

**HW\_MSDK API Programming Guide**

Version: 0.1

Release date: 2011-12-12

© 2011 MediaTek Inc.

This document contains information that is proprietary to MediaTek Inc.

Unauthorized reproduction or disclosure of this information in whole or in part is strictly prohibited.

Specifications are subject to change without notice.

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Author** | **Description** |
| 0.1 | 2011-12-9 | Jialei Shi | Initial Draft |
|  |  |  |  |

Table of Contents

[Document Revision History 2](#_Toc311192692)

[Table of Contents 3](#_Toc311192693)

[1 Architecture 4](#_Toc311192694)

[2 API 定义 5](#_Toc311192695)

[3 API 使用方法 6](#_Toc311192696)

[4 Sample 及使用注意 7](#_Toc311192697)

1. Architecture

APP通过调用HW\_MSDK 模块的接口完成解码,并且获取解码后数据,通过Direct draw传递给MTK display模块完成显示.



图 1-1

1. API 定义

Video decoder部分接口及codec类型定义在HW\_Vdec.h

typedef enum \_VCODECID

{

VCODEC\_ID\_UNKNOWN, //< unknow type, used for debug

VCODEC\_ID\_MP2V, //< mpeg 1/2

VCODEC\_ID\_MP4V, //< mpeg 4

VCODEC\_ID\_DIVX311, //< Divx 3.11

VCODEC\_ID\_DIVX4, //< Divx 4

VCODEC\_ID\_DIVX6, //< Divx 5/6

VCODEC\_ID\_WMV1, //< WMV7

VCODEC\_ID\_WMV2, //< WMV8

VCODEC\_ID\_WMV3, //< WMV9

VCODEC\_ID\_VC1, //< VC1

VCODEC\_ID\_H263, //< H.263

VCODEC\_ID\_SH263, //< H.263 Sorenson version

VCODEC\_ID\_RV30, //< Real Video 8

VCODEC\_ID\_RV40, //< Real Video 9,10

VCODEC\_ID\_H264 //< H.264

} VCODECID\_T;

typedef struct

{

GVOID \*pvInputBuff;//存放数据的起始地址

GUINT32 u4DataSize; //buffer大小

GUINT32 u4Height; //当前帧所代表的图像的高

GUINT32 u4Width; //当前帧所代表的图像的宽

GUINT32 u4PicType; //I B P帧等

GUCHAR ucPicStruct; //帧类型，场类型(上半场，下半场)

GUINT64 u8PTS; //此帧显示时间（-1 video decoder 做调整）

GUINT32 u4Duration; //此帧显示时长

GUINT32 u4BitRate; //比特率

GUINT32 u4FFRate; //快进使用标志，、、分别代码正常解码，倍速倍速的解码

GBOOL fgSeek; //是否为seek后的帧

}VDECINFO\_INPUT\_T;

typedef struct

{

GVOID \*pvOutputBuff;//输出buffer

GUINT32 u4OutBufSize; //buffer大小

GUINT32 u4Height; //解码后得到宽

GUINT32 u4Width; //解码后得到高

GUINT32 u8Pts; //decoder 调整后的pts

GBOOL fgDisplay; //此输出帧是否可以显示，TRUE：此解码后的帧可以给display 显示（h264用）

GBOOL fgNoNeedData; //下一帧是否需要输入解码数据（h264用）

}VDECINFO\_OUTPUT\_T;

//创建vdec 实例

**HVDECINST Vdec\_CreateInstance(VCODECID\_T eCodecId);**

//传递需要解析的数据,

//参数1:[in]创建的vdec实例,

//参数2:[in]解码信息，VDECINFO\_INPUT\_T参加上述定义

**GBOOL Vdec\_SetInputData(HVDECINST hVdecInst,** **VDECINFO\_INPUT\_T\* prInputInfo);**

//获取解码后的数据

//参数1:[in]创建的vdec实例,

//参数2:[out]获取解码后数据以及控制信息

**GBOOL Vdec\_GetOutputData(HVDECINST hVdecInst,VDECINFO\_OUTPUT\_T\* prOutputInfo);**

//释放输出buff:解码后的数据存放buff必须为连续物理内存供displayer使用,video会在底层自己分配buff保存解//码后数据,app在使用后(每一帧显示后)必须显示的调用接口来释放,否则会导致内存不足.

//参数1:[in]创建的vdec实例,

//参数2:[in]需要释放的buffer, VDECINFO\_OUTPUT\_T中的pvOutputBuff

**GBOOL Vdec\_FreeOutputBuf(HVDECINST hVdecInst,GVOID\* pvOutputBuf);**

//解码结束后instance释放

**GBOOL Vdec\_ReleaseInstance(HVDECINST hVdecInst);**

1. API 使用方法

下图所示调用相应API完成的解码过程。



图3-1

Steep：

[1] 首先根据codec创建解码实例

[2]设置解码数据，需要告之原始数据的地址、大小和此帧数据类型

[3]获取解码后的数据，

[4]释放保存解码后数据的buffer（此buffer为底层分配的连续物理地址，使用后 需要显示释放）

[5]继续下一帧解码循环步骤2和3

[6]结束解码释放资源

1. Sample 及使用注意
2. 关于此API的使用可以参照同目录下sample code包括解码和显示部分
3. 解码后的数据必须使用MTK提供的display，即使用DirectDraw创建oversurface的pixformattype必须为私有格式，定义如下（详细使用请参见sample code）：

static DDPIXELFORMAT ddpfOverlayFormats[] = {

{sizeof(DDPIXELFORMAT), DDPF\_FOURCC, MAKEFOURCC('M','T','K','1'), 0, 0, 0, 0, 0}

};

ddsd.ddpfPixelFormat = ddpfOverlayFormats[0];

3、h264部分解码需要注意并非每一帧输入都会有得到输出，而是一帧输入后会等到一段时间后才会有输出，因此帧是否需要显示，紧接着的下一帧是否需要输入可通过VDECINFO\_OUTPUT\_T 中 fgDisplay; //此输出帧是否可以显示，TRUE：此解码后的帧可以给display 显示（h264用）

fgNoNeedData; //下一帧是否需要输入解码数据（h264用）

来判断。