱动即可。 **Windows 平台:** 所有模块均需要安装美格提供的驱动。 ● 拨号后获取的是从运营商处分配的公网 IP; ● **占用 2 个端口,数据+控制；**
RNDIS	● 网口，Linux/Windows 下都免驱。 ● 一般获取的是局域网 IP，如 192.168.200.3 ● **占用 2 个端口,数据+控制；**
DIAG、LOG、3G DIAG	获取模块日志，诊断问题使用
UAC	● 实现音频控制、传输功能，Linux/Windows 都免驱 ● **占用 3 个端口,数据收发+控制；**

2.3 拨号方式支持

表 3 拨号方式

方案	型号	PPP	ECM	NCM	Gobinet	Qmi_wwan	MBIM	RNDIS	多路

高通	SLM750/SLM730	Y	Y	N	Y	N	N	Y	Y
海思	SLM790	Y	Y	Y	N	N	N	N	N
高通	SLM868/SLM820/SLM828/SLM828G/SRM815EA/SRM815CN/SRM815W-EA/SRM815GL/SRM825W/SRM825W-EA/SRM825-CK/SRM825-EU/SRM815L/SRM825L/SRM825WN	Y	Y	N	Y	Y	Y	Y	Y
展锐	SRM811/SRM821	Y	Y	Y	N	N	Y	Y	Y
高通	SLT156/SLT152T	Y	N	N	N	N	N	N	N

3 适配文件列表

表 4 适配文件列表

文件	说明
ppp_script_for_linux.tar.gz	ppp 拨号脚本
udhcpc_script.tar.gz	udhcpc 脚本，在默认没有脚本的平台上可以使用

4 USB 转串口驱动适配

模块 usb 口是复用的，故需要使用 option 驱动分离出多个串口来使用。

4.1 内核配置

在内核配置文件中添加如下项,

```
CONFIG_USB_SERIAL_GENERIC=y
CONFIG_USB_SERIAL_OPTION=y
CONFIG_USB_SERIAL_QT2=y
```

备注：PC 上内核编译可在 make menuconfig 后，再将上述配置加入到.config 文件中

4.2 修改 option 驱动

Linux 上位机上串口适配有两种方式，一种按 Interface 顺序，此方法的劣势是固定不灵活，对于新增加的 interface 可能无法适配。另一种是按照 class info 方法，此方法是直接识别 interface 功能，建议使用，但需要模块软件支持，早期的高通方案模块不支持。

可以在 linux 上位机上使用如下命令来确认是否支持 class 信息：

```
#vid 为 2dee 的模块
 cat /sys/kernel/debug/usb/devices | grep -i 2dee -A 20 | grep -i Prot
#vid 为 05c6 的早期版本模块
 cat /sys/kernel/debug/usb/devices | grep -i 2dee -A 20 | grep -i Prot
```

Sub 和 Prot 都不为 ff 或者 00，表示可以支持 class 方式。

支持的情况：

```
I:* If#= 5 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=02 Prot=03 Driver=option
root@ubuntu-System-Product-Name:/home/ubuntu/linux-4.20/drivers/usb/serial# cat /sys/kernel/debug/usb/devices | grep -i 2dee -A 20 | grep -i Prot
I:* If#= 0 Alt= 0 #EPs= 1 Cls=02(comm.) Sub=06 Prot=00 Driver=cdc_ether
I:* If#= 1 Alt= 1 #EPs= 3 Cls=0a(data ) Sub=00 Prot=00 Driver=cdc_ether
I:* If#= 2 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=02 Prot=12 Driver=option
I:* If#= 3 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=02 Prot=14 Driver=option
I:* If#= 4 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=02 Prot=13 Driver=option
I:* If#= 5 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=02 Prot=03 Driver=option
root@ubuntu-System-Product-Name:/home/ubuntu/linux-4.20/drivers/usb/serial#
```

不支持的情况：

```
root@ubuntu-System-Product-Name:/home/ubuntu/linux-4.20/drivers/usb/serial# cat /sys/kernel/debug/usb/devices | grep -i 2dee -A 20 | grep -i Pr
ot
I:* If#= 0 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff Driver=(none)
I:* If#= 1 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=(none)
I:* If#= 2 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=(none)
I:* If#= 3 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=(none)
I:* If#= 4 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=42 Prot=01 Driver=usbfs
root@ubuntu-System-Product-Name:/home/ubuntu/linux-4.20/drivers/usb/serial#
```

4.2.1 Interface 方法

对于比较旧的模块，因为没有 usb 信息里没有 class, subcls, protocol 信息，需要按此方法进行适配。

在 option 驱动中添加模块信息，4.17 及以上版本内核修改方法:

```
drivers/usb/serial/option.c
@@ -85,6 +85,13 @@ static int  option_probe(struct usb_serial *serial,
 #define HUAWEI_PRODUCT_K3765                0x1465
 #define HUAWEI_PRODUCT_K4605                0x14C6
 #define HUAWEI_PRODUCT_E173S6               0x1C07

#define MEIG_VENDOR_ID                      0x2DEE
#define MEIG_QCM_VENDOR_ID                  0x05C6
#define MEIG_QCM_PRODUCT_Q                  0xF601
#define MEIG_PRODUCT_Q                      0x4D22
#define MEIG_PRODUCT_Q_ECM                  0x4D23
#define MEIG_PRODUCT_Q_RNDIS                0x4D38
#define MEIG_PRODUCT_H                      0x4D20
#define MEIG_PRODUCT_U_ECM                  0x4D50
#define MEIG_PRODUCT_U_RNDIS                0x4D51
#define MEIG_PRODUCT_U_NCM                  0x4D52
#define MEIG_PRODUCT_A_RNDIS                0x4D57
#define MEIG_PRODUCT_A_ECM                  0x4D58


 #define QUANTA_VENDOR_ID                   0x0408
 #define QUANTA_PRODUCT_Q101                0xEA02
@@ -564,6 +571,12 @@ static int  option_probe(struct usb_serial *serial,


 static const struct usb_device_id option_ids[] = {

       //H series
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x03) }, //3g app
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x13) }, //app
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x01) }, //modem
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x12) }, //at
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x14) }, //gprs

        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x03) }, //3g app
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x13) }, //app
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x01) }, //modem
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x12) }, //at
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x14) }, //gprs



       { USB_DEVICE(MEIG_VENDOR_ID,  MEIG_PRODUCT_Q),
       .driver_info = RSVD(4) | RSVD(5) | RSVD(6) | RSVD(7) },

       { USB_DEVICE(MEIG_VENDOR_ID,  MEIG_PRODUCT_Q_ECM),
       .driver_info = RSVD(4) | RSVD(5) | RSVD(6) | RSVD(7)},

       { USB_DEVICE(MEIG_QCM_VENDOR_ID,  MEIG_QCM_PRODUCT_Q),
       .driver_info = RSVD(4) | RSVD(5) | RSVD(6) | RSVD(7)},

       { USB_DEVICE(MEIG_VENDOR_ID,  MEIG_PRODUCT_Q_RNDIS),
       .driver_info = RSVD(0) | RSVD(1) | RSVD(6) | RSVD(7) | RSVD(8) | RSVD(9)},
```

```
        { USB_DEVICE(MEIG_VENDOR_ID,  MEIG_PRODUCT_U_RNDIS),
        .driver_info = RSVD(0) | RSVD(1) | RSVD(6) | RSVD(7) | RSVD(8) | RSVD(9)},

        { USB_DEVICE(MEIG_VENDOR_ID,  MEIG_PRODUCT_U_ECM),
        .driver_info = RSVD(0) | RSVD(1) | RSVD(6) | RSVD(7) | RSVD(8) | RSVD(9)},

        { USB_DEVICE(MEIG_VENDOR_ID,  MEIG_PRODUCT_U_NCM),
        .driver_info = RSVD(0) | RSVD(1) | RSVD(6) | RSVD(7) | RSVD(8) | RSVD(9)},

        { USB_DEVICE(MEIG_VENDOR_ID, MEIG_PRODUCT_A_RNDIS),
        .driver_info = RSVD(0)|RSVD(1) | RSVD(6) | RSVD(7) | RSVD(8) },

        { USB_DEVICE(MEIG_VENDOR_ID, MEIG_PRODUCT_A_ECM),
        .driver_info = RSVD(0)|RSVD(1)| RSVD(6) | RSVD(7) | RSVD(8) },


        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_COLT) },
```

