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 Android RIL 适配指导

受控版本：V3.8

发布时间：2022 年 12 月 06 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	2020-03-01	初次建立
V2.0	2020-05-15	添加 Android8.x 和所有模块的通用配置
V3.0	2020-06-01	添加 Android7.0,10.0 支持 添加 gps 功能启用方法 添加 hidl 和 sepolicy 配置方法
V3.1	2020-06-04	完善 GPS 配置方法
V3.2	2020-06-20	添加 Android5.0，Android6.0，IPv6 设置和验证方法
V3.3	2020-07-31	添加 SIM 卡热插拔配置方法
V3.4	2021-01-11	1.添加网口驱动(GobiNet&NCM)源码集成编译方法 2.添加 5G 速率优化方法 3.添加 SIM 卡热插拔配置方法 4.添加 APGS 启用说明 5.添加 ril 中所有属性使用说明 6.添加模块 log 抓取方法
V3.5	2022-03-09	添加 U 系列模块适配
V3.6	2022-05-31	适配 U 系列模块 NCM 拨号
V3.7	2022-06-23	1.所有”美格模块”修改为“美格智能模块” 2.热插拔描述删除表一中“热插拔”一列
V3.8	2022-12-06	1.更新所有模块端口定义及说明 2.调整所有模块端口适配方法，新增 A 系列模块端口适配方法 3.新增 PPP，GobiNet，NCM，ECM，RNDIS 驱动适配方法 4.补充 Android10 及以上版本 hidl 适配方法 5.更新网络相关配置 6.调整所有章节结构 7.修正 AGPS 错误 8.更新拨号上网方式选择部分 9.更新属性支持列表 10.常见问题中新增驱动检查和典型不拨号问题分析方法 11.更新所有章节中引用代码样式

目 录

表格索引

图片索引

1 引言

1.1 文档目的

本文档介绍如何在 Android 系统上集成美格智能 RIL(Radio Interface Layer)库，GPS 库，以及相关配置文件的修改。主要面向 FAE、Android 开发人员，引导其快速适配美格智能模块到设备上，以给设备提供数据，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：RIL 及驱动适配说明；

第 4 章：低版本 ANDROID 5G 支持；

第 5 章：GPS 功能添加方法；

第 6 章：RIL 扩展特性；

第 7 章：RIL 属性支持列表；

第 8 章：常见问题分析方法；

2 模块基本信息

本文介绍的模块都是通过 usb 与 Android 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；**

3 RIL 适配

RIL 主要使用 AT 来和模块交互，AT 通信需要借助于 USB 串口驱动。同样的，MODEM,, NMEA，DIAG 口等都需要加载 USB 串口驱动才能工作。

3.1 添加 USB 串口驱动

3.1.1 内核配置打开相应宏

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

3.1.2 修改 OPTION 驱动，添加模块支持

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 },
```

3.1.3 添加 USB 设备节点权限

Android 设备通常都会有一个 uevent.rc 文件用于配置设备节点权限，我们需要在此处给 ril 读写的设备加上 radio 权限，如

```
common/products/mbox/ueventd.amlogic.rc
@@ -75,6 +75,7 @@
 /dev/ttyHS2              0666   bluetooth  bluetooth
 /dev/ttyS20              0664   system     system

 /dev/ttyUSB*            0664   radio      radio

#如果用gobinet驱动
  /dev/qcqmi*            0664   radio      radio
#如果用PPP方式
  /dev/ppp               0664   radio      vpn
```

3.1.4 编译并更新内核

也可以编译出 option.ko 在使用 insmod 命令加载。

驱动加载后，如插入 Q 系列模块时，dev 目录会生成 4 个 ttyUSB 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 端口信息

3.2 添加 RIL 配置

如果当前平台已经配置过 RIL，且能正常工作，则可以跳过此节，直接使用 Meig_Android_Driver_XXX.tar.gz 中对应 android 版本的 ril 库替换当前运行的库文件即可。

为了便于集成 ril，gps 等功能，我们将需要的 ril 库文件、gps 库文件、apn 配置文件、驱动源码、日志工具等都打包并集成到 mk 文件中，然后在目标 device 中通过 include 的方式包含进来即可。

1) 先将 Meig_Android_Driver_XXX.tar.gz 解压到 Android 源码根目录下 vendor/meig 目录，

```
.
├── android.hardware.gnss@1.0-service.rc
├── buildinfo.txt
├── etc
├── gps-4.4
├── gps-5.0
├── gps-6.0
├── gps-7.0-later
├── init.meig.radio.rc
├── init.meig.radio.system.rc
├── init.meig.radio.system_x64.rc
├── init.meig.radio_x64.rc
├── install_meigdrv
├── libmeig-ril-4.4
├── libmeig-ril-5.0
├── libmeig-ril-6.0
├── libmeig-ril-7.0-later
├── meigdrv
├── meiglog
├── meig_radio.mk
├── meig_radio_x64.mk
├── ppp
├── readme.txt
├── RIL功能支持列表.xlsx
└── uninstall_meigdrv
```

图 2 适配文件列表

2) 在当前产品的 mk 文件中添加美格 ril 配置

配置时注意区分 64 位和 32 位。

如:device/khadas/kvim/kvim.mk:

```
kvim/kvim.mk
@@ -520,4 +520,9 @@ $(warning $(shell ($(AUTO_PATCH_AB) $(PRODUCT_DIR))))
 endif
 endif

+################################################################################
+#
+#                     MEIG RIL CONFIG
+#
+################################################################################
+#如果是32位
+-include vendor/meig/meig_radio.mk
+#如果是64位
+-include vendor/meig/meig_radio_x64.mk
```

3) Android6.0 及以下版本，需要额外添加如下配置

将 init.meig.radio.rc 文件包含在在对应 device 的 rc 文件中。如:

