H.264 PC Decoding Library Software

API Reference

Issue   **09**

Date   **2013-04-08**

HiSilicon Technologies Co., Ltd.

Address: Huawei Industrial Base

Bantian, Longgang

Shenzhen 518129

People's Republic of China

Website: http://www.hisilicon.com

Email: support@hisilicon.com

About This Document

Purpose

This document describes the document contents, related product versions, intended audience, conventions and update history.

Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3516	V100
Hi3531	V100
Hi3532	V100
Hi3521	V100
Hi3520A	V100
Hi3518	V100
Hi3516C	V100
Hi3520D	V100
Hi3515A	V100

Intended Audience

This document is intended for the programmers who meet the following requirements:

- Be familiar with C/C++ programming language
- Be familiar with 32-bit Windows environment

Conventions

Symbol Conventions

The following symbols may be found in this document. They are defined as follows.

Symbol	Description
DANGER	Indicates a hazard with a high level of risk which, if not avoided, will result in death or serious injury.
WARNING	Indicates a hazard with a medium or low level of risk that, if not avoided, could result in minor or moderate injury.
CAUTION	Indicates a potentially hazardous situation that, if not avoided, could cause equipment or component damage, data loss, and performance degradation, or unexpected results.
☜ TIP	Indicates a tip that may help you solve a problem or save time.
📖 NOTE	Provides additional information to emphasize or supplement important points of the main text.

General Conventions

Convention	Description
Times New Roman	Normal paragraphs are in Times New Roman.
Boldface	Names of files, directories, folders, and users are in **boldface**. For example, log in as user **root**.
Italic	Book titles are in *italics*.
Courier New	Terminal display is in Courier New.

Table Contents

Content	Description
-	Not applicable
*	A wild card

Change History

Updates between document issues are cumulative. Therefore, the latest document issue contains all updates made in previous issues.

Updates in Issue 09 (2013-04-08)

This issue is the sixth official release, which incorporates the following changes:

Chapter 1 Overview

Table 1-2 is updated.

Chapter 2 API Description

The **Parameter** part of Hi264DecCreate is updated.

The **Parameter** part of Hi264DecGetInfo is updated.

The **Parameter** part of Hi264DecAU is updated.

The **Parameter** part of Hi264DecImageEnhance is updated.

Updates in Issue 07 (2011-07-05)

This issue is the fifth official release, which incorporates the following changes:

Chapter 3 Data Types and Data Structures

H264_OUTPUT_INFO_S is updated.

The **Definition** part of H264_DEC_FRAME_S is updated.

Updates in Issue 06 (2011-04-29)

This issue is the fifth official release, which incorporates the following changes:

Chapter 1 Overview

The number of library files to be ignored in Table 1-1 is changed from 4 to 6.

Chapter 2 API Description

The **Note** part of Hi264DecCreate is updated.

Chapter 3 Data Types and Data Structures

The **Definition** part of H264_DEC_ATTR_S is updated.

Chapter 4 API Application Instances

The description in section 4.2 is updated.

Updates in Issue 05 (2008-11-10)

This issue is the fourth official release, which incorporates the following changes:

Chapter 2 API Description

Descriptions of bit2 and bit1 of the parameter FunctionSet are changed in section 2.3 "Hi264DecGetInfoare."

Chapter 2 API Description

Information about the Hi3512 is added.

Chapter 3 Data Types and Data Structures

- Descriptions of bit2 and bit1 of the decoding library function set are changed in section 3.2.1 "H264_LIBINFO_S."

- Descriptions of bit2 and bit1 of the decoder mode are changed in section 3.2.3 "H264_DEC_ATTR_S."

Updates in Issue 04 (2008-05-26)

This issue is the third official release, which incorporates the following changes:

Chapter 2 API Description

Information about Hi264DecImageEnhance is added.

Chapter 4 API Application Instances

Application instances are changed and the "Picture Enhance" program is added in section 4.2 "Program Instances."

Updates in Issue 03 (2008-04-03)

This issue is the second official release, which incorporates the following changes:

Chapter 1 Overview

Descriptions of the dynamic library are deleted in table 1-1.

Chapter 2 API Description

- Descriptions of the parameter uWorkMode are changed in section 2.1 "Hi264DecCreate."

- Value range of the parameter *pUserData is changed in section 2.1 "Hi264DecCreate."

- "Note" part is added in section 2.1 "Hi264DecCreate."

- Descriptions of bit[10] of the parameter uFunctionSet are changed in section 2.3 "Hi264DecGetInfo."

- Parameter *pUserData and descriptions are added in section 2.4 "Hi264DecFrame" and section 2.5 "Hi264DecAU."

- "Description" part is changed in section 2.5 "Hi264DecAU."

Chapter 3 Data Types and Data Structures

Descriptions of bit[10] in the part "Definition" are changed in section 3.2.1 "H264_LIBINFO_S."

