



MJPEG PC 解码库软件

API 参考

文档版本 01

发布日期 2011-07-12

版权所有 © 深圳市海思半导体有限公司 2011。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

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

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市海思半导体有限公司

地址： 深圳市龙岗区坂田华为基地华为电气生产中心 邮编：518129

网址： <http://www.hisilicon.com>

客户服务电话： +86-755-28788858

客户服务传真： +86-755-28357515

客户服务邮箱： support@hisilicon.com



前 言

概述

本参考首先介绍 MJPEG PC 解码库 API 函数种类及其关联，再分别详细介绍各种参考信息，最后通过实例介绍 MJPEG PC 解码库 API 的使用方法。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3507 芯片	V100



读者对象

本文档（本指南）主要适用于以下工程师：




- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
 警告	表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。



符号	说明
 注意	表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 窍门	表示能帮助您解决某个问题或节省您的时间。
 说明	表示是正文的附加信息，是对正文的强调和补充。

数值单位约定

数据容量、频率、数据速率等的表达方式说明如下。

类别	符号	对应的数值
数据容量（如 RAM 容量）	1K	1024
	1M	1,048,576
	1G	1,073,741,824
频率、数据速率等	1k	1000
	1M	1,000,000
	1G	1,000,000,000

地址、数据的表达方式说明如下。

符号	举例	说明
0x	0xFE04、0x18	用 16 进制表示的数据值、地址值。
0b	0b000、0b00 00000000	表示 2 进制的数据值以及 2 进制序列（寄存器描述中除外）。
X	00X、1XX	在数据的表达方式中，X 表示 0 或 1。 例如：00X 表示 000 或 001； 1XX 表示 100、101、110 或 111。



修订记录

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

文档版本 01 (2011-06-16)

第 1 次正式发布。



目 录

前 言.....	i
1 概述.....	1
1.1 概述.....	1
1.2 函数列表.....	2
1.3 函数描述方式.....	2
1.4 结构体描述方式.....	3
2 API 函数	4
2.1 HiMJPEGDecCreate	4
2.2 HiMJPEGDecDestroy	5
2.3 HiMJPEGDecGetInfo	6
2.4 HiMJPEGDecFrame	9
3 数据类型与数据结构	13
3.1 通用数据类型描述.....	13
3.2 数据结构描述.....	13
3.2.1 MJPEG_USERDATA_S	13
3.2.2 MJPEG_LIBINFO_S	14
3.2.3 MJPEG_DEC_ATTR_S.....	15
3.2.4 MJPEG_DEC_FRAME_S	15
4 应用实例.....	17
4.1 MJPEG 解码流程图	17
4.2 程序实例.....	18



插图目录

图 4-1 解码库 API 函数使用流程图	18
-----------------------------	----



表格目录

表 1-1 解码库开发包组件..... 1

表 1-2 解码库运行环境 2



1 概述

1.1 概述

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

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

表1-1 解码库开发包组件

组件	名称	说明
API 接口	hi_config.h hi_mjpeg_api.h	用户工程中，应该保证先包含 hi_config.h，再包含 hi_mjpeg_api.h。
静态库	hi_mjpeg_dec_w.lib	使用静态库时，应该在编译器选项中选择忽略下面的四个库文件：libm.lib、libguide.lib、libirc.lib 和 svml_disp.lib，否则编译时会出现链接不成功告警。
动态库	hi_mjpeg_dec_w.lib hi_mjpeg_dec_w.dll	-
示范代码	hi_mjpeg_sample.c	以读文件解码为例，示范解码库 API 的调用方式。

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



表1-2 解码库运行环境

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

1.2 函数列表

函数	功能	页码
HiMJPEGDecCreate	创建、初始化解码器。	4
HiMJPEGDecDestroy	销毁解码器。	5
HiMJPEGDecGetInfo	查询解码库版本信息和当前版本能力集。	6
HiMJPEGDecFrame	对输入的一帧码流进行解码并输出当前帧。	9

1.3 函数描述方式

本文档用七个域对 API 参考信息进行描述。

参数域	作用
目的	简要描述 API 的主要功能。
语法	列出 API 的语法样式。



参数域	作用
描述	简要描述 API 的工作过程。
参数	列出 API 的参数、参数说明及参数属性。
返回值	列出 API 的返回值及返回值说明。
需求	列出本 API 要包含的头文件和 API 编译时要链接的库文件。
注意	使用 API 时应注意的事项。

