H.265 PC Decoding Library Software

API Reference

Issue 09

Date 2018-05-30

Copyright © HiSilicon (Shanghai) Technologies Co., Ltd. 2019. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of HiSilicon Technologies Co., Ltd.

Trademarks and Permissions

 , HISILICON, and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

HiSilicon (Shanghai) Technologies Co., Ltd.

Address: New R&D Center, 49 Wuhe Road, Bantian,
Longgang District,
Shenzhen 518129 P. R. 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 change history.

NOTE

- This document uses the Hi3516A as an example. Unless otherwise specified, the descriptions of the Hi3516A also apply to Hi3516D, and those of the Hi3516A V200 also applies to Hi3519 V101.
- Unless otherwise stated, Hi3519A V100, Hi3559C V100, and Hi3559A V100 contents are consistent.

Related Versions

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

Product Name	Version
Hi3516A	V100
Hi3516D	V100
Hi3536	V100
Hi3519	V100
Hi3519	V101
Hi3559	V100
Hi3516C	V300
Hi3516E	V100
Hi3556	V100
Hi3531D	V100
Hi3521D	V100
Hi3536C	V100
Hi3516A	V200
Hi3536D	V100

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

Product Name	Version
Hi3520D	V400
Hi3559A	V100
Hi3559C	V100
Hi3519A	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 an imminently hazardous situation which, if not avoided, will result in death or serious injury.
WARNING	Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.
CAUTION	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury.
NOTICE	Indicates a potentially hazardous situation which, if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results. NOTICE is used to address practices not related to personal injury.
NOTE	Calls attention to important information, best practices and tips. NOTE is used to address information not related to personal injury, equipment damage, and environment deterioration.

Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

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.

Issue 09 (2018-05-30)

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

Section 2.1 is modified.

Issue 08 (2018-04-04)

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

Sections 2.1, 2.4, 2.5, 3.2.6, 3.2.8, and 4.3 are modified.

The descriptions of the Hi3521DV100 and Hi3531DV100 are added.

Issue 07 (2017-08-15)

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

The description of the Hi3536DV100 is added.

Issue 06 (2017-04-10)

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

The description of the Hi3536CV100 is added.

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

Issue 05 (2017-02-25)

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

The description of the Hi3556 V100 is added.

Issue 04 (2017-02-14)

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

Chapter 2 API Description

In this chapter, the descriptions of some APIs are modified, and the return values of the error types are updated.

Issue 03 (2015-12-01)

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

Chapter 2 API Description

A **Note** symbol is added.

Issue 02 (2015-02-10)

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

Chapter 2 API Description

In section 2.1, the value range of **pstInitParam** is modified.

Issue 01 (2014-12-20)

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

The contents related to the Hi3516D are added.

Issue 00B01 (2014-08-11)

This issue is the first draft release.

Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

Contents

Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

Figures

Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

Tables

Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

1 Overview

1.1 Scope

The H.265 PC decoding library provided by HiSilicon is the decoding software of high performance, high reliability, and superior compatibility. The decoding library completes the main processes for H.265 decoding and provides flexible and simple APIs for developers. In this case, developers can develop application programs quickly.

The decoding library provides the dynamic library and static library on Windows OS for developers to facilitate application program development. Table 1-1 describes main components of the decoding library.

Table 1-1 Main components of the decoding library

Component	Name	Description
API	IHWVideo_Typedef.h IHW265Dec_Api.h	In a project, IHWVideo_Typedef.h must be included before IHW265Dec_Api.h in the code for calling the APIs.
32-bit static library	hi_h265_dec_w32.lib	-
32-bit dynamic library	hi_h265_dec_w32.lib hi_h265_dec_w32.dll	-
64-bit static library	hi_h265_dec_w64.lib	-
64-bit dynamic library	hi_h265_dec_w64.lib hi_h265_dec_w64.dll	-
Sample code	hi_h265sample.c	The read file decoding is used as an example to illustrate how to call APIs of the decoding library.

Developers can develop application programs based on the decoding library in multiple environments. The decoding library is compatible with mainstream OSs provided by Microsoft, such as Windows XP, Windows 7, and Windows 8 and PC-oriented CPU chipset

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

provided by Intel and AMD. Table 1-2 describes the running environments of the decoding library.

Table 1-2 Running environments of the decoding library

Type	Compatible Configuration	Recommended Configuration	Description
Compiler	Visual Studio 2008 Visual Studio 2010 Visual Studio 2012	Visual Studio 2010	None
OS	Windows XP Windows 7 (32-bit) Windows 7 (64-bit) Windows 8 (32-bit) Windows 8 (64-bit)	Windows 7	None
Hardware	Intel Core series CPU Intel Pentium series CPU AMD Ax series CPU	Intel Core i5 series CPU	The decoding library is optimized based on the multimedia command sets, such as SSE, SSE2, SSE3, SSE4, and AVX. On PCs that these command sets are not supported, the decoding library performance is low.

1.2 API Format

None

1.3 Function List

Function	Function	Page
IHW265D_Create	Create and initialize a decoder handle.	4
IHW265D_Delete	Destroy the decoder handle.	7
IHW265D_GetVersion	Query the decoding library version information.	7
IHW265D_DecodeFrame	Decode an input code stream and output the image.	8
IHW265D_DecodeAU	Decode an input code stream and output the frame image immediately.	12

Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

1.4 Function Description Fields

This section describes API reference information from six fields.

Field	Function
Purpose	Describes the major function of an API.
Syntax	Lists the syntax of an API.
Description	Describes the working process of an API.
Parameter	Describes the parameters and attributes of an API.
Return Value	Describes the return values of an API.
Note	Describes the notes for calling an API.

1.5 Structure Description Fields

Field	Function
Description	Describes the function of a data structure.
Definition	Describes the definition of a data structure.
Precautions	Lists the precautions of a data structure.

Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

2 API Description

NOTE

All the APIs provided in the HW265 decoding library are called in _cdecl mode. When the compilation option used by the user is set to any other mode, stack use conflict may occur.

2.1 IHW265D_Create

[Purpose]

Create and initialize a decoder handle.

[Syntax]

```
INT32 IHW265D_Create(IH265DEC_HANDLE *phDecoder, IHW265D_INIT_PARAM
*pstInitParam);
```

[Description]

This API is used to create a decoder handle. During decoding, this API is called and performs the following operations:

Allocate the decoding space and initialize decoder-related variables and status.

Set decoder attributes, such as the maximum image width and height supported by the decoder, the maximum number of reference frames supported by the decoder, decoder thread type, and output image sequence.

Allocate the memory callback function and release the memory callback function and log callback function.

Upper-layer applications can create multiple decoders to ensure multi-channel decoding.

