

MI RGN API

Version 2.07

© 2019 SigmaStar Technology Corp. All rights reserved.

SigmaStar Technology makes no representations or warranties including, for example but not limited to, warranties of merchantability, fitness for a particular purpose, non-infringement of any intellectual property right or the accuracy or completeness of this document, and reserves the right to make changes without further notice to any products herein to improve reliability, function or design. No responsibility is assumed by SigmaStar Technology arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others.

SigmaStar is a trademark of SigmaStar Technology Corp. Other trademarks or names herein are only for identification purposes only and owned by their respective owners.

REVISION HISTORY

Revision No.	Description	Date
2.03	<ul style="list-style-type: none">Initial release	04/12/2018
2.04	<ul style="list-style-type: none">Updated GetCanvas limitation	01/10/2019
2.05	<ul style="list-style-type: none">Removed GetCanvas limitation	01/25/2019
2.06	<ul style="list-style-type: none">Added OSD alpha setting	02/21/2019
2.07	<ul style="list-style-type: none">Added LDC module ID	09/16/2019

TABLE OF CONTENTS

REVISION HISTORY	i
TABLE OF CONTENTS.....	ii
1. Scope	错误!未定义书签。
1.1. Module description.....	1
1.2. Flow chart.....	2
1.3. Keyword.....	2
2. API LIST	3
2.1. API List	3
2.1.1 MI_RGN_Init.....	4
2.1.2 MI_RGN_DeInit.....	5
2.1.3 MI_RGN_Create	5
2.1.4 MI_RGN_Destroy	7
2.1.5 MI_RGN_GetAttr	7
2.1.6 MI_RGN_SetBitMap	8
2.1.7 MI_RGN_AttachToChn	9
2.1.8 MI_RGN_DetachFromChn	12
2.1.9 MI_RGN_SetDisplayAttr	13
2.1.10 MI_RGN_GetDisplayAttr.....	15
2.1.11 MI_RGN_GetCanvasInfo	16
2.1.12 MI_RGN_UpdateCanvas.....	17
3. RGN Data types.....	19
3.1. MI_RGN_MAX_HANDLE.....	20
3.2. MI_RGN_MAX_PALETTE_TABLE_NUM	20
3.3. MI_RGN_HANDLE	20
3.4. MI_RGN_Type_e	21
3.5. MI_RGN_PixelFormat_e.....	21
3.6. MI_RGN_InvertColorMode_e	23
3.7. MI_RGN_AlphaMode_e.....	23
3.8. MI_RGN_Size_t	24
3.9. MI_RGN_OsdInvertColorAttr_t.....	25
3.10. MI_RGN_OsdAlphaAttr_t	26
3.11. MI_RGN_OsdInitParam_t	26
3.12. MI_RGN_PaletteElement_t	27
3.13. MI_RGN_PaletteTable_t	27
3.14. MI_RGN_Attr_t.....	28
3.15. MI_RGN_Bitmap_t	28
3.16. MI_RGN_ModId_e	29
3.17. MI_RGN_ChnPort_t.....	30
3.18. MI_RGN_Point_t	30
3.19. MI_RGN_CoverChnPortParam_t.....	31
3.20. MI_RGN_OsdChnPortParam_t.....	31
3.21. MI_RGN_OsdArgb1555Alpha_t	32
3.22. MI_RGN_ChnPortParamUnion_u	33

3.23. MI_RGN_AlphaModePara_u	33
3.24. MI_RGN_ChnPortParam_t	34
3.25. MI_RGN_CanvasInfo_t.....	34
4. Return Value	36

1. SUMMARY

1.1. Module Description

The region module is a part of an internal stream processing in 'VPE', 'DIVP' and 'LDC' modules. It is supported by hardware module named GOP (graphic output path), it is a set of software interface abstracted by GOP characteristics. OSD (on screen display) or cover attaches to each channel in the way of using time sharing multiplexing.

The region module mainly provides the control and management functions of region resources. It includes the creation, destruction, acquisition and the setting of region attributes, acquisition and the setting of channel display attributes of the region, etc.