1.4 结构体描述方式

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



2 API 函数

2.1 HiMJPEGDecCreate

【目的】

创建、初始化解码器。

【语法】

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

【描述】

创建解码器。在解码开始时，分配解码空间和初始化解码器相关的变量及状态，设置解码器支持的最大图像的宽高等属性。

上层应用可以使用多线程创建多个解码器，实现多路解码。

【参数】

参数	成员	取值范围	输入/输出	描述
pDecAttr	uPictureFormat	-	输入	保留。
	uStreamInType	-	输入	保留。
	uPicWidth	[16, 4096]	输入	解码器对象支持的最大图像宽度。 (以像素为单位。超出取值范围时解码器自动默认为 2048。)
	uPicHeight	[16, 4096]	输入	解码器支持的最大图像高度。 (以像素为单位。超出取值范围时解码器自动默认为 2048。)
	uWorkMode	-	输入	保留。
	pUserData	-	输入	用户数据。



参数	成员	取值范围	输入/输出	描述
	uReserved	-	输入	保留。

【返回值】

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

【需求】

- 头文件：hi_config.h、hi_mjpeg_api.h
- 库文件：hi_mjpeg_dec_w.lib

【注意】

无。

2.2 HiMJPEGDecDestroy

【目的】

销毁解码器。

【语法】

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

【描述】

销毁解码器，释放工作时分配的内存空间。解码结束后调用此函数以防止内存泄漏。

【参数】

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

【返回值】

无。

【需求】



- 头文件：hi_config.h、hi_mjpeg_api.h
- 库文件：hi_mjpeg_dec_w.lib

【注意】

销毁后的句柄应该手动置为 NULL。

2.3 HiMJPEGDecGetInfo

【目的】

查询解码库版本信息和当前版本能力集。

【语法】

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

【描述】

用户可在创建解码器之前调用此函数查看解码库版本信息、解码库能力集。

【参数】

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



参数	成员	取值范围	输入/输出	描述
	uPictureFormat	-	输出	图像格式信息，含义如下： bit[31:5]：保留。 bit[4]：YUV400（黑白）图像格式支持标示。 0 为不支持； 1 为支持。 bit[3]：YUV422 (MCU 1x2)图像格式支持标示。 0 为不支持； 1 为支持。 bit[2]：YUV444 图像格式支持标示。 0 为不支持； 1 为支持。 bit[1]：YUV422 图像格式支持标示。 0 为不支持； 1 为支持。 bit[0]：YUV420 图像格式支持标示。 0 为不支持； 1 为支持。



参数	成员	取值范围	输入/输出	描述
	uFrameMarkersSet	-	输出	<p>解码库当前支持的帧标记。每 bit 代表一种帧，为 1 表示解码库支持的帧，为 0 表示解码库不支持的帧。</p> <p>bit[31:16]: 保留。</p> <p>bit[15]: 差分无损（顺序），算术编码。</p> <p>bit[14]: 差分渐进 DCT，算术编码。</p> <p>bit[13]: 差分顺序 DCT，算术编码。</p> <p>bit[12]: 保留。</p> <p>bit[11]: 无损（顺序），算术编码。</p> <p>bit[10]: 渐进 DCT，算术编码。</p> <p>bit[9]: 扩展顺序 DCT，算术编码。</p> <p>bit[8]: JPG 扩展。</p> <p>bit[7]: 差分无损（顺序），霍夫曼编码。</p> <p>bit[6]: 差分渐进 DCT，霍夫曼编码。</p> <p>bit[5]: 差分顺序 DCT，霍夫曼编码。</p> <p>bit[4]: 保留。</p> <p>bit[3]: 无损（顺序），霍夫曼编码。</p> <p>bit[2]: 渐进 DCT，霍夫曼编码。</p> <p>bit[1]: 扩展顺序 DCT，霍夫曼编码。</p> <p>bit[0]: 基本顺序 DCT，霍夫曼编码。</p>
	uStreamInType	0	输出	保留。
	uPicWidth	4096	输出	解码库当前支持的最大图像宽度（以像素为单位）。
	uPicHeight	4096	输出	解码库当前支持的最大图像高度（以像素为单位）。



