密级状态：绝密(　　)　　秘密(　　)　　内部资料()　　公开(√)

3G for RockChip

文件状态： ［√］草稿 ［　］正式发布 ［　］正在修改	文件标识：	ROCKCHIP-MID-3G
	当前版本：	1.0.0
	作　　者：	许学辉
	完成日期：	2014-1-25

RockChip MID 3G 联系人：

许学辉　　xxh@rock-chips.com

历 史 版 本

版本	日 期	描 述	作 者	审核
V1.0	2014-1-25	建立文档，主要介绍 RK 平台上 MID 3G 模块移植方法	许学辉	

目录

1. 文档功能描述....4
2. 软件版本....4
2.1 支持数据业务的 RIL 软件版本号....4
3. 配置说明....4
3.1 配置 device.mk....4
3.2 kernel 配置....5
3.2.1 4.1 系统 3G 模块通用驱动选择....5
3.2.2 配置上电，下电....6
3.2.3 4.1 系统内置 3G 模块/driver/misc 驱动文件配置....6
3.2.4 4.2 系统、4.4 系统内核配置....9
3.2.5 开启内核 PPP 协议栈....10
3.2.6 3G dongle 内核配置....11
3.3 3G 内置模块硬件....12
4. 3G 功能模块 Android 3G 移植涉及到的文件目录....12
4.1 device 目录....12
4.2 hardware/ril hardware/rild3 目录....13
4.3 system/vold 目录....14
4.4 external/ppp 目录....15
4.5 frameworks/base/telephony 目录....15
4.6 Packages/apps/Phone 目录....15
4.6 Packages/apps/Setting 目录....15
4.6 Packages/apps/MMS 目录....15
4.7 frameworks/base/packages/SystemUI 目录....15
4.7 frameworks/opt/telephony 语音通路....16
5. 3G 常见问题....19
5.1 无信号图出现....19
5.2 内置模块 SIM 卡不识别或检测不到信号....20
5.3 出现 3G 图标但是上不了网....20
5.4 有信号无 3G 图标出现....21
5.5 radio log 中不断打印“Do not switch user to radio”....22
5.6 radio log 中不断打印“wait 3G device……”....22
5.7 识别不到 modem 设备....22
5.8 3G dongle 硬件上电排查....23
5.9 如何打开内核的 USB 驱动开关....24
5.10 OTG 线排查....24
5.11 天线匹配....24
5.11 国外 3G dongle 问题排查....24
6. USB 模块的配置说明....25
6.1 USB host 相关配置....25
6.1.1 USB Mass Storage....27
6.1.2 USB Serial Converter....28
6.2 USB gadget 配置....28

6.3 芯片 USB 控制器配置.....28
6.3.1 USB 1.1 host 控制器.....28
6.3.2 USB 2.0 OTG 控制器配置.....29
7. 3G 相关日志信息的获取.....30
7.1 使用串口捕捉 LOG 信息.....30
7.2 使用 adb 工具捕捉 LOG 信息.....30
8. Android 4.2.2 3G 内置模块电源选项[可选].....30
9. Android 4.1/Anroid 4.2 wifi-only 配置.....31
10. 3G 通话项目设置.....32
11. RIL 库移植说明.....32

1. 文档功能描述

本文档主要描述了 3G 模块以及 3G dongle 的移植、3G 功能所涉及到的文件/目录的各自用途、常见问题的解决方法,本文档适合于在 RK 平台上进行 3G 功能的移植,仅供 MID 内部交流使用。

2. 软件版本

2.1 支持数据业务的 RIL 软件版本号

方法一：adb shell 查看

RIL 版本查看：串口或 ADB 中输入 logcat –b radio & 看打印信息，例如 RK RIL 库版本信息如下：

I/RILC (74): RIL Daemon version: ril-rk29-dataonly v2.2.08

方法二：通过源码查看

进入 device/rockchip/rk30sdk/phone/lib$ 目录输入一下命令

strings libril-rk29-dataonly.so | grep dataonly 即可查看

3. 配置说明

3.1 配置 device.mk

文件路径：device/rockchip/rkxxsdk/common/ rk30_phone.mk 确认以下内容，默认情况下 SDK 这些都已经配置好了，如果需要新添加 RIL 库支持，可以在该文件中修改。

```
PRODUCT_COPY_FILES += \
    $(CUR_PATH)/phone/etc/ppp/ip-down:system/etc/ppp/ip-down \
    $(CUR_PATH)/phone/etc/ppp/ip-up:system/etc/ppp/ip-up \
    $(CUR_PATH)/phone/etc/ppp/call-pppd:system/etc/ppp/call-pppd \
```

$(CUR_PATH)/phone/etc/operator_table:system/etc/operator_table

PRODUCT_COPY_FILES += \

$(CUR_PATH)/phone/bin/usb_modeswitch.sh:system/bin/usb_modeswitch.sh \

$(CUR_PATH)/phone/bin/usb_modeswitch:system/bin/usb_modeswitch \

$(CUR_PATH)/phone/lib/libril-rk29-dataonly.so:system/lib/libril-rk29-dataonly.so \

Overlay 配置

frameworks/base/core/res/res/values/config.xml 文件

<!-- This device is not "voice capable"; it's data-only->

<bool name="config_voice_capable">false</bool>

3.2 kernel 配置

3.2.1 4.1 系统 3G 模块通用驱动选择

进入 menuconfig 的配置，开启内置模块的驱动，先选上 HOST2.0 Device Drivers->Misc devices ->选中 RK29 support Modem ---> 选择支持 3G 内置 modem，修改内置模块的 IO 配置，配置 power 引脚的 gpio，文件在 board-rk30-sdk.c,

```
#ifdef CONFIG_RK29_SUPPORT_MODEM

#define RK30_MODEM_POWER          RK30_PIN4_PD1
#define RK30_MODEM_POWER_IOMUX    rk29_mux_api_set(GPIO4D1_SMCDATA9_TRACEDATA9_NAME, GPIO

static int rk30_modem_io_init(void)
{
    printk("%s\n", __FUNCTION__);
    RK30_MODEM_POWER_IOMUX;

    return 0;
}

static struct rk29_io_t rk30_modem_io = {
    .io_addr    = RK30_MODEM_POWER,
    .enable     = GPIO_HIGH,
    .disable    = GPIO_LOW,
    .io_init    = rk30_modem_io_init,
};

static struct platform_device rk30_device_modem = {
    .name   = "rk30_modem",
    .id     = -1,
    .dev    = {
        .platform_data  = &rk30_modem_io,
    }
};
#endif
```

3.2.2 配置上电，下电

文件在 drivers/misc/rk29_modem/rk29_modem.c

int rk29_modem_change_status(struct rk29_modem_t *rk29_modem, int status)

{

int ret = 0;

switch(status)

{

case MODEM_DISABLE:

rk29_modem_turnon(rk29_modem->modem_power, 0);

gpio_direction_output(power 的 gpio, 0); ----------→3G 模块 power 的 gpio

break;

case MODEM_ENABLE :

rk29_modem_turnon(rk29_modem->modem_power, 1);

gpio_direction_output(power 的 gpio, 1); ----------→3G 模块 power 的 gpio

break;

case MODEM_SLEEP:

ret = -1;

case MODEM_WAKEUP:

ret = -1;

break;

}

return ret;

}

注意：3G 模块上电和下电需要根据模块本身设计要求，控制上电和下电时序。休眠唤醒采用硬件休眠机制。

3.2.3 4.1 系统内置 3G 模块/driver/misc 驱动文件配置

如果不使用 3G 模块通用驱动，可以选择/driver/misc 下的驱动文件，目前 RK30 3G 模块驱动文件有：MU509、MW100、MT6229、SEW868，以下以 MU509/SEV868 为例，

