



MJPEG PC Decoding Library Software

API Reference

Issue	01
Date	2008-11-30
BOM	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 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. 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



HISILICON, and other Hisilicon icons 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 Overview.....	1-1
1.1 Introduction	1-2
1.2 Function List.....	1-3
1.3 Function Description Fields	1-3
1.4 Data Structure Description Fields.....	1-4
2 API Functions.....	2-1
2.1 HiMJPEGDecCreate	2-2
2.2 HiMJPEGDecDestroy	2-3
2.3 HiMJPEGDecGetInfo	2-4
2.4 HiMJPEGDecFrame.....	2-7
3 Data Types and Data Structures.....	3-1
3.1 Common Data Types	3-2
3.2 Data Structures	3-2
3.2.1 MJPEG_USERDATA_S.....	3-2
3.2.2 MJPEG_LIBINFO_S	3-2
3.2.3 MJPEG_DEC_ATTR_S	3-4
3.2.4 MJPEG_DEC_FRAME_S	3-4
4 Application Instances.....	4-1
4.1 Working Flow of the M-JPEG Decoding	4-2
4.2 Application Instances.....	4-2



Figures

Figure 4-1 Working flow of the M-JPEG decoding.....	4-2
--	-----



Tables

Table 1-1 SDK components of the decoding library	1-2
Table 1-2 Running environments of the decoding library	1-2



About This Document

Purpose

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

This document describes the API function types of the the MJPEG PC decoding library and their relations, describes all the reference information in details, and then describes how to use the APIs of the MJPEG PC decoding library through instances.

Related Versions

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

Product Name	Version
Hi3511 H.264 Encoding and Decoding Processor	V100
Hi3512 H.264 Encoding and Decoding Processor	V100

Intended Audience

The document describes types of reference information related to the HiSilicon H.264 PC decoding library. Therefore, 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

Organization

This document is organized as follows:








Chapter	Description
1 Overview	Describes the SDK components and software/hardware development environments of the MJPEG PC decoding library.
2 API Functions	Describes the API functions of the MJPEG PC decoding library..
3 Data Types and Data Structures	Describes the definitions of API common data types and data structures.
4 Application Instances	Describes how to use APIs of the MJPEG PC decoding library through instances.

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 .



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

Table Contents

Content	Description
-	Not applicable
*	A wild card

Update History

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

Updates in Issue 01 (2008-11-27)

Initial release.



1 Overview

About This Chapter

The following table lists the contents of this chapter.

Section	Describes
1.1 Introduction	The SDK components and the running environments of the MJPEG PC decoding library.
1.2 Function List	The functions of the decoding library.
1.3 Function Description Fields	The API function description fields and functions.
1.4 Data Structure Description Fields	The data structure description fields and functions.



1.1 Introduction

The M-JPEG PC decoding library provided by HiSilicon Technologies Co., Ltd. is a set of decoding software with high performance, reliability, and compatibility. The decoding library implements the M-JPEG decoding process internally and provides flexible APIs for users to develop applications quickly.

In Windows, the decoding library provides a dynamic link library and a static link library for calling API functions. This helps users easier develop applications. [Table 1-1](#) lists the key software development kit (SDK) components of the decoding library.

Table 1-1 SDK components of the decoding library

Component	Item	Description
API	hi_config.h hi_mjpeg_api.h	hi_config.h should be included in the user project first, and then hi_mjpeg_api.h is included.
Static link library	hi_mjpeg_dec_w.lib	When using the static link library, ignore the following four library files in the compiler option group: libm.lib, libguide.lib, libirc.lib, and svml_disp.lib. Otherwise, an alarm indicating an failed link is displayed during the compiling process.
Dynamic link library	hi_mjpeg_dec_w.lib hi_mjpeg_dec_w.dll	–
Demo code	hi_mjpeg_sample.c	Demonstrate how to call the decoding library APIs by taking decoding a local stream file as an example.

Users can develop applications based on the decoding library in multiple compiling environments. The decoding library is compatible with the Windows 2000 or the mainstream Windows operating systems of the later versions. The decoding library is also compatible with most PC-oriented CPU chipsets provided by Intel or AMD since 2002. [Table 1-2](#) lists the major development and running environments for 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.Net 2003 Intel C++ 9.0/10.0	Visual Studio.Net 2003	None.



