



H.264 PC 解码库软件

API 参考

文档版本 09

发布日期 2013-04-08

版权所有 © 深圳市海思半导体有限公司 2013。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。

商标声明



HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

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。

深圳市海思半导体有限公司

地址：深圳市龙岗区坂田华为基地华为电气生产中心 邮编：518129

网址：<http://www.hisilicon.com>

客户服务电话：+86-755-28788858

客户服务传真：+86-755-28357515

客户服务邮箱：support@hisilicon.com



前 言

概述

本节介绍本文档的内容、对应的产品版本、适用的读者对象、行文表达约定、历史修订记录等。

产品版本

与本文档相对应的产品版本如下所示。

产品名称	产品版本
Hi3516	V100
Hi3531	V100
Hi3532	V100
Hi3521	V100
Hi3520A	V100
Hi3518	V100
Hi3516C	V100
Hi3520D	V100
Hi3515A	V100

读者对象

本参考适用于程序员阅读，描述了基于海思 H.264 PC 解码库开发的各种参考信息。使用本参考的程序员应该：

- 熟练使用 C/C++ 语言
- 掌握基本的 Windows32 调用



内容简介



本参考首先概述了 H.264 PC 解码库 API 函数种类及其关联，然后分别详细介绍了各种参考信息。本参考内容组织如下。

章节	内容
1 概述	<ul style="list-style-type: none">介绍 H.264 PC 解码库开发包组件和软硬件开发环境。阅读本主题后，您将对客户端 H.264 PC 解码库有一个整体了解。
2 API 函数	本主题供您查阅 H.264 PC 解码库的 API 参考信息，详细介绍每一个 API 接口函数。
3 通用类型及数据类型定义	介绍 API 用到的通用数据类型定义及结构体定义。
4 API 应用实例	通过实例介绍 H.264 PC 解码库 API 的使用方法。

约定

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	以本标志开始的文本表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
 警告	以本标志开始的文本表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 注意	以本标志开始的文本表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 窍门	以本标志开始的文本能帮助您解决某个问题或节省您的时间。
 说明	以本标志开始的文本是正文的附加信息，是对正文的强调和补充。



通用格式约定

格式	说明
宋体	正文采用宋体表示。
黑体	一级、二级、三级标题采用黑体。
楷体	警告、提示等内容一律用楷体，并且在内容前后增加线条与正文隔离。
“Terminal Display” 格式	“Terminal Display” 格式表示屏幕输出信息。此外，屏幕输出信息中夹杂的用户从终端输入的信息采用加粗字体表示。

表格内容约定

内容	说明
-	表格中的无内容单元。
*	表格中的内容用户可根据需要进行配置。

修改记录

修改记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修改日期	版本	修改说明
2013-04-08	09	<ul style="list-style-type: none">运行环境增加 Windows7 操作系统解码器支持的最大图像宽高修改为最大支持 5632 x 4096Hi264DecAU 和 Hi264DecImageEnhance 接口结构体 H264_DEC_FRAME_S 描述增加 H264_DEC_OUTPUT_INFO_S *pFrameInfo
2011-07-05	08	<ul style="list-style-type: none">新增 H264_OUTPUT_INFO_S在 H264_DEC_FRAME_S 中新增变量 pFrameInfo



修改日期	版本	修改说明
2010-11-26	07	<ul style="list-style-type: none">• 修改 2.1 Hi264DecCreate 中参数 uWorkMode 的描述。• 修改 2.3 Hi264DecGetInfo 中参数 uFunctionSet 的 bit[12]的描述支持 High Profile。• 修改 2.3 Hi264DecGetInfo 中参数 uFunctionSet 的 bit[13]的描述支持多线程。• 增加第二章 Hi264DecCreate 的注意事项• 增加表 1-1 的静态库为 6 个
2008-08-30	06	<ul style="list-style-type: none">• 修改 2.3 Hi264DecGetInfo 中参数 uFunctionSet 的 bit2 和 bit1 的描述。• 修改 3.2.1 H264_LIBINFO_S 中解码库能力集的 bit2 和 bit1 的描述。• 修改 3.2.3 H264_DEC_ATTR_S 中解码器工作模式的 bit2 和 bit1 的描述。• 增加适用 Hi3512 芯片的信息。
2008-05-26	05	<ul style="list-style-type: none">• 增加 2.6 Hi264DecImageEnhance。• 修改 4.2 程序实例，增加“图像增强”的程序。
2008-04-03	04	<ul style="list-style-type: none">• 修改 2.1 Hi264DecCreate 中参数 uWorkMode 的描述。• 修改 2.1 Hi264DecCreate 中参数 *pUserData 的取值范围。• 增加 2.1 Hi264DecCreate 中注意的内容。• 修改 2.3 Hi264DecGetInfo 中参数 uFunctionSet 的 bit[10]的描述支持 De-interlace 功能。• 增加 2.4 Hi264DecFrame 和 2.5 Hi264DecAU 中的参数 *pUserData 及描述。• 修改 2.5 Hi264DecAU 中的描述部分。• 增加 3.2.1 H264_LIBINFO_S 中定义部分 bit[10]的描述。• 修改 4.2 程序实例中设置解码器属性。• 删除表 1-1 动态库中的说明。
2008-01-15	03	<ul style="list-style-type: none">• 修改 2.4 Hi264DecFrame 中返回值。• 修改 2.4 Hi264DecFrame 中注意涉及结构体的由“H264_DEC_FRAME_S”改为“Hi264DecFrame”。• 增加一个 API 函数 Hi264DecAU。



