MI VDEC API

Version 2.07

REVISION HISTORY

Revision No.	Description	Date
2.03	• Initial release	04/12/2018
2.04	• Added MI_VDEC_SetOutputPortAttr and MI_VDEC_GetOutputPortAttr	08/01/2019
2.05	• Added MI_VDEC_SetOutputPortLayoutMode and MI_VDEC_GetOutputPortLayoutMode	10/23/2019
2.06	• 新增 api 支持动态切换 lowlatency 模式	12/09/2019
2.07	• 新增 api 支持清理解码通道缓存数据	12/30/2019

TABLE OF CONTENTS

REVISION HISTORY ..... **i**
TABLE OF CONTENTS ..... **ii**
1. 概述 **1**
1.1. 模块简介 1
1.2. 解码流程图 1
1.2.1 621 数据流程图 1
1.2.2 201/202 数据流程图 2
1.3. 重要概念 2
2. API 参考 **4**
2.1. MI_VDEC_InitDev 5
2.2. MI_VDEC_DeInitDev 6
2.3. MI_VDEC_CreateChn 8
2.4. MI_VDEC_DestroyChn 10
2.5. MI_VDEC_GetChnAttr 11
2.6. MI_VDEC_StartChn 12
2.7. MI_VDEC_StopChn 13
2.8. MI_VDEC_GetChnStat 14
2.9. MI_VDEC_FlushChn 15
2.10. MI_VDEC_ResetChn 16
2.11. MI_VDEC_SetChnParam 17
2.12. MI_VDEC_GetChnParam 17
2.13. MI_VDEC_SendStream 18
2.14. MI_VDEC_GetUserData 20
2.15. MI_VDEC_ReleaseUserData 21
2.16. MI_VDEC_SetDisplayMode 22
2.17. MI_VDEC_GetDisplayMode 23
2.18. MI_VDEC_SetOutputPortAttr 24
2.19. MI_VDEC_GetOutputPortAttr 25
2.20. MI_VDEC_SetOutputPortLayoutMode 26
2.21. MI_VDEC_GetOutputPortLayoutMode 27
3. 数据类型 **29**
3.1. MI_VDEC_CodecType_e 30
3.2. MI_VDEC_DPB_BufMode_e 30
3.3. MI_VDEC_JpegFormat_e 31
3.4. MI_VDEC_VideoMode_e 31
3.5. MI_VDEC_ErrCode_e 32
3.6. MI_VDEC_DecodeMode_e 32
3.7. MI_VDEC_OutputOrder_e 33
3.8. MI_VDEC_VideoFormat_e 34
3.9. MI_VDEC_DisplayMode_e 34
3.10. MI_VDEC_OutbufLayoutMode_e 35
3.11. MI_VDEC_InitParam_t 35
3.12. MI_VDEC_ChnAttr_t 36

3.13. MI_VDEC_JpegAttr_t 37
3.14. MI_VDEC_VideoAttr_t 37
3.15. MI_VDEC_ChnStat_t 38
3.16. MI_VDEC_ChnParam_t.....38
3.17. MI_VDEC_VideoStream_t 39
3.18. MI_VDEC_UserData_t 40
3.19. MI_VDEC_OutputPortAttr_t 40
4. 错误码 42

1. 概述

1.1. 模块简介

视频解码功能参考，提供解码通道创建、码流传送及控制、输出图像缩放等功能。

VDEC 模块的输入源主要是两类：

- 用户读取码流文件向解码模块发送数据；
- 用户将网络接收到的码流数据直接发送到解码模块。

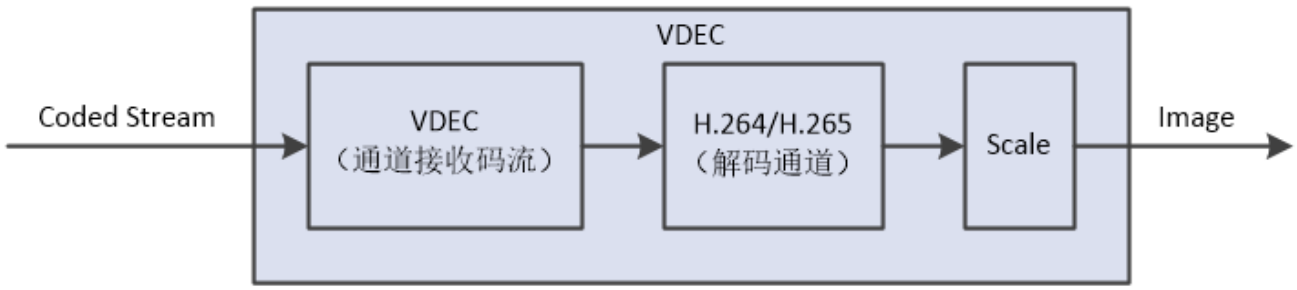
表 1-1 芯片解码规格

芯片	硬件解码模块	支持最大通道数	支持协议	分辨率范围	最大性能
621	DEC2.0	16	H.264/H.265	H.264: max 8192x8192, min 32x32 H.265: max 8192x8192, min 8x8	H.264: 3840x2160@30fps H.265: 3840x2160@30fps
201/202	DEC2.0	4	H.264/H.265	H.264: max 8192x8192, min 32x32 H.265: max 8192x8192, min 8x8	H.264: 1920x1080@30fps H.265: 1920x1080@30fps

1.2. 解码流程图

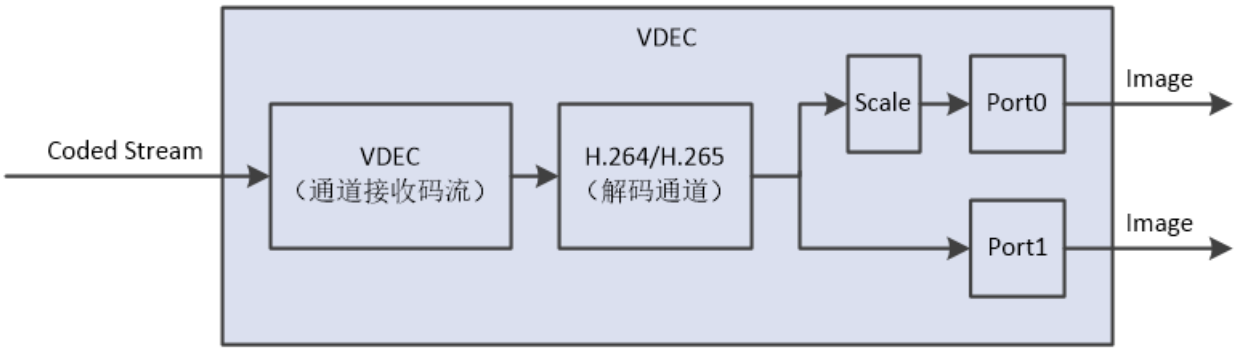
1.2.1 621 数据流程图

图 1-1 VDEC 数据流程图



1.2.2 201/202 数据流程图

图 1-2 VDEC 数据流程图



1.3. 重要概念

- 码流发送方式

  VDEC 解码器提供两种码流发送方式：

  - 按流发送（E_MI_VDEC_VIDEO_MODE_STREAM）：用户每次可发送任意长度码流到解码器，由解码器内部完成码流帧的解析和拼接过程。对于 H.264/H.265，需在收到下一帧码流才能确认为当前码流帧的结束，所以在该模式下，输入一帧 H.264/H.265 码流，不会立刻解出图像。
  - 按帧发送（E_MI_VDEC_VIDEO_MODE_FRAME）：用户每次发送完整的一帧码流到解码器，每调用一次发送接口，解码器就认为该帧码流已经结束，并开始解码图像。因此，用户需保证每次发送的码流必须为完整的一帧，否则会出现解码错误。使用该模式可以达到快速解码的目的。

