MI DIVP API

Version 2.04

REVISION HISTORY

Revision No.	Description	Date
2.03	• Initial release	04/12/2018
2.04	• Added new API: • MI_DIVP_StretchBuf	12/16/2019

TABLE OF CONTENTS

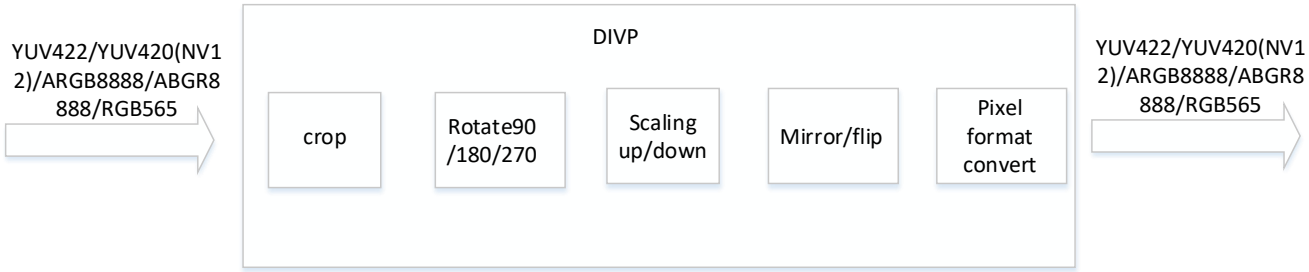
1. 概述

1.1. 模块说明

DIVP 支持对一幅输入图像进行预处理，如裁剪、图像像素格式转换、旋转、镜像等，然后再对各通道分别进行缩放处理，最后输出多种不同分辨率的图像。

DIVP 支持的具体图像处理功能包括图像的裁剪、图像像素格式转换、图像的旋转、图像的镜像操作以及图像的拉伸和缩放。

1.2. 流程框图



※ 注意

不支持 rotate 的 chip 系列有 325、327、621、623、201、202、335、337 系列。

1.3. 关键字说明

- crop
  图像裁剪
- rotate
  图像旋转
- scaling up/down
  图像拉伸和缩放
- Mirror/flip
  图像镜像操作
- Pixel format convert

图像像素格式转换

2. API 参考

该功能模块提供以下 API:

API 名	功能
MI_DIVP_CreateChn	创建一个 DIVP channel。
MI_DIVP_DestroyChn	销毁一个 DIVP channel。
MI_DIVP_SetChnAttr	设置 DIVP channel 的属性。
MI_DIVP_GetChnAttr	获取 DIVP channel 的属性。
MI_DIVP_StartChn	开启一个通道。
MI_DIVP_StopChn	禁用一个通道。
MI_DIVP_SetOutputPortAttr	设置 DIVP output port 的属性。
MI_DIVP_GetOutputPortAttr	获取 DIVP output port 的属性。
MI_DIVP_RefreshChn	在暂停的状态下刷新某个 channel
MI_DIVP_StretchBuf	缩放、拉伸、裁剪指定内存中的图像数据

2.1. MI_DIVP_CreateChn

➢ 功能

创建一个新的 DIVP channel。

➢ 语法

MI_S32 MI_DIVP_CreateChn (
MI_DIVP_CHN DivpChn,
MI_DIVP_ChnAttr_t* pstAttr);

➢ 形参

参数名称	描述	输入/输出
DivpChn	返回被创建的 DIVP 通道的 ID，DIVP 最多支援 6 个通道，取值范围[0,5]。	输入
pstAttr	DVIP 通道属性的指针，用于设定被创建的通道的属性。	输入

➢ 返回值

返回值：
- MI_SUCCESS 成功创建一个新的 DIVP 通道。
- MI_DIVP_ERR_FAILED 创建 DIVP 通道失败
- MI_DIVP_ERR_NO_RESOUCE 系统资源不足，创建 DIVP 通道失败