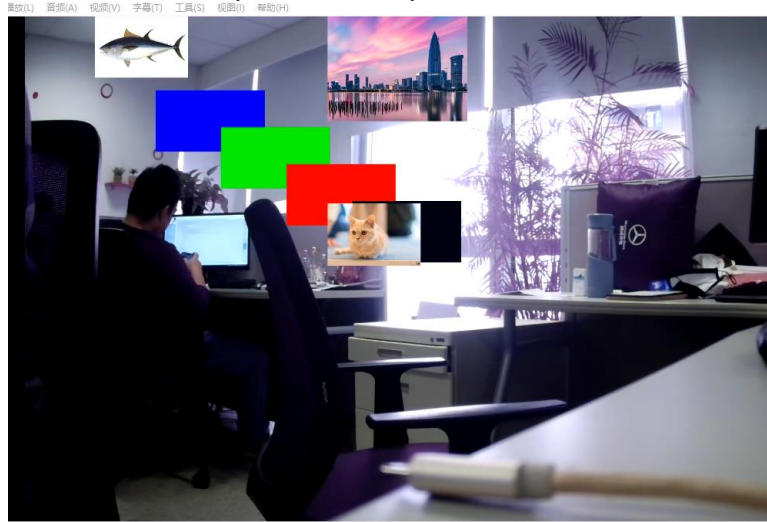
There are two kind of region attributions, one is called 'cover', which is order to cover the video by a designated color. The hardware driver of 'cover' only need to get the cover's position, size and the color, so that it will not get a display buffer.

The other region attribution is 'osd', also called 'overlay', It is same to 'cover', it can be attached to the display video area, but it is worked by display buffer, this kind of region can do a drew operation in point to point way, and the color format of 'osd' can be support ARGB/INDEX. User can choose the color format according to the practical application scenarios.

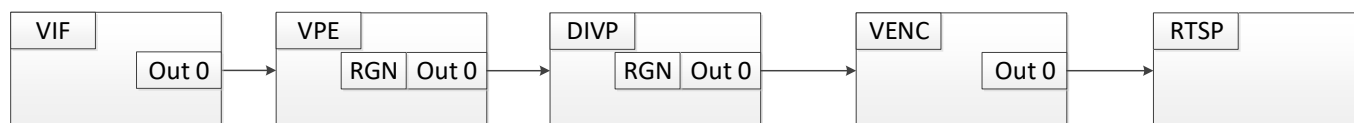
Currently osd format supports 'argb8888', 'argb1555', 'argb4444', 'rgb565', 'i2', 'i4', 'i8'. The format supporting conditions of each chip will be different. The following will be explained in detail. YUV format is not supported. I2, I4 and I8 are in bitmap format, and the memory data of a pixel is used as an index. Through the index, the color data in the color palette can be found, which the pixel displays is.

No matter 'cover' or 'osd', the display contents blending to the output of 'vpe/divp/lcd' module. Through experiments, when region display has been set, dump the output of 'vpe/divp/lcd' module into a file, and you can see the content of region through the tool.

'OSD' can display a picture, 'Cover' can only set different color, 'Osd' always display above 'cover' when they are attached the same path. This is a picture showed four 'cover' and three 'osd' attached to the same path.



1.2. Flow Chart



1.3. Keyword

- I2:
4 colors bitmap, 2 bits refer to an index, so that there are 4 colors. It can find the color by the index in palette.
- I4:
16 colors bitmap, 4 bits refer to an index and find the color by the index in palette.
- I8:
256 colors bitmap, 8 bits refer to an index and find the color by the index in palette.
- Palette:
The palette of bitmap, one color is represented by four 8bit variables: alpha, red, green and blue, there are totally 256 colors in palette with index range from 0 to 255.
- OSD:
Abbreviation of on-screen display, It is used to display some text, pictures, UI menus and other contents.
- GOP:
Abbreviation of graphic output path, it is a graphic layer above video.

