

**广和通H350 AndroidRil 软件设计文档**

版权声明

版权所有©深圳市广和通实业发展有限公司 2012。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

注意

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

商标申明

为深圳市广和通实业发展有限公司的注册商标，由所有人拥有。



版本记录

|  |  |  |  |
| --- | --- | --- | --- |
| 文档版本 | 更新日期 | 说明 | 编写人 |
| V1.0.0 |  | 初始版本 | 刘其峰 |

**目录**

目录

[1 引言 4](#_Toc402279570)

[1.1 编写目的 4](#_Toc402279571)

[1.2 项目背景 4](#_Toc402279572)

[1.3 运行环境 4](#_Toc402279573)

[2 软件概述 5](#_Toc402279574)

[2.1 RIL 软件架构 5](#_Toc402279575)

[3 软件功能 7](#_Toc402279576)

[3.1 RIL初始化 7](#_Toc402279577)

[3.2 requestRadioPower 开启通讯模块 7](#_Toc402279578)

[3.3 requestQueryNetworkSelectionMode 查询网络模式 8](#_Toc402279579)

[3.4 requestGetCurrentCalls 获取当前通话 9](#_Toc402279580)

[3.5 requestDial 请求呼叫 9](#_Toc402279581)

[3.6 requestHangup 挂断电话 10](#_Toc402279582)

[3.7 requestSignalStrength 获取信号强度 10](#_Toc402279583)

[3.8 requestRegistrationState 获取卡注册状态 11](#_Toc402279584)

[3.9 requestOperator 获取运营商名称 12](#_Toc402279585)

[3.10 requestSendSMS 请求发送短信 13](#_Toc402279586)

[3.11 requestSetupDataCall 拨号上网 13](#_Toc402279587)

[3.12 requestEnterSimPin 请求输入SIM PIN码 14](#_Toc402279588)

[3.13 requestSendUSSD 请求USSD业务 14](#_Toc402279589)

[3.14 模块AT主动上报处理 15](#_Toc402279590)

[4 总结 17](#_Toc402279591)

# 引言

## 编写目的

本文档主要是针对广和通H350模块设备基于Android 系统的RIL集成开发活动进行相关的指导说明。本文档主要面向基于Android 系统的产品开发商的软件开发人员。

广和通公司研发的RIL可支持android 4.0 4.2 4.4等系统。

## 项目背景

Radio Interface Layer简称 RIL。当Android系统需要由模块提供的数据、语音、短信服务时，Android系统必须与模块进行通信，RIL就是模块与android系统之前的通信桥梁。模块用AT命令进行通信，但更模块厂商的AT命令都各有不同，所以广和通实现符合自己AT的vendor ril。