MU509 内置内核配置：

终端内执行 make menuconfig

```
Device Drivers --->
        <*> Misc devices --->
            <*> MU509 modem control driver
```

确认 kernel/drivers/usb/serial/option.c 文件中以下内容：

```
#define HUAWEI_VENDOR_ID 0x12D1
#define HUAWEI_PRODUCT_E600 0x1001
{ USB_DEVICE(HUAWEI_VENDOR_ID, HUAWEI_PRODUCT_E600)},
if(status == -EPROTO && err_times++ >10)
{
err_times = 0;
printk("%s,recieve -71 error more than 10 times,so reset usb\n",__FUNCTION__);
usb_queue_reset_device(port->serial->interface); return;
}else
 err("%s : error %d",__func__, status);
```

在项目的 board 文件中的添加 MU509 的配置，文件目录：kernel/arch/arm/mach-rk29/

配置如下：

```
#if defined(CONFIG_MU509)
#include <linux/mu509.h>
#endif
#if defined(CONFIG_MU509)
static int mu509_io_init(void)
 {
    return 0;
 }
 static int mu509_io_deinit(void)
 {
     return 0;
```

```
 }
struct rk29_mu509_data rk29_mu509_info =
{
 .io_init = mu509_io_init,
 .io_deinit = mu509_io_deinit,
 .modem_power_en = RK29_PIN6_PC2, //给模块供电的 DCDC 电路使能脚
.bp_power = RK29_PIN6_PB1, //模块电源管脚
.bp_power_active_low = 1,
.bp_reset = RK29_PIN6_PC7, //模块复位管脚
.bp_reset_active_low = 1,
.bp_wakeup_ap = RK29_PIN0_PA4, //BP 唤醒 AP 管脚
.ap_wakeup_bp = RK29_PIN2_PB3, //AP 唤醒 BP 管脚
 };
```

驱动配置好后，当模块上电，会出现/dev/ttyUSB*节点，RIL 库使用 TTYUSB2, TTYUSB0 这两个节点来作为 modem 口和 AT 口。注意：3.5.3 和 3.5.1 不可同时配置，只能选择其中一种配置。

SEW868 内核配置：

执行 Make menuconfig

```
Device Drivers    --->
        <*> Misc devices    --->
            <*>     3G module for phonepad    --->
                Select 3G Module (SEW868)    --->选上 sew868 驱动。
```

需要在对应项目的 board 中添加以下代码，具体可以参考 arch/arm/mach-rk30/board-rk30-sdk.c，注意各个 GPIO 的配置需要与电路图一致。

```
#if defined(CONFIG_SEW868)
#include <linux/sew868.h>
#endif
#if defined(CONFIG_SEW868)
static int sew868_io_init(void)
{
    rk30_mux_api_set(GPIO2B6_LCDC1DATA14_SMCADDR18_TSSYNC_NAME,
GPIO2B_GPIO2B6);
    rk30_mux_api_set(GPIO4D2_SMCDATA10_TRACEDATA10_NAME, GPIO4D_GPIO4D2);
    rk30_mux_api_set(GPIO4D4_SMCDATA12_TRACEDATA12_NAME, GPIO4D_GPIO4D4);
    return 0;
}
```

```
static int sew868_io_deinit(void)
{
	return 0;
}
struct rk30_sew868_data rk30_sew868_info = {
	.io_init = sew868_io_init,
	.io_deinit = sew868_io_deinit,
	.bp_power = RK30_PIN6_PB2, //给模块供电的 DCDC 电路使能脚
	.bp_power_active_low = 1,
	.bp_sys = RK30_PIN2_PB6, //模块电源控制管脚
	.bp_reset = RK30_PIN4_PD2, //模块复位管脚
	.bp_reset_active_low = 1,
	.bp_wakeup_ap = RK30_PIN4_PD4, //bp 唤醒 ap 管脚
	.ap_wakeup_bp = NULL,
};

struct platform_device rk30_device_sew868 = {
		.name = "sew868",
		.id = -1,
	.dev		= {
		.platform_data = &rk30_sew868_info,
	}
	};
#endif#if defined(CONFIG_SEW868)
	&rk30_device_sew868,
#endif
```

确认串口起始号，如果 usb-serial.c 文件修改了串口起始号，默认的串口是给手机项目使用的，修改如下：

```
--- a/drivers/usb/serial/usb-serial.c
+++ b/drivers/usb/serial/usb-serial.c
@@ -135,7 +135,7 @@ static struct usb_serial *get_free_serial(struct usb_serial *serial,
	if (SEW868_USB)
		a= SEW868_USB_PORT;
 #endif
-	for (i = a; i < SERIAL_TTY_MINORS; ++i) {
+	 for (i = 0; i < SERIAL_TTY_MINORS; ++i) {
		if (serial_table[i])
			continue;
```

3.2.4 4.2 系统、4.4 系统内核配置

在 4.2 系统以后，为了方便配置 3G 模块，可以直接使用 bp_auto 机制来进行模块的配置，语音和数据都可以使用这套机制。

内核驱动 menuconfig 需要打开宏 CONFIG_BP_AUTO=y ，如下

Device Drivers ---><*> Misc devices ---> <*> voice modem support

```
        Texas Instruments shared transport line discipline  --->
< >     STMicroeletronics LIS3LV02Dx three-axis digital accelerometer (I2C)
<*>     voice modem support
< >     3G module audio for phonepad  --->
[ ]     RK2928 callpad audio switch
```

3G 模块 board info 参看 arch/arm/mach-rk30/board-rk30-sdk.c 文件中 3G 模块（宏 CONFIG_BP_AUTO 宏括内容）,如果使用的控制管脚不一样的话，需要修改以下相关管脚定义：

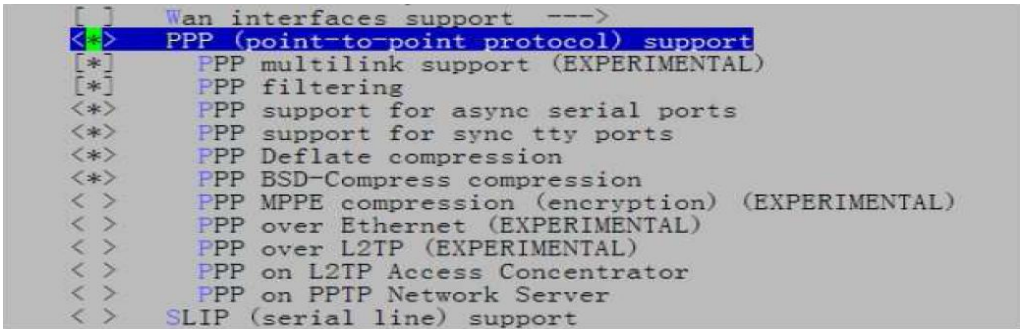
该结构体定义了全部的 3G 功能模块管脚，有的 3G 模块没有对应管脚，直接设置为 UNK

```
        .init_platform_hw  = bp_io_init,
        .exit_platform_hw  = bp_io_deinit,
        .get_bp_id                 = bp_id_get,
        .bp_power          = RK30_PIN6_PB2,     // 3g_power
        .bp_en             = RK30_PIN2_PB6,     // 3g_en
        .bp_reset          = RK30_PIN4_PD2,
        .bp_usb_en         = BP_UNKNOW_DATA,    //W_disable
        .bp_uart_en        = BP_UNKNOW_DATA,    //EINT9
        .bp_wakeup_ap      = RK30_PIN4_PC6,     //2G/3G 模块唤醒 AP 脚
        .ap_wakeup_bp      = RK30_PIN4_PC4, //AP 唤醒 2G/3G 模块脚
        .ap_ready          = BP_UNKNOW_DATA,    //
        .bp_ready      = BP_UNKNOW_DATA,
        .gpio_valid        = 1,       //if 1:gpio is define in bp_auto_info,if 0:is not use gpio in
bp_auto_info
    };
```