Type	Compatible Configuration	Recommended Configuration	Description
Operating system	Windows 98 Windows 2000 Windows XP Windows 2003 Windows Vista	Windows XP	In Windows 98, the decoding library is in 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 is in decay mode, and the decoding performance is low.

1.2 Function List

Function	Description	Page
HiMJPEGDecCreate	Create and initialize a handle for a decoder.	2-2
HiMJPEGDecDestroy	Destroy the created handle.	2-3
HiMJPEGDecGetInfo	Query the version information of the decoding library and the function set of the current version.	2-3
HiMJPEGDecFrame	Decode a segment of input stream and output the current frame.	2-7

1.3 Function Description Fields

This document describes the API reference information in the following seven 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	Lists the return values of an API and describes each return value.



Field	Function
Request	Lists the required header files and library files when an API is called.
Note	Lists the precautions when an API is called.

1.4 Data Structure Description Fields

Field	Function
Description	Describes the functions implemented by a structure.
Definition	Lists the definition of a structure.
Note	Lists structure-related precautions.



2 API Functions

About This Chapter

The following table lists the contents of this chapter.

Section	Describes
2.1 HiMJPEGDecCreate	The HiMJPEGDecCreate function.
2.2 HiMJPEGDecDestroy	The HiMJPEGDecDestroy function.
2.3 HiMJPEGDecGetInfo	The HiMJPEGDecGetInfo function.
2.4 HiMJPEGDecFrame	The HiMJPEGDecFrame function.



2.1 HiMJPEGDecCreate

[Purpose]

Create and initialize a handle for a decoder.

[Syntax]

```
HI_HANDLE HiMJPEGDecCreate( MJPEG_DEC_ATTR_S *pDecAttr );
```

[Description]

Before decoding, the function performs the following operations:

- Allocate the decoding memory.
- Initialize the decoder-related variables and states.
- Set the attributes of the decoder, such as the maximum width and height of a picture.

To support multi-channel decoding, the upper-layer applications use multiple threads to create multiple decoders.

[Parameter]

Parameter	Member	Value Range	Input/Output	Description
pDecAttr	uPictureFormat	–	Input	Reserved.
	uStreamInType	–	Input	Reserved.
	uPicWidth	[16, 4096]	Input	Maximum picture width, in the unit of pixel. (The value is 2,048 by default when the width exceeds the value range.)
	uPicHeight	[16, 4096]	Input	Maximum picture height, in the unit of pixel. (The value is 2,048 by default when the height exceeds the value range.)
	uWorkMode	–	Input	Reserved.
	pUserData	–	Input	User data.
	uReserved	–	Input	Reserved.

[Return Value]



Return Value	Macro Definition	Description
0	NULL	The decoder fails to be created. (The memory fails to be allocated or errors occur during the parameter configuration.)
Non-zero	–	The decoder is successfully created. The return value is the decoder handle.

[Request]

- Header files: hi_config.h and hi_mjpeg_api.h.
- Library file: hi_mjpeg_dec_w.lib.

[Note]

None.

2.2 HiMJPEGDecDestroy

[Purpose]

Destroy a decoder.

[Syntax]

```
void HI_HiMJPEGDecDestroy( HI_HANDLE hDec );
```

[Description]

Destroy the decoder to release the allocated memory. When the decoding is finished, this function can be called to avoid the memory leakage.

[Parameter]

Parameter	Member	Value Range	Input/Output	Description
hDec	–	–	Input	The handle of a decoder to be destroyed.

[Return Value]

None.

[Request]

- Header files: hi_config.h and hi_mjpeg_api.h.
- Library file: hi_mjpeg_dec_w.lib.

[Note]

The destroyed handle must be set to NULL manually.



2.3 HiMJPEGDecGetInfo

[Purpose]

Query the version information of the decoding library and the function set of the current version.