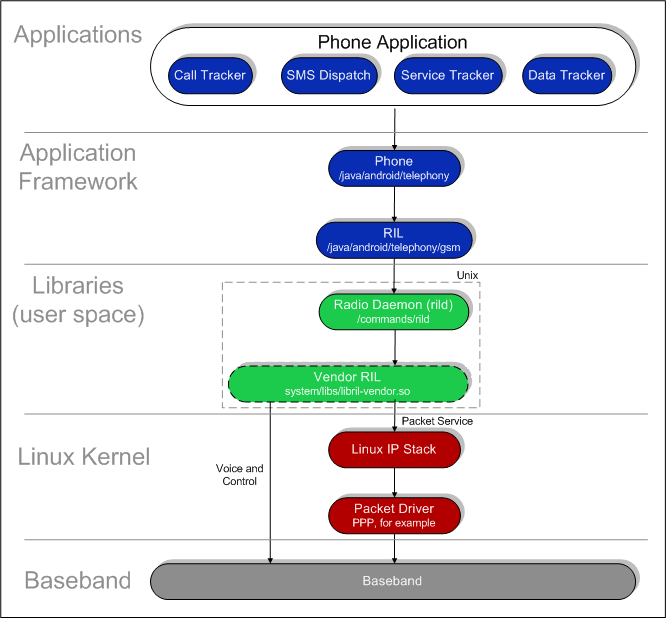
## 运行环境

本产品的硬件运行环境是H350 3G模块，软件环境是Android系统，编译环境为arm-gcc，采用C语言开发。

# 软件概述

RIL(Radio Interface Layer)，是Android电信业务系统和底层硬件之间的抽象层，Android提供该层方便用户移植。手机每次请求相应的电信服务，如电话、短信、网络连接等，都是通过AT指令与底层硬件进行交互的，拿电话来讲，用户请求拨打电话，需要向RIL发送相关的请求，接着RIL按照用户的请求，调用相应的AT指令发送接口给底层发送AT指令，完成用户拨打电话的流程。通过研究RIL的代码可以看到，Android的rild库是介于HAL接口与basebandmodem之间，它需要针对不同的GSM模块进行不同的GSM驱动开发，公用的部分google给你做好了，特定的部分需要你自己去定制(即vendor RIL)，这样做可以大大地提高开发效率。那么RIL在Android系统中的位置是如何呢。请看下一节：

## RIL 软件架构



根据上图可知RIL处理上层与底层中间是一座AT通信桥梁，Vendor RIL是提供AT指令发送接口，RIL Daemon是加载RIL相关的库同时初始化RIL，注册RIL相关函数接口。同时RILD也是上层和底层RIL的调用中转站。RIL JNI Layer，提供上层Telephony APP Framework调用。我司的RIL一般集中在Vendor RIL部分，即对reference-ril.c的处理。

# 软件功能

## RIL初始化

1)      Init.rc执行rild,并创建两个socket:/dev/socket/rild和/dev/socket/rild-debug

service ril-daemon /system/bin/rild

socket rild stream 660 root radio

socket rild stream 660 radio system

2)      进入rild.cpp的main函数，读取rild.lib的path和rild.libargs系统属性，确定厂商的RIL库和初始化参数。

3)      执行RIL\_startEventLoop开启事件队列，进行事件监听。这个函数会建立s\_tid\_dispatch线程。

4)      加载厂商的RIL库，调用RIL\_Init初始化RIL，建立s\_tid\_mainloop线程。在该线程主循环中会调用at\_open建立另一个线程s\_tid\_reader。

5)      调用RIL\_register建立vender ril和ril库之间的联系。获取init.rc中建立的两个socket(rild,rild-debug)，进行侦听，并加入消息事件循环中(s\_tid\_dispatch负责轮询分发)。

## requestRadioPower 开启通讯模块

此请求用来设定模块的各种工作模式，此请求下来时需要根据对应情况下对应的CFUN值。在飞行模式时下CFUN=4，在全功能模式时下CFUN=1，CFUN命令详细格式如：AT+CFUN=<fun>,fun值可为

功能模式：

|  |
| --- |
| 功能模式：  0 模块关机 (去注册网络).  1 开启全功能，意味着从模块未注册的模式开启模块  4 关闭RF的传输和接收（飞行模式）  6 开启STK功能，允许接收从SIM卡发送的SIM-APP命令。  7 关闭STK功能  8 禁止接收从SIM卡发送的SIM-APPL指令 |

且对应还需要用ATS24来设置H330进入休眠模式的时间，ATS24命令详细格式如：ATS24=[<value>],value值可为：

|  |
| --- |
| 时间：秒 (0 <= n <= 255) |
| 0 禁止模块进入休眠  >0 允许模块在一定时间后进入休眠  默认值是 000. |