[Parameter]

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

Parameter	Member	Value Range	Input/ Output	Description
phDecoder	-	-	Input and Output	Handle of a decoder. During handle input, you need to ensure the handle correctness. The system will be suspended when the handle is abnormal.
pstInitParam	uiChannelID	-	Input	Channel ID. The interface does not check whether the channel ID exists. You need to ensure that IDs of two channels successfully created are different.
	iMaxWidth	[8,16384]	Input	Image width (in pixel)
	iMaxHeight	[8,16384]	Input	Image height (in pixel)
	iMaxRefNum	[0,15]	Input	Maximum number of reference frames
	uiBitDepth	8, 10	Input	8: Supports only 8-bit decoding. 10: Supports 10-bit and 8-bit decoding.
	uiMvcEnable	0, 1	Input	0: Does not supports MVC decoding 1: Supports MVC decoding
	iMaxVPSNum	[1,16]	Input	Maximum number of video parameter sets (VPSs) of the streams. The value ranges from **1** to **16**.
	iMaxSPSNum	[1,16]	Input	Maximum number of sequence parameter sets (SPSs) of the streams. The value ranges from **1** to **16**.
	iMaxPPSNum	[1,64]	Input	Maximum number of picture parameter set (PPSs) of the streams. The value ranges from **1** to **64**.
	eThreadType	[0,1]	Input	Thread type Single-thread and multi-thread are supported.
	eOutputOrder	[0,1]	Input	Output sequence Decoding sequence and display sequence are supported. When an invalid value, such as –1 or 2 is entered, a warning is displayed. However, the data can be decoded and output in display sequence.

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

Parameter	Member	Value Range	Input/ Output	Description
	pstUserData	-	Input	Input user data For details about the data type, see HW265D_USERDATA.
	MallocFxn	-	Input	Allocation of the memory callback function
	FreeFxn	-	Input	Release of the memory callback function
	LogFxn	-	Input	Log output callback function

[Return Value]

Return Value	Macro Definition	Description
0	IHW265D_OK	The function is successfully called.
0xF0401000	IHW265D_INVALID_ARGUMENT	The input parameter is incorrect.
0xF0402000	IHW265D_MALLOC_FAIL	The memory fails to be allocated.
0xF0402001	IHW265D_INVALID_MAX_WIDTH	The maximum width of the picture exceeds the limit.
0xF0402002	IHW265D_INVALID_MAX_HEIGHT	The maximum height of the picture exceeds the limit.
0xF0402003	IHW265D_INVALID_MAX_REF_PIC	The maximum number of reference frames exceeds the limit.
0xF0402004	IHW265D_INVALID_THREAD_CONTROL	The thread control input exceeds the limit.
0xF0402005	IHW265D_INVALID_MALLOC_FXN	The Malloc callback function pointer is invalid.
0xF0402006	IHW265D_INVALID_FREE_FXN	The Free callback function pointer is invalid.
0xF0402007	IHW265D_INVALID_LOG_FXN	The Log callback function pointer is invalid.
0xE0404008	IHW265D_THREAD_ERROR	An error occurs during multi-thread creating.

[Note]

- The decoding sequence output mode can be used only if the decoding sequence is consistent with the image output sequence. In normal cases, if an image does not contain the B frame, the decoding sequence output mode can be used to shorten the output delay.

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

- Multi-thread decoding applies to scenarios that have high requirements for single channel decoding performance. The scenarios include high-resolution and high frame rate and insufficient performance for a single CPU core. Internal overheads are generated during multi-thread scheduling. Therefore, multi-thread decoding is not recommended in multi-channel scenarios.
- You must ensure sufficient system memory when creating a decoder for decoding oversized pictures. Otherwise, the decoder fails to be created.

2.2 IHW265D_Delete

[Purpose]

Destroy the decoder handle.

[Syntax]

```
INT32 IHW265D_Delete( IH265DEC_HANDLE hDecoder);
```

[Description]

After decoding is complete, this API is used to destroy the memory space allocated for the decoder to prevent memory leakage.

[Parameter]

Parameter	Member	Value Range	Input/Output	Description
hDecoder	-	-	Input	Decoder handle to be destroyed

[Return Value]

Return Value	Macro Definition	Description
0	IHW265D_OK	The function is successfully called.
0xF0401000	IHW265D_INVALID_ARGUMENT	The parameter is invalid.
0xF0401001	IHW265D_DECODER_NOT_CREATE	The decoder is not created.

[Note]

After the handle is destroyed, set **hDecoder** to **0** manually.

2.3 IHW265D_GetVersion

[Purpose]

Query the decoding library version information.

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

[Syntax]

```
INT32 IHW265D_GetVersion(IHWVIDEO_ALG_VERSION_STRU *pstVersion);
```

[Description]

This API is used to query the decoding library version information.

[Parameter]

Parameter	Member	Value Range	Input/Output	Description
pstVersion	cVersionChar	-	Output	Decoding library version number
	cReleaseTime	-	Output	Compiling time
	uiCompileVersion	-	Output	Compiler version number

[Return Value]

Return Value	Macro Definition	Description
0	IHW265D_OK	The function is successfully called.
0xF0401000	IHW265D_INVALID_ARGUMENT	The parameter is invalid.

[Note]

None

2.4 IHW265D_DecodeFrame

[Purpose]

Decode an input code stream and output the image.

[Syntax]

```
INT32 IHW265D_DecodeFrame(IH265DEC_HANDLE hDecoder, IH265DEC_INARGS
*pstInArgs, IH265DEC_OUTARGS *pstOutArgs);
```

[Description]

This API supports stream decoding. For continuous and linear H.265 code streams using 000001 as the separator of nalu, users can configure any length for decoding.

[Parameter]

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

Parameter	Member	Value Range	Input/ Output	Description
hDecoder	-	-	Input	Handle of a decoder
pstInArgs	pStream	-	Input	Start address of a code stream
	uiStreamLen	-	Input	Code stream length (in the unit of byte). The member indicates the length of a NalUnit, which needs to be used with pStream, and be smaller than or equal to the memory allocated by pStream. When the value of this member is greater than the memory size allocated by pStream, the system may be suspended. You need to ensure that this requirement is met.
	uiTimeStamp	-	Input	Time stamp
	eDecodeMode	-	Input	Decoding mode In normal cases, IH265D_DECODE is used. When decoding the file end, IH265D_DECODE_END is used.
pstOutArgs	uiChannelID	-	Output	Channel ID
	uiBytsConsumed	-	Output	Number of consumed bytes
	uiTimeStamp	-	Output	Time stamp
	eFrameType	[0, 2]	Output	Frame type
	eDecodeStatus	-	Output	Status of the decoder
	uiDecWidth	-	Output	Image width
	uiDecHeight	-	Output	Image height
	uiYStride	-	Output	Luminance component stride
	uiUVStride	-	Output	Chrominance component stride
	uiBitDepthY	-	Output	Bit width of the output luminance component
	uiBitDepthC	-	Output	Bit width of the output chrominance component
	iDisplayPrimariesX0	-	Output	Horizontal coordinate of the green point of the output chrominance