[Syntax]

```
HI_S32 HiMJPEGDecGetInfo( MJPEG_LIBINFO_S *pLibInfo );
```

[Description]

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

[Parameter]

Parameter	Member	Value Range	Input/Output	Description
pLibInfo	uMajor	–	Output	Major serial number of the decoding library.
	uMinor	–	Output	Minor serial number of the decoding library.
	uRelease	–	Output	Release number of the decoding library.
	uBuild	–	Output	Build number of the decoding library.
	sVersion	–	Output	Version information of the decoding library.
	sCopyRight	–	Output	Copy right information of the decoding library.
	uPictureFormat	–	Output	<p>The value 1 indicates that the picture format is supported; the value 0 indicates that the picture format is not supported.</p> <p>The supported picture formats are as follows:</p> <p>bit[31:5]: reserved.</p> <p>bit[4]: Indicate whether the picture format YUV400 (black-and-white) is supported.</p> <p>bit[3]: Indicate whether the picture format YUV422 (MCU 1x2) is supported.</p>



Parameter	Member	Value Range	Input/Output	Description
				bit[2]: Indicate whether the picture format YUV444 is supported. bit[1]: Indicate whether the picture format YUV422 is supported. bit[0]: Indicate whether the picture format YUV420 is supported.
	uFrameMarkersSet	–	Output	Start of frame (SOF) markers supported by the decoding library. The value 1 indicates that the frame is supported by the decoding library; the value 0 indicates that the frame is not supported by the decoding library. bit[31:16]: reserved. bit[15]: differential lossless (sequential), arithmetic. bit[14]: differential progressive DCT, arithmetic. bit[13]: differential sequential DCT, arithmetic. bit[12]: reserved. bit[11]: lossless (sequential), arithmetic. bit[10]: progressive DCT, arithmetic. bit[9]: extended sequential DCT, arithmetic. bit[8]: JPG extensions. bit[7]: differential lossless (sequential), arithmetic. bit[6]: differential progressive DCT,



Parameter	Member	Value Range	Input/Output	Description
				huffman. bit[5]: differential sequential DCT, huffman. bit[4]: reserved. bit[3]: lossless (sequential), huffman. bit[2]: progressive DCT, huffman. bit[1]: extended sequential DCT, huffman. bit[0]: baseline sequential DCT, huffman.
	uStreamInType	0	Output	Reserved.
	uPicWidth	4096	Output	The maximum picture width of the docoding library (in the unit of pixel).
	uPicHeight	4096	Output	The maximum picture height of the docoding library (in the unit of pixel).
	uReserved	–	Output	Reserved.

[Return Value]

Return Value	Macro Definition	Description
0	–	The decoding library information is successful obtained.
–1	–	The decoding library information fails to be obtained because the input parameter is incorrect.

[Request]

- Header files: hi_config.h and hi_mjpeg_api.h.
- Library file: hi_mjpeg_dec_w.lib.

[Note]

None.



2.4 HiMJPEGDecFrame

[Purpose]

Decode the input stream containing only one JPEG frame and output this frame.

[Syntax]

```
HI_S32 HiMJPEGDecFrame(  
    HI_HANDLE          hDec,  
    HI_U8              *pStream,  
    HI_U32             iStreamLen,  
    HI_U64             ullPTS,  
    MJPEG_DEC_FRAME_S *pDecFrame,  
    HI_U32             uFlags  
);
```

[Description]

Each time the HiMJPEGDecFrame function is called, the configured stream should contain one JPEG picture only. In addition, the frame is output immediately by the decoder.

The HiMJPEGDecFrame provides the time stamp transparent transmission function. The input time stamp is saved in the current decoded picture structure MJPEG_DEC_FRAME_S and is output with the decoded picture. For details, see section [3.2.4 "MJPEG_DEC_FRAME_S."](#)

[Parameter]

Parameter	Member	Value Range	Input/Output	Description
hDec	–	–	Input	Decoder handle.
pStream	–	–	Input	Start address of the stream.
iStreamLen	–	–	Input	Stream length, in the unit of byte.
ullPTS	–	–	Input	Time stamp information
pDecFrame	pY	–	Output	Address of the output Y component.
	pU	–	Output	Address of the output U component.
	pV	–	Output	Address of the output V component.
	uYStride	–	Output	Luminance stride of the output picture, in the unit of pixel.
	uCStride	–	Output	Chrominance stride of the output picture, in the unit of pixel.
	uWidth	–	Output	Width of the output picture, in the unit of pixel.
	uHeight	–	Output	Height of the output picture, in the unit of pixel.



Parameter	Member	Value Range	Input/Output	Description
	uPictureFormat	[0, 5]	Output	Output picture format. 0: YUV420 1: YUV422 2: YUV444 3: YUV422 (MCU 1x2) 4: YUV400 5: The picture format that is not supported.
	bError	0, 1	Output	Indicate that whether the current picture has an error. 0: The output picture has no error. 1: The output picture has an error.
	ullPTS	–	Output	Time stamp information of the output picture.
	reserved	–	Output	Reserved.
	pUserData	–	Output	User data pointer.
uFlags	–	0	Input	Reserved.