Chapter 4 API Application Instances

Descriptions of the "Set the decoder attributes" part are changed in section 4.2 "Program Instances."

Updates in Issue 02 (2008-02-15)

This issue is the first official release, which incorporates the following changes:

Chapter 2 API Description

- Return values are changed in section 2.4 "Hi264DecFrame."

- Descriptions of the data structures are changed in section 2.4 "Hi264DecFrame."

- API function Hi264DecAU is added.

Updates in Issue 01 (2007-11-30)

This issue is used for first office application (FOA).

Contents

Figures

Tables

1 Overview

1.1 Scope

The H.264 PC decoding library software provided by Shenzhen Hisilicon Semiconductor Co., Ltd. is a set of decoding software with high efficiency, reliability, and compatibility. The decoding library implements the H.264 decoding process internally and provides flexible application programming interfaces (APIs) for users to develop application programs quickly.

The decoding library software provides two calling modes in Windows, the dynamic library and the static library. As a result, the application program development becomes easier. Table 1-1 describes the major components of the decoding library.

Table 1-1 Components of the decoding library

Component	Item	Description
API	hi_config.h hi_h264api.h	hi_config.h should be included in the user project first, and then hi_h264api.h is included.
Static library	hi_h264dec_w.lib	When the static library is used, ignore the following six library files in the compiler options: libm.lib, libguide.lib, libirc.lib, libc.lib, libmmt.lib, and svml_disp.lib. Otherwise, an alarm indicating failed link is displayed during compilation.
Dynamic library	hi_h264dec_w.lib hi_h264dec_w.dll	None
Demo	hi_h264sample.c	Taking decoding a stream file as an example, demonstrate how to call the decoding library APIs.

Users can develop application programs based on the decoding library in multiple compiling environments. The decoding library is compatible with Microsoft Windows 2000 or the mainstream Windows operating systems of the later versions. The decoding library is also compatible with most of the PC-oriented CPU chipsets launched by Intel or AMD since 2002. Table 1-2 lists the major development and running environments of the decoding library.

Table 1-2 Running environments of the decoding library

Type	Compatible Configuration	Recommended Configuration	Description
Compiler	Visual C++6.0 Visual Studio.net2003 Intel C++ 9.0 / 10.0	Visual Studio.net 2003	None.
Operating system	Windows 98 Windows 2000 Windows XP Windows 2003 Windows Vista Windows 7 (32bit) Windows 7 (64bit)	Windows XP Windows 7	In Windows 98, the decoding library works in the decay mode, and the decoding performance is low.
Hardware	Intel P3 series Intel P4 series Intel Core series AMD Athlon64 series AMD Sempron series AMD Athlon series	The CPU dominant frequency is above 3.0 GHz, and the memory capacity is above 512 MB.	In Intel P3, AMD AthlonXP, or the earlier CPU environments, the decoding library works in the decay mode, and the decoding performance is low.

1.2 Function List

Function	Description	Page
Hi264DecCreate	Create and initialize the decoder handle.	5
Hi264DecDestroy	Destroy the created handle.	7
Hi264DecGetInfo	Query the version information of the decoding library and the function set of the current version.	8
Hi264DecFrame	Decode a segment of input stream and output pictures in the unit of frame.	10
Hi264DecAU	Decode the streams from a frame of picture and then output the picture immediately.	13
Hi264DecImageEnhance	Enhance the decoded picture.	16

1.3 Function Description Fields

This document describes the API reference information in the following six fields.

Field	Function
Purpose	Describes the major function of an API.
Syntax	Lists the syntax of an API.
Description	Describes the work flow of an API.
Parameter	Lists the parameters of an API and the related information.
Return Value	Lists the return values of an API and the related information.
Note	Describes matters you need to pay attention to when using an API.

1.4 Data Structure Description Fields

Field	Function
Description	Describes functions implemented by a structure.
Definition	Lists the definition of a structure.
Note	Lists structure-related matters you need to pay attention to.

2 API Description

2.1 Hi264DecCreate

[Purpose]

To create and initialize the handle of a decoder.

[Syntax]

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

[Description]

Create the handle of a decoder.

Before decoding, the function performs the following operations:

- Allocate the decoding space
- Initialize the decoder-related variables and states
- Set the input stream type, the output picture format, and decoder attributes such as the maximum width and height and the maximum count of reference frames.

With this API, the upper application can create many decoders in multiple threads for multi-channel decoding.

[Parameter]

Parameter	Member	Value Range	Input/Output	Description
pDecAttr	uPictureFormat	0x00	Input	Output picture format. 0x00 indicates that the output picture format is 4:2:0. The decoding library does not support other formats at present.
	uStreamInType	0x00	Input	Input stream format. 0x00 indicates that the input stream is the H.264 stream with the NALU separator 00 00 01.

