密级状态：绝密（　　）　秘密（　　）　内部（ ）　公开（√ ）

RK3399 双屏异显音频说明

（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：	V1.2
	作　　者：	郑应航、刘兴亮
	完成日期：	2019-5-6
	审　　核：	
	完成日期：	

福州瑞芯微电子股份有限公司

Fuzhou Rockchip Electronics Co . , Ltd

版 本 历 史

版本号	作者	修改日期	修改说明	备注
V1.0	刘兴亮	2017.10.23	发布初始版本	
V1.1	郑应航	2017.12.20	增加相关说明	
V1.2	刘兴亮	2019.5.6	增加最新修改	

目 录

1 简介

本文主要介绍了 RockChip(以下简称 RK) SDK 平台上支持的异显音频方案，包括系统原理、补丁以及配置方法等，适用于以下 SDK 平台：

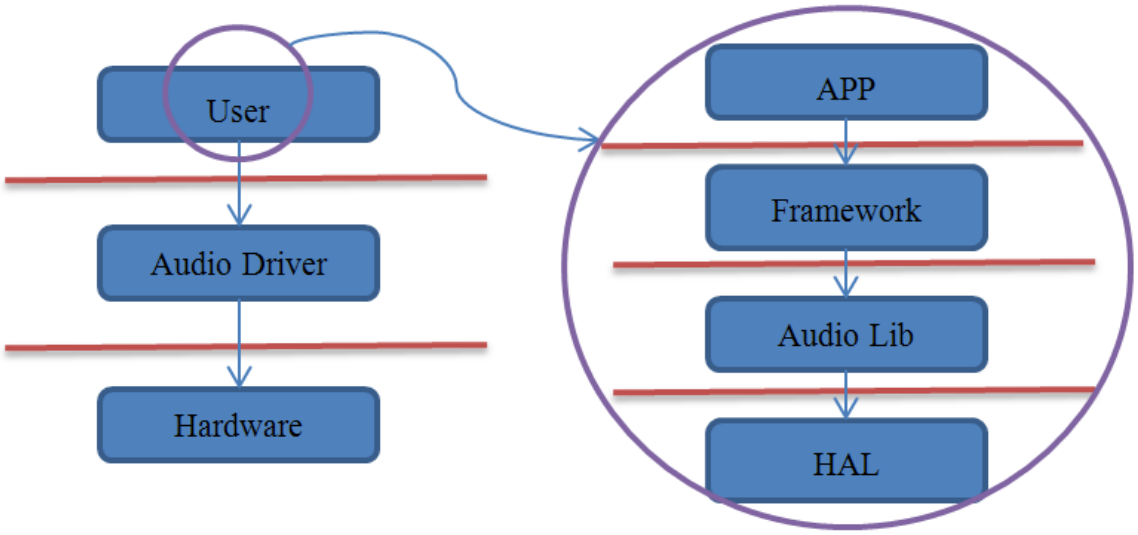
- RK3399

目前 RK 双屏异显方案有两套，分别是 Google 的 Android Presentation 和 RK 的 RK dualscreen；本文异声主要是配合 RK dualscreen 方案进行开发设计。

2 系统原理

关于双屏异显两路音频目前还没有通用的方法，音频方案思路是借鉴 A2DP（蓝牙音频传输协议）；A2DP 的场景是：铃声、触摸声等系统声音从 speaker 直接输出，音乐通过 bt 输出；这种模式和异显的需求是类似的，异显要求主屏的声音从主屏对应的声卡输出、副屏的声音从副屏对应声卡输出，不能有混音。

2.1Android 系统音频框架



整个框架包括应用层、framework 层、lib 层、hal 层、驱动以及硬件。

- APP

这是整个音频体系的最上层，因此并不是 Android 系统实现异显两路音频输出的重点。比如

厂商根据特定需求自己写的一个音乐播放器，游戏中使用到声音，或者调节音频的一类软件等等。

- Framework

Android 提供了两个功能类，AudioTrack 和 AudioRecorder；除此以外，Android 系统还为我们控制音频系统提供了 AudioManager、AudioService 及 AudioSystem 类。这些都是 framework 为便利上层应用开发所设计的。

