密级状态：绝密(　　)　　秘密(　　)　　内部(　)　　公开(　√　)

ROCKCHIP_3G_DONGLE_配置说明

（技术部，第二系统产品部）

文件状态： [　] 正在修改 [√] 正式发布	当前版本：	V1.0
	作　　者：	ZZC
	完成日期：	2017-12-21
	审　　核：	
	完成日期：	

福州瑞芯微电子股份有限公司

Fuzhou　Rockchips　Semiconductor　Co . , Ltd

版 本 历 史

版本号	作者	修改日期	修改说明	备注
V1.0	ZZC	2017-11-15	发布初稿	

目 录

1 目的

明确 android 平台上 3g dongle 支持原理和注意事项。

1.1 Kernel 注意事项

如果插入 3G dongle 无法被系统识别需要添加 3G dongle 的 pid 和 vid 支持。

以华为 MU509 为例：

```
diff --git a/drivers/usb/serial/option.c b/drivers/usb/serial/option.c
index 2317e59..74be8d1 100755
--- a/drivers/usb/serial/option.c
+++ b/drivers/usb/serial/option.c
@@ -80,6 +80,7 @@ static void option_instat_callback(struct urb *urb);
 #define OPTION_PRODUCT_GTM380_MODEM            0x7201

 #define HUAWEI_VENDOR_ID                       0x12D1
+#define HUAWEI_PRODUCT_MU509                    0x1001
 #define HUAWEI_PRODUCT_E173                    0x140C
 #define HUAWEI_PRODUCT_E140C                    0x140C
 #define HUAWEI_PRODUCT_K4505                   0x1464
@@ -597,6 +598,7 @@ static const struct usb_device_id option_ids[] = {
        { USB_DEVICE(QUANTA_VENDOR_ID, QUANTA_PRODUCT_GLX) },
        { USB_DEVICE(QUANTA_VENDOR_ID, QUANTA_PRODUCT_GKE) },
        { USB_DEVICE(QUANTA_VENDOR_ID, QUANTA_PRODUCT_GLE) },
+       { USB_DEVICE(HUAWEI_VENDOR_ID, HUAWEI_PRODUCT_MU509) },
         { USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, HUAWEI_PRODUCT_E140C, 0xff, 0xff, 0xff) },
        //{ USB_DEVICE_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID, HUAWEI_PRODUCT_E173, 0xff, 0xff, 0xff),
        //      .driver_info = (kernel_ulong_t) &net_intf1_blacklist },
```

1.2Android 注意事项

SDK 通常默认 dongle 功能是关闭的，需要 dongle 功能的项目可以在 BoradConfig.mk 中将 BOARD_HAVE_DONGEL 设置为 true 启用 dongle 功能。

```
#enable 3g dongle
BOARD_HAVE_DONGLE ?= false
```

Android8.0 注意：（由于大部分项目没有 radio 且需要过 CTS/VTS，因此在 manifest 中没有加入 radio 的相关项，不然会导致测试不过）
需要在 device/rockchip/common/manifest.xml 加入以下修改：

```
    <hal format="hidl">
        <name>android.hardware.radio</name>
        <transport>hwbinder</transport>
        <version>1.0</version>
        <interface>
            <name>IRadio</name>
```

```
            <instance>slot1</instance>
        </interface>
        <interface>
            <name>ISap</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>
```

2 Dongle 上网原理说明

插入 dongle 后，通过 usb modem switch 通过 PID/VID 完成从 CDROM 到 MODEM 的切换，切换完成后，usb 加载相应的驱动。加载完驱动后，系统中会出现 ttyACM*或 ttyUSB*的串口。之后通过此串口发送 AT Command 完成拨号上网。

2.1 Dongle 模式切换

市面上的打部分 dongle 默认模式是存储模式，需要切换到 modem 模式才能用于上网。这里我们用到一个开源的工具（usb_modeswitch）来对 dongle 模式进行切换。