Parameter	Member	Value Range	Input/Output	Description
	uPicWidthInMB	[0x06, 0x160]	Input	Maximum picture width, in the unit of MB. When the width value is not within the value range, use the default value 120, namely, the width of the 1080p picture.
	uPicHeightInMB	[0x02, 0x100]	Input	Maximum picture height, in the unit of MB. When the height value is not within the value range, use the default value 68, namely, the height of the 1080p picture.
	uBufNum	[0x01, 0x10]	Input	Assign the buffer count for reference frames in decoder. When the value is not within the value range, use the default value 0x04.
	uWorkMode	None	Input	bit[31:6]: reserved. bit[5]: multi-thread decoding enable. 0: disabled. 1: enabled. bit[4]: enable the internal deinterlace function of the decoding library. 0: disabled. 1: enabled. bit[3:1]: reserved. bit[0]: the decoder working mode. 0: quick output mode (outputting a frame of picture immediately after decoding the frame). 1: the picture output mode defined in the H.264 protocol.
	*pUserData;	None	Input	Point to the input user data. For details about the data type, refer to the definition of H264_USERDATA_S. The decoder does not parse the parameter at present.
	uReserved	0	Input	Reserved.

[Return Value]

Return Value	Macro Definition	Description
0	NULL	Creating the decoder is failed. (The memory allocation is failed or errors occur during the parameter configuration.)
Non-zero	None	Creating the decoder is successful. The return value is the decoder handle.

[Note]

- The quick output mode is available only when the decoding order is the same as the image output order. If the images do not contain B frames, the output delay time can be reduced by using the quick output mode.

- If the internal deinterlace function of a decoder is enabled, this function takes effect only when the input video is encoded in field mode. When frame video images exist, the decoder skips the deinterlace process automatically.

- Multi-thread decoding is applicable only when a single channel is used for decoding and each frame contains multiple slices.

- To start multi-thread decoding, ensure that input streams are not deblocked at the slice boundary. If the input streams do not meet the requirement, the decoder disables the deblocking function at the slice boundary.

- The decoding performance is improved by using multiple threads only when the multi-core CPU decodes the pictures from one channel. This is because thread scheduling occupies the CPU usage. Therefore, you are not advised to perform multi-thread decoding when a single-core CPU is used.

- The preceding working modes are independent. Therefore, the user can set a working mode or multiple working modes.

- You must ensure sufficient system memory when creating a decoder for decoding oversized pictures. Otherwise, the decoder fails to be created.

2.2 Hi264DecDestroy

[Purpose]

To destroy the handle of a decoder.

[Syntax]

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

[Description]

When decoding ends, the API destroys the memory space allocated to the decoder to avoid the memory leakage.

[Parameter]

Parameter	Member	Value Range	Input/Output	Description
hDec	None	None	Input	The decoder handle to be destroyed

[Return Value]

None

[Note]

The destroyed handle should be set to null manually.

2.3 Hi264DecGetInfo

[Purpose]

To query the version information of the decoding library and the function set of the current version.

[Syntax]

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

[Description]

Before creating the decoder, users can call this function to query the version information of the decoding library and the function set of the current version.

[Parameter]

Parameter	Member	Value Range	Input/Output	Description
pLibInfo	uMajor	None	Output	Major serial number of the decoder.
	uMinor	None	Output	Minor serial number of the decoder.
	uRelease	None	Output	Release serial number of the decoder.
	uBuild	None	Output	Build serial number of the decoder.
	sVersion	None	Output	Version information of the decoder.
	sCopyRight	None	Output	Copyright information of the decoder.
	uFunctionSet	None	Output	Decoder function information: bit[31:14]: reserved. bit[13] 0: Multi-thread decoding is not supported. 1: Multi-thread decoding is supported. bit[12] 0: The high profile is not supported.

Parameter	Member	Value Range	Input/Output	Description
				1: The high profile is supported.
				bit[31:11]: reserved.
				bit[10]
				0: The internal deinterlace function is not supported.
				1: The internal deinterlace function is supported.
				bit[9]
				0: The cabac decoding is not supported.
				1: The cabac decoding is supported.
				bit[8]
				0: The weighted prediction decoding is not supported.
				1: The weighted prediction decoding is supported.
				bit[7]
				0: The B-slice decoding is not supported.
				1: The B-slice decoding is supported.
				bit[6]
				0: The MBAFF decoding is not supported.
				1: The MBAFF decoding is supported.
				bit[5]
				0: The PAFF decoding is not supported.
				1: The PAFF decoding is supported.
				bit[4]
				0: The FMO decoding is not supported.
				1: The FMO decoding is supported.
				bit[3]: reserved.
				bit[2]
				0: The Hi351x digital water mark extraction is not supported.
				1: The Hi351x digital water mark extraction is supported.
				bit[1]: reserved.
				bit[0]
				0: The quick picture output mode is supported.
				1: The quick picture output mode is not supported.
	uPictureFormat	0x00	Output	The supported picture format.