4.17 以下版本内核修改方法：

```
kernel/drivers/usb/serial/option.c
@@ -86,12 +86,11 @@ static void option_instat_callback(struct urb *urb);
 #define HUAWEI_PRODUCT_K4605                    0x14C6
 #define HUAWEI_PRODUCT_E173S6                   0x1C07

#define MEIG_VENDOR_ID                          0x2DEE
#define MEIG_QCM_VENDOR_ID                      0x05C6
#define MEIG_QCM_PRODUCT_Q                      0xF601
#define MEIG_PRODUCT_Q                          0x4D22
#define MEIG_PRODUCT_Q_ECM                      0x4D23
#define MEIG_PRODUCT_Q_RNDIS                    0x4D38
#define MEIG_PRODUCT_H                          0x4D20
#define MEIG_PRODUCT_U_ECM                      0x4D50
#define MEIG_PRODUCT_U_RNDIS                    0x4D51
#define MEIG_PRODUCT_U_NCM                      0x4D52
#define MEIG_PRODUCT_A_RNDIS                    0x4D57
#define MEIG_PRODUCT_A_ECM                      0x4D58


#if (LINUX_VERSION_CODE < KERNEL_VERSION( 3,10,0 ))
#define USB_VENDOR_AND_INTERFACE_INFO(vend, cl, sc, pr) \
    .match_flags = USB_DEVICE_ID_MATCH_INT_INFO \
        | USB_DEVICE_ID_MATCH_VENDOR, \
    .idVendor = (vend), \
    .bInterfaceClass = (cl), \
    .bInterfaceSubClass = (sc), \
    .bInterfaceProtocol = (pr)

 #endif


@@ -701,13 +700,23 @@ static const struct option_blacklist_info yuga_clm920_nc5_blacklist
= {
        .reserved = BIT(1) | BIT(4),
 };


static const struct option_blacklist_info meig_u_blacklist = {
        .reserved = BIT(0) | BIT(1) | BIT(6) | BIT(7) | BIT(8) | BIT(9)
        };

static const struct option_blacklist_info meig_q_blacklist = {
        .reserved = BIT(4) | BIT(5) | BIT(6)|BIT(7),
        };

static const struct option_blacklist_info meig_q_rndis_blacklist = {
        .reserved = BIT(0) | BIT(1) | BIT(6)|BIT(7)||BIT(8),
        };

static const struct option_blacklist_info meig_a_blacklist = {
        .reserved = BIT(0) | BIT(1) | BIT(6)|BIT(7)||BIT(8),
        };
```

```
static const struct usb_device_id option_ids[] = {


       //H series
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x03) }, //3g app
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x13) }, //app
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x01) }, //modem
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x12) }, //at
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x14) }, //gprs

        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x03) }, //3g app
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x13) }, //app
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x01) }, //modem
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x12) }, //at
        { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x14) }, //gprs


       { USB_DEVICE(MEIG_QCM_VENDOR_ID, MEIG_QCM_PRODUCT_Q),
          .driver_info = (kernel_ulong_t)&meig_q_blacklist },

       { USB_DEVICE(MEIG_VENDOR_ID, MEIG_PRODUCT_Q),
          .driver_info = (kernel_ulong_t)&meig_q_blacklist },

       { USB_DEVICE(MEIG_VENDOR_ID, MEIG_PRODUCT_Q_ECM),
          .driver_info = (kernel_ulong_t)&meig_q_blacklist },

       { USB_DEVICE(MEIG_VENDOR_ID, MEIG_PRODUCT_Q_RNDIS),
          .driver_info = (kernel_ulong_t)&meig_q_rndis_blacklist },

       { USB_DEVICE(MEIG_VENDOR_ID, MEIG_PRODUCT_U_RNDIS),
          .driver_info = (kernel_ulong_t)&meig_u_blacklist },

       { USB_DEVICE(MEIG_VENDOR_ID, MEIG_PRODUCT_U_ECM),
          .driver_info = (kernel_ulong_t)&meig_u_blacklist },

       { USB_DEVICE(MEIG_VENDOR_ID, MEIG_PRODUCT_U_NCM),
          .driver_info = (kernel_ulong_t)&meig_u_blacklist },

       { USB_DEVICE(MEIG_VENDOR_ID, MEIG_PRODUCT_A_RNDIS),
          .driver_info = (kernel_ulong_t)&meig_a_blacklist },

       { USB_DEVICE(MEIG_VENDOR_ID, MEIG_PRODUCT_A_ECM),
          .driver_info = (kernel_ulong_t)&meig_a_blacklist },
```

4.2.2 Class Info 方法

此方法是使用 usb 的 class 信息来匹配模块各个接口功能，优点是与端口顺序无关，比较灵活。较早版本的高通方案模块不支持此方法。SRM811 展锐模组暂时不支持此方案，待后续板侧支持后更新此部分

适配方法如下：

```
--- a/drivers/usb/serial/option.c
+++ b/drivers/usb/serial/option.c
@@ -85,6 +85,13 @@ static int  option_probe(struct usb_serial *serial,
 #define HUAWEI_PRODUCT_K3765                0x1465
 #define HUAWEI_PRODUCT_K4605                0x14C6
 #define HUAWEI_PRODUCT_E173S6               0x1C07
+/*[MEIG-zhaopf-2021-11-09]add for meig modem supported {*/
+#define MEIG_VENDOR_ID                       0x2DEE
+/*[MEIG-zhaopf-2021-11-09]add for meig modem supported }*/

 #define QUANTA_VENDOR_ID                    0x0408
 #define QUANTA_PRODUCT_Q101                 0xEA02
@@ -564,6 +571,12 @@ static int  option_probe(struct usb_serial *serial,
#if (LINUX_VERSION_CODE < KERNEL_VERSION( 3,10,0 ))
+#define USB_VENDOR_AND_INTERFACE_INFO(vend, cl, sc, pr) \
+   .match_flags = USB_DEVICE_ID_MATCH_INT_INFO \
+       | USB_DEVICE_ID_MATCH_VENDOR, \
+   .idVendor = (vend), \
+   .bInterfaceClass = (cl), \
+   .bInterfaceSubClass = (sc), \
+   .bInterfaceProtocol = (pr)
+

+ #endif

 static const struct usb_device_id option_ids[] = {
+      /*[MEIG-zhaopf-2021-11-09]add for meig modem supported {*/

+       //SLM790
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x03) }, //3g app
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x13) }, //app
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x01) }, //modem
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x12) }, //at
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x03, 0x14) }, //gprs

+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x03) }, //3g app
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x13) }, //app
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x01) }, //modem
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x12) }, //at
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x02, 0x14) }, //gprs

+       //others
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x10, 0x01) }, //diag
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x10, 0x02) }, //modem
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x10, 0x03) }, //at
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x10, 0x04) }, //nmea
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x10, 0x07) }, //edl
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x10, 0x08) }, //log
+       { USB_VENDOR_AND_INTERFACE_INFO(MEIG_VENDOR_ID, 0xff, 0x10, 0x019 }, //dl
+       /*[MEIG-zhaopf-2021-11-09]add for meig modem supported }*/
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_COLT) },
```

3.10 以下内核要再打上如下补丁：

```
--- a/include/linux/usb.h
+++ b/include/linux/usb.h
@@ -861,6 +861,27 @@ static inline int usb_make_path(struct usb_device *dev, char *buf, size_t
size)
    .bInterfaceSubClass = (sc), \
    .bInterfaceProtocol = (pr)

+#define USB_VENDOR_AND_INTERFACE_INFO(vend, cl, sc, pr) \
+   .match_flags = USB_DEVICE_ID_MATCH_INT_INFO \
+       | USB_DEVICE_ID_MATCH_VENDOR, \
+   .idVendor = (vend), \
+   .bInterfaceClass = (cl), \
+   .bInterfaceSubClass = (sc), \
+   .bInterfaceProtocol = (pr)
+
 /* ------------------------------------------------------------------------ */
 /* Stuff for dynamic usb ids */
```

4.3 编译并加载驱动

也可以编译出 option.ko 在使用 insmod 命令加载。

驱动加载后，如插入 SRM815 模块时，设备目录会生成 4 个串口设备:

```
kvim:/ # ls -la /dev/ttyUSB*
crw-rw-r-- 1 radio radio 188,   0 2020-04-16 09:41 /dev/ttyUSB0
crw-rw-r-- 1 radio radio 188,   1 2020-04-16 09:41 /dev/ttyUSB1
crw-rw-r-- 1 radio radio 188,   2 2020-04-16 10:16 /dev/ttyUSB2
crw-rw-r-- 1 radio radio 188,   3 2020-04-16 09:41 /dev/ttyUSB3
```