➢ 依赖

头文件：mi_divp.h、mi_divp_datatype.h
库文件：libmi_divp.so

※ 注意

DIVP 模块最多支援 6 个 channel，当上层创建的 channel 总数大于 6 个或者系统资源不足时会创建失败。

➢ 举例

```
MI_DIVP_CHN u32ChnId = 0;
MI_DIVP_ChnAttr_t stDivpChnAttr;
MI_DIVP_OutputPortAttr_t stDivpOutputPortAttr;

memset(&stDivpChnAttr,0,sizeof(MI_DIVP_ChnAttr_t));
memset(&stDivpOutputPortAttr,0,sizeof(MI_DIVP_OutputPortAttr_t));

stDivpChnAttr.bHorMirror =  false;
stDivpChnAttr.bVerMirror = false;
stDivpChnAttr.eDiType = E_MI_DIVP_DI_TYPE_OFF;
stDivpChnAttr.eRotateType = E_MI_SYS_ROTATE_90;
```

```
stDivpChnAttr.eTnrLevel = E_MI_DIVP_TNR_LEVEL_OFF;
stDivpChnAttr.stCropRect.u16X = 0;
stDivpChnAttr.stCropRect.u16Y = 0;
stDivpChnAttr.stCropRect.u16Width = 1280;
stDivpChnAttr.stCropRect.u16Height = 720;
stDivpChnAttr.u32MaxWidth = 1920;
stDivpChnAttr.u32MaxHeight = 1080;
MI_DIVP_CreateChn(u32ChnId,&stDivpChnAttr);

MI_DIVP_GetChnAttr(u32ChnId,&stDivpChnAttr);
stDivpChnAttr.stCropRect.u16X = 0;
stDivpChnAttr.stCropRect.u16Y = 0;
stDivpChnAttr.stCropRect.u16Width = 1920;
stDivpChnAttr.stCropRect.u16Height = 1080;
MI_DIVP_SetChnAttr(u32ChnId,&stDivpChnAttr);

stDivpOutputPortAttr.eCompMode = E_MI_SYS_COMPRESS_MODE_NONE;
stDivpOutputPortAttr.ePixelFormat = E_MI_SYS_PIXEL_FRAME_YUV422_YUYV;
stDivpOutputPortAttr.u32Width = 1920;
stDivpOutputPortAttr.u32Height = 1080;
MI_DIVP_SetOutputPortAttr(u32ChnId,&stDivpOutputPortAttr);

MI_DIVP_StartChn(u32ChnId);

//exit flow
MI_DIVP_StopChn(u32ChnId);
MI_DIVP_DestroyChn(u32ChnId);
```

- 相关主题

  MI_DIVP_DestroyChn

2.2. MI_DIVP_DestroyChn

- 功能

  销毁一个 DIVP 通道。

- 语法

  MI_S32 MI_DIVP_DestroyChn (MI_DIVP_CHN DivpChn);

- 形参

参数名称	描述	输入/输出
DivpChn	被销毁的 DVIP 通道的 ID	输入

- 返回值

返回值		
	MI_SUCCESS	成功销毁 DVIP 通道。

MI_DIVP_ERR_FAILED　　销毁 DVIP 通道失败

- 依赖

  头文件：mi_divp.h、mi_divp_datatype.h

  库文件：libmi_divp.so

※ 注意

无

➢ 相关主题

MI_DIVP_CreateChn

2.3. MI_DIVP_SetChnAttr

➢ 功能

设置 DIVP channel 的属性。

➢ 语法

```
MI_S32 MI_DIVP_SetChnAttr(
MI_DIVP_CHN DivpChn,
MI_DIVP_ChnAttr_t* pstAttr);
```

➢ 形参

参数名称	描述	输入/输出
DivpChn	DIVP 通道的 ID。	输入
pstAttr	设定的 DVIP 通道的属性的结构体指针。	输入

➢ 返回值

返回值		
	MI_SUCCESS	成功设定 DIVP 通道的属性
	MI_DIVP_ERR_FAILED	设置 DIVP 通道的属性失败
	MI_ERR_INVALID_PARAMETER	传入的参数有问题，请参考 MI_DIVP_ChnAttr_t