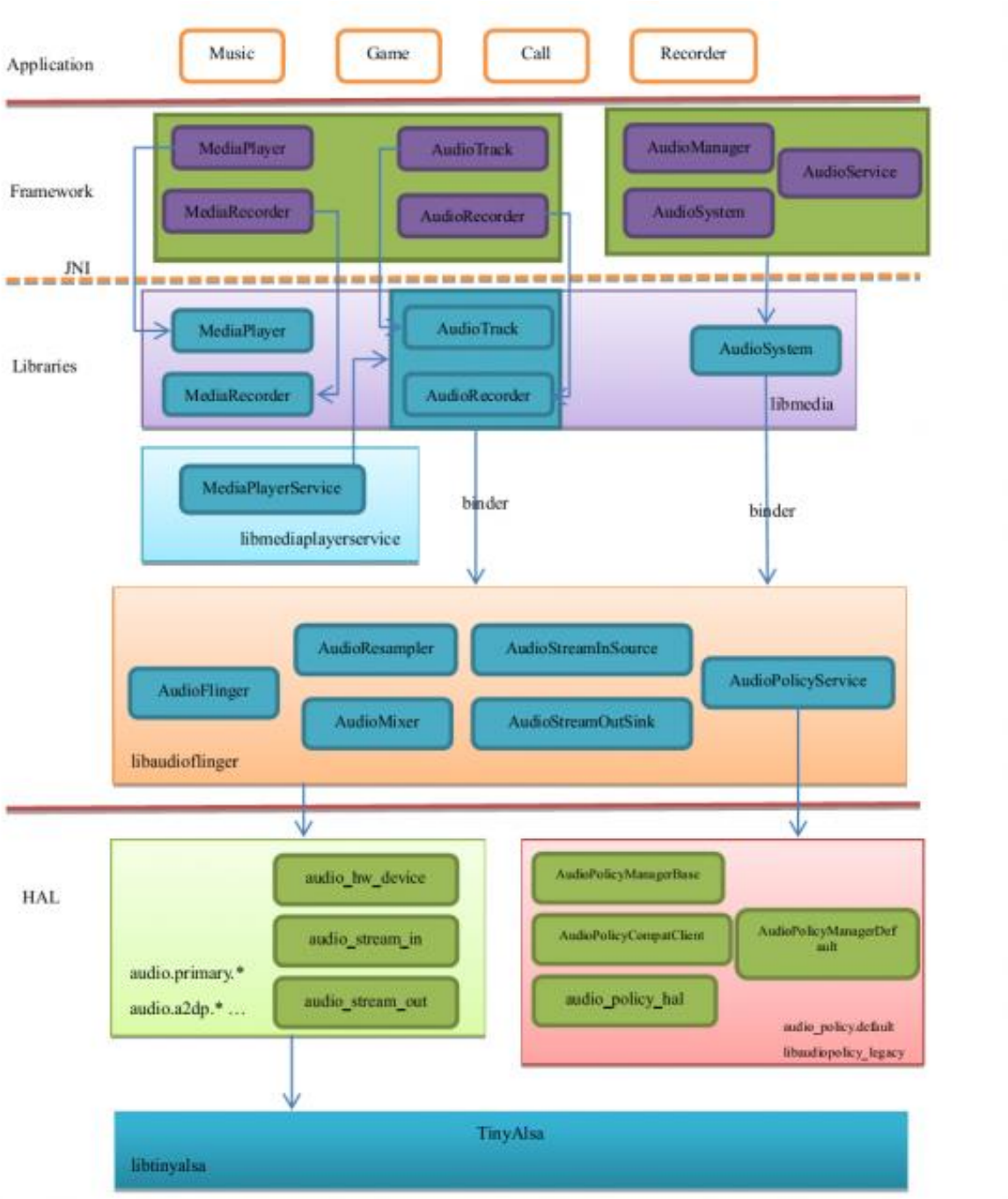
- Libraries

Framework 层的很多类，实际上只是应用程序使用 Android 库文件的"中介"而已。因为上层应用采用 java 语言编写，它们需要最直接的 java 接口的支持，这就是 framework 层存在的意义之一。而作为"中介"，它们并不会真正去实现具体的功能，或者只实现其中的一部分功能，而把主要重心放在库中来完成。比如上面的 AudioTrack、AudioRecorder 等等在库中都能找到相对应的类，这些库多数是 C++语言编写的。除了上面的类库实现外，音频系统还需要一个"核心中控"，或者用 Android 中通用的实现来讲，需要一个系统服务，这就是 AudioFlinger 和 AudioPolicyService。

- HAL

从设计上来看，硬件抽象层是 AudioFlinger 直接访问的对象。这说明了两个问题，一方面 AudioFlinger 并不直接调用底层的驱动程序；另一方面，AudioFlinge 上层(包括和它同一层的 MediaPlayerService)的模块只需要与它进行交互就可以实现音频相关的功能了。因而我们可以认为 AudioFlinger 是 Android 音频系统中真正的"隔离板"，无论下面如何变化，上层的实现都可以保持兼容。音频方面的硬件抽象层主要分为两部分，即 AudioFlinger 和 AudioPolicyService。实际上后者并不是一个真实的设备，只是采用虚拟设备的方式来让厂商可以方便地定制出自己的策略。抽象层的任务是将 AudioFlinger/AudioPolicyService 真正地与硬件设备关联起来，但又必须提供灵活的结构来应对变化——特别是对于 Android 这个更新相当频繁的系统。比如以前 Android 系统中的 Audio 系统依赖于 alsa-lib，但后期就变为了 tinyalsa，这样的转变不应该对上层造成破坏。因而 Audio HAL 提供了统一的接口来定义它与 AudioFlinger/AudioPolicyService 之间的通信方式，这就是 audio_hw_device、audio_stream_in 及 audio_stream_out 等等存在的目的，这些 Struct 数据类型内部大多只