修改日期	版本	修改说明
2007-11-16	02	<ul style="list-style-type: none">• 修改 2.1 Hi264DecCreate 中参数 uWorkMode 的 bit0 的含义描述，将 0、1 的含义反过来。• 修改 “Hi264DecFrame” 中注意的内容，便于英文翻译。• 将概述中表 1-1 解码库开发包组件中的 API 接口的说明中，用户工程中，应该保证先包含 hi_config.h，再包含 hi_h264api.h。两个包含的关系搞反了，已经修改完毕。• 将 “1.3 函数列表” 中 Hi264DecDestroy 的功能修描述中，“释放” 改为 “销毁”。• 将 “2.1 Hi264DecCreate” 的参数成员 “uPicHeightInMB” 的取值范围由 “[0x80,0x06]” 改为 “[0x06,0x80]”。
2007-09-05	01	第 1 次版本。



目 录

前 言.....	i
1 概述.....	2-1
1.1 描述范围.....	2-1
1.2 接口格式.....	2-2
1.3 函数列表.....	2-2
1.4 函数描述方式.....	2-3
1.5 结构体描述方式.....	2-3
2 API 函数说明	2-1
2.1 Hi264DecCreate	2-1
2.2 Hi264DecDestroy.....	2-3
2.3 Hi264DecGetInfo	2-3
2.4 Hi264DecFrame	2-6
2.5 Hi264DecAU.....	2-9
2.6 Hi264DecImageEnhance.....	2-12
3 数据类型与数据结构	3-1
3.1 通用数据类型描述.....	3-1
3.2 数据结构描述.....	3-1
3.2.1 H264_LIBINFO_S.....	3-1
3.2.2 H264_USERDATA_S.....	3-2
3.2.3 H264_DEC_ATTR_S	3-3
3.2.4 H264_OUTPUT_INFO_S	3-4
3.2.5 H264_DEC_FRAME_S.....	3-4
4 API 应用实例	4-1
4.1 流式解码流程图.....	4-1
4.2 程序实例.....	4-2



插图目录

图 4-1 解码库 API 函数使用流程图4-1



表格目录

表 1-1 解码库开发包组件.....2-1

表 1-2 解码库运行环境2-2



1 概述

1.1 描述范围

海思提供的 H.264 PC 解码库软件是一套高性能、高可靠性、兼容性良好的解码软件。解码库内部完成了 H.264 解码的主要流程，并对外提供了灵活简单的 API，用户可快速地开发应用程序。

解码库软件为用户提供 Windows 环境下的动态库和静态库两种调用形式，可更方便地开发应用程序。解码库的主要组件及相关说明如表 1-1 所示。

表1-1 解码库开发包组件

组件	名称	说明
API 接口	hi_config.h hi_h264api.h	用户工程中，应该保证先包含 hi_config.h，再包含 hi_h264api.h。
静态库	hi_h264dec_w.lib	使用静态库时，应该在编译器选项中选择忽略下面的六个库文件：libm.lib、libguide.lib 和 libirc.lib 和 libc.lib、libmmt.lib、svml_disp.lib，否则编译时会告警链接不成功。
动态库	hi_h264dec_w.lib hi_h264dec_w.dll	-
示范代码	hi_h264sample.c	以读文件解码为例，示范解码库 API 的调用方式。

用户可在多种编译环境上进行基于解码库的应用程序开发，解码库兼容微软公司的 Windows 2000 或更高版本的主流视窗操作系统，兼容 Intel 公司和 AMD 公司自 2002 年来推出的绝大部分面向 PC 机的 CPU 芯片组。其主要开发以及运行环境说明如表 1-2 所示。



表1-2 解码库运行环境

分类	兼容配置	推荐配置	说明
编译器	Visual C++6.0 Visual Studio.net2003 Intel C++ 9.0 / 10.0	Visual Studio.net 2003	无。
操作系统	Windows 98 Windows 2000 Windows XP Windows 2003 Windows Vista Windows 7 (32bit) Windows 7 (64bit)	Windows XP Windows 7	在 Windows 98 系统上，解码库将进入衰退工作模式，解码性能较低。
硬件	Intel P3 系列 Intel P4 系列 Intel Core 系列 AMD Athlon64 系列 AMD Sempron 系列 AMD Athlon 系列	CPU 主频在 3.0GHz 以上、内存大小在 512MB 以上的 PC	在 Intel P3、AMD AthlonXP 或更早期的 CPU 上，解码库将进入衰退工作模式，解码性能较低。

1.2 接口格式

无。

1.3 函数列表

函数	功能	页码
2.1 Hi264DecCreate	创建、初始化解码器句柄。	v
Hi264DecDestroy	销毁解码器句柄。	2-3
修改 2.3 Hi264DecGetInfo	查询解码库版本信息和当前版本能力集。	iv
Hi264DecFrame	对输入的一段码流进行解码并按帧输出图像。	v
Hi264DecAU	对输入的一帧图像对应的码流进行解码并立即输出此帧图像。	iv
Hi264DecImageEnhance	解码后的图像增强。	2-12



1.4 函数描述方式

本章用 6 个域对 API 参考信息进行描述。

参数域	作用
目的	简要描述 API 的主要功能。
语法	列出 API 的语法样式。
描述	简要描述 API 的工作过程。
参数	列出 API 的参数、参数说明及参数属性。
返回值	列出 API 的返回值及返回值说明。
注意	使用 API 时应注意的事项。