```
device/samsung/manta/init.manta.rc
@@ -1,4 +1,5 @@
 import init.manta.usb.rc
+import init.meig.radio.rc

 on init
     start watchdogd
```

3.3 添加网口驱动

如果使用 PPP 方式拨号，可以跳过此节。

网口驱动主要有 PPP, ECM, GobiNet, NCM, RNDIS 几种，我们的适配文件中提供了 Q 系列模块的 GobiNet 驱动，以及 H 系列模块的 NCM 驱动。

其他驱动 Android 内核源码中都有包含，且一般默认都有开启，无需额外配置；如未开启，后文会介绍开启方法。

3.3.1 集成 PPP 驱动

美格智能模块基本上都支持 PPP 方式拨号上网。

一般情况下，android 内核中默认支持 PPP 驱动，并且 SYSTEM 分区也会会打包依赖的脚本，并且我们提供的 meig_radio_xxx.mk 编译文件中也会打包 ppp 需要的脚本。

```
#VERSION: v2.0
#CREATE DATA: 2020/05/26
#meig ril ppp&apns&diagtool

PRODUCT_COPY_FILES += \
    vendor/meig/ppp/ip-down:system/etc/ppp/ip-down \
    vendor/meig/ppp/ip-up:system/etc/ppp/ip-up \
    vendor/meig/ppp/ip-up-vpn:system/etc/ppp/ip-up-vpn \
    vendor/meig/ppp/chat:system/bin/chat \
```

不过，有时候内核中的 PPP 支持并不完整，可能会有无法打开/dev/ppp 节点或者拨号后无法获取 IP 的问题，可参考如下方式在内核配置项中添加完整 PPP 驱动。

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

3.3.2 集成美格 GobiNet&NCM 驱动

1) 添加美格驱动到内核源码

为了便于集成驱动到 Android 内核源码中，我们提供了一键集成脚本，会自动将 **Q 系列**的 **GobiNet** 驱动和 **H 系列**的 **NCM 驱动**集成到对应平台内核源码中，注意，需要使用下一节的宏使能驱动才算完成。

执行前需要先加上可执行权限，

```
chmod +x vendor/meig/install_meigdrv

chmod +x vendor/meig/uninstall_meigdrv
```

执行./vendor/meig/install_meigdrv.sh，传参为内核源码目录，执行后会内核源码目录下创建 meigdrv 驱动目录。以下是几种平台的添加方法

如 amlogic 方案的上位机平台:

```
./vendor/meig/install_meigdrv common
#提示:
meig driver installed
```

高通方案的上位机平台：

```
./vendor/meig/install_meigdrv kernel/msm-3.18

#提示:
meig driver installed
```

2) 使能美格 USB 网卡驱动

驱动默认会直接打包到内核里，如不关注驱动加载方式，可跳过这一步。

如果需要编译成模块，则在对应的内核配置文件中添加如下配置,编出驱动后通过 insmod 命令自行加载 ko 文件， 如高通上位机平台:

```
/kernel/msm-5.4/arch/arm64/configs/vendor/lahaina_QGKI.config

#Q系列模块Gobi驱动
CONFIG_MEIG_GOBINET=m

#H系列模块NCM驱动
CONFIG_MEIG_NCM=m
```

```
insmod GobiNet.ko
insmod meig_cdc_driver.ko
```

如果某个驱动不想编译到，可在对应的内核配置文件中添加如下配置：

```
#Q系列模块Gobi驱动
CONFIG_MEIG_GOBINET=y

#H系列模块NCM驱动
CONFIG_MEIG_NCM=y
```

3) 卸载美格 USB 网卡驱动

卸载同样需要在 Android 源码根目录执行脚本，./vendor/meig/install_meigdrv，传参为内核源码目录.

如 amlogic 方案的上位机平台:

```
./vendor/meig/uninstall_meigdrv common

meig driver unistalled
```

3.3.3 原生 NCM 驱动支持

对于 NCM 端口的模块，需要在上位机上使能此驱动。如果确认驱动已经支持了，可以跳过此步。

如不能确定可按如下步骤检查。

1) 检查是否已经支持 NCM 驱动

插入模块后使用”ifocnfig -a”命令查看是否有新增名称为”usbX”或”ethX”的网卡(X 网卡实际序号),有则表示已支持。

```
root@zhaopf-ubuntu:/# ifconfig -a

usb0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
       ether a6:d5:26:84:81:7f  txqueuelen 1000  (Ethernet)
       RX packets 0  bytes 0 (0.0 B)
       RX errors 0  dropped 0  overruns 0  frame 0
       TX packets 0  bytes 0 (0.0 B)
       TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

或者查看内核 log 中是否有 ncm 相关的打印,有打印说明驱动已经支持。

```
root@zhaopf-ubuntu:/# dmesg | grep ncm

[1048429.882474] cdc_ncm 1-8:1.0: MAC-Address: a6:d5:26:84:81:7f
[1048429.883060] cdc_ncm 1-8:1.0 usb0: register 'cdc_ncm' at usb-0000:00:14.0-8, CDC NCM,
a6:d5:26:84:81:7f
```

2) 添加 NCM 支持

当前内核默认未添加 NCM 驱动时，需要在内核配置文件中打开 NCM 驱动宏来添加，

方法一(建议)、打包到内核中，如高通上位机平台:

```
kernel/msm-5.4/arch/arm64/configs/vendor/lahaina_QGKI.config

CONFIG_MII=y
CONFIG_USB_USBNET=y
CONFIG_USB_NET_CDCETHER=y
CONFIG_USB_NET_CDC_NCM=y
```

方法二、编译出 ko 模块，单独加载:

```
/kernel/msm-5.4/arch/arm64/configs/vendor/lahaina_QGKI.config

CONFIG_MII=m
CONFIG_USB_USBNET=m
CONFIG_USB_NET_CDCETHER=m
CONFIG_USB_NET_CDC_NCM=m
```