是函数指针的定义，是一些"壳"。当 AudioFlinger/AudioPolicyService 初始化时，它们会去寻找系统中最匹配的实现(这些实现驻留在以 audio.primary.*,audio.a2dp.*为名的各种库中)来填充这些"壳"。根据产品的不同，音频设备存在很大差异，在 Android 的音频架构中，这些问题都是由 HAL 层的 audio.primary 等库来解决的，而不需要大规模地修改上层实现。换句话说，厂商在定制时的重点就是如何提供这部分库的高效实现了。



2.2异显音频方案

2.2.1 双路触发

异声的触发会发生在异显的触发时刻，主要是对属性 media.audio.activity.pid 进行设置，同显时设置此属性为-1，如下所示：

```
SystemProperties.set("media.audio.activity.pid", String.valueOf(-1));
```

异显时，将副屏 activity 对应的 pid 设置给该属性，如下所示：

```
SystemProperties.set("media.audio.activity.pid", String.valueOf(win.mSession.mPid));
```

设置该属性后，native 层 audio 相应进程内可通过该属性值来做相应的判断，该属性值有以下作用：

1. 属性值大于 0，说明当前系统处于异显状态，并且可以获取副屏对应 apk 的 pid，此 pid 可以引导 native 层当前创建的 audiotrack 应该从那个声卡输出
2. 属性值为-1，说明当前系统处于同显状态

这部分修改位置为：

services/core/java/com/android/server/wm/WindowManagerService.java

具体修改见补丁 framework-base.patch

2.2.2 切换策略

插入 hdmi 的情况下，RK 平台默认声音从 hdmi 输出，由于双屏场景很多，本文只针对一种情况进行说明。Android 框架 AUDIO_DEVICE_OUT_AUX_DIGITAL 表示 hdmi 声卡输出，AUDIO_DEVICE_OUT_WIRED_HEADSET 表示 speaker 输出，即 codec 输出。