1.5 结构体描述方式

参数域	作用
说明	简要描述结构体所实现的功能。
定义	列出结构体的定义。
注意事项	列出结构体的注意事项。



2 API 函数说明

2.1 Hi264DecCreate

【目的】

创建、初始化解码器句柄。

【语法】

```
HI_HDL Hi264DecCreate(H264_DEC_ATTR_S *pDecAttr );
```

【描述】

创建解码器句柄。在解码开始时，分配解码空间和初始化解码器相关的变量及状态，设置解码器输入码流类型、输出图像格式、解码器支持的最大图像的宽高、解码器支持的最大参考帧数目等解码器属性。

上层应用可以使用多线程创建多个解码器，实现多路解码。

【参数】

参数	成员	取值范围	输入/输出	描述
pDecAttr	uPictureFormat	0x00	输入	输出图像格式。 0x00 表示输出图像为 4:2:0 格式。解码库暂不支持其他格式。
	uStreamInType	0x00	输入	输入码流格式。 0x00 表示输入码流为以“00 00 01”为 nalu 分割符的 H.264 码流。
	uPicWidthInMB	[0x06,0x160]	输入	解码器支持的图像宽度。 (以 MB 为单位。超出取值范围时，默认为 120，即 1080p 图像宽。)
	uPicHeightInMB	[0x02,0x100]	输入	解码器支持的图像高度。 (以 MB 为单位。超出取值范围时，默认为 68，即 1080p 图像高。)



参数	成员	取值范围	输入/输出	描述
	uBufNum	[0x01,0x10]	输入	分配给解码器可用作参考帧的缓冲区数目。 (超出取值范围时，默认为 0x04)
	uWorkMode	-	输入	bit[31:6]: 保留。 bit[5]: 启动多线程解码。 0: 不启动。 1: 启动。 bit[4]: 启动解码库内部 Deinterlace 功能。 0: 不启动。 1: 启动。 bit[3:1]: 保留。 bit[0]: 解码器工作模式。 0: 快速输出模式，即解完一帧立即输出。 1: H.264 协议定义的图像输出模式。
	*pUserData	-	输入	指向输入的用户数据，数据类型请参见 H264_USERDATA_S 定义。 (解码器暂不解析此参数)
	uReserved	0	输入	保留字。

【返回值】

返回值	宏定义	描述
0	NULL	解码器创建失败（内存分配失败或者参数配置错误）。
非 0	-	解码器创建成功，返回值为解码器句柄。

【注意】

- 只有解码顺序和图像输出顺序一致时，才能使用快速输出模式，一般而言，如果图像不包含 B 帧，则可以使用快速输出模式以降低输出时延。
- 当启动解码器内部 Deinterlace 功能时，只有输入视频采用场模式编码时才有效；对于帧模式的视频图像，解码器将自动跳过 Deinterlace 过程。
- 多线程解码适用于单通道解码且每帧包含多个 slice 的应用场合，对于多通道解码或每帧仅包含单 slice 的应用场合不推荐使用。



- 启动多线程解码需要保证输入码流在 slice 边界不做 Deblock 滤波。如果码流不能满足上述条件，解码器会强制在 slice 边界不做 Deblock 滤波。
- 多线程解码只在多核 CPU 解码单路图像时才能提高解码性能，由于线程调度需要占用一定的开销，所以对于单核 CPU 的情况推荐不要使用多线程解码。
- 以上各种工作模式互相独立，用户可单独或组合设置。
- 对超大图像的解码，创建解码器时应留意系统内存容量，如果内存不够，创建解码器可能会失败。

2.2 Hi264DecDestroy

【目的】

销毁解码器句柄。

【语法】

```
void Hi264DecDestroy(HI_HDL hDec);
```

【描述】

解码结束后，销毁解码器工作时分配的内存空间，以防止内存泄漏。

【参数】

参数	成员	取值范围	输入/输出	描述
hDec	-	-	输入	待销毁的解码器句柄。

【返回值】

无。

【注意】

销毁后的句柄需要手动置空。

2.3 Hi264DecGetInfo

【目的】

查询解码库版本信息和当前版本能力集。

【语法】

```
HI_S32 Hi264DecGetInfo(H264_LIBINFO_S *pLibInfo);
```

【描述】

用户可在创建解码器之前调用此函数察看解码库版本信息、解码库能力集。



【参数】

参数	成员	取值范围	输入/ 输出	描述
pLibInfo	uMajor	-	输出	解码库主编号
	uMinor	-	输出	解码库次编号
	uRelease	-	输出	解码库发布编号
	uBuild	-	输出	解码库建构编号
	sVersion	-	输出	解码库版本信息
	sCopyRight	-	输出	解码库版权信息



参数	成员	取值范围	输入/ 输出	描述
	uFunctionSet	-	输出	解码器能力信息，含义如下： bit[31:14]：保留位。 bit[13] 0：不支持多线程解码。 1：支持多线程解码。 bit[12] 0：不支持 High Profile。 1：支持 High Profile。 bit[10] 0：不支持内部 Deinterlace。 1：支持内部 Deinterlace。 bit[9] 0：不支持 cabac 解码。 1：支持 cabac 解码。 bit[8] 0：不支持加权预测解码。 1：支持加权预测解码。 bit[7] 0：不支持 B-slice 解码。 1：支持 B-slice 解码。 bit[6] 0：不支持 MBAFF 解码。 1：支持 MBAFF 解码。 bit[5] 0：不支持 PAFF 解码。 1：支持 PAFF 解码。 bit[4] 0：不支持 FMO 解码。 1：支持 FMO 解码。 bit[3]：保留位。 bit[2] 0：不支持 Hi351x 数字水印解码。 1：支持 Hi351x 数字水印解码。 bit[1]：保留位。 bit[0] 0：支持快速图像输出模式。 1：不支持快速图像输出模式。