涉及代码：

system/vold

external/usb_modeswitch

vold 负责监视系统是否有 3G dongle 设备出现，并调用 usb_modeswitch 进行 USB 模式切换。

代码如下：

```
int G3Dev::handleUsbEvent(NetlinkEvent *evt) {
 const char *devtype = evt->findParam("DEVTYPE");
    if( devtype!=NULL &&strcmp(devtype, "usb_device") )
        return 0;
        pid_t status ;

        NetlinkEvent::Action action = evt->getAction();
        if( action == NetlinkEvent::Action::kAdd)
        {
        const char *product = evt->findParam("PRODUCT");
        if(product!=NULL && product[0] != 0 && devtype[0] != 0 )
        {
            // 获取VID/PID
            int vid = 0;
            int pid = 0;
            char * next = (char*)product;
            vid = strtol(product, &next, 16);

                        //char pre[]="sVe.GT";
            ++next;
            pid = strtol(next, NULL, 16);

            SLOGD("== current usb device: %04X/%04X ===", vid, pid);

            char configure_file[2048];


            sprintf(configure_file, "/etc/usb_modeswitch.d/%04x_%04x", vid, pid);
            if( access(configure_file, 0) == 0 )
            {
                sprintf(modeswitch_cmd, "/system/bin/usb_modeswitch -W -v %04x -p %04x -c %s &", vid, pid,configure_file);
                SLOGD("=== USB Switch: %s", modeswitch_cmd);
                system(modeswitch_cmd);
            }
        }
        }

    return 0;
}
```

usb_modeswitch 已经支持大市面上的大部分 dongle 的模式切换，相关配置文件可以查看：external/usb_modeswitch/usb_modeswitch.d/目录，配置文件命名方式为 pid_vid。

如果所使用的 dongle 没有在 SDK 支持范围可以按照《3G 数据卡 USB 切换文件制作说明_v1.2.pdf》来制作配置文件。

2.2 拨号上网

Dongle 拨号上网我们使用的是 Android 自带的 3G 上网流程。（相应的流程都是 Android 标准的，可自行网上查看）

主要涉及代码：

hardware/ril/ril-rk29-dataonly

external/ppp/chat

external/ppp/pppd

ril-rk29-dataonly 是 RK 根据 Android 上网流程实现的 vendor ril，用于和 dongle 通信，接收 Android 命令并调用 chat 和 ppd 实现网上功能。chat 是拨号工具。pppd 是点对点协议(Point to Point Protocol)：

3 Android 部分移植涉及到的文件目录

3.1 Vendor 目录

该目录下的 vendor/rockchip/common/phone/文件夹内包含了 usb_modeswitch、chat、RIL 库的编译以及 PPP 拨号脚本的拷贝，这些拷贝的文件最终在 system/etc/ 目录下，如果手动改写了这些文件，可以通过一下命令修改权限:

chown root /system/bin/pppd

chmod 4755 /system/bin/pppd

chown root /system/bin/chat

chmod 4755 /system/bin/chat

chmod 755 /system/etc/ppp/ip-up

3.2 hardware/ril

负责 RILD 的启动以及 ril-rk29-dataonly 库的启动。

Rild 服务是在 init.rc 中开启。

```
service ril-daemon /system/bin/rild
socket rild stream 660 root radio
socket rild-debug stream 660 radio system
user root group radio cache inet misc
```