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

Parameter	Member	Value Range	Input/ Output	Description
	iDisplayPrimariesX1	-	Output	Horizontal coordinate of the blue point of the output chrominance
	iDisplayPrimariesX2	-	Output	Horizontal coordinate of the red point of the output chrominance
	iDisplayPrimariesY0	-	Output	Vertical coordinate of the green point of the output chrominance
	iDisplayPrimariesY1	-	Output	Vertical coordinate of the blue point of the output chrominance
	iDisplayPrimariesY2	-	Output	Vertical coordinate of the red point of the output chrominance
	iWhitePointX	-	Output	Horizontal coordinate of the white point of the output luminance
	iWhitePointY	-	Output	Vertical coordinate of the white point of the output luminance
	iMaxDisplayMasteringLuma	-	Output	Maximum output display luminance of HDR 10
	iMinDisplayMasteringLuma	-	Output	Minimum output display luminance of HDR 10
	iMaxContentLightLevel	-	Output	Maximum luminance level of the output content
	iMaxPicAverageLightLevel	-	Output	Average luminance level of the output image
	pucOutYUV	-	Output	Address of the Y, U, and V components
	pstUserData	-	Output	User data
	uiCodingBytesOfCurFrm	-	Output	Source code stream length (Number of bytes)
	uiAspectRatioIdc	-	Output	Aspect ratio
	uiSarWidth	-	Output	Aspect ratio width It is valid only when **uiAspectRatioIdc** is set to **255**.
	uiSarHeight	-	Output	Aspect ratio height It is valid only when **uiAspectRatioIdc** is set to **255**.
	uiVpsNumUnitsInTick	-	Output	Frame rate information

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

Parameter	Member	Value Range	Input/ Output	Description
	uiVpsTimeScale	-	Output	
	stCuOutInfo	-	Output	Current output frame information, including number of various types of CUs in a frame
	bIsError	[0,1]	Output	Frame error flag
	uiLayerIdx	[0,1]	Output	Layer information flag
	uiPoc	-	Output	Picture order count (POC) value of the frame information

[Return Value]

Return Value	Macro Definition	Description
0	IHW265D_OK	The function is successfully called. A frame image is output.
1	IHW265D_NEED_MORE_BITS	The remained code stream is insufficient for decoding data of a frame. More code streams need to be configured. This value is returned only if the decoding mode is IH265D_DECODE.
2	IHW265D_FIND_NEW_PIC	This value is returned when the start information of a new frame is detected.
0xF0401000	IHW265D_INVALID_ARGUMENT	The parameter is invalid.
0xF0401001	IHW265D_DECODER_NOT_CREATE	The decoder is not created.
0xF0402008	IHW265D_STREAMBUF_NULL	The stream address is empty.
0xF0404001	IHW265D_NAL_HEADER_ERR	The NAL header is incorrectly parsed.
0xF0404002	IHW265D_VPS_ERR	The VPS is incorrectly parsed.
0xF0404003	IHW265D_SPS_ERR	The SPS is incorrectly parsed.
0xF0404004	IHW265D_PPS_ERR	The PPS is incorrectly parsed.
0xF0404005	IHW265D_SLICEHEADER_ERR	The slice header is incorrectly parsed.
0xF0404006	IHW265D_SLICEDATA_ERR	The slice data is incorrectly parsed.

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

Return Value	Macro Definition	Description
-1	-	Error occurs when multiple streams parse the code streams.

[Note]

Note the following when calling this function:

- This function can be used to decode a code stream of any length. The input parameters include the code stream buffer address **pStream** and the number of bytes in a code stream **uiStreamLen**. In normal cases, the decoding mode is set to IH265D_DECODE. If IH265D_GETDISPLAY is returned, a frame image is output. Users must process images stored in pstOutArgs in time.
- If a code stream is insufficient for decoding a frame, more code streams need to be input for decoding.
- If a code stream includes multiple frames, this function must be called repeatedly to decode remained code streams after a frame image is decoded.
- After the code streams are decoded, the decoding mode is set to IH265D_DECODE_END to enter the flush mode to clear remain images in the decoder. This function needs to be called repeatedly until IH265D_NO_PICTURE is returned, which indicates that no remaining image exists in the decoder.
- This function provides the function of transparent transmission of the time stamp. The input time stamp is stored in the image structure after decoding and output with the decoded image.

2.5 IHW265D_DecodeAU

[Purpose]

Decode an input code stream and output the frame image immediately.

[Syntax]

```
INT32 IHW265D_DecodeAU(IH265DEC_HANDLE hDecoder, IH265DEC_INARGS *pstInArgs,
IH265DEC_OUTARGS *pstOutArgs);
```

[Description]

This API supports decoding by frame. The input stream must contain only one frame of H.265 code streams and output immediately based on the decoding sequence.

[Parameter]

Parameter	Member	Value Range	Input/ Output	Description
hDecoder	-	-	Input	Handle of a decoder
pstInArgs	pStream	-	Input	Start address of a stream

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

Parameter	Member	Value Range	Input/ Output	Description
	uiStreamLen	-	Input	Stream length (in the unit of byte). The member indicates the length of the stream of one frame, which needs to be used with pStream, and be smaller than or equal to the memory allocated by pStream. When the value of this member is greater than the memory size allocated by pStream, the system may be suspended. You need to ensure that this requirement is met.
	uiTimeStamp	-	Input	Time stamp
	eDecodeMode	-	Input	Decoding mode (Invalid)
pstOutArgs	uiChannelID	-	Output	Channel ID
	uiBytsConsumed		Output	Number of consumed bytes
	uiTimeStamp		Output	Time stamp
	eFrameType		Output	Frame type
	eDecodeStatus		Output	Status of the decoder
	uiDecWidth		Output	Image width
	uiDecHeight		Output	Image height
	uiYStride		Output	Luminance component stride
	uiUVStride		Output	Chrominance component stride
	uiBitDepthY	-	Output	Bit width of the output luminance component
	uiBitDepthC	-	Output	Bit width of the output chrominance component
	iDisplayPrimariesX0	-	Output	Horizontal coordinate of the green point of the output chrominance
	iDisplayPrimariesX1	-	Output	Horizontal coordinate of the blue point of the output chrominance
	iDisplayPrimariesX2	-	Output	Horizontal coordinate of the red point of the output chrominance
	iDisplayPrimariesY0	-	Output	Vertical coordinate of the green point of the output chrominance