3.3.4 RNDIS 驱动支持

对于 RNDIS 端口的模块，需要在上位机上使能此驱动。如果确认驱动已经支持了，可以跳过此步。

如不能确定可按如下步骤检查,步骤与 NCM 驱动方法类似。

1) 检查是否已经支持 RNDIS 驱动

插入模块后使用”ifocnfig -a”命令查看是否有新增名称为”usbX”或”ethX”的网卡(X 网卡实际序号),有则表示已支持。

```
root@zhaopf-ubuntu:/# ifconfig -a

usb0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
       ether a6:d5:26:84:81:7f  txqueuelen 1000  (Ethernet)
       RX packets 0  bytes 0 (0.0 B)
       RX errors 0  dropped 0  overruns 0  frame 0
       TX packets 0  bytes 0 (0.0 B)
       TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

或者查看内核 log 中是否有 rndis 相关的打印,有打印说明驱动已经支持。

```
root@zhaopf-ubuntu:/# dmesg | grep -i rndis

[1047763.008189] usbcore: registered new interface driver rndis_host
```

2) 添加 RNDIS 支持

当前内核默认未添加 RNDIS 驱动时，需要在内核配置文件中打开 RNDIS 驱动宏来添加，

方法一(建议)、打包到内核中，如高通上位机平台:

```
kernel/msm-5.4/arch/arm64/configs/vendor/lahaina_QGKI.config

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

方法二、编译出 ko 模块，单独加载:

```
kernel/msm-5.4/arch/arm64/configs/vendor/lahaina_QGKI.config

CONFIG_MII=m
CONFIG_USB_USBNET=m
CONFIG_USB_NET_CDCETHER=m
CONFIG_USB_NET_RNDIS_HOST=m
```

3.3.5 ECM 驱动支持

对于 ECM 端口的模块，需要在上位机上使能此驱动。如果确认驱动已经支持了，可以跳过此步。

如不能确定可按如下步骤检查，与 NCM/RNDIS 驱动方法类似。

1) 检查是否已经支持 ECM 驱动

插入模块后使用”ifocnfig -a”命令查看是否有新增名称为”usbX”或”ethX”的网卡(X 网卡实际序号),有则表示已支持。

```
root@zhaopf-ubuntu:/# dmesg | grep cdc_ether

[1125114.340355] cdc_ether 1-8:1.0 usb0: register 'cdc_ether' at usb-0000:00:14.0-8, CDC
Ethernet Device, 56:17:6f:80:fa:c6
[1125114.340391] usbcore: registered new interface driver cdc_ether
```

或者查看内核 log 中是否有 ecm 相关的打印,有打印说明驱动已经支持。

```
root@zhaopf-ubuntu:/# dmesg | grep cdc_ether

[1125114.340355] cdc_ether 1-8:1.0 usb0: register 'cdc_ether' at usb-0000:00:14.0-8, CDC
Ethernet Device, 56:17:6f:80:fa:c6
[1125114.340391] usbcore: registered new interface driver cdc_ether
```

2) 添加 ECM 支持

当前内核默认未添加 ECM 驱动时，需要在内核配置文件中打开 ECM 驱动宏来添加，

方法一(建议)、打包到内核中，如高通上位机平台:

```
kernel/msm-5.4/arch/arm64/configs/vendor/lahaina_QGKI.config

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

方法二、编译出 ko 模块，单独加载:

```
/kernel/msm-5.4/arch/arm64/configs/vendor/lahaina_QGKI.config

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

3.4 添加 sepolicy 权限

Android 使用 selinux 对所有进程强制执行强制访问控制

libmeig-ril 库在使用 ppp 方式拨号时需要调用 pppd 和 chat 进程，

使用 Gobinet 拨方式号时需要读写/dev/qcqmiN 节点，因此都需要额外添加如下权限。

如下以 Android7-9 平台为例，其他平台可以参考。

配置 sepolicy 前，可以先检查下当前平台是否将 selinux 设为 permissive，如果是，则可以跳过这一步。查看方法：

```
adb shell getenforce
```

sepolicy 配置文件一般在 device/[实际平台]/common/sepolicy 目录，

如 rockchip 的 rild 相关权限配置文件 device/rockchip/common/sepolicy/rild.te

(1) Android7.0 配置

(2) Android8.0，9.0，10.0 配置

可参考如下方式配置

```
device/rockchip/common/sepolicy/file.te
@@ -37,3 +37,6 @@ type sysfs_lcdc, fs_type, sysfs_type, mlstrustedobject;
 type aplog_data_file, file_type, data_file_type;
 type fuseblk, sdcard_type, fs_type, mlstrustedobject;
 type proc_ostype, fs_type, mlstrustedobject;

#chat
type chat_exec, exec_type, file_type;
device/rockchip/common/sepolicy/file_contexts
@@ -32,6 +32,8 @@

 /radical_update(/.*)? u:object_r:ru_file:s0
 /dev/rtk_btusb      u:object_r:rtkbt_device:s0
 /dev/qcqmi[0-9]*       u:object_r:qmi_device:s0
 /system/bin/chat       u:object_r:chat_exec:s0
 /dev/rknand_sys_storage             u:object_r:rknand_device:s0
 #/system/bin/e2fsck                 u:object_r:ext4_exec:s0

device/rockchip/common/sepolicy/rild.te
@@ -13,3 +13,11 @@ allow rild net_dns_prop:file { getattr open read };
 allow rild rootfs:dir { open read };
 allow rild system_prop:property_service set;
 allow rild toolbox_exec:file { execute execute_no_trans getattr open read };
 allow rild qmi_device:chr_file { read open write };
 allow rild ueventd:file { read };
 allow rild device:file { open read };
 allow rild device:dir { open read };
 allow rild dhcp_data_file:file { write create open getattr };
 allow rild dhcp_data_file:dir { search add_name };
 allow rild chat_exec:file { execute getattr read open execute_no_trans };
 allow rild system_file:file { execute_no_trans };
```

3.5 添加 HIDL 配置

对于 Android8.0 及以上版本，需要配置 HIDL。

一般 device 下都有个宏 DEVICE_MANIFEST_FILE 配置了对应的 manifest.xml 文件，需要在此文件中配置 radio 相关的 hidl，telephone 才能找到 radio service,否则 RIL 跑不起来。如

```
device/khadas/kvim/BoardConfig.mk:183:DEVICE_MANIFEST_FILE:=
device/khadas/common/products/mbox/manifest/manifest_aosp.xml
```