参数	成员	取值范围	输入/输出	描述
	uReserved	-	输出	保留。

【返回值】

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

【需求】

- 头文件：hi_config.h、hi_mjpeg_api.h
- 库文件：hi_mjpeg_dec_w.lib

【注意】

无。

2.4 HiMJPEGDecFrame

【目的】

对输入的仅包含一帧 JPEG 的码流进行解码并立即输出当前帧。

【语法】

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

【描述】

本函数仅支持按帧配置码流和按帧输出图像，解码器默认为每次配置的码流仅包含一帧 JPEG 图像并且在解码之后立即输出此帧图像。

本函数提供时间戳透传功能，输入的时间戳将保存在当前码流解码后的图像结构体 MJPEG_DEC_FRAME_S 中，并随解码图像一起输出。详细信息请参见“[3.2.4 MJPEG_DEC_FRAME_S](#)”。



【参数】

参数	成员	取值范围	输入/输出	描述
hDec	-	-	输入	解码器句柄。
pStream	-	-	输入	码流起始地址。
iStreamLen	-	-	输入	码流长度（以字节为单位）。
ullPTS	-	-	输入	时间戳信息。
pDecFrame	pY	-	输出	输出 Y 分量地址。
	pU	-	输出	输出 U 分量地址。
	pV	-	输出	输出 V 分量地址。
	uYStride	-	输出	输出图像亮度跨度（以像素为单位）。
	uCStride	-	输出	输出图像色度跨度（以像素为单位）。
	uWidth	-	输出	输出图像宽（以像素为单位）。
	uHeight	-	输出	输出图像高（以像素为单位）。
	uPictureFormat	[0, 5]	输出	输出图像格式。 0: YUV420; 1: YUV422; 2: YUV444; 3: YUV422 (MCU 1x2); 4: YUV400; 5: 不支持的图像格式。
	bError	0, 1	输出	当前图像错误标示。 0: 输出图像无错; 1: 输出图像有错。
	ullPTS	-	输出	输出图像时间戳信息。
uFlags	reserved	-	输出	保留。
	pUserData	-	输出	用户数据指针。
uFlags	-	0	输入	保留。

【返回值】



返回值	宏定义	含义
0	HI_MJPEG_DEC_OK	函数执行成功，输出一帧图像。
-1	HI_MJPEG_NO_PICTURE	输入码流错误，解码中断，此时没有图像输出。
-2	HI_MJPEG_ERR_HANDLE	函数输入参数错误。

【需求】

- 头文件：hi_config.h、hi_mjpeg_api.h
- 库文件：hi_mjpeg_dec_w.lib

【注意】

解码库只支持以帧为单位的 JPEG 图像解码，所以每次调用时输入的码流必须包含且仅包含一帧 JPEG 图像。若输入的码流不足一帧图像，则只能输出部分图像；若输入的码流超过一帧图像，则只能解码并输出第一帧图像。



3 数据类型与数据结构

3.1 通用数据类型描述

在 win32 环境下，API 用到的主要数据类型定义如下：

```
typedef unsigned char    HI_U8;  
typedef unsigned char    HI_UCHAR;  
typedef unsigned short   HI_U16;  
typedef unsigned 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 数据结构描述

3.2.1 MJPEG_USERDATA_S

【说明】

用户数据结构。

【定义】

```
/* 用户数据结构 */  
typedef struct hiMJPEG_USERDATA_S  
{  
    HI_U32  uUserDataTyep;    /* 用户数据类型 */  
    HI_U32  uUserDataSize;    /* 用户数据长度 */  
};
```



```
HI_UCHAR* pData;          /* 用户数据缓冲区 */  
struct hiMJPEG_USERDATA_S* pNext; /* 指针, 指向下一个用户数据 */  
} MJPEG_USERDATA_S;
```

【注意事项】

无。

3.2.2 MJPEG_LIBINFO_S

【说明】

解码库版本、版权和能力集信息数据结构。

【定义】

```
/* 解码库版本、版权和能力集信息数据结构 */  
typedef struct hiMJPEG_LIBINFO_S  
{  
    HI_U32  uMajor;          /* 解码库主编号 */  
    HI_U32  uMinor;          /* 解码库次编号 */  
    HI_U32  uRelease;        /* 解码库发布编号 */  
    HI_U32  uBuild;          /* 解码库构建编号 */  
    const HI_CHAR*  sVersion; /* 解码库版本信息 */  
    const HI_CHAR*  sCopyRight; /* 解码库版权信息 */  
  