3.2.5 开启内核 PPP 协议栈

```
        Userspace binary formats  --->
        Power management options  --->
    [*] Networking support  --->
        Device Drivers  --->
        File systems  --->
        Kernel hacking  --->
        Security options  --->
```

```
[*] Multiple devices driver support (RAID and LVM)  --->
[*] Network device support  --->
[ ] ISDN support  --->
< > Telephony support  --->
```



备注：内核 ppp 协议实现，在移植的时候多不需要修改

3.2.6 3G dongle 内核配置

默认情况下，SDK 上内核都已经开启了支持 DONGLE 的相关配置，见 “USB 模块的配置说明 ”，3G dongle 跟内核 USB 是一脉相承的，区别在于 DOGNLE 属于网络设备，可以使用内核自带的 CDC 驱动，usb serial 驱动,，在实际调试 dongle 的时候，可以开启 CDC ACM 驱动选项，也可以上网查询 dongle 驱动的相关信息，将其移植到我们的内核上面即可。因此移植的时候，主要关注 USB NET 相关部分

USB 驱动目录：

drivers/usb/serial 目录

drivers/usb/storage 目录

drivers/usb/dwc_otg 目录

drivers/net/usb 目录

3G dongle 和 3G 模块常常常修改的驱动文件如下：

drivers/usb/serial/option.c //USB Serial 相关

drivers/usb/serial/usb-serial.c

drivers/base/devices_filter.h

drivers/usb/storage/unusual_devs.h //USB 存储设备相关

drivers/usb/storage/initializers.c

drivers/usb/storage/initializers.h

drivers/usb/dwc_otg/dwc_otg_cil_intr.c //otg 相关

drivers/net/usb/cdc_ether.c //CDC 以太网，ECM 驱动相关

备注：修改 USB 驱动读写缓存区大小，如果有些版本内核默认太小，对 3G 高速下载会有所限制，需将其增大，可以参考如下方式修改：

diff --git a/drivers/usb/serial/usb-serial.c b/drivers/usb/serial/usb-serial.c

index c20bd51..ac7bf34 100755

```
--- a/drivers/usb/serial/usb-serial.c
+++ b/drivers/usb/serial/usb-serial.c
@@ -76,7 +76,9 @@ static int SEW868_USB = 0;
        via modprobe, and modprobe will load usbserial because the serial
        drivers depend on it.
 */
+static ushort maxRSize=4096;
+static ushort maxWSize=1024;
+static ushort maxISize=1024;
 static int debug;
 /* initially all NULL */
 static struct usb_serial *serial_table[SERIAL_TTY_MINORS];
@@ -948,6 +950,8 @@ int usb_serial_probe(struct usb_interface *interface,
                        }
                        buffer_size = max_t(int, serial->type->bulk_in_size,
                                                le16_to_cpu(endpoint->wMaxPacketSize));
+                       if(buffer_size<maxRSize)
+                       buffer_size=maxRSize;
```

3.3 3G 内置模块硬件

内置模块涉及到的电路模块：供电电路、串口电路、USB 电路、休眠唤醒电路、SIM 卡电路。这里以 MU509 内置模块为例，首先确认 VCC3G 端供电正常，可以在 VCC3G 端接一个示波器观察整个开机过程电源是否有塌陷；注意：使用单电池供电的请严格按照我们参考电路进行设计，使用双电池的硬件上设计推荐将 VCC3G 稳定在 3.8V；对于 USB 电路确认共模电感已经正确焊接，确认硬件上 DM、DP 端有 15k 电阻，用万用表测量实际硬件；由于模块休眠唤醒采用硬件休眠机制，因此需要确认休眠唤醒管脚硬件上已正确连接。

4. 3G 功能模块 Android 3G 移植涉及到的文件目录

4.1 device 目录

该目录下的 Device/rockchip/rkxxsdk/common/phone/文件夹内包含了 usb_modeswitch 程序、切换文件、RIL 库、PPP 拨号脚本、apn 配置文件, 这些脚本和 BIN 文件都在这个目录进

