MI RGN API

Version 2.07

REVISION HISTORY

Revision No.	Description	Date
2.03	• Initial release	04/12/2018
2.04	• Updated GetCanvas limitation	01/10/2019
2.05	• Removed GetCanvas limitation	01/25/2019
2.06	• Added OSD alpha setting	02/21/2019
2.07	• Added LDC module ID	09/16/2019

TABLE OF CONTENTS

1. 概 述

1.1. 模块说明

区域管理模块参与 VPE、DIVP、LDC 模块的内部流处理的一个环节。底层硬件模块支持是 GOP(Graphic output path)，区域管理模块利用 GOP 的特性抽象出来的一套软件接口，利用分时复用的原理使 OSD(On-screen display)或者 Cover 贴到各个通道上。

区域管理模块提供区域资源的控制管理功能，包括区域的创建、销毁、获取与设置区域属性、获取与设置区域的通道属性等。

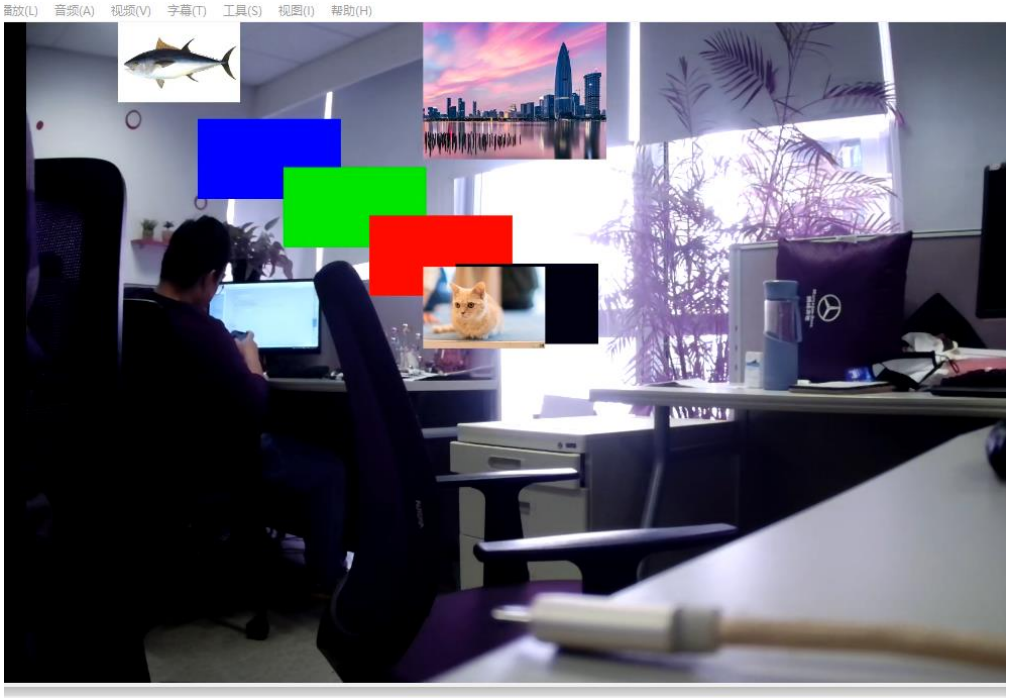
区域属性分为两种，一种是 cover，Cover 只是按照设定，给一块区域做遮挡，底层驱动只要知道显示的位置大小和颜色即可，因此创建 Cover 时，不会给 cover 分配内存做显示。

另外一种区域属性称为 osd,有些地方亦称为 overlay，它同样可以贴在 video 显示的区域，这块区域用内存来描述显示的内容，所以这块内容可以做点对点画图操作，显示内存支持 argb、位图两种格式，使用者可以根据实际的场景来选择所需要的格式。

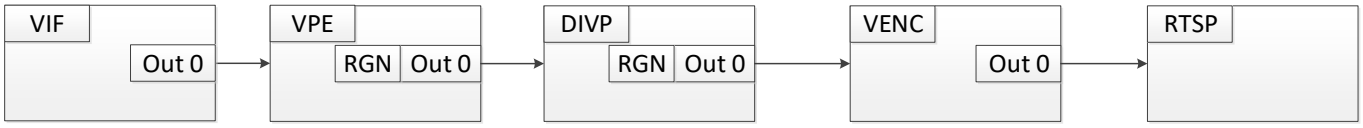
目前 OSD 可以支持的格式有 ARGB1555、ARGB4444、ARGB8888、RGB565、I2、I4、I8，每个芯片支持的情况会有差异，下文会详细说明。暂不支持 YUV 格式。I2、I4、I8 格式为位图格式，一个 pixel 的内存数据当作一个索引，通过索引能找到调色盘中的颜色数据，当前 pixel 显示的就是此颜色。

无论是 Cover 还是 Osd 其显示的内容均是叠加到 VPE/DIVP/LDC 模块的输出上 ,通过实验 ,当设定了 region 显示之后，把 VPE/DIVP/LDC 模块的输出 dump 成文件,通过工具就可以看到 region 的内容.

OSD 可以显示一张图片，COVER 仅仅可以设定颜色，在同一通道上 OSD 的内容永远在 COVER 之上，下图在同一个通道上显示了四个 cover 区域以及三张 osd。



1.2. 流程框图



1.3. 关键字说明

- I2:
  4 色位图格式，用 2 个 bit 表示一个索引，因而有 4 种颜色，在调色盘中通过索引找到对应的颜色。
- I4：
  16 色位图格式，与 I2 格式类似，不同的是用 4 个 bit 表示一个索引，因而有 16 种颜色。
- I8：
  256 色位图格式，不同的是用 8 个 bit 表示一个索引，因而有 256 种颜色。
- Palette:
  位图的调色盘，由 Alpha、Red、Green、Blue 四个 8bit 变量表示一个 pixel 的颜色，一共有 256 个颜色，对应 0-255 的索引序号。
- OSD
  On-screen display 的简称，用来显示一些文字、图片、以及人机交互的菜单等内容。
- GOP
  Graphic output path 的简称，可以理解为在 video 层之上的一个图形层。

2. API 参考

2.1. API List

API 名	功能
MI_RGN_Init	初始化
MI_RGN_DeInit	反初始化
MI_RGN_Create	创建区域
MI_RGN_Destroy	销毁区域
MI_RGN_GetAttr	获取区域属性
MI_RGN_SetBitMap	设置区域位图
MI_RGN_AttachToChn	将区域叠加到通道上
MI_RGN_DetachFromChn	将区域从通道中撤出
MI_RGN_SetDisplayAttr	设置区域的通道显示属性
MI_RGN_GetDisplayAttr	获取区域的通道显示属性
MI_RGN_GetCanvasInfo	获取区域画布信息
MI_RGN_UpdateCanvas	更新区域画布信息
MI_RGN_ScaleRect	放大区域画布

2.1.1 MI_RGN_Init

➢ 功能

初始化。

➢ 语法

MI_S32 MI_RGN_Init(MI_RGN_PaletteTable_t *pstPaletteTable);