  码流发送方式 eVideoMode 可在接口 MI_VDEC_CreateChn 中设置。

- 图像输出顺序

  根据 H.264/H265 协议，解码图像可能不会在解码后立即输出。VDEC 解码器可以通过设置不同的图像输出方式达到尽快输出的目的。图像输出方式包括以下两种：

  - 解码序：解码图像按照解码的先后顺序输出。
  - 显示序：解码图像按照协议中的显示顺序输出。

  根据 H.264/H.265 协议，视频的解码顺序（即解码序）和解码图像的显示顺序（即显示序）可能不一致。例如：解码 B 帧时，需要前后的 P 帧作为参考，所以 B 帧后的 P 帧先于 B 帧解码，但 B 帧先于 P 帧输出。按解码序输出是保证快速输出的一个必要条件，用户选择按解码序输出，需保证码流的解码序与显示序相同。

  按帧发送码流与按解码序输出相结合能达到快速解码和快速输出的目的，用户必须保证每次发送的是完整的一帧码流以及码流的解码序和显示序相同。

  图像输出方式 eOutputOrder 可在接口 MI_VDEC_SetChnParam 中设置。

- 时间戳（PTS）处理

  在按帧模式发送码流时，解码输出的图像时间戳 PTS 为发送码流接口(MI_VDEC_SendStream)中用户送入的 PTS，解码器不会更改此值；如果用户配置的 PTS 值为 0，则表示用户不进行帧率控制，而是由视频显示模块（DISP）进行帧率控制；如果用户送入的 PTS 值为-1，则表示此图像不会被视频显示模块（DISP）显示；如果是其他值，则表示视频显示模块（DISP）根据用户设置的 PTS 值进行帧率控制。

  注意：不能出现 PTS 值为 0 和非 0 混合的情况。

  在按流模式下发送码流时，解码输出图像的 PTS 统一设为 0，表示用户不进行帧率控制，而是由视频显示模块（DISP）进行帧率控制。

- 码流 Buffer
  码流 Buffer 用于缓存用户输入的码流，然后送入解码器解码。码流 buffer 的大小没有固定的计算公式，一般按照经验值，分辨率小于或等于 D1,建议配置为 512KB；分辨率在(D1,1080P],建议配置为 1MB；分辨率在(1080P，4K]，建议配置为 2MB；用户可根据实际需求，选择合理的值设置。
- eDpbBufMode
  eDpbBufMode 可供用户设置不同的 buffer 模式来进行解码。根据不同的场景，来选择不同的 buffer 模式，可以达到省内存的目的。当设为 E_MI_VDEC_DPB_MODE_NORMAL，不能节省内存；当设为 E_MI_VDEC_DPB_MODE_INPLACE_ONE_BUF，用于解码仅包含一个参考帧的码流类型，此时仅需一个 DPB buffer 即可；当设为 E_MI_VDEC_DPB_MODE_INPLACE_TWO_BUF，用于解码仅包含两个参考帧的码流类型，此时仅需两个 DPB buffer 即可。
- u32RefFrameNum
  u32RefFrameNum 表示最大参考帧个数。由于在实际的场景中，系统的内存并不是无限大，用户可以根据产品的定义对参考帧的个数进行限制，以便提示用户当前码流可能超规格。如果用户设置了 eDpbBufMode 为 E_MI_VDEC_DPB_MODE_INPLACE_ONE_BUF 或者 E_MI_VDEC_DPB_MODE_INPLACE_TWO_BUF，则该参数无效。如果用户设置了 eDpbBufMode 为 E_MI_VDEC_DPB_MODE_NORMAL，则解码器在参考帧个数受到限制时进行解码，解码图像可能会异常或者解码超时。
- 输出图像缩放
  用户调用 MI_VDEC_SetOutputPortAttr 可以对输出的图像进行缩放，以输出需要的分辨率的图像。但需注意，解码器仅支持对图像缩小，不支持对图像放大。
- 低延时
  这里的延时是指解码 B 帧后，需要对待显示的图像队列进行重排序（reorder）所产生的延时。当解码不含 B 帧的码流时，解码顺序和显示顺序相同，因此不会出现延时输出，即低延时；而解码含 B 帧的码流时，由于 B 帧有前后向参考，需要等待所参考的图像解码完后，才能解当前帧，并且需要对解码后的图像重排序（reorder），由此产生输出延时。这两种模式需要通过设置 bDisableLowLatency 来完成。默认 bDisableLowLatency 为 FALSE，用户无需设置；当需要解码 B 帧码流时，需要将 bDisableLowLatency 设为 TRUE。需要注意，bDisableLowLatency 为 TRUE 时，不能用于解码非 B 帧码流。
- 输出 buffer 模式
  用户可以调用 MI_VDEC_SetOutputPortLayoutMode 设置输出 buffer 的模式。该接口仅使用于芯片 201/202。当用户需要手动控制输出 buffer 的模式时，可调用该接口。当为 E_MI_VDEC_OUTBUF_LAYOUT_LINEAR 模式时，表示输出 buffer 为线性模式；当为 E_MI_VDEC_OUTBUF_LAYOUT_TILE 模式时，表示输出 buffer 为 TILE 模式；默认为 E_MI_VDEC_OUTBUF_LAYOUT_AUTO 模式，表示输出 buffer 会自动在线性模式和 TILE 模式间切换。

2. API 参考

该功能模块提供以下 API:

API名	功能
MI_VDEC_InitDev	设备初始化
MI_VDEC_DeInitDev	设备去初始化
MI_VDEC_CreateChn	创建解码通道
MI_VDEC_DestroyChn	销毁解码通道
MI_VDEC_GetChnAttr	获取解码通道属性
MI_VDEC_StartChn	解码通道开始接收码流
MI_VDEC_StopChn	解码通道停止接收码流
MI_VDEC_GetChnStat	获取解码通道状态
MI_VDEC_FlushChn	清理解码通道缓存数据
MI_VDEC_ResetChn	重置解码通道
MI_VDEC_SetChnParam	设置解码通道参数
MI_VDEC_GetChnParam	获取解码通道参数
MI_VDEC_SendStream	向解码通道发送码流数据
MI_VDEC_GetUserData	获取解码通道的用户数据
MI_VDEC_ReleaseUserData	释放解码通道的用户数据
MI_VDEC_SetDisplayMode	设置解码通道显示模式
MI_VDEC_GetDisplayMode	获取解码通道显示模式
MI_VDEC_SetOutputPortAttr	设置解码通道输出端口属性
MI_VDEC_GetOutputPortAttr	获取解码通道输出端口属性
MI_VDEC_SetOutputPortLayoutMode	设置输出端口Buffer模式
MI_VDEC_GetOutputPortLayoutMode	获取输出端口Buffer模式

2.1. MI_VDEC_InitDev

➢ 功能

解码设备初始化。

➢ 定义

MI_S32 MI_VDEC_InitDev(MI_VDEC_InitParam_t *pstVdecInitParam);

➢ 形参

参数名称	描述	输入/输出
pstVdecInitParam	解码设备初始化参数指针，数据类型：MI_VDEC_InitParam_t。	输入

➢ 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_FAILED	失败。
MI_ERR_VDEC_INITED	设备已经初始化。
MI_ERR_VDEC_NULL_PTR	输入参数为空指针

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 该接口为可选调用，如果仅使用默认的初始化参数，则无需调用；
- 若选择使用该接口，则必须在创建解码通道前调用，否则会返回设备已初始化的错误码MI_ERR_VDEC_INITED；
- 若重复调用该接口，则会返回设备已初始化的错误码MI_ERR_VDEC_INITED。

