基于 Android 的有线电话 CID 功能①

陈成伟, 周渊平

(四川大学 电子信息学院, 成都 610065)

摘 要: Android 是一个基于 Linux 内核的操作系统, 被广泛用于移动及其他设备,而在有线电话功能上的开发尚显不足. 本文通过对来电显示电路、Android 系统内核驱动、JNI 层调用以及其消息驱动机制 Looper 与 Handler 的分析和运用, 将 Android 平台与有线电话来电显示(Calling Identity Delivery)结合起来, 实现了一个在 Android 应用中显示来电信息的系统, 并在实际系统中进行测试. 测试结果表明 Android 能够完成有线电话信息的接收和处理, 实现来电显示功能, 丰富了安卓系统的功能和应用.

关键词: Android; 来电显示; Linux 驱动; JNI; Looper; Handler

CID Function for Wire Telephones Based on Android

CHEN Cheng-Wei, ZHOU Yuan-Ping

(College of Electronic Information, Sichuan University, Chengdu 610065, China)

Abstract: Android, an operating system based on Linux kernel, has been widely applied in mobile and other devices, while its development on the function of wire telephones is still under exploration. This research goes through a comprehensive analysis and application of software and hardware, including Calling Identity Delivery (CID) circuits, Android system kernel driver, JNI invoking and message driving mechanism of Looper and Handler. Eventually it realizes a system to display caller information in the form of an Android APP by combining Android and CID. Tests have been conducted in the real system, and the experimental results prove Android's ability of receiving and processing messages from the wired telephone which contributes to its CID function. In summary, it enriches the function and application of Android.

Key words: Android; CID; Linux driver; JNI; Looper; Handler

Android 是一种基于 Linux 内核、自由开放源代码的操作系统, 由 Google 公司和开放手机联盟共同领导及开发, 主要使用于移动设备. 但随着 Android 的发展, 其友善的界面和完善的开发工具使它被用于越来越多其他场合.

日常办公通讯通常使用的固定电话, 具有抗干扰能力强, 通话质量好, 保密性高的特点, 最突出的是辐射小[1]. 在如今移动设备功能日益丰富的背景下, 将 Android 与有线电话有机的结合起来, 在 Android 系统实现来电号码显示功能, 既方便大家见"机"行事, 也丰富了 Android 的应用功能.

1 硬件系统

1.1 来电显示标准

根据中国通信行业标准[2], 电话线中数据信号特征如下:

① 调制方式: 连续相位二进制频移键控(BFSK)

② 数据传送方式: 二进制异步串行方式

③ 传输速率: 1200 Baud

④ 逻辑 1/逻辑 0 频率: 1200Hz/2200Hz

其中, 信号以 FSK 数据字节为单位. FSK 数据字节由一位起始位(0)和一位停止位(1)和中间的八位 FSK 数据位组成. 而消息帧由 FSK 数据字节组成, 由到达的先后分成四个部分: 信道占用信号、标志信号、消息层数据信号和校验和. 其中的消息层数据里包含了对我们有用的消息, 消息有单数据和复合数据两种格式, 经验证本地电话局采用的是单数据消息格式, 如图 1 所示.

① 收稿时间:2015-04-22;收到修改稿时间:2015-05-25

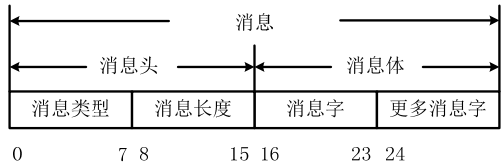

图 1　单数据消息

单数据消息由消息头和消息体组成, 消息头由消息类型和消息长度组成. 它们均为 8bit 字. 消息类型的值用来识别消息的特征. 消息长度指明后面所跟消息的字节长度. 消息体包含了终端交换机传来的日期、时间和主叫号码等信息.

1.2 硬件电路

本系统的硬件部分主要由 Android 平台(TQ210 开发板)和来电识别模块两部分组成, 如图 2 所示. TQ210 开发板, 采用底板加核心板的结构. 核心板为 ARM Cortex-A8 内核的 S5PV210 微处理器. S5PV210 性能高功耗低, 能够满足大多数应用需求, 已在通信系统、医疗电子、工业控制等场合得到广泛应用.