图 1　串口信息

对于 SRM811 模组加载串口驱动后会有 5 个串口设备，如下图所示：

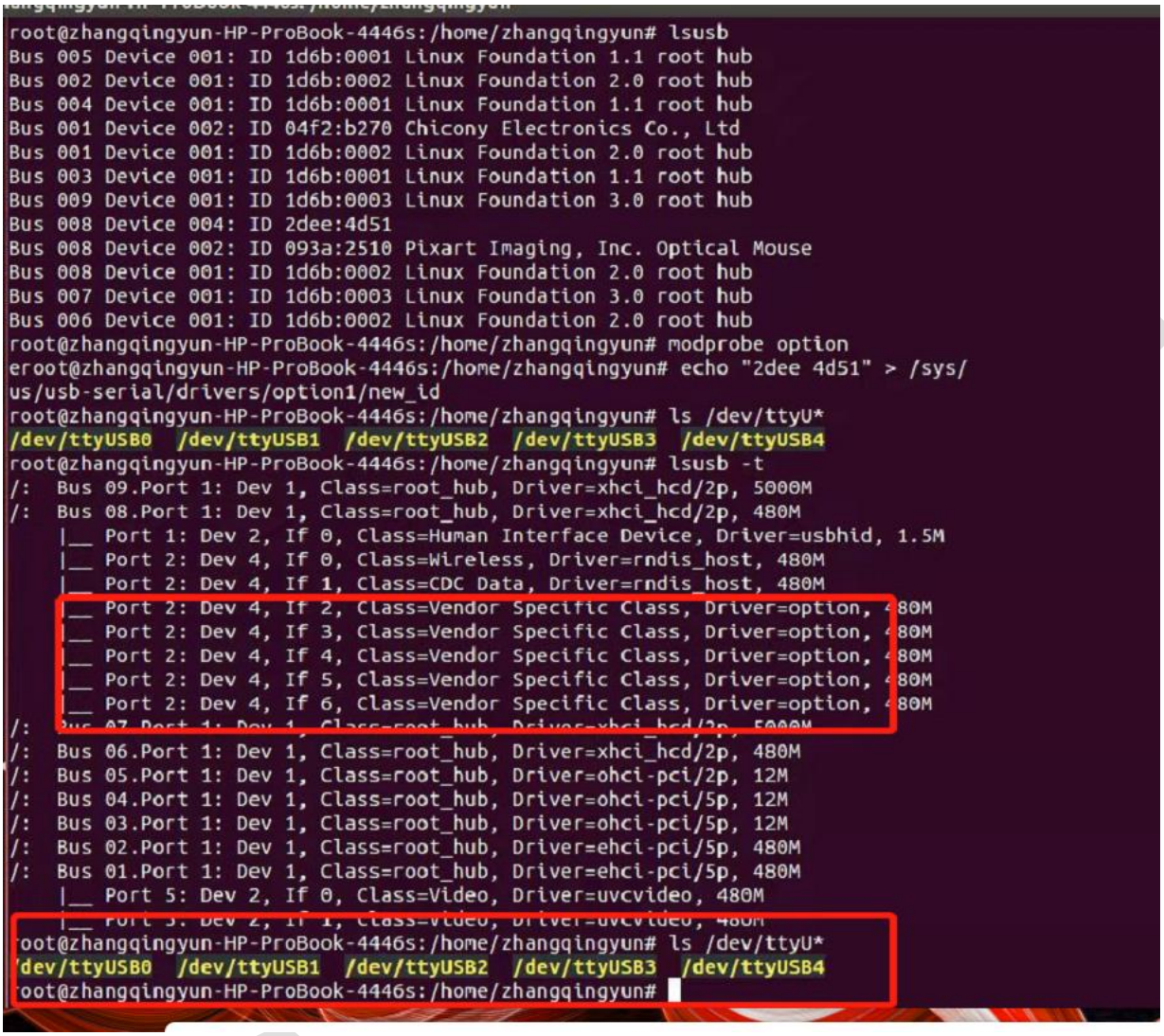


图 2 SRM811 串口

5 PPP 拨号

5.1 内核添加 PPP 驱动支持

在内核配置文件中添加 PPP 驱动，如:

```
+++ b/osdrv/opensource/kernel/linux-3.18.y/arch/arm/configs/hi3520dv400_full_defconfig
@@ -2439,4 +2439,10 @@ CONFIG_USB_SERIAL_OPTION=y
 CONFIG_USB_NET_CDCETHER=y
 CONFIG_USB_USBNET=y
 CONFIG_USB_NET_MEIG_CDC_NCM=y
+CONFIG_PPP=y
+CONFIG_PPP_FILTER=y
+CONFIG_PPP_MULTILINK=y
+CONFIG_PPP_ASYNC=y
+CONFIG_PPP_SYNC_TTY=y
+CONFIG_PPP_DEFLATE=y
```

5.2 拨号脚本准备

```
#将脚本ppp_script_for_linux.tar.gz加压到/etc/ppp目录下
tar xzvf ppp_script_for_linux.tar.gz -C /etc/
#给脚本加可执行权限
chmod 755 -R /etc/ppp
#修改端口
#修改文件/etc/ppp/peers/gprs-dial中端口号与实际一致
如: /dev/ttyUSB1
```

5.3 拨号

5.3.1 设置 APN

拨号前必须先设置 APN

一般情况下，国内运营商是不需要设置 APN 的，模块会根据 SIM 卡选择预置的 APN。

如：

中国移动—cmnet

中国电信—ctnet 或 ctlte

中国联通—3gnet

```
OK
at+cgdcont=1,"IPV4V6","cmnet"
OK
at+cgdcont?
+CGDCONT: 1,"IPV4V6","cmnet","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 2,"IPV4V6","ims","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 3,"IPV4V6","CMNET","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 38,"IPV6","v2x_ip","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 39,"IPV6","v2x_non_ip","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 4,"IPV4V6","CMWAP","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 5,"IPV4V6","SOS","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,1,,,,,,,,,"",,,,0

OK
at+cgdcont=1,"IPV4V6","ctnet"
OK
at+cgdcont?
+CGDCONT: 1,"IPV4V6","ctnet","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 2,"IPV4V6","ims","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 3,"IPV4V6","CMNET","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 38,"IPV6","v2x_ip","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 39,"IPV6","v2x_non_ip","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 4,"IPV4V6","CMWAP","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 5,"IPV4V6","SOS","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,1,,,,,,,,,"",,,,0

OK
```

图 3　APN 设置

如果默认 APN 不符合需求，可使用指令”AT+CGDCONT”来设置，如

5.3.2　进行拨号

拨号前先检查一下/etc/ppp/peers/gprs_dial 中端口是否正确，如与实际不一致则需修改后再使用

```
pppd call gprs_dial
```

```
Script /usr/sbin/chat -s -v -f /etc/ppp/ppp-on-dialer finished (pid 12784), status = 0x0
Serial connection established.
using channel 3
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB1
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x54b3e1ca> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x8 <asyncmap 0x0> <auth chap MD5> <magic 0xf7971c9e> <pcomp> <accomp>]
No auth is possible
sent [LCP ConfRej id=0x8 <auth chap MD5>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x54b3e1ca> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x9 <asyncmap 0x0> <magic 0xf7971c9e> <pcomp> <accomp>]
sent [LCP ConfAck id=0x9 <asyncmap 0x0> <magic 0xf7971c9e> <pcomp> <accomp>]
sent [CCP ConfReq id=0x1 <deflate 15> <deflate(old#) 15> <bsd v1 15>]
sent [IPCP ConfReq id=0x1 <compress VJ 0f 01> <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [LCP DiscReq id=0xa magic=0xf7971c9e]
rcvd [LCP ProtRej id=0xb 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
sent [IPCP ConfReq id=0x1 <compress VJ 0f 01> <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [IPCP ConfReq id=0x2]
sent [IPCP ConfNak id=0x2 <addr 0.0.0.0>]
rcvd [IPCP ConfRej id=0x1 <compress VJ 0f 01>]
sent [IPCP ConfReq id=0x2 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [IPCP ConfReq id=0x3]
sent [IPCP ConfAck id=0x3]
rcvd [IPCP ConfNak id=0x2 <addr 10.154.69.135> <ms-dns1 211.137.130.2> <ms-dns2 211.137.130.18>]
sent [IPCP ConfReq id=0x3 <addr 10.154.69.135> <ms-dns1 211.137.130.2> <ms-dns2 211.137.130.18>]
rcvd [IPCP ConfAck id=0x3 <addr 10.154.69.135> <ms-dns1 211.137.130.2> <ms-dns2 211.137.130.18>]
Could not determine remote IP address: defaulting to 10.64.64.64
not replacing existing default route via 192.168.147.2
local  IP address 10.154.69.135
remote IP address 10.64.64.64
primary   DNS address 211.137.130.2
secondary DNS address 211.137.130.18
Script /etc/ppp/ip-up started (pid 12797)
Script /etc/ppp/ip-up finished (pid 12797), status = 0x0
```

图 4　PPP 拨号

5.4 拨号验证

国内：

ping 114.114.114.114

ping www.baidu.com

国外:

ping 8.8.8.8

ping www.google.com

6 ECM 拨号

6.1 加载驱动

ECM 驱动一般 linux 内核默认都有加载。

如未加载,对于 linux PC 可按如下方式加载

```
modprobe usbnet
modprobe cdc_ether
```

对于嵌入式 linux 环境，可以在内核配置中开启 usbnet 和 ecm 开关

```
CONFIG_USB_USBNET=y
CONFIG_USB_NET_CDCETHER=y
```

6.2 拨号

一般情况下 ECM 版本模块默认是自动拨号的，类似即插即用，通常会生成名称未 ethX 或 usbX 的网口。

比如生成网卡时 usb0,可通过 ifconfig usb0 来查看是否获得 IP，如已获得，可直接 ping 验证.