➢ 举例

```
MI_S32 StartVdec(void)
{
    MI_S32 s32Ret = MI_ERR_VDEC_FAILED;
    MI_VDEC_CHN VdecChn = 0;
    MI_VDEC_InitParam_t stVdecInitParam;
    MI_VDEC_ChnAttr_t stChnAttr;

    memset(&stVdecInitParam, 0x0, sizeof(MI_VDEC_InitParam_t));
    memset(&stChnAttr, 0x0, sizeof(MI_VDEC_ChnAttr_t));

    stVdecInitParam.bDisableLowLatency = FALSE;
    s32Ret = MI_VDEC_InitDev(&stVdecInitParam);
```

```
    if(MI_SUCCESS != s32Ret)
    {
        printf("MI_VDEC_InitDev failed, s32Ret: 0x%x.\n", s32Ret);
        return s32Ret;
    }

    VdecChn = 0;
    stChnAttr.eCodecType      = E_MI_VDEC_CODEC_TYPE_H264;
    stChnAttr.u32PicWidth     = 1920;
    stChnAttr.u32PicHeight    = 1080;
    stChnAttr.eVideoMode      = E_MI_VDEC_VIDEO_MODE_FRAME;
    stChnAttr.u32BufSize      = 1024*1024;
    stChnAttr.eDpbBufMode     = E_MI_VDEC_DPB_MODE_INPLACE_ONE_BUF;
    stChnAttr.stVdecVideoAttr.u32RefFrameNum = 1;
    stChnAttr.u32Priority = 0;

    s32Ret = MI_VDEC_CreateChn(VdecChn, &stChnAttr);
    if(MI_SUCCESS != s32Ret)
    {
        printf("MI_VDEC_CreateChn failed, s32Ret: 0x%x.\n", s32Ret);
        return s32Ret;
    }

    return MI_SUCCESS;
}
```

- 相关主题
  无

2.2. MI_VDEC_DeInitDev

- 功能
  解码设备去初始化。

- 定义
  MI_S32 MI_VDEC_DeInitDev(void);

- 形参
  无

➢ 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_FAILED	失败。
MI_ERR_VDEC_NOT_INIT	设备未初始化。

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 该接口为可选调用；
- 若选择使用该接口，则必须在销毁解码通道后调用，否则会返回设备未初始化的错误码MI_ERR_VDEC_NOT_INIT；
- 若重复调用该接口，则会返回设备未初始化的错误码MI_ERR_VDEC_NOT_INIT。

➢ 举例

```
MI_S32 StopVdec(void)
{
    MI_S32 s32Ret = MI_ERR_VDEC_FAILED;
    MI_VDEC_CHN VdecChn = 0;

    s32Ret = MI_VDEC_DestroyChn(VdecChn);
    if(MI_SUCCESS != s32Ret)
    {
        printf("MI_VDEC_DestroyChn failed, s32Ret: 0x%x.\n", s32Ret);
        return s32Ret;
    }

    //Confirm that you have destroyed all channels
    s32Ret = MI_VDEC_DeInitDev();
    if(MI_SUCCESS != s32Ret)
    {
        printf("MI_VDEC_DeInitDev failed, s32Ret: 0x%x.\n", s32Ret);
        return s32Ret;
    }

    return MI_SUCCESS;
}
```

➢ 相关主题

无

2.3. MI_VDEC_CreateChn

➢ 功能

创建视频解码通道。

➢ 定义

MI_S32 MI_VDEC_CreateChn(MI_VDEC_CHN VdecChn, MI_VDEC_ChnAttr_t *pstChnAttr)

➢ 形参

参数名称	描述	输入/输出
VdecChn	解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入
pstChnAttr	解码通道属性指针，数据类型：MI_VDEC_ChnAttr_t。	输入

➢ 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_FAILED	失败。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围，取值范围：[0, MI_VDEC_MAX_CHN_NUM)。
MI_ERR_VDEC_NOT_INIT	模块没有成功加载。
MI_ERR_VDEC_ILLEGAL_PARAM	非法参数或输入参数超过通道解码能力。
MI_ERR_VDEC_CHN_EXIST	试图创建已经存在的通道。
MI_ERR_VDEC_NOMEM	分配内存失败（如系统内存不足）。
MI_ERR_VDEC_NULL_PTR	输入参数为空指针
MI_ERR_VDEC_NOT_SUPPORT	该操作或者功能不支持

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 通道号不能超出最大的通道号范围。

➢ 举例

```
MI_S32 StartVdec(void)
{
    MI_S32 s32Ret = MI_ERR_VDEC_FAILED;
    MI_VDEC_CHN VdecChn = 0;
    MI_VDEC_DisplayMode_e eDisplayMode = E_MI_VDEC_DISPLAY_MODE_MAX;
    MI_VDEC_ChnAttr_t stChnAttr;
    MI_VDEC_OutputPortAttr_t stOutputPortAttr;

    memset(&stChnAttr, 0x0, sizeof(MI_VDEC_ChnAttr_t));
    memset(&stOutputPortAttr, 0x0, sizeof(MI_VDEC_OutputPortAttr_t));

    VdecChn = 0;
    stChnAttr.eCodecType      = E_MI_VDEC_CODEC_TYPE_H264;
    stChnAttr.u32PicWidth     = 1920;
    stChnAttr.u32PicHeight    = 1080;
    stChnAttr.eVideoMode      = E_MI_VDEC_VIDEO_MODE_FRAME;
    stChnAttr.u32BufSize      = 1024*1024;
    stChnAttr.eDpbBufMode     = E_MI_VDEC_DPB_MODE_INPLACE_ONE_BUF;
    stChnAttr.stVdecVideoAttr.u32RefFrameNum = 1;
    stChnAttr.u32Priority = 0;

    s32Ret = MI_VDEC_CreateChn(VdecChn, &stChnAttr);
    if(MI_SUCCESS != s32Ret)
    {
        printf("MI_VDEC_CreateChn failed, s32Ret: 0x%x.\n", s32Ret);
        return s32Ret;
    }

    eDisplayMode = E_MI_VDEC_DISPLAY_MODE_PLAYBACK;
    s32Ret = MI_VDEC_SetDisplayMode (VdecChn, eDisplayMode);
    if(MI_SUCCESS != s32Ret)
    {
        printf("MI_VDEC_SetDisplayMode failed, s32Ret: 0x%x.\n", s32Ret);
        return s32Ret;
    }

    s32Ret = MI_VDEC_StartChn(VdecChn);
    if(MI_SUCCESS != s32Ret)
    {
        printf("MI_VDEC_StartChn failed, s32Ret: 0x%x.\n", s32Ret);
        return s32Ret;
    }
```

```
    stOutputPortAttr.u16Width  = 720;
    stOutputPortAttr.u16Height = 576;
    s32Ret = MI_VDEC_SetOutputPortAttr(VdecChn, &stOutputPortAttr);
    if(MI_SUCCESS != s32Ret)
    {
        printf("MI_VDEC_SetOutputPortAttr failed, s32Ret: 0x%x.\n", s32Ret);
        return s32Ret;
    }

    return MI_SUCCESS;
}
```

➢ 相关主题

无

2.4. MI_VDEC_DestroyChn

➢ 功能

销毁视频解码通道。

➢ 定义

MI_S32 MI_VDEC_DestroyChn(MI_VDEC_CHN VdecChn);

➢ 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入

➢ 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_CHN_NOT_STOP	停止接收码流前不允许关闭通道。
MI_ERR_VDEC_CHN_UNEXIST	通道不存在。
MI_ERR_VDEC_NOT_INIT	模块没有成功加载
MI_ERR_VDEC_NOT_SUPPORT	该操作或者功能不支持

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 销毁前需停止接收码流。

➢ 举例

```
MI_S32 StopVdec(void)
{
    MI_S32 s32Ret = MI_ERR_VDEC_FAILED;
    MI_VDEC_CHN VdecChn = 0;

    // Destroy send stream thread
    ...

    s32Ret = MI_VDEC_StopChn(VdecChn);
    if(MI_SUCCESS != s32Ret)
    {
        printf("MI_VDEC_StopChn failed, s32Ret: 0x%x.\n", s32Ret);
        return s32Ret;
    }

    s32Ret = MI_VDEC_DestroyChn(VdecChn);
    if(MI_SUCCESS != s32Ret)
    {
        printf("MI_VDEC_DestroyChn failed, s32Ret: 0x%x.\n", s32Ret);
        return s32Ret;
    }

    return MI_SUCCESS;
}
```

