密级状态：绝密( ) 秘密( ) 内部( ) 公开( √ )

**Rockit项目需求分析**

（技术部，系统产品一部）

|  |  |  |
| --- | --- | --- |
| 文件状态：  [√] 正在修改  [ ] 正式发布 | 当前版本： | V0.2 |
| 作 者： | Martin.Cheng |
| 完成日期： | 2018/6/25 |
| 审 核： |  |
| 完成日期： |  |

福州瑞芯微电子有限公司

Fuzhou Rockchips Semiconductor Co . , Ltd

(版本所有,翻版必究)

版本历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 作者 | 创建日期 | 修改说明 | 备注 |
| V0.2 | Martin | 2018/6/6 | 初始版本 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

相关文档

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 文档名称 | 完成度 |
| V0.2 | Martin | Rockit项目实施计划.docx | 10% |
| V0.2 | Martin | Rockit项目需求分析.docx | 40% |
|  |  | Rockit项目概要设计.docx | 0% |
|  |  | Rockit项目详细设计.docx | 0% |
|  |  | Rockit项目软件测试.docx | 0% |

Rockit项目需求分析

[1. Rockit项目需求分析 4](#_Toc518485720)

[1. 前言 4](#_Toc518485721)

[1.1 编写目的 4](#_Toc518485722)

[1.2 文档约定 4](#_Toc518485723)

[1.3 参考文献 4](#_Toc518485724)

[2. 播放器的常见框架调研 5](#_Toc518485725)

[2.1 GStreamer框架的调研 5](#_Toc518485726)

[2.2 VLC框架的的调研 6](#_Toc518485727)

[2.3 Stagefright架构的调研 7](#_Toc518485728)

[2.4 通用播放器的一般要求 9](#_Toc518485729)

[3. 播放器的数据流需求 9](#_Toc518485730)

[3.1 IPTV的数据流 10](#_Toc518485731)

[3.2 AIVideo的数据流 10](#_Toc518485732)

[3.3 转码的数据流 11](#_Toc518485733)

[3.4 播放器的理想特性 11](#_Toc518485734)

[4. 播放器的对外接口需求 12](#_Toc518485735)

[4.1 IPTV Interface 12](#_Toc518485736)

[4.2 Android Media Interface 12](#_Toc518485737)

[5. 硬件设备接口需求 12](#_Toc518485738)

[6. 播放器的备选技术 13](#_Toc518485739)

[6.1 播放器架构备选 13](#_Toc518485740)

[6.2 版本管理技术备选 13](#_Toc518485741)

[6.3 代码风格工具 14](#_Toc518485742)

[6.4 代码自动测试工具 14](#_Toc518485743)

[6.5 代码检查高级工具 14](#_Toc518485744)

[7. 多媒体词汇表 14](#_Toc518485745)

[8. Rockit功能需求(20180614 初稿) 15](#_Toc518485746)

[9. 待定问题列表 15](#_Toc518485747)

# Rockit项目需求分析

# 前言

文档对比了Rockit和通用视频架构之间的区别。列举了典型的播放器的视频流，Rockit怎么设计才能满足所有的应用场景呢？列举了播放器需要支持的对外接口和设备接口。

## 编写目的

梳理播放器的需求。借鉴通用架构，评估实现新播放器的必要性和可行性。

## 文档约定

文档风格朴实，建议采用图表来表达思想，最大限度消除歧义，提高沟通效率。

## 参考文献

[Google C++ Style Guide](https://google.github.io/styleguide/cppguide.html)

[[FFmpeg] a complete, cross-platform solution to record, convert and stream audio and video.](https://ffmpeg.org/)

[[GStreamer] is a library for constructing graphs of media-handling components.](https://gstreamer.freedesktop.org/)

[[VLC] is a free and open source cross-platform multimedia player and framework](https://www.videolan.org/vlc/)

[[OpenFrameworks] is a community-developed cross platform toolkit for creative coding.](https://openframeworks.cc/)

[[OpenCV] is an open source computer vision and machine learning software library](https://opencv.org/)