## requestQueryNetworkSelectionMode 查询网络模式

此请求用来查询当前网络的选择模式，使用的是COPS?命令，读取命令返回当前的模式和当前被选择的运营商。COPS？命令详细格式如：

|  |  |  |
| --- | --- | --- |
| Read | AT+COPS? | +COPS: <mode>[,<format>,<oper>[,< AcT>]]  OK  Or  +CME ERROR: <err> |

|  |  |
| --- | --- |
| 参数 | 说明 |
| <mode> | 确定H330搜网类型：  0 自动搜网 (<oper> 参数不需要)  1 手动搜网 (<oper> 参数必须给出)  2 注销网络  3 设置<format> (<oper>参数不需要); 仅在读取命令中有效，该命令不会进行网络的注册或注销  4 手动/自动注册 (<oper>参数必须给出; 如果手动选择网络失败，则自动搜索网络)  默认值是 0. |
| <format> | 运营商名字格式：  0 长名字  1 短名字  2 数字  默认值是 0 |
| <oper> | 显示的运营商名字.  长字母数字格式的可以长达16个字符. 短字母数字格式的可以长达8个字符。  数字格式是GSM位置区域认证码 (参考 GSM 04.08 的10.5.1.3章节)，由1个3位的BCD形式的国家码 (参见 ITU-T E.212 Annex A [10]), 加上1个2位的BCD形式的网络码（由管理机构分配）组成。  返回的<oper> 不是以BCD 格式, 但是以从BCD转换的IRA字符的格式。 |
| <AcT> | 指示无线接入技术和它的可能值:  0 GSM  2 UMTS |

## requestGetCurrentCalls 获取当前通话

此请求用来获取当前通话，显示当前的呼叫和各方的状态，使用的是CLCC命令，详细格式如：

|  |  |  |
| --- | --- | --- |
| Execute | AT+CLCC | +CLCC: <idx>,<dir>,<call state>, <mode>,<mpty>,<number>,<type> [<CR><LF>  + CLCC: <idx>,<dir>,<call state>,<mode>,<mpty>,<number>,<type>  […]]  OK |

主要参数详解：

|  |  |
| --- | --- |
| <dir> | 0 呼出 (MO)  1 呼入 (MT) |
| <call state> | 呼叫状态 |
|  | 0 激活  1 保持  2 呼出(MO call)  3 呼出，对方振铃 (MO call)  4 呼入 (MT call)  5 等待 (MT call)  6 终止 |

当获取到有当前通话时，RIL需要给上层上报主动信息，通过调用RIL\_onUnsolicitedResponse (

**RIL\_UNSOL\_RESPONSE\_CALL\_STATE\_CHANGED**,

NULL, 0);

最主要的是将红色部分信息报给上层。上层根据此信息就知道当前通话状态已经改变而发起对应操作。

## requestDial 请求呼叫

请求呼叫功能，在当前网络上请求建立语音呼叫，如果当前已经存在由语音通话连接，再发起新的语音请求，那么之前的语音通话将被挂起，然后建立新的语音通话。使用命令格式为：ATD<number>;参数解释：

|  |  |
| --- | --- |
| 参数 | 说明 |
| <number> | 有效的电话号码字符包含: 0 1 2 3 4 5 6 7 8 9 \* # + |

## requestHangup 挂断电话

请求挂断语音通话，使用命令AT+CHLD=<n>,参数n解释：

|  |  |
| --- | --- |
| 参数n | 呼叫保持操作说明 |
| 0 | 释放所有保持通话 |
| 1 | 释放所有激活的通话且接受挂起或等待的通话。 |
| 1x | 释放指定的通话x，x是一个参与活跃的MTPY通话的通话序列号 |
| 2 | 将所有激活的通话挂起并接受已挂起或等待的通话. |
| 2x | 在有激活的MTPY通话的情况下，将所有激活的通话挂起，除了Call x |
| 3 | 添加一个保持的通话到多方通话中 |
| 4 | Connects the two calls and disconnects the subscriber from both calls (Explicit Call Transfer) |
| 5 | ?? |
| 6 | Puts an active call on hold or an held call to active, while another call is waiting |
| 7 | Disconnect users in multiparty without accepting incoming call |
| 8 | release all calls |