➢ 相关主题

无。

2.5. MI_VDEC_GetChnAttr

➢ 功能

获取视频解码通道属性。

➢ 定义

MI_S32 MI_VDEC_GetChnAttr(MI_VDEC_CHN VdecChn, MI_VDEC_ChnAttr_t*pstChnAttr);

➢ 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入
pstChnAttr	解码通道属性指针。参数类型：MI_VDEC_ChnAttr_t。	输出

➢ 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_ILLEGAL_PARAM	非法参数。
MI_ERR_VDEC_CHN_UNEXIST	通道不存在。
MI_ERR_VDEC_NOT_INIT	模块没有成功加载

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

无。

➢ 举例

无。

➢ 相关主题

无。

2.6. MI_VDEC_StartChn

➢ 功能

开启解码器。

➢ 定义

MI_S32 MI_VDEC_StartChn(MI_VDEC_CHNVdecChn);

➢ 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入

➢ 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_CHN_UNEXIST	通道未创建或者已销毁。
MI_ERR_VDEC_NOT_INIT	模块没有加载。
MI_ERR_VDEC_NOT_PERM	非法操作，如在进行此操作前必须先禁止插入用户图片。

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 启动接收码流前必须保证通道已创建，否则会返回通道未创建的错误码 MI_ERR_VDEC_CHN_UNEXIST。
- 启动接收码流前必须保证已经禁止使能用户图片，否则返回该操作不允许的错误码 MI_ERR_VDEC_NOT_PERM。
- 启动接收码流之后，才能调用 MI_VDEC_SendStream 发送码流成功。
- 重复调用启动接收码流接口时，返回成功。

➢ 举例

请参见MI_VDEC_CreateChn的举例。

➢ 相关主题

无。

2.7. MI_VDEC_StopChn

➢ 功能

停止解码器。

➢ 语法

```
MI_S32 MI_VDEC_Stop(MI_VDEC_CHN VdecChn);
```

➢ 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入

➢ 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_FAILED	失败。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_CHN_NOT_START	通道未使能
MI_ERR_VDEC_CHN_UNEXIST	通道未创建或者已销毁。
MI_ERR_VDEC_NOT_INIT	系统没有初始化或者相关依赖的模块没有加载

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 调用此接口后，调用发送码流接口 MI_VDEC_SendStream 会返回失败。
- 重复调用停止接收码流接口时，返回MI_ERR_VDEC_CHN_NOT_START

➢ 举例

请参见MI_VDEC_DestroyChn的举例。

➢ 相关主题

无。

2.8. MI_VDEC_GetChnStat

➢ 功能

查询解码通道状态。

➢ 语法

MI_S32 MI_VDEC_GetChnStat (MI_VDEC_CHN VdecChn, MI_VDEC_ChnStat_t *pstChnStat);

➢ 形参

参数名称	描述	输入/输出
VdecChn	解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入
pstChnStat	解码通道状态结构体指针。	输出

- 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_FAILED	失败。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_CHN_UNEXIST	通道未创建或已销毁。
MI_ERR_VDEC_ILLEGAL_PARAM	参数指针为空。

- 依赖
  - 头文件：mi_vdec.h、mi_common.h
  - 库文件：libmi.a

※ 注意

无。

- 举例

请参见MI_VDEC_SendStream的举例。

- 相关主题

无。

2.9. MI_VDEC_FlushChn

- 功能

清理解码通道缓存数据。

- 定义

MI_S32 MI_VDEC_FlushChn(MI_VDEC_CHN VdecChn);

- 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入

- 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_FAILED	失败。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_CHN_UNEXIST	通道未创建或已销毁。
MI_ERR_VDEC_CHN_NOT_START	通道未使能。

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 该接口可用于解码过程中，切换到其他GOP时，清理解码通道缓存的数据，从而使解码继续进行。

➢ 举例

无。

➢ 相关主题

无。

2.10. MI_VDEC_ResetChn

➢ 功能

复位解码通道。

➢ 定义

MI_S32 MI_VDEC_ResetChn(MI_VDEC_CHN VdecChn);

➢ 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入

➢ 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_FAILED	失败。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_CHN_UNEXIST	通道未创建或已销毁。
MI_ERR_VDEC_NOT_PERM	无法重置通道，操作不能与 MI_VDEC_SendStream同时调用。

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 该接口暂不支持。

➢ 举例

无。

➢ 相关主题

无。

2.11. MI_VDEC_SetChnParam

➢ 功能

设置解码通道参数。

➢ 语法

MI_S32 MI_VDEC_SetChnParam(MI_VDEC_CHN VdecChn,MI_VDEC_ChnParam_t* pstChnParam);

➢ 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入
pstChnParam	通道参数。	输入

➢ 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_CHN_UNEXIST	通道未创建或已销毁。
MI_ERR_VDEC_ILLEGAL_PARAM	参数不合法。
MI_ERR_VDEC_NOT_PERM	非法操作，操作前需调用 MI_VDEC_StopChn停止通道解码。

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 该接口暂不支持。
- JPEG解码时，不支持调用该接口。

➢ 举例

无。

➢ 相关主题

无。

2.12. MI_VDEC_GetChnParam

➢ 功能

获取解码通道参数。

➢ 语法

MI_S32 MI_VDEC_GetChnParam(MI_VDEC_CHN VdecChn,MI_VDEC_ChnParam_t* pstChnParam);

➢ 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入
pstChnParam	通道参数。	输出

➢ 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_CHN_UNEXIST	通道未创建或已销毁。
MI_ERR_VDEC_ILLEGAL_PARAM	参数不合法。

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 该接口暂不支持。

➢ 举例

无。

➢ 相关主题

无。

2.13. MI_VDEC_SendStream

➢ 功能

向解码通道发送码流数据。

➢ 语法

MI_S32 MI_VDEC_SendStream(MI_VDEC_CHN VdecChn, MI_VDEC_VideoStream_t *pstVideoStream, MI_S32 s32MilliSec);

➢ 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入
pstVideoStream	解码码流数据指针。参考 MI_VDEC_VideoStream_t定义。	输入
s32MilliSec	设定推送数据超时时间参数。取值范围： -1：阻塞。 0：非阻塞。 正值：超时时间，单位为毫秒。	输入

➢ 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_ILLEGAL_PARAM	非法参数。
MI_ERR_VDEC_CHN_UNEXIST	通道未创建或者已销毁。
MI_ERR_VDEC_NOT_PERM	非法操作，如通道未准备好。
MI_ERR_VDEC_BUF_FULL	解码后的数据帧缓冲区满。
MI_ERR_VDEC_BUSY	解码前数据帧缓存区满。

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 支持按帧或按流传送数据。按帧传送时，每次需完整传送一帧数据。暂不支持按流发送。
- 一次发送的1帧数据的大小不能超过码流buffer大小的一半。

➢ 举例

每次传送完整一帧数据。以帧为单位，如果当前数据帧传送失败，需要重新传送。

```
MI_S32 VdecSendStream(void)
{
    MI_S32 s32Ret = MI_ERR_VDEC_FAILED;
    MI_VDEC_CHN VdecChn = 0;
    MI_S32 s32MilliSec = 0;
    MI_VDEC_VideoStream_t stVideoStream;
    MI_VDEC_ChnStat_t stChnStat;

    do{
        //Check if you need stop sending stream
        if(bStop)
        {
            break;
        }
```

```
        memset(&stChnStat, 0x0, sizeof(MI_VDEC_ChnStat_t));
        s32Ret = MI_VDEC_GetChnStat(VdecChn, &stChnStat);
        if(MI_SUCCESS != s32Ret)
        {
            printf("MI_VDEC_SendStream failed, s32Ret: 0x%x.\n", s32Ret);
            return s32Ret;
        }

        //suggest to check chn status
        if(stChnStat.u32LeftStreamBytes < u32StreamSize)
        {
            continue;
        }

        memset(&stVideoStream, 0x0, sizeof(MI_VDEC_VideoStream_t));
        stVideoStream.pu8Addr = pu8StreamBuf;
        stVideoStream.u32Len   = u32StreamSize;
        stVideoStream.u64PTS   = u64StreamPts;
        stVideoStream.bEndOfFrame   = TRUE;
        stVideoStream.bEndOfStream = FALSE;
        s32MilliSec = 30;                           //30ms
        s32Ret = MI_VDEC_SendStream(VdecChn, &stVideoStream, s32MilliSec);
        if(MI_SUCCESS != s32Ret)
        {
            printf("MI_VDEC_SendStream failed, s32Ret: 0x%x.\n", s32Ret);
            continue;
        }
    }while(!ShouldStop);

    return MI_SUCCESS;
}
```