[MediaPlayer  |  Android Developers](https://developer.android.google.cn/reference/android/media/MediaPlayer)

[HDR 视频播放  |  Android Open Source Project](https://source.android.com/devices/tech/display/hdr)

[媒体框架强化  |  Android Open Source Project](https://source.android.com/devices/media/framework-hardening)

[HDMI-CEC 控制服务  |  Android Open Source Project](https://source.android.com/devices/tv/hdmi-cec)

[DRM  |  Android Open Source Project](https://source.android.com/devices/drm)

[相机 HAL3  |  Android Open Source Project](https://source.android.com/devices/camera/camera3)

# 播放器的常见框架调研

## GStreamer框架的调研



图-1 gstreamer的架构图(数据来源: gstreamer官网)



图-2 gstreamer的数据流图(数据来源: gstreamer官网)



图-3 gsteamer的类图(数据来源: 代码阅读)

## VLC框架的的调研

如图-4所示，VLC架构图。VLC核心模块采用C语言编写，使用了大量的C语言技巧，实现了复杂的插件架构。



图-4 VLC架构图(数据来源: videolan官网)

如图-5所示，VLC的数据流图。基本的数据流走向所有的多媒体架构均是相似的，万变不离其宗。



图-5 VLC数据流图(数据来源: 代码阅读)

## Stagefright架构的调研

如图-6所示，Stagefright架构包含应用框架、Binder IPC、本地多媒体框架和OpenMAX集成层。

* 应用程序框架层面利用android.media API与多媒体硬件进行交互。
* Binder IPC利用Android的跨进程通信机制访问本地多媒体框架。
* 在本机级别，Android提供了一个多媒体框架，利用Stagefright引擎进行音频和视频录制和播放。 Stagefright附带支持的软件编解码器的默认列表，您可以使用OpenMax集成层标准实现自己的硬件编解码器。有关更多实现细节，请参阅位于frameworks / av / media中的各种MediaPlayer和Stagefright组件。
* OpenMAX IL为Stagefright提供了一种标准化的方式来识别和使用称为组件的定制基于硬件的多媒体编解码器。您必须以名为libstagefrighthw.so的共享库的形式提供OpenMAX插件。该插件将您的自定义编解码器组件链接到Stagefright。您的自定义编解码器必须根据OpenMAX IL组件标准实施。



图-6 libstagefright架构(数据来源: <https://source.android.com/devices/media/>)

如图-7所示，libstagefright的数据流图。基本的数据流走向所有的多媒体架构均是相似的，万变不离其宗。



图-7 libstagefright的数据流图(数据来源: 代码阅读)

如图-8所示，Android的media.codec架构从2018年开始引入codec2.0架构。Google给出了2年的过渡期，2020年开始废弃omx架构的支持。当前采用omx和codec2.0共存模式，OEM根据自身研发情况选择配置omx还是codec2.0。



图-8 media.codec架构的演进

## 通用播放器的一般要求

Rockit的通用多媒体架构的特性对比可以看出，Rockit有非常大的提升空间。期望Rockit能够成为一个紧凑，高效，通用，开放的多媒体播放器。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 架构特性 | Rockit | gstreamer | vlc | ffmpeg |
| 跨平台支持 | ☆ | ★ | ★ | ★ |
| 插件化架构 | ☆ | ★ | ★ | ★ |
| OSAL | low | ★ | ★ | ★ |
| 精简的API | ☆ | ★ | ★ | ★ |
| 高内聚，低耦合 | ☆ | ★ | ★ | ★ |
| 性能调试 | ☆ | ★ | ★ | ★ |
| 内存调试 | ☆ | ★ | ★ | ★ |
| Trace特性 | low | ★ | ★ | ★ |
| 扩展性 | low | ★ | ★ | ★ |
| 设备支持 | low | ★ | ★ | ★ |

表-1 Rockit与通用多媒体架构的特性对比

# 播放器的数据流需求

如何设计一个通用的播放器架构，支持几乎所有的应用场景？这个通用的播放器架构具有哪些特性才具有这样的能力呢？

## IPTV的数据流



图-8 IPTV数据流(数据来源: 原创)