[Return Value]

Return Value	Macro Definition	Description
0	HI_MJPEG_DEC_OK	The function is called successfully, and a frame is output.
–1	HI_MJPEG_NO_PICTURE	The decoding is interrupted due to the input stream error and no picture is output.
–2	HI_MJPEG_ERR_HANDLE	The input parameters of a function are incorrect.

[Request]

- Header files: hi_config.h and hi_mjpeg_api.h.
- Library file: hi_mjpeg_dec_w.lib.

[Note]

The decoding library only allows that the JPEG pictures are decoded by frame. Therefore, the input stream must only contain one JPEG frame when the 2.4 HiMJPEGDecFrame is called. If the input stream cannot form a frame, only a part of the picture is output. If the input stream forms multiple frames, only the first frame is decoded and output.



3 Data Types and Data Structures

About This Chapter

The following table lists the contents of this chapter.

Section	Describes
3.1 Common Data Types	The common data types.
3.2 Data Structures	The data structures.



3.1 Common Data Types

The major data types 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 long    HI_U32;  
typedef signed char      HI_S8;  
typedef signed short     HI_S16;  
typedef signed long      HI_S32;  
typedef __int64          HI_S64;  
typedef unsigned __int64 HI_U64;  
typedef char             HI_CHAR;  
typedef char*            HI_PCHAR;  
typedef void*            HI_HANDLE;
```

3.2 Data Structures

3.2.1 MJPEG_USERDATA_S

[Description]

User data structure.

[Definition]

```
/*User data structure*/  
typedef struct hiMJPEG_USERDATA_S  
{  
    HI_U32  uUserDataTypes;    /*User data type */  
    HI_U32  uUserDataSize;    /*User data length */  
    HI_UCHAR* pData;          /*Pointer to the user data area */  
    struct hiMJPEG_USERDATA_S* pNext; /*Pointer to the next segment of user  
data*/  
} MJPEG_USERDATA_S;
```

[Note]

None.

3.2.2 MJPEG_LIBINFO_S

[Description]

Data structure of the version, copyright, and function set information of the decoding library.

[Definition]



```
/* Data structure of the version, copyright, and function set information of
the decoding library */
typedef struct hiMJPEG_LIBINFO_S
{
    HI_U32  uMajor;           /* Major serial number of the decoding library
*/
    HI_U32  uMinor;          /* Minor serial number of the decoding library
*/
    HI_U32  uRelease;         /* Release number of the decoding library */
    HI_U32  uBuild;           /* Build number of the decoding library */
    const HI_CHAR*  sVersion; /* Version information of the decoding library
*/
    const HI_CHAR*  sCopyRight; /* Copyright information of the decoding
library */

    HI_U32  uPictureFormat;    /* Picture format */
    /* bit0: YUV420 */
    /* bit1: YUV422 */
    /* bit2: YUV444 */
    /* bit3: YUV422 (MCU 1x2) */
    /* bit4: YUV400 */
    /* bit5-bit31: Reserved */

    HI_U32  uFrameMarkersSet; /* Frame tag set or SOF marker set */
    /* bit0: SOF0 baseline sequential DCT, huffman */
    /* bit1: SOF1 extended sequential DCT, huffman */
    /* bit2: SOF2 progressive DCT, huffman */
    /* bit3: SOF3 lossless sequential DCT, huffman */
    /* bit4: Reserved */
    /* bit5: SOF5 differential sequential DCT, huffman */
    /* bit6: SOF6 differential progressive DCT, huffman */
    /* bit7: SOF7 differential lossless (sequential), huffman */
    /* bit8: JPG JPG extensions */
    /* bit9: SOF9 extended sequential DCT, arithmetic */
    /* bit10: SOF10 progressive DCT, arithmetic */
    /* bit11: SOF11 lossless (sequential), arithmetic */
    /* bit12: Reserved */
    /* bit13: SOF13 differential sequential DCT, arithmetic */
    /* bit14: SOF14 differential progressive DCT, arithmetic */
    /* bit15: SOF15 differential lossless (sequential), arithmetic */
    /* bit16 ~ bit31 reserved */