对于某些嵌入式 linux 平台不能自动请求 dhcp 的情况，可以使用 udhcpc、dhclient、dhcpcd 等工具来获取并设置 ip 信息。如:

```
#注意udhcpc能否成功设置ip,网关，dns等信息，依赖于配置脚本，
默认路径：”/etc/udhcpc/default.script”。也可以通过-s参数指定脚本
udhcpc -i usb0 -s /etc/udhcpc/default.script
```

7 NCM 拨号

对于支持 NCM 拨号方式的模块，SRM811 需要使用内核自带的 ncm 驱动，其他产品需要编译加载 meig ncm 驱动。同时注意，如已加载 option 驱动，需要注意先按照”usb 转串口驱动适配章节”屏蔽模块的网络端口，否则会导致 ncm 驱动找不到口。

7.1 编译加载驱动

```
#解压驱动
tar xzvf MeiG_NCM_V0.5.1.tar.gz

#pc上编译
cd MeiG_NCM_V0.5.1
make

#加载驱动
insmod meig_cdc_driver.ko

#启用网卡
ifconfig usb0 up
```

对于嵌入式 linux 设备可按如下方法添加驱动，

将驱动文件 meig_cdc_driver.c 拷贝到驱动目录 drivers/net/usb/下,并按如下方法修改 Kconfig 和 Makefile,修改完成后编译并更新内核。

- drivers/net/usb/Kconfig

```
--- a/osdrv/opensource/kernel/linux-3.18.y/drivers/net/usb/Kconfig
+++ b/osdrv/opensource/kernel/linux-3.18.y/drivers/net/usb/Kconfig
@@ -262,6 +262,20 @@ config USB_NET_HUAWEI_CDC_NCM
             To compile this driver as a module, choose M here: the module will be
             called huawei_cdc_ncm.ko.

+config USB_NET_MEIG_CDC_NCM
+       tristate "Meig NCM embedded AT channel support"
+       depends on USB_USBNET
+       select USB_WDM
+       select USB_NET_CDC_NCM
+       help
+         This driver supports meige-style NCM devices, that use NCM as a
+         transport for other protocols, usually an embedded AT channel.
+         Good examples are:
+         * MEIG SLM790
+
+         To compile this driver as a module, choose M here: the module will be
+         called meig_cdc_driver.ko.
+
```

- drivers/net/usb/Makefile

```
--- a/osdrv/opensource/kernel/linux-3.18.y/drivers/net/usb/Makefile
+++ b/osdrv/opensource/kernel/linux-3.18.y/drivers/net/usb/Makefile
@@ -37,4 +37,4 @@ obj-$(CONFIG_USB_NET_HUAWEI_CDC_NCM)  += huawei_cdc_ncm.o
```

```
 obj-$(CONFIG_USB_VL600)             += lg-vl600.o
 obj-$(CONFIG_USB_NET_QMI_WWAN) += qmi_wwan.o
 obj-$(CONFIG_USB_NET_CDC_MBIM) += cdc_mbim.o
-
+obj-y  += meig_cdc_driver.o
```

驱动加载成功后，当插入模块时，会出现名称为 usbX 的网卡，一般是 usb0。

7.2 拨号

网卡生成成功后，需要使用 minicom 等串口工具发指令 AT^NDISDUP 来拨号，此时需要确保 usb 转串口驱动已经适配(drivers/usb/serial/option.c)，如:

```
#设置APN,移动卡为例:
AT+CGDCONT=1,"IPV4V6","cmnet"
#拨号
AT^NDISDUP=1,1
#断开拨号
AT^NDISDUP=1,0
```

```
at+cgdcont=1,"IP","cmnet"
OK
at^ndisdup=1,1
OK

^DATACONNECT

^NDISSTAT:1,,,"IPV4"
```

图 5 NDIS 拨号

拨号后一般平台上都会有 dhcp 客户端自动请求 IP 信息。如当前平台不支持，可手动使用 udhcpc、dhtool、dhcpcd、dhclient 等 dhcp 客户端来请求。

```
root@zhangqingyun:/home/zhangqingyun/MeiG_NCM_V0.5.1# udhcpc -i usb0
udhcpc (v1.21.1) started
Sending discover...
Sending select for 10.11.180.237...
Lease of 10.11.180.237 obtained, lease time 518400
/etc/udhcpc/default.script: Resetting default routes
SIOCDELRT: No such process
/etc/udhcpc/default.script: Adding DNS 211.137.130.2
/etc/udhcpc/default.script: Adding DNS 211.137.130.4
root@zhangqingyun:/home/zhangqingyun/MeiG_NCM_V0.5.1# ifconfig usb0
usb0      Link encap:Ethernet  HWaddr 00:1e:10:1f:00:01
          inet addr:10.11.180.237  Bcast:10.11.180.239  Mask:255.255.255.252
          inet6 addr: fe80::21e:10ff:fe1f:1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:96 errors:0 dropped:0 overruns:0 frame:0
          TX packets:362 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10416 (10.4 KB)  TX bytes:66656 (66.6 KB)

root@zhangqingyun:/home/zhangqingyun/MeiG_NCM_V0.5.1# ping -I usb0 www.baidu.com
PING www.a.shifen.com (36.152.44.96) from 10.11.180.237 usb0: 56(84) bytes of data.
64 bytes from 36.152.44.96: icmp_seq=1 ttl=55 time=47.5 ms
64 bytes from 36.152.44.96: icmp_seq=2 ttl=55 time=43.8 ms
```

图 6 网络连通验证

8 GOBINET(单路)拨号

GoBiNet 拨号仅仅适用于高通系列的模组，非高通平台模组不支持此种拨号，对于高通平台模组，调试如下

8.1 加载驱动

解压驱动

```
tar xzvf Meig_GobiNet_Driver_V1.3.2.tar.gz
```

对于 PC 环境，在驱动目录下执行脚本，可加载好所有驱动

```
source go_gobi.sh
```

对于嵌入式 linux 环境，需要交叉编译

```
#编译驱动
#make -C [内核路径] M=[Gobinet驱动所在绝对路径] CROSS_COMPILE=[交叉编译工具前缀] modules
make  -C  /home/zhaopf/work/linux-4.14.148  M=/home/zhaopf/work/release/GobiNet  modules
CROSS_COMPILE=aarch64-linux-android-
#生成驱动GobiNet.ko
#在对应平台上加载即可
insmod GobiNet.ko

#编译拨号工具
cd meig-cm
make CROSS_COMPILE=aarch64-linux-android-
#生成拨号工具meig-cm，将其拷如机器里
```

8.2 拨号验证

8.2.1 使用 CM 拨号

执行编译出来的 meig-cm 来拨号

```
#参数说明:
#-s 指定apn名称
#-6 支持ipv4v6双栈
#-i 指定网卡名称，针对部分网卡名称被修改的情况
#如拨移动卡:
meig-cm -s cmnet
#如拨电信卡:
meig-cm -s ctnet
```

```
#如拨联通卡：
meig-cm –s 3gnet
```

8.2.2 使用 AT 拨号

使用 AT 拨号依赖于 dhcp 客户端，对于 ubuntu 环境可使用如下命令安装,，如：

sudo apt-get install udhcpc

单 IPv4 拨号:

```
#启用网卡
ifconfig <网卡名称> up
#AT口下发
AT$QCRMCALL=1,1,1
#立即请求dhcp
udhcpc -i <网卡名称>
```

单 IPv6 拨号:

```
#启用网卡
ifconfig <网卡名称> down
#AT口下发
AT$QCRMCALL=1,1,2
#启用网卡，自动拿到无状态V6地址
ifconfig <网卡名称> up
```

双栈拨号:

```
#停用网卡
ifconfig <网卡名称> down
#AT口下发
AT$QCRMCALL=1,1,3
#启用网卡并立即请求dhcp
ffconfig <网卡名称> up
udhcpc -i <网卡名称>
```

注意：ubuntu 系统上因为有 NetworkManager 服务，可以不用手动去发 DHCP 请求。直接 down/up 网卡，由系统去自己去发 DHCP 请求。如：

```
#停用网卡
Ifconfig <网卡名称> down
#AT口下发
AT$QCRMCALL=1,1,3
#启用网卡
ifconfig <网卡名称> up
```

8.2.3 GobiNet 网卡名称修改

修改驱动中的.flags 标志可以定制不同的网卡名称，具体如下

FLAG_WWAN，	“wwanN”
FLAG_POINTTOPOINT	“usbN”
FLAG_ETHER	“ethN”
FLAG_WLAN	“wlanN”

Flags 的位置在 GobiUSBNet.c 中如下结构体里，

```
1904 /*===========================================================================*/
1905 // Struct driver_info
1906 /*===========================================================================*/
1907 static struct driver_info GobiNetInfo = {
1908     .description   = "Meig GobiNet Ethernet Device",
1909 #ifdef CONFIG_ANDROID
1910     .flags         =FLAG_WWAN, //FLAG_POINTTOPOINT, //usb0
1911 #else
1912     .flags         =  FLAG_WWAN, //FLAG_POINTTOPOINT,
1913 #endif
1914     .bind          = GobiNetDriverBind,
1915     .unbind        = GobiNetDriverUnbind,
```