行拷贝，并最终在 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

modeswitch：用于切换 3g usb dongle 的光驱模式。注：需要 SCSI 驱动的支持

usb_modeswitch.d 是 usb 模式切换使用的切换脚本，

ppp 文件夹包含 PPPD 拨号需要的脚本文件，

libil-rk29-dataonly.so 是支持数据业务的 ril 库，

目前支持的 3G 模块和 3G dongle 详见<<3G_Support_List.xls>>支持列表。

4.2 hardware/ril hardware/rild3 目录

负责 RILD 的启动以及各个 RIL 库的启动，动态切换 RIL 库和各种 RIL 请求都封装在该目录下的各个文件中

RILD3 是 RK 新增加目录，原生态没有，主要用于 3G Dongle 的使用（语音项目），其实就是 RILD 的阉割版本

Rild 服务是在 init.rc 中开启的，也可以在 init.rc 中关闭该服务（主要用于有 WIFI_ONLY 项目）

```
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，

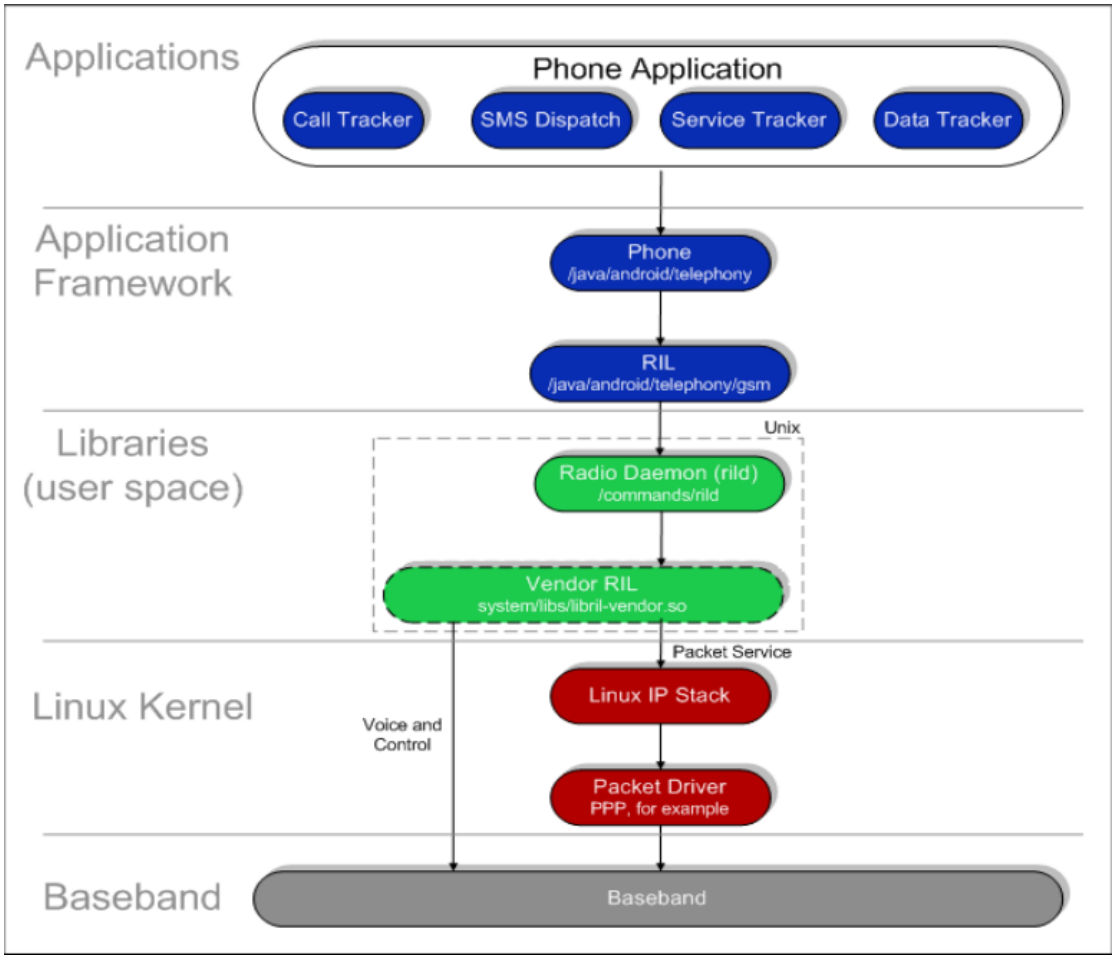
3G dongle VID PID 识别，

内置 3G 模块上电，下点，Reset 操作，其实是通过 IOCTL 来实现的

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，以支不同应用需求。建议开发人员先了解下“3G 模块的特点和构造”。

4.3 system/vold 目录

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

该目录，只有 3G dongle 才能用到，内置 3G 模块不需要，见 MID 该目录提交记录

4.4 external/ppp 目录

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。

该目录，修改较少，MID主要添加了DEBUG功能，查看协议数据包的分发流程

该目录，没有提交记录，在调试3G模块的时候才会用到

4.5 frameworks/base/telephony目录

负责3G与RIL通信，完成各种phone 逻辑。

该目录， RK MID主要修改了信号查询，

手机组主要修改了来电时候，对数据业务的监听

4.6 Packages/apps/Phone目录

Phone应用,Radio状态控制，可以在该目录中添加3G应用，比如添加内置模块选项，控制内置模块电源。

MID 主要移植了3G模块，Radio状态的控制，内核节点上电下点的控制，

手机组 主要修改来点过程电话正常挂断的BUG

4.6 Packages/apps/Setting目录

- 该目录，MID在4.2系统上增加弹出DONGEL窗口的提示，配合RILD3使用

4.6 Packages/apps/MMS目录

Phone短信应用

该APK是不支持所有的3G模块短信功能，3G模块都需要配合RIL库，对该APK进行修改

具体看该目录下提交记录，该目录MID和手机组都在维护

MID添加了发完彩信后，无法上网的问题

手机主要添加了多媒体附件的发送

4.7 frameworks/base/packages/SystemUI目录

负责3G UI,比如信号图标以及网络名称显示。

MID修改了

packages/SystemUI/src/com/android/systemui/statusbar/policy/NetworkController.java //信号图标出现时间

telephony/java/android/telephony/SignalStrength.java 信号显示，ASU和DBM 电平转换

具体看该目录下提交记录

4.7 frameworks/opt/telephony语音通路

该目录主要负责语音通路切换，涉及到修改的源文件比较多，针对对数据，短信，APN，语音都做过改动，具体改动也请看提交记录，对于 RIL部分详细说明如下：

音频通路切换在 ril 中新增加了两个 request ID，分别是设置音频通路模式和设置音频通路模式的下的音量大小，如下：

#define RIL_REQUEST_SET_AUDIO_MODE 110 //设置模式

#define RIL_REQUEST_SET_AUDIO_MODE_VOLUME 112 //设置音量大小

具体定义在 hardware/ril/include/ril.h 文件中。

要实现音频通路的切换，需要在 ril 代码中实现以上两个 request，实现方法实例及说明如下。

1， 设置音频通路模式

#define RIL_REQUEST_SET_AUDIO_MODE 110 //设置模式

参数： int data[0] 音频通路模式；

音频通路模式有如下 5 种：

```
typedef enum{
 AUDIO_MODE_EARPHONE=0,  //听筒模式
 AUDIO_MODE_SPEAKER,     //免提模式
 AUDIO_MODE_HPWITHMIC,     // 4 段耳机模式（带 MIC）
 AUDIO_MODE_BT ,    //蓝牙模式
 AUDIO_MODE_HPNOMIC,       //3 段耳机模式（不带 MIC）
 AUDIO_MODE_STOPPHONE,     //通话结束
}Modem_Audiotype;
```

具体定义可以参见 hardware/ril/include/ril.h 文件

各种模式下 ril 的具体实现需要根据每个模块的实现而定，下面代码供参考：

```
static void
onRequest (int request, void *data, size_t datalen, RIL_Token t)
{
    case RIL_REQUEST_SET_AUDIO_MODE:
```

```
            ALOGD("got RIL_REQUEST_SET_AUDIO_MODE");
            setAudioMode(data, datalen, t);
        break;
}

static void setAudioMode(void *data, size_t datalen, RIL_Token t){

 //ATResponse *p_response = NULL;
        int err = -1;
        int response;
        char *line = NULL;
        char *report = NULL;
 char *cmd;
 int mode = ((int *)data)[0];

        ATResponse *p_response = NULL;
        int responsea[2] = {0};
 switch(mode){
        case AUDIO_MODE_EARPHONE:
            at_send_command("AT+ESAM=0",NULL);//设置模式
            at_send_command("AT+ESLT=2,200",NULL);//设置该模式下的 MIC 增益
            break;
        case AUDIO_MODE_SPEAKER:
        case AUDIO_MODE_HPNOMIC:
            at_send_command("AT+ESAM=2",NULL);
            at_send_command("AT+ESLT=2,100",NULL);
            break;
        case AUDIO_MODE_HPWITHMIC:
            at_send_command("AT+ESAM=1",NULL);
            at_send_command("AT+ESLT=2,200",NULL);
            break;
        case AUDIO_MODE_BT:
            at_send_command("AT^PCM=1",NULL);      //打开蓝牙
            break;
        case AUDIO_MODE_STOPPHONE:
            break;
        default:
            break;
 }
 at_send_command("AT+ESLT=5,0",NULL);//关闭侧音
 RIL_onRequestComplete(t, RIL_E_SUCCESS, NULL, 0);
 return;
error:
 RIL_onRequestComplete(t, RIL_E_GENERIC_FAILURE, NULL, 0);
```

```
}
```

2,设置音量大小

```
#define RIL_REQUEST_SET_AUDIO_MODE_VOLUME 112     //设置音量大小
```

参数：

```
int mode = ((int *)data)[0];   //音频模式
int index = ((int *)data)[1];   //音频当前值
int iMax = ((int *)data)[2];   //音量最大值
```

在应用层中没每种模式下可调节的音量范围不一样，所以需要根据音量最大值和当前值来算出要设置给模块的音量大小，具体实现可参考如下代码：

