

Rockchip Introduction Android Libhwjpeg Interface

文件标识: RK-SM-YF-E10

发布版本: V1.0.0

日期: 2024-08-20

文件密级: ☐绝密 ☐秘密 ☐内部资料 ☒公开

免责声明

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

版权所有 © 2024 瑞芯微电子股份有限公司

超越合理使用范畴，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。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址: 福建省福州市铜盘路软件园A区18号

网址: www.rock-chips.com

客户服务电话: +86-4007-700-590

客户服务传真: +86-591-83951833

客户服务邮箱: fae@rock-chips.com

前言

概述

本文档主要介绍 jpeg 硬件编解码封装库 libhwjpeg 接口及使用方法，开发人员可通过此文档接口说明在应用中集成 jpeg 硬件编解码功能。

芯片名称	内核版本
适配所有芯片	Linux-4.19、Linux-5.10

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	陈锦森	2024-08-20	初始版本

目录

Rockchip Introduction Android Libhwjpeg Interface

- 概述
- MjpegDecoder
- MjpegEncoder

1. 概述

libhwjpeg 库用于支持 Rockchip 平台 jpeg 硬编解码，是平台 MPP（Media Process Platform）库 JPEG 编解码逻辑的封装。

其中 MjpegEncoder 类封装了硬编码相关接口，MjpegDecoder 类封装了硬解码相关接口，用于支持图片或 MJPEG 码流解码。

工程包含主要目录：

- inc: libhwjpeg 头文件
- src: libhwjpeg 实现代码
- test: libhwjpeg 测试实例

工程代码使用 bp 文件组织，在 Android SDK 环境下直接编译使用即可。

2. MjpegDecoder

MjpegDecoder 类是平台 JPEG 硬解码的封装，支持输入 JPG 图片及 MJPEG 码流，同时支持同步及异步解码方式。

```
- decodePacket(char* data, size_t size, OutputFrame_t *frameOut);
- decodeFile(const char *inputFile, const char *outputFile);
```

decodePacket & decodeFile 为同步解码方式，同步解码方式使用简单，阻塞等待解码输出，OutputFrame_t 为解码输出封装，包含输出帧宽、高、物理地址、虚拟地址等信息。

```
- sendpacket(char* data, size_t size);
- getoutframe(OutputFrame_t *frameOut);
```

sendpacket & getoutframe 用于配合实现异步解码输出，应用端开启两个线程，一路线程送输入 sendpacket，另一路线程异步取输出 getoutframe。

注意事项:

1. 解码默认输出 RAW NV12 数据
2. 硬件解码器特性，解码输出 YUV 默认经过对齐，实际使用送显前需根据 buffer 的虚宽虚高裁剪出真实图像，否则可能出现绿边等问题。libhwjpeg 提供通道实现内部自裁剪，通过属性设置 - setprop hwjpeg_dec_debug 0x10。
3. OutFrame buffer 为解码输出 dmaBuffer 封装，在解码库内部循环使用，在解码显示完成之后使用 deinitOutputFrame 释放内存。
4. 默认情况下，输出 buffer 使用内部申请的 buffer 轮转池，但是也允许外部传递 dmaBuffer fd，额外传递的 dmaBuffer fd 由 decodePacket 函数参数 OutputFrame_t->outputPhyAddr 带入。

解码使用示例:

```
MjpegDecoder decoder;
MjpegDecoder::OutputFrame_t frameOut;

memset(&frameOut, 0, sizeof(frameOut));

err = decoder.prepareDecoder();
if (!err) {
    ALOGE("failed to prepare JPEG decoder");
    goto cleanup;
}

err = decoder.decodePacket(data, size, &frameOut);
if (!err) {
    ALOGE("failed to decode packet");
    goto cleanup;
}

/* TODO - Get display for the frameOut.
 * - frame address: frameOut.MemVirAddr
 * - frame size: frameOut.OutputSize */

/* output buffer count within limits, so release frame buffer if one
   frame has been display successfully. */
decoder.deinitOutputFrame(&frameOut);

decoder.flushBuffer();
```

3. MjpegEncoder

MpijpegEncoder 类是平台 JPEG 硬编码的封装，目前主要接口提供有：

```
- encodeFrame(char *data, OutputPacket_t *packetOut);  
- encodeFile(const char *inputFile, const char *outputFile);
```

encodeFrame & encodeFile 为同步阻塞编码方式，OutputPacket_t 为编码输出封装，包含输出数据的内存地址信息。

```
- encode(EncInInfo *inInfo, EncOutInfo *outInfo);
```

encode 接口是为 cameraHal 设计的一套编码方式，接收用户传递的输入输出 dmaBuffer fd，输出 JPEG 包含编码缩略图、APP1 EXIF 头信息等信息。

注意事项：

1. OutputPacket_t 为编码输出封装，buffer 在编码库内部循环使用，在编码处理完成之后使用 deinitOutputPacket 释放内存。

编码使用示例：

```
MjpegEncoder encoder;  
MjpegEncoder::OutputPacket_t pktOut;  
  
memset(&pktOut, 0, sizeof(pktOut));  
  
err = encoder.prepareEncoder();  
if (!err) {  
    ALOGE("failed to prepare JPEG encoder");  
    goto cleanup;  
}  
  
err = encoder.updateEncodeCfg(  
    720 /*width*/, 1080 /*height*/, MjpegEncoder::INPUT_FMT_YUV420SP);  
if (!err) {  
    ALOGE("failed to update encode config");  
    goto cleanup;  
}  
  
err = encoder.encodeFrame(data, &pktOut);  
if (!err) {  
    ALOGE("failed to encode packet");  
    goto cleanup;  
}  
  
/* TODO - Get display for the PacketOut.  
 * - Packet address: pktOut.data  
 * - Packet size: pktOut.size */  
  
/* output buffer count within limits, so release frame buffer if one  
   frame has been display successful. */  
encoder.deinitOutputPacket(&pktOut);  
  
encoder.flushBuffer();
```