Parameter	Member	Value Range	Input/Output	Description
				0x00 indicates that only the 4:2:0 picture format is supported.
	uStreamInType	0x00	Output	The supported stream format. 0x00 indicates that only the H.264 stream with the NALU separator 00 00 01 is supported.
	uPicWidth	0x1600	Output	The supported maximum picture width, in the unit of pixel.
	uPicHeight	0x1000	Output	The supported maximum picture height, in the unit of pixel.
	uBufNum	0x10	Output	The supported maximum count of reference frames.
	uReserved	None	Output	Reserved.

[Return Value]

Return Value	Macro Definition	Description
0	None	Getting the decoding library information is successful.
–1	None	Getting the decoding library information is failed because the input parameter is incorrect.

[Note]

None

2.4 Hi264DecFrame

[Purpose]

To decode a segment of input stream and output pictures in the unit of frame.

[Syntax]

```
HI_S32 Hi264DecFrame
(
HI_HDL hDec
HI_U8 *pStream
HI_U32 iStreamLen
HI_U64 ullPTS
H264_DEC_FRAME_S *pDecFrame
HI_U32 uFlags
```

```
);
```

[Description]

The function supports only the stream decoding. For the continuous, linear H.264 stream with the NALU separator 00 00 01, decoding can starts from any address and the decoding length is customized.

[Parameter]

Parameter	Member	Value Range	Input/Output	Description
hDec	None	None	Input	Decoder handle.
pStream	None	None	Input	Start address of the stream.
iStreamLen	None	None	Input	Stream length (in bytes).
ullPTS	None	None	Input	Presentation time stamp (PTS).
pDecFrame	pY	None	Output	Address of the output Y component.
	pU	None	Output	Address of the output U component.
	pV	None	Output	Address of the output V component.
	uWidth	None	Output	Width of the output picture (in pixels).
	uHeight	None	Output	Height of the output picture (in pixels).
	uYStride	None	Output	Stride of the output Y component (in pixels).
	uUVStride	None	Output	Stride of the output U/V component (in pixels).
	uCropping LeftOffset	None	Output	Left cropping offset of the output picture (in pixels).
	uCropping RightOffs et	None	Output	Right cropping offset of the output picture (in pixels).
	uCropping TopOffset	None	Output	Top cropping offset of the output picture (in pixels).
	uCropping BottomOff set	None	Output	Bottom cropping offset of the output picture (in pixels).
	uDpbIdx	None	Output	ID of the buffer for storing the output picture (not used currently).

Parameter	Member	Value Range	Input/Output	Description
	bError	0 or 1	Output	Error flag of the current picture. 0: No error occures in the output picture. 1: Errors occur in the output picture.
	uPicFlag	0, 1, 2	Output	Attributes of the output picture. 0: Frame. 1: Top field. 2: Bottom field.
	bIntra	0 or 1	Output	IDR frame flag. 0: Non-IDR frame. 1: IDR frame.
	ullPTS	None	Output	PTS of the output picture.
	uPictureID	None	Output	ID of the output picture.
	uReserved	None	Output	Reserved.
	*pUserData	None	Output	Point to the output user data.
	*pFrameInfo	None	Output	Pointer to the output information of one frame
uFlags	None	0 or 1	Input	Decoding mode. 0: Normal decoding. 1: Indicate the end of decoding and flush the remaining pictures.

[Return Value]

Return Value	Macro Definition	Description
0	HI_H264DEC_OK	The function is called successfully, and a frame of picture is output.
–1	HI_H264DEC_NEED_MORE_ BITS	The remaining stream is not enough for a frame, so more stream is needed. The value is probably returned only when uFlags is set to 0.

–2	HI_H264DEC_NO_PICTURE	All the remaining pictures in the decoder are output. The value is returned when uFlags is 1.
–3	HI_H264DEC_ERR_HANDLE	The decoder handle is null or the pointer to the output picture structure is null.

[Note]

Please note the following two items when calling Hi264DecFrame:

- During the decoding process, the user should divide the stream into segments and allocate them to the decoder in order. The user allocates segments of the stream to the decode by calling the function and configures parameters as follows: pStream = NULL; iStreamLen = 0; uFlags = 0. Then call the function repeatedly and configure a new segment of stream till the function returns HI_H264DEC_NEED_MORE_BITS.

  If the function returns HI_H264DEC_OK in the repeated calling process,, it indicates that a remaining picture is output. The user must process and save the picture in pDecFrame in time during the repeated calling process.

- To output the remaining pictures in the decoder when the decoding process ends, the user can configure parameters as follows: uFlags = 1, pStream = NULL. Then call the function repeatedly, and stop the decoder till the function returns HI_H264DEC_NO_PICTURE.

  If the function returns HI_H264DEC_OK in the repeated calling process,, it indicates that a remaining picture is output. The user must process and save the picture in pDecFrame in time during the repeated calling process.

This function can transmit the time stamp bypass, this is, the time stamp input with the stream will be output with the decoded frame together. For details, see section 3.2.5 "H264_DEC_FRAME_S."