## AIVideo的数据流



图-9 AI视频处理的数据流(数据来源: 原创)

## 转码的数据流



图-10 视频转码的数据流(数据来源: 原创)

## 播放器的理想特性

|  |  |
| --- | --- |
| 特性 | Rockit |
| 跨平台 | 全平台 |
| 核心特性 | * Graph-based structure allows arbitrary pipeline construction * Based on GLib 2.0 object model for object-oriented design * Compact core library * Multi-threaded pipelines are trivial and transparent to construct * Clean, simple and stable API for developers * Extremely lightweight data passing * Complete debugging system for developers * Clocking to ensure global inter-stream synchronization (a/v sync) * Quality of service (qos) to ensure best possible quality |
| 插件架构 | * 轻量级的插件系统 * 支持多种Filter插件 * 显示多种Display插件，如: Overlay&OpenGL * 支持人工智能插件(做成Filter?) |
| 全格式支持 | * 支持常见网络协议(ffmpeg) * 支持常见视频容器ffmpeg) * 支持常见音频编解码格式(ffmpeg,mediacodec,omx,vaapi,mpp) * 支持常见视频编解码格式(ffmpeg,mediacodec,omx,vaapi,mpp) |
| 高级特性 | * 多核编程&线程池技术 * 动态负载平衡&全局服务质量 * 多种调试技术和日志追踪技术 |
| 设备支持 | * HDMI * Camera * OpenGL |
| 高级接口 | * IPTV Interface * Android MediaPlayer |

表-2 播放器的理想特性

# 播放器的对外接口需求

优先级最高的两个对外接口是IPTV接口和Android Media接口。

## IPTV Interface

略，参考运营商IPTV接口代码

## Android Media Interface

略，参考MediaPlayer接口代码

# 硬件设备接口需求

Windows系统的硬件特性：

|  |  |  |
| --- | --- | --- |
| 硬件 | 硬件接口 | 重要性 |
| OpenGL | OpenGL接口/ Direct3D接口 | ★ |
| Audio | WASAPI接口 | ★ |
| OPENCL | OPENCL接口 | ★ |
| VIDEO | Video Acceleration API (VA API)  Video Decode and Presentation API(Nvidia) | ★ |
| Display | OpenGL接口/ Direct3D接口 | ★ |
| cAMERA | N/A |  |

Android的硬件特性

|  |  |  |
| --- | --- | --- |
| 硬件 | 硬件接口 | 重要性 |
| OpenGL | OpenGLES | ★ |
| Audio | ALSA接口/Audiotrack | ★ |
| OPENCL | OPENCL接口 | ★ |
| VIDEO | MediaCodec/OpenMax/mpp | ★ |
| Display | Surface/ OpenGLES | ★ |
| cAMERA | hardware/camera.h接口 | ★ |
| HDMI | 访问节点 | ★ |

# 播放器的备选技术

## 播放器架构备选

Rockit建议采用多层架构解耦，以支持尽可能多的功能。

|  |  |  |
| --- | --- | --- |
| 级别抽象 | 特性要求 | 重要性 |
| 命令行 | 调试Tools | ★ |
| 语言绑定 | Python/Java | ☆ |
| 高级接口 | IPTV/Android Media Interlace(C++语言) | ★ |
| 插件接口层 | Rockit 插件接口层/ Rockit基础层(C语言) | ★ |
| 插件实现层 | AIParser/ AICodec/ AIFilter/AISink(C语言) | ★ |
| 设备层 | HDMI/ OpenGL/Camera/Audio(C语言) | ☆ |
| OSAL层 | OS api (C语言) | ★ |

表-3 播放器的备选技术

技术路径：开发轻量级的插件架构+自行实现插件(难，工作量大)

技术路径：基于GObject开发插件架构+自行实现插件(难，工作量大)

技术路径：应用其他架构的插件架构和插件，实现播放器(简单，工作量小)