2.6.33 版本以下的 Linux 内核默认无法支持 FLAG_WWAN,需要打上如下补丁

```
--- a/drivers/net/usb/usbnet.c
+++ b/drivers/net/usb/usbnet.c
@@ -1295,6 +1295,9 @@ usbnet_probe (struct usb_interface *udev, const struct usb_device_id *prod)
                        /* WLAN devices should always be named "wlan%d" */
                        if ((dev->driver_info->flags & FLAG_WLAN) != 0)
                                strcpy(net->name, "wlan%d");
+                       /* WWAN devices should always be named "wwan%d" */
+                       if ((dev->driver_info->flags & FLAG_WWAN) != 0)
+                               strcpy(net->name, "wwan%d");

                        /* maybe the remote can't receive an Ethernet MTU */
                        if (net->mtu > (dev->hard_mtu - net->hard_header_len))
diff --git a/include/linux/usb/usbnet.h b/include/linux/usb/usbnet.h
index f814730..86c31b7 100644
--- a/include/linux/usb/usbnet.h
+++ b/include/linux/usb/usbnet.h
@@ -90,6 +90,7 @@ struct driver_info {
 #define FLAG_WLAN       0x0080          /* use "wlan%d" names */
 #define FLAG_AVOID_UNLINK_URBS 0x0100   /* don't unlink urbs at usbnet_stop() */
 #define FLAG_SEND_ZLP   0x0200          /* hw requires ZLPs are sent */
+#define FLAG_WWAN       0x0400          /* use "wwan%d" names */
```

9 GOBINET(多路)拨号

GobiNet 多路拨号同样仅仅适用于高通系列模组，非高通平台不支持。对于高通平台适配如下：

9.1 AT 方式

9.1.1 加载驱动

解压驱动

```
tar xzvf GobiNet_v1.4.3.tar.gz
```

对于 PC 环境，在驱动目录下执行脚本，可加载好所有驱动、准备好拨号环境，

```
source go_gobi.sh
```

对于嵌入式 linux 环境，需要交叉编译，

```
#编译驱动
#make -C [内核路径] M=[Gobinet驱动所在绝对路径] CROSS_COMPILE=[交叉编译工具前缀] modules
make   -C   /home/xxx/work/linux-4.14.148   M=/home/zhaopf/work/release/GobiNet   modules
CROSS_COMPILE=aarch64-linux-android-
#生成驱动GobiNet.ko
#在对应平台上加载即可
insmod GobiNet.ko

#启动dhcp客户端，需要在拨号后立即发起dhcp请求才能拨号成功，故可以预先启动后台进程，需要几路起几路，最多可以起4路，如：
#第1路
ifconfig bmwan0 up
udhcpc -f -t 0 -i bmwan0 -x hostname:test-C -O 121 > /dev/null &

#第2路
ifconfig bmwan1 up
udhcpc -f -t 0 -i bmwan1 -x hostname:test-C -O 121 > /dev/null &

#第3路
ifconfig bmwan2 up
udhcpc -f -t 0 -i bmwan2 -x hostname:test-C -O 121 > /dev/null &
#第4路
ifconfig bmwan3 up
udhcpc -f -t 0 -i bmwan3 -x hostname:test-C -O 121 > /dev/null &
```

9.2 拨号验证

多路拨号需要使用 minicom 等串口工具通过指令 AT+CGDCONT 先设置每路 APN，再使用 AT$QCRMCALL 来拨号。两个指令的详细使用方法参见附录。

如使用移动卡拨号 4 路的情况：

设置 APN，

```
at+cgdcont=1,"IP","cmnet"
OK
at+cgdcont=3,"IP","APN2"
OK
at+cgdcont=4,"IP","APN3"
OK
at+cgdcont=5,"IP","APN4"
OK
at+cgdcont?
+CGDCONT: 1,"IP","cmnet","0.0.0.0",0,0,0,0
+CGDCONT: 2,"IPV4V6","IMS","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0
+CGDCONT: 3,"IP","APN2","0.0.0.0",0,0,0,0
+CGDCONT: 4,"IP","APN3","0.0.0.0",0,0,0,0
+CGDCONT: 5,"IP","APN4","0.0.0.0",0,0,0,0
```

图 7　多路 APN 设置

拨号，注意:qcrmcall 的第 2 个参数 Instance 为实际网卡 ID+1,第 5 个参数为 profile number 对应 APN ID。

如 AT$QCRMCALL=X,2,X,X,1 使用的是 ID 为 1 的 APN，网口为 bmwan3.

```
at$qcrmcall=1,1,1,2,1
$QCRMCALL: 1, V4

OK
at$qcrmcall=1,2,1,2,3
$QCRMCALL: 2, V4

OK
at$qcrmcall=1,3,1,2,4
$QCRMCALL: 3, V4

OK
at$qcrmcall=1,4,1,2,5
$QCRMCALL: 4, V4

OK
```

图 8　多路拨号

9.3 QMI 方式

qmi 拨号方式仅仅适用于高通平台模组，非高通平台不支持，对于高通平台适配如下：

9.3.1 加载驱动

```
#qmap参数对应需要虚拟出来的网卡数量，GobiNet驱动默认最多4路，可以修改
insmod GobiNet.ko qmap_mode=4
```

加载驱动且检测到模块后，会生成网卡 usb0.1, usb0.2, usb0.3....

```
usb0.1    Link encap:Ethernet  HWaddr 22:17:36:73:63:24
          inet addr:10.8.18.230  Bcast:10.8.18.231  Mask:255.255.255.252
          inet6 addr: fe80::2017:36ff:fe73:6324/64 Scope:Link
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1400  Metric:1
          RX packets:45 errors:0 dropped:0 overruns:0 frame:0
          TX packets:119 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6052 (6.0 KB)  TX bytes:11528 (11.5 KB)

usb0.2    Link encap:Ethernet  HWaddr 22:17:36:73:63:24
          inet6 addr: fe80::2017:36ff:fe73:6324/64 Scope:Link
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1400  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:40 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:612 (612.0 B)  TX bytes:4407 (4.4 KB)

usb0.3    Link encap:Ethernet  HWaddr 22:17:36:73:63:24
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

usb0.4    Link encap:Ethernet  HWaddr 22:17:36:73:63:24
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:25 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:7180 (7.1 KB)
```

9.3.2 拨号验证

拨号时需要指定网卡名称和 channelID(pdn id),如

```
root@ubuntu-System-Product-Name:/home/zhaopf/work/1123/Meig_GobiNet_Driver_V1.4.2/meig-cm#
root@ubuntu-System-Product-Name:/home/zhaopf/work/1123/Meig_GobiNet_Driver_V1.4.2/meig-cm# ./meig-cm -s cmnet -4 -i usb0 -n 1 &
[1] 13350
root@ubuntu-System-Product-Name:/home/zhaopf/work/1123/Meig_GobiNet_Driver_V1.4.2/meig-cm# [11-24_10:20:35:687] Meig_QConnectManager_Linux_V1.4.
3
[11-24_10:20:35:688] Find qmichannel = /dev/qcqmi0
[11-24_10:20:35:688] net_path=/sys/class/net/usb0

[11-24_10:20:35:688] use fixed adapter
[11-24_10:20:35:688] use fixed adapter
[11-24_10:20:35:688] qmap_mode = 4, muxid = 0x81, qmap_netcard = usb0.1
[11-24_10:20:35:688] Modem works in QMI mode
[11-24_10:20:35:688] qmap_mode = 4, muxid = 0x81, qmap_netcard = usb0.1
[11-24_10:20:35:688] [zpf]qmap_mode=4
[11-24_10:20:35:724] Get clientWDS = 7
[11-24_10:20:35:756] Get clientDMS = 8
[11-24_10:20:35:788] Get clientNAS = 9
[11-24_10:20:35:820] Get clientUIM = 10
[11-24_10:20:35:852] requestBaseBandVersion SRM815_6.0.2_EQ102  1  [Aug 17 2021 06:00:00]
[11-24_10:20:35:980] requestGetSIMStatus SIMStatus: SIM_READY
[11-24_10:20:35:980] requestSetProfile[1] cmnet///0
[11-24_10:20:36:045] requestGetProfile[1] cmnet///0
[11-24_10:20:36:076] requestRegistrationState2 SrvStatus = 0, TrueSrvsStatus = 0
[11-24_10:20:36:076] requestRegistrationState2 MCC: 460, MNC: 1, PS: Attached, DataCap: LTE
[11-24_10:20:36:108] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[11-24_10:20:36:108] ifconfig usb0.1 down
[11-24_10:20:36:114] ifconfig usb0.1 0.0.0.0
[11-24_10:20:36:172] requestRegistrationState2 SrvStatus = 0, TrueSrvsStatus = 0
[11-24_10:20:36:172] requestRegistrationState2 MCC: 460, MNC: 1, PS: Attached, DataCap: LTE
[11-24_10:20:36:492] requestSetupDataCall WdsConnectionIPv4Handle: 0xf20829c0
[11-24_10:20:36:588] requestQueryDataCall IPv4ConnectionStatus: CONNECTED
```

移动卡 4 路拨号命令示例：

```
./meig-cm -s cmnet -4 -i usb0 -n 1
./meig-cm -s cmnet2 -4 -i usb0 -n 2
./meig-cm -s cmnet3 -4 -i usb0 -n 3
./meig-cm -s cmnet4 -4 -i usb0 -n 4
```

10 QMI_WWAN 拨号

适配前可以先检查下驱动中是否存在文件 drivers/net/usb/qmi_wwan.c，如果存在则说明支持 qmi_wwan 拨号，部分低版本内核是不支持。Qmi_wwan 仅仅适用于高通平台，非高通平台不支持。对于高通平台适配如下：