2. API LIST

2.1. API List

API	Function
MI_RGN_Init	Initialization
MI_RGN_DeInit	De-initialization
MI_RGN_Create	Create region
MI_RGN_Destroy	Destroy region
MI_RGN_GetAttr	Get region attributes
MI_RGN_SetBitMap	Set bitmap region
MI_RGN_AttachToChn	Attach region to channel
MI_RGN_DetachFromChn	Detach region from channel
MI_RGN_SetDisplayAttr	Set display attributes of region
MI_RGN_GetDisplayAttr	Get display attributes of region
MI_RGN_GetCanvasInfo	Get canvas information of region
MI_RGN_UpdateCanvas	Update canvas information of region

2.1.1 MI_RGN_Init

➤ Function

Initialization

➤ Syntax

```
MI_S32 MI_RGN_Init(MI\_RGN\_PaletteTable\_t *pstPaletteTable);
```

➤ Parameters

Parameter	Description	Input / Output
pstPaletteTable	Pointer to the palette table	Input

➤ Return Value

- [MI_RGN_OK](#): Successful
- [MI_ERR_RGN_BUSY](#): Failed

➤ Requirement

- Header file: mi_sys.h, mi_rgn.h.
- Library file

➤ Note

The palette table can only be done once during initialization and cannot be set again.
It is not used for RGB color format.
The index of '0' in palette array is used for colorkey, which will not display anything.

➤ Example

```
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();
```

➤ Related APIs

[MI_RGN_DeInit](#)

2.1.2 MI_RGN_DeInit

- Function
De-initialization
- Syntax
`MI_S32 MI_RGN_DeInit();`
- Parameters
N/A.
- Return Value
 - [MI_RGN_OK](#): Successful
 - [MI_ERR_RGN_BUSY](#): Failed
- Requirement
 - Header file: `mi_sys.h`, `mi_rgn.h`.
 - Library file
- Note
N/A.
- Example
Refer to [MI_RGN_Init](#)
- Related APIs
[MI_RGN_Init](#)

2.1.3 MI_RGN_Create

- Function
Create region
- Syntax
`MI_S32 MI_RGN_Create(MI_RGN_HANDLE hHandle, MI_RGN_Attr_t *pstRegion);`
- Parameters

Parameter	Description	Input / Output
hHandle	Control code of region Must be an unused hHandle Range: [0, MI_RGN_MAX_HANDLE].	Input
pstRegion	Pointer to the attribute of region	Input
- Return Value
 - [MI_RGN_OK](#): Successful
 - Not [MI_RGN_OK](#): Failed. Please refer to [Return Value](#)

➤ Requirement

- Header file: mi_sys.h, mi_rgn.h.
- Library file

➤ Note

- This handle is specified by the user and has the same meaning as the ID.
- Repeated creation is not supported.
- The attribute of region must be legal. Refer to MI_RGN_Attr_t
- The pointer to attribute of region cannot be null.
- When creating a Cover, it only needs to assign the region type. Other attributes, such as location, hierarchy, etc., are assigned when the MI_RGN_AttachToChn interface is called.
- When creating a region, this only checks the basic parameters such as minimum width and height, maximum width and height. When this region is attached to the channel, more targeted parameters can be checked based on constraints of the supported type of channel module. Such as supported pixel formats, etc.

➤ Example

```
MI_S32 s32Result = 0;
MI_RGN_HANDLE hHandle = 0;
MI_RGN_Attr_t stRegion;
stRegion.eType = E_MI_RGN_TYPE_OSD;
stRegion.stOsdInitParam.ePixelFormat = 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;
}
```

➤ Related APIs

- [MI_RGN_Destroy](#)
- [MI_RGN_GetAttr](#)

2.1.4 MI_RGN_Destroy

➤ Function

Destroy region

➤ Syntax

MI_S32 MI_REG_Destroy ([MI_RGN_HANDLE](#) hHandle);

➤ Parameters

Parameter	Description	Input / Output
hHandle	Control code of region Range: [0, MI_RGN_MAX_HANDLE).	Input

➤ Return Value

- [MI_RGN_OK](#): Successful
- Not [MI_RGN_OK](#): Failed. Please refer to [Return Value](#)

➤ Requirement

- Header file: mi_sys.h, mi_rgn.h.
- Library file:

➤ Note

Region must have been created.

➤ Example

Refer to [MI_RGN_Create](#)

➤ Related APIs

[MI_RGN_Create](#)

2.1.5 MI_RGN_GetAttr

➤ Function

Get the attribute of region.