- 相关主题
  无。

2.14. MI_VDEC_GetUserData

- 功能
  获取解码通道的用户数据。

- 语法
  MI_S32 MI_VDEC_GetUserData(MI_VDEC_CHN VdecChn,MI_VDEC_UserData_t *pstUserData, MI_S32 s32MilliSec);

➢ 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入
s32MilliSec	获取用户数据超时定义。取值范围： -1：阻塞。 0：非阻塞。 正值：推送数据超时时间，以 毫秒为单位。	输入
pstUserData	获取的解码用户数据，参考MI_VDEC_UserData_t定义	输出

➢ 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_ILLEGAL_PARAM	非法参数。
MI_ERR_VDEC_CHN_UNEXIST	通道未创建或者已销毁。
MI_ERR_VDEC_NOT_PERM	非法操作，如通道未准备好。
MI_ERR_VDEC_BUF_FULL	解码后的数据帧缓冲区满。

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 该接口暂不支持。
- 用户数据是 H.264/H.265 编码时，插入码流 SEI segment 的一段用户私有数据。
- JPEG 解码通道不支持获取用户数据。
- MI_VDEC_GetUserData 获取解码用户数据后，需要调用MI_VDEC_ReleaseUserData释放获取的数据。
- 如果获取用户数据不及时，会造成用户数据缓冲区满而出现丢弃用户数据的情况。

➢ 举例

无。

➢ 相关主题

无。

2.15. MI_VDEC_ReleaseUserData

➢ 功能

释放用户数据。

➢ 语法

MI_S32 MI_VDEC_ReleaseUserData(MI_VDEC_CHN VdecChn, MI_VDEC_UserData_t* pstUserData);

➢ 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入
pstUserData	解码后的用户数据指针，由 MI_VDEC_UserData_t 接口获取。	输入

➢ 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_ILLEGAL_PARAM	非法参数。
MI_ERR_VDEC_CHN_UNEXIST	通道未创建或者已销毁。
MI_ERR_VDEC_NOT_PERM	非法操作，如通道未准备好。

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 该接口暂不支持。
- 此接口需与 MI_VDEC_GetUserData 配对使用，获取的数据应当在使用完之后立即释放，如果不及时释放，会造成缓冲区满时丢弃用户数据的情况。

➢ 举例

无。

➢ 相关主题

无。

2.16. MI_VDEC_SetDisplayMode

➢ 功能

设置显示模式。

➢ 语法

MI_S32 MI_VDEC_SetDisplayMode(MI_VDEC_CHN VdecChn,MI_VDEC_DisplayMode_e eDisplayMode);

➢ 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入
eDisplayMode	显示模式枚举。参考MI_VDEC_DisplayMode_e定义。	输入

➢ 返回值

返回值	描述
MI_SUCCESS	成功。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_CHN_UNEXIST	通道未创建或已销毁。
MI_ERR_VDEC_ILLEGAL_PARAM	非法参数。

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

无。

➢ 举例

请参见MI_VDEC_CreateChn的举例。

➢ 相关主题

无。

2.17. MI_VDEC_GetDisplayMode

➢ 功能

获取显示模式。

➢ 语法

MI_S32 MI_VDEC_GetDisplayMode(MI_VDEC_CHN VdecChn,MI_VDEC_DisplayMode_e *peDisplayMode);

➢ 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入
peDisplayMode	显示模式枚举指针，参考MI_VDEC_DisplayMode_e定义。	输入

➢ 返回值

接口返回值	含义
MI_SUCCESS	成功。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_CHN_UNEXIST	通道未创建或已销毁。
MI_ERR_VDEC_ILLEGAL_PARAM	非法参数。

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

无。

➢ 举例

无。

➢ 相关主题

无。

2.18. MI_VDEC_SetOutputPortAttr

➢ 功能

设置解码通道输出端口属性。

➢ 语法

MI_S32 MI_VDEC_SetOutputPortAttr (MI_VDEC_CHN VdecChn, MI_VDEC_OutputPortAttr_t *pstOutputPortAttr);

➢ 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入
pstOutputPortAttr	输出端口属性，数据类型：MI_VDEC_OutputPortAttr_t	输入

➢ 返回值

接口返回值	含义
MI_SUCCESS	成功。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_CHN_UNEXIST	通道未创建或已销毁。
MI_ERR_VDEC_ILLEGAL_PARAM	非法参数。
MI_ERR_VDEC_NULL_PTR	输入参数空指针

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 若通道未创建，则返回错误码MI_ERR_VDEC_CHN_UNEXIST
- 缩放范围[1/8, 1]。
- 不支持放大。

➢ 举例

请参见MI_VDEC_CreateChn的举例。

➢ 相关主题

无。

2.19. MI_VDEC_GetOutputPortAttr

➢ 功能

获取解码通道输出端口属性。

➢ 语法

MI_S32 MI_VDEC_GetOutputPortAttr(MI_VDEC_CHN VdecChn, MI_VDEC_OutputPortAttr_t *pstOutputPortAttr);

➢ 形参

参数名称	描述	输入/输出
VdecChn	视频解码通道号。 取值范围：[0, MI_VDEC_MAX_CHN_NUM)。	输入
pstOutputPortAttr	输出端口属性，数据类型：MI_VDEC_OutputPortAttr_t	输出

➢ 返回值

接口返回值	含义
MI_SUCCESS	成功。
MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围。
MI_ERR_VDEC_CHN_UNEXIST	通道未创建或已销毁。
MI_ERR_VDEC_ILLEGAL_PARAM	非法参数。
MI_ERR_VDEC_NULL_PTR	输入参数空指针

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 若通道未创建，则返回错误码MI_ERR_VDEC_CHN_UNEXIST

➢ 举例

无。

➢ 相关主题

无。

2.20. MI_VDEC_SetOutputPortLayoutMode

➢ 功能

设置输出端口Buffer模式。

➢ 语法

MI_S32 MI_VDEC_SetOutputPortLayoutMode(MI_VDEC_OutbufLayoutMode_e eBufTileMode);

➢ 形参

参数名称	描述	输入/输出
eBufTileMode	输出Buffer模式，数据类型： MI_VDEC_OutbufLayoutMode_e	输入

➢ 返回值

接口返回值	含义
MI_SUCCESS	成功。
MI_ERR_VDEC_NOT_INIT	设备未初始化。
MI_ERR_VDEC_ILLEGAL_PARAM	非法参数。