参数	成员	取值范围	输入/ 输出	描述
	uPictureFormat	0x00	输出	解码库当前支持的图像格式。 0x00 表示仅支持 4:2:0 格式的图像。
	uStreamInType	0x00	输出	解码库当前支持码流格式。 0x00 表示仅支持以“00 00 01”为 nalu 分隔符的 H.264 码流。
	uPicWidth	0x1600	输出	解码库当前支持的最大图像宽度。 (以像素为单位)
	uPicHeight	0x1000	输出	解码库当前支持的最大图像高度。 (以像素为单位)
	uBufNum	0x10	输出	解码库最大可支持的参考帧数目。
	uReserved	-	输出	保留字。

【返回值】

返回值	宏定义	描述
0	-	成功获取解码库信息。
-1	-	参数输入错误，获取失败。

【注意】

无。

2.4 Hi264DecFrame

【目的】

对输入的一段码流进行解码并按帧输出图像。

【语法】

```
HI_S32 Hi264DecFrame
(
    HI_HDL hDec,
    HI_U8 *pStream,
    HI_U32 iStreamLen,
    HI_U64 ullPTS,
```



```
H264_DEC_FRAME_S *pDecFrame,  
HI_U32 uFlags  
);
```

【描述】

本函数仅支持流式解码，对于以“00 00 01”为 nalu 分隔符的连续、线性 H.264 码流，用户可从任意起始地址、任意长度配置给解码器解码。

【参数】

参数	成员	取值范围	输入/输出	描述
hDec	-	-	输入	解码器句柄。
pStream	-	-	输入	码流起始地址。
iStreamLen	-	-	输入	码流长度（字节为单位）。
ullPTS	-	-	输入	时间戳信息。
pDecFrame	pY	-	输出	输出 Y 分量地址。
	pU	-	输出	输出 U 分量地址。
	pV	-	输出	输出 V 分量地址。
	uWidth	-	输出	输出图像宽度。 (以像素为单位)
	uHeight	-	输出	输出图像高度。 (以像素为单位)
	uYStride	-	输出	输出 Y 分量的 stride。 (以像素为单位)
	uUVStride	-	输出	输出 U/V 分量 stride。 (以像素为单位)
	uCroppingLeftOffset	-	输出	输出图像左边裁减量。 (以像素为单位)
	uCroppingRightOffset	-	输出	输出图像右边裁减量。 (以像素为单位)
	uCroppingTopOffset	-	输出	输出图像上边裁减量。 (以像素为单位)
	uCroppingBottomOffset	-	输出	输出图像下边裁减量。 (以像素为单位)



参数	成员	取值范围	输入/ 输出	描述
	uDpbIdx	-	输出	输出图像缓冲区编号。（暂不使用）
	bError	0 或 1	输出	当前图像错误标示。 0: 输出图像无错。 1: 输出图像有错。
	uPicFlag	0, 1, 2	输出	输出图像属性。 0: 输出为帧。 1: 输出为顶场。 2: 输出为底场。
	bIntra	0 或 1	输出	输出图像是否为 IDR（Instantaneous Decoding Refresh）帧标示。 0: 非 IDR 帧。 1: IDR 帧。
	ullPTS	-	输出	输出图像时间戳信息。
	uPictureID	-	输出	输出图像序号。
	uReserved	-	输出	保留字。
	*pUserData	-	输出	指向输出的用户数据。
	*pFrameInfo	-	输出	输出当前帧信息，包含一帧码流字节数，各种类型宏块的个数。
	uFlags	0 或 1	输入	解码模式。 0: 正常解码。 1: 解码完毕并要求解码器输出残留图像。

【返回值】

返回值	宏定义	含义
0	HI_H264DEC_OK	函数执行成功，有一帧图像输出。
-1	HI_H264DEC_NEED_MORE_BITS	剩余码流不够解码一帧的数据，需要重新配置更多的码流。 uFlags 为 0 时才会返回此值。



返回值	宏定义	含义
-2	HI_H264DEC_NO_PICTURE	解码器内部残留的图像已经全部输出完毕。 uFlags 为 1 时才会返回此值。
-3	HI_H264DEC_ERR_HANDLE	解码器句柄为空或输出图像结构体为空。

【注意】

在调用本函数过程中需要注意以下两点：

- 在解码过程中，用户应该将码流分段，并依次配置给解码器。当用户调用此函数，将一段码流配置给解码器之后，应对函数的参数做如下配置：
pStream=NULL; iStreamLen=0; uFlags=0。然后循环调用此函数，直到函数返回 HI_H264DEC_NEED_MORE_BITS 时才能再次配置一段新的码流。
在上述循环调用的过程中，如果函数返回 HI_H264DEC_OK 则表明有一帧图像输出，用户必须在循环调用内部及时处理存储在 pDecFrame 中的图像。
- 在解码结束时，为了输出解码器内部可能的残留图像，用户可对函数的参数做如下配置：uFlags=1、pStream=NULL。然后循环调用此函数，直到函数返回 HI_H264DEC_NO_PICTURE 时才能停止解码。
在上述循环调用的过程中，如果函数返回 HI_H264DEC_OK 则表明有一帧图像输出，用户必须在循环调用内部及时处理存储在 pDecFrame 中的图像。