2.5 Hi264DecAU

[Purpose]

To decode the stream containing one frame only and output this frame immediately.

[Syntax]

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

[Description]

The Hi264DecAU only supports the input stream that has been split into frames before decoding. The distributed stream with a frame only should be in the standard H.264 format with "00 00 01" nalu delimiter.

[Parameter]

Parameter	Member	Value Range	Input/Output	Description
hDec	None	None	Input	Decoder handle.
pStream	None	None	Input	Start address of the stream.
iStreamLen	None	None	Input	Stream length (in bytes).
ullPTS	None	None	Input	PTS.
pDecFrame	pY	None	Output	Address of the output Y component.
	pU	None	Output	Address of the output U component.
	pV	None	Output	Address of the output V component.
	uWidth	None	Output	Width of the output picture (in pixels).
	uHeight	None	Output	Height of the output picture (in pixels).
	uYStride	None	Output	Stride of the output Y component (in pixels).
	uUVStride	None	Output	Stride of the output U/V component (in pixels).
	uCropping LeftOffset	None	Output	Left cropping offset of the output picture (in pixels).
	uCropping RightOffset	None	Output	Right cropping offset of the output picture (in pixels).
	uCropping TopOffset	None	Output	Top cropping offset of the output picture (in pixels).
	uCropping BottomOffset	None	Output	Bottom cropping offset of the output picture (in pixels).
	uDpbIdx	None	Output	ID of the buffer for storing the output picture (not used currently).

Parameter	Member	Value Range	Input/Output	Description
	bError	0 or 1	Output	Error flag of the current picture. 0: No error occurs in the output picture. 1: Errors occur in the output picture.
	uPicFlag	0, 1, 2	Output	Attributes of the output picture. 0: Frame. 1: Top field. 2: Bottom field.
	bIntra	0 or 1	Output	IDR frame flag. 0: Non-IDR frame. 1: IDR frame.
	ullPTS	None	Output	PTS of the output picture.
	uPictureID	None	Output	ID of the output picture.
	uReserved	None	Output	Reserved.
	*pUserData	None	Output	Point to the output user data.
	*pFrameInfo	None	Output	Pointer to the output information of one frame
uFlags	None	None	None	Reserved.

[Return Value]

Return Value	Macro Definition	Description
0	HI_H264DEC_OK	The function is called successfully, and a frame is output.
–2	HI_H264DEC_NO_PICTURE	No invalid picture is output.
–3	HI_H264DEC_ERR_HANDLE	The decoder handle is null or the pointer to the output picture structure is null.

[Note]

Hi264DecAU is similar with Hi264DecFrame, the user can call one of them as required.

- The user should split the stream in the unit of frame during the decoding process. Each time the stream distributed to Hi264DecAU should contain one frame only. Otherwise, an output exception may arise.

During the preceding cyclic calling process, if the function returns HI_H264DEC_OK, it indicates a frame is output; if the function returns HI_H264DEC_NO_PICTURE, it indicates the configured stream at this time cannot generate any frame.

- This function supposes the stream input each time contains one frame only, and this frame is output regardless of whether it is completed.

This function can transmit the time stamp bypass, this is, the time stamp input with the stream will be output with the decoded frame together. For details, see section 3.2.5 "H264_DEC_FRAME_S."

2.6 Hi264DecImageEnhance

[Purpose]

To enhance the decoded pictures.

[Syntax]

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

[Description]

After a picture is decoded, you can process the picture to improve the picture quality in some applications.

[Parameter]

Parameter	Member	Value Range	Input/Output	Description
hDec	None	None	Input	Handle of a decoder.
pDecFrame	pY	None	Input/Output	Address of the output Y component.
	pU	None	Input/Output	Address of the output U component.
	pV	None	Input/Output	Address of the output V component.
	uWidth	None	Input/Output	Width of the output picture (in pixels).
	uHeight	None	Input/Output	Height of the output picture (in pixels).
	uYStride	None	Input/Output	Stride of the output Y component (in pixels).

Parameter	Member	Value Range	Input/Output	Description
	uUVStride	None	Input/Output	Stride of the output U/V component (in pixels).
	uCroppingLeftOffset	None	Input/Output	Left cropping offset of the output picture (in pixels).
	uCroppingRightOffset	None	Input/Output	Right cropping offset of the output picture (in pixels).
	uCroppingTopOffset	None	Input/Output	Top cropping offset of the output picture (in pixels).
	uCroppingBottomOffset	None	Input/Output	Bottom cropping offset of the output picture (in pixels).
	uDpbIdx	None	Input/Output	ID of the buffer for storing the output picture (not used currently).
	bError	0 or 1	Input/Output	Error flag of the current picture. 0: No error occurs in the output picture. 1: Errors occur in the output picture.
	uPicFlag	0, 1, 2	Input/Output	Attributes of the output picture. 0: Frame. 1: Top field. 2: Bottom field.
	bIntra	0 or 1	Input/Output	Flag indicating whether the output picture is an IDR frame. 0: Non-IDR frame. 1: IDR frame.
	ullPTS	None	Input/Output	PTS of the output picture.
	uPictureID	None	Input/Output	ID of the output picture.
	uReserved	None	Input/Output	Reserved.
	*pUserData	None	Input/Output	Point to the output user data.
	*pFrameInfo	None	Output	Pointer to the output information of one frame
uEnhanceCoeff	None	(0, 128]	Input	Picture enhance coefficient. Generally, the value range is [30, 50]. The greater the value is, the more the picture is changed. The recommended value is 40.

