



Client Deinterlace Library

API Reference

Issue	04
Date	2008-11-30
Part Number	N/A

Shenzhen Hisilicon Semiconductor Co., Ltd. provides customers with comprehensive technical support and service. Please feel free to contact our local office or company headquarters.

Shenzhen Hisilicon Semiconductor Co., Ltd.

Address: Manufacture Center of Huawei Electric,
Huawei Industrial Base,
Bantian, Longgang District,
Shenzhen, 518129
People's Republic of China

Website: <http://www.hisilicon.com>

Tel: +86-755-28788858

Fax: +86-755-28357515

Email: support@hisilicon.com

Copyright © Shenzhen Hisilicon Semiconductor Co., Ltd. 2007-2008. 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 Shenzhen Hisilicon Semiconductor Co., Ltd.

Trademarks and Permissions



and Hisilicon are trademarks of Shenzhen Hisilicon Semiconductor Co., Ltd.

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

Notice

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 the warranty of any kind, express or implied.



Contents

About This Document.....	1
1 Introduction.....	1-1
1.1 Overview of Deinterlace Library.....	1-2
1.2 Document Notes	1-2
1.3 Deinterlace SDK.....	1-3
2 API Overview.....	2-1
2.1 Deinterlace Initialization and Release APIs	2-2
2.2 Deinterlace Main API.....	2-2
2.3 API Calling Procedure.....	2-2
3 APIs.....	3-1
3.1 HI_InitDeinterlace.....	3-2
3.2 HI_ReleaseDeinterlace	3-4
3.3 HI_Deinterlace	3-5
3.4 HI_GetVersion.....	3-7
3.5 HI_SetOsd	3-8
4 Other Information.....	4-1
4.1 Common Data Structures and Data Types	4-2
4.2 Error Codes	4-3



Figures

Figure 2-1 Flow chart of calling deinterlace APIs	2-3
--	-----



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
Hi3510 Communications Media Processor	V100
Hi3511 H.264 Encoding and Decoding Processor	V100
Hi3512 H.264 Encoding and Decoding Processor	V100

Intended Audience

This document describes the reference information based on the deinterlace library and is intended for the programmers who meet the following requirements:

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

Organization

This document is organized as follows:

Chapter	Content
1 Introduction	Describes the functions of the deinterlace library, deinterlace API names, and the content of the SDK.
2 API Overview	Provides an overview of the deinterlace APIs.








Chapter	Content
3 APIs	Describes all the APIs in detail according to the functions.
4 Other Information	Provides some supplementary information, including the data types, common structures, and error codes.

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 .
<i>Italic</i>	Book titles are in <i>italics</i> .
Courier New	Terminal display is in Courier New.



Update History

Updates between document versions are cumulative. Therefore, the latest document version contains all updates made to previous versions.

Updates in Issue 04 (2008-11-10)

The third commercial release has the following updates:

Information about the Hi3512 is added.

Updates in Issue 03 (2008-04-21)

The second commercial release has the following updates:

Information about the Hi3511 is added.

Updates in Issue 02 (2007-11-30)

The initial commercial release has the following updates:

Chapter 1 Introduction

Deinterlace performance comparison file "Introdection to Client Deinterlace.pdf" is deleted in section 1.3 "Deinterlace SDK."

Chapter 3 APIs

API function HI_SetOsd is added.

Chapter 4 Other Information

- Data structure DEINTERLACE_OSDRECT_S is added in section 4.1 "Common Data Structures and Data Types."
- Error code HI_ERR_OSDNUM is added in section 4.2 "Error Codes."

Updates in Issue 01 (2007-04-20)

Initial release.



1 Introduction

About This Chapter

The following table lists the contents of this chapter.

Section	Describes
1.1 Overview of Deinterlace Library	The functions and features of the deinterlace library.
1.2 Document Notes	The APIs contained in this document.
1.3 Deinterlace SDK	The documents included in the deinterlace SDK.



1.1 Overview of Deinterlace Library

At present, TV programs still use interlaced video signals. In addition, current video sampling devices also scan in an interlaced manner. The time deviation between the odd field and the even field that compose a frame causes quality problems, such as sawtooth and stringing, during the play of interlaced displays. Therefore, the video requires the deinterlaced processing before being played.

The D1 pictures sampled by Hisilicon are encoded on a field basis. After decoded at the PC end, the pictures need to be deinterlaced. Hisilicon provides the deinterlace library, namely, DllDeinterlace.lib.