解码函数提供时间戳透传功能，输入的时间戳将保存在当前码流解码后的图像结构体 H264_DEC_FRAME_S 中，并随解码图像一起输出。详细信息请参见“3.2.5 H264_DEC_FRAME_S”。

2.5 Hi264DecAU

【目的】

对输入的一帧图像对应的码流进行解码并立即输出此帧图像。

【语法】

```
HI_S32 Hi264DecAU
(
    HI_HDL hDec,
    HI_U8 *pStream,
    HI_U32 iStreamLen,
    HI_U64 ullPTS,
    H264_DEC_FRAME_S *pDecFrame,
    HI_U32 uFlags
);
```



【描述】

本函数仅支持按帧解码，要求每次配送的仅包含一帧图像的码流内部必须是符合以“00 00 01”为 nalu 分隔符的 H.264 标准格式。

【参数】

参数	成员	取值范围	输入/ 输出	描述
hDec	-	-	输入	解码器句柄。
pStream	-	-	输入	码流起始地址。
iStreamLen	-	-	输入	码流长度（字节为单位）。
ullPTS	-	-	输入	时间戳信息。
pDecFrame	pY	-	输出	输出 Y 分量地址。
	pU	-	输出	输出 U 分量地址。
	pV	-	输出	输出 V 分量地址。
	uWidth	-	输出	输出图像宽度。 (以像素为单位)
	uHeight	-	输出	输出图像高度。 (以像素为单位)
	uYStride	-	输出	输出 Y 分量的 stride。 (以像素为单位)
	uUVStride	-	输出	输出 U/V 分量 stride。 (以像素为单位)
	uCropping LeftOffset	-	输出	输出图像左边裁减量。 (以像素为单位)
	uCropping RightOffset	-	输出	输出图像右边裁减量。 (以像素为单位)
	uCropping TopOffset	-	输出	输出图像上边裁减量。 (以像素为单位)
	uCropping BottomOffset	-	输出	输出图像下边裁减量。 (以像素为单位)
	uDpbIdx	-	输出	输出图像缓冲区编号。（暂不使用）



参数	成员	取值范围	输入/输出	描述
	bError	0 或 1	输出	当前图像错误标示。 0: 输出图像无错。 1: 输出图像有错。
	uPicFlag	0, 1, 2	输出	输出图像属性。 0: 输出为帧。 1: 输出为顶场。 2: 输出为底场。
	bIntra	0 或 1	输出	输出图像是否为 IDR 帧标示。 0: 非 IDR 帧。 1: IDR 帧。
	ullPTS	-	输出	输出图像时间戳信息。
	uPictureID	-	输出	输出图像序号。
	uReserved	-	输出	保留字。
	*pUserData	-	输出	指向输出的用户数据。
	*pFrameInfo	-	输出	输出当前帧信息, 包含一帧码流字节数, 各种类型宏块的个数。
uFlags	-	-	输入	保留字。

【返回值】

返回值	宏定义	含义
0	HI_H264DEC_OK	函数执行成功, 有一帧图像输出。
-2	HI_H264DEC_NO_PICTURE	没有有效图像输出。
-3	HI_H264DEC_ERR_HANDLE	解码器句柄为空或输出图像结构体为空。

【注意】

本函数和 [Hi264DecFrame](#) 是并列关系, 用户根据需要调用其中一个即可。在调用本函数过程中需要注意以下两点:

- 在解码过程中, 用户应该将码流以帧为单位进行分割, 每次配送给此函数的码流必须且只能包含一帧图像, 否则会导致图像输出异常。



在上述循环调用的过程中，如果函数返回 **HI_H264DEC_OK** 则表明有一帧图像输出，返回 **HI_H264DEC_NO_PICTURE** 则表明用户本次配置的码流无法解出一帧图像。

- 解码器默认用户每次调用函数时配送的码流仅包含一帧图像，因此无论码流是否完整，都会在函数执行后将本帧图像输出。

解码函数提供时间戳透传功能，输入的时间戳将保存在当前码流解码后的图像结构 **H264_DEC_FRAME_S** 中，并随解码图像一起输出。详细信息请参见“[3.2.5 H264_DEC_FRAME_S](#)”。

2.6 Hi264DecImageEnhance

【目的】

解码后的图像增强。

【语法】

```
HI_S32 Hi264DecImageEnhance  
(  
    HI_HDL hDec,  
    H264_DEC_FRAME_S *pDecFrame,  
    HI_U32 uEnhanceCoeff  
);
```

【描述】

成功解码一幅图像后，做解码图像后处理，改善某些场景下图像质量。

【参数】

参数	成员	取值范围	输入/输出	描述
hDec	-	-	输入	解码器句柄。
pDecFrame	pY	-	输入/输出	输出 Y 分量地址。
	pU	-	输入/输出	输出 U 分量地址。
	pV	-	输入/输出	输出 V 分量地址。
	uWidth	-	输入/输出	输出图像宽度。 (以像素为单位)
	uHeight	-	输入/输出	输出图像高度。 (以像素为单位)
	uYStride	-	输入/输出	输出 Y 分量的 stride。 (以像素为单位)