➢ 依赖

头文件：mi_divp.h、mi_divp_datatype.h

库文件：libmi_divp.so

※ 注意

通道属性结构体成员中输入图像的最大宽度与最大高度为静态属性，通道创建后不能改变。

➢ 举例

无

➢ 相关主题

MI_DIVP_GetChnAttr

2.4. MI_DIVP_GetChnAttr

- 功能

  获取 DIVP channel 的属性。

- 语法

  MI_S32 MI_DIVP_GetChnAttr(
  MI_DIVP_CHN DivpChn,
  MI_DIVP_ChnAttr_t* pstAttr);

- 形参

参数名称	描述	输入/输出
DivpChn	DIVP 通道的 ID。	输入
pstAttr	传回的 DVIP 通道的属性的结构体指针。	输出

- 返回值

返回值	MI_SUCCESS	成功获取 DIVP 通道的属性。
	MI_DIVP_ERR_FAILED	获取 DIVP 通道的属性失败。

- 依赖

  头文件：mi_divp.h、mi_divp_datatype.h
  库文件：libmi_divp.so

※ 注意

  第一次获取时返回创建时的初始值。

- 举例

  NA。

- 相关主题

  MI_DIVP_SetChnAttr

2.5. MI_DIVP_StartChn

- 功能

  开启一个通道。

- 语法

  MI_S32 MI_DIVP_StartChn(MI_DIVP_CHN DivpChn);

- 形参

参数名称	描述	输入/输出
DivpChn	DIVP 通道的 ID。	输入

- 返回值

返回值：
- MI_SUCCESS 成功启用 DIVP 通道。
- MI_DIVP_ERR_FAILED 启用 DIVP 通道失败。

- 依赖

头文件：mi_divp.h、mi_divp_datatype.h
库文件：libmi_divp.so

※ 注意

通道必须已经被创建，且没有被销毁。

- 举例

NA。

- 相关主题

MI_DIVP_StopChn

2.6. MI_DIVP_StopChn

- 功能

禁用一个通道。

- 语法

MI_S32 MI_DIVP_StopChn(MI_DIVP_CHN DivpChn);

- 形参

参数名称	描述	输入/输出
DivpChn	DIVP 通道的 ID。	输入

- 返回值

- MI_SUCCESS 成功禁用 DIVP 通道。

返回值

MI_DIVP_ERR_FAILED　禁用 DIVP 通道失败。

➢ 依赖

头文件：mi_divp.h、mi_divp_datatype.h

库文件：libmi_divp.so

※ 注意

重复禁用一个通道会返回 MI_SUCCESS。

➢ 举例

NA。

➢ 相关主题

MI_DIVP_StartChn

2.7. MI_DIVP_SetOutputPortAttr

➢ 功能

DIVP 通道上 output port 的属性。

➢ 语法

```
MI_S32 MI_DIVP_SetOutputPortAttr (
MI_DIVP_CHN DivpChn,
MI_DIVP_OutputPortAttr_t * pstOutputPortAttr);
```

➢ 形参

参数名称	描述	输入/输出
DivpChn	DIVP 通道的 ID。	输入
pstOutputPortAttr	Output port 的属性的指针。	输入

➢ 返回值

返回值	MI_SUCCESS	成功设定 DIVP 通道绑定的 output port 的属性。
	MI_DIVP_ERR_FAILED	设定 DIVP 通道绑定的 output port 的属性失败。

➢ 依赖

头文件：mi_divp.h、mi_divp_datatype.h

库文件：libmi_divp.so

※ 注意

无。

➢ 举例

NA。

➢ 相关主题

MI_DIVP_GetOutputPortAttr

2.8. MI_DIVP_GetOutputPortAttr

➢ 功能

获取 DIVP output port 的属性。

➢ 语法

```
MI_S32 MI_DIVP_GetOutputPortAttr(
MI_DIVP_CHN DivpChn,
MI_DIVP_OutputPortAttr_t * pstOutputPortAttr);
```