S5PV210 总共有 93 个中断源, 其中 32 个为外部中断. 开发板想要获得外部信息就必须使用 S5PV210 的外部中断对外部事件进行处理[3].

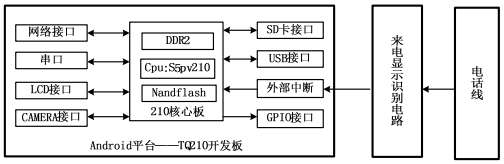

图 2　硬件系统框架

为了获取电话线路中传送的消息, 需要使用来电号码识别芯片对信号进行提取. 本文选用 MT88E39 来电显示解码芯片. 它具有较好的适应性能兼容多种标准, 且是一种供电电压范围广(3-5V)的低功耗 CMOS 器件[4]. 根据数据手册电路连接图(图 3), 各个原件的参数根据电话标准以及供电电压不同而有差别, 具体计算参见数据手册. 图中 TIP/RING 为输入电话线端子, DATA 为数据输出引脚, DCLK 为数据时钟, CD 为数据检测, DR 为数据就绪, 后两者上面的一杠表示低电平有效.

电话线上的信号经过识别芯片之后输出如图 4 所示. 不难注意到, DCLK 的每个上升沿正好处于 DATA 数据的正中. 另外 CD 在数据传输时从高电平降为低电平, 当数据传送完毕时跳变成为高电平. 实际只使用了此三个引脚且没有使用图 3 左下方的振铃检测电路.

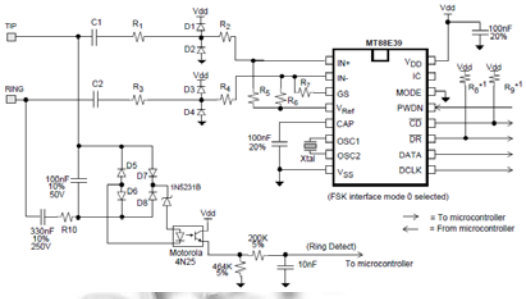

图 3　芯片电路连接图

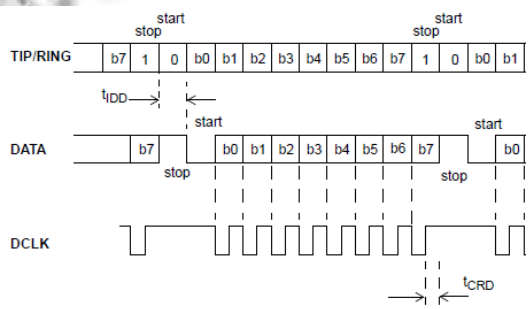

图 4　串行数据时序

2　软件系统

2.1 软件总体结构

Android 系统架构分为 4 层, 分别为: ①应用层(Applications)该层由运行在 Dalvik 虚拟机上的应用程序组成. ②应用框架层(Application Framework)该层主要由开发人员调用的 API 组成. ③系统运行库层(Android Runtime & Libraries)该层主要包括 C 语言标准库、多媒体库、Dalvik 等. ④Linux 内核层(Linux Kernel)该层主要包括驱动、内存管理、进程管理、网络协议栈等[5].

Java 本身不能直接访问硬件, 要访问硬件必须使用一些由 C/C++编写的库(主要是些*.so 文件)[5]. 因此要实现来电的显示除了内核驱动之外还必须开发由上层 Java 类调用的动态库, 将 Java 与由 C 语言编写的驱动联系起来, 具体实现框图如图 5 所示.

其中 JNI 层负责本地代码的调用.JNI 是 Java native interface 的缩写, 中文译为“Java 本地调用”. 通过 JNI 我们可以做到以下两点: ①Java 程序中的函数可以调用 Native 语言编写的函数②Native 函数可以调用 Java 层的函数. 这样可以使用 JNI 调用由 C/C++编

写的代码, 同时为底层回调上层函数实现丰富的功能提供了途径. JNI 将 Java 世界与 Native 世界紧密的结合在了一起[6].