DllDeinterlace.lib is a static library in Windows to convert interlaced pictures into line-by-line pictures, that is, to convert two field pictures to a frame picture. You can get one YUV420 frame after inputting every two decoded YUV420 fields. The advantages of DllDeinterlace.lib are as follows:

- Clear static pictures
The vertical resolution of a static picture reaches 500 lines, which well maintains the detailed grains. The picture edges are clear and sharp, and sawtooth that is easy to occur on picture edges is also eliminated.
- No stringing in moving pictures
The edges of the moving objects are smooth without clear sawtooth. For the strenuous moving objects, the deinterlace library removes the stringing.
- Anti-blinking and anti-noise
The deinterlace library is added with the anti-blinking function that stabilizes pictures. In addition, with the anti-noise function, the deinterlace library is highly advantageous for the pictures with many noise dots.
- Elimination of field loss
In some special test stream, the pictures exist only in odd fields or even fields. In such a case, some field information is lost in the previous solutions. However, the deinterlace library solves this problem.
- High efficiency that meets project requirements
In project applications, the picture quality and the picture processing efficiency restricts each other. They cannot be preferred at the same time. Guaranteeing the picture quality, the deinterlace library also provides satisfactory picture processing efficiency.

1.2 Document Notes

This document describes a set of APIs, including:

- Deinterlace initialization API
- Deinterlace release API
- Deinterlace main API



1.3 Deinterlace SDK

The deinterlace SDK consists of the following five files:

- Header file: *DllDeinterlace.h*
- Static library file: *DllDeinterlace.lib*
- Dll file: *DllDeinterlace.lib*, *DllDeinterlace.dll*
- API calling sample: *sample.cpp*
- Deinterlace application reference file: *Client Deinterlace Library API Reference.pdf*



2 API Overview

About This Chapter

The following table lists the contents of this chapter.

Section	Describes
2.1 Deinterlace Initialization and Release APIs	The functions of the deinterlace initialization API and the deinterlace release API.
2.2 Deinterlace Main API	The functions of the deinterlace main function and lists the precautions.
2.3 API Calling Procedure	The steps to call the deinterlace APIs.



2.1 Deinterlace Initialization and Release APIs

Before the deinterlaced processing for the first time, the deinterlace initialization application programming interface (API) performs the following operations:

- Allocating the deinterlace space
- Initializing the deinterlace variables and status
- Setting the height, width, and pitch of the input field picture
- Setting the pitch of the output frame picture

The deinterlace handle is returned if the initialize API is successfully called.

After the deinterlace processing is finished, the deinterlace release API releases the deinterlace library space to prevent memory leakage.



CAUTION

After calling the deinterlace release API, you need to set the deinterlace handle to null to prevent this handle from being used by mistake.

2.2 Deinterlace Main API



CAUTION

Before calling the deinterlace main API, set the value for the even/odd field flag. When you call the main API, you need to input the flags in turn (odd and then even, repetitively) when you call a deinterlace API each time.

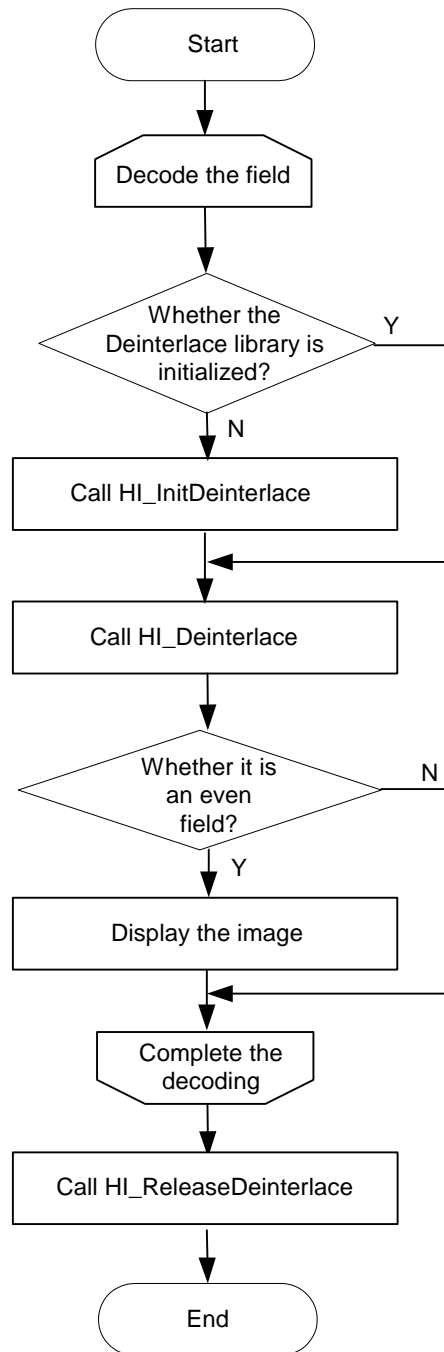
The deinterlace main API outputs a YUV420 frame picture every two YUV420 field pictures. That is, if you input three components, Y, U, and V, of a YUV420 picture, then the components are output with their respective storage space doubled.