10.1 添加内核配置项

```
CONFIG_USB_WDM=y
CONFIG_USB_NET_DRIVERS=y
#如果想编译成模块方式，可设置为CONFIG_USB_NET_QMI_WWAN=m
CONFIG_USB_NET_QMI_WWAN=y
```

10.2 驱动中添加美格智能模块

在数组 static const struct usb_device_id products[]末尾追加

```
--- a/drivers/net/usb/qmi_wwan.c
+++ b/drivers/net/usb/qmi_wwan.c

+#ifndef USB_VENDOR_AND_INTERFACE_INFO
+#define USB_VENDOR_AND_INTERFACE_INFO(vend, cl, sc, pr) \
+   .match_flags = USB_DEVICE_ID_MATCH_INT_INFO \
+       | USB_DEVICE_ID_MATCH_VENDOR, \
+   .idVendor = (vend), \
+   .bInterfaceClass = (cl), \
+   .bInterfaceSubClass = (sc), \
+   .bInterfaceProtocol = (pr)
+

+ #endif

@@ -1351,7 +1351,8 @@ static int qmi_wwan_resume(struct usb_interface *intf)
       {QMI_GOBI_DEVICE(0x1199, 0x901b)},     /* Sierra Wireless MC7770 */
       {QMI_GOBI_DEVICE(0x12d1, 0x14f1)},     /* Sony Gobi 3000 Composite */
       {QMI_GOBI_DEVICE(0x1410, 0xa021)},      /* Foxconn Gobi 3000 Modem device (Novatel
E396) */

+      {QMI_FIXED_INTF(0x05c6, 0xf601, 5)},    /* Meig SLM868 */
+      {QMI_FIXED_INTF(0x2dee, 0x4d22, 5)},    /* Meig SRM815 */
+      {USB_VENDOR_AND_INTERFACE_INFO(0x2dee, 0xff, 0x10, 0x05)}, //rmnet
       { }                                     /* END */
```

```
};
MODULE_DEVICE_TABLE(usb, products);
```

10.3 编译拨号工具

qmi_wwan 需要使用 meig-cm 工具来拨号。Meig-cm 编译方法如下：

```
#编译拨号工具
cd meig-cm

#pc编译方法
Make

#交叉编译方法
make CROSS_COMPILE=aarch64-linux-android-

#生成拨号工具meig-cm，将其拷如机器里
```

插入模块后，将生成名称未 wwan0 的网卡。

10.4 拨号

执行编译出来的 meig-cm 来拨号

```
#参数说明:
#-s 指定apn名称
#-6 支持ipv4v6双栈
#-i 指定网卡名称，针对部分网卡名称被修改的情况
#如拨移动卡:
meig-cm -s cmnet

#如拨电信卡:
meig-cm -s ctnet

#如拨联通卡:
meig-cm -s 3gnet
```

11 MBIM 拨号

11.1 添加内核配置项

```
CONFIG_USB_NET_DRIVERS=y
CONFIG_USB_NET_CDC_NCM=y
#如果想编译成模块方式，可设置为CONFIG_USB_NET_CDC_MBIM=m
CONFIG_USB_NET_CDC_MBIM=y
```

修改以上配置项后，内核将默认支持 mbim 驱动。

11.2 拨号

目前 windows 和 ubuntu18 及以上版本 mbim 是免驱的，直接在网络连接可以使能。

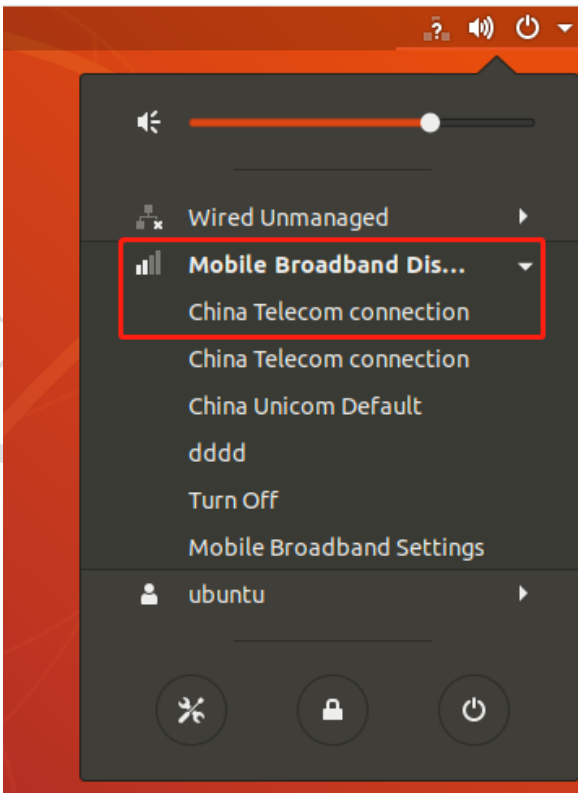


图 9 MBIM 拨号

12 RNDIS 拨号

RNDIS 拨号与 ECM 拨号类似，一般都是自动拨号方式.

12.1 添加内核配置项

ECM 驱动一般 linux 内核默认都有加载。

如未加载,对于 linux PC 可按如下方式加载，

```
modprobe usbnet
modprobe cdc_ether
modprobe rndis_host
```

对于嵌入式 linux 环境，可以在内核配置中开启以下开关,编译并更新内核后验证。

```
CONFIG_USB_USBNET=y
CONFIG_USB_NET_CDCETHER=y
CONFIG_USB_NET_RNDIS_HOST=y
```

12.2 拨号

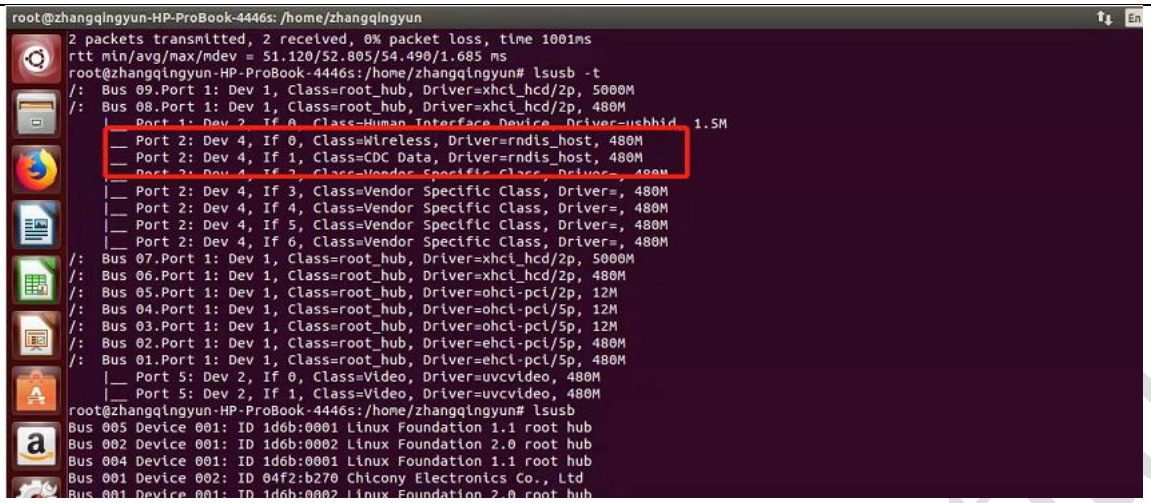
一般情况下 RNDIS 版本模块默认是自动拨号的，类似即插即用，通常会生成名称为 usbX 的网口。

比如生成网卡是 usb0,可通过 ifconfig usb0 来查看是否获得 IP，如已获得，可直接 ping 验证.

对于某些嵌入式 linux 平台不能自动请求 dhcp 的情况，可以使用 udhcpc、dhclient、dhcpcd 等工具来获取并设置 ip 信息。如:

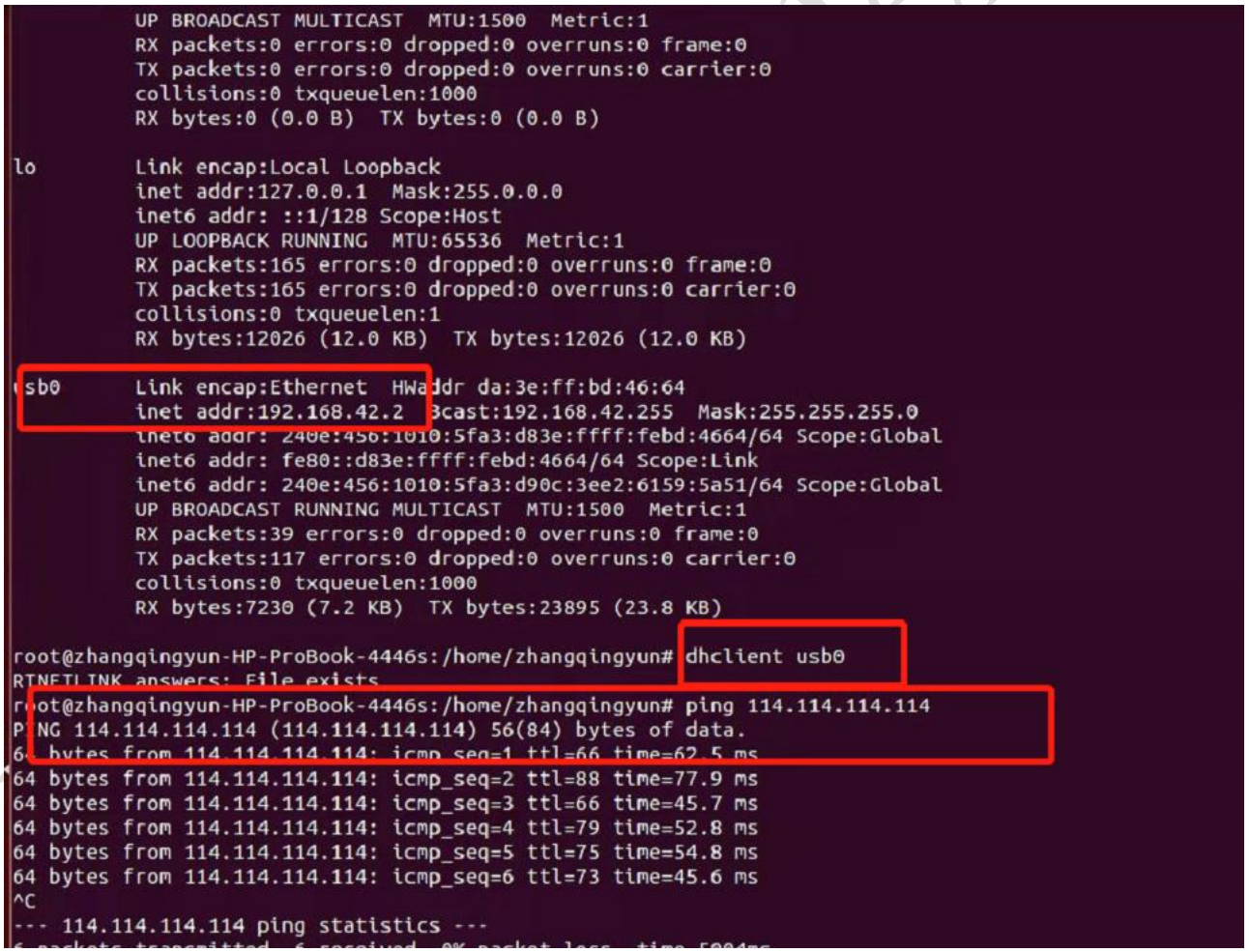
```
#注意udhcpc能否成功设置ip,网关，dns等信息，依赖于配置脚本，
默认路径：”/etc/udhcpc/default.script”。也可以通过-s参数指定脚本

udhcpc -i usb0 -s /etc/udhcpc/default.script
```

SRM811(展锐系列)默认是 RNDIS 拨号，一般来说，linux 内核默认都支持 rndis。如不支持请参考 12.1 进行内核配置。当模组插到 linux 系统上，驱动会自动加载，usb0 网卡会自动生成。如下图所示：



驱动加载后，正常会有 usb0 网卡出来，rndis 一般是自动拨号，上位机只用 dhcp 获取 ip 地址即可。

如下所示：



13 PCIE 拨号

对于嵌入式平台，需要使用交叉编译工具链和内核来编译生成 mhi 驱动(pcie_mhi.ko)，用于生成网卡设备。

13.1 准备并加载驱动

拨号前先使用 lspci 查看是否有检测到模块的 vid 和 pid 信息，如果没有则需要检查模块是否连接好。

如：

```
~ # busybox lspci
03:00.0 Class ff00: 17cb:0306
```

加载 mhi 驱动

```
~ # insmod pcie_mhi.ko
#将会生成如下设备节点
~ # ls /dev/mhi_*
/dev/mhi_BHI      /dev/mhi_DIAG     /dev/mhi_DUN      /dev/mhi_LOOPBACK  /dev/mhi_MBIM
```

13.2 拨号

```
单IPv4拨号
~ #./meig-cm -d /dev/mhi_MBIM
#IPv4v6双栈拨号
~ #./meig-cm -d /dev/mhi_MBIM -4 -6
拨号成功后生成的网卡是mhi0，可使用ping来验证
ping -I mhi0 www.baidu.com
```

14 SIM 卡热插拔支持

对于支持 SIM 卡热插拔的模块，可以使用如下 AT 指令来启用。

注意：设置后重启模块才生效

```
#启用SIM卡热插拔，检测脚低电平有效
AT+MGCFG=2,1,0

#启用SIM卡热插拔，检测脚高电平有效
AT+MGCFG=2,1,1

#停用SIM卡热插拔
AT+MGCFG=2,0,0
```

注意：SLM750(1.0 版本)/SLM730/SLM868/SLM790 和高通 NDIS 方案设备不支持 SIM 卡热插拔功能

15 IPV6 功能验证

目前所有模块基本都支持 IPv6 功能，实际使用时可以与 FAE 确认。

15.1 IPv6 连通性验证

可以使用 ping6 命令 ping IPv6 地址来验证，已知如下地址可用:

```
北京邮电大学DNS服务器
2001:da8:202:10::36
2001:da8:202:10::37

北京科技大学DNS服务器
2001:da8:208:10::6
```

```
root@56iqDS:/etc # ifconfig ppp0
ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.187.213.130  P-t-P:10.64.64.64  Mask:255.255.255.255
          inet6 addr: 240e:bf:d427:f0df:acb1:4d03:d9fc:c534/64 Scope: Global
          inet6 addr: fe80::acb1:4d03:d9fc:c534/10 Scope: Link
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1280  Metric:1
          RX packets:47 errors:0 dropped:0 overruns:0 frame:0
          TX packets:95 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:4113 TX bytes:6416

root@56iqDS:/etc # ping6 2001:da8:202:10::36
PING 2001:da8:202:10::36(2001:da8:202:10::36) 56 data bytes
64 bytes from 2001:da8:202:10::36: icmp_seq=1 ttl=46 time=200 ms
64 bytes from 2001:da8:202:10::36: icmp_seq=2 ttl=46 time=98.7 ms
64 bytes from 2001:da8:202:10::36: icmp_seq=3 ttl=46 time=86.6 ms
^C
--- 2001:da8:202:10::36 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 86.621/128.648/200.561/51.091 ms
root@56iqDS:/etc #
```

图 10 IPv6 ping

15.2 IPv6 功能测试

在浏览器中访问地址 http://www.test-ipv6.com/，可以验证 IPv6 支持情况。

图 11 IPv6 连接测试-概述

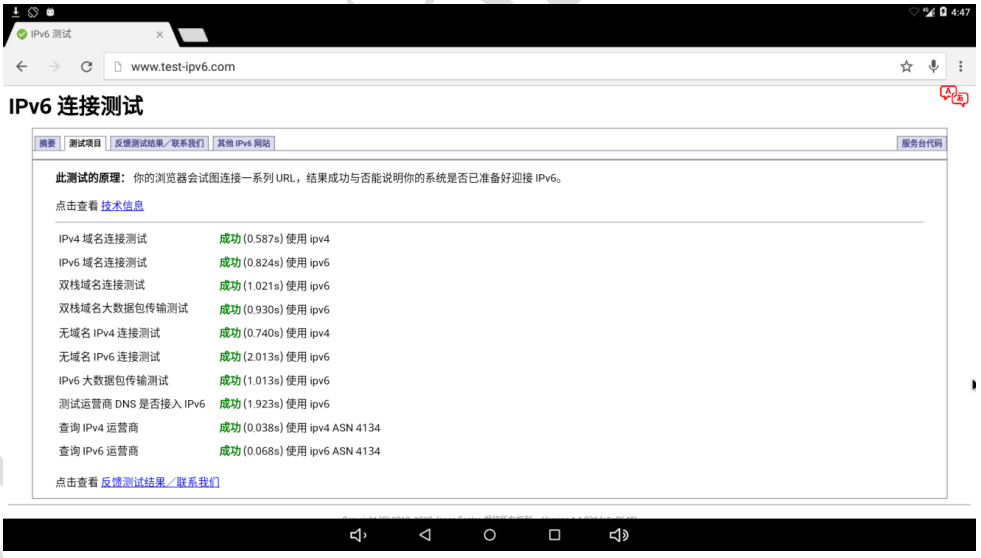


图 12 连接测试

16 常见问题处理

16.1 模块是否正常连接

使用 lsusb 可以查看到所有连接的 usb 设备的 vendor id 和 product id，可以来确认模块是否连接好。如：

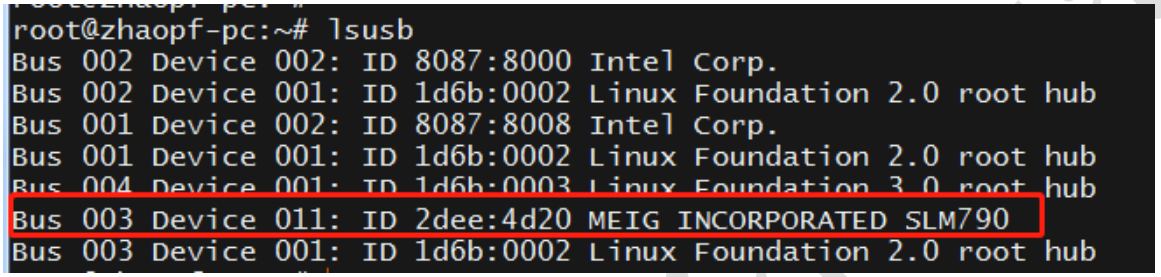


图 13 检查 usb 设备

如果未检测到对应模块，则首先需要检查模块所连接的 usb 口是否为 host 模式。

如果是 host，则需要检查模块供电是否正常、测量确认模块是否开机，

如模块已开机，则需要继续检查 usb 接线是否正常。

16.2 SIM 卡是否在位

先在 log 里查找关键字“CPIN”，以确认是否检测到 sim 卡，如：

```
AT> AT+CPIN?
AT< +CPIN: READY
```

16.3 信号检查

然后查找关键字“CSQ”，以确认天线是否插好。如:

```
AT> AT+COPS=3,0;+COPS?;+COPS=3,1;+COPS?;+COPS=3,2;+COPS?
AT< +COPS: 0,0,"004300 003F",7
AT< +COPS: 0,1,"00 003F",7
AT< +COPS: 0,2,"46011",7
```