    HI_U32  uPictureFormat; /* 图像格式 */  
    /* bit0: YUV420 */  
    /* bit1: YUV422 */  
    /* bit2: YUV444 */  
    /* bit3: YUV422 (MCU 1x2) */  
    /* bit4: YUV400 */  
    /* bit5~bit31: 保留 */  
  
    HI_U32  uFrameMarkersSet; /* 帧标记集 */  
    /* bit0: SOF0 基本顺序DCT, 霍夫曼编码 */  
    /* bit1: SOF1 扩展顺序DCT, 霍夫曼编码 */  
    /* bit2: SOF2 渐进DCT, 霍夫曼编码 */  
    /* bit3: SOF3 无损(顺序), 霍夫曼编码 */  
    /* bit4: 保留 */  
    /* bit5: SOF5 差分顺序DCT, 霍夫曼编码 */  
    /* bit6: SOF6 差分渐进DCT, 霍夫曼编码 */  
    /* bit7: SOF7 差分无损(顺序), 霍夫曼编码 */  
    /* bit8: JPG JPG扩展 */
```



```
/* bit9:  SOF9   扩展顺序DCT, 算术编码 */
/* bit10: SOF10  渐进DCT, 算术编码 */
/* bit11: SOF11  无损(顺序), 算术编码 */
/* bit12: 保留 */
/* bit13: SOF13  差分顺序DCT, 算术编码*/
/* bit14: SOF14  差分渐进DCT, 算术编码*/
/* bit15: SOF15  差分无损(顺序), 算术编码*/
/* bit16 ~ bit31 保留 */

HI_U32  uStreamInType;      /* 保留 */
HI_U32  uPicWidth;          /* 图像最大宽度(以像素为单位) */
HI_U32  uPicHeight;        /* 图像最大高度(以像素为单位) */
HI_U32  uReserved;         /* 保留 */
} MJPEG_LIBINFO_S;
```

【注意事项】

无。

3.2.3 MJPEG_DEC_ATTR_S

【说明】

解码器属性信息数据结构。

【定义】

```
/* 解码器属性数据结构 */
typedef struct hiMJPEG_DEC_ATTR_S
{
    HI_U32  uPictureFormat;      /* 保留 */
    HI_U32  uStreamInType;      /* 保留 */
    HI_U32  uPicWidth;          /* 图像最大宽度(以像素为单位) */
    HI_U32  uPicHeight;        /* 图像最大高度(以像素为单位) */
    HI_U32  uWorkMode;          /* 保留 */
    MJPEG_USERDATA_S  *pUserData; /* 用户数据 */
    HI_U32  uReserved;         /* 保留 */
} MJPEG_DEC_ATTR_S;
```

【注意事项】

无。

3.2.4 MJPEG_DEC_FRAME_S

【说明】



解码器输出图像信息数据结构。

【定义】

```
/* 解码器输出图像信息数据结构 */
typedef struct hiMJPEG_DEC_FRAME_S
{
    HI_U8 *pY;                /* Y像素指针 */
    HI_U8 *pU;                /* U像素指针 */
    HI_U8 *pV;                /* V像素指针 */
    HI_U32 uYStride;          /* 亮度跨度(以像素为单位) */
    HI_U32 uCStride;          /* 色度跨度(以像素为单位) */
    HI_U32 uWidth;            /* 图像宽度(以像素为单位) */
    HI_U32 uHeight;           /* 图像高度(以像素为单位) */
    HI_U32 uPictureFormat;    /* 图像格式 */
    /* 0: YUV420; */
    /* 1: YUV422; */
    /* 2: YUV444; */
    /* 3: YUV422 (MCU 1x2); */
    /* 4: YUV400; */
    /* >=5: reserved */