对于 Android8.x/9.x 在 manifest_aosp.xml 中添加如下内容即可

```
common/products/mbox/manifest/manifest_aosp.xml
@@ -28,6 +28,24 @@
         </interface>
     </hal>
     <hal format="hidl">
        <name>android.hardware.radio</name>
        <transport>hwbinder</transport>
        <version>1.0</version>
        <interface>
            <name>IRadio</name>
            <instance>slot1</instance>
        </interface>
    </hal>
    <hal format="hidl">
        <name>android.hardware.radio.deprecated</name>
        <transport>hwbinder</transport>
        <version>1.0</version>
        <interface>
            <name>IOemHook</name>
            <instance>slot1</instance>
        </interface>
    </hal>
    <hal format="hidl">
         <name>android.hardware.bluetooth</name>
         <transport>hwbinder</transport>
         <version>1.0</version>
```

Android10 及以上版本:

```
common/products/mbox/manifest/manifest_aosp.xml
@@ -28,6 +28,24 @@
         </interface>
     </hal>
     <hal format="hidl">
        <name>android.hardware.radio</name>
        <transport>hwbinder</transport>
        <version>1.0</version>
        <interface>
            <name>IRadio</name>
            <instance>slot1</instance>
        </interface>
    </hal>
    <hal format="hidl">
         <name>android.hardware.bluetooth</name>
         <transport>hwbinder</transport>
         <version>1.0</version>
```

在部分平台上需要同时调整兼容性矩阵的配置，否则编译不过，如:

```
hardware/interfaces/compatibility_matrices/compatibility_matrix.4.xml
@@ -341,17 +341,15 @@
    <hal format="hidl" optional="true">
        <name>android.hardware.radio</name>
        <version>1.0-1</version>
        <interface>
            <name>IRadio</name>
            <instance>slot1</instance>
        </interface>
    </hal>

    <hal format="hidl" optional="true">
        <name>android.hardware.radio.config</name>
```

```
        <version>1.0-1</version>
        <interface>
            <name>IRadioConfig</name>
            <instance>default</instance>
        </interface>
    </hal>


hardware/interfaces/compatibility_matrices/compatibility_matrix.5.xml
@@ -371,18 +371,15 @@
    <hal format="hidl" optional="true">
        <name>android.hardware.radio</name>
        <version>1.0-1</version>
        <interface>
            <name>IRadio</name>
            <instance>slot1</instance>
        </interface>
    </hal>

    <hal format="hidl" optional="true">
        <name>android.hardware.radio.config</name>
        <version>1.0-1</version>
        <interface>
            <name>IRadioConfig</name>
            <instance>default</instance>
        </interface>
    </hal>
```

其中 compatibility_matrix.N.xml 以平台实际使用的为准,如不确定，可全部都修改。

3.6 添加网络相关配置

在网络属性中添加移动网络支持，在配置文件 frameworks/base/core/res/res/values/config.xml，一般在对应 device 下 overlay 掉这个配置，需要注意以 overlay 下的 config.xml 为准。

```
frameworks/base/core/res/res/values/config.xml
@@ -294,11 +294,11 @@
         is available at all is controlled by the flag: config_moble_data_capable. -->
    <string-array translatable="false" name="networkAttributes">
        <item>"wifi,1,1,1,-1,true"</item>
        <item>"mobile,0,0,0,-1,true"</item>

@@ -496,7 +496,7 @@
    <integer-array translatable="false" name="config_tether_upstream_types">
        <item>1</item>
        <item>7</item>
        <item>0</item>
    </integer-array>

    <!-- If the DUN connection for this CDMA device supports more than just DUN -->
@@ -1754,7 +1754,7 @@
         PackageManager.FEATURE_TELEPHONY system feature, which is
         available on *any* device with a telephony radio, even if the
         device is data-only. -->
    <bool name="config_voice_capable">true</bool>

    <!-- Flag indicating whether all audio streams should be mapped to
         one single stream. If true, all audio streams are mapped to
@@ -1776,7 +1776,7 @@

         Note: Disable SMS also disable voicemail waiting sms,
               cell broadcasting sms, and MMS. -->
    <bool name="config_sms_capable">true</bool>

    <!-- Default SMS Application. This will be the default SMS application when
         the phone first boots. The user can then change the default app to one
@@ -1794,7 +1794,7 @@
```

```
        If true, this means that the device supports data connectivity through
        the telephony network.
        This can be overridden to false for devices that support voice and/or sms . -->
    <bool name="config_mobile_data_capable">true</bool>
```

4 低版 ANDROID 5G 支持

Android10 以下版本默认不支持 5G，为了便于低版本系统集成 5G 模块，我们提供了补丁，用于显示 5G 信号栏、图标，以及优化 5G 速率。如果确认当前模块不支持 5G，可跳过此章节。

这些补丁是基于通用 Android 系统生成，在部分平台上会有一些差异，需要客户参考修改。

补丁文件列表:

文件	说明
5g_patches_for_android9.x.tar.gz	android9.0 5G 支持补丁，基于 Amlogic S905X 平台生成
5g_patches_for_android8.x.tar.gz	Android8.0 5G 支持补丁

4.1 Android9.0 添加 5G 支持

解压 5g_patches_for_android9.x.tar.gz 后会看到如下文件,

表 3 Android9.0 补丁文件列表

Patches	文件说明
frameworks_base.patch	SystemUI 和 Frameworks
frameworks_opt_telephony.patch	Telephony service
hardware_ril.patch	ril service

打上补丁方法：

```
cd frameworks/base
patch -p1 < frameworks_base.patch

cd frameworks/opt/telephony/
patch -p1 < frameworks_opt_telephony.patch

cd hardware/ril
patch -p1 < hardware_ril.patch
```