```
static void
onRequest (int request, void *data, size_t datalen, RIL_Token t)
{
    case RIL_REQUEST_SET_AUDIO_MODE_VOLUME:
        requestSetAudioModeVolume(data,datalen,t);
     break;.
}

static void requestSetAudioModeVolume(void *data, size_t datalen, RIL_Token t){

 ATResponse *p_response = NULL;
      int err = -1;
      int response;
      char *line = NULL;
      char *report = NULL;
 char *cmd;
 int volume = 0;
 int tmp = 0;
 int mode = ((int *)data)[0];
 int index = ((int *)data)[1];
 int iMax = ((int *)data)[2];
 ALOGD("<-requestSetAudioModeVolume mode=%d,index=%d->",mode,index);
 if(index == 0) index = 1;
 if(index == iMax){
      volume = 0x7fff;
 }else{
      volume = (32767/iMax)*index;
 }

 switch(mode){
      case AUDIO_MODE_EARPHONE:
          tmp = (volume*255)/32767;
          break;
      case AUDIO_MODE_SPEAKER:
```

```
            case AUDIO_MODE_HPNOMIC:
                tmp = (volume*70)/32767;
                break;
            case AUDIO_MODE_HPWITHMIC:
                tmp = (volume*100)/32767;
                break;
            case AUDIO_MODE_BT:
                tmp = (volume*255)/32767;
                break;
            case AUDIO_MODE_STOPPHONE:
            default:
                goto error;
        }

        asprintf(&cmd, "AT+ESLT=4,%d",tmp);
            err = at_send_command(cmd,NULL);
        free(cmd);
            if(err < 0) goto error;

        RIL_onRequestComplete(t, RIL_E_SUCCESS, NULL, 0);

        return;

       error:
        RIL_onRequestComplete(t, RIL_E_GENERIC_FAILURE, NULL, 0);