➢ 形参

参数名称	参数含义	输入/输出
pstPaletteTable	调色板指针。	输入

➢ 返回值

返回值
- MI_RGN_OK 成功。
- MI_ERR_RGN_BUSY 已初始化。

➢ 依赖

- 头文件：mi_sys.h、mi_rgn.h。
- 库文件

※ 注意

- I2/I4/I8 的图像格式是共用一份 palette table，palette table 只能在初始化时做一次，不可再次设置。
- RGB 格式不会参考 palette。
- Palette 的第 0 号成员为透明色，上层无法指定。

➢ 举例

```
MI_S32 s32Result = 0;
MI_RGN_PaletteTable_t stPaletteTable;
memset(&stPaletteTable, 0, sizeof(MI_RGN_PaletteTable_t));
stPaletteTable.astElement[1].u8Alpha = 255;
stPaletteTable.astElement[1].u8Red = 255;
stPaletteTable.astElement[1].u8Green = 0;
stPaletteTable.astElement[1].u8Blue = 0;
stPaletteTable.astElement[2].u8Alpha = 255;
stPaletteTable.astElement[2].u8Red = 0;
stPaletteTable.astElement[2].u8Green = 255;
stPaletteTable.astElement[2].u8Blue = 0;
stPaletteTable.astElement[3].u8Alpha = 255;
stPaletteTable.astElement[3].u8Red = 0;
stPaletteTable.astElement[3].u8Green = 0;
stPaletteTable.astElement[3].u8Blue = 255;
s32Result = MI_RGN_Init(&stPaletteTable);
```

```
s32Result = MI_RGN_DeInit();
```

➢ 相关主题

MI_RGN_DeInit

2.1.2 MI_RGN_DeInit

➢ 功能

反初始化。

➢ 语法

MI_S32 MI_RGN_DeInit();

➢ 形参

无。

➢ 返回值

返回值		
	MI_RGN_OK	成功。
	MI_ERR_RGN_BUSY	未初始化。

➢ 依赖

- 头文件：mi_sys.h、mi_rgn.h。
- 库文件

※ 注意

无。

➢ 举例

参见 MI_RGN_Init 举例。

➢ 相关主题

MI_RGN_Init

2.1.3 MI_RGN_Create

➢ 功能

创建区域。

➢ 语法

MI_S32 MI_RGN_Create(MI_RGN_HANDLE hHandle, MI_RGN_Attr_t *pstRegion);

➢ 形参

参数名称	参数含义	输入/输出
hHandle	区域句柄号。 必须是未使用的 hHandle 号 取值范围：[0, MI_RGN_MAX_HANDLE]。	输入
pstRegion	区域属性指针。	输入

➢ 返回值

返回值
- MI_RGN_OK 成功。
- 非 MI_RGN_OK 失败，参照返回值。

➢ 依赖

- 头文件：mi_sys.h、mi_rgn.h。
- 库文件

※ 注意

- 该句柄由用户指定，意义等同于 ID 号。
- 不支持重复创建。
- 区域属性必须合法，具体约束参见 MI_RGN_Attr_t。
- MI_RGN_Attr_t 中指定 Cover 还是 OSD
- 区域属性指针不能为空。
- 创建 Cover 时，只需指定区域类型即可。其它的属性，如区域位置，层次等信息在调用 MI_RGN_AttachToChn 接口时指定。
- 创建区域时，本接口只进行基本的参数的检查，例如：最小宽高，最大宽高等；当区域 attach 到通道上时，根据各通道模块支持类型的约束条件进行更加有针对性的参数检查，譬如支持的像素格式等；

➢ 举例

```
MI_S32 s32Result = 0;
MI_RGN_HANDLE hHandle = 0;
MI_RGN_Attr_t stRegion;
stRegion.eType = E_MI_RGN_TYPE_OSD;
stRegion.stOsdInitParam.ePixelFmt = E_MI_RGN_PIXEL_FORMAT_RGB1555;
stRegion.stOsdInitParam.stSize.u32Width = 40;
stRegion.stOsdInitParam.stSize.u32Height = 40;

s32Result = MI_RGN_Create(hHandle, &stRegion);
if (s32Result != MI_RGN_OK)
{
                    return s32Result;
```

```
}

s32Result = MI_RGN_GetAttr(hHandle, &stRegion);
if (s32Result != MI_RGN_OK)
{
                    return s32Result;
}

s32Result = MI_RGN_Destroy(hHandle);
if (s32Result != MI_RGN_OK)
{
                    return s32Result;
}
```

➢ 相关主题

MI_RGN_Destroy
MI_RGN_GetAttr

2.1.4 MI_RGN_Destroy

➢ 功能

销毁区域。

➢ 语法

MI_S32 MI_REG_Destroy (MI_RGN_HANDLE hHandle);

➢ 形参

参数名称	参数含义	输入/输出
hHandle	区域句柄号。 取值范围：[0, MI_RGN_MAX_HANDLE)。	输入

➢ 返回值

返回值
- MI_RGN_OK 成功。
- 非 MI_RGN_OK 失败，参照返回值。

➢ 依赖

- 头文件：mi_sys.h、mi_rgn.h。
- 库文件：

※ 注意

- 区域必须已创建。

- **举例**

  参见 MI_RGN_Create 举例。

- **相关主题**

  MI_RGN_Create

2.1.5 MI_RGN_GetAttr

- **功能**

  获取区域属性。

- **语法**

  MI_S32 MI_RGN_GetAttr(MI_RGN_HANDLE hHandle, MI_RGN_Attr_t *pstRegion);

➢ 形参

参数名称	参数含义	输入/输出
hHandle	区域句柄号。 取值范围：[0, MI_RGN_MAX_HANDLE)。	输入
pstRegion	区域属性指针。	输出

➢ 返回值

MI_RGN_OK 成功。

非 MI_RGN_OK 失败，参照返回值。

➢ 依赖

- 头文件：mi_sys.h、mi_rgn.h。
- 库文件：

※ 注意

- 区域必须已创建。
- 区域属性指针不能为空。

➢ 举例

参见 MI_RGN_Create 举例。

➢ 相关主题

无。

2.1.6 MI_RGN_SetBitMap

➢ 功能

设置区域位图，即对区域进行位图填充。

➢ 语法

```
MI_S32 MI_RGN_SetBitMap(MI_RGN_HANDLE hHandle,
                        MI_RGN_Bitmap_t *pstBitmap);
```

➢ 形参

参数名称	参数含义	输入/输出
hHandle	区域句柄号。 取值范围：[0, MI_RGN_MAX_HANDLE)。	输入
pstBitmap	位图属性指针。	输入