After decoding a field, call the deinterlace main API, and you will get a frame picture every two field pictures. **NOTE**

An odd field is also called a top field, and an even field is also called a bottom field.

2.3 API Calling Procedure

Figure 2-1 shows the flow chart of calling the deinterlace APIs.

Figure 2-1 Flow chart of calling deinterlace APIs

If you perform the deinterlace processing after the first decoding, call `Hi_InitDeinterlace` first, and then call `Hi_Deinterlace`. If it is not the first deinterlace processing, directly call `Hi_Deinterlace` after decoding.

Before calling `Hi_InitDeinterlace`, you need to set the following structure parameters:

- Width, height, and pitch of the input field picture
- Pitch of the output frame picture



The deinterlace library saves the information, and allocates corresponding deinterlace space for Hi_Deinterlace.

Before calling the Hi_Deinterlace each time, you need to set the field flag. The deinterlace library outputs a frame picture every even field according to the input field flag. If the input flag is illegal, the input picture is not processed and the error code is returned.

When you finish the deinterlace processing, call Hi_ReleaseDeinterlace to release the allocated deinterlace space.



CAUTION

- In the deinterlace library, the size of the buffer allocated for the Y, U, or V components of the output frame should be at least $Stride \times FiledHeight \times 2$.
- After calling Hi_ReleaseDeinterlace, you need to set the deinterlace handle to null.



3 APIs

About This Chapter

The following table lists the contents of this chapter.

Section	Describes
3.1 HI_InitDeinterlace	The deinterlace initialize API HI_InitDeinterlace.
3.2 HI_ReleaseDeinterlace	The deinterlace release API HI_ReleaseDeinterlace.
3.3 HI_Deinterlace	The deinterlace main API HI_Deinterlace.
3.4 HI_GetVersion	The version get API HI_GetVersion.
3.5 HI_SetOsd	The OSD area setting API HI_SetOsd.



3.1 HI_InitDeinterlace

[Purpose]

Initializes the deinterlace library.

[Syntax]

```
int HI_InitDeinterlace(void **pHandle, DEINTERLACE_PARA_S struPara);
```

[Description]

HI_InitDeinterlace is used to:

- Create the deinterlace handle.
- Initialize the deinterlace library.
- Save the input parameters.

[Parameters]

Parameter	Description	Input/Output	Global/Local
**pHandle	The deinterlace handle, a pointer to another pointer. This handle is returned for users to use. It is used by both HI_Deinterlace and HI_ReleaseDeinterlace. This handle is used to distinguish the deinterlace libraries to prevent the library interference when playing multiple channels of D1 pictures.	Output	Local
struPara	Structure parameters to be set, including the height, width, and pitch of the input YUV420 field picture, and the pitch of the output YUV420 frame picture. In the deinterlace library, <ul style="list-style-type: none">• The width of the input field picture should be 128 at least.• The height of the input field picture should be 4 at least.• The pitch of the input Y components should not be less than the picture width.• The pitch of the input U or V components should not be less than half of the picture width. The output picture has the same pitch requirements as the input one. For details, refer to DEINTERLACE_PARA_S .	Input	Local



[Return Value]

Return Value	Description
0	Success.
Non-zero	Failure. The return value is the error code.

[Error Code]

Error Code	Description
HI_SUCCESS_DEINTERLACE	Success.
HI_ERR_MALLOC	Failed to create the space.
HI_ERR_PITCH	The width or the height of the input or output picture is invalid.

[Request]

- Header file: DllDeinterlace.h
- Lib file: DllDeinterlace.lib

[Note]

The deinterlace handle must be null before HI_InitDeinterlace is called.

[Example]

Refer to the code in sample.cpp.

3.2 HI_ReleaseDeinterlace

[Purpose]

Releases the space in the deinterlace library.

[Syntax]

```
int HI_ReleaseDeinterlace(void *pHandle);
```

[Description]

HI_ReleaseDeinterlace is used to release the space in the deinterlace library.