    }
```

备注：以上修改，每个3G模块都会用到，针对具体模块出现的BUG，以上每个目录都可能涉及到修改，由于3G功能模块相对独立，所有提交都是针对3G功能的，因此在该项目分支查看提交记录即可。

5. 3G 常见问题

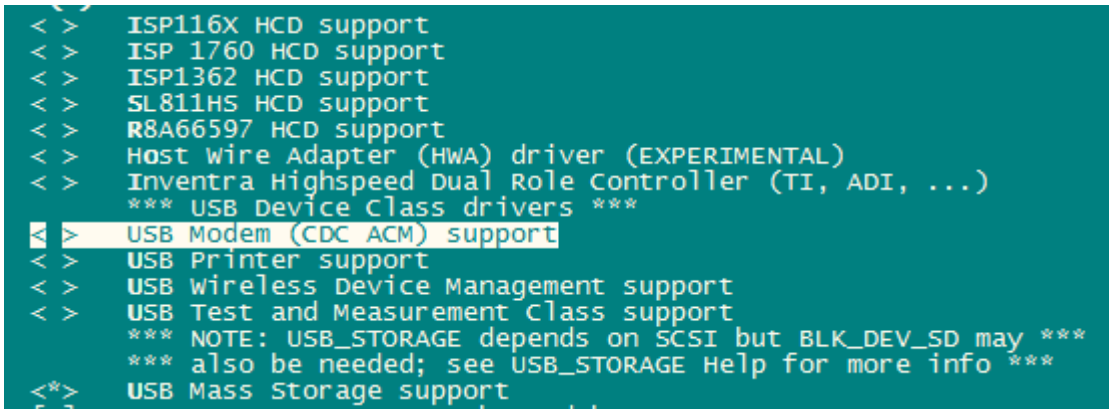
5.1 无信号图出现

1. 在串口或者 ADB 上输入 logcat –b radio & 出现“not support modem”，说明目前 RIL 库不支持该模块或者 dongle，RK 3G dongle 和 3G 模块支持列表详见“RK_3G_Support_List.xls”文档。
2. 在串口或者 ADB 上输入 logcat –b radio & 出现“AT error on at_open”或者“AT handshake failed”，则是由于 AT 通信口不能正常使用引起的。

原因排查：是否开启了 CDC ACM 驱动，目前 RK 3G 驱动中没有使用 CDC ACM 驱

动， 在内核配置中将该选项去掉；若配置正确，请将 dongle 寄给我们分析。

Device Drivers ->USB support->



3. 内置模块上电后，如果检查到没有 SIM 卡，RIL 会关闭模块电源，降低机器功耗，不会出现型号图标。

5.2 内置模块 SIM 卡不识别或检测不到信号

可以通过以下方式排查：

1. 在串口或者 ADB 上输入 logcat –b radio & 查看信号强度，当 CSQ 或者 RSSI 的值为 6 或者更小，说明信号很差，导致网络注册失败，这时需要检查天线，或者到信号好的地方测试。

2. 若是没插 SIM 卡，软件会自动关闭模块电源，UI 上没有信号显示，属于正常情况。

3. 确认 SIM 卡电路硬件上已经正常连接，用万用表测试是否已经正常连接。

4. 查看模块电源供电是否正常，在 VCC3G 端连接示波器观察是否有塌陷现象。

5.3 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，如果一直都是获取到无效 DNS，请通知我们，我们会另外给出解决方案，将系统 DNS 设置成通用的 DNS 地址。

c).当是有 3G dongle 或者模块去连接私有网络或者私有 APN，DNS 服务器也会分配给终端 10.11.13.14/10.11.12.14 IP 地址,这种情况请通知我们去掉 DNS 检查，否则会一直连接不上 3G 网络。

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。

3. PPP 协议栈是否正常开启，见内核配置部分。

5.4 有信号无 3G 图标出现

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 或者内置模块是否匹配。

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。

5.5 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

5.6radio log 中不断打印“wait 3G device……”

检查/dev/ttyUSB*设备节点，如果没有/dev/ttyUSB0，而有/dev/ttyUSB244,/dev/ttyUSB245,只需修改内核 driver/usb/usb-serail.c 将串口起始号变量 a 改为从 0 开始即可，因为目前 MID 和语言平板在一套 SDK 中，这个文件有些许差异，语音平板使用/dev/ttyUSB244,而 MID 代码涉及到的 3G dongle 和 3G modem 在 android 默认是去访问/dev/ttyUSB0 节点，

5.7 识别不到 modem 设备

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：

a). 检查一些必要的文件是否存在：

ls /system/bin/usb_modeswitch

ls /etc/usb_modeswitch.sh

ls /etc/usb_modeswitch.d

b). 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。

5.8 3G dongle 硬件上电排查

当使用支持列表中的 3G dongle 时，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

系 USB 驱动问题出问题概率较大，请将 kernel/drivers/usb/dwc_otg 目录下文件发送给我们分析或者更新到最新的代码。

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?

该问题系 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。

5.9 如何打开内核的 USB 驱动开关

这些开发默认 SDK 上都已经开启了，如果发现 ttyUSB 设备没有正常枚举，可以先检这些配置，内核打开 CONFIG_USB_SERIAL_GENERICCONFIG_USB_SERIAL_OPTION 开关，

make menuconfig----->Device Drivers

USB Support

USB Serial Converter support

将 USB Generic Serial Driver 选上

5.10 OTG 线排查

3G dongle 供电正常，正常出现/dev/ttyUSB*节点，3G 连接不上或者很长时间连接不上，请跟换一根 OTG 线，可能是 OTG 线导致设备节点工作异常。

5.11 天线匹配

对于内置模块，需要对天线进行匹配，匹配后的天线在信号较弱的区域会体现出优势，若 logat –b radio &的 log 中出现

SIGNAL_STRENGTH {99, 99, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1}表明信号强度不可测或者信号未知。

SIGNAL_STRENGTH {9, 99, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1}表明该区域信号强度较弱。以上两种情况会导致模块注册网络失败，请更换天线。

正常情况下型号的强度：SIGNAL_STRENGTH {31, 99, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1}

5.11 国外 3G dongle 问题排查

国外 3G Dongle 在国内测试无法使用的问题排除：

在 PC 上使用 dongle 自带的驱动，如果提示：“SIM card is locked”或者“NO SIM card”等提示，这类需要对 Dongle 解锁，那么这类 dongle 只能使用国外当地的 3G SIM 卡。原因：由于国外 3G dongle 大多数都与运营商绑定，虽然都支持 900/2100 频段，但是 dongle 本身固件对 SIM 卡已经经过烧号处理或者对 Dongle 进行了锁定，无法使用国内 3G SIM 卡注册联通，移动网络。

6. USB 模块的配置说明

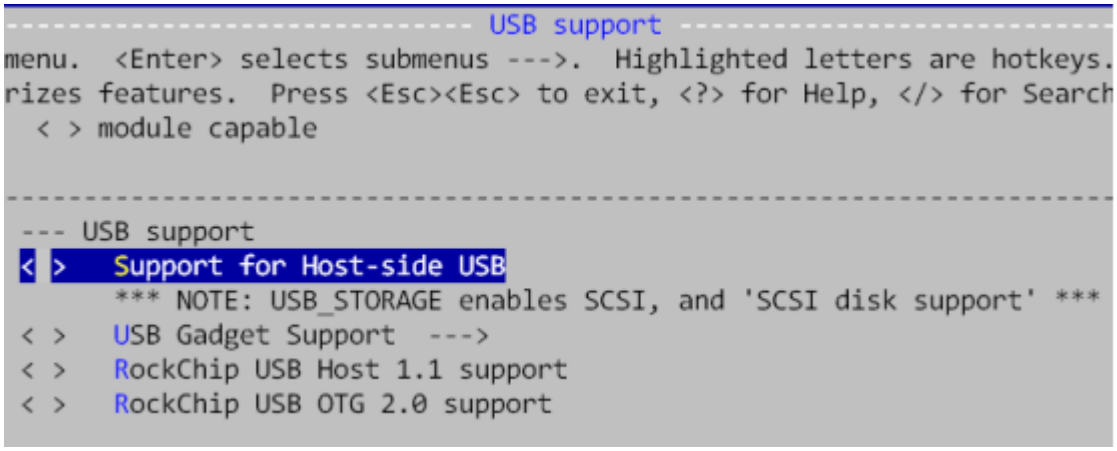
USB 模块的配置位于 kernel 的 make menuconfig

[ ] Device Drivers

[ ] USB support -

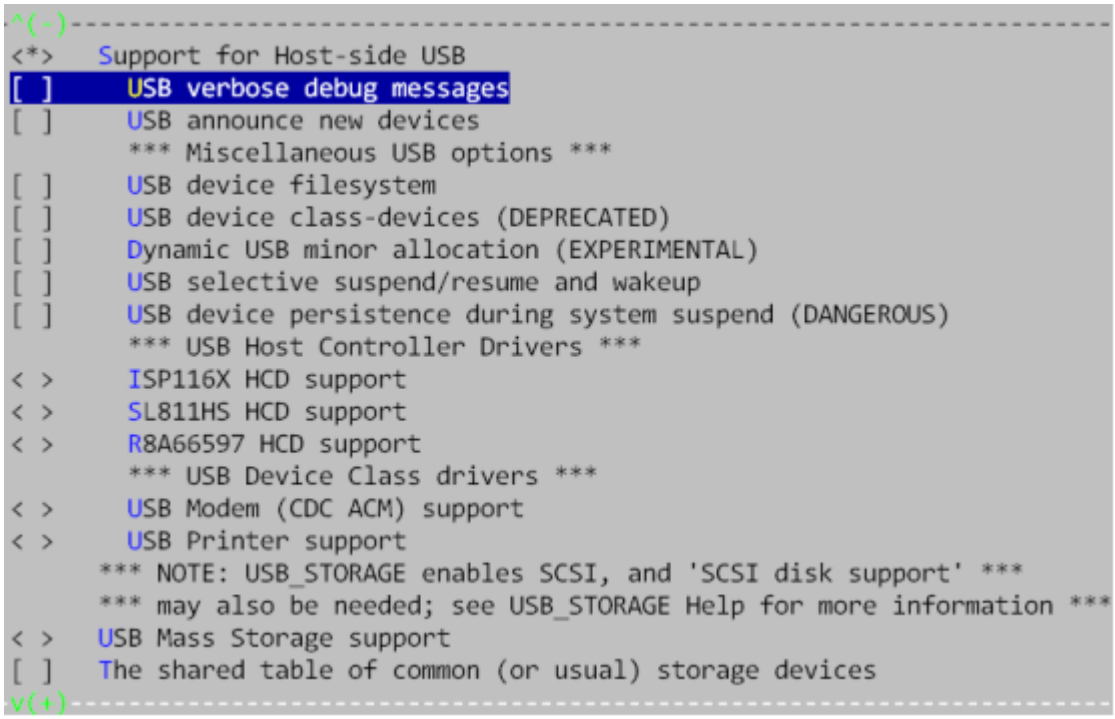
必须选上 USB support 项后才能支持 USB 模块并进行进一步的配置。

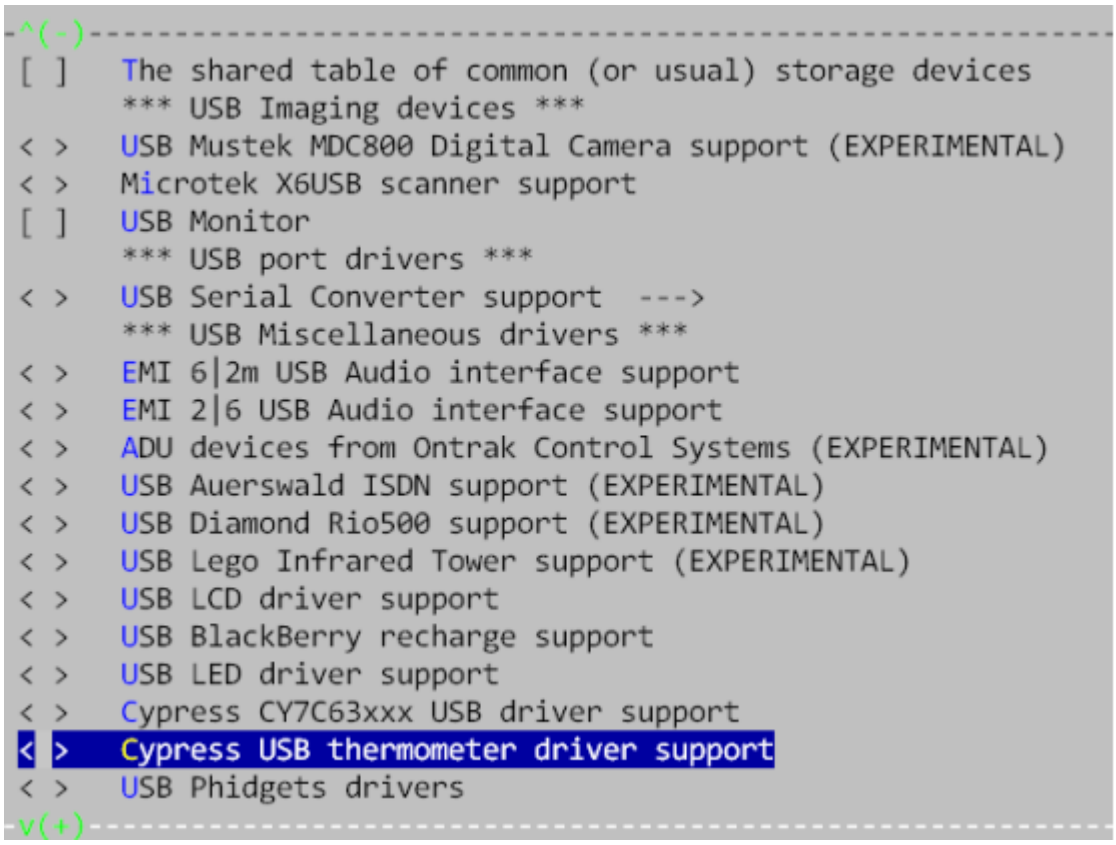
USB support 选项如下，后面详细说明每一项的具体配置。



6.1 USB host 相关配置

需要支持 USB host，首先需要选上< >Support for Host-side USB 项，然后会有出现很多 host 相关的配置，我们应根据实际需求来配置。如产品不需支持 host，应不选这一项。