➢ 返回值

返回值
- MI_RGN_OK 成功。
- 非 MI_RGN_OK 失败，参照返回值。

➢ 依赖

- 头文件：mi_sys.h、mi_rgn.h。
- 库文件：

※ 注意

- 区域必须已创建。
- 支持位图的大小和区域的大小可以不一致。
- 位图从区域的(0,0)点开始加载。当位图比区域大时，将会自动将图像剪裁成区域大小。
- 位图的像素格式必须和区域的像素格式一致。
- 位图属性指针不能为空。
- 支持多次调用。
- 此接口只对 Overlay 有效。
- 调用了 MI_RGN_GetCanvasInfo 之后，调用本接口无效，除非 MI_RGN_UpdateCanvas 更新画布生效后再调用。

➢ 举例

```
MI_S32 s32Result = 0;
MI_HANDLE hHandle = 0;
MI_RGN_Bitmap_t stBitmap;
MI_U32 u32FileSize = 200 * 200 * 2;
MI_U8 *pu8FileBuffer = NULL;

FILE *pFile = fopen("200X200.argb1555", "rb");
if (pFile == NULL)
{
       printf("open file failed \n");
       return -1;
}

pu8FileBuffer = (MI_U8*)malloc(u32FileSize);
if (pu8FileBuffer == NULL)
{
       printf("malloc failed fileSize=%d\n", u32FileSize);
       fclose(pFile);
       return -1;
}

memset(pu8FileBuffer, 0, u32FileSize);
fread(pu8FileBuffer, 1,  u32FileSize, pFile);
fclose(pFile);
stBitmap.stSize.u32Width = 200;
stBitmap.stSize.u32Height = 200;
stBitmap.ePixelFormat = E_MI_RGN_PIXEL_FORMAT_RGB1555;
stBitmap.pData = pu8FileBuffer;
free(pu8FileBuffer);
s32Result = MI_RGN_SetBitMap(hHandle, &stBitmap);
```

➢ **相关主题**

无。

2.1.7 MI_RGN_AttachToChn

➢ **功能**

将区域叠加到通道上。

➢ **语法**

```
MI_S32 MI_RGN_AttachToChn(MI_RGN_HANDLE hHandle,
                          MI_RGN_ChnPort_t* pstChnPort,
                          MI_RGN_ChnPortParam_t *pstChnAttr);
```

➢ **形参**

参数名称	参数含义	输入/输出
hHandle	区域句柄号。 取值范围：[0, MI_RGN_MAX_HANDLE)。	输入
pstChnPort	通道端口结构体指针。	输入
pstChnAttr	区域通道显示属性指针。	输入

➢ **返回值**

返回值：
- MI_RGN_OK　成功。
- 非 MI_RGN_OK　失败，参照返回值。

➢ **依赖**

- 头文件：mi_sys.h、mi_rgn.h。
- 库文件：

※ **注意**

- 区域必须已创建。
- 通道结构体指针不能为空。
- 区域通道显示属性指针不能为空。
- 多个 osd 区域类叠加到同一个通道上，每个 osd 必须是同一个图像格式。
- 并不是所有的通道上都有叠加 region 的能力，下表列出芯片差异。
- COVER 只有硬件 layer. OSD 支持软件 layer，底层会用软件拼图实现。
- OSD、COVER 叠加到通道上对 channel id 不会有要求。
- 叠加到通道上的 OSD 小于或等于硬件 layer 个数，则全部使用硬件 layer，反之则会用软件拼图。

芯片名称	通道位置	OSD 各通道支持		Cover Layer 个数	OSD COLOR FORMAT 支持						
		硬件 layer 个数	最大 layer 个数		ARGB1555	ARGB4444	I2	I4	I8	RGB565	ARGB8888
SSC328Q SSC329D SSC326D	VPE PORT 0	2	默认 8(可调整)	4	支持	支持	支持	支持	支持	支持	支持
	VPE PORT 1	2	默认 8(可调整)	4	支持	支持	支持	支持	支持	支持	支持
	VPE PORT 2	NA	NA	4	NA	NA	NA	NA	NA	NA	NA
	VPE PORT 3	2	默认 8(可调整)	NA	支持	支持	支持	支持	支持	支持	支持
	DIVP PORT 0	2	默认 8(可调整)	NA	支持	支持	支持	支持	支持	支持	支持
	LDC PORT 0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

芯片名称	通道位置	OSD 各通道支持		Cover Layer 个数	OSD COLOR FORMAT 支持						
		硬件 layer 个数	最大 layer 个数		ARGB1555	ARGB4444	I2	I4	I8	RGB565	ARGB8888
SSC325 SSC325DE SSC327DE	VPE PORT 0	4	128	4	支持	支持	支持	支持	支持	NA	NA
	VPE PORT 1	4	128	4	支持	支持	支持	支持	支持	NA	NA
	VPE PORT 2	NA	NA	4	NA	NA	NA	NA	NA	NA	NA
	VPE PORT 3	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
	DIVP PORT 0	4	128	4	支持	支持	支持	支持	支持	NA	NA
	LDC PORT 0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

芯片名称	通道位置	OSD 各通道支持		Cover Layer 个数	OSD COLOR FORMAT 支持						
		硬件 layer 个数	最大 layer 个数		ARGB1555	ARGB4444	I2	I4	I8	RGB565	ARGB8888
SSC335 SSC337DE	VPE PORT 0	4	128	4	支持	支持	支持	支持	支持	NA	NA
	VPE PORT 1	4	128	4	支持	支持	支持	支持	支持	NA	NA
	VPE PORT 2	4	128	4	支持	支持	支持	支持	支持	NA	NA
	VPE PORT 3	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
	DIVP PORT 0	4	128	4	支持	支持	支持	支持	支持	NA	NA
	LDC PORT 0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

芯片名称	通道位置	OSD 各通道支持		Cover Layer 个数	OSD COLOR FORMAT 支持						
		硬件 layer 个数	最大 layer 个数		ARGB1555	ARGB4444	I2	I4	I8	RGB565	ARGB8888
SSC336D SSC336Q SSC339G	VPE PORT 0	8	128	4	支持	支持	支持	支持	支持	NA	NA
	VPE PORT 1	8	128	4	支持	支持	支持	支持	支持	NA	NA
	VPE PORT 2	NA	NA	4	NA	NA	NA	NA	NA	NA	NA
	VPE PORT 3	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
	DIVP PORT 0	8	128	4	支持	支持	支持	支持	支持	NA	NA
	LDC PORT 0	8	128	NA	支持	支持	支持	支持	支持	NA	NA

- **举例**

```
MI_S32 s32Result = 0;
MI_RGN_HANDLE hHandle = 0;
MI_RGN_ChnPort_t stChnPort;
MI_RGN_ChnPortParam_t stChnAttr;

memset(stChnPort, 0, sizeof(MI_RGN_ChnPort_t));
memset(stChnAttr, 0, sizeof(MI_RGN_ChnPortParam_t));
stChnPort.eModId = E_MI_RGN_MODID_VPE;
stChnPort.s32DevId = 0;
stChnPort.s32ChnId = 0;
stChnPort.s32OutputPortId = 0;
stChnAttr.bShow = TRUE;
stChnAttr.stPoint.u32X = 0;
stChnAttr.stPoint.u32Y = 0;
stChnAttr.unPara. stCoverChnPort.u32Layer = 0;
stChnAttr.unPara. stCoverChnPort.stSize.u32Width = 200;
stChnAttr.unPara. stCoverChnPort.stSize.u32Height = 200;
stChnAttr.unPara. stCoverChnPort.u32Color = 0;

s32Result = MI_RGN_AttachToChn(hHandle, &stChnPort, &stChnAttr);
if (s32Result != MI_RGN_OK)
{
            return s32Result;
}

s32Result = MI_RGN_DetachFromChn(hHandle, &stChnPort);
if (s32Result != MI_RGN_OK)
{
            return s32Result;
}
```