此命令用以”保持通话”或”激活通话”, 意味着一个保持或激活的单方或多方通话。

## requestSignalStrength 获取信号强度

请求获取模块信号强度，使用命令格式：AT+CSQ? 返回参数：+CSQ: <rssi>,<ber>

参数解释：

|  |  |
| --- | --- |
| 参数 | 说明 |
| <rssi> | 0 到 31 - 覆盖的范围从-113 dbm 到 -51 dbm  0 -113 dBm或更低  1 -111 dBm  2…30 -109… -53 dBm  31 -51 dBm或更大  99 未知或不可检测 |
| <ber> | 信道误码率 (用百分比表示)  0-7 TS 45.008中描述的接收质量值  99 未知或不可检测 |

将返回的rssi存储在signal[0]，ber存储在signale[1],将赋值给

response.GW\_SignalStrength.signalStrength = signal[0];

response.GW\_SignalStrength.bitErrorRate = signal[1];

再调用RIL\_onRequestComplete(t, RIL\_E\_SUCCESS, &response, sizeof(RIL\_SignalStrength\_v6));

将response的地址返回给上层，上层将获取到rssi及ber值，再根据不同的rssi值显示不同的信号格。

## requestRegistrationState 获取卡注册状态

请求获取卡注册状态，使用AT+CREG？该命令返回的结果为当前网络注册状态和服务小区的信息，返回结果格式: +CREG: <n>,<stat>[,<lac>,<ci>[,<AcT>] ],主要参数说明：

|  |  |
| --- | --- |
| 参数 | 说明 |
| <stat> | 0 未注册, 且当前ME没有搜索运营商  1 注册上本地网  2 未注册，但当前ME正在搜索可用的网络  3 注册被拒绝.  4 未知.  5 注册上，漫游中. |
| <AcT> | 0 GSM  2 UTRAN  3 GSM w/EGPRS  4 UTRAN w/HSDPA  5 UTRAN w/HSUPA  6 UTRAN w/HSDPA and HSUPA |

使用CGREG? 读取当前GPRS网络注册状态和服务小区的信息,返回结果格式：+CGREG: <n>,<stat>[,<lac>,<ci>[,<AcT>,<rac>]]，主要参数说明：

|  |  |
| --- | --- |
| 参数 | 说明 |
| <stat> | 0 未注册, 且当前ME没有搜索运营商  1 注册上本地网  2 未注册，但当前ME正在搜索可用的网络  3 注册被拒绝.  4 未知.  5 注册上，漫游中. |
| <AcT> | 0 GSM  2 UTRAN  3 GSM w/EGPRS  4 UTRAN w/HSDPA  5 UTRAN w/HSUPA  6 UTRAN w/HSDPA and HSUPA |

查询到当前的网络后需要根据AcT值来给registration[3]赋值，registration[3]可为：

RADIO\_TECH\_GPRS，RADIO\_TECH\_HSPA, RADIO\_TECH\_LTE,之后再将调RIL\_onRequestComplete(t, RIL\_E\_SUCCESS, responseStr, numElements\*sizeof(responseStr));

将存储在responseStr的查询结果反馈给上层，上层将显示对应网络状态是2G/3G/4G。

## requestOperator 获取运营商名称

请求获取运营商名称，使用AT+COPS=3,**0**;+COPS?;+COPS=3,**1**;+COPS?;+COPS=3,**2**;+COPS?，这是一条连续查询的AT指令，红色参数即依次分别查询运营商名字格式：

0 长名字

1 短名字

2 数字

返回格式为：+COPS: <mode>[,<format>,<oper>[,< AcT>]]