假如主屏是 hdmi 对应的声卡是 hdmi，副屏是 LCD 对应声卡是 speaker。同屏情况下 AudioPolicy 会选择 AUDIO_DEVICE_OUT_AUX_DIGITAL；异显情况下，当新的 audiotrack 创建时，判断 pid 是否为副屏对应的 pid，如果是，说明当前创建 audiotrack 的是副屏上的 apk，AudioPolicy 会选 AUDIO_DEVICE_OUT_WIRED_HEADSET，否则说明是主屏在创建

audiotarck，此时 AudioPolicy 会选择 AUDIO_DEVICE_OUT_AUX_DIGITAL。基本切换策略就是如此。

其它应用场景需要具体情况具体分析。

3 HAL 配置

AudioFlinger 服务开机启动时，会去遍历加载当前系统存在的硬件输出模块，如下所示

```
static const char * const audio_interfaces[] = {
        AUDIO_HARDWARE_MODULE_ID_PRIMARY,
        AUDIO_HARDWARE_MODULE_ID_HDMI,
        AUDIO_HARDWARE_MODULE_ID_A2DP,
        AUDIO_HARDWARE_MODULE_ID_USB,
};
```

其中，PRIMARY 是系统默认输出设备，HDMI 是 hdmi 声卡，A2DP 对应的是蓝牙，USB 对应的是外接 usb 声卡。AUDIO_HARDWARE_MODULE_ID_HDMI 默认是没有的，这个是为了实现异声新增。

这几个模块对应都有一个 HAL 库，如下

```
audio.primary.rk30board.so
audio.hdmi.rk30board.so
audio.a2dp.default.so
audio.usb.default.so
```

看名称可以很容易和上面的数组一一对应，其中 audio.hdmi.rk30board.so 是为了实现异声添加的库，除了修改这个之外，还需要对 audio_policy.conf 这个配置文件进行修改。

要实现异声，hdmi 和 speaker 必须各自有独立的 module，才能实现不同声音从不同设备输出，具体修改以下会一一说明。

3.1audio_policy.conf 修改

文件路径：./device/rockchip/common/audio_policy_rk30board.conf

默认情况下 hdmi 是包含在 primary 里面的，因此要将 hdmi 从 primary 中分离出来，让 hdmi 也成为一个独立的 module。

具体修改请参考：

0001-rk3399_mid-add-some-change-for-dual-audio-in-device.patch

3.2增加 HDMI 库

复制 hardware/rockchip/audio/tinyalsa_hal 到 hardware/rockchip/audio/tinyalsa_hal_hdmi，并修改 Android.mk，如下：

```
LOCAL_MODULE := audio.hdmi.$(TARGET_BOARD_HARDWARE)
```

此时，编译可生成 audio.hdmi.rk30board.so

4 补丁

双屏异声由于场景很多，补丁无法通用需要具体情况具体分析，针对不同的情况出相应的补丁。客户如果有异声需求，请在 redmine 上提出需求并大致描述一下使用场景，并在机器上运行一下 cat proc/asound/cards 命令查看声卡情况，如下所示：

```
rk3399_all:/ # cat proc/asound/cards
 0 [realtekrt5651co]: realtekrt5651co - realtekrt5651codec_hdmiin
                      realtekrt5651codec_hdmiin
 1 [ROCKCHIPSPDIF   ]: ROCKCHIP_SPDIF - ROCKCHIP,SPDIF
                      ROCKCHIP,SPDIF
 2 [rkhdmidpsound   ]: rk-hdmi-dp-soun - rk-hdmi-dp-sound
                      rk-hdmi-dp-sound
```

再次强调：必须要有两路独立声卡才可以实现异声