插件架构：插件接口定义，插件能力定力，插件构建工厂，插件管线，数据轻量拷贝。GStreamer核心架构层有65K行代码。假设代码不用调试，每天写300行，大约需要22天时间写完。

并行架构：多线程/多核变成/动态负载/线程池。

其他考虑：字幕流如何处理？蓝光如何处理? 3D视频如何处理？

调试考虑：内存如何调试？日志格式及日志级别如何定义？

## 版本管理技术备选

|  |  |  |
| --- | --- | --- |
| 版本细分 | 备选技术说明 | 重要性 |
| 代码管理 | github | ★ |
| 文档 | github/doc (markdown) | ★ |
| 开源库管理 | git脚本+patch | ★ |
| 跨平台编译配置 | cmake | ★ |

表-4版本管理技术备选

版本管理工具尝试将格式检查工作，让工具自动完成。测试工具提供单元测试，静态代码检查，

## 代码风格工具

|  |  |  |
| --- | --- | --- |
| 版本细分 | 备选技术说明 | 重要性 |
| 代码风格 | [Google C++ Style Guide](https://google.github.io/styleguide/cppguide.html)  [C Coding Standard[ece.cmu.edu]](https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html)  [C++ Coding Standard[ece.cmu.edu]](https://users.ece.cmu.edu/~eno/coding/CppCodingStandard.html) | ★ |
| 代码自动格式化 | [Artistic Style 3.1: A Fast Small Automatic Formatter for C, C++](http://astyle.sourceforge.net/astyle.html) | ★ |

代码风格工具提供代码风格模板和代码风格规约，即C/C++代码风格标准。在版本集成方面，提供代码自动格式化脚本，提交代码时自动格式化。

## 代码自动测试工具

|  |  |  |
| --- | --- | --- |
| 目标 | 备选技术 | 重要性 |
| 覆盖率测试 | <https://codecov.io/> (与github集成) | ★ |
| 单元测试工具 | https://github.com/google/googletest | ★ |
| 集成测试 | https://jenkins.io/index.html | ★ |

## 代码检查高级工具

|  |  |  |
| --- | --- | --- |
| 目标 | 备选技术 | 重要性 |
| 内存泄露工具 | valgrind | ★ |

# 多媒体词汇表

|  |  |
| --- | --- |
| 词汇名称 | 具体含义 |
| gstreamer | 面向对象的开源多媒体框架 |
| ffmpeg | 开源多媒体框架 |
| vlc | 开源多媒体框架 |
| opencv | 开源多媒体框架，支持计算机视觉和人工智能 |
| webrtc | 开源视频通话框架 |

# Rockit功能需求(20180614 初稿)

|  |  |  |
| --- | --- | --- |
| 模块 | 功能需求 | 优先级 |
| 命令行 | 测试命令行程序 | ★ |
| 复用器 | FFMPEG DEMUXER/MUXER | ★ |
| 编解码器 | MediaCodec/FFMPEG(Software)/VPUAPI/MPP | ★ |
| 后处理 | 后处理(AVFilter); Resampler; | ★ |
| 音视频输出 | 视频显示(overlay&opengl) / 音频输出(ALSA/) | ★ |
| 接口要求 | IPTV接口支持  Android MediaPlayer接口; Android版本无关。  蓝光支持  3D支持  音视频倍速播放  缩略图接口 | ★ |
| 高级特性 | 动态调试(日志等)  异常处理  单元测试，接口一致性测试，压力测试代码显示Buffer的管理独立化考虑  KPI(QOS)  配置管理模块 | ★ |
| 测试数据源 | 测试数据源 (生成假的视频流和音频流)  测试码流库 | ☆ |
| 其他接口 | DRM支持(TEE)  编码转码  RTP/RTSP服务器  跨平台(ANDROID/LINUX/WINDOWS) | ☆ |

# 待定问题列表

待确定问题的列表，把每一个表项都编上号，以便跟踪调查。

|  |  |
| --- | --- |
| 编号 | 问题描述 |
| ID-100 |  |
| ID-101 |  |
| ID-102 |  |
|  |  |