[Return Value]

Return Value	Macro Definition	Description
0	HI_H264DEC_OK	The function is called successfully and a frame of picture is enhanced and can be output.
–3	HI_H264DEC_ERR_HANDLE	The handle of the decoding library is empty or the input parameters are incorrect.

[Note]

The data structure pointer pDecFrame can function as an input or output parameter. After obtaining a frame of picture, the user should use the output parameter pDecFrame of Hi264DecFrame or Hi264DecAU as the input parameter of Hi264DecImageEnhance without any modification.

3 Data Types and Data Structures

3.1 Common Data Types

The major data structures used in the APIs of the win32 environment are defined as follows:

```
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 Data Structures

3.2.1 H264_LIBINFO_S

[Description]

The version, copy right, and function set information of the decoder.

[Definition]

```
/*Data structure of the version, copy right, and function set information of
the decoder*/
typedef struct hiH264_LIBINFO_S
{
HI_U32  uMajor;             /*Major serial number*/
HI_U32  uMinor;             /*Minor serial number*/
```

```
HI_U32 uRelease;               /*Release serial number*/
HI_U32 uBuild;                 /*Build serial number*/
const HI_CHAR* sVersion;       /*Version information*/
const HI_CHAR* sCopyRight;     /*Copy right information*/
HI_U32 uFunctionSet;           /*Decoder function set*/
/*If each of the bits (except bit0) is 1, the current version supports the
function; if each of the bits (except bit0) is 0, the current version does
not support the function.*/
/* bit0 : Quick output mode      */
/* bit1 :  Reserved bits           */
/* bit2 : Hi351x digital watermark    */
/* bit3 : Reserved bits          */
/* bit4 : FMO decoding        */
/* bit5 : PAFF decoding        */
/* bit6 : MBAFF decoding       */
/* bit7 : B segment decoding         */
/* bit8 : Weighted prediction        */
/* bit9 : CABAC calculation decoding     */
/* bit10: Internal integrated deinterlace */
/* bit11-bit 31 : Reserved bits*/
HI_U32 uPictureFormat;/* Supported picture output format*/
                      /* 0x00: supports only the YUV420 format at present*/
HI_U32 uStreamInType; /* Input stream format*/
                      /*0x00: Only the H.264 stream with the NALU separator
00 00 01 is supported.*/
                      /*H.264 stream*/
HI_U32  uPicWidth;     /*The maximum picture width (in the unit of pixel)*/
HI_U32  uPicHeight;    /*The maximum picture height (in the unit of pixel)*/
HI_U32  uBufNum;       /*The maximum count of the reference frames*/
HI_U32  uReserved;     /*Reserved */
} H264_LIBINFO_S;
```

[Note]

None

3.2.2 H264_USERDATA_S

[Description]

The user data information.

[Definition]

```
/*User data structure*/
typedef struct hiH264_USERDATA_S
{
HI_U32  uUserDataType;              /*User data type     */
```

```
HI_U32 uUserDataSize;               /*User data length      */
HI_UCHAR *pData;                    /*User data buffer area    */
struct hiH264_USERDATA_S *pNext;  /*Point to the next segment of user data*/
} H264_USERDATA_S;
```

[Note]

None

3.2.3 H264_DEC_ATTR_S

[Description]

The decoder attribute information.

[Definition]

```
/*Data structure of the decoder attribute information.*/
typedef struct hiH264_DEC_ATTR_S
{
HI_U32  uPictureFormat;      /*Output picture format*/
                             /* 0x00: The decoder only supports only the YUV420
format at present*/
HI_U32  uStreamInType;       /*Input stream format*/
                             /* 0x00: The decoder supports only the H.264*/
                             /*stream with the NALU separator 00 00 01*/
HI_U32  uPicWidthInMB;       /*Picture width (in the unit of macro block)*/
HI_U32  uPicHeightInMB;      /*Picture height (in the unit of macro block)*/
HI_U32  uBufNum;             /*Count of the reference frames*/
HI_U32  uWorkMode;           /*Decoder working mode*/
                             /* bit 0: 0: quick output mode; 1: Normal output
                              mode*/
                             /* bit 1 to bit 2: */
                             /* 00: Decode only pictures*/
                             /* 01: Reserved*/
                             /* 10: Decode the Hi351x digital watermark*/
                             /* 11: Reserved */
                                /* bit 4: 0: The field picture is not
deinterlaced.*/
                                /*        1: The field picture is
deinterlaced.*/
                                /* bit 5: 0: Single-thread decoding is
performed.*/
                                /*        1: Multi-thread decoding is
performed on multiple slices.*/
                                /* bit 6 to bit 31: reserved*/
H264_USERDATA_S *pUserData;   /*User data*/
HI_U32 uReserved;             /*Reserved */
```