打完补丁，编译、升级软件后，注册 5G 网络后信号栏会出现 5G 图标，如果现实效果与实际平台不一致，可自行调整一下图标文件。

4.2 Android8.0 添加 5G 支持

解压补丁文件 5g_patches_for_android8.x.tar. Gz 得到 patch 文件：

表 4 Android8.0 5G 补丁文件

Patches	文件说明
add-for-meig-5g-device-support.patch	SystemUI 和 Frameworks 及 RIL

进入 Android 源码根目录，打上补丁：

```
patch -p1 < add-for-meig-5g-device-support.patch
```

请注意，此 patch 文件是根据瑞芯微 ROC_RK3399_PC 8.1 平台生成，对于有的平台可能需要参考此补丁手动合入。

至此 5G 图标显示功能添加完成，需要注册上 5G 网络才能显示出来。

4.3 5G 速率适配

对于 5G 设备，因速率相比 4G 有很大提升，需要调大 Android 的 TCP buffer 大小才能体现出 5G 的优势。修改之前请先完成图标添加，因为有依赖关系。具体修改方法如下：

```
frameworks/opt/telephony/src/java/com/android/internal/telephony/dataconnection/DataConnection.java
@@ -760,6 +760,9 @@ public class DataConnection extends StateMachine {
    private static final String TCP_BUFFER_SIZES_LTE  =
          "524288,1048576,2097152,262144,524288,1048576";
    private static final String TCP_BUFFER_SIZES_HSPAP=
"122334,734003,2202010,32040,192239,576717";
     private static final String TCP_BUFFER_SIZES_5G =
"2097152,6291456,16777216,512000,2097152,8388608";

    private void updateTcpBufferSizes(int rilRat) {
       String sizes = null;
@@ -827,6 +830,11 @@ public class DataConnection extends StateMachine {
              case ServiceState.RIL_RADIO_TECHNOLOGY_HSPAP:
                  sizes = TCP_BUFFER_SIZES_HSPAP;
                  break;
              case ServiceState.RIL_RADIO_TECHNOLOGY_NR5G:
                 sizes = TCP_BUFFER_SIZES_5G;
                 break;
              default:
                  // Leave empty - this will let ConnectivityService use the system default.
                  break;
```

5 GPS 功能支持

如果不需要 GPS 功能，或者当前模块不支持 GPS，可以跳过此节。

GPS 功能主要用来给 Android 上各种 APP 提供定位数据，适配的目的是将模块的 GPS 功能通过 USB NMEA 口、gps hal， 与 Android 的 gnss 服务对接起来。

5.1 配置 HAL

与 4.3 节添加 radio hidl 配置方法相同，Android8.0 及以上版本需要在对应的 manifest.xml 文件中添加 gnss 的 hidl 配置,如:

```
common/products/mbox/manifest/manifest_aosp.xml
@@ -46,6 +46,15 @@
         </interface>
     </hal>
    <hal format="hidl">
        <name>android.hardware.gnss</name>
        <transport>hwbinder</transport>
        <version>1.0</version>
        <interface>
            <name>IGnss</name>
            <instance>default</instance>
        </interface>
    </hal>
```

5.2 启用模块 GPS

如需启用 gps 支持，需要在对应的 meig_radio.mk 或 meig_radio_x64.mk 文件中，将宏 BUILD_WITH_MEIG_GPS 设为 true。

注意: 如果 gps 实际端口或模块名称等与 gps_cfg.inf 中不一致，需要修改后再使用，编译后存放在系统目录/system/etc/下。

```
50 #enable meig gps
51 BUILD_WITH_MEIG_GPS := true
52 ifeq ($(BUILD_WITH_MEIG_GPS),true)
53 #gps
54 PRODUCT_PACKAGES += android.hardware.gnss@1.0-impl android.hardware.gnss@1.0-service
55 PRODUCT_COPY_FILES += \
56        vendor/meig/gps/lib64/gps.default.so:$(OUT_PARTITION)/lib64/hw/gps.default.so \
57        vendor/meig/gps/lib/gps.default.so:$(OUT_PARTITION)/lib/hw/gps.default.so \
58        vendor/meig/android.hardware.gnss@1.0-service.rc:$(OUT_PARTITION)/etc/init/android.hardware.gnss@1.0-service.rc
59
60 #if need, default auto detect
61 #PRODUCT_COPY_FILES += \
62     vendor/meig/gps/etc/gps_cfg.inf:$(OUT_PARTITION)/etc/gps_cfg.inf
63
64 #enable gps of modem
65 PRODUCT_PROPERTY_OVERRIDES += \
66     ril.gps.enable=true
67 endif
```

图 3 启用 GPS

5.3 AGPS 配置

为提高 GPS 定位时的搜星速度，缩短定位时长，可以通过如下属性开启 AGPS 辅助定位，同时须进行 SUPL 配置。

```
PRODUCT_PROPERTY_OVERRIDES += \
       ril.agps.enable=true
```

配置文件添加 SUPL 配置信息：/system/etc/gps.conf，注意，该配置文件的 SUPL_HOST 和 SUPL_PORT 只需定义一次，文件内出现多处定义时，以第一处为准。

```
76
77  ##################################
78  ##### AGPS server settings #####
79  ##################################
80
81  # FOR SUPL SUPPORT, set the following
82  SUPL_HOST=supl.qxwz.com
83  SUPL_PORT=7276
```

图 4 AGPS 配置

6 RIL 扩展特性

通用 ril 默认情况下可以支持大部分场景的使用，对于其他有特殊需求的用户，可按照如下方式进行适配

6.1 SIM 卡热插拔支持

一般情况下 SIM 卡热插拔功能是没有开启的，因为依赖于硬件设计是否支持。

如确认硬件是支持的，可通过设置属性来启用此功能.

ril.simhotplug.enable=true 用于启用热插拔，

ril.simhotplug.polarity 用于设置检测脚有效电平状态(与实际硬件一致)，0：低有效，1：高有效。

当系统第一次启动时会将修改保存到模块里，模块再次上电时生效。配置方法如：

```
PRODUCT_PROPERTY_OVERRIDES += \
    ril.simhotplug.enable=true \
    ril.simhotplug.polarity=0
```