[Parameters]



Parameter	Description	Input/Output	Global/Local
*pHandle	The deinterlace handle created in deinterlace initialization. This handle is used to distinguish the deinterlace libraries to prevent the library interference when playing multiple channels of D1 pictures.	Input	Local

[Return Value]

Return Value	Description
0	Success.
Non-zero	Failure. The return value is the error code.

[Error Code]

Error Code	Description
HI_SUCCESS_DEINTERLACE	Success.
HI_ERR_POINT_NULL	The input pointer is null.

[Request]

- Header file: DllDeinterlace.h
- Library file: DllDeinterlace.lib

[Note]

The deinterlace handle must be null before HI_ReleaseDeinterlace is called.

[Example]

Refer to the code in sample.cpp.

3.3 HI_Deinterlace

[Purpose]

Outputs a frame picture every two fields.

YUV420 field pictures are input for the output of YUV420 frame pictures.

[Syntax]

```
int HI_Deinterlace(  
    void *pHandle,  
    DEINTERLACE_FRAME_S struDstFrame,
```



```
    unsigned char *pszSrcY,  
    unsigned char *pszSrcU,  
    unsigned char *pszSrcV,  
    PIC_TYPE_E tFieldFlag  
);
```

[Description]

Hi_Deinterlace is used to perform deinterlace processing, specifically, outputs a frame picture every two field pictures. Input two fields of YUV420 picture, and one frame of YUV420 picture is output. After the decoding is finished, you can directly call this interface.



NOTE

The YUV420 picture for video display can be converted to a picture format supported by the display adapter. The destination space for the format conversion can be the video memory.

[Parameters]

Parameter	Description	Input/Output	Global/Local
*pHandle	The deinterlace handle, which is used to distinguish the deinterlace libraries to prevent the library interference when playing multiple channels of D1 pictures.	Input	Local
struDstFrame	The output frame picture. The frame consists of three members, namely, Y, U, and V components. For details, refer to DEINTERLACE_FRAME_S .	Output	Local
*pszSrcY	The address of the Y components in the input YUV420 field picture.	Input	Local
*pszSrcU	The address of the U components in the input YUV420 field picture.	Input	Local
*pszSrcV	The address of the V components in the input YUV420 field picture.	Input	Local
tFieldFlag	Flag that indicates an odd field or even field. A frame picture is output only when a bottom field picture is input. If a top picture is input, it is returned without being processed and no frame picture is output.	Input	Local

[Return Value]



Return Value	Description
0	Success.
Non-zero	Failure. The return value is the error code.

[Error Code]

Error Code	Description
HI_SUCCESS_DEINTERLACE	Success.
HI_ERR_POINT_NULL	The input pointer is null.
HI_ERR_FIELD_FLAG	The input is not the field flag.

[Request]

- Header file: DllDeinterlace.h
- Library file: DllDeinterlace.lib

[Note]

Before calling HI_Deinterlace, you need to set the field flag.

[Example]

Refer to the code in sample.cpp.

3.4 HI_GetVersion

[Purpose]

Gets the version of the deinterlace library.

[Syntax]

```
int HI_GetVersion(char **pszVersion);
```

[Description]

HI_GetVersion is used to get the version of the deinterlace library.

[Parameters]

Parameter	Description	Input/Output	Global/Local
**pszVersion	Character string of the output deinterlace version	Output	Local

[Return Value]



Return Value	Description
0	Success.
Non-zero	Failure. The return value is the error code.

[Error Code]

Error Code	Description
HI_SUCCESS_DEINTERLACE	Success.
HI_ERR_POINT_NULL	The input pointer is null.

[Request]

- Header file: DllDeinterlace.h
- Library file: DllDeinterlace.lib

[Note]

The input parameter is the address of the character string pointer.

[Example]

```
char *pszVersion = NULL;  
pszVersion = new char[256];  
HI_GetVersion((char **)&pszVersion);
```



NOTE

For details, refer to the code in sample.cpp.

3.5 HI_SetOsd

[Purpose]

Set an OSD rectangle area.

[Syntax]

```
int HI_SetOsd(void *pHandle, DEINTERLACE_OSDRECT_S rcOsd[], int iOsdNum);
```

[Description]

- When the HI_SetOsd interface is called, the OSD rectangle area is subject to the following four qualifications:
 - The configured OSD rectangle area must be translucence or opacity.
 - The position information of the OSD rectangle area is passed from the board OSD rectangle area. And the OSD rectangle area is in the upper left or the upper right corner of the video.