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

Parameter	Member	Value Range	Input/ Output	Description
	iDisplayPrimariesY1	-	Output	Vertical coordinate of the blue point of the output chrominance
	iDisplayPrimariesY2	-	Output	Vertical coordinate of the red point of the output chrominance
	iWhitePointX	-	Output	Horizontal coordinate of the white point of the output luminance
	iWhitePointY	-	Output	Vertical coordinate of the white point of the output luminance
	iMaxDisplayMasteringLuma	-	Output	Maximum output display luminance of HDR 10
	iMinDisplayMasteringLuma	-	Output	Minimum output display luminance of HDR 10
	iMaxContentLightLevel	-	Output	Maximum luminance level of the output content
	iMaxPicAverageLightLevel	-	Output	Average luminance level of the output image
	pucOutYUV		Output	Output of the Y, U, V component addresses
	pstUserData		Output	User data
	uiCodingBytesOfCurFrm	-	Output	Source code stream length (Number of bytes)
	uiAspectRatioIdc	-	Output	Aspect ratio
	uiSarWidth	-	Output	Aspect ratio width It is valid only when **uiAspectRatioIdc** is set to **255**.
	uiSarHeight	-	Output	Aspect ratio height It is valid only when **uiAspectRatioIdc** is set to **255**.
	uiVpsNumUnitsInTick	-	Output	Frame rate information
	uiVpsTimeScale	-	Output	
	stCuOutInfo	-	Output	Current output frame information, including number of various types of CUs in a frame
	bIsError	[0,1]	Output	Frame error flag
	uiLayerIdx	[0,1]	Output	Layer information flag

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

Parameter	Member	Value Range	Input/ Output	Description
	uiPoc	-	Output	POC value of the frame information

[Return Value]

Return Value	Macro Definition	Description
0	IHW265D_OK	The function is successfully called.
2	IHW265D_FIND_NEW_PIC	This value is returned when the start information of a new frame is detected.
0xF0401000	IHW265D_INVALID_ARGUMENT	The parameter is invalid.
0xF0401001	IHW265D_DECODER_NOT_CREATE	The decoder is not created.
0xF0402008	IHW265D_STREAMBUF_NULL	The stream address is empty.
0xF0404001	IHW265D_NAL_HEADER_ERR	The NAL header is incorrectly parsed.
0xF0404002	IHW265D_VPS_ERR	The VPS is incorrectly parsed.
0xF0404003	IHW265D_SPS_ERR	The SPS is incorrectly parsed.
0xF0404004	IHW265D_PPS_ERR	The PPS is incorrectly parsed.
0xF0404005	IHW265D_SLICEHEADER_ERR	The slice header is incorrectly parsed.
0xF0404006	IHW265D_SLICEDATA_ERR	The slice data is incorrectly parsed.
0xE0404007	IHW265D_FRAME_DECODE_WARN	Error codes may exist in the output pictures due to the abnormal streams.
-1	-	Error occurs when multiple streams parse the code streams.

[Note]

Note the following when calling this function:

- The input code stream of this function must contain only one frame of video streams. The video streams exceeding or less than one frame are considered as incorrect code streams.
- This function supports only the decoding sequence output mode, and the image is output immediately after being decoded. This function cannot be used for video streams that have different decoding sequence and display sequence.

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

- This function provides the function of transparent transmission of the time stamp. The input time stamp is stored in the image structure after decoding and output with the decoded image.

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

3 Data Type and Structure

3.1 Description of Common Data Types

The following are the main data types used by an API:

```
typedef signed char        INT8;
typedef signed short       INT16;
typedef signed int         INT32;
typedef unsigned char      UINT8;
typedef unsigned short     UINT16;
typedef unsigned int       UINT32;
typedef          __int64       INT64;
typedef unsigned __int64       UINT64;
typedef void*                  IH265DEC_HANDLE;
```

The following are the main function pointer types used by an API:

```
typedef void *(* IHWVIDEO_ALG_MALLOC_FXN)( UINT32 uiChannelID, UINT32
    uiSize);
typedef void  (* IHWVIDEO_ALG_FREE_FXN)( UINT32 uiChannelID, void *pMem);
typedef void  (* IHWVIDEO_ALG_LOG_FXN)( UINT32 uiChannelID,
IHWVIDEO_ALG_LOG_LEVEL eLevel, INT8 *pszMsg, ...);
```

3.2 Description of the Data Structure

3.2.1 HW265D_THREADTYPE

[Description]

Thread type enumeration

[Syntax]

```
typedef enum tagHW265D_THREADTYPE
{
    IH265D_SINGLE_THREAD = 0,   /* Single-thread */
    IH265D_MULTI_THREAD         /* Multi-thread */
} HW265D_THREADTYPE;
```

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

[Note]

None

3.2.2 HW265D_OUTPUTORDER

[Description]

Output sequence enumeration

[Syntax]

```
typedef enum tagHW265D_OUTPUTORDER
{
   IH265D_DECODE_ORDER = 0,     /*output in the decoding sequence*/
   IH265D_DISPLAY_ORDER,       /*output in the display sequence*/
} HW265D_OUTPUTORDER;
```

[Note]

None

3.2.3 HW265D_DECODEMODE

[Description]

Decoding mode

In a normal decoding process, some code streams and images are remained in the decoder. After the decoding is complete, these images must be output forcibly.

[Syntax]

```
typedef enum tagHW265D_DECODEMODE
{
   IH265D_DECODE = 0,           /* Normal decoding */
   IH265D_DECODE_END        /* The decoding completes, and the decoder is
   required to output remaining images.*/
} HW265D_DECODEMODE;
```

[Note]

None

3.2.4 HW265D_FRAMETYPE

[Description]

Frame type

[Syntax]

```
typedef enum tagHW265D_FRAMETYPE
{
    IH265D_FRAME_I = 0,
    IH265D_FRAME_P,
```

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

```
    IH265D_FRAME_B,
    IH265D_FRAME_UNKNOWN
} HW265D_FRAMETYPE;
```

[Note]

None

3.2.5 HW265D_DECODESTATUS

[Description]

Status of the decoder

[Syntax]

```
typedef enum tagHW265D_DECODESTATUS
{
  IH265D_GETDISPLAY = 0,        /* A frame has been decoded and output, which
  can be displayed. */
  IH265D_NEED_MORE_BITS,        /* The decoder does not output an image, and
  more code streams are required. */
  IH265D_NO_PICTURE,            /* The decoding is complete, and all images
  are output.*/
  IH265D_ERR_HANDLE         /* Handle error */
} HW265D_DECODESTATUS;
```

[Note]

None

3.2.6 IHW265D_INIT_PARAM

[Description]