    HI_U32  uStreamInType;     /* Reserved */
    HI_U32  uPicWidth;         /* Maximum picture width (in the unit of pixel)
```



```

*/
    HI_U32  uPicHeight;          /* Maximum picture height (in the unit of pixel)
*/
    HI_U32  uReserved;          /* Reserved */
} MJPEG_LIBINFO_S;

```

[Note]

None.

3.2.3 MJPEG_DEC_ATTR_S

[Description]

Data structure of the decoder attribute information.

[Definition]

```

/* Data structure of the decoder attribute information */
typedef struct hiMJPEG_DEC_ATTR_S
{
    HI_U32  uPictureFormat;      /* Reserved */
    HI_U32  uStreamInType;      /* Reserved */
    HI_U32  uPicWidth;          /* Maximum picture width (in the unit of
pixel) */
    HI_U32  uPicHeight;         /* Maximum picture height (in the unit of
pixel) */
    HI_U32  uWorkMode;          /* Reserved */
    MJPEG_USERDATA_S *pUserData; /* User data */
    HI_U32  uReserved;          /* Reserved */
} MJPEG_DEC_ATTR_S;

```

[Note]

None.

3.2.4 MJPEG_DEC_FRAME_S

[Description]

Data structure of the output picture information of the decoder.

[Definition]

```

/* Data structure of the output picture information of the decoder */
typedef struct hiMJPEG_DEC_FRAME_S
{
    HI_U8 *pY;                  /* Pixel Y pointer */
    HI_U8 *pU;                  /* Pixel U pointer */
    HI_U8 *pV;                  /* Pixel V pointer */
    HI_U32 uYStride;            /* Luminance stride (in the unit of pixel)
*/
}

```




```
        HI_U32 uCStride;                /* Chrominance stride (in the unit of
pixel) */
        HI_U32 uWidth;                  /* Picture width (in the unit of pixel)*
/
        HI_U32 uHeight;                 /* Picture height (in the unit of pixel)*
/
        HI_U32 uPictureFormat;          /* Picture format */
/* 0: YUV420; */
/* 1: YUV422; */
/* 2: YUV444; */
/* 3: YUV422 (MCU 1x2); */
/* 4: YUV400; */
/* >=5: reserved */

        HI_S32 bError;                  /* Error flag */
/* 0: no error */
/* 1: MCU error */
        HI_U64 ullPTS;                  /* Time stamp */
        HI_U32 reserved;                /* Reserved */
        MJPEG_USERDATA_S *pUserData;    /* User data pointer */
    } MJPEG_DEC_FRAME_S;
```

[Note]

None.



4 Application Instances

About This Chapter

The following table lists the contents of this chapter.

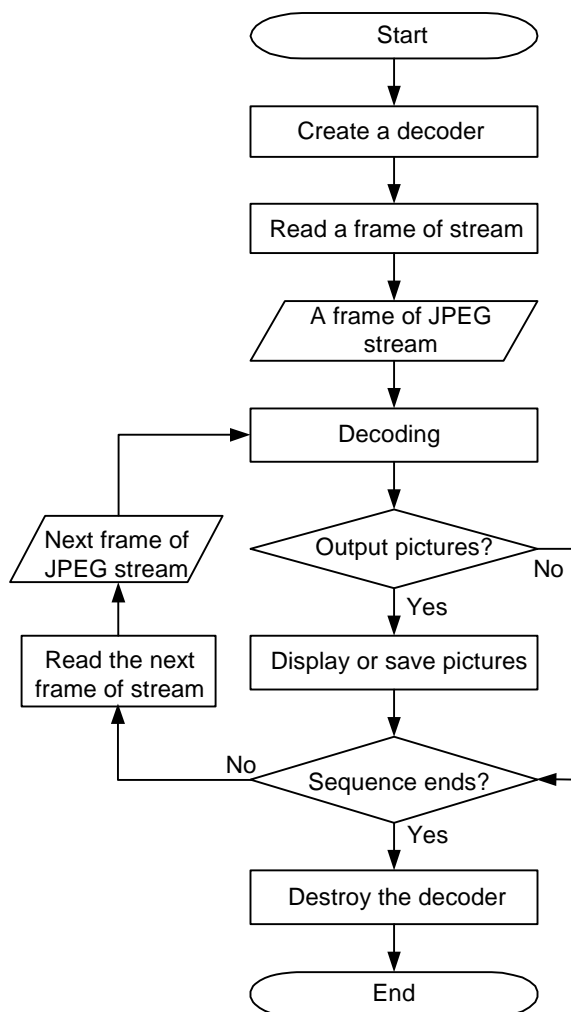
Section	Describes
4.1 Working Flow of the M-JPEG Decoding	The working flow of the M-JPEG decoding.
4.2 Application Instances	The API application instances.



4.1 Working Flow of the M-JPEG Decoding

Figure 4-1 shows the working flow of the M-JPEG decoding.

Figure 4-1 Working flow of the M-JPEG decoding



4.2 Application Instances