参数	成员	取值范围	输入/输出	描述
	uUVStride	-	输入/输出	输出 U/V 分量 stride。 (以像素为单位)
	uCroppingLeftOffset	-	输入/输出	输出图像左边裁减量。 (以像素为单位)
	uCroppingRightOffset	-	输入/输出	输出图像右边裁减量。 (以像素为单位)
	uCroppingTopOffset	-	输入/输出	输出图像上边裁减量。 (以像素为单位)
	uCroppingBottomOffset	-	输入/输出	输出图像下边裁减量。 (以像素为单位)
	uDpbIdx	-	输入/输出	输出图像缓冲区编号。 (暂不使用)
	bError	0 或 1	输入/输出	当前图像错误标示。 0: 输出图像无错。 1: 输出图像有错。
	uPicFlag	0, 1, 2	输入/输出	输出图像属性。 0: 输出为帧。 1: 输出为顶场。 2: 输出为底场。
	bIntra	0 或 1	输入/输出	输出图像是否为 IDR 帧标示。 0: 非 IDR 帧。 1: IDR 帧。
	uIPTS	-	输入/输出	输出图像时间戳信息。
	uPictureID	-	输入/输出	输出图像序号。
	uReserved	-	输入/输出	保留字。
uEnhanceCoeff	*pUserData	-	输入/输出	指向输出的用户数据。
	*pFrameInfo	-	输出	输出当前帧信息, 包含一帧码流字节数, 各种类型宏块的个数。
uEnhanceCoeff	-	(0, 128]	输入	图像增强系数, 一般取值范围是 [30, 50], 数值越大对图像的改变越大, 推荐值为 40。



【返回值】

返回值	宏定义	含义
0	HI_H264DEC_OK	函数执行成功，有一帧图像增强并可以输出。
-3	HI_H264DEC_ERR_HANDLE	解码库句柄为空或输入参数错误。

【注意】

结构体指针 pDecFrame 既是输入参数也是输出参数，用户在获取一帧图像后，应立即将 [Hi264DecFrame](#) 或 [Hi264DecAU](#) 的输出参数 pDecFrame 作为 [Hi264DecImageEnhance](#) 的输入参数，无需做任何修改。



3 数据类型与数据结构

3.1 通用数据类型描述

在 win32 环境下，API 用到的主要数据类型定义如下：

```
typedef unsigned char    HI_U8;  
typedef unsigned char    HI_UCHAR;  
typedef unsigned short   HI_U16;  
typedef unsigned int     HI_U32;  
typedef signed char      HI_S8;  
typedef signed short     HI_S16;  
typedef signed int       HI_S32;  
typedef __int64          HI_S64;  
typedef unsigned __int64 HI_U64;  
typedef char             HI_CHAR;  
typedef char*            HI_PCHAR;  
typedef void*            HI_HDL;
```

3.2 数据结构描述

3.2.1 H264_LIBINFO_S

【说明】

解码库版本、版权和能力集信息。

【定义】

```
/* 解码库版本、版权和能力集信息数据结构 */  
typedef struct hiH264_LIBINFO_S  
{  
    HI_U32  uMajor;           /* 主编号 */  
    HI_U32  uMinor;          /* 次编号 */
```



```
HI_U32 uRelease;          /* 发布编号 */
HI_U32 uBuild;            /* 建构编号 */
const HI_CHAR* sVersion;  /* 版本信息 */
const HI_CHAR* sCopyRight; /* 版权信息 */
HI_U32 uFunctionSet;      /* 解码库能力集 */
/* 除bit0位外, 其他标志位为1时表示当前版本支持该功能, 为0时表示当前版本不支持该功能 */
/* bit0 : 快速输出模式 */
/* bit1 : 保留位 */
/* bit2 : Hi351x数字水印 */
/* bit3 : 保留位 */
/* bit4 : FMO 解码 */
/* bit5 : PAF 解码 */
/* bit6 : MBAFF 解码 */
/* bit7 : B 片解码 */
/* bit8 : 加权预测 */
/* bit9 : CABAC算术解码 */
/* bit10 : 内部集成Deinterlace */
/* bit12 : High profile */
/* bit13 : 多线程解码 */
/* bit11 ~ bit 31 : 保留位 */
HI_U32 uPictureFormat; /* 支持的输出图像格式 */
/* 0x00:当前仅支持YUV420图像格式 */
HI_U32 uStreamInType; /* 输入码流格式 */
/* 0x00:当前仅支持以“00 00 01”为nalu分割符的流式 */
/* H.264码流 */
HI_U32 uPicWidth;      /* 最大图像宽度(以像素为单位) */
HI_U32 uPicHeight;     /* 最大图像高度(以像素为单位) */
HI_U32 uBufNum;        /* 最大参考帧数目 */
HI_U32 uReserved;      /* 保留字 */
} H264_LIBINFO_S;
```

【注意事项】

无。

3.2.2 H264_USERDATA_S

【说明】

用户私有数据信息。

【定义】



```
/* 用户私有数据结构 */
typedef struct hiH264_USERDATA_S
{
    HI_U32  uUserDataTypes;          /* 用户数据类型 */
    HI_U32  uUserDataSize;          /* 用户数据长度 */
    HI_UCHAR *pData;                /* 用户数据缓冲区 */
    struct hiH264_USERDATA_S *pNext; /* 指向下一段用户数据 */
} H264_USERDATA_S;
```