➢ 依赖

- 头文件：mi_vdec.h、mi_common.h
- 库文件：libmi.a

※ 注意

- 若设备未初始化，则返回错误码MI_ERR_VDEC_NOT_INIT

➢ 举例

```
MI_S32 StartVdec(void)
{
    MI_S32 s32Ret = MI_ERR_VDEC_FAILED;
    MI_VDEC_CHN VdecChn = 0;
    MI_VDEC_OutbufLayoutMode_e eBufTileMode = E_MI_VDEC_OUTBUF_LAYOUT_MAX;
    MI_VDEC_ChnAttr_t stChnAttr;

    memset(&stChnAttr, 0x0, sizeof(MI_VDEC_ChnAttr_t));
```

```
    eBufTileMode = E_MI_VDEC_OUTBUF_LAYOUT_AUTO;
    s32Ret = MI_VDEC_SetOutputPortLayoutMode(eBufTileMode) ;
    if(MI_SUCCESS != s32Ret)
    {
        printf("MI_VDEC_SetOutputPortLayoutMode failed, s32Ret: 0x%x.\n", s32Ret);
        return s32Ret;
    }

    VdecChn = 0;
    stChnAttr.eCodecType      = E_MI_VDEC_CODEC_TYPE_H264;
    stChnAttr.u32PicWidth     = 1920;
    stChnAttr.u32PicHeight    = 1080;
    stChnAttr.eVideoMode      = E_MI_VDEC_VIDEO_MODE_FRAME;
    stChnAttr.u32BufSize      = 1024*1024;
    stChnAttr.eDpbBufMode     = E_MI_VDEC_DPB_MODE_INPLACE_ONE_BUF;
    stChnAttr.stVdecVideoAttr.u32RefFrameNum = 1;
    stChnAttr.u32Priority = 0;

    s32Ret = MI_VDEC_CreateChn(VdecChn, &stChnAttr);
    if(MI_SUCCESS != s32Ret)
    {
        printf("MI_VDEC_CreateChn failed, s32Ret: 0x%x.\n", s32Ret);
        return s32Ret;
    }

    return MI_SUCCESS;
}
```

- 相关主题
  无。

2.21. MI_VDEC_GetOutputPortLayoutMode

- 功能
  获取输出端口Buffer模式。

- 语法
  MI_S32 MI_VDEC_GetOutputPortLayoutMode(MI_VDEC_OutbufLayoutMode_e *peBufTileMode);

- 形参

参数名称	描述	输入/输出
eBufTileMode	输出Buffer模式，数据类型： MI_VDEC_OutbufLayoutMode_e	输入

- 返回值

接口返回值	含义
MI_SUCCESS	成功。
MI_ERR_VDEC_NOT_INIT	设备未初始化。
MI_ERR_VDEC_NULL_PTR	输入参数空指针

- 依赖
  - 头文件：mi_vdec.h、mi_common.h
  - 库文件：libmi.a

※ 注意
  - 若设备未初始化，则返回错误码MI_ERR_VDEC_NOT_INIT

- 举例

  无。

- 相关主题

  无。

3. 数据类型

视频解码相关数据类型、数据结构定义如下：

数据结构	说明
MI_VDEC_CodecType_e	定义解码类型
MI_VDEC_DPB_BufMode_e	定义Inplace内存模式
MI_VDEC_JpegFormat_e	定义Jpeg 图片数据格式
MI_VDEC_VideoMode_e	定义码流发送方式枚举
MI_VDEC_VideoFormat_e	定义解码图像数据格式枚举
MI_VDEC_DisplayMode_e	定义显示模式枚举
MI_VDEC_OutbufLayoutMode_e	定义输出buffer模式枚举
MI_VDEC_InitParam_t	定义解码设备初始化参数结构体
MI_VDEC_ChnAttr_t	定义视频解码通道属性
MI_VDEC_JpegAttr_t	定义JPEG视频解码通道属性结构体
MI_VDEC_VideoAttr_t	定义视频解码通道属性
MI_VDEC_ChnStat_t	定义通道状态结构体
MI_VDEC_ChnParam_t	定义解码通道参数结构体
MI_VDEC_VideoStream_t	定义推送码流结构体
MI_VDEC_UserData_t	定义用户数据结构体
MI_VDEC_OutputPortAttr_t	定义输出端口属性结构体

3.1. MI_VDEC_CodecType_e

➢ 说明

定义解码类型。

➢ 定义

```
typedef enum
{
    E_MI_VDEC_CODEC_TYPE_H264 = 0x0,        /* H264 */
    E_MI_VDEC_CODEC_TYPE_H265,              /* H265 */
    E_MI_VDEC_CODEC_TYPE_JPEG,              /* JPEG */
    E_MI_VDEC_CODEC_TYPE_MAX
} MI_VDEC_CodecType_e;
```

※ 注意事项

无。

➢ 相关数据类型及接口

无。

3.2. MI_VDEC_DPB_BufMode_e

➢ 说明

定义DPB buffer模式。

➢ 定义

```
typedef enum
{
    E_MI_VDEC_DPB_MODE_NORMAL=0,
    E_MI_VDEC_DBP_MODE_INPLACE_ONE_BUF=1,
    E_MI_VDEC_DBP_MODE_INPLACE_TWO_BUF=2,
    E_MI_VDEC_DPB_MODE_INVALID=0xFFFFFFFF
} MI_VDEC_DPB_BufMode_e;
```

➢ 成员

成员名称	描述
E_MI_VDEC_DPB_MODE_NORMAL	普通模式
E_MI_VDEC_DBP_MODE_INPLACE_ONE_BUF	Inplace one buffer模式。
E_MI_VDEC_DBP_MODE_INPLACE_TWO_BUF	Inplace two buffer模式

※ 注意事项

无。

➢ 相关数据类型及接口

无。

3.3. MI_VDEC_JpegFormat_e

➢ 说明

Jpeg 图片的存储格式。

➢ 定义

```
typedef enum
{
    E_MI_VDEC_JPEG_FORMAT_YCBCR400 = 0x0,  /*YUV400*/
    E_MI_VDEC_JPEG_FORMAT_YCBCR420, /*YUV420*/
    E_MI_VDEC_JPEG_FORMAT_YCBCR422, /*YUV 422*/
    E_MI_VDEC_JPEG_FORMAT_YCBCR444, /*YUV 444*/
    E_MI_VDEC_JPEG_FORMAT_MAX
} MI_VDEC_JpegFormat_e;
```

➢ 成员

略。

※ 注意事项

该参数暂不支持设置。

➢ 相关数据类型及接口

无。

3.4. MI_VDEC_VideoMode_e

➢ 说明

定义码流发送方式。

➢ 定义

```
typedef enum
{
    E_MI_VDEC_VIDEO_MODE_STREAM = 0x0,
    E_MI_VDEC_VIDEO_MODE_FRAME,
    E_MI_VDEC_VIDEO_MODE_MAX
} MI_VDEC_VideoMode_e;
```

➢ 成员

成员名称	描述
E_MI_VIDEO_MODE_STREAM	按流方式发送码流。
E_MI_VIDEO_MODE_FRAME	按帧方式发送码流，以帧为单位。

※ 注意事项

无。

➢ 相关数据类型及接口

无。

3.5. MI_VDEC_ErrCode_e

➢ 说明

定义码流发送方式。

➢ 定义

```
typedef enum
{
    E_MI_VDEC_ERR_CODE_UNKNOW = 0x0,
    E_MI_VDEC_ERR_CODE_ILLEGAL_ACCESS,
    E_MI_VDEC_ERR_CODE_FRMRATE_UNSUPPORT,
    E_MI_VDEC_ERR_CODE_DEC_TIMEOUT,
    E_MI_VDEC_ERR_CODE_OUT_OF_MEMORY,
    E_MI_VDEC_ERR_CODE_CODEC_TYPE_UNSUPPORT,
    E_MI_VDEC_ERR_CODE_ERR_SPS_UNSUPPORT,
    E_MI_VDEC_ERR_CODE_ERR_PPS_UNSUPPORT,
    E_MI_VDEC_ERR_CODE_REF_LIST_ERR,
    E_MI_VDEC_ERR_CODE_MAX
} MI_VDEC_ErrCode_e;
```

➢ 成员

成员名称	描述
E_MI_VDEC_ERR_CODE_UNKNOW	未知错误。
E_MI_VDEC_ERR_CODE_ILLEGAL_ACCESS	不可访问，例如:未初始化或硬件出问题。
E_MI_VDEC_ERR_CODE_FRMRATE_UNSUPPORT	帧率不支持
E_MI_VDEC_ERR_CODE_DEC_TIMEOUT	解帧超时
E_MI_VDEC_ERR_CODE_OUT_OF_MEMORY	内存不足
E_MI_VDEC_ERR_CODE_CODEC_TYPE_UNSUPPORT	解码类型不支持
E_MI_VDEC_ERR_CODE_ERR_SPS_UNSUPPORT	不支持的SPS或SPS出错
E_MI_VDEC_ERR_CODE_ERR_PPS_UNSUPPORT	不支持的PPS或PPS出错
E_MI_VDEC_ERR_CODE_REF_LIST_ERR	参考帧列表出错