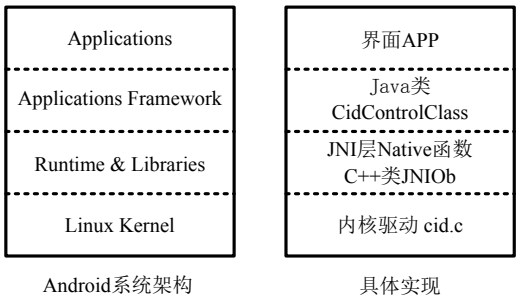

图 5 软件系统框架

Android 系统上的 Java 程序是靠消息驱动来工作的. 程序中存在一个消息队列和一个消息循环, 可以往消息队列中投递消息, 同时程序会从消息队列中取出消息, 然后进行处理. 驱动中提取出来电消息后, 把消息加入到消息队列中, 而处理线程则不断地把消息从消息队列中取出并进行处理. Android 系统中这些工作主要由 Looper 和 Handler 来实现, Looper 类用于封装消息循环, 并且有一个消息循环; Handler 类封装了消息投递、消息处理等的接口[5].

2.2 软件系统实现

2.2.1 Linux 驱动

Linux 内核驱动程序完成对 CID 数据的提取工作, 根据芯片的输出特性需要两个中断程序对信息进行处理, 如图 6 所示. 第一个中断在 DCLK 的上升沿读取 DATA 进行采集数据, 第二个中断程序由 CD 上升沿驱动告知数据提取完毕. 在完成数据采集之后上层程序需要把数据取走, 相比于直接在驱动中编写阻塞 I/O, 本文采用 poll 和 select 系统调用配合非阻塞 I/O, 此方案编程要简单同时方便日后拓展更多的 I/O. 上层用户空间中, select 调用会阻塞调用进程, 直到文件描述符集合中的一个或者多个可读或可写[7]. 而在驱动程序中需要自己编写 poll 方法, 在 poll 函数中调用 poll_wait()向 poll_table 结构中添加一个等待队列. 当数据就绪, 在驱动程序中调用 wake_up()唤醒相应的队列, 此时 poll 返回可以立即执行的操作掩码, 唤醒由 select 阻塞的进程进行读写[8].

2.2.2 JNI 层

JNI 层内包含一个 C++类 JNIOb,它有一个方法 OnEvent(), JNI 还有一些 Native 方法, 如 native_setup()、_receiveMsgFromCid()等方法. 上层 Java 类实例化时会在构造函数中调用本地方法 native_setup()新建一个 JNIOb 对象, 同时上层 Java 类打开驱动, 设置回调参数, 调用本地方法 _receiveMsgFromCid(). 本地方法 _receiveMsgFromCid()通过调用线程启动函数 pthread_create()创建一个新的线程, 在新线程中调用 select(), 如果驱动有数据可读则读取数据, 并调用 C++类 JNIOb: : OnEvent()方法. 在 OnEvent()函数中通过 CallStaticVoid-Method 调用 Java 类中的 postEventFromNative()函数, 同时把来电显示数据作为参数传递到 Java 类[9]. 这个过程实现了从 JNI 层向上调用 Java 类中函数的功能, 在 Java 层的消息循环处理机制在下节介绍, 调用关系如图 7 所示.

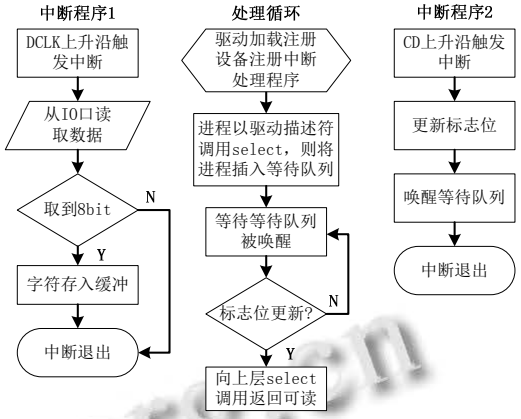

图 6 驱动程序处理流程

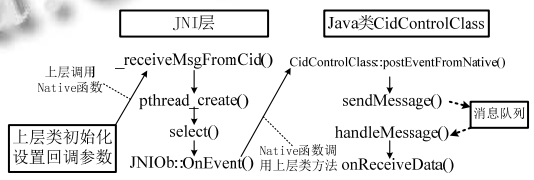

图 7 函数调用关系

2.2.3 Java 类

框架层是一个 Java 类 CidControlClass, 此类里面包含了 Looper 消息队列以及 Handler. 内部类 EventHandler 继承自 Handler, 它重写了 handleMessage()函数用于处理消息, JNI 层调用 postEventFromNative()提取从 Native 层传上来的数据并调用 sendMessage()将其封装成 message 添加的到消

息队列, handleMessage()函数对 message 进行处理, 即调用回调接口 onReceiveData(). 我们只要在上层应用类中实现了此接口, 那么消息来到的时候就会自动调用上层应用类实现的 onReceiveData()函数进行数据处理. 同时, 为了上层更好的使用 CidControlClass 类, 对它进行了简单的封装. 由于需要访问 I/O, 为了减小系统开销和资源多重占用, 采用了单例模式 .