Decoder initialization parameters

[Syntax]

```
/* Decoder initialization parameters*/
typedef struct tagIHW265D_INIT_PARAM
{
UINT32  uiChannelID;                /*Channel ID*/
    INT32   iMaxWidth;                  /*Maximum width */
    INT32   iMaxHeight;                 /*Maximum height */
    INT32   iMaxRefNum;                 /* Maximum number of reference
frames*/
    UINT32  uiBitDepth;                 /*Number of input bits*/
    INT32   iMaxVPSNum;                 /*Maximum number of input VPSs*/
    INT32  iMaxSPSNum;                  /*Maximum number of input SPSs*/
    INT32  iMaxPPSNum;                  /*Maximum number of input PPSs*/
    HW265D_THREADTYPE  eThreadType;     /*Thread type*/
```

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

```
    HW265D_OUTPUTORDER eOutputOrder;    /* Output sequence. It is valid only
for decoding frames. By default, decoding sequence is used for decoding AUs.
*/
    HW265D_USERDATA    *pstUserData;    /* User data */
    IHWVIDEO_ALG_MALLOC_FXN  MallocFxn; /*Allocation of the memory callback
function*/
    IHWVIDEO_ALG_FREE_FXN    FreeFxn;   /*Release of the memory callback
function*/
    IHWVIDEO_ALG_LOG_FXN     LogFxn;    /* Log callback function */
}IHW265D_INIT_PARAM;
```

[Note]

None

3.2.7 IH265DEC_INARGS

[Description]

Decoder input information

[Syntax]

```
/* Code stream, time stamp, and decoding mode inputted by the decoder*/
typedef struct tagIH265DEC_INARGS
{
    UINT8  *pStream;                /*Stream address*/
    UINT32 uiStreamLen;             /*Stream length*/
    UINT64 uiTimeStamp;             /*Time stamp*/
    HW265D_DECODEMODE eDecodeMode;  /* Decoding mode 0: Normal mode; 1: Flush
mode*/
}IH265DEC_INARGS;
```

[Note]

None

3.2.8 IH265DEC_OUTARGS

[Description]

Decoder output information

[Syntax]

```
/*Decoder output information*/
typedef struct tagIH265DEC_OUTARGS
{
    UINT32           uiChannelID;      /*Channel ID*/
    UINT32           uiBytsConsumed;   /* Number of consumed bytes*/
    UINT64           uiTimeStamp;      /*Time stamp*/
    HW265D_FRAMETYPE  eFrameType;      /*Output frame type*/
```

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

```
    HW265D_DECODESTATUS eDecodeStatus; /*Status of the decoder*/
    UINT32            uiDecWidth;       /* Image width*/
    UINT32            uiDecHeight;      /* Image height*/
    UINT32            uiYStride;        /* Image luminance stride*/
    UINT32            uiUVStride;       /* Image chrominance stride*/
    UINT32            uiBitDepthY;      /*Bit width of the output luminance
    component*/
    UINT32            uiBitDepthC;      /*Bit width of the output chrominance
    component*/
    INT32             iDisplayPrimariesX0;   /*  Horizontal coordinate of the
    green point of the output chrominance*/
    INT32             iDisplayPrimariesX1;   /*Horizontal coordinate of the
    blue point of the output chrominance*/
    INT32             iDisplayPrimariesX2;   /*Horizontal coordinate of the
    red point of the output chrominance*/
    INT32             iDisplayPrimariesY0;   /*Vertical coordinate of the
    green point of the output chrominance*/
    INT32             iDisplayPrimariesY1;   /*Vertical coordinate of the blue
    point of the output chrominance*/
    INT32             iDisplayPrimariesY2;   /*Vertical coordinate of the red
    point of the output chrominance*/
    INT32             iWhitePointX;      /*Horizontal coordinate of the white
    point of the output luminance*/
    INT32             iWhitePointY;      /*Vertical coordinate of the white
    point of the output luminance*/
    INT32             iMaxDisplayMasteringLuma;   /*Maximum output display
    luminance of HDR 10*/
    INT32             iMinDisplayMasteringLuma;   /*Minimum output display
    luminance of HDR 10*/
    INT32             iMaxContentLightLevel;   /*Maximum luminance level of the
    output content*/
    INT32             iMaxPicAverageLightLevel;   /*Average luminance level of
    the output image*/
    UINT8             *pucOutYUV[3];     /* Addresses of the Y, U, and V
    components*/
    HW265D_USERDATA   *pstUserData;      /* User data */

    UINT32            uiCodingBytesOfCurFrm; /* Original code stream length
    (number of bytes) of the current frame*/
    // vui information
    UINT32            uiAspectRatioIdc;     /* Aspect ratio*/
    UINT32            uiSarWidth;
    UINT32            uiSarHeight;
    // vps information
```

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

```
    UINT32            uiVpsNumUnitsInTick;  /* Frame rate information*/
    UINT32            uiVpsTimeScale;
    // cuinfo
    CU_OUTPUT_INFO    stCuOutInfo;          /*  */
    // errorinfo
    BOOL32            bIsError;             /* Frame error flag*/
    UINT32            uiLayerIdx;          /*Interlayer frame information*/
    UINT32            uiPoc;               /*POC value of the frame
information*/
}IH265DEC_OUTARGS;
```

3.2.9 HW265D_USERDATA

[Description]

User data

[Syntax]

```
/* User data */
typedef struct tagHW265D_USERDATA
{
    UINT32  uiUserDataType;     /*User data type*/
    UINT32  uiUserDataSize;     /*User data length*/
    UINT8   *pucData;           /* User data buffer*/
    struct tagHW265D_USERDATA  *pNext;  /* Point to the next segment of user
data */
}HW265D_USERDATA;
```

[Note]

None

3.2.10 IHWVIDEO_ALG_VERSION_STRU

[Description]

Decoding library version information

[Syntax]

```
/*Decoding library version information*/
typedef struct tagIHWVIDEO_ALG_VERSION
{
    INT8    cVersionChar[IHWVIDEO_ALG_VERSION_LENGTH];  /*Version number*/
    INT8    cReleaseTime[IHWVIDEO_ALG_TIME_LENGTH];     /* Compiling time*/
UINT32  uiCompileVersion;                          /*Compiler version
number*/
}IHWVIDEO_ALG_VERSION_STRU;
```

[Note]

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

None

3.2.11 CU_OUTPUT_INFO

[Description]