➤ Syntax

MI_S32 MI_RGN_GetAttr([MI_RGN_HANDLE](#) hHandle, [MI_RGN_Attr_t](#) *pstRegion);

➤ Parameters

Parameter	Description	Input / Output
hHandle	Control code of region Range: [0, MI_RGN_MAX_HANDLE).	Input
pstRegion	Pointer to the attribute of region	Output

- Return Value
 - [MI_RGN_OK](#): Successful
 - Not [MI_RGN_OK](#): Failed. Please refer to [Return Value](#)
- Requirement
 - Header file: mi_sys.h, mi_rgn.h.
 - Library file:
- Note
 - Region must have been created.
 - The pointer to the attribute of region cannot be null.
- Example

Refer to [MI_RGN_Create](#).
- Related APIs

N/A.

2.1.6 MI_RGN_SetBitMap

- Function

Set the region bitmap, that is, fill the region with bitmap.
- Syntax


```
MI_S32 MI_RGN_SetBitMap(MI\_RGN\_HANDLE hHandle,  
                        MI\_RGN\_Bitmap\_t *pstBitmap);
```

- Parameters

Parameter	Description	Input / Output
hHandle	Control code of region Range: [0, MI_RGN_MAX_HANDLE).	Input
pstBitmap	The pointer to the attribute of bitmap	Input

- Return Value
 - [MI_RGN_OK](#): Successful
 - Not [MI_RGN_OK](#): Failed. Please refer to [Return Value](#)
- Requirement
 - Header file: mi_sys.h, mi_rgn.h.
 - Library file:
- Note
 - Region must have been created.
 - The size of the supported bitmap is inconsistent with the size of the region.

- The bitmap is loaded from (0,0) of the region. When the bitmap is larger than the region, the image will be automatically cropped into region.
- The pixel format of the bitmap must be the same as the pixel format of the region.
- The pointer to the attribute of bitmap cannot be null.
- Support multiple calls.
- This interface is only valid for overlay.
- After calling [MI_RGN_GetCanvasInfo](#), calling this interface is invalid unless [MI_RGN_UpdateCanvas](#) is in effect.

➤ Example

```
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);
```

➤ Related APIs

N/A.

2.1.7 MI_RGN_AttachToChn

➤ Function

Attach region to channel

➤ Syntax

```
MI_S32 MI_RGN_AttachToChn(MI\_RGN\_HANDLE hHandle,  
                           MI\_RGN\_ChnPort\_t* pstChnPort,  
                           MI\_RGN\_ChnPortParam\_t*pstChnAttr);
```

➤ Parameters

Parameter	Description	Input / Output
hHandle	Control code of region Range: [0, MI_RGN_MAX_HANDLE).	Input
pstChnPort	Pointer to channel port When region type is Cover and the output port is VPE, only port 0 and port 3 are valid.	Input
pstChnAttr	Pointer to the attribute of channel port.	Input

➤ Return Value

- [MI_RGN_OK](#): Successful
- Not [MI_RGN_OK](#): Failed. Please refer to [Return Value](#)

➤ Requirement

- Header file: mi_sys.h, mi_rgn.h.
- Library file:

➤ Note

- Region must have been created.
- Pointer to channel port cannot be null.
- Pointer to the attribute of channel port cannot be null.
- There are two places in the region that can overlay: 4 output ports of MI_VPE module channel and the output port of MI_DIVP channel.
- If two or more OSD regions are attached to the same channel, the format of these OSD regions must be the same.
- Not only all the path has the ability to display region, the chart below list all the capabilities about region of all chips.
- 'Cover' can be worked by hardware layer. 'Osd' can be worked by software layer, which is implemented by software jigsaw.
- The channel id of 'vpe/divp/ldc' module do not has any limitation about 'Osd' and 'cover'.
- If the OSD attached on the channel is less than or equal to the number of hardware layers, all the hardware layers will be used, otherwise, the software jigsaw will be used.