- **相关主题**

MI_RGN_DetachFromChn

2.1.8 MI_RGN_DetachFromChn

➢ **功能**

将区域从通道中撤出。

➢ **语法**

MI_S32 MI_RGN_DetachFromChn(MI_RGN_HANDLE hHandle,
MI_RGN_ChnPort_t *pstChnPort);

➢ **形参**

参数名称	参数含义	输入/输出
hHandle	区域句柄号。 取值范围：[0, MI_RGN_MAX_HANDLE)。	输入
pstChnPort	通道端口结构体指针。	输入

➢ **返回值**

返回值
- MI_RGN_OK 成功。
- 非 MI_RGN_OK 失败，参照返回值。

➢ **依赖**

- 头文件：mi_sys.h、mi_rgn.h。
- 库文件：

※ **注意**

- 区域必须已创建。
- 通道结构体指针不能为空。
- 支持多次调用。
- 被叠加区域的通道或组（如 VENC，VPE 等）销毁前，需要调用本接口将区域
- 通道或组中撤出。

➢ **举例**

参见 MI_RGN_AttachToChn 举例。

➢ **相关主题**

MI_RGN_AttachToChn

2.1.9 MI_RGN_SetDisplayAttr

➢ **功能**

设置区域的通道显示属性。

➢ 语法

MI_S32 MI_RGN_SetDisplayAttr(MI_RGN_HANDLE hHandle,
MI_RGN_ChnPort_t *pstChnPort,
MI_RGN_ChnPortParam_t *pstChnPortAttr);

➢ 形参

参数名称	参数含义	输入/输出
hHandle	区域句柄号。 取值范围：[0, MI_RGN_MAX_HANDLE)。	输入
pstChnPort	通道端口结构体指针。	输入
pstChnPortAttr	区域通道端口显示属性指针。	输入

➢ 返回值

返回值
- MI_RGN_OK 成功。
- 非 MI_RGN_OK 失败，参照返回值。

➢ 依赖

- 头文件：mi_sys.h、mi_rgn.h。
- 库文件：

※ 注意

- 区域必须已创建。
- 建议先获取属性，再设置。
- 通道结构体指针不能为空。
- 区域通道显示属性指针不能为空。
- 区域必须先叠加到通道上。

➢ 举例

```
MI_S32 s32Result = 0;
MI_RGN_HANDLE hHandle = 0;
MI_RGN_ChnPort_t stChnPort;
MI_RGN_ChnPortParam_t stChnAttr;

stChnPort.eModId = E_MI_RGN_MODID_VPE;
stChnPort.s32DevId = 0;
stChnPort.s32ChnId = 0;
stChnPort.s32OutputPortId = 0;
s32Result = MI_RGN_GetDisplayAttr(hHandle, &stChnPort, &stChnAttr);
if (s32Result != MI_RGN_OK)
{
    return s32Result;
}


stChnAttr.bShow = TRUE;
stChnAttr.stPoint.u32X = 0;
stChnAttr.stPoint.u32Y = 0;
```

```
stChnAttr.stCoverPara.u32Layer = 0;
stChnAttr.stCoverPara.stSize.u32Width = 200;
stChnAttr.stCoverPara.stSize.u32Height = 200;
stChnAttr.stCoverPara.u32Color = 0;

s32Result = MI_RGN_SetDisplayAttr(hHandle, &stChnPort, &stChnAttr);
if (s32Result != MI_RGN_OK)
{
    return s32Result;
}
```

➢ 相关主题

MI_RGN_GetDisplayAttr

2.1.10 MI_RGN_GetDisplayAttr

➢ 功能

获取区域的通道显示属性。

➢ 语法

MI_S32 MI_RGN_GetDisplayAttr(MI_RGN_HANDLE hHandle,
MI_RGN_ChnPort_t *pstChnPort,
MI_RGN_ChnPortParam_t *pstChnPortAttr);

➢ 形参

参数名称	参数含义	输入/输出
hHandle	区域句柄号。 取值范围：[0, MI_RGN_MAX_HANDLE)。	输入
pstChnPort	通道端口结构体指针。	输入
pstChnPortAttr	区域通道端口显示属性指针。	输出

➢ 返回值

返回值
- MI_RGN_OK 成功。
- 非 MI_RGN_OK 失败，参照返回值。

➢ 依赖

- 头文件：mi_sys.h、mi_rgn.h。
- 库文件：

※ 注意

- 区域必须已创建。
- 通道结构体指针不能为空。
- 区域通道显示属性指针不能为空。

➢ **举例**

请参见 MI_RGN_SetDisplayAttr 的举例。

➢ **相关主题**

MI_RGN_SetDisplayAttr

2.1.11 MI_RGN_GetCanvasInfo

➢ **功能**

获取区域的显示画布信息。

➢ **语法**

```
MI_S32 MI_RGN_GetCanvasInfo(MI_RGN_HANDLE hHandle,
                            MI_RGN_CanvasInfo_t* pstCanvasInfo);
```

➢ **形参**

参数名称	参数含义	输入/输出
hHandle	区域句柄号。 取值范围：[0, MI_RGN_MAX_HANDLE)。	输入
pstCanvasInfo	区域显示画布信息。	输出

➢ **返回值**

返回值
- MI_RGN_OK 成功。
- 非 MI_RGN_OK 失败，参照返回值。

➢ **依赖**

- 头文件：mi_sys.h、mi_rgn.h。
- 库文件：

※ **注意**

- 区域必须已创建。
- 本接口与 MI_RGN_SetBitMap 功能类似，主要用于 overlay 类型
- 导入位图数据。本接口相对于 MI_RGN_SetBitMap 接口，用户可以直接更新
- 显示内部画布数据，节省一次内存拷贝和一张画布内存。
- 本接口用于获取区域对应的画布信息，在得到画布地址之后，用户可直接对画布进行操作，譬如：将 bmp 数据直接填写到该画布中。然后通过调用 MI_RGN_UpdateCanvas 接口，更新显示画布数据。
- 本接口与 MI_RGN_SetBitMap 接口互斥。如果已经使用了本接口，那么在调用 MI_RGN_UpdateCanvas 前，调用 MI_RGN_SetBitMap 不生效。