Number of various types of CUs in each frame image

[Syntax]

```
/* Number of various types of CUs in each frame image*/
typedef struct tagCU_OUTPUT_INFO
{
    UINT32 uiCuNumIntra4;       /* intra 4x4 CU number */
    UINT32 uiCuNumIntra8;       /* intra 8x8 CU number */
    UINT32 uiCuNumIntra16;      /* intra 16x16 CU number */
    UINT32 uiCuNumIntra32;      /* intra 32x32 CU number */
   UINT32 uiCuNumIntra64;      /* intra 64x64 CU number */
    UINT32 uiCuNumPcm4;         /* IPCM 4x4 CU number */
    UINT32 uiCuNumPcm8;         /* IPCM 8x8 CU number */
    UINT32 uiCuNumPcm16;        /* IPCM 16x16 CU number */
    UINT32 uiCuNumPcm32;        /* IPCM 32x32 CU number */
    UINT32 uiCuNumPcm64;        /* IPCM 64x64 CU number */
    UINT32 uiCuNumInter8;       /* inter 8x8 CU number */
    UINT32 uiCuNumInter16;      /* inter 16x16 CU number */
    UINT32 uiCuNumInter32;      /* inter 32x32 CU number */
    UINT32 uiCuNumInter64;      /* inter 64x64 CU number */
    UINT32 uiCuNumSkip8;        /* skip 8x8 CU number */
    UINT32 uiCuNumSkip16;       /* skip 16x16 CU number */
    UINT32 uiCuNumSkip32;       /* skip 32x32 CU number */
    UINT32 uiCuNumSkip64;       /* skip 64x64 CU number */
} CU_OUTPUT_INFO;
```

[Note]

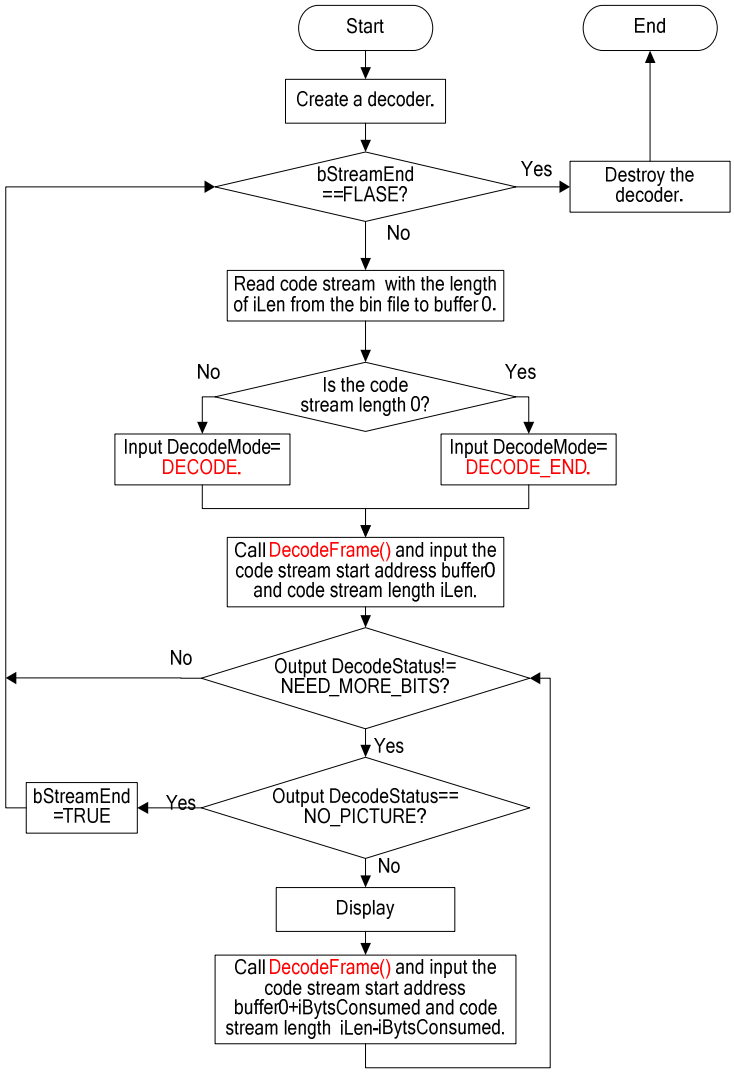
None

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

4 API Application Instance

4.1 Stream Decoding Process

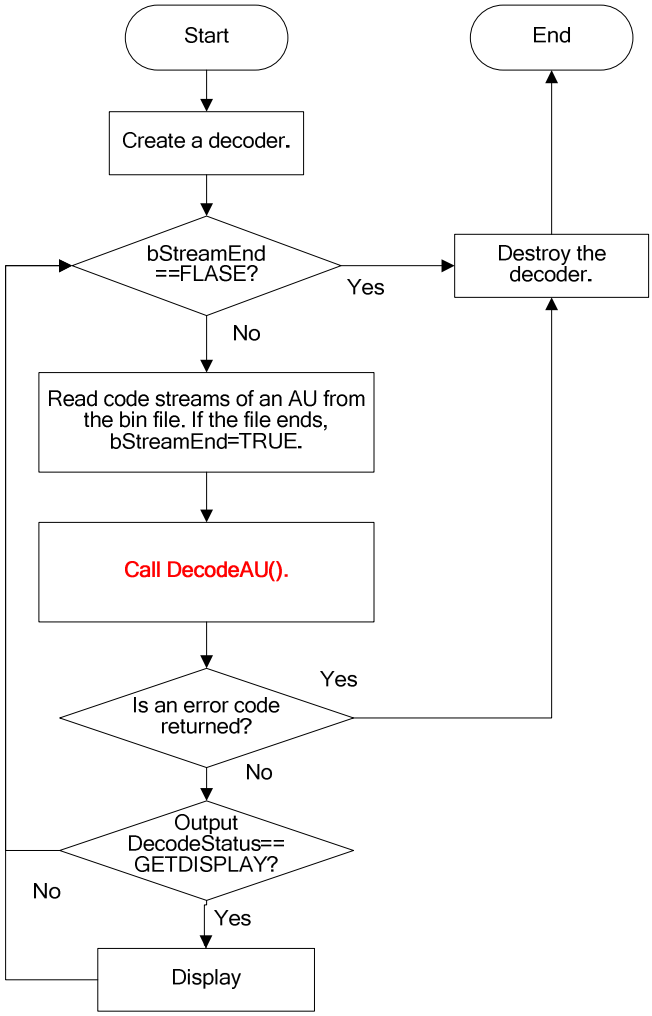
Figure 4-1 Stream decoding process



 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

4.2 Frame Decoding Process

Figure 4-2 Frame decoding process



4.3 Program Instance

```
void *HW265D_Malloc(UINT32 channel_id, UINT32 size)
{
    return (void *)malloc(size);
}
void HW265D_Free(UINT32 channel_id, void * ptr)
{
    free(ptr);
}

void HW265D_Log( UINT32 channel_id, IHWVIDEO_ALG_LOG_LEVEL eLevel, INT8
```