```
} H264_DEC_ATTR_S;
```

[Note]

None

3.2.4 H264_OUTPUT_INFO_S

[Description]

The decoder output information for each frame.

[Definition]

```
/* Data structure of the decoder output information for each frame */
typedef struct hiH264_OUTPUT_INFO_S
{
    HI_U32 uPicBytes;        /*total bytes of one frame*/
    HI_U32 uI4MbNum;         /*number of I4x4 macroblocks in one frame*/
    HI_U32 uI8MbNum;         /*number of I8x8 macroblocks in one frame*/
    HI_U32 uI16MbNum;        /*number of I16x16 macroblocks in one frame*/
    HI_U32 uP16MbNum;        /*number of P16x16 macroblocks in one frame*/
    HI_U32 uP16x8MbNum;      /*number of P16x8 macroblocks in one frame*/
    HI_U32 uP8x16MbNum;      /*number of P8x16 macroblocks in one frame*/
    HI_U32 uP8MbNum;         /*number of P8x8 macroblocks in one frame*/
    HI_U32 uPskipMbNum;      /*number of PSkip macroblocks in one frame*/
    HI_U32 uIpcmMbNum;       /*number of IPCM macroblocks in one frame*/
} H264_OUTPUT_INFO_S;
```

[Note]

None

3.2.5 H264_DEC_FRAME_S

[Description]

The decoder output picture information.

[Definition]

```
/*Data structure of the decoder output picture information*/
typedef struct hiH264_DEC_FRAME_S
{
HI_U8  *pY;       /*Address of the Y component*/
HI_U8  *pU;       /*Address of the U component*/
HI_U8  *pV;       /*Address of the V component*/
HI_U32  uWidth;   /*The maximum picture width (in the unit of pixel)*/
HI_U32  uHeight;  /*The maximum picture height (in the unit of pixel)*/
HI_U32  uYStride; /*Stride of the output Y component, (in the unit of pixel)*/
HI_U32  uUVStride; /*Stride of the output U/V component, (in the unit of
                    pixel)*/
```

```
HI_U32   uCroppingLeftOffset;    /*Picture cropping information: the cropped
                                   pixels of the left margin*/
HI_U32   uCroppingRightOffset;   /*Picture cropping information: the cropped
                                   pixels of the right margin*/
HI_U32   uCroppingTopOffset;     /*Picture cropping information: the cropped
                                   pixels of the top margin*/
HI_U32   uCroppingBottomOffset;  /*Picture cropping information: the cropped
                                   pixels of the bottom margin*/
HI_U32   uDpbIdx;        /*The dpb serial number of the output picture*/
HI_U32   tPicFlag;       /*Types of pictures: 0: frame; 1: top fields; 2: bottom
                          fields*/
HI_U32   bError;         /*Picture correctness: 0: correct; 1: incorrect*/
HI_U32   bIntra;         /*Frame type of the picture; 0: non-IDR frame; 1:
                          Instantaneous decoding refresh (IDR) frame*/
HI_U64   ullPTS;         /*Time stamp*/
HI_U32   uPictureID;     /*Picture serial number*/
HI_U32   uReserved;      /*Reserved */
H264_USERDATA_S *pUserData;     /*Point to the user data*/
H264_OUTPUT_INFO_S *pFrameInfo; /*Pointer to the output information of one
frame*/