RILD 目录是 google 原生态目录，RK 做了如下修改：

RILD 启动方式，

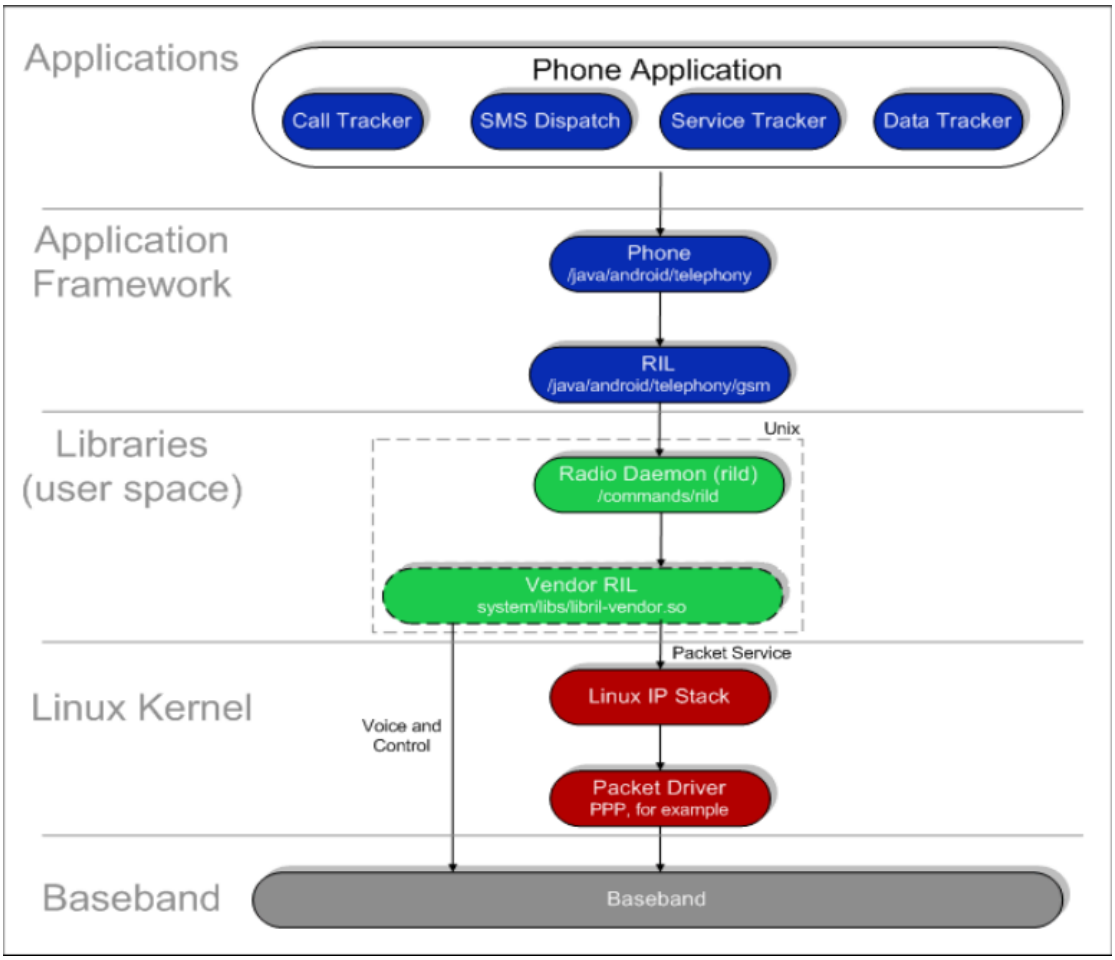
RIL so 文件调用，

3G dongle VID PID 识别，

当模块驱动被内核加载并正确驱动后，3G 模块才能进行无线通信功能的应用开发。这在 Android 系统下称为无线接口层——RIL。即该文档所要说明的 RIL 驱动。android 的 ril 位于应用程序框架与内核之间，分成了两个部分，一个部分是 rild,它负 socket 与应用程序框架进行

通信。另外一个部分是 Vendor RIL，通过这两种方式与 radio 进行通信。通信通道有两个，AT 指令通道和用于传输数据包的通道，数据通道用于上网功能。 也就是 RK 提供的 RIL 驱动，是实现通信业务的核心功能模块，AT 通道用于直接与模块通信，控制模块。

对于 RIL 的 java 框架部分，也被分成了两个部分，一个是 RIL 模块，这个模块主要用于与下层的 rild 进行通信，另外一个是 Phone 模块，这个模块直接暴露电话功能接口给应用开发用户，供他们调用以进行电话功能的实现。这是属于 Android 应用程序的开发部分。



所以 RIL 驱动模块，必需是针对不同的 3G 模块（dongle）指令和通信业务功能定制的，以满足不同应用需求。建议开发人员先了解下“3G 模块的特点和构造”。

3.3 system/vold

vold 负责监视系统是否有 3G dongle 设备出现，并调用 usb_modeswitch 进行 USB 模式切换,包含以下源文件:

G3dev.cpp ， G3dev.h ,Misc.h ,Misc.cpp,MiscManage.cpp ,MiscManager.h, NetlinkHander.cpp

3.4 external/ppp

该目录包含 pppd 程序和 chat 程序，pppd 辅助 PPP 进行 LCP/NCP 的配置以及身份证，chat 程序则辅助 pppd 程序进行拨号，pppd 可执行程序位于/system/bin/目录，pppd 的参数很多，其中，“/dev/ttyUSB*”就是用于 PPP 协商的设备节点，“connect”后所带的是拨号脚本，“disconnect”后所带的是断开连接的脚本，这两个脚本都是通过 chat 程序来执行的，其它的参数，可以查看 pppd 帮助文档。当 ppp 协商完成后，会调用/etc/ppp/ip-up 脚本来设置 android 中的属性值 IP、DNS 等。