- The OSD rectangle area must be character or time OSD.
- The OSD rectangle area cannot be too big; otherwise, the Deinterlace performance is affected.
- After being set, an OSD rectangle area has the Deinterlace processing so that its image quality can be improved.
- Up to four OSDs are supported, and the OSD rectangle area must be within the bound of the video image.

[Parameters]

Parameter	Description	Input/Output	Global/Local
*pHandle	Deinterlace handle.	Input	Local
rcOsd []	Input OSD rectangle area array. For details, refer to the DEINTERLACE_OSDRECT_S structure.	Input	Local
iOsdNum	Number of the input OSD rectangle area arrays. The maximum value is four.	Input	Local

[Return Value]

Return Value	Description
0	Success.
Non-zero	Failure. The return value is the error code.

[Error Code]

Error Code	Description
HI_SUCCESS_DEINTERLACE	Success.
HI_ERR_POINT_NULL	The input pointer is null.
HI_ERR_OSDNUM	The input OSD number is invalid.

[Request]

- Header file: DllDeinterlace.h
- Library file: DllDeinterlace.lib

[Example]

```
DEINTERLACE_OSDRECT_S rcOsd[1];  
rcOsd[0].x = 515;  
rcOsd[0].y = 0;  
rcOsd[0].w = 185;
```



```
rcOsd[0].h = 20;  
HI_SetOsd(m_pHandle, rcOsd, 1);
```



4 Other Information

About This Chapter

The following table lists the contents of this chapter.

Section	Describes
4.1 Common Data Structures and Data Types	The common structures and data types that are related to the APIs.
4.2 Error Codes	The error codes related to the APIs.



4.1 Common Data Structures and Data Types

PIC_TYPE_E

```

/*****
Field flag, specifically, the flag for an even or odd field.
*****/
typedef enum
{
    PIC_PROGRESSIVE = 0,          //Frame
    PIC_INTERLACED_ODD,          //Odd field (top field)
    PIC_INTERLACED_EVEN          //Even field (bottom field)
}PIC_TYPE_E;

```

DEINTERLACE_PARA_S

```

/*****
*The structure provides parameters input for deinterlace initialization. The caller needs to configure the
members.
*The members include the width and height of the input field picture, and the pitch of the YUV components.
*The members also include the pitch of the YUV components in the output frame picture.
*****/
typedef struct hiDEINTERLACE_PARA_S
{
    int iFieldWidth;              //The width of the input field picture, 128 at least.
    int iFieldHeight;             //The height of the input field picture, 4 at least.
    int iSrcYPitch;               //Pitch of the Y components in the input field picture,
                                //not less than the picture width.
    int iSrcUVPitch;              //Pitch of the UV components in the input field picture,
                                //at least half of the picture width.
    int iDstYPitch;               //Pitch of the Y components in the output frame picture,
                                //not less than the picture width.
    int iDstUVPitch;              //Pitch of the UV components in the output frame picture,
                                //at least half of the picture width.
}DEINTERLACE_PARA_S;

```

DEINTERLACE_FRAME_S

```

/*****
*This structure contains the storage addresses of the YUV420 picture.
*The members include the storage addresses of the Y, U, and V components.
*****/
typedef struct hiDEINTERLACE_FRAME_S

```



```
{
    unsigned char *pszY;        //Address of the Y components.
    unsigned char *pszU;        //Address of the U components.
    unsigned char *pszV;        //Address of the V components.
}DEINTERLACE_FRAME_S;
```

DEINTERLACE_OSDRECT_S

```
/******
*The structure is the value of the OSD area position.
*****/
typedef struct hiDEINTERLACE_OSDRECT_S
{
    int x;        //X coordinate of the upper left corner of the OSD area.
    int y;        //Y coordinate of the upper left corner of the OSD area.
    int w;        //Width of the OSD area.
    int h;        //Height of the OSD area.
}DEINTERLACE_OSDRECT_S;
```

4.2 Error Codes

```
/******
*The error codes are defined as follows:
*****/
#define HI_SUCCESS_DEINTERLACE    0    //Success.
#define HI_ERR_MALLOC             -1    //Failed to create the space.
#define HI_ERR_PITCH              -2    //The width or the height of the input or output picture
                                        //is invalid.
#define HI_ERR_POINT_NULL        -3    //The input pointer is null.
#define HI_ERR_FIELD_FLAG        -4    //What is input not the field flag.
#define HI_ERR OSDNUM            -5    //The input OSD number is invalid.
```