➢ 形参

参数名称	描述	输入/输出
DivpChn	DIVP 通道的 ID。	输入
pstOutputPortAttr	Output port 的属性的指针	输出

➢ 返回值

返回值		
	MI_SUCCESS	成功获取 DIVP 通道绑定的 output port 的属性。
	MI_DIVP_ERR_FAILED	获取 DIVP 通道绑定的 output port 的属性失败。

➢ 依赖

头文件：mi_divp.h、mi_divp_datatype.h

库文件：libmi_divp.so

※ 注意

无。

➢ 举例

NA

- 相关主题

MI_DIVP_SetOutputPortAttr

2.9. MI_DIVP_RefreshChn

- 功能

暂停状态下刷新 DIVP channel。

- 语法

MI_S32 MI_DIVP_RefreshChn (
MI_DIVP_CHN DivpChn);

- 形参

参数名称	描述	输入/输出
DivpChn	被刷新的 channel 的 ID。	输入

- 返回值

返回值：
- MI_SUCCESS 成功刷新 DIVP 的 channel。
- MI_DIVP_ERR_NO_CONTENT 该通道没有内容。

- 依赖

头文件：mi_divp.h、mi_divp_datatype.h
库文件：libmi_divp.so

※ 注意

该接口适用于 channel 在暂停的状态下重新刷新。

- 举例

DIVP 绑定 VDEC 的场景下，调用该接口，DIVP 可以保留 VDEC 停止送流后的最后一帧画面，并重复处理然后送给后端，直到 VDEC 重新开始送流，DIVP 才会处理新的画面。

- 相关主题

NA

2.10. MI_DIVP_StretchBuf

- 功能

  缩放、拉伸、裁剪指定内存区域中的图像数据，并输出到目标内存。

- 语法

  MI_S32 MI_DIVP_StretchBuf(MI_DIVP_DirectBuf_t *pstSrcBuf, MI_SYS_WindowRect_t *pstSrcCrop, MI_DIVP_DirectBuf_t *pstDstBuf);

➢ 形参

参数名称	描述	输入/输出
pstSrcBuf	用来存放待处理图像内存信息的结构体指针，不可以为 NULL	输入
pstSrcCrop	图像裁剪相关信息的结构体指针，如果不需要做裁剪，可以传 NULL	输入
pstDstBuf	保存处理后图像内存信息的结构体指针，不可以为 NULL	输入