6.2 IPV6 功能验证

因部分用户比较关注 IPv6 功能，因此我们在这一节着重说明一下 IPv6 的配置和验证方法。

目前基本所有的美格智能模块都支持 IPv6 功能，实际使用时可以与我们 FAE 同事确认。

6.2.1 IPv6 配置

在系统设置中找到移动网络子项，并找到当前 APN 设置，

确保”APN 协议”包含”IPv6”，如不包含，可修改为”IPv4/IPv6”或者”IPv6”，然后保存，即可生效。

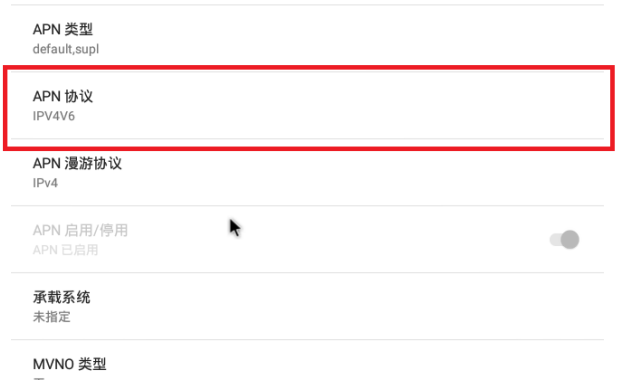

图 5 APN 设置

6.2.2 命令方式验证

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 6 IPv6 PING

6.2.3 Web 方式验证

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

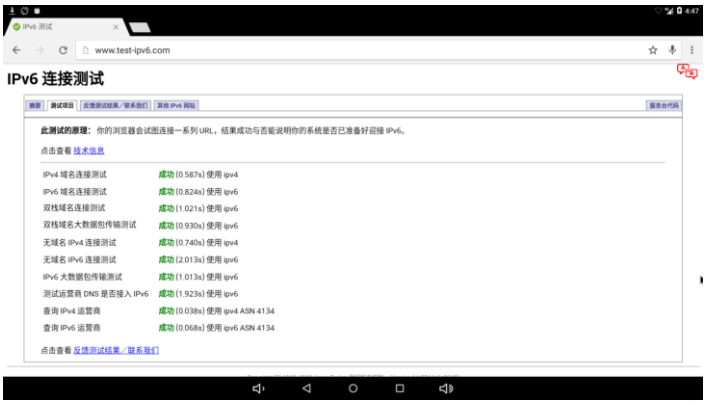

图 7　IPv6 测试

6.3　拨号上网方式选择

在 Android 平台上，美格智能模块支持的拨号方式一般有 QMI(rmnet)，NCM， ECM， RNDIS， PPP 等方式，

默认情况下，RIL 库会根据驱动加载情况优先选择速率最高的方式，用户无需关注。

不过，对于某些需要固定为特定拨号方式的场景，如驱动加载过晚导致 RIL 未能识别到最佳驱动时，也可以通过属性来配置。

6.3.1　自动选择

RIL 中自动选择拨号方式是的优先级如下:

- Q 系列模块: QMI > ECM/RNDIS > PPP
- H/U 系列模块: NCM > ECM/RNDIS > PPP
- A 系列模块：ECM/RNDIS > PPP

6.3.2　固定配置

可以通过修改 meig_radio.mk 中的属性 ril.dail.mode 来强制设置成固定的拨号方式，对应的值为“qmi”，“ncm”，“ecm”，“rndis”，“ppp”，需要注意要和模块实际网口形态保持一致。

配置方法如下:

```
PRODUCT_PROPERTY_OVERRIDES += \
    ril.dial.mode=ppp
```

7 RIL 属性支持列表

为了满足用户各种不同的使用场景，RIL 中通过属性扩展了一些非通用功能，可按如下方式来启用或关闭。详细如下表：

表 5 RIL 属性支持列表

属性名称	取值范围	默认值	功能
persist.sys.meig.srvdomain	cs,ps,both	auto	锁注册域，开关飞行模式或者重启后生效
persist.sys.meig.5gmode	sa,sansa,auto	both	锁注册域，开关飞行模式或者重启后生效
ril.prefernet.disable	false,true	false	禁用网络优先级设置
ril.fixed.radiotech	同 hardware / ril / include / telephony/ril.h 中 RIL_RadioTechnology 定义,5G 取值 20	无	仅调试使用，用于调试信号栏显示，重启 ril 生效
ril.meig.need.attachapn	false,true	false	专网卡需要特定 APN 才能注网的情况下开启此功能
ril.sleep.work	false,true	Q 系列二代模块：true; 其他:false	是否支持休眠唤醒功能,目前仅 Q 系列模块支持
ril.sleep.enable	false,true	true	是否启用休眠唤醒功能(需要先支持)
ril.meig.modem.reset	false,true	false	当 RIL 检测到不可恢复的错误时设置此属性为 true，无需认为设置。 如果硬件设计上支持 gpio 复位模块功能，可将属性变化与复位 io 联动起来。
ril.simhotplug.polarity	1,0	1	sim 卡热插拔触发电平
ril.simhotplug.enable	false,true	false	是否启用 SIM 卡热插拔功能
ril.gps.enable 或 persist.vendor.ril.gps.enable	false,true	false	是否启用 gps
ril.agps.enable	false,true	false	是否启用 agps 辅助定位功能
ril.codec.reset	false,true	false	初始化时是否需要复位 codec