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

```
*p_msg, ...)
{
    printf("%s.\n", p_msg);
}

INT32 H265DecLoadAU(UINT8* pStream, UINT32 iStreamLen, UINT32* pFrameLen)
{
   UINT32 i;
   UINT32 state = 0xffffffff;
   BOOL32 bFrameStartFound=0;

   *pFrameLen = 0;
   if( NULL == pStream || iStreamLen <= 4)
   {
      return -1;
   }

   for( i = 0; i < iStreamLen; i++)
   {
      if( (state & 0xFFFFFF7E) >= 0x100 &&
          (state & 0xFFFFFF7E) <= 0x13E )
      {
         if( 1 == bFrameStartFound )
         {
            if( (pStream[i+1]>>7) == 1)
            {
               *pFrameLen = i - 4;
               return 0;
            }
         }
         else
         {
            bFrameStartFound = 1;
         }
      }
      /*find a vps, sps, pps*/
      if(  (state&0xFFFFFF7E) == 0x140 ||
           (state&0xFFFFFF7E) == 0x142 ||
           (state&0xFFFFFF7E) == 0x144)
      {
         if(1 == bFrameStartFound)
         {
            *pFrameLen = i - 4;
            return 0;
```

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

```
            }
            else
            {
                bFrameStartFound = 1;
            }
        }

        state = (state << 8) | pStream[i];
    }

    *pFrameLen = i;
    return -1;
}

int main(int argc, unsigned char** argv)
{
    FILE *fpInFile = NULL;
    FILE *fpOutFile = NULL;
    INT32 iRet = 0;
    UINT8 *pInputStream = NULL, *pStream;
    UINT32 uiChannelId = 0;
    UINT32 iFrameIdx = 0;
    BOOL32 bStreamEnd = 0;
    INT32 iFileLen;

    IH265DEC_HANDLE hDecoder = NULL;
    IHW265D_INIT_PARAM stInitParam = {0};
    IH265DEC_INARGS stInArgs;
    IH265DEC_OUTARGS stOutArgs = {0};

    /* open input stream file and output yuv file  */
    fpInFile = fopen(argv[1], "rb");
    fpOutFile = fopen(argv[2], "wb");
    if (NULL == fpInFile || NULL == fpOutFile)
    {
        fprintf(stderr, "Unable to open h265 stream file %s or yuv file %s.\n",
                    argv[1], argv[2]);
        goto exitmain;
    }

    printf("decoding file: %s...\n", argv[1]);
    printf("save yuv file: %s...\n", argv[2]);

    /* malloc stream buffer */
```

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

```
    fseek( fpInFile, 0, SEEK_END);
    iFileLen = ftell( fpInFile);
    fseek( fpInFile, 0, SEEK_SET);
    pInputStream = (unsigned char *) malloc(iFileLen);
    if (NULL == pInputStream)
    {
        fprintf(stderr, "Malloc failed! \n");
        goto exitmain;
    }

    /* create decode handle */
stInitParam.uiChannelID    = 0;
    stInitParam.iMaxWidth      = 1920;
    stInitParam.iMaxHeight     = 1088;
    stInitParam.iMaxRefNum     = 2;
    stInitParam.eThreadType    = IH265D_SINGLE_THREAD;    // or
IH265D_MULTI_THREAD;
    stInitParam.eOutputOrder   = IH265D_DECODE_ORDER;     // or
IH265D_DISPLAY_ORDER;
    stInitParam.MallocFxn      = HW265D_Malloc;
    stInitParam.FreeFxn        = HW265D_Free;
    stInitParam.LogFxn         = HW265D_Log;
    iRet = IHW265D_Create(&hDecoder, &stInitParam);
    if (IHW265D_OK != iRet)
    {
        fprintf(stderr, "Unable to create decoder.\n");
        goto exitmain;
    }

    /* read H.265 stream to stream buffer */
    fread(pInputStream, 1, iFileLen, fpInFile);
    pStream = pInputStream;

while(!bStreamEnd)
   {
       INT32 iNaluLen;
       H265DecLoadAU(pStream, iFileLen, &iNaluLen);

       stInArgs.eDecodeMode =  iNaluLen>0 ? IH265D_DECODE :
IH265D_DECODE_END;
       stInArgs.pStream = pStream;
       stInArgs.uiStreamLen = iNaluLen;

       pStream += iNaluLen;
```

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

```
        iFileLen-= iNaluLen;

        stOutArgs.eDecodeStatus = -1;
        stOutArgs.uiBytsConsumed = 0;

        while(stOutArgs.eDecodeStatus != IH265D_NEED_MORE_BITS)
        {
            if(stOutArgs.eDecodeStatus == IH265D_NO_PICTURE)
            {
                bStreamEnd = 1;
                break;
            }
            if (stOutArgs.eDecodeStatus == IH265D_GETDISPLAY)
            {
#if COUNT_SINGLE_FRAME_TIME
                  EndTimeS = GetTime_us();
                SumTimeS = EndTimeS - StartTimeS;
                  if (IH265D_FRAME_I == stOutArgs.eFrameType)
                  {
                      IFrameSum += SumTimeS;
                      IFrameMin = IFrameMin > SumTimeS ? SumTimeS : IFrameMin;
                      IFrameMax = IFrameMax < SumTimeS ? SumTimeS : IFrameMax;
                      iFrameIdxI++;
                  }
                  else if (IH265D_FRAME_P == stOutArgs.eFrameType)
                  {
                      PFrameSum += SumTimeS;
                      PFrameMin = PFrameMin > SumTimeS ? SumTimeS : PFrameMin;
                      PFrameMax = PFrameMax < SumTimeS ? SumTimeS : PFrameMax;
                      iFrameIdxP++;
                  }
                  StartTimeS = GetTime_us();
#endif
                if (fpOutFile != NULL && stOutArgs.uiLayerIdx == 0)
                {
                   UINT32 i;
                      UINT32 j;
                     UINT8 Const_0[1] ={0} ;
                   if (TRUE != pstDecParam->bCheckMd5)
                   {
                          if(stOutArgs.uiBitDepthY == 8)
                          {
                              if(stOutArgs.uiBitDepthC == 8)
                              {
```