} H264_DEC_FRAME_S;
```
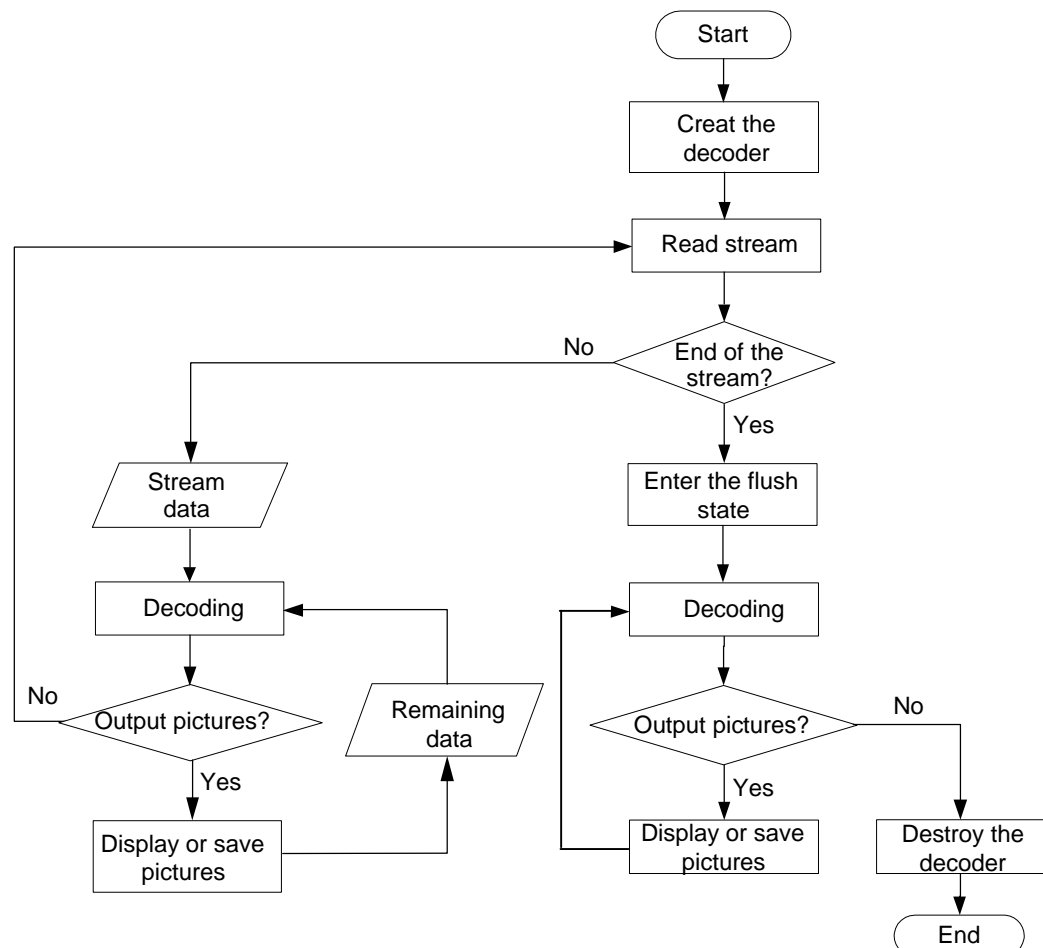
[Note]

None

4 API Application Instances

4.1 Flow for Steam Decoding Mode

Figure 4-1 Working flow of the decoding library APIs

4.2 Program Instances

```
H264_DEC_ATTR_S  dec_attrbute;
H264_DEC_FRAME_S dec_frame;
HI_HDL handle = NULL;
HI_S32 end    = 0;
HI_U8  buf[0x8000];       /*Stream buffer area*/
FILE *h264 = NULL;        /*H.264 stream file*/
FILE *yuv = NULL;         /*File keeping the YUV picture*/
HI_U32 ImageEnhanceEnable = 0;    /* 0: Disable the picture enhance function;
1: Enable the picture enhance function */
HI_U32 StrenthCoeff = 40;         /* Picture enhance coefficient */


/*Open the H.264 stream file and the file keeping the YUV picture*/
h264 = fopen(argv[1], "rb");
yuv  = fopen(argv[2], "wb");
if(NULL == h264 || NULL == yuv)
{
    goto exit;
}


/*Set the decoder attributes*/
dec_attrbute.uBufNum      = 16;       /*16 reference frames*/
dec_attrbute.uPicHeight   = 68;   /*Size of the 1080p picture*/
dec_attrbute.uPicWidth    = 120;
dec_attrbute.pUserData    = NULL;     /*No user data*/
dec_attrbute.uStreamInType = 0;        /*Input the stream starting with 00
00 00 01 ...*/
dec_attrbute.uWorkMode    = 0x31; /*The normal output mode is selected.
                         The internal deinterlace function is enabled.
                               Multi-thread decoding is performed.*/


/*Create a decoder*/
handle = Hi264DecCreate(&dec_attrbute);
if(NULL ==  handle)
{
    goto exit;
}


/*Start the decoding process*/
while (!end)
{
    HI_U32  len = fread(buf,1,sizeof(buf),h264); /*Read a segment of stream*/
    HI_U32  flags = (len>0)?0:1;                 /*Flag to indicate the end
```

```
of the stream*/
    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;                      /*All the pictures in the decoder are
output and the decoding process ends*/
            break;
        }
        if(HI_H264DEC_OK == result) /*Output a frame of picture*/
        {
            /* Enhance a picture */
            if(ImageEnhanceEnable)
            {
                Hi264DecImageEnhance(handle, &dec_frame, StrenthCoeff);
            }
            /*Save a picture*/
            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);
    }
}

/*Destroy the decoder and release the handle*/
Hi264DecDestroy(handle);
Handle = NULL;

Exit:
if(NULL != h264)
fclose(h264);
if(NULL != yuv)
```

```
fclose(yuv);
```