➢ 举例

```
MI_RGN_HANDLE hHandle;
MI_RGN_Attr_t stRegion;
MI_RGN_PaletteTable_t stPaletteTable;
MI_RGN_CanvasInfo_t stCanvasInfo;

memset(&stPaletteTable, 0, sizeof(MI_RGN_PaletteTable_t));
stPaletteTable.astElement[0].u8Alpha = 0;
stPaletteTable.astElement[0].u8Red = 255;
stPaletteTable.astElement[0].u8Green = 255;
stPaletteTable.astElement[0].u8Blue = 255;
if (MI_RGN_OK != MI_RGN_Init(&stPaletteTable))
{
  printf("Init error!\n");

  return -1;
}
hHandle = 10;
stRegion.eType = E_MI_RGN_TYPE_OSD;
stRegion.stOsdInitParam.ePixelFmt = E_MI_RGN_PIXEL_FORMAT_ARGB1555;
stRegion.stOsdInitParam.stSize.u32Width = 100;
stRegion.stOsdInitParam.stSize.u32Height = 100;
if (MI_RGN_OK != MI_RGN_Init(hHandle, &stRegion))
{
  printf("Create handle error!\n");

  return -1;
}
if (MI_RGN_OK != MI_RGN_Create(hHandle, &stRegion))
{
  printf("Create handle error!\n");

  return -1;
}
FILE *pFile = fopen("100X100.argb1555", "rb");
if (pFile == NULL)
{
  printf("open file failed \n");
  MI_RGN_Destroy(hHandle);

  return -1;
}
if (MI_RGN_GetCanvas(hHandle, &stCanvasInfo) != MI_RGN_OK)
{
  return s32Result;
}

for (int i = 0; i < 100; i++)
{
  fread((MI_U8*)stCanvasInfo.virtAddr + i * stCanvasInfo.u32Stride, 1, 100 *
2, pFile);
}
fclose(pFile);
if (MI_RGN_UpdateCanvas(hHandle) != MI_RGN_OK)
{
  return s32Result;
}
```

➢ 相关主题

MI_RGN_UpdateCanvas

2.1.12 MI_RGN_UpdateCanvas

➢ 功能

更新显示画布，若画布有叠加到通道上则更新显示画布，若没有叠加到通道，则在执行叠加操作后才会在通道上显示出画布的内容。

➢ 语法

MI_S32 MI_RGN_UpdateCanvas(MI_RGN_HANDLE hHandle);

➢ 形参

参数名称	参数含义	输入/输出
hHandle	区域句柄号。 取值范围：[0, MI_RGN_MAX_HANDLE)。	输入

➢ 返回值

返回值
- MI_RGN_OK 成功。
- 非 MI_RGN_OK 失败，参照返回值。

➢ 依赖

- 头文件：mi_sys.h、mi_rgn.h。
- 库文件：

※ 注意

- 区域必须已创建。
- 本接口配合 MI_RGN_GetCanvasInfo 使用。主要用于画布内存数据更新之后，进行画布切换显示。
- 本接口必须与 MI_RGN_GetCanvasInfo 成对调用，同一个 handler 先执行 MI_RGN_GetCanvasInfo 获取画布的内存指针，待绘图操作完成后执行本接口，若没有先执行 MI_RGN_GetCanvasInfo，则会返回 MI_ERR_RGN_NOT_PERM。
- 当多个 handler 叠加到同一个通道上显示，并且都会使用 MI_RGN_GetCanvasInfo 及本接口进行绘图操作，在这个通道上的所有绘图操作可以在同一个线程上执行，若在不同线程上，必须使用锁保护起来，否则有可能会出现 OSD 闪烁，或者 osd 不刷新的问题。
- 以上多线程的情况下锁的使用详细说明如下：
  rgn 的 handle 假设有 handler0 和 handler1 都贴到同一个通道上
  分别两个线程
  handler0
  T0_0 = getcanvas
  T0_1 = update
  handler1
  T1_0 = getcanvas

T1_1 = update
按照时间 T 的顺序执行
有问题的情况：
T0_0 -> T1_0 -> T0_1 -> T1_1，并伴随底层 error 的打印：“Front buf state error!!!”。
加 mutex 后，能够正常。调整后的时序：
lock-> T0_0 -> T0_1 -> unlock -> lock -> T1_0 -> T1_1 -> unlock

➢ **举例**

请参见 MI_RGN_GetCanvasInfo 的举例。

➢ **相关主题**

MI_RGN_GetCanvasInfo

2.1.13 MI_RGN_ScaleRect

➢ **功能**

OSD 区域放大。

➢ **语法**

MI_S32 MI_RGN_ScaleRect(MI_RGN_ChnPort_t *pstChnPort, MI_RGN_Size_t *pstCanvasSize, MI_RGN_Size_t *pstScreenSize);

➢ 型参

参数名称	参数含义	输入/输出
pstChnPort	通道端口结构体指针。	输入
pstCanvasSize	放大前 OSD 所在图层宽高的结构体指针。 要求宽按 16byte 对齐，如 I4 格式，需按 32pixel 对齐。	输入
pstScreenSize	放大后 OSD 所在屏幕宽高的结构体指针。	输入

➢ **返回值**

返回值
- MI_RGN_OK　成功。
- 非 MI_RGN_OK　失败，参照返回值。

➢ **依赖**

- 头文件：mi_sys.h、mi_rgn.h。
- 库文件：

※ **注意**

- 区域必须已创建。
- 放大后的屏幕宽高不能超过视频帧的宽高，视频帧的最大宽高不超过 3840x2160。

- 放大前的图层宽高不能超过放大后的屏幕宽高。
- 放大前的宽需按 16byte 对齐。硬件上要求放大前宽按 2pixel 对齐，但当 OSD 区域出现在显示区域边缘时，放大处理后可能会与屏幕边缘存在间隔，所以要求放大前的宽也按 Gwin 对齐要求对齐。
- 做 OSD 放大时，OSD 反色功能失效。
- OSD 区域放大功能目前只有 SSC328Q、SSC329D、SSC326D 能够支持，其他的芯片暂不支持。

➢ **举例**

```
MI_S32 s32Result = 0;
MI_RGN_ChnPort_t stChnPort;
MI_RGN_Size_t stCanvasSize;
MI_RGN_Size_t stScreenSize;

memset(&stChnPort, 0, sizeof(MI_RGN_ChnPort_t));
memset(&stCanvasSize, 0, sizeof(MI_RGN_Size_t));
memset(&stScreenSize, 0, sizeof(MI_RGN_Size_t));

stChnPort.eModId = E_MI_RGN_MODID_VPE;
stChnPort.s32DevId = 0;
stChnPort.s32ChnId = 0;
stChnPort.s32OutputPortId = 0;
stCanvasSize.u32Width = 1280;
stCanvasSize.u32Height = 720;
stScreenSize.u32Width = 1920;
stScreenSize.u32Height = 1080;

s32Result = MI_RGN_ScaleRect(&stChnPort, &stCanvasSize, &stScreenSize);
if (s32Result != MI_RGN_OK)
{
return s32Result;
}
```