```
< >   USB Phidgets drivers
< >   Siemens ID USB Mouse Fingerprint sensor support
< >   Elan PCMCIA CardBus Adapter USB Client
< >   Apple Cinema Display support
< >   USB LD driver
< >   PlayStation 2 Trance Vibrator driver support
< >   IO Warrior driver support
< >   USB Gadget Support  --->
< >   RockChip USB Host 1.1 support
< >   RockChip USB OTG 2.0 support
```

RK29 作为 host 支持的常用设备有：

U 盘/CDROM (USB Mass Storage support)

3G modem (USB Serial Converter support -- >)

6.1.1 USB Mass Storage

U 盘属于 SCSI 设备，所以在配置 USB 模块之前需要配置 Device Drivers-- >SCSI device support。

U 盘属于 SCSI disk 设备，另外有些 U 盘含多个盘符，需要注意选上相关选项，如下图

```
                    SCSI device support
menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.
rizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  < > module capable

< > RAID Transport Class
<*> SCSI device support
< > SCSI target support
[*] legacy /proc/scsi/ support
    *** SCSI support type (disk, tape, CD-ROM) ***
<*> SCSI disk support
< > SCSI tape support
< > SCSI OnStream SC-x0 tape support
<*> SCSI CDROM support
[ ]   Enable vendor-specific extensions (for SCSI CDROM)
<*> SCSI generic support
< > SCSI media changer support
    *** Some SCSI devices (e.g. CD jukebox) support multiple LUNs ***
[*] Probe all LUNs on each SCSI device
[ ] Verbose SCSI error reporting (kernel size +=12K)
[ ] SCSI logging facility
[ ] Asynchronous SCSI scanning
    SCSI Transports  --->
[ ] SCSI low-level drivers  --->
```

配置完 SCSI device support 后，可以在 USB support 中找到如下选项，选上即可。

```
<*>   USB Mass Storage support
[ ]     USB Mass Storage verbose debug
```

6.1.2 USB Serial Converter

USB 3G modem 使用的是 USB 转串口，使用是需要选上如下选项

```
<*>   USB Serial Converter support  --->
```

进入该项配置，选上如下选项：

```
--- USB Serial Converter support
[ ]   USB Serial Console device support (EXPERIMENTAL)
[ ]   Functions for loading firmware on EZUSB chips
[*]   USB Generic Serial Driver
```

```
<*>   USB driver for GSM and CDMA modems
```

6.2 USB gadget 配置

RK29 作为 device 使用时，需要配置 USB gadget：

```
<*>   USB Gadget Support  --->
```

进入选项后配置如下：

```
------------------------- USB Gadget Support ----------------------
enu.  <Enter> selects submenus --->.  Highlighted letters are hotke
izes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Sea
 < > module capable


---------------------------------------------------------------------
--- USB Gadget Support
[ ]   Debugging messages
[ ]   Debugging information files
[ ]   Debugging information files in debugfs
      USB Peripheral Controller (Synopsys DWC OTG Controller)  --->
        Synopsys DWC OTG Controller
<*>   USB Gadget Drivers (Android Gadget)  --->
```

6.3 芯片 USB 控制器配置

控制器分为 USB HOST 1.1 和 USB OTG 2.0 两个控制器

```
< >   RockChip USB Host 1.1 support
< >   RockChip USB OTG 2.0 support
```

6.3.1 USB 1.1 host 控制器

USB 1.1 host 控制器只能作为 host 使用，配置较为简单，如需使用，直接选上该项即可。

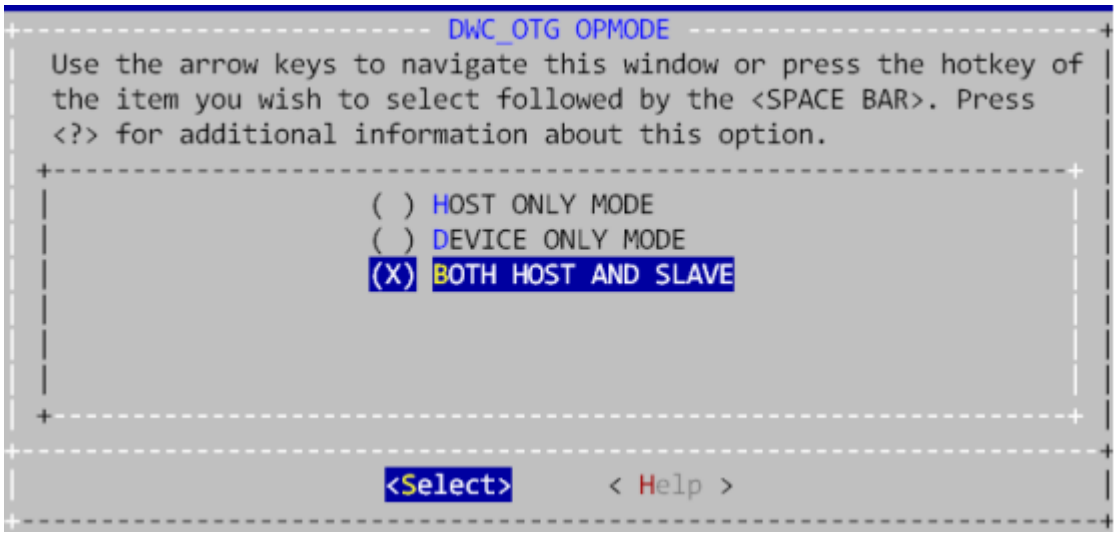
6.3.2 USB 2.0 OTG 控制器配置

USB 2.0 OTG 控制器在此可以配置是否支持 host，device 模式，并在此配置开机默认角色为 HOST 还是 DEVICE。选项如下：