※ 注意事项

无。

➢ 相关数据类型及接口

无。

3.6. MI_VDEC_DecodeMode_e

➢ 说明

定义码流解码方式。

➢ 定义

```
typedef enum
{
    E_MI_VDEC_DECODE_MODE_ALL = 0x0,
    E_MI_VDEC_DECODE_MODE_I,
    E_MI_VDEC_DECODE_MODE_IP,
    E_MI_VDEC_DECODE_MODE_MAX
} MI_VDEC_DecodeMode_e;
```

➢ 成员

成员名称	描述
E_MI_VDEC_DECODE_MODE_ALL	解码IPB数据帧模式
E_MI_VDEC_DECODE_MODE_I	只解I帧模式
E_MI_VDEC_DECODE_MODE_IP	只解IP帧，跳过B帧

※ 注意事项

无。

➢ 相关数据类型及接口

无。

3.7. MI_VDEC_OutputOrder_e

➢ 说明

定义码流发送方式。

➢ 定义

```
typedef enum
{
    E_MI_VDEC_OUTPUT_ORDER_DISPLAY = 0x0,
    E_MI_VDEC_OUTPUT_ORDER_DECODE,
    E_MI_VDEC_OUTPUT_ORDER_MAX,
} MI_VDEC_OutputOrder_e;
```

➢ 成员

成员名称	描述
E_MI_VDEC_OUTPUT_ORDER_DISPLAY	按显示顺序输出数据帧
E_MI_VDEC_OUTPUT_ORDER_DECODE	按解帧顺序列输出数据帧

➢ 数据类型

无。

➢ 相关数据类型及接口

无。

3.8. MI_VDEC_VideoFormat_e

➢ 说明

定义解码图像数据格式枚举。

➢ 定义

```
typedef enum
{
    E_MI_VDEC_VIDEO_FORMAT_TILE = 0x0,
    E_MI_VDEC_VIDEO_FORMAT_REDUCE,
    E_MI_VDEC_VIDEO_FORMAT_MAX
} MI_VDEC_VideoFormat_e;
```

➢ 成员

成员名称	描述
E_MI_VDEC_VIDEO_FORMAT_TILE	TILE数据格式。
E_MI_VDEC_VIDEO_FORMAT_REDUCE	数据帧压缩模式，减少数据帧内存使用量。

※ 注意事项

- 当前数据类型不支持上层设定，只能返回支持数据类型。

➢ 相关数据类型及接口

无。

3.9. MI_VDEC_DisplayMode_e

➢ 说明

定义显示模式枚举。

➢ 定义

```
typedef enum
{
    E_MI_VDEC_DISPLAY_MODE_PREVIEW = 0x0,
    E_MI_VDEC_DISPLAY_MODE_PLAYBACK,
    E_MI_VDEC_DISPLAY_MODE_MAX,
} MI_VDEC_DisplayMode_e;
```

➢ 成员

成员名称	描述
E_MI_DISPLAY_MODE_PREVIEW	预览模式。不参考PTS输出。
E_MI_DISPLAY_MODE_PLAYBACK	回放模式。参考PTS数值输出。

※ 注意事项

无。

➢ 相关数据类型及接口

无。

3.10. MI_VDEC_OutbufLayoutMode_e

➢ 说明

定义输出buffer模式枚举。

➢ 定义

```
typedef enum
{
    E_MI_VDEC_OUTBUF_LAYOUT_AUTO = 0x0,
    E_MI_VDEC_OUTBUF_LAYOUT_LINEAR,
    E_MI_VDEC_OUTBUF_LAYOUT_TILE,
    E_MI_VDEC_OUTBUF_LAYOUT_MAX
} MI_VDEC_OutbufLayoutMode_e;
```

➢ 成员

成员名称	描述
E_MI_VDEC_OUTBUF_LAYOUT_AUTO	输出buffer模式自适应。
E_MI_VDEC_OUTBUF_LAYOUT_LINEAR	输出buffer为LINEAR模式。
E_MI_VDEC_OUTBUF_LAYOUT_TILE	输出buffer为TILE模式

※ 注意事项

无。

➢ 相关数据类型及接口

无。

3.11. MI_VDEC_InitParam_t

➢ 说明

定义解码设备初始化参数结构体。

➢ 定义

```
typedef struct MI_VDEC_InitParam_s
{
    MI_BOOL bDisableLowLatency;
} MI_VDEC_InitParam_t;
```

➢ 成员

成员名称	描述
bDisableLowLatency	开启/关闭低延时模式，默认为FALSE。参数类型：MI_BOOL

※ 注意事项

- 若bDisableLowLatency设为TRUE，表示关闭低延时；这种情况适用于解码B帧，用于对B帧的输出做reorder，输出画面会有延时。另外，做reorder操作，也会增加内存使用量。
- 默认开启了低延时模式，即bDisableLowLatency为FALSE。如果场景不涉及延时模式，则无需调用MI_VDEC_InitDev。

➢ 相关数据类型及接口

无。

3.12. MI_VDEC_ChnAttr_t

➢ 说明

定义解码通道属性结构体。

➢ 定义

```
typedef struct MI_VDEC_ChnAttr_s
{
    MI_VDEC_CodecType_e eCodecType;
    MI_U32    u32BufSize
    MI_U32    u32Priority
    MI_U32    u32PicWidth
    MI_U32    u32PicHeight
    MI_VDEC_VideoMode_e eVideoMode
    MI_VDEC_DPB_BufMode_e eDpbBufMode;
    union
    {
        MI_VDEC_JpegAttr_t stVdecJpegAttr;
        MI_VDEC_VideoAttr_t stVdecVideoAttr;
    };
} MI_VDEC_ChnAttr_t;
```

➢ 成员

成员名称	描述
eCodecType	解码类型枚举值。参数类型：MI_VDEC_CodecType_e
u32BufSize	码流缓存的大小。
u32Priority	通道优先级，取值范围为 1 ~ 255，值越大优先级越高。 注意:当前暂不支持此功能。
u32PicWidth	通道支持的解码图像最大宽（以像素为单位）
u32PicHeight	通道支持的解码图像最大高（以像素为单位）
eVideoMode	码流传送方式。 注意：当前只支持按帧。
eDpbBufMode	DPB buffer 模式。 注意：eDpbBufMode=1，只适用于一个参考帧的码流； eDpbBufMode=2，适用于两个参考帧的码流；如果超过两个参考帧，则需要设置为eDpbBufMode=0
stVdecJpegAttr	JPEG 通道的相关属性，暂不支持设置。
stVdecVideoAttr	除 JPEG 以外其它所支持类型的通道的相关属性

※ 注意事项

- 当设置了eDpbBufMode为E_MI_VDEC_DBP_MODE_INPLACE_ONE_BUF或者E_MI_VDEC_DBP_MODE_INPLACE_TWO_BUF时，u32RefFrameNum的值将无效；
- 当eInplaceMode为E_MI_VDEC_DPB_MODE_NORMAL时，需要设置u32RefFrameNum的值，用于限制最大参考帧个数，避免分配过多的frame buffer；若u32RefFrameNum的值小于当前解码所需的参考帧个数，则可能不解码或者解码花屏。

➢ 相关数据类型及接口

无。

3.13. MI_VDEC_JpegAttr_t

➢ 说明

定义JPEG 解码通道属性。

➢ 定义

```
typedef struct MI_VDEC_JpegAttr_s
{
    MI_VDEC_JpegFormat_e eJpegFormat;
}MI_VDEC_JpegAttr_t;
```