【注意事项】

无。

3.2.3 H264_DEC_ATTR_S

【说明】

解码器属性信息。

【定义】

```
/* 解码器属性数据结构 */
typedef struct hiH264_DEC_ATTR_S
{
    HI_U32  uPictureFormat;          /* 解码器输出图像格式 */
                                        /* 0x00: 目前解码库只支持YUV420图像格式 */
    HI_U32  uStreamInType;          /* 输入码流格式 */
                                        /* 0x00: 目前解码库只支持以“00 00 01”为 */
                                        /* nalu分割符的流式H.264码流 */
    HI_U32  uPicWidthInMB;          /* 图像宽度(以宏块为单位) */
    HI_U32  uPicHeightInMB;         /* 图像高度(以宏块为单位) */
    HI_U32  uBufNum;                /* 参考帧数目 */
    HI_U32  uWorkMode;              /* 解码器工作模式 */
                                        /* bit0: 0: 快速输出模式; 1: 正常输出模式 */
                                        /* bit1 ~ bit2: */
                                        /* 00: 仅解码图像 */
                                        /* 01: 保留 */
                                        /* 10: 解码Hi351x数字水印 */
                                        /* 11: 保留 */
                                        /* bit 4: 0: 场图像不做deinterlace */
                                        /*          1: 场图像做deinterlace */
                                        /* bit 5: 0: 使用单线程解码 */
                                        /*          1: 多slice使用多线程解码 */
}
```



```
/* bit6 ~ bit31: 保留位*/  
H264_USERDATA_S *pUserData; /* 用户私有数据 */  
HI_U32 uReserved; /* 保留字 */  
} H264_DEC_ATTR_S;
```

【注意事项】

无。

3.2.4 H264_OUTPUT_INFO_S

【说明】

解码器输出帧信息。

【定义】

```
/* 解码器输出帧信息数据结构 */  
typedef struct hiH264_OUTPUT_INFO_S  
{  
    HI_U32 uPicBytes; /* 当前帧的字节数 */  
    HI_U32 uI4MbNum; /* 当前帧的I4x4宏块个数 */  
    HI_U32 uI8MbNum; /* 当前帧的I8x8宏块个数 */  
    HI_U32 uI16MbNum; /* 当前帧的I16x16宏块个数 */  
    HI_U32 uP16MbNum; /* 当前帧的P16x16宏块个数 */  
    HI_U32 uP16x8MbNum; /* 当前帧的P16x8宏块个数 */  
    HI_U32 uP8x16MbNum; /* 当前帧的P8x16宏块个数 */  
    HI_U32 uP8MbNum; /* 当前帧的P8x8宏块个数 */  
    HI_U32 uPskipMbNum; /* 当前帧的PSkip宏块个数 */  
    HI_U32 uIpcmMbNum; /* 当前帧的IPCM宏块个数 */  
} H264_OUTPUT_INFO_S;
```

【注意事项】

无。

3.2.5 H264_DEC_FRAME_S

【说明】

解码器输出图像信息。

【定义】

```
/* 解码器输出图像信息数据结构 */  
typedef struct hiH264_DEC_FRAME_S  
{  
    HI_U8 *pY; /* Y分量地址 */
```



```
HI_U8    *pU;                /* U分量地址 */
HI_U8    *pV;                /* V分量地址 */
HI_U32    uWidth;            /* 图像宽度(以像素为单位) */
HI_U32    uHeight;           /* 图像高度(以像素为单位) */
HI_U32    uYStride;          /* 输出Y分量的stride (以像素为单位) */
HI_U32    uUVStride;         /* 输出U/V分量stride (以像素为单位) */
HI_U32    uCroppingLeftOffset; /* 图像裁减信息:左边界裁减像素数 */
HI_U32    uCroppingRightOffset; /* 图像裁减信息:右边界裁减像素数 */
HI_U32    uCroppingTopOffset;  /* 图像裁减信息:上边界裁减像素数 */
HI_U32    uCroppingBottomOffset; /* 图像裁减信息:下边界裁减像素数 */
HI_U32    uDpbIdx;           /* 输出图像在dpb中的序号 */
HI_U32    tPicFlag;          /* 图像类型: 0:帧; 1:顶场; 2:底场 */
HI_U32    bError;            /* 图像是否有错: 0:正确;1:图像有错 */
HI_U32    bIntra;            /* 图像是否为IDR帧: 0:非IDR帧;1:IDR帧 */
HI_U64    ullPTS;            /* 时间戳 */
HI_U32    uPictureID;        /* 图像序号 */
HI_U32    uReserved;         /* 保留字 */
H264_USERDATA_S *pUserData;  /* 指向用户私有数据 */
H264_OUTPUT_INFO_S *pFrameInfo; /* 指向当前帧输出信息 */
} H264_DEC_FRAME_S;
```