➢ 返回值

返回值
- MI_SUCCESS 图像处理成功。
- MI_DIVP_ERR_FAILED 图像处理失败。

➢ 依赖

头文件：mi_divp.h、mi_divp_datatype.h

库文件：libmi_divp.so

※ 注意

该接口只支持 YUV420SP 或者 ARGB8888 的像素格式。

➢ 举例

```
#define SRC_WIDTH 1280
#define SRC_HEIGHT 720
#define SRC_BUFF_STRIDE (ALIGN_UP(SRC_WIDTH,16))
#define SRC_BUFF_SIZE (SRC_BUFF_STRIDE*SRC_HEIGHT*3/2)

#define DST_WIDTH 640
#define DST_HEIGHT 480
#define DST_BUFF_STRIDE (ALIGN_UP(DST_WIDTH,16))
#define DST_BUFF_SIZE (DST_BUFF_STRIDE*DST_HEIGHT*3/2)

#define CROP_X 200
#define CROP_Y 100
#define CROP_W 68
#define CROP_H 48

int main(void)
{
    MI_PHY phySrcBufAddr = 0;
    MI_PHY phyDstBufAddr = 0;
    MI_DIVP_DirectBuf_t stSrcBuf;
    MI_DIVP_DirectBuf_t stDstBuf;
    MI_SYS_WindowRect_t stSrcCrop;
```

```
    MI_SYS_MMA_Alloc(NULL, SRC_BUFF_SIZE, &phySrcBufAddr);
    MI_SYS_MMA_Alloc(NULL, DST_BUFF_SIZE, &phyDstBufAddr);

    stSrcBuf.ePixelFormat = E_MI_SYS_PIXEL_FRAME_YUV_SEMIPLANAR_420;
    stSrcBuf.u32Width = SRC_WIDTH;
    stSrcBuf.u32Height = SRC_HEIGHT;
    stSrcBuf.u32Stride[0] = SRC_BUFF_STRIDE;
    stSrcBuf.u32Stride[1] = SRC_BUFF_STRIDE;
    stSrcBuf.phyAddr[0] = phySrcBufAddr;
    stSrcBuf.phyAddr[1] = stSrcBuf.phyAddr[0] + SRC_BUFF_STRIDE*SRC_HEIGHT;
    stDstBuf.ePixelFormat = E_MI_SYS_PIXEL_FRAME_YUV_SEMIPLANAR_420;
    stDstBuf.u32Width = DST_WIDTH;
    stDstBuf.u32Height = DST_HEIGHT;
    stDstBuf.u32Stride[0] = DST_BUFF_STRIDE;
    stDstBuf.u32Stride[1] = DST_BUFF_STRIDE;
    stDstBuf.phyAddr[0] = phyDstBufAddr;
    stDstBuf.phyAddr[1] = stDstBuf.phyAddr[0] + DST_BUFF_STRIDE*DST_HEIGHT;

    stSrcCrop.u16X = CROP_X;
    stSrcCrop.u16Y = CROP_Y;
    stSrcCrop.u16Width = CROP_W;
    stSrcCrop.u16Height = CROP_H;

    if(FillSrcBuf("./1280x720_yuv420.yuv", &stSrcBuf))
        return NULL;
    if(MI_SUCCESS == MI_DIVP_StretchBuf(&stSrcBuf, &stSrcCrop, &stDstBuf))
    {
         if(DumpDstBuf(&stDstBuf))
            return NULL;
    }

    MI_SYS_MMA_Free(phySrcBufAddr);
    MI_SYS_MMA_Free(phyDstBufAddr);
}

//fill src buff with yuv420/argb8888 image data
static int FillSrcBuf(const char* FilePath,MI_DIVP_DirectBuf_t *pstDirectSrcBuf)
{
    int ret = 0;
    FILE *fp;
    void *pVirSrcBufAddr = NULL;
    int LineIdx = 0;
    int ReadSize = 0;

    fp = fopen(FilePath,"r");
    if(!fp)
    {
        divp_ut_dbg("open file[%s] failed\n",FilePath);
        ret = -1;
        goto EXIT;
    }

    MI_SYS_Mmap(pstDirectSrcBuf->phyAddr[0], SRC_BUFF_SIZE, &pVirSrcBufAddr, FALSE);
    if(!pVirSrcBufAddr)
    {
        divp_ut_dbg("mmap dst buff failed\n");
        ret = -1;
        goto EXIT;
```

```
    }

    for(LineIdx = 0; LineIdx < SRC_HEIGHT*3/2; LineIdx++)
    {
        ReadSize += fread(pVirSrcBufAddr+LineIdx*SRC_BUFF_STRIDE, 1, SRC_WIDTH, fp);
    }
    if(ReadSize < SRC_WIDTH*SRC_HEIGHT*3/2)
    {
        fseek(fp, 0, SEEK_SET);
        ReadSize = 0;
        for(LineIdx = 0; LineIdx < SRC_HEIGHT*3/2; LineIdx++)
        {
            ReadSize += fread(pVirSrcBufAddr+LineIdx*SRC_BUFF_STRIDE, 1, SRC_WIDTH,
fp);
        }
        if(ReadSize < SRC_WIDTH*SRC_HEIGHT*3/2)
        {
            divp_ut_dbg("read file failed, read size:%d\n",ReadSize);
            ret = -1;
            goto EXIT;
        }
    }

EXIT:
    if(fp)
        fclose(fp);
    if(pVirSrcBufAddr)
        MI_SYS_Munmap(pVirSrcBufAddr, SRC_BUFF_SIZE);

    return ret;
}

//image processing result is stored in dst buff
static int DumpDstBuf(MI_DIVP_DirectBuf_t *pstDirectDstBuf)
{
    int ret = 0;
    FILE *fp;
    void *pVirDstBufAddr = NULL;
    int LineIdx = 0;
    int WriteSize = 0;
    char outputfile[128];
    struct timeval timestamp;

    gettimeofday(&timestamp, 0);
    sprintf(outputfile,
"output_%dx%d_%d_%08d.yuv",pstDirectDstBuf->u32Width,pstDirectDstBuf->u32Height,(i
nt)timestamp.tv_sec,(int)timestamp.tv_usec);

    fp = fopen(outputfile,"w+");
    if(!fp)
    {
        divp_ut_dbg("open file[%s] failed\n",outputfile);
        ret = -1;
        goto EXIT;
    }

    MI_SYS_Mmap(pstDirectDstBuf->phyAddr[0], DST_BUFF_SIZE, &pVirDstBufAddr, FALSE);
    if(!pVirDstBufAddr)
```

```
    {
        divp_ut_dbg("mmap dst buff failed\n");
        ret = -1;
        goto EXIT;
    }

    for(LineIdx = 0; LineIdx < DST_HEIGHT*3/2; LineIdx++)
    {
        WriteSize += fwrite(pVirDstBufAddr+LineIdx*DST_BUFF_STRIDE, 1, DST_WIDTH, fp);
    }
    if(WriteSize < DST_WIDTH*DST_HEIGHT*3/2)
    {
        divp_ut_dbg("write file failed, write size:%d\n",WriteSize);
    }
    fflush(fp);
    sync();
    divp_ut_dbg("save stretch dst buff to[%s]\n",outputfile);

EXIT:
    if(fp)
        fclose(fp);
    if(pVirDstBufAddr)
        MI_SYS_Munmap(pVirDstBufAddr, DST_BUFF_SIZE);

    return 0;
}
```