16.4 注网检查

再查找关键字“COPS”，以确认是否注网成功。如:

```
AT> AT+COPS=3,0;+COPS?;+COPS=3,1;+COPS?;+COPS=3,2;+COPS?
AT< +COPS: 0,0,"004300 003F",7
AT< +COPS: 0,1,"00 003F",7
AT< +COPS: 0,2,"46011",7
```

16.5 usb 串口驱动检查

如果没有/dev/ttyUSB*设备，则需要检查 option 驱动是否加载

```
lsmod | grep  option
```

17 附录

17.1 定义 PDP 上下文命令 AT+CGDCONT

使用设置指令，可为 PDP 上下文定义参数，该 PDP 上下文是由本地上下文标识参数<cid>标识的。该设置指令的特殊形式+CGDCONT=<cid>将使上下文号码<cid>的取值成为未定义取值。测试指令返回一个复合值。若 MT 支持几种 PDP 类型<PDP_type>，则每个<PDP_type> 的参数值范围在单独一行上返回。

表 5 AT+CGDCONT 操作指令

类型	指令	可能的返回结果	说明
设置指令	AT+CGDCONT=[<cid>[,<PDP_type>[,<APN>[,<PDP_addr>[,<d_comp>[,<h_comp>]]]]]]	OK	-
		ERROR/+CME ERROR: <err>	失败
查询指令	AT+CGDCONT?	+CGDCONT: <cid>,<PDP_type>,<APN>,<PDP_addr>,<d_comp>,<h_comp>[<CR><LF>+CGDCONT:<cid>,<PDP_type>,<APN>,<PDP_addr>,<d_ comp>,<h_comp>] OK	-
测试指令	AT+CGDCONT=?	+CGDCONT: (range of supported <cid>s),<PDP_type>,,,(<d_comp>取值列表),(<h_comp>取值列表) OK	-
指令例程	AT+CGDCONT?	+CGDCONT: 1,"IPV4V6","","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0 +CGDCONT: 2,"IPV4V6","cmnet","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0 OK	-
	AT+CGDCONT＝1	OK	删除<cid>
	AT+CGDCONT?	+CGDCONT: 1,"IPV4V6","","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0	

	OK	
AT+CGDCONT=1,"IP","CMNET"	OK	APN为CMNET，PDP类型为IP
AT+CGDCONT=?	+CGDCONT: (1-16),"IP",,, (0-2),(0-4),(0-1),(0-1) +CGDCONT: (1-16),"PPP",,, (0-2),(0-4),(0-1),(0-1) +CGDCONT: (1-16),"IPV6",,, (0-2),(0-4),(0-1),(0-1) +CGDCONT: (1-16),"IPV4V6",,, (0-2),(0-4),(0-1),(0-1) OK	

表 6　AT+CGDCONT 参数详细说明

参数	取值	说明
<cid>	(1-16)	数值型参数；用于指定 PDP上下文标识。该参数对TE-MT接口而言是本地参数，并且可用于其他PDP上下文相关指令
<PDP_type>	["IP"]	(分组数据协议类型)字符型参数；用于指定分组数据协议的类型。默认支持"IP"互联网协议IP(Internet Protocol)(IETF STD5)
	X.25	ITU-T/CCITT X.25 layer 3 (Obsolete)
	IPV6	Internet Protocol, version 6 (IETF RFC 2460)
	OSPIH	Internet Hosted Octect Stream Protocol (Obsolete)
	PPP	Point to Point Protocol (IETF STD 51)
<APN>	-	接入点名称；表示一个字符串参数，用于选择GGSN或外部分组数据网络的逻辑名称。若该参数取值为空或省略，则需要请求签约值。
<PDP_address>	-	字符型参数；用于标识对于特定PDP上下文，MT分配的地址空间。若该参数取值为空或省略，则TE在PDP启动过程中提供其他取值；若不能提供其他取值，则需要请求动态地址。即便在PDP启动过程中已经分配地址，该指令的读出形式仍继续返回为空。使用+CGPADDR指令，可读出该分配地址。
<d_comp>	0	关闭(若取值省略，则该参数为缺省值)数值型参数；用于控制PDP数据压缩
	1	打开(厂商首选的PDP数据压缩)
	2	V.42
	3	V.44

		其它值保留
<h_comp>	0	关闭(若取值省略，则该参数为缺省值)数值型参数；用于控制PDP头压缩
	1	打开(厂商首选的PDP头数据压缩)
	2	RFC114(仅适用于SNDCP)
	3	RFC2507
	4	RFC3095 (applicable for PDCP only)
		其它值保留

所定义的<cid>不能与+CGDSCONT 中定义的<cid>重复。

17.2 RMNET 拨号命令 AT$QCRMCALL

该命令是基于 RMNET 的拨号命令，使用该指令可以进行数据的连接和断开。

表 7 AT$QCRMCALL 操作指令

类型	指令	可能的返回结果	说明
设置命令	AT$QCRMCALL=<Action>,<Instance> [,<IP Type> [,<Tech Pref > [,<umts profile number> [,<cdma profile number > [,<APN>]]]]]	OK	拨号成功
		NO CARRIER	拨号失败
查询命令	AT$QCRMCALL?	断开： OK 连接： $QCRMCALL: 1, V4 $QCRMCALL: 1, V6 OK	-
测试命令	AT$QCRMCALL=?	$QCRMCALL: (0-1),(1,2,3,4,5,6,7,8),(1-3),(1-2),(1-16), OK	-
指令例程	AT$QCRMCALL=1,1,1,2,1	$QCRMCALL: 1, V4 OK	拨号
	AT$QCRMCALL=0,1,1,2,1	OK	断开拨号

$QCRMCALL: 0, V4

表 8 AT$QCRMCALL 参数说明

参数	取值	说明
< Action >	0	Stop
	1	Start
<Instance>		1 to RMNET_ NUM_LAPTOP_ INSTANCES
<IP Type>	1	Ipv4
	2	Ipv6
	3	Ipv4v6
<Tech Pref>	1	3GPP2
	2	3GPP
<umts_profile>	1 to 16	-
<APN >	1	String type, maximum length is 100

17.3 NDIS 拨号 ^NDISDUP

说明

本命令用于实现 NDIS 拨号。

- at^ndisdup=1,1：NDIS 拨号。
- at^ndisdup=1,0：断开 NDIS 网络连接。本命令只用于 NDIS 端口形态

表 9 语法

命令类型	返回值

^NDISDUP=<pdpid>,<connect>[,<APN>[<username>[,<passwd>[,<authpref>]]]]	<CR><LF>OK<CR><LF> 错误情况: <CR><LF>+CME ERROR: <err><CR><LF>
^NDISDUP?	<CR><LF>OK<CR><LF>
^NDISDUP=?	GU 模： <CR><LF>^NDISDUP：(list of supported<pdpid>s),<0-1><CR><LF><CR><LF>OK<CR><LF> GUL 模： <CR><LF>^NDISDUP：(list of supported <pdpid>s),<0-1><CR><LF><CR><LF>OK<CR><LF>

表 10 参数

参数	说明
<pdpid>	整型值，PDP 上下文标识符。 GU 为 1～16（目前只支持 11，后续可扩展到 16）。 GUL 为 1～20。
<connect>	整型值，设置连接状态。取值如下： 0：断开连接； 1：建立连接。
<APN>	字符串类型，接入点名字，0～99byte。
<username>	字符串类型，用户名，0～255byte。
<passwd>	字符串类型，密码，0～255byte。
<authpref>	整型值，认证协议。取值如下： 1：PAP； 2：CHAP； 3：MsChapV2（目前暂不支持）。

示例

- NDIS 拨号

AT^NDISDUP=1,1

OK

^DATACONNECT

^NDISSTAT:1,,,"IPV4"

查询命令

AT^NDISDUP?

OK

- 测试命令

AT^NDISDUP=?

^NDISDUP：(1-20),(0-1)

OK