ril.dial.mode 或 ro.dial.mode	ppp,ncm,ecm,rndis,qmi, multiqmi	无	选择拨号方式
ril.ndismulti.num	0,1,2,3,4	0	多路 NDIS(QMI)拨号的个数
ril.ndismulti.apn2 ril.ndismulti.apn3 ril.ndismulti.apn4	APN 字符串	无	用于设置第 2-4 路 APN 名称，第 1 路由系统设置不需要使用属性。 目前仅 Q 系列模块支持持，需要配合 v1.4.3 及以上版本 GobiNet 驱动使用 如: ril.ndismulti.num=3 ril.ndismulti.apn2=”APN2” ril.ndismulti.apn3=”APN3”
ril.debug.enable	false,true	false	打开调试 log
ril.use.csq	false,true	false	使用 AT+CSQ 方式上报信号，5G 模块不可用此功能
sys.meig.modem.state	connected,disconnected	无	RIL 会根据模块连接状态变更此属性值，无需人为设置，可用于应用层获取模块状态
persist.ril.use.oldgobi	true,false	false	当 v5.0.0 及以上版本 RIL 配合 v1.4.2 及以下版本 gobinet 驱动时候时使用此属性可解决兼容性问题
persist.meig.ims.disable	true,false	false	禁用 ims 注册，目前仅对 Q 系列模块生效
ril.menusearch.enable	true,false	false	改善手动搜网结果，对于比较关注手动搜网的用户可以设为 true
persist.sys.meig.uacsample	-1,0,1	-1	-1：关闭 UAC 0：8K 采样率 1:16K 采样率

8 常见问题分析

8.1 抓取日志

通常分析拨号相关问题，我们只需要抓取 radio 日志即可。

RIL 日志：

```
#清除缓存日志，非必须
adb logcat -b radio -c

#将radio 日志导出到radio.txt文件
adb logcat -b radio  -v time > radio.txt
```

常见日志标签说明:

表 6 常见日志标签

RIL	RIL 库通用日志
RIL-AT	RIL 库中 AT 收发日志
RIL-CM	RIL 库中 Connection Manager 日志，使用 QMI 通信
RILC	libril 库日志
RILJ	framework 层 RIL.java 日志

系统日志:

对于 RIL 工作正常，但信号栏显示异常，或者拨号成功却不能上网的情况，则需要抓取系统的日志来分析 APP、路由等工作情况

```
#清除缓存日志，非必须
adb logcat -b system -b main -c

#将系统日志导出到system.txt文件
adb logcat -b system -b main  -v time > system.txt ‘’
```

模块日志:

对于模块本身问题，需要抓取模块的 log，一般不需要。

抓取模块日志时，需要先将 MeigLogTool 推到机器里，对于 Q 系列模块，需要使用-f 指定 mask 文件其他模块则不用，如：

```
#Q系列模块
MeigLogTool -f /etc/meiglog.cfg -s /data/misc/meig

#其他模块
MeigLogTool -s /data/misc/meig

#取出日志
adb pull /data/misc/meig
```

如果已经集成了我们提供的 init.meig.radio*.rc 文件，可按如下方式抓取。

adb shell 到机器里去操作，或将属性操作进程到开发者选项里

```
#清除本地日志
setprop sys.meig.log.clear true

#开始抓取日志
setprop persist.sys.radio.log

停止抓取日志
setprop persist.sys.radio.log

#取出日志
adb pull /data/misc/meig
```

8.2 模块状态查看

8.2.1 是否检测到模块端口

Log 找那个过滤关键字”RIL-DEV”,以确认是否有识别到模块，如：

```
D/RIL-DEV ( 1563): Get modem info
D/RIL-DEV ( 1563): version is * 2.0*
D/RIL-DEV ( 1563): Find idVendor=05c6, idProduct=f601
D/RIL-DEV ( 1563): Find port = ttyUSB2
D/RIL-DEV ( 1563): Find modem path=/sys/bus/usb/devices/1-1.3:1.1
D/RIL-DEV ( 1563): Find port = ttyUSB1
D/RIL-DEV ( 1563): Find net mode in path=/sys/bus/usb/devices/1-1.3:1.5
D/RIL-DEV ( 1563): get_netif_mode_by_path=/sys/bus/usb/devices/1-1.3:1.5
D/RIL-DEV ( 1563): Find net interface in path=/sys/bus/usb/devices/1-1.3:1.5
E/RIL-DEV ( 1563): didn't found net interface
D/RIL-DEV ( 1563): vid:05c6
D/RIL-DEV ( 1563): pid:f601
D/RIL-DEV ( 1563): at port:2
D/RIL-DEV ( 1563): ppp port:1
D/RIL-DEV ( 1563): net port:5
D/RIL-DEV ( 1563): solution:QCM
D/RIL-DEV ( 1563): net mode:ppp
D/RIL-DEV ( 1563): net: not 5G
D/RIL-DEV ( 1563): interface:ppp0
D/RIL-DEV ( 1563): at port:/dev/ttyUSB2
D/RIL-DEV ( 1563): modem port:/dev/ttyUSB1
```

8.2.2 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
```

8.2.3 信号检查

然后查找关键字“CSQ”，以确认天线是否插好。如:

```
AT> AT^HCSQ?

AT< ^HCSQ: 0,0,"LTE",54,14,52,186
```

注意：CSQ 不支持 5G 信号，5G 信号需要使用 HCSQ

8.2.4 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
```

8.3 驱动加载失败问题

8.3.1 usb 连接检查

使用 lsusb 命令，可以查看 usb 连接是否正常，如果能查到模块的 vid&pid 信息，则正常，否则得检查模块是否上电，或 usb 连接是否正常。

如下图中查到的是美格智能的 vid:2dee, pid: 4d57 的模块。

```
adb root
adb shell
lsusb
Bus 003 Device 003: ID 2dee:4d57 Marvell Mobile Composite Device Bus
```

8.3.2 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,可以通过启过滤内核日志中是否有串口相关打印来判断

```
adb root
adb shell
dmesg | grep option

[ 1001.363923] usbcore: registered new interface driver option
```