- **相关主题**

NA

3. DIVP 数据类型

DIVP 相关数据类型定义如下:

MI_DIVP_DiType_e	定义 DIVP 的 deinterlace 的类型。
MI_DIVP_TnrLevel_e	定义 DIVP TNR 的等级。
MI_DIVP_OutputPortAttr_t	定义 DIVP 绑定的 output port 的属性参数。
MI_DIVP_ChnAttr_t	定义 DIVP 通道的属性参数。
MI_DIVP_CHN	DIVP 通道的 ID。
MI_DIVP_DirectBuf_t	用于图像处理的内存信息结构体

注：本节已涵盖各重要的数据类型，部分未列出数据类型请参见 mi_divp_datatype.h

3.1. MI_DIVP_DiType_e

➢ 说明

定义 DIVP 的 deinterlace 的类型。

➢ 定义

```
typedef enum
{
 E_MI_DIVP_DITYPE_OFF,//off
 E_MI_DIVP_DITYPE_2D,///2.5D DI
 E_MI_DIVP_DITYPE_3D,///3D DI
 E_MI_DIVP_DITYPE_NUM,
} MI_DIVP_DiType_e;
```

➢ 成员

成员	描述
E_MI_DIVP_DITYPE_OFF	DIVP 通道上关闭 deinterlace。
E_MI_DIVP_DITYPE_2D	DIVP 通道上开启 2.5D deinterlace。
E_MI_DIVP_DITYPE_3D	DIVP 通道上开启 3D deinterlace。
E_MI_DIVP_DITYPE_NUM	DIVP 通道上 deinterlace 类型个数。

※ 注意事项

1. 开启 DI 时必须开 TNR，MSR930 只能支援 3D DI，TNR level 设定为 E_MI_DIVP_TNRLEVEL_MIDDLE。
2. 3D DI 与 rotation 冲突，两个功能不能同时打开。
3. 不支持 DI 功能的 chip 如下：
   328Q/329D/326D
   325/325DE/327DE
   621/623/201/202
   336D/336Q/339G
   335/337DE