3.5 frameworks/opt/telephony

该目录主要在原生的基础上做修改以符合 dongle 的需求，具体可以 git log 查看提交记录。

3.6 packages/services/Telephony

该目录主要在原生的基础上做修改以符合 dongle 的需求，具体可以 git log 查看提交记录。

4 Dongle 相关日志的获取

4.1 使用串口抓取 log 信息

打开串口，输入一下命令，并把串口输出的信息保存成文件

logcat –b radio &　　//ril log

logcat –s pppd &　　//ppp 拨号 log

logcat –c –b radio & //清除以前 raido log

4.2 使用 adb 工具抓取 log 信息

打开 adb shell，输入以下命令：

$ logcat –b radio > /data/radio.log &

$ logcat –s pppd > /data/pppd.log &

抓取 kernel 的打印：

cat /proc/kmsg > /data/kernel.log &

退出 adb shell，把机器中的 log 文件 pull 到本地

adb pull /data/*.log　d:\

5 Dongle 常见问题

5.1 无信号图标

在串口或者 ADB 上输入 logcat –b radio & 出现"not support modem"，说明目前 RIL 库不支持该 dongle。

5.2 出现 3G 图标但是上不了网

1.请先检查 ppp 网络接口是否存在：

busybox ifconfig

ppp0 Link encap:Point-to-Point Protocol

inet addr:10.119.45.174 P-t-P:192.200.1.21 Mask:255.255.255.255

UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1

RX packets:4 errors:0 dropped:0 overruns:0 frame:0

TX packets:7 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:3

RX bytes:58 (58.0 B) TX bytes:135 (135.0 B)

A. ppp0 存在，说明 3G 网络连接还存在，此时再作如下检查：

使用 ping 命令来检查网络情况：

ping -c 4 www.baidu.com

a). ping 网络正常（正常情况下响应时间几十个毫秒），则可能是上层浏览器的问题，请检查或者更换其它浏览器

b). ping 不通，则可能是当前的网络存在异常，比如信号弱、或者网络拥塞，获取的 DNS 不正确；可以通过 getprop 查看 DNS，net.dns1 和 net.dns2,目前 RK 的 RIL 库中有 DNS 检测的功能，当获取到 10.11.12.13 或者 10.11.12.14 无效的 DNS，会断开上次拨号连接重新拨号，一般后两次拨号都会获取到正确的 DNS。

B. ppp0 不存在，说明 3G 网络连接已经断开，但上层没有接收到相应消息，错误认为 3G 连接还存在。

2.SIM 卡无数据业务。

5.3 有信号无 3G 图标

1. APN 信息是否正确

进入设置-> 无线和网络->接入点名称 查看是否有 APN，如果没有，可能是没有正确拷贝 apns-conf.xml 请确认 device/rockchip/rkxxsdk /common/rkxxsdk.mk

PRODUT_COPY_FILES:=

Device/rockchip/rkxxsdk/

common/phone/etc/apns-full-conf.xml:system/etc/apns-conf.xml

检查该文件中是否包含所使用的运营商 APN 信息，如果没有，添加上相应的 APN 信息

比如：<apn carrier="Operator" mcc="" mnc="" apn="" type="default,supl,mms"/>

使用 MCC/MNC 来确认， 而 MCC/MNC 的值是通过模块查询到的 IMSI 码的前五位来确定的。

2. SIM 卡是否有数据流量。

3. SIM 卡与 dongle 是否匹配。

4. 若中途有单独修改过 apns-full-con.xml，需要单独 mmm 模块编译 packages/providers/TelephonyProvider/src/com/android/providers/telephony 目录或者去掉 out 目录重新编译 system.img，否则 APN 不会重新生成 telephony.db 数据库,导致系统找不到 APN 信息。

5.4 radio log 中不断打印“Do not switch user to radio”

这是由于 RIL 库路径不正确，在串口或者 adb 中输入 getprop 查看 gsm.version.ril-impl 属性值，是否正确调用了 RIL 库，

[gsm.version.ril-impl]: [libril-rk29-dataonly.so 2.2.08]