8.3.3 网卡驱动检查

网卡驱动检查方法详见 3.3 节驱动加载部分。

8.3.4 驱动匹配检查

驱动加载成功，不代表这驱动和模块端口匹配成功，只有匹配成功了才能正常工作。

查看是否匹配成功可以通过 lsusb -t 可以看到命令：

```
$ lsusb
Bus 002 Device 002: ID 8087:8000 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 8087:8008 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 003: ID 2dee:4d57 Marvell Mobile Composite Device Bus
Bus 003 Device 002: ID 1c4f:0034 SiGma Micro XM102K Optical Wheel Mouse
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

#查到2dee:4d57的模块再bus:003 device 003上

$ lsusb -t
/:  Bus 04.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/2p, 5000M
/:  Bus 03.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/10p, 480M
    |__ Port 1: Dev 2, If 0, Class=Human Interface Device, Driver=usbhid, 1.5M
    |__ Port 5: Dev 3, If 0, Class=Wireless, Driver=rndis_host, 480M
    |__ Port 5: Dev 3, If 1, Class=CDC Data, Driver=rndis_host, 480M
    |__ Port 5: Dev 3, If 2, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 5: Dev 3, If 3, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 5: Dev 3, If 4, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 5: Dev 3, If 5, Class=Vendor Specific Class, Driver=option, 480M
    |__ Port 9: Dev 4, If 0, Class=Human Interface Device, Driver=usbhid, 1.5M
    |__ Port 9: Dev 4, If 1, Class=Human Interface Device, Driver=usbhid, 1.5M
/:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/2p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/6p, 480M
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/2p, 480M
|__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/4p, 480M

#可以看到bus:003 device 003 每个If(Interface)驱动加载情况:0,1: rndis_host, 2-5:option
```

或者，在不支持 lsusb 命令的设备上使用 cat /sys/kernel/debug/usb/devices 节点查看：

```
adb root
adb shell
# cat /sys/kernel/debug/usb/devices

T:  Bus=03 Lev=01 Prnt=01 Port=04 Cnt=02 Dev#=  3 Spd=480  MxCh= 0
D:  Ver= 2.00 Cls=ef(misc ) Sub=02 Prot=01 MxPS=64 #Cfgs=  1
P:  Vendor=2dee ProdID=4d57 Rev= 1.00
S:  Manufacturer=Marvell
S:  Product=Mobile Composite Device Bus
S:  SerialNumber=200806006809080000
C:* #Ifs= 6 Cfg#= 1 Atr=c0 MxPwr=500mA
A:  FirstIf#= 0 IfCount= 2 Cls=e0(wlcon) Sub=01 Prot=03
I:* If#= 0 Alt= 0 #EPs= 1 Cls=e0(wlcon) Sub=01 Prot=03 Driver=rndis_host
E:  Ad=87(I) Atr=03(Int.) MxPS=  64 Ivl=4096ms
I:* If#= 1 Alt= 0 #EPs= 2 Cls=0a(data ) Sub=00 Prot=00 Driver=rndis_host
E:  Ad=83(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=0c(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
I:* If#= 2 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=00 Prot=00 Driver=option
E:  Ad=82(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=0b(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
I:* If#= 3 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=option
E:  Ad=88(I) Atr=03(Int.) MxPS=  64 Ivl=4096ms
E:  Ad=81(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=0a(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
I:* If#= 4 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=option
E:  Ad=89(I) Atr=03(Int.) MxPS=  64 Ivl=4096ms
E:  Ad=86(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=0f(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
I:* If#= 5 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=00 Prot=00 Driver=option
E:  Ad=85(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=0e(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms

#可以看到vid:2dee pid:4d22 每个If(Interface)驱动加载情况:0,1: rndis_host, 2-5:option
```

8.4 不拨号问题

8.4.1 HAL 通信未建立

一般是 manifest.xml 文件配置不正确，见 hal 配置章节描述。

或者 radio service 未启动导致，Android8 及以上版本可以查看机器里是否有如下服务

```
lahaina:/ # ps -ef | grep android.hardware.radio
system       956     1 0 10:00:55 ?    00:00:00 android.hardware.radio.config@1.0-service
system       960     1 0 10:00:55 ?    00:00:00 android.hardware.radio@1.2-radio-service
system       963     1 0 10:00:55 ?    00:00:00 android.hardware.radio@1.2-sap-service
```

如果没启动可以当前项目对应的 mk 文件里添加进去

```
PRODUCT_PACKAGES += \
                    android.hardware.radio@1.2-radio-service \
                    android.hardware.radio.config@1.0-service
```

8.4.2 权限问题

权限问题一般是没有权限读取设备节点

如/dev/ttyUSB*，或者/dev/qcqmi*不是 radio 权限，

或者 selinux 权限不够。

8.4.3 未匹配到有效 APN

从 radio log 中过滤出 AT+CIMI 的结果，前 6 位数字为 PLMN 值(如下是 460 11)，根据此值去 apns-conf.xml 中搜索，看是否有默认 APN，如果没有则可以参考已支持的 APN 来添加。

```
12-05 06:51:19.030  408  408 D RIL-AT  : AT> AT+CIMI
12-05 06:51:19.034  770  770 D CarrierKeyDownloadManager: Carrier not enabled or invalid values.
mKeyAvailability=0 mURL=null
12-05 06:51:19.035  770  770 D CarrierKeyDownloadManager: Cleaning up existing renewal alarms
12-05 06:51:19.035  408  519 D RIL-AT  : AT< 460110411576662
12-05 06:51:19.035  408  519 D RIL-AT  : AT< OK
```

apns-conf.xml 中

```
<apn carrier="China Telecom" apn="CTLTE"  mcc="460" mnc="11" user="ctnet@mycdma.cn"
password="vnet.mobi" server="" proxy="" port="" mmsproxy="" mmsport="" mmsc=""
type="default,hipri,supl" />
```

如果想临时更新 apns-conf.xml 文件来验证，则需要按如下优先级依次检查哪个文件存在，就修改哪个文件。

```
#顺序越靠前优先级越高
1./data/misc/apns/apns-conf.xml
2./product/etc/apns-conf.xml
3./oem/etc/apns-conf.xml
4./system/etc/apns-conf.xml
```

修改后需要删除 telephone 数据库文件然后重启才能生效(如果是系统升级，则对应的需要进行一次恢复出厂)

```
adb root
adb remount
adb shell
#删除数据库，重启设备后会根据 apns-conf.xml 新建
rm $(find /data -iname "tele*.db*")
```

8.4.4 数据开关未使能

检查系统设置里的数据开关是否打开。