Chip name	path	Osd capabilities.		Cover layer count	OSD COLOR FORMAT SUPPORT						
		Hardware layer count	Max layer count		ARGB1555	ARGB4444	I2	I4	I8	RGB565	ARGB8888
SSC328Q SSC329D SSC326D	VPE PORT 0	2	Default 8(Adjustable)	4	Support	Support	Support	Support	Support	Support	Support
	VPE PORT 1	2	Default 8(Adjustable)	4	Support	Support	Support	Support	Support	Support	Support
	VPE PORT 2	NA	NA	4	NA	NA	NA	NA	NA	NA	NA
	VPE PORT 3	2	Default 8(Adjustable)	NA	Support	Support	Support	Support	Support	Support	Support
	DIVP PORT 0	2	Default 8(Adjustable)	NA	Support	Support	Support	Support	Support	Support	Support
	LDC PORT 0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

Chip name	path	Osd capabilities.		Cover layer count	OSD COLOR FORMAT SUPPORT						
		Hardware layer count	Max layer count		ARGB1555	ARGB4444	I2	I4	I8	RGB565	ARGB8888
SSC325 SSC325DE SSC327DE	VPE PORT 0	4	128	4	Support	Support	Support	Support	Support	NA	NA
	VPE PORT 1	4	128	4	Support	Support	Support	Support	Support	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	Support	Support	Support	Support	Support	NA	NA
	LDC PORT 0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

Chip name	path	Osd capabilities.		Cover layer count	OSD COLOR FORMAT SUPPORT						
		Hardware layer count	Max layer count		ARGB1555	ARGB4444	I2	I4	I8	RGB565	ARGB8888
SSC335 SSC337DE	VPE PORT 0	4	128	4	Support	Support	Support	Support	Support	NA	NA
	VPE PORT 1	4	128	4	Support	Support	Support	Support	Support	NA	NA
	VPE PORT 2	4	128	4	Support	Support	Support	Support	Support	NA	NA
	VPE PORT 3	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
	DIVP PORT 0	4	128	4	Support	Support	Support	Support	Support	NA	NA
	LDC PORT 0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

Chip name	path	Osd capabilities.		Cover layer count	OSD COLOR FORMAT SUPPORT						
		Hardware layer count	Max layer count		ARGB1555	ARGB4444	I2	I4	I8	RGB565	ARGB8888
SSC336D SSC336Q SSC339G	VPE PORT 0	8	128	4	Support	Support	Support	Support	Support	NA	NA
	VPE PORT 1	8	128	4	Support	Support	Support	Support	Support	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	Support	Support	Support	Support	Support	NA	NA
	LDC PORT 0	8	128	NA	Support	Support	Support	Support	Support	NA	NA

➤ Example

```

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;
}

```

➤ Related APIs

[MI_RGN_DetachFromChn](#)

2.1.8 MI_RGN_DetachFromChn

➤ Function

Detach region from channel

➤ Syntax

```
MI_S32 MI_RGN_DetachFromChn(MI\_RGN\_HANDLE hHandle,  
                             MI\_RGN\_ChnPort\_t *pstChnPort);
```

➤ Parameters

Parameter	Description	Input / Output
hHandle	Control code of region Range: [0, MI_RGN_MAX_HANDLE).	Input
pstChnPort	The pointer to channel port	Input

➤ Return Value

- [MI_RGN_OK](#): Successful
- Not [MI_RGN_OK](#): Failed. Please refer to [Return Value](#)

➤ Requirement

- Header file: mi_sys.h, mi_rgn.h.
- Library file:

➤ Note

- Region must have been created.
- The pointer to channel port cannot be null.
- Support multiple calls.
- Before the overlaid channel or modules, such as VENC or VPE, is destroyed, it needs to call this interface to detach from region channel or module.

➤ Example

Refer to [MI_RGN_AttachToChn](#).

➤ Related APIs

[MI_RGN_AttachToChn](#)

2.1.9 MI_RGN_SetDisplayAttr

➤ Function

Set the display attribute of region.