检查系统路径中是否有：/system/lib/libril-rk29-dataonly.so

5.5 识别不到 dongle

在/dev 下没有找到 ttyUSB* 设备，此时可通过观察内核 LOG 来定位问题：

1.USB 设备枚举失败或者系统根本就没有发现 USB 设备，此时应检查硬件电路

2.USB 枚举成功，但没有注册到 ttyUSB*设备，此时应检查内核：

a) 内核没有开启 usb serial 功能

b) 内核代码中的 usb serial 相应驱动中没有添加该设备的 VID/PID，请修改 kernel/drivers/usb/serial/option.c，在数组 static struct usb_device_id option_ids[] 的末尾添加上新设备的 VID/PID 。USB 枚举成功，且相关配置且 ID 都已添加，但还是不出来 ttyUSB*设备，此时可观察系统是否有对它执行 usb mode switch，可通过 logcat–s Vold & 观察是否有调用了 usb_modeswitch 程序，如果没有执行，则检查如下：

1). 检查一些必要的文件是否存在：

ls /system/bin/usb_modeswitch

ls /etc/usb_modeswitch.d

2). VOLD 中关于 usb_modeswitch 这部分的代码没有被编译，可查看 Vold 的 log 中是否有“Start Misc devices Manager…”的字样，如果没有这串字符，请检查你的/system/vold/下的代码。

5.6 Dongle 上电硬件排查

在 Dongle 插入瞬间会有塌陷和较为客观的瞬态电流；在 Dongle 插入后，OTG_5V 输出趋于稳定，出现/dev/ttyUSB*节点，浏览网页时电流小于 200mA，输出电压波动峰峰值小于 0.1V。如果在使用过程中或者插上 3G dongle 后 ttyUSB 设备节点出现，然后又消失，可能是硬件供电不足或者电压塌陷引起，导致 3G dongle 不工作。Dongle 插入瞬间都会有比较大的瞬间电流和电压塌陷，如果持续时间较长，会对 dongle 的识别和使用造成影响。可以使用外部供电的方法来排查是否 OTG 供电是否有问题。

3G dongle 机器休眠前后的 DP 电压说明如下：

3G dognle 待机唤醒后就开始传送数据了，如果是高速的 3G dongle，唤醒后高电平有 0.4V 左右。如果是全速的 3G dongle，唤醒后高电平会有 3V 左右。其他的 dongle 二级待机时 3G dongle 的 DP 电平正常一直为高，大概 3V，唤醒后为低，大概 0.4V,如果电平出现异常，USB 会重新去枚举 ttyUSB 节点，3G 会重新去初始化一些 AT 指令，在 UI 界面上就会出现 3G 图标消失一会儿才会出现。

Log 如果是出现如下信息：

DWC_OTG: dwc_otg_core_host_init: Halt channel 4

DWC_OTG: dwc_otg_core_host_init: Halt channel 5

DWC_OTG: dwc_otg_core_host_init: Halt channel 6

DWC_OTG: dwc_otg_core_host_init: Halt channel 7

DWC_OTG: dwc_otg_core_host_init: Halt channel 8

DWC_OTG: dwc_otg_core_host_init: Halt channel 9

DWC_OTG: dwc_otg_core_host_init: Halt channel 10

DWC_OTG: dwc_otg_core_host_init: Halt channel 11

DWC_OTG: dwc_otg_core_host_init: Halt channel 12

DWC_OTG: dwc_otg_core_host_init: Halt channel 13

DWC_OTG: dwc_otg_core_host_init: Halt channel 14

是 USB 驱动问题出问题概率较大。

Log 如出现如下：

hub 2-0:1.0: Cannot enable port 1. Maybe the USB cable is bad?

hub 2-0:1.0: Cannot enable port 1. Maybe the USB cable is bad?

hub 2-0:1.0: Cannot enable port 1. Maybe the USB cable is bad?

hub 2-0:1.0: Cannot enable port 1. Maybe the USB cable is bad?

该问题是 USB 信号问题，请检查 USB 供电电路的电信号是否符合 USB spec 规定的值，比如 USB Host 电压，DM/DM 信号，USB 阻抗是否匹配等。