```

/* Static constant character strings, available for the intuitive picture
format */
static const char *PictureFormatString[6] = {
    "YUV420", "YUV422", "YUV444", "YUV422 (MCU 1x2)", "YUV400",
    "not support"
};

```



```
MJPEG_DEC_ATTR_S    dec_attribute; /* Create parameter structures for the
decoder */
MJPEG_DEC_FRAME_S    dec_frame;    /* Output picture structures */
HI_S32 len, result;
HI_U8 *bitstream = NULL;           /*Stream buffer area*/
HI_HANDLE handle = NULL;
FILE *jpeg = NULL;                 /* JPEG stream file */
FILE *yuv = NULL;                  /* File used to store the YUV picture */

/* Allocate memory for the stream buffer area. The allocated memory must be
greater than the size of a frame of picture */
bitstream = malloc(BUFF_LEN);

/* Create a decoder */
dec_attribute.uPicWidth = WIDTH;    /* Maximum picture width (in the unit of
pixel) */
dec_attribute.uPicHeight = HEIGHT;  /* Maximum picture height (in the unit
of pixel) */
handle = HiMJPEGDecCreate(&dec_attribute);

/* Open the JPEG stream file and the file used to store the YUV picture */
jpeg = fopen(argv[1], "rb");
yuv = fopen(argv[2], "wb");
if(NULL == jpeg || NULL == yuv || NULL == bitstream || NULL == handle)
{
    goto END;
}

/* Read a frame of JPEG stream data from the JPEG stream file */
len = fread(bitstream, 1, BUFF_LEN, jpeg);

/* Decoding process:
    Return HI_JPEG_DEC_OK          : The decoding is successful and a picture is
output;
    Return HI_JPEG_NO_PICTURE     : The decoding fails and no picture is output;
    Return HI_JPEG_ERR_HANDLE     : The input parameters are incorrect.
*/
result = HiMJPEGDecFrame(handle, bitstream, len, 0, &dec_frame, 0);

if( HI_JPEG_DEC_OK == result && dec_frame.uPictureFormat <= 4)
{
    HI_U32 yStride = dec_frame.uYStride;
    HI_U32 cStride = dec_frame.uCStride;
```



```

HI_U32 yHeight = dec_frame.uHeight;
HI_U32 cHeight;

switch(dec_frame.uPictureFormat )
{
case 0: /* YUV420 */
    cHeight = (yHeight + 1) / 2;
    break;
case 1: /* YUV422 */
    cHeight = yHeight;
    break;
case 2: /* YUV444 */
    cHeight = yHeight;
    break;
case 3: /* YUV422 (MCU 1x2) */
    cHeight = (yHeight + 1) / 2;
    break;
default: /* YUV400 */
    cHeight = 0;
    break;
}

/* If the decoding is successful, the information about the picture format,
picture width, and picture height can be obtained, and a picture is output
or displayed */
printf("picture format: %s. width: %d, height: %d.\n",
PictureFormatString[dec_frame.uPictureFormat], dec_frame.uWidth,
dec_frame.uHeight);
fwrite(dec_frame.pY, 1, yStride * yHeight, yuv);
fwrite(dec_frame.pU, 1, cStride * cHeight, yuv);
fwrite(dec_frame.pV, 1, cStride * cHeight, yuv);
}

END:
/* Release the stream buffer area */
if(NULL != bitstream)
    free(bitstream);

/* Close the input JPEG stream file and output the YUV picture */
if(NULL != jpeg)
    fclose(jpeg);
if(NULL != yuv)
    fclose(yuv);

/* Destroy the decoder */

```



```
if (NULL != handle)
    HiMJPEGDecDestroy(handle);
handle = NULL;
```