➤ Syntax

```
MI_S32 MI_RGN_SetDisplayAttr(MI\_RGN\_HANDLE hHandle,  
                              MI\_RGN\_ChnPort\_t *pstChnPort,  
                              MI\_RGN\_ChnPortParam\_t *pstChnPortAttr);
```

➤ Parameters

Parameter	Description	Input / Output
hHandle	Control code of region Range: [0, MI_RGN_MAX_HANDLE).	Input
pstChnPort	The pointer to channel port	Input
pstChnPortAttr	The pointer to the parameter of channel port	Input

➤ Return Value

- [MI_RGN_OK](#): Successful
- Not [MI_RGN_OK](#): Failed. Please refer to [Return Value](#)

➤ Requirement

- Header file: mi_sys.h, mi_rgn.h.
- Library file:

➤ Note

- Region must have been created.
- It is recommended to obtain the attributes first and then set them.
- The pointer to channel port cannot be null.
- The pointer to the parameter of channel port cannot be null.
- The region must overlay with channel first.
- Static properties cannot be modified, dynamic properties can be modified. Refer to [MI_RGN_ChnPortParam_t](#)

➤ Example

```
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;
    }
```

➤ Related APIs

[MI_RGN_GetDisplayAttr](#)

2.1.10 MI_RGN_GetDisplayAttr

➤ Function

Get the display attribute of region.

➤ Syntax

```
MI_S32 MI_RGN_GetDisplayAttr(MI\_RGN\_HANDLE hHandle,
                             MI\_RGN\_ChnPort\_t *pstChnPort,
                             MI\_RGN\_ChnPortParam\_t *pstChnPortAttr);
```

➤ Parameters

Parameter	Description	Input / Output
hHandle	Control code of region Range: [0, MI_RGN_MAX_HANDLE).	Input
pstChnPort	The pointer to channel port	Input
pstChnPortAttr	The pointer to the parameter of channel port	Output

➤ Return Value

- [MI_RGN_OK](#): Successful
- Not [MI_RGN_OK](#): Failed. Please refer to [Return Value](#)

➤ Requirement

- Header file: mi_sys.h, mi_rgn.h.
- Library file:

➤ Note

- Region must have been created.
- The pointer to channel port cannot be null.
- The pointer to the parameter of channel port cannot be null.

➤ Example

Refer to [MI_RGN_SetDisplayAttr](#).

➤ Related APIs

[MI_RGN_SetDisplayAttr](#)

2.1.11 MI_RGN_GetCanvasInfo

➤ Function

Get the canvas information of region.

➤ Syntax

```
MI_S32 MI_RGN_GetCanvasInfo(MI\_RGN\_HANDLE hHandle,  
                             MI\_RGN\_CanvasInfo\_t* pstCanvasInfo);
```

➤ Parameters

Parameter	Description	Input / Output
hHandle	Control code of region Range: [0, MI_RGN_MAX_HANDLE).	Input
pstCanvasInfo	Pointer to the canvas information of region	Output

➤ Return Value

- [MI_RGN_OK](#): Successful
- Not [MI_RGN_OK](#): Failed. Please refer to [Return Value](#)

➤ Requirement

- Header file: mi_sys.h, mi_rgn.h.
- Library file:

➤ Note

- Region must have been created.
- This interface is similar to [MI_RGN_SetBitMap](#) and is mainly used for overlay types.
- Import bitmap data. This interface is relative to [MI_RGN_SetBitMap](#), users can update them directly.
- Display the internal data of canvas. Save one memory copy and one memory's allocation.
- This interface is used to obtain canvas information corresponding to a region. After getting the canvas address, the user can directly operate the canvas. For example: Fill bmp data directly into the canvas, then update the display canvas data by calling [MI_RGN_UpdateCanvas](#).
- This interface is mutually exclusive with [MI_RGN_SetBitMap](#). If this interface is already in use, calling [MI_RGN_SetBitMap](#) does not take effect until [MI_RGN_UpdateCanvas](#) is called.

➤ Example

```
MI_RGN_HANDLE hHandle;  
MI_RGN_Attr_t stRegion;  
MI_RGN_PaletteTable_t stPaletteTable;  
  
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.ePixelFormat = 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;
}
}

```

➤ Related APIs

[MI_RGN_UpdateCanvas](#)

2.1.12 MI_RGN_UpdateCanvas

➤ Function

Update canvas. If the canvas is attached to the path, it will display on video, otherwise it will display after attach.