【注意事项】

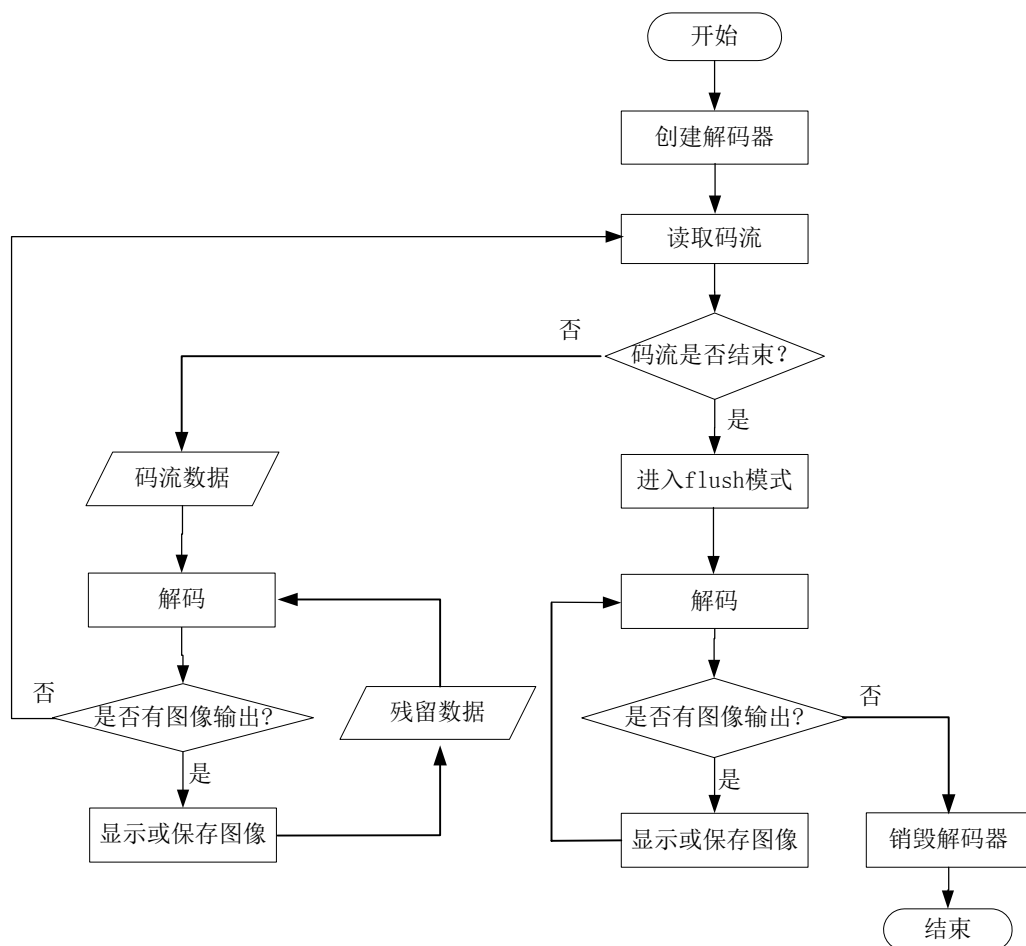
无。



4 API 应用实例

4.1 流式解码流程图

图4-1 解码库 API 函数使用流程图





4.2 程序实例

```
H264_DEC_ATTR_S  dec_attrbute;
H264_DEC_FRAME_S dec_frame;
HI_HDL handle = NULL;
HI_S32 end      = 0;
HI_U8  buf[0x80000];          /* 码流缓冲区 */
FILE *h264 = NULL;            /* H.264 码流文件 */
FILE *yuv = NULL;             /* 存放YUV图像的文件 */
HI_U32 ImageEnhanceEnable = 0; /* 0: 关闭图像增强; 1: 使能图像增强 */
HI_U32 StrenthCoeff = 40;     /* 图像增强系数 */

/* 打开264码流文件和存储YUV图像的文件 */
h264 = fopen(argv[1], "rb");
yuv  = fopen(argv[2], "wb");
if(NULL == h264 || NULL == yuv)
{
    goto exit;
}

/* 设置解码器属性 */
dec_attrbute.uBufNum      = 16; /* 16个参考帧 */
dec_attrbute.uPicHeight   = 68; /* 1080p图像大小 */
dec_attrbute.uPicWidth    = 120;
dec_attrbute.pUserData    = NULL; /* 无用户私有数据 */
dec_attrbute.uStreamInType = 0;   /* 输入以00 00 00 01 ...起始的码流 */
dec_attrbute.uWorkMode    = 0x31; /* 正常输出模式, 启动内部Deinterlace */
                                /* 使用多线程解码 */

/* 创建解码器 */
handle = Hi264DecCreate(&dec_attrbute);
if(NULL == handle)
{
    goto exit;
}

/* 开始解码流程 */
while (!end)
{
    HI_U32 len = fread(buf, 1, sizeof(buf), h264); /* 读取一段码流 */
```



```
HI_U32  flags = (len>0)?0:1;          /* 码流是否结束标志 */
HI_S32  result = 0;

result = Hi264DecFrame(handle, buf, len, 0, &dec_frame, flags);
while(HI_H264DEC_NEED_MORE_BITS != result)
{
    if(HI_H264DEC_NO_PICTURE == result)
    {
        end = 1;          /* 解码器内所有图像都已输出, 解码结束 */
        break;
    }
    if(HI_H264DEC_OK == result) /* 有一帧图像输出 */
    {
        /* 图像增强 */
        if(ImageEnhanceEnable)
        {
            Hi264DecImageEnhance(handle, &dec_frame, StrenthCoeff);
        }
        /* 存储一幅图像 */
        const HI_U8 *pY = dec_frame.pY;
        const HI_U8 *pU = dec_frame.pU;
        const HI_U8 *pV = dec_frame.pV;
        HI_U32 width    = dec_frame.uWidth;
        HI_U32 height    = dec_frame.uHeight;
        HI_U32 yStride   = dec_frame.uYStride;
        HI_U32 uvStride  = dec_frame.uUVStride;

        fwrite(pY, 1, height* yStride, yuv);
        fwrite(pU, 1, height* uvStride/2, yuv);
        fwrite(pV, 1, height* uvStride/2, yuv);
    }
    result = Hi264DecFrame(handle, NULL, 0, 0, &dec_frame, flags);
}

/* 销毁解码器, 释放句柄 */
Hi264DecDestroy(handle);
Handle = NULL;
exit:
if(NULL != h264)
fclose(h264);
if(NULL != yuv)
fclose(yuv);
```