```
<*>   RockChip USB OTG 2.0 support
[ ]     enable debug mode
<*>     DWC_OTG OPMODE (BOTH HOST AND SLAVE)  --->
        USB controller mode (HOST PREFERENCE MODE)  --->
```

6.3.3 DWC_OTG OPMODE 配置

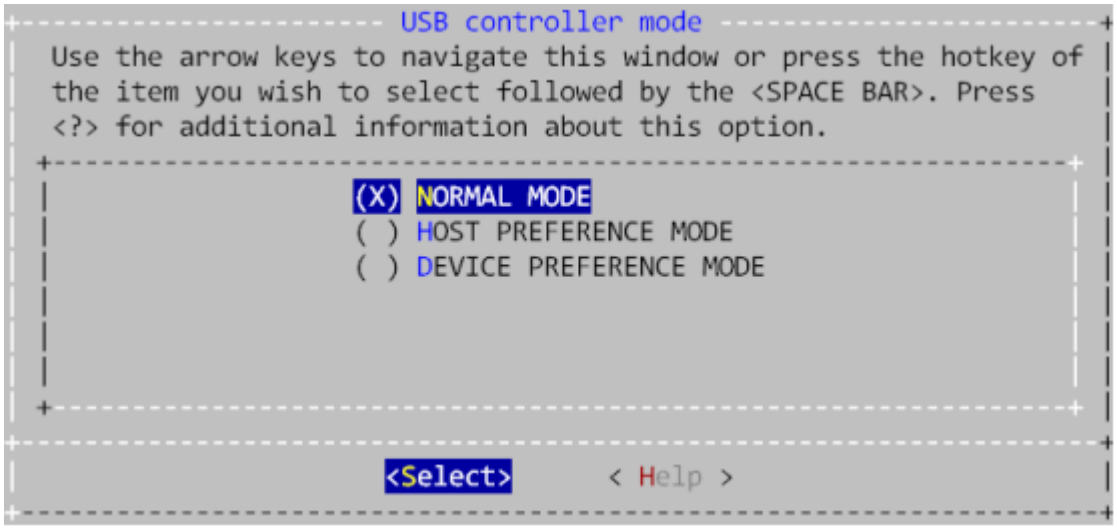
其中 DWC_OTG OPMODE 用于选择控制器是否支持 host 或者 device，其选项如下：



其中 HOST 相关选项(HOST ONLY MODE 和 BOTH HOST AND SLAVE)需要先完成 6.1 的配置，而 DEVICE 相关选项(DEVICE ONLY MODE 和 BOTH HOST AND SLAVE)需先完成 6.3 的配置。

6.3.4 USB controller mode 配置

USB controller mode 用于选择系统开机时控制器默认角色，其选项如下：



各选项含义如下：

NORMAL MODE: 控制器工作于 OTG 模式，角色由 USB_ID 决定

HOST PREFERENCE MODE: 默认为 HOST 模式，不 care USB_ID 状态。

DEVICE PREFERENCE MODE：默认为 device 模式，不 care USB_ID 状态。

7. 3G 相关日志信息的获取

7.1 使用串口捕捉 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

7.2 使用 adb 工具捕捉 LOG 信息

打开 adb shell,输入以下命令

$ logcat –b radio > /cache/radio.log &

$ logcat –s pppd > /cache/pppd.log &

抓取 kernel 的打印：

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

退出 adb shell，把机器中的 log 文件 pull 到本地

adb pull /cache/*.log d:\

8. Android 4.2.2 3G 内置模块电源选项[可选]

当项目有内置模块的时候，为了方便对模块进行开关，可以在设置选项中添加电源控制，当没有内置模块的时候，“内置模块电源”选项是不会显示出来的。代码位于 packages/apps/Phone/src/com/android/phone/ MobileNetworkSettings.java

添加电源选项的功能是为了在同一固件上实现内置 3G 模块和外置 3G dongle 都可以使用的需求。

内置/外置 3G dongle 只能工作在一种模式下，当使用外置 3G dongle 需要关掉模块电压，反之，要用内置模块的时候，需要拔掉 3G dongle.说明设置电源开关选项需要在驱动文件中创建相应的 modem_status 节点，具体实现可以参见 RK 各个驱动文件。目录位于：/drivers/misb/3g_modem/目录

9. Android 4.1/Anroid 4.2 wifi-only 配置

如果 android 不需要使用移动网络服务，那系统是 wifi-only 版本的，需要将移动网络 config.xml 文件中有关网络属性去掉

4.2 系统文件路径

device/rockchip/rk30sdk/overlay/frameworks/base/core/res/res/values/config.xml

4.1 系统文件路径：

3066_4.1_sdk/device/rockchip/rk30sdk/overlay/frameworks/base/core/res/res/values

修改如下：

```
diff --git a/overlay/frameworks/base/core/res/res/values/config.xml b/overlay/frameworks/base/core/res/res/values/config.xml
index be6db12..e9cdef5 100755
--- a/overlay/frameworks/base/core/res/res/values/config.xml
+++ b/overlay/frameworks/base/core/res/res/values/config.xml
@@ -40,13 +40,13 @@
     <!-- the 6th element indicates boot-time dependency-met value. -->
     <string-array translatable="false" name="networkAttributes">
         <item>"wifi,1,1,1,-1,true"</item>
+        <!--item>"mobile,0,0,0,-1,true"</item>
         <item>"mobile_mms,2,0,2,60000,true"</item>
         <item>"mobile_supl,3,0,2,60000,true"</item>
         <item>"mobile_hipri,5,0,3,60000,true"</item>
         <item>"mobile_fota,10,0,2,60000,true"</item>
         <item>"mobile_ims,11,0,2,60000,true"</item>
```

```
+         <item>"mobile_cbs,12,0,2,60000,true"</item-->
          <item>"wifi_p2p,13,1,0,-1,true"</item>
          <item>"eth,9,9,4,60000,true"</item>
      </string-array>
@@ -55,7 +55,7 @@
                          [# simultaneous connection types]"  -->
      <string-array translatable="false" name="radioAttributes">
          <item>"1,1"</item>
+         <!--item>"0,1"</item-->
          <item>"9,1"</item>
      </string-array>
```

考虑到 CPU 使用情况，在 init.rc 文件中将 3G 相关的 RILD 服务关闭

```
service ril-daemon /system/bin/rild
```

10. 3G 通话项目设置

短信 APK 默认在 Dataonly 项目中是没有加载的，源码路径：

packages/apps/Mms

目前 4.4 系统上在 packages/apps/Provision 可以打开和关闭 MMS 应用的显示和关闭，电话应用在 overlay 中直接设置 voice_capable 属性为 true 即可，4.2 系统在 parmeter 中设置

4.4 系统改动如下提交：

```
commit 2e3f95e2a02faa09ebd9b6ae9e2f44e55bad009f
Author: xxh <xxh@rock-chips.com>
Date:   Tue Nov 5 19:43:20 2013 +0800

    remove SMS and contanct app if the device was not configured as phone
```

4.2 系统 parameter 设置如下，具体设置也可以参考“RK30SDK 语音平板使用文档-v1.7”

```
CMDLINE: board.ap_mdm=4 board.ap_has_alsa=1 board.ap_data_only=1 console=ttyFIQ0
```

注意：使用的 3G 模块必须与 RIL 库匹配

11. RIL 库移植说明

注意：该章节对具有开发能力的客户提供

目前 RK 平台上的 RIL 库针对 RK 3G 支持列表中的模块都是做过兼容的，客户使用默认 SDK 上的 RIL 即可，若 3G 模块不在支持列表中，但是客户想兼容 RK RIL 需要使用动态切换的功能，如果不想兼容 RK RIL 客户可以自行调试 RIL 库，方法如下：

1. 动态切换功能在 hardware/ril/rild.c 中实现

根据 3G 模块 VID PID 确定使用支持的 RIL 库，默认调用 RK RIL 库, VID PID 在文件系统中路径如下：

添加 int get_devices(char * modem_path)方法读取 VID PID

/sys/bus/usb/devices/1-1/idVendor

/sys/bus/usb/devices/1-1/idProduct

或者

/sys/bus/usb/devices/2-1/idVendor

/sys/bus/usb/devices/2-1/idProduct

2.rild.c 中添加如下代码：

```
+     switch(get_devices())
+     {
+    // libzte-xxx.so   模块支持的特定 RIL 库
+     case MODEM_XXXX:
+    sprintf(rilLibPath, "/system/lib/%s", "libxxx-xxx.so");
+     break;
+     default:
+    //RK 3G 支持列表中模块使用的 RIL 库
+     sprintf(rilLibPath, "/system/lib/%s", "libril-rk29-dataonly.so");
+     break;
+     }
```

备注：如果不做兼容 RK RIL 的功能，可以直接替换系统默认的 RIL 库