主要参数解释：

|  |  |
| --- | --- |
| 参数 | 说明 |
| <mode> | 确定H330搜网类型：  0 自动搜网 (<oper> 参数不需要)  1 手动搜网 (<oper> 参数必须给出)  2 注销网络  3 设置<format> (<oper>参数不需要); 仅在读取命令中有效，该命令不会进行网络的注册或注销  4 手动/自动注册 (<oper>参数必须给出; 如果手动选择网络失败，则自动搜索网络)  默认值是 0. |
| <format> | 运营商名字格式：  0 长名字  1 短名字  2 数字  默认值是 0 |
| <oper> | 显示的运营商名字.  长字母数字格式的可以长达16个字符. 短字母数字格式的可以长达8个字符。  数字格式是GSM位置区域认证码 (参考 GSM 04.08 的10.5.1.3章节)，由1个3位的BCD形式的国家码 (参见 ITU-T E.212 Annex A [10]), 加上1个2位的BCD形式的网络码（由管理机构分配）组成。  返回的<oper> 不是以BCD 格式, 但是以从BCD转换的IRA字符的格式。 |
| <AcT> | 指示无线接入技术和它的可能值:  0 GSM  2 UMTS |

将查询的结果存储于response数组中，并调用RIL\_onRequestComplete(t, RIL\_E\_SUCCESS, response, sizeof(response));将数组信息反馈给上层，最明显的是在解锁界面将显示运营商的名称。

## requestSendSMS 请求发送短信

请求发送短信，使用AT+CMGS=<length><CR>PDU 以<ctrl-Z/ESC>结束信息输入，该命令通过H330将SMS发送到网络。在成功发送SMS之后，模块将返回消息发送参考序号<mr>

参数说明：

|  |  |
| --- | --- |
| 参数 | 说明 |
| <da> | 由引号括起来的接收方地址 |
| <toda> | 接收方地址类型，范围128-255 (参考 GSM 03.40, 9.1.2.5).  如果不提供，当<da>的第一个字符是'+'的时候，<toda>为145，否则为129。 |
| <length> | PDU模式下的以字节数表示的数据长度，不包含短信中心的数据。 |
| <mr> | 发送SMS的参考序号。 |
| PDU | 包含消息头和消息数据在内的PDU格式的数据。参考 “+CMGW, 向存储区中写入消息”。 |

将返回结果存储于response中，并调用RIL\_onRequestComplete(t, RIL\_E\_SUCCESS, &response, sizeof(response));将结果反馈给上层。

## requestSetupDataCall 拨号上网

请求发起数据上网，发起数据上网的方式有三种，分别为：PPP(3G)、RAW-IP(3G)、NCM(4G)这三种拨号方式。本文件以PPP为例。在发起数据上网时，必须先去激活PPP上下文件及清空当前pppd进程，之后再激活PPP上下文，下发ATD\*99#拨号，获取CONNECT返回后调用pppd程序发起PPP拨号。通过下发CGDCONT?及XDNS命令查询是否有IP地址及DNS地址，再将查询到的对应地下存储于responses中，并调用

RIL\_onRequestComplete(\*t, RIL\_E\_SUCCESS, responses, n\*sizeof(RIL\_Data\_Call\_Response\_v6));

将拨号结果反馈给上层，上层最终会在信号格的左上角显示H/E/3G/4G图标。

## requestEnterSimPin 请求输入SIM PIN码

此请求对于需要输入PIN码的SIM卡，使用该命令输入正确的PIN码并解锁SIM卡。对于需要输入PUK码的SIM卡，使用该命令输入正确的PUK码来重新激活SIM卡。命令格式AT+CPIN=<passwd>

参数表示所需要的pin码。

PIN输入正确时调用：RIL\_onRequestComplete(t, RIL\_E\_SUCCESS, &num\_retries, sizeof(int \*));

PIN输入错误时调用：RIL\_onRequestComplete(t, RIL\_E\_PASSWORD\_INCORRECT, & num\_retries, sizeof(int \*));

最终都需要将输入PIN码的次数反馈给上层。因为PIN码连续输错3次后，将需要PUK码来解决，PUK连续输错10次后SIM卡将被锁掉，需要去营业厅重新开卡。

## requestSendUSSD 请求USSD业务

USSD（Unstructured Supplementary Service Data）即非结构化补充数据业务，是一种新型基于GSM网络的交互式数据业务。调试时在上层拨号键盘输入 \*125#, 上层将输入结果以传参方式带给RIL, requestSendUSSD(void \*data, size\_t datalen, RIL\_Token t)中的第1个data参数，即为上层的输入结果；