➤ Syntax

MI_S32 MI_RGN_UpdateCanvas([MI_RGN_HANDLE](#) hHandle);

➤ Parameters

Parameter	Description	Input / Output
hHandle	Control code of region Range: [0, MI_RGN_MAX_HANDLE).	Input

➤ Return Value

- [MI_RGN_OK](#): Successful
- Not [MI_RGN_OK](#): Failed. Please refer to [Return Value](#)

➤ Requirement

- Header file: mi_sys.h, mi_rgn.h.
- Library file:

➤ Note

- Region must have been created.
- This interface is used with [MI_REG_GetCanvasInfo](#). It is mainly used for canvas switching display after the canvas memory data is updated.
- This interface must be called by pair with [MI_REG_GetCanvasInfo](#). It is used after user calls [MI_REG_GetCanvasInfo](#) to get the pointer of canvas and the draw operation has done. Without calling [MI_REG_GetCanvasInfo](#) first, this interface will return [MI_ERR_RGN_NOT_PERM](#).
- If two or more handler have attached to the same path, and all this handlers use [MI_REG_GetCanvasInfo](#) and this interface to draw, draw operation can work normally on the same thread. If it work on different thread, it must use 'mutex' lock to protect, or it may cause 'osd' flicker or disappearance.
- The detail note of multithreading's 'mutex' lock usage is as below:
Suppose 'handler0' and 'handler1', which run on a different thread, attach to the same path.

Let:

```

handler0
    T0_0 = getcanvas
    T0_1 = update
handler1
    T1_0 = getcanvas
    T1_1 = update

```

The wrong situation below is running in order of time, which is showing error log "Front buf state error!!!":

T0_0 -> T1_0 -> T0_1 -> T1_1.

After using 'mutex', it will work fine:

Lock-> T0_0 -> T0_1 -> unlock -> lock -> T1_0 -> T1_1 -> unlock

➤ Example

Refer to [MI_RGN_GetCanvasInfo](#).

➤ Related APIs

[MI_RGN_GetCanvasInfo](#)

3. RGN DATA TYPES

The data types of video pre-processing and their data structures are defined as follows:

MI_RGN_MAX_HANDLE	The maximum number of control codes of region.
MI_RGN_OSD_MAX_NUM	The maximum region number that OSD can support.
MI_RGN_COVER_MAX_NUM	The maximum region number that COVER can support.
MI_RGN_VPE_MAX_CH_NUM	The maximum channel number that VPE can support.
MI_RGN_DIVP_MAX_CH_NUM	The maximum channel number that DIVP can support.
MI_RGN_VPE_PORT_MAXNUM	The maximum port number that VPE can support.
MI_RGN_DIVP_PORT_MAXNUM	The maximum port number that DIVP can support.
MI_RGN_MAX_PALETTE_TABLE_NUM	The maximum number of palette table
MI_RGN_HANDLE	The definition for the control code of region
MI_RGN_Type_e	The enumeration of region type
MI_RGN_PixelFormat_e	The enumeration of Pixel format
MI_RGN_InvertColorMode_e	The enumeration of inverted mode
MI_RGN_Size_t	The structure of the region size
MI_RGN_OsdInitParam_t	The structure of the attribute of overlaid region
MI_RGN_PaletteElememt_t	The structure of Palette element
MI_RGN_PaletteTable_t	The structure of Palette table
MI_RGN_Attr_t	The structure of region attribute
MI_RGN_Bitmap_t	The attribute of bitmap attribute
MI_RGN_ModId_e	The enumeration of module ID
MI_RGN_ChnPort_t	The structure of channel port
MI_RGN_Point_t	The structure of coordinates
MI_RGN_CoverChnPortParam_t	The structure of the covered channel port
MI_RGN_ChnPortParam_t	The structure of the attribute of channel port
MI_RGN_CanvasInfo_t	The structure of canvas information
MI_RGN_OsdInvertColorAttr_t	The structure of the attribute of inverted mode of OSD

3.1. MI_RGN_MAX_HANDLE

- Description
The definition for the maximum control code of region.
- Definition
`#define MI_RGN_MAX_HANDLE 1024`
- Members
N/A.
- Note
N/A.
- Relevant data types and interface
N/A.

3.2. MI_RGN_MAX_PALETTE_TABLE_NUM

- Description
The maximum number of palette table.
- Definition
`#define MI_RGN_MAX_PALETTE_TABLE_NUM 256`
- Members
N/A.
- Note
N/A.
- Relevant data types and interface
N/A.