➢ 成员

成员名称	描述
eJpegFormat	JPEG图片的存储格式。 取值范围是[0,E_MI_JPEG_FORMAT_MAX)。

※ 注意事项

暂不支持设置该参数。

➢ 相关数据类型及接口

无。

3.14. MI_VDEC_VideoAttr_t

➢ 说明

定义视频解码通道属性。

➢ 定义

```
typedef struct MI_VDEC_VideoAttr_s
{
    MI_U32 u32RefFrameNum;
}MI_VDEC_VideoAttr_t;
```

➢ 成员

成员名称	描述
u32RefFrameNum	参考帧的数目。 取值范围：[0, 16]，以帧为单位。 参考帧的数目决定解码时需要的参考帧个数，影响内存使用量，根据实际情况设置合适的值。测试码流：推荐设为 2。

※ 注意事项

- 当设置了eDpbBufMode为E_MI_VDEC_DBP_MODE_INPLACE_ONE_BUF或者E_MI_VDEC_DBP_MODE_INPLACE_TWO_BUF时，u32RefFrameNum的值将无效；
- 当eInplaceMode为E_MI_VDEC_DPB_MODE_NORMAL时，需要设置u32RefFrameNum的值，用于限制最大参考帧个数，避免分配过多的frame buffer；若u32RefFrameNum的值小于当前解码所需的参考帧个数，则可能不解码或者解码花屏。

➢ 相关数据类型及接口

无。

3.15. MI_VDEC_ChnStat_t

➢ 说明

定义通道状态结构体。

➢ 定义

```
typedef MI_VDEC_ChnStat_s
{
    MI_VDEC_CodecType_e eCodecType;
    MI_U32     u32LeftStreamBytes;
    MI_U32    u32LeftStreamFrames;
    MI_U32     u32LeftPics;
    MI_BOOL bChnStart;
    MI_U32 u32RecvStreamFrames;
    MI_U32 u32DecodeStreamFrames;
    MI_VDEC_ErrCode_e eErrCode;
} MI_VDEC_ChnStat_t;
```

➢ 成员

成员名称	描述
eCodecType	解码类型
u32LeftStreamBytes	码流 buffer 中待解码的 byte 数。
u32LeftStreamFrames	码流 buffer 中待解码的帧数。 –1 表示无效。 仅按帧发送时有效。
u32LeftPics	图像 buffer 中剩余的 pic 数目。
bChnStart	解码器是否已经启动接收码流。
u32RecvStreamFrames	码流 buffer 中已接收码流帧数。 -1 表示无效。 仅按帧发送时有效。
u32DecodeStreamFrames	码流 buffer 中已解码帧数。
eErrCode	解码错误码信息。

※ 注意事项

无。

➢ 相关数据类型及接口

无。

3.16. MI_VDEC_ChnParam_t

➢ 说明

定义解码通道参数。

➢ 定义

```
typedef struct MI_VDEC_ChnParam_s
{
    MI_VDEC_DecodeMode_e eDecMode;
    MI_VDEC_OutputOrder_e eOutputOrder;
    MI_VDEC_VideoFormat_e eVideoFormat;
} MI_VDEC_ChnParam_t;
```

➢ 成员

成员名称	描述
eDecMode	解码模式，参考MI_VDEC_DecodeMode_e定义。默认为E_MI_VDEC_DECODE_MODE_IP。 E_MI_VDEC_DECODE_MODE_ALL模式需要足够内存。
eOutputOrder	参考MI_VDEC_OutputOrder_e定义，默认为E_MI_VDEC_OUTPUT_ORDER_DECODE 按解码顺序输出数制帧
eVideoFormat	参考MI_VDEC_VideoFormat_e定义。

※ 注意事项

无。

➢ 相关数据类型及接口

无。

3.17. MI_VDEC_VideoStream_t

➢ 说明

定义视频解码的码流结构体。

➢ 定义

```
typedef struct MI_VDEC_VideoStream_s
{
    MI_U8*    pu8Addr;
    MI_U32 u32Len;
    MI_U64 u64PTS;
    MI_BOOL bEndOfFrame;
    MI_BOOL bEndOfStream;
}MI_VDEC_VideoStream_t;
```

➢ 成员

成员名称	描述
pu8Addr	码流包的地址。
u32Len	码流包的长度，以字节为单位。
u64PTS	码流包的时间戳，以毫秒为单位。
bEndOfFrame	当前帧是否结束，预留，当前只支持帧模式下按帧传送。
bEndOfStream	是否发完所有码流。当所有码流数据帧传送完毕时，bEndOfStream值需置为TRUE。

※ 注意事项

- 支持按帧或按流传送数据。按帧传送时，每次需完整传送一帧数据。暂不支持按流发送。
- 码流帧数据附带PTS时，解码后输出数据输出相同PTS。当PTS=-1时，不参考系统时钟输出数据帧。

➢ 相关数据类型及接口

无。

3.18. MI_VDEC_UserData_t

➢ 说明

定义用户数据结构体。

➢ 定义

```
typedef struct MI_VDEC_UserData_s
{
    MI_U8* pu8Addr;
    MI_U32 u32Len;
    MI_BOOL bValid;
} MI_VDEC_UserData_t;
```

➢ 成员

成员名称	描述
pu8Addr	用户数据的虚拟地址。
u32Len	用户数据的长度。以 byte 为单位。
bValid	当前数据的有效标识。取值范围：{TRUE, FALSE}。 TRUE：有效。 FALSE：无效。

※ 注意事项

无。

➢ 相关数据类型及接口

无。

3.19. MI_VDEC_OutputPortAttr_t

➢ 说明

定义输出端口属性结构体。

➢ 定义

```
typedef struct MI_VDEC_OutputPortAttr_s
{
    MI_U16 u16Width;                    // Width of target image
    MI_U16 u16Height;                   // Height of target image
}MI_VDEC_OutputPortAttr_t;
```

➢ 成员

成员名称	描述
u16Width	输出图像宽
u16Height	输出图像高

※ 注意事项

- 缩放范围[1/8, 1]。
- 不支持放大。

➢ 相关数据类型及接口

无。

4. 错误码

视频解码错误码如表 3-1 所示：

表 3-1 VDEC API 错误码

错误代码	宏定义	描述
0xA0082001	MI_ERR_VDEC_INVALID_DEVID	设备 ID 超出合法范围
0xA0082002	MI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围
0xA0082003	MI_ERR_VDEC_ILLEGAL_PARAM	非法参数或输入参数超过通道解码能力
0xA0082004	MI_ERR_VDEC_CHN_EXIST	试图创建已经存在的通道
0xA0082005	MI_ERR_VDEC_CHN_UNEXIST	通道不存在
0xA0082006	MI_ERR_VDEC_NULL_PTR	输入参数为空指针
0xA0082007	MI_ERR_VDEC_NOT_CONFIG	使用前未配置
0xA0082008	MI_ERR_VDEC_NOT_SUPPORT	该操作或者功能不支持
0xA0082009	MI_ERR_VDEC_NOT_PERM	非法操作
0xA008200C	MI_ERR_VDEC_NOMEM	分配内存失败，如系统内存不足
0xA008200D	MI_ERR_VDEC_NOBUF	分配缓存失败，如申请的数据缓冲区太大
0xA008200E	MI_ERR_VDEC_BUF_EMPTY	缓冲区中无数据
0xA008200F	MI_ERR_VDEC_BUF_FULL	缓冲区中数据满
0xA0082010	MI_ERR_VDEC_SYS_NOTREADY	系统没有初始化或者相关依赖的模块没有加载
0xA0082011	MI_ERR_VDEC_BADADDR	地址错误
0xA0082012	MI_ERR_VDEC_BUSY	系统忙
0xA0082013	MI_ERR_VDEC_CHN_NOT_START	通道未使能
0xA0082014	MI_ERR_VDEC_CHN_NOT_STOP	通道未禁用
0xA0082015	MI_ERR_VDEC_NOT_INIT	模块没有成功加载
0xA0082016	MI_ERR_VDEC_INITED	模块已成功加载
0xA008201F	MI_ERR_VDEC_FAILED	失败