之后RIL将上层的输入结果赋值给CUSD的第一个参数如：asprintf(&cmd, "AT+CUSD=1,\"%s\"", ussdRequest);发起USSD业务。

AT+CUSD=[<n>[,<str>[,<dcs>]]] 命令格式及返回格式+CUSD: <m>[,<str>[,<dsc>]]]

参数说明：

|  |  |
| --- | --- |
| 参数 | 说明 |
| <n> | 0 禁止非请求结果码上报  1 开启非请求结果码上报.  2 取消当前会话(不适用于读取命令响应). |
| <str> | 字符串类型的 USSD串 (当<str>参数没有被给出，网络没有协商成功):  如果<dcs>显示GSM 03.38 的默认字母表被使用：  • 如果TE字符集不是”HEX”（参考选择字符集命令 +CSCS）：根据GSM 07.05附录A的规则，ME/TA将GSM字母表转换成当前TE字符集。  • 如果TE字符集是”HEX”: ME/TA将每个7位GSM字母表的字符转换成两个IRA字符长的十六进制数（例如字符？（GSM23）被转换成为17（IRA49和55））。  如果<dcs>显示8位数据编码模式被使用：ME/TA将每个8位字节转换成两个IRA字符长的十六进制数（例如8位字节的整型值42被转换为2个字符2A（IRA50和65））。 |

在CUSD成功下发后，模块将主动上报+CUSD，RIL将CUSD上报的信息解释后，将结果于response结构体中，并调用

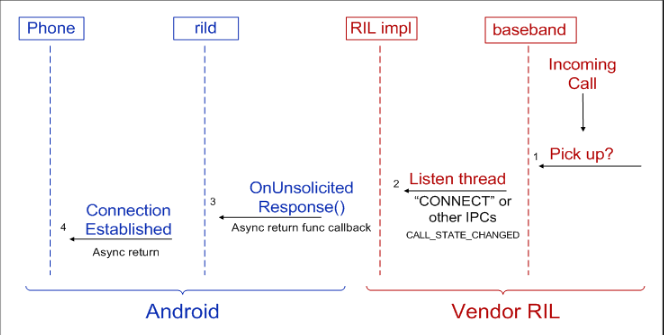
RIL\_onUnsolicitedResponse(RIL\_UNSOL\_ON\_USSD, response, n \* sizeof(const char \*));

将信息反馈给上层，上层将显示对应的USSD菜单。

## 模块AT主动上报处理

前面几节介绍的都是上层主动发起的一系列请求，那模块主动发起的AT主动上报RIL又如何处理呢。

下图为模块发起AT主动上报时的流程图(以来电为例)：

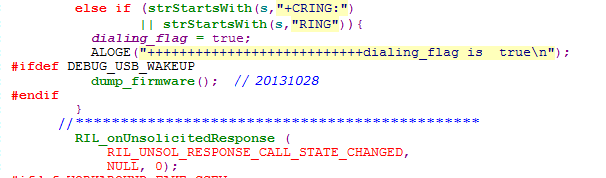


被动请求由模块的主动上报发起，在RIL里面处理函数为

static void onUnsolicited (const char \*s, const char \*sms\_pdu)//主动上报响应

{//onUnsolicited只解析出头部(一般是+XXXX的形式)，然后按类型决定下一步操作

以来电为例，来电时模块会上动上报RING的动作。



上图为对应函数处理，当RING上报时RIL需要给上层反馈CALL状态发生变化。上层根据这个变化将显示来电界面。对应的RIL\_UNSOL\_RESPONSE\_CALL\_STATE\_CHANGED参数可以在ril.h中查询到相关的解释。

# 总结

Vendor ril的处理主要体现在上层每下一个request时，RIL要对应下什么AT，模块每主动上报一个AT时，RIL要对应给上层报什么状态。上层传入的参数及需要的返回都可以从ril.h中获得对应的信息。