➢ 相关数据类型及接口

无。

3.2. MI_DIVP_TnrLevel_e

➢ 说明

定义 DIVP TNR 的等级。

➢ 定义

```
typedef enum
{
 E_MI_DIVP_TNRLEVEL_OFF,
 E_MI_DIVP_TNRLEVEL_LOW,
 E_MI_DIVP_TNRLEVEL_MIDDLE,
 E_MI_DIVP_TNRLEVEL_HIGH,
 E_MI_DIVP_TNRLEVEL_NUM,
} MI_DIVP_TnrLevel_e;
```

➢ 成员

成员	描述
E_MI_DIVP_TNRLEVEL_OFF	DIVP 通道上关闭 TNR。
E_MI_DIVP_TNRLEVEL_LOW	DIVP 通道上开启弱等级的 TNR。
E_MI_DIVP_TNRLEVEL_MIDDLE	DIVP 通道上开启中等等级的 TNR。
E_MI_DIVP_TNRLEVEL_HIGH	DIVP 通道上开启强等级的 TNR。
E_MI_DIVP_TNRLEVEL_NUM	DIVP 通道上 TNR 等级的数量。

※ 注意事项

MSR930 支援 TNR，但是不支持调节 TNR level。

不支持 TNR 功能的 chip 如下：

328Q/329D/326D
325/325DE/327DE
621/623/201/202
336D/336Q/339G
335/337DE

➢ 相关数据类型及接口

无。

3.3. MI_DIVP_OutputPortAttr_t

➢ 说明

定义 DIVP 绑定的 output port 的属性参数。

➢ 定义

```
typedef struct MI_DIVP_OutputPortAttr_s
    {
        MI_U32 u32Width;//output width
        MI_U32 u32Height;//output height
        MI_SYS_PixelFormat_e ePixelFormat;
```

```
    MI_SYS_CompressMode_e  eCompMode;//compress  mode
}MI_DIVP_OutputPortAttr_t;
```

➢ 成员

成员名称	描述
u32Width	DIVP 通道输出画面的宽度。
u32Height	DIVP 通道输出画面的高度。
ePixelFormat	DIVP 通道输出画面的像素格式。
eCompMode	DIVP 通道输出图像的压缩格式，DIVP 通道只能输出非压缩格式的图像。

※ 注意事项

各系列 chip DIVP 输出属性差异

输出属性 / 芯片系列	Output Pixel format	Output Stride alignment	Output Width alignment	Output Height alignment	Output Min size	Output Max size
MSR930	YUV422/YUV420(NV12)/ ARGB8888/ ABGR8888/ARGB1555/ MST420	32	2	2	128x64	4096x4096
328Q/329D/326D	YUV422	32	2	2	64x4	3840x3840
	YUV420(NV12)	16				
	ARGB8888/ABGR8888	64				
	RGB565	32				
325/325DE/327DE	YUV422	32	2	2	64x4	2688x2688
	YUV420(NV12)	16				
	ARGB8888/ABGR8888	64				
	RGB565	32				
621/623/201/202	YUV422	32	2	2	64x4	1920x1920
	YUV420(NV12)	16				
	ARGB8888/ABGR8888	64				
	RGB565	32				
336D/336Q/339G	YUV422	32	2	2	Rotate:16x2 No rotate:32x4	3840x3840
	YUV420(NV12)	16				
	ARGB8888/ABGR8888	64				
	RGB565	32				
335/337DE	YUV422	32	2	2	64x4	2688x2688