2.2.4 Activity 界面

应用的主界面是一个简单的拨号键盘, 带有拨号用的数字键和功能键, 在文本框中可以显示想要拨号的数字. 当有来电的时候, 从主 Activity 跳转到另一个 Activity 并在此 Activity 显示来电时间和号码. 在主 Activity 中要实现 CidControlClass 提供的接口 onReceiveData(), 在 onReceiveData()获取来电显示的数据, 提取出参数, 然后使用 Intent 传递到新的 Activity 中同时实现跳转, 在新 Activity 中处理来电显示数据进行显示[10].

2.3 软件编译与安装

开发驱动和应用程序需要在 Ubuntu 系统的电脑搭建开发平台, 同时安装 arm-linux-gcc 交叉编译器、JDK、Eclipse、ADT 等软件. 在 PC 上使用交叉编译工具对内核和 Android 源代码进行编译, 然后便可以进行自己的驱动程序与应用的开发与调试.

2.3.1 驱动编译与安装

将编写好的驱动代码 cid.c 放入内核文件的根目录的/driver/char 目录下, 并修改该目录下的 Kconfig 文件与 Makefile 文件, 为了方便调试设置为生成模块. 在内核根目录下, 使用 make menuconfig、make modules 命令, 成功则生成.ko 文件. 将.ko 文件使用 adb 工具的 push 命令传送文件到开发板上, 如图 8 所示.

```
root@12:33:55[/opt/EmbedSky/TQ210/Kernel_3.0.8_TQ210_for_Android_v1.4]# make mod
ules
  CHK     include/linux/version.h
  CHK     include/generated/utsrelease.h
make[1]: "include/generated/mach-types.h"是最新的。
  CALL    scripts/checksyscalls.sh
  CC [M]  drivers/char/mydrivers/cid_1/cid_1.o
  Building modules, stage 2.
  MODPOST 31 modules
  LD [M]  drivers/char/mydrivers/cid_1/cid_1.ko
root@12:34:13[/opt/EmbedSky/TQ210/Kernel_3.0.8_TQ210_for_Android_v1.4]# adb push
 drivers/char/mydrivers/cid_1/cid_1.ko /data/local/mydata
122 KB/s (6956 bytes in 0.055s)
```

图 8　生成模块并传送到开发板上

使用 adb shell 切换到开发板上模块的保存目录, 使用 insmod 命令加载 cid_1.ko 模块, 加载成功后可以用 lsmod 命令查看加载的模块, 如图 9 所示.

```
root@android:/data/local/mydata # insmod cid_1.ko
root@android:/data/local/mydata # lsmod
cid_1 3262 0 - Live 0xbf003000
s5pv210_hdmi 2227 0 [permanent], Live 0xbf0b2000
wm8960 19652 1 - Live 0xbf0a0000
rt5370sta 528264 0 - Live 0xbf000a000
ds18b20 5020 1 - Live 0xbf005000
Acceleration_sensor 5514 0 - Live 0xbf000000
root@android:/data/local/mydata #
```

图 9　加载模块

2.3.2 JNI 本地方法与类

JNI 层使用 C++编写, 完成后放入 Android 源码根目录下/packages/apps/jni 文件夹中并编写 Android.mk 文件, 然后在 Android 源码根目录下执行 mmm /packages/apps/jni 命令便可生成 libcid.so 库, 接着将库文件放入到开发板上的/system/lib 目录下. 当应用程序运行时就会自动搜索并加载库文件.

2.3.3 上层类与主界面

在 Eclipse 中建立 Android 工程, 工程内编写 CidControlClass 类以及 Activity 程序. Android 工程目录如图 10 所示.