    HI_S32 bError;            /* 错误标识 */
    /* 0: 无错误 */
    /* 1: MCU错误 */
    HI_U64 ullPTS;            /* 时间戳 */
    HI_U32 reserved;          /* 保留 */
    MJPEG_USERDATA_S *pUserData; /* 用户数据指针 */
} MJPEG_DEC_FRAME_S;
```

【注意事项】

无。



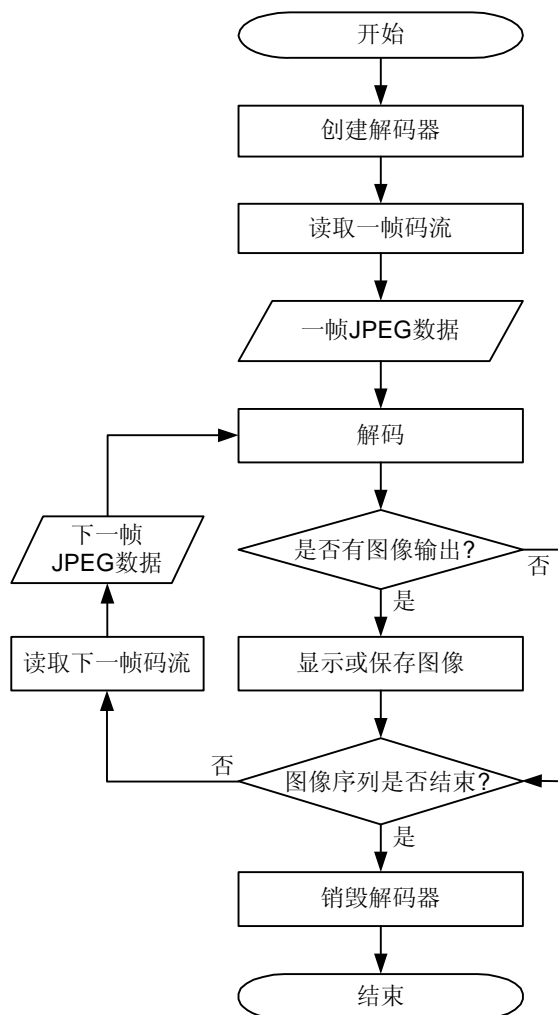
4 应用实例

4.1 MJPEG 解码流程图

MJPEG 解码流程如[图 4-1](#) 所示。



图4-1 解码库 API 函数使用流程图



4.2 程序实例

```
/* 静态常量字符串，用于比较直观的显示图像格式 */
static const char *PictureFormatString[6] = {
    "YUV420", "YUV422", "YUV444", "YUV422 (MCU 1x2)", "YUV400",
    "not support"
};

MJPEG_DEC_ATTR_S   dec_attribute; /* 创建解码器参数结构体 */
MJPEG_DEC_FRAME_S  dec_frame;     /* 输出图像结构体 */
HI_S32 len, result;
HI_U8 *bitstream = NULL;          /* 码流缓冲区 */
HI_HANDLE handle = NULL;
```



```
FILE *jpeg = NULL;          /* JPEG 码流文件 */
FILE *yuv = NULL;           /* 存放YUV图像的文件 */

/* 为码流缓冲区分配内存，分配的内存大小必须保证大于一帧图像的大小 */
bitstream = malloc(BUFF_LEN);

/* 创建解码器 */
dec_attribute.uPicWidth = WIDTH;    /* 图像最大宽度(以像素为单位) */
dec_attribute.uPicHeight = HEIGHT;  /* 图像最大高度(以像素为单位) */
handle = HiMJPEGDecCreate(&dec_attribute);

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

/* 从文件中读取一帧JPEG码流数据 */
len = fread(bitstream, 1, BUFF_LEN, jpeg);

/* 解码过程:
    返回 HI_JPEG_DEC_OK      : 解码成功，可以输出图像；
    返回 HI_JPEG_NO_PICTURE : 解码失败，没有图像输出；
    返回 HI_JPEG_ERR_HANDLE : 输入参数有错。
*/
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;
}

/* 如果解码成功, 可以得到图像格式、宽高等信息, 可以输出或显示图像 */
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:

/* 释放码流缓冲区 */
if(NULL != bitstream)
    free(bitstream);

/* 关闭输入JPEG码流文件和输出YUV图像文件 */
if(NULL != jpeg)
    fclose(jpeg);
if(NULL != yuv)
    fclose(yuv);

/* 销毁解码器 */
if(NULL != handle)
    HiMJPEGDecDestroy(handle);
handle = NULL;
```