	YUV420(NV12)	16				
	ARGB8888/ABGR8888	64				
	RGB565	32				

➢ 相关数据类型及接口

无。

3.4. MI_DIVP_ChnAttr_t

➢ 说明

定义 DIVP 通道的属性参数。

➢ 定义

```
typedef struct MI_DIVP_ChnAttr_s
{
    MI_U32 u32MaxWidth;//support max input width
    MI_U32 u32MaxHeight;//support max input height
    MI_DIVP_TnrLevel_e eTnrLevel;//TNR level
    MI_DIVP_DiType_e eDiType;//DI type
    MI_SYS_Rotate_e eRotateType;//rotate angle
    MI_SYS_WindowRect_t stCropRect;//crop information
    MI_BOOL bHorMirror;//horizontal mirror
    MI_BOOL bVerMirror;//vertical mirror
}MI_DIVP_ChnAttr_t;
```

➢ 成员

成员名称	描述
u32MaxWidth	DIVP 通道支援的 input 的最大宽度。
u32MaxHeight	DIVP 通道支援的 input 的最大高度。
eTnrLevel	DIVP 通道上 TNR 的等级。
eDiType	DIVP 通道上 DI 的类型。
eRotateType	DIVP 通道上画面旋转的角度。
stCropRect	DIVP 通道上的 crop 信息。
bHorMirror	DIVP 通道上水平方向翻转
bVerMirror	DIVP 通道上垂直方向翻转

※ 注意事项

输入属性 / 芯片系列	Input Pixel format	Input Stride alignment	Input Width alignment	Input Height alignment	Input Min size	Input Max size
MSR930	YUV422/YUV420(NV12)/ARGB8888/ ABGR8888/ARGB1555/Tile Mode	32	YUV422:16 NV12:32	2	128x64	4096x4096
328Q/329D/326D	YUV422	32	2	2	64x4	3840x3840
	YUV420(NV12)	16				
	ARGB8888/ABGR8888	64				
	RGB565	32				
325/325DE/327DE	YUV422	32	2	2	64x4	2688x2688
	YUV420(NV12)	16				
	ARGB8888/ABGR8888	64				
	RGB565	32				
621/623/201/202	YUV422	32	2	2	64x4	1920x1920
	YUV420(NV12)	16				
	ARGB8888/ABGR8888	64				
	RGB565	32				
336D/336Q/339G	YUV422	32	2	2	Rotate:128x128 No rotate:32x4	3840x3840
	YUV420(NV12)	16				
	ARGB8888/ABGR8888	64				
	RGB565	32				
335/337DE	YUV422	32	2	2	64x4	2688x2688
	YUV420(NV12)	16				
	ARGB8888/ABGR8888	64				
	RGB565	32				

➢ 相关数据类型及接口

无。

3.5. MI_DIVP_DirectBuf_t

➢ 说明

定义用于图像处理的内存信息

- 定义

```
typedef struct MI_DIVP_DirectBuf_s
{
    MI_SYS_PixelFormat_e ePixelFormat; //YUV420SP or ARGB888 only
    MI_U32 u32Width;
    MI_U32 u32Height;
    MI_U32 u32Stride[3];
    MI_PHY phyAddr[3];
}MI_DIVP_DirectBuf_t;
```

- 成员

成员名称	描述
ePixelFormat	图像的像素格式
u32Width	图像画面的宽度
u32Height	图像画面的高度
u32Stride	图像每行所占字节数
phyAddr	Buffer 的起始物理地址

- 注意事项

只支持 YUV420SP 或者 ARGB8888 的图像像素格式

u32Stride 不能小于 64

- 相关数据类型及接口

MI_DIVP_StretchBuf

4. DIVP 错误码

DIVP API 返回值如表 3-1 所示：

表 3-1 DIVP API 返回值

错误代码	宏定义	描述
0x0	MI_SUCCESS	succeeded
0xa00c2002	MI_DIVP_ERR_INVALID_CHNID	无效的 channel ID。
0xa00c2003	MI_DIVP_ERR_INVALID_PARAM	传入的参数无效
0xa00c2006	MI_DIVP_ERR_NULL_PTR	空指针异常
0xa00c201c	MI_DIVP_ERR_FAILED	DIVP 的操作失败
0xa00c2005	MI_DIVP_ERR_NO_RESOUCE	无资源可以使用
0xa00c201c	MI_DIVP_ERR_NO_CONTENT	通道中无显示内容