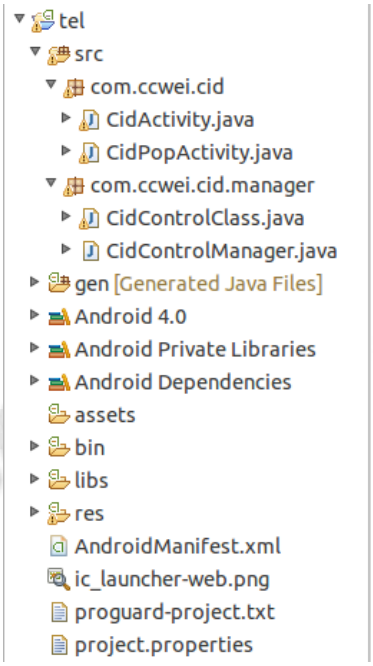

图 10　Eclipse 工程目录

工程编写完成后, 使用 USB 线链接开发板, 通过 ADT 自带的 adb 工具就可以直接在 Eclipse 中调试开发板上的程序.

3　系统测试

测试系统能否正常工作分为以下几步:

1)将 Android 平台开发板、来电显示模块、电话线

连接起来, 打开开发板电源.

2)点击桌面的 tel 图标, 即可出现拨号界面, 如图 11 所示. 使用手机拨打实验室固定电话, 经测试能顺利拨通, 在第一声振铃之后第二声振铃之前便会收到来电显示信息, 然后跳转到新的 Activity 显示如图 12 所示.

3)在图 12 界面中点击屏幕下方的“back”按钮, 就能够顺利的回到图 11 中的拨号界面. 再次拨打还会跳出来新的 Activity 显示.

经上述测试, 该系统能实现来电显示功能, 如图 12 所示, 开发板显示的来电, 正是由右侧手机拨打, 通过设计的来电显示应用程序从实验室电话线上提取的.

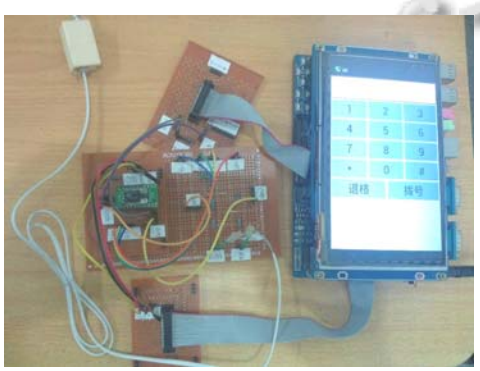

图 11 打开应用的显示效果

图 12 电话打入时的显示效果

4 结语

以上分析与测试结果显示, 该系统能够从电话线上获取电话号码以及来电时间, 并成功显示, 证明了在 Android 平台上实现来电显示方案的可行性. 事实上, Android 在有线电话上的开发和应用仍大有可为. 在本文的基础上, 如果拨号功能[11]被一并添加就能组成一个完整的电话系统. 通过记录来电号码与时间, 还可实现通讯录等功能. 同时备忘录、时钟、记事本、天气预报、室内温度、Google 地图定位等与办公、生活相关的软件都可以搭载 Android 而融入到系统当中, 从而丰富有线电话的实用功能.

参考文献

1 薛莹,徐慨,黄麟舒.来电显示电路的设计.舰船电子工程, 2008,28(9).

2 YD/T 1277.1-2003, 固定电话网主叫识别信息传送技术要求及测试方法(第一部分):技术要求.

3 S5PV210 RSIC Microprocessor User’s Manual. 1.10 ed. Samsung Electronics Co. Ltd .2010.

4 MT88E39Datasheet.Mitel Semiconductor.1999.

5 邓凡平.深入理解 Android:卷 I.北京:机械工业出版社,2011.

6 陈去疾,李敬华,郭华磊.JNI 在 Android 硬件开发中的应用. 电信快报:网络与通信,2014,(1):27–29.

7 宋宝华.Linux 设备驱动开发详解(第 2 版).北京:人民邮电出版社,2010.

8 李宁.Android 深度探索.卷 1: HAL 与驱动开发.北京:人民邮电出版社,2013.

9 高海彬.JNI 在 Android 系统下串口控制的应用.信息技术, 2013,(10):173–176.

10 李宁.Android 开发权威指南(第 2 版).北京:人民邮电出版社,2013.

11 杜江,周渊平.基于 Android 的电话拨号功能. 计算机系统应用, 2014,(12):245–248.