➢ **相关主题**

无。

3. RGN 数据类型

视频前处理相关数据类型、数据结构定义如下：

MI_RGN_MAX_HANDLE	定义区域的最大句柄数
MI_RGN_OSD_MAX_NUM	Osd 支持最大区域个数
MI_RGN_COVER_MAX_NUM	Cover 支持最大区域个数
MI_RGN_VPE_MAX_CH_NUM	Vpe 支持最大通道数
MI_RGN_DIVP_MAX_CH_NUM	Divp 支持最大通道数
MI_RGN_VPE_PORT_MAXNUM	Vpe 最大输出端口数
MI_RGN_DIVP_PORT_MAXNUM	Divp 最大输出端口数
MI_RGN_MAX_PALETTE_TABLE_NUM	颜色表最大元素个数
MI_RGN_HANDLE	定义区域句柄
MI_RGN_Type_e	定义区域类型
MI_RGN_PixelFormat_e	RGB or Index 格式
MI_RGN_InvertColorMode_e	反色模式
MI_RGN_Size_t	大小信息
MI_RGN_OsdInitParam_t	定义 osd 区域属性结构体
MI_RGN_PaletteElememt_t	颜色元素
MI_RGN_PaletteTable_t	颜色表
MI_RGN_Attr_t	定义区域类型结构体
MI_RGN_Bitmap_t	bitmap 属性
MI_RGN_ModId_e	定义模块 ID 枚举类型
MI_RGN_ChnPort_t	定义模块设备通道结构体
MI_RGN_Point_t	定义坐标信息结构体
MI_RGN_CoverChnPortParam_t	定义遮挡区域的通道显示属性
MI_RGN_ChnPortParam_t	定义区域通道显示属性结构体
MI_RGN_CanvasInfo_t	定义画布信息结构体
MI_RGN_OsdInvertColorAttr_t	反色属性

3.1. MI_RGN_ MAX_HANDLE

➢ 说明

定义区域的最大句柄。

➢ 定义

#define MI_RGN_MAX_HANDLE 1024

➢ 成员

无

※ 注意事项

无

➢ 相关数据类型及接口

无

3.2. MI_RGN_MAX_PALETTE_TABLE_NUM

➢ 说明

颜色表最大元素个数。

➢ 定义

#define MI_RGN_MAX_PALETTE_TABLE_NUM 256

➢ 成员

无

※ 注意事项

无

➢ 相关数据类型及接口

无

3.3. MI_RGN_HANDLE

➢ 说明

定义区域句柄。

➢ 定义

typedef MI_U32 MI_RGN_HANDLE;

➢ 成员

成员名称	描述
MI_RGN_HANDLE	区域句柄。

※ 注意事项

无

➢ 相关数据类型及接口

无

3.4. MI_RGN_Type_e

➢ 说明

定义区域类型。

➢ 定义

```
typedef enum
{
    E_MI_RGN_TYPE_OSD = 0,
    E_MI_RGN_TYPE_COVER,
    E_MI_RGN_TYPE_MAX
} MI_RGN_Type_e;
```

➢ 成员

成员名称	描述
E_MI_ REG_OSD	视频叠加区域。
E_MI_REG_COVER	视频遮挡区域。

※ 注意事项

无

➢ 相关数据类型及接口

无

3.5. MI_RGN_PixelFormat_e

➢ 说明

定义叠加区域属性结构体。

➢ 定义

```
typedef enum
{
    E_MI_RGN_PIXEL_FORMAT_ARGB1555 = 0,
    E_MI_RGN_PIXEL_FORMAT_ARGB4444,
    E_MI_RGN_PIXEL_FORMAT_I2,
    E_MI_RGN_PIXEL_FORMAT_I4,
    E_MI_RGN_PIXEL_FORMAT_I8,
    E_MI_RGN_PIXEL_FORMAT_RGB565,
    E_MI_RGN_PIXEL_FORMAT_ARGB8888,
    E_MI_RGN_PIXEL_FORMAT_MAX
} MI_RGN_PixelFormat_e;
```

➢ 成员

成员名称	描述
E_MI_RGN_PIXEL_FORMAT_ARGB1555	ARGB1555 格式
E_MI_RGN_PIXEL_FORMAT_ARGB4444	ARGB4444 格式
E_MI_RGN_PIXEL_FORMAT_RGBI2	I2 格式（两个 bit 表示，支援 4 种颜色，调色板查色）
E_MI_RGN_PIXEL_FORMAT_RGBI4	I4 格式（4 个 bit 表示，支援 16 种颜色，调色板查色）
E_MI_RGN_PIXEL_FORMAT_I8	I4 格式（8 个 bit 表示，支援 256 种颜色，调色板查色）
E_MI_RGN_PIXEL_FORMAT_RGB565	RGB565
E_MI_RGN_PIXEL_FORMAT_ARGB8888	ARGB8888 格式

※ 注意事项

每个 chip 支持的图像格式不一样，API 种极大化地罗列出了所有的图像格式，但是有些格式存在 API 不支援的情况，如果使用者需要 chip 支援的情况，请查看 region procfs。
使用命令： echo getcap > /proc/mi_modules/mi_rgn/mi_rgn0。
使用者无法设定 index 0 的调色盘，I2/I4/I8 这些格式的 index 0 被底层 driver 用作 color key，表示这种颜色不被硬件识别，所以当全 0 的数据叠加到通道上时，是不显示任何颜色的。
Color key 的数值可以在 procfs 的 getcap 中查看，colorkey 的数值是一个 16bit 整型，它的高 8 位和低 8 位是一样的值，当使用 Index 类型的 colorformat 时，对内存数据进行 memset 0 即可让硬件不识别，当使用 RGB 或者 ARGB 格式时无论使用的是何种排列，对内存数据进行 memeset (colorkey & 0xFF)数值即可。

➢ 相关数据类型及接口

无

3.6. MI_RGN_InvertColorMode_e

➢ 说明

定义反色区域的反色模式。

➢ 定义

```
typedef enum
{
    E_MI_RGN_ABOVE_LUMA_THRESHOLD = 0,
    E_MI_RGN_BELOW_LUMA_THRESHOLD,
    E_MI_RGN_LUMA_THRESHOLD_BUTT
} MI_RGN_InvertColorMode_e;
```

➢ 成员

成员名称	描述
E_MI_RGN_ABOVE_LUMA_THRESHOLD	大于阈值做反色。
E_MI_RGN_BELOW_LUMA_THRESHOLD	小于阈值做反色。

※ 注意事项

无

➢ 相关数据类型及接口

无

3.7. MI_RGN_AlphaMode_e

➢ 说明

设定 Osd Alpha 显示模式。

➢ 定义

```
typedef enum
{
    E_MI_RGN_PIXEL_ALPHA = 0,
    E_MI_RGN_CONSTANT_ALPHA
}MI_RGN_AlphaMode_e;
```