3.3. MI_RGN_HANDLE

- Description
The definition for the control code of region
- Definition
`typedef MI_U32 MI_RGN_HANDLE;`
- Members

Member	Description
MI_RGN_HANDLE	The control code of region.

➤ Note

N/A.

➤ Relevant data types and interface

N/A.

3.4. MI_RGN_Type_e

➤ Description

The enumeration of region type

➤ Definition

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

➤ Members

Member	Description
E_MI_REG_OSD	Video overlaid region
E_MI_REG_COVER	Video covered region

➤ Note

N/A.

➤ Relevant data types and interface

N/A.

3.5. MI_RGN_PixelFormat_e

➤ Description

The enumeration of Pixel format.

➤ Definition

```
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;
```

➤ Members

Member	Description
E_MI_RGN_PIXEL_FORMAT_ARGB1555	ARGB1555 format
E_MI_RGN_PIXEL_FORMAT_ARGB4444	ARGB4444 format
E_MI_RGN_PIXEL_FORMAT_RGBI2	2 bit representation. Can support 4 colors. Check color by the palette table.
E_MI_RGN_PIXEL_FORMAT_RGBI4	4 bit representation. Can support 16 colors. Check color by the palette table.
E_MI_RGN_PIXEL_FORMAT_I8	8 bit representation. Can support 256 colors. Check color by the palette table.
E_MI_RGN_PIXEL_FORMAT_RGB565	RGB565
E_MI_RGN_PIXEL_FORMAT_ARGB8888	ARGB8888 format

➤ Note

The image format supported by each chip is different. The APIs list all image formats, but some formats may not be supported by the API. If users need chip support, please refer to region procs. Command: `echo getcap > /proc/mi_modules/mi_rgn/mi_rgn0`

Users cannot set the index 0 of the color palette. The index 0 of these formats is used as the color key by the underlying driver. That means this color is not recognized by hardware, so when all 0 data is covered on the channel, no color is displayed.

The value of the Color key can be viewed in the getcap of procs. The value of color key is a 16bit integer. Its high 8 bits and low 8 bits are the same value. When using the color format of the Index type, memset 0 on the memory data can make the hardware not recognized. When using the RGB or ARGB format, no matter what arrangement is used, the memset (color key & 0xFF) value can be used for the memory data.

➤ Relevant data types and interface

N/A.

3.6. MI_RGN_InvertColorMode_e

➤ Description

The enumeration of inverted mode.

➤ Definition

```
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;
```

➤ Members

Member	Description
E_MI_RGN_ABOVE_LUMA_THRESHOLD	If it is greater than the threshold, then invert color.
E_MI_RGN_BELOW_LUMA_THRESHOLD	If it is less than the threshold, then invert color.

➤ Note

N/A.

➤ Relevant data types and interface

N/A.

3.7. MI_RGN_AlphaMode_e

➤ Description

The way to display with alpha mode.

➤ Definition

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

➤ Members

Member	Description
E_MI_RGN_PIXEL_ALPHA	OSD display every pixel with alpha value, which can support by these color formats 'argb1555/argb4444/arb8888/i2/i4/i8'. Rgb565 without alpha value does not make sense.
E_MI_RGN_CONSTANT_ALPHA	The hardware ignores the alpha value (if exist) from the pixel format, In fact it use the specified alpha value to take the place of the original alpha value of each pixel.

➤ Note

N/A.

➤ Relevant data types and interface

N/A.

3.8. MI_RGN_Size_t

➤ Description

The structure of the region size

➤ Definition

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

➤ Members

Member	Description
u32Width	Width
u32Height	Height

➤ Note

N/A.

➤ Relevant data types and interface

N/A.

3.9. MI_RGN_OsdInvertColorAttr_t

➤ Description

The structure of the attribute of inverted mode of OSD.

➤ Definition

```
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;
```

➤ Members

Member	Description
bEnableColorInv	Enable color inverter
eInvertColorMode	Inverted color mode
u16LumaThreshold	Luma threshold. Range: 0~255
u16WDivNum	The horizontal cutting number of inverted region. The width of the inverted color region should be divisible by the horizontal cutting number.
u16HDivNum	The vertical cutting number of inverted region. The height of the inverted color region should be divisible by the vertical cutting number.

➤ Note

Inverted