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

```
                                for (i=0;i<stOutArgs.uiDecHeight;i++)
                                {

    fwrite(stOutArgs.pucOutYUV[0]+i*stOutArgs.uiYStride, 1,
stOutArgs.uiDecWidth, fpOutFile);
                                }
                            }
                            else
                            {
                                for (i=0;i<stOutArgs.uiDecHeight;i++)
                                {
                                    for(j=0;j<stOutArgs.uiDecWidth;j++)
                                    {

   fwrite(stOutArgs.pucOutYUV[0]+i*stOutArgs.uiYStride+j, 1, 1, fpOutFile);

                                        fwrite(Const_0, 1, 1, fpOutFile);

                                    }
                                }
                            }
                        }
                        else
                        {
                            for (i=0;i<stOutArgs.uiDecHeight;i++)
                            {

    fwrite(stOutArgs.pucOutYUV[0]+i*stOutArgs.uiYStride*sizeof(INT16), 1,
stOutArgs.uiDecWidth*sizeof(INT16), fpOutFile);    //modified by gaoshiyi

                            }
                        }
                        if(stOutArgs.uiBitDepthC == 8)
                        {
                            if(stOutArgs.uiBitDepthY == 8)
                            {
                                for
(i=0;i<((stOutArgs.uiDecHeight)>>1);i++)
                                {

    fwrite(stOutArgs.pucOutYUV[1]+i*stOutArgs.uiUVStride, 1,
(stOutArgs.uiDecWidth>>1), fpOutFile);
                                }
                                for
```

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

```
(i=0;i<((stOutArgs.uiDecHeight)>>1);i++)
                                {

    fwrite(stOutArgs.pucOutYUV[2]+i*stOutArgs.uiUVStride, 1,
(stOutArgs.uiDecWidth>>1), fpOutFile);
                                }
                            }
                            else
                            {
                                for
(i=0;i<((stOutArgs.uiDecHeight)>>1);i++)
                                {

    for(j=0;j<((stOutArgs.uiDecWidth)>>1);j++)
                                    {

    fwrite(stOutArgs.pucOutYUV[1]+i*stOutArgs.uiUVStride+j, 1, 1,
fpOutFile);
                                        fwrite(Const_0, 1, 1, fpOutFile);

                                    }
                                }
                                for
(i=0;i<((stOutArgs.uiDecHeight)>>1);i++)
                                {

    for(j=0;j<((stOutArgs.uiDecWidth)>>1);j++)
                                    {

    fwrite(stOutArgs.pucOutYUV[2]+i*stOutArgs.uiUVStride+j, 1, 1,
fpOutFile);
                                        fwrite(Const_0, 1, 1, fpOutFile);

                                    }
                                }
                            }
                        }
                        else
                        {
                            for (i=0;i<((stOutArgs.uiDecHeight)>>1);i++)
                            {

    fwrite(stOutArgs.pucOutYUV[1]+i*stOutArgs.uiUVStride*sizeof(INT16), 1,
(stOutArgs.uiDecWidth>>1)*sizeof(INT16), fpOutFile);
```

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

```
                        }
                        for (i=0;i<((stOutArgs.uiDecHeight)>>1);i++)
                        {

    fwrite(stOutArgs.pucOutYUV[2]+i*stOutArgs.uiUVStride*sizeof(INT16), 1,
(stOutArgs.uiDecWidth>>1)*sizeof(INT16), fpOutFile);
                        }
                    }
                }
              else                  {
                  for (i = 0; i < stOutArgs.uiDecHeight; i++)
                  {
                      MD5Update (&context,
stOutArgs.pucOutYUV[0]+i*stOutArgs.uiYStride, stOutArgs.uiDecWidth);
                  }

                  for (i = 0; i < ((stOutArgs.uiDecHeight)>>1); i++)
                  {
                      MD5Update (&context,
stOutArgs.pucOutYUV[1]+i*stOutArgs.uiUVStride, stOutArgs.uiDecWidth>>1);
                  }

                  for (i = 0; i < ((stOutArgs.uiDecHeight)>>1); i++)
                  {
                      MD5Update (&context,
stOutArgs.pucOutYUV[2]+i*stOutArgs.uiUVStride, stOutArgs.uiDecWidth>>1);
                  }
              }
                  fprintf( stdout, "Layer %d  POC %d  FrameType %d\n",
stOutArgs.uiLayerIdx, stOutArgs.uiPoc, stOutArgs.eFrameType);
          }

            if (fpOutFileIL != NULL && stOutArgs.uiLayerIdx == 1)
            {
                UINT32 i;
                if (TRUE != pstDecParam->bCheckMd5)
                {
                    for (i=0;i<stOutArgs.uiDecHeight;i++)
                    {

    fwrite(stOutArgs.pucOutYUV[0]+i*stOutArgs.uiYStride, 1,
stOutArgs.uiDecWidth, fpOutFileIL);
                    }
```

```
                        for (i=0;i<((stOutArgs.uiDecHeight)>>1);i++)
                        {

    fwrite(stOutArgs.pucOutYUV[1]+i*stOutArgs.uiUVStride, 1,
stOutArgs.uiDecWidth>>1, fpOutFileIL);
                        }
                        for (i=0;i<((stOutArgs.uiDecHeight)>>1);i++)
                        {

    fwrite(stOutArgs.pucOutYUV[2]+i*stOutArgs.uiUVStride, 1,
stOutArgs.uiDecWidth>>1, fpOutFileIL);
                        }
                    }
                    else                    {
                        for (i = 0; i < stOutArgs.uiDecHeight; i++)
                        {
                            MD5Update (&context,
stOutArgs.pucOutYUV[0]+i*stOutArgs.uiYStride, stOutArgs.uiDecWidth);
                        }

                        for (i = 0; i < ((stOutArgs.uiDecHeight)>>1); i++)
                        {
                            MD5Update (&context,
stOutArgs.pucOutYUV[1]+i*stOutArgs.uiUVStride, stOutArgs.uiDecWidth>>1);
                        }

                        for (i = 0; i < ((stOutArgs.uiDecHeight)>>1); i++)
                        {
                            MD5Update (&context,
stOutArgs.pucOutYUV[2]+i*stOutArgs.uiUVStride, stOutArgs.uiDecWidth>>1);
                        }
                    }
                        fprintf( stdout, "Layer %d  POC %d  FrameType %d\n",
stOutArgs.uiLayerIdx, stOutArgs.uiPoc, stOutArgs.eFrameType);
                }

             if (pstDecParam->bDisplayEnable == TRUE)
             {
#ifdef ARCH_X86
                /* init display */
                if (0==iFrameIdx)
                {
                   Init_SDL((stOutArgs.uiYStride -
stOutArgs.uiDecWidth)/2, stOutArgs.uiDecWidth, stOutArgs.uiDecHeight);
```

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.

```
                }
                SDL_Display((stOutArgs.uiYStride -
stOutArgs.uiDecWidth)/2, stOutArgs.uiDecWidth, stOutArgs.uiDecHeight,
                    stOutArgs.pucOutYUV[0], stOutArgs.pucOutYUV[1],
stOutArgs.pucOutYUV[2]);
#endif
            }
//            printf("frame index %6d\r", iFrameIdx);
            iFrameIdx++;
        }
    }
}
```

 Copyright © HiSilicon (Shanghai) Technologies Co., Ltd.