➢ 成员

成员名称	描述
E_MI_RGN_PIXEL_ALPHA	Osd 显示每个 pixel 对应的 alpha 效果，例如 argb1555/argb4444/arbg8888/i2/i4/i8 这些格式都能支持 pixel alpha，rgb565 则不会生效。
E_MI_RGN_CONSTANT_ALPHA	Osd 硬件会忽略图像格式中的 alpha 位，使用统一的值设定 alpha 值。例如 rgb565 能设定其透明度 。

※ 注意事项

无

➢ 相关数据类型及接口

无

3.8. MI_RGN_Size_t

➢ 说明

定义大小信息结构体。

➢ 定义

```
typedef struct MI_RGN_Size_s
{
    MI_U32 u32Width;
    MI_U32 u32Height;
} MI_RGN_Size_t;
```

➢ 成员

成员名称	描述
u32Width	宽度
u32Height	高度

※ 注意事项

无

➢ 相关数据类型及接口

无

3.9. MI_RGN_OsdInvertColorAttr_t

➢ 说明

定义反色区域属性结构体。

➢ 定义

```
typedef struct MI_RGN_OsdInvertColorAttr_s
{
    MI_BOOL bEnableColorInv;
    MI_RGN_InvertColorMode_e eInvertColorMode;
    MI_U16 u16LumaThreshold;
    MI_U16 u16WDivNum;
    MI_U16 u16HDivNum;
} MI_RGN_OsdInvertColorAttr_t;
```

➢ 成员

成员名称	描述
bEnableColorInv	反色使能。
eInvertColorMode	反色模式。
u16LumaThreshold	发色阈值。取值范围为 0~255
u16WDivNum	反色区域横向切割数。反色区域宽要求能被横向分割数整除。
u16HDivNum	反色区域纵向切割数。反色区域高要求能被纵向分割数整除。

※ 注意事项

反色仅在 region 可显示，开启反色且设置参数合法的情况下才会起作用。
反色区域的位置和大小与 region 相同，建议根据实际需要做反色的区域大小来创建 region。
反色区域可切块的数量为 1~2048，每个切割块的最大范围为 64pixel*64pixel。u16WdivNum 与 u16HDivNum 的乘积不能超过 2048。也不能设置过小，使切割块的范围超出最大限制。同时还需满足宽高能分别被横向切割数和纵向切割数整除，否则反色无效果。
同一个 ChnPort 绑定多个 region，当多个 region 开启反色时，各个 region 的反色模式和阈值须设为一致。若反色模式和阈值设置不一致时，实际反色模式与阈值与最上层显示且开启反色的 region 的设定相同。
反色功能目前只有 SSC328Q、SSC329D、SSC326D 能够支持，其他的芯片暂不支持。

➢ 相关数据类型及接口

无

3.10. MI_RGN_OsdAlphaAttr_t

➢ 说明

定义 Osd Alpha 属性的结构体体。

➢ 定义

```
typedef struct MI_RGN_OsdAlphaAttr_s
{
    MI_RGN_AlphaMode_e eAlphaMode;
    MI_RGN_AlphaModePara_u stAlphaPara;
}MI_RGN_OsdAlphaAttr_t;
```

➢ 成员

成员名称	描述
eAlphaMode	Osd Alpha 的使用模式。
stAlphaPara	模式对应的参数。

※ 注意事项

无

➢ 相关数据类型及接口

无

3.11. MI_RGN_OsdInitParam_t

➢ 说明

定义叠加区域属性结构体。

➢ 定义

```
typedef struct MI_RGN_OsdInitParam_s
{
    MI_RGN_PixelFormat_e ePixelFmt;
    MI_RGN_Size_t stSize;
}MI_RGN_OsdInitParam_t;
```

➢ 成员

成员名称	描述
ePixelFmt	像素格式。
stSize	区域的宽高。 宽最大不超过 3840 高最大不超过 2160。

※ 注意事项

ePixelFmt、stSize 只在调用 MI_RGN_AttachToChn 后，MI_RGN_DetachFromChn 之前为静态变量。

➢ 相关数据类型及接口

无

3.12. MI_RGN_PaletteElement_t

➢ 说明

定义

➢ 定义

```
typedef struct MI_RGN_PaletteElement_s
{
    MI_U8 u8Alpha;
    MI_U8 u8Red;
    MI_U8 u8Green;
    MI_U8 u8Blue;
}MI_RGN_PaletteElement_t;
```

➢ 成员

成员名称	描述
u8Alpha	透明度
u8Red	红色
u8Green	绿色
u8Blue	蓝色

※ 注意事项

无

➢ 相关数据类型及接口

无

3.13. MI_RGN_PaletteTable_t

➢ 说明

定义

➢ 定义

```
typedef struct MI_RGN_PaletteTable_s
{
    MI_RGN_PaletteElement_t astElement[MI_RGN_MAX_PALETTE_TABLE_NUM];
}MI_RGN_PaletteTable_t;
```

➢ 成员

成员名称	描述
astElement	颜色元素

※ 注意事项

无

➢ 相关数据类型及接口

无

3.14. MI_RGN_Attr_t

➢ 说明

定义区域属性结构体。

➢ 定义

```
typedef struct MI_RGN_Attr_s
{
    MI_RGN_Type_e eType;
    MI_RGN_OsdInitParam_t stOsdInitParam;
}MI_RGN_Attr_t;
```

➢ 成员

成员名称	描述
eType	区域类型。
stOsdInitParam	osd 区域属性。

※ 注意事项

无

➢ 相关数据类型及接口

无

3.15. MI_RGN_Bitmap_t

➢ 说明

定义位图图像信息结构。

➢ 定义

```
typedef struct MI_RGN_Bitmap_s
{
    MI_RGN_PixelFormat_e ePixelFormat;
    MI_RGN_Size_t stSize;
    void *pData;
} MI_RGN_Bitmap_t;
```

➢ 成员

成员名称	描述
ePixelFormat	位图像素格式
stSize	位图宽度，高度，stride
pData	位图数据

※ 注意事项

无

➢ 相关数据类型及接口

无

3.16. MI_RGN_ModId_e

➢ 说明

定义模块 ID 枚举类型。

➢ 定义

```
typedef enum
{
```

```
    E_MI_RGN_MODID_VPE = 0,
    E_MI_RGN_MODID_DIVP,
     E_MI_RGN_MODID_LDC,
    E_MI_RGN_MODID_MAX
}MI_RGN_ModId_e;
```

- 成员

成员名称	描述
E_MI_RGN_MODID_VPE	VPE 模块 ID
E_MI_RGN_MODID_DIVP	DIVP 模块 ID
E_MI_RGN_MODID_LDC	LDC 模块 ID

※ 注意事项

无

- 相关数据类型及接口

无

3.17. MI_RGN_ChnPort_t

➢ 说明

定义模块设备通道结构体。

➢ 定义

```
typedef struct MI_RGN_ChnPort_s
{
    MI_RGN_ModId_e  eModId;
    MI_S32 s32DevId;
    MI_S32 s32ChnId;
    MI_S32 s32OutputPortId;
}MI_RGN_ChnPort_t;
```

➢ 成员

成员名称	描述
eModId	模块号
s32DevId	设备号
s32ChnId	通道号
s32OutputPortId	输出端口 id

※ 注意事项

无

➢ 相关数据类型及接口

无

3.18. MI_RGN_Point_t

➢ 说明

定义坐标信息结构体。

➢ 定义

```
typedef struct MI_RGN_Point_s
{
    MI_U32 u32X;
    MI_U32 u32Y;
}MI_RGN_Point_t;
```

➢ 成员

成员名称	描述
u32X	横坐标
u32Y	纵坐标

※ 注意事项

无

➢ 相关数据类型及接口

无

3.19. MI_RGN_CoverChnPortParam_t

➢ 说明

定义遮挡区域的通道显示属性。

➢ 定义

```
typedef struct MI_RGN_CoverChnPortParam_s
{
    MI_U32 u32Layer;
    MI_RGN_Size_t stSize;
    MI_U32 u32Color;
}MI_RGN_CoverChnPortParam_t;
```

➢ 成员

成员名称	描述
u32Layer	区域层次，数值低的在底层
stSize	Cover 宽，高，stride
u32Color	颜色，VYU444

※ 注意事项

无

➢ 相关数据类型及接口

无

3.20. MI_RGN_OsdChnPortParam_t

➢ 说明

定义 OSD 区域的通道显示属性。

➢ 定义

```
typedef struct MI_RGN_OsdChnPortParam_s
{
    MI_U32 u32Layer;
    MI_RGN_OsdAlphaAttr_t stOsdAlphaAttr;
    MI_RGN_OsdInvertColorAttr_t stColorInvertAttr;
}MI_RGN_OsdChnPortParam_t;
```

➢ 成员

成员名称	描述
u32Layer	区域层次，数值低的在底层

※ 注意事项

无

➢ 相关数据类型及接口

无

3.21. MI_RGN_OsdArgb1555Alpha_t

➢ 说明

Argb1555 格式的前景、背景 Alpha 设定。

➢ 定义

```
typedef struct MI_RGN_OsdArgb1555Alpha_s
{
    MI_U8 u8BgAlpha;
    MI_U8 u8FgAlpha;
}MI_RGN_OsdArgb1555Alpha_t;
```

➢ 成员

成员名称	描述
u8BgAlpha	背景 Alpha，Alpha bit 为 0 时对应的 Alpha 值，取值范围 0~0xFF。
u8FgAlpha	前景 Alpha， Alpha bit 为 1 时对应的 Alpha 值，取值范围 0~0xFF。

※ 注意事项

无

➢ 相关数据类型及接口

无

3.22. MI_RGN_ChnPortParamUnion_u

➢ 说明

定义 Region 属性的联合体。

➢ 定义

```
typedef union
{
    MI_RGN_CoverChnPortParam_t stCoverChnPort;
    MI_RGN_OsdChnPortParam_t stOsdChnPort;
} MI_RGN_ChnPortParamUnion_u;
```

➢ 成员

成员名称	描述
stCoverChnPort	遮挡区域在通道上的属性设置
stOsdChnPort	Osd 区域在通道上的属性设置

※ 注意事项

无

➢ 相关数据类型及接口

无

3.23. MI_RGN_AlphaModePara_u

➢ 说明

定义 Osd Alpha Mode 参数联合体。

➢ 定义

```
typedef union
{
    MI_RGN_OsdArgb1555Alpha_t stArgb1555Alpha;
    MI_U8 u8ConstantAlpha;
} MI_RGN_AlphaModePara_u;
```

➢ 成员

成员名称	描述
stArgb1555Alpha	Pixel alpha 时 Argb1555 格式的前景 Alpha 及背景 Alpha 设定。
u8ConstantAlpha	Constant Alpha 时 Alpha 值设定，取值范围 0~0xFF。

※ 注意事项

无

➢ 相关数据类型及接口

无

3.24. MI_RGN_ChnPortParam_t

➢ 说明

定义区域通道显示属性结构体。

➢ 定义

```
typedef struct MI_RGN_ChnPortParam_s
{
    MI_BOOL bShow;
    MI_RGN_Point_t stPoint;
    MI_RGN_ChnPortParamUnion_u unPara;
} MI_RGN_ChnPortParam_t;
```

➢ 成员

成员名称	描述
bShow	区域是否显示。 取值范围：MI_TRUE 或者 MI_FALSE。 动态属性。
stPoint	区域起始点坐标。 当 RGN 类型为 OSD 时，要求 X 坐标按 1byte 对齐。
unPara	Region 通道显示属性。

※ 注意事项

无

➢ 相关数据类型及接口

无

3.25. MI_RGN_CanvasInfo_t

➢ 说明

定义画布信息结构体。

➢ 定义

```
typedef struct MI_RGN_CanvasInfo_s
{
    MI_PHY phyAddr;
    MI_VIRT virtAddr;
    MI_RGN_Size_t stSize;
    MI_U32 u32Stride;
    MI_RGN_PixelFormat_e ePixelFmt;
} MI_RGN_CanvasInfo_t;
```

➢ 成员

成员名称	描述
phyAddr	画布物理地址。
virtAddr	画布虚拟地址
stSize	画布尺寸
u32Stride	画布的 stride
ePixelFmt	画布的像素格式

※ 注意事项

无

➢ 相关数据类型及接口

无

4. 返回值

区域管理 API 返回值如表 3-1 所示。

表 3-1 区域管理 API 返回值

宏定义	描述
MI_RGN_OK	成功
MI_NOTICE_RGN_BUFFER_CHANGE	Buffer 发生改变。当设置属性时会发生，需要重新 remap.
MI_ERR_RGN_INVALID_HANDLE	非法的句柄
MI_ERR_RGN_INVALID_DEVID	设备 ID 超出合法范围
MI_ERR_RGN_INVALID_CHNID	通道组号错误或无效区域句柄
MI_ERR_RGN_ILLEGAL_PARAM	参数超出合法范围
MI_ERR_RGN_EXIST	重复创建已存在的设备、通道或资源
MI_ERR_RGN_UNEXIST	试图使用或者销毁不存在的设备、通道或者资源
MI_ERR_RGN_NULL_PTR	函数参数中有空指针
MI_ERR_RGN_NOT_CONFIG	模块没有配置
MI_ERR_RGN_NOT_SUPPORT	不支持的参数或者功能
MI_ERR_RGN_NOT_PERM	该操作不允许，如试图修改静态配置参数
MI_ERR_RGN_NOMEM	分配内存失败，如系统内存不足
MI_ERR_RGN_NOBUF	分配缓存失败，如申请的数据缓冲区太大
MI_ERR_RGN_BUF_EMPTY	缓冲区中无数据
MI_ERR_RGN_BUF_FULL	缓冲区中数据满
MI_ERR_RGN_BADADDR	地址非法
MI_ERR_RGN_BUSY	系统忙
MI_ERR_RGN_NOTREADY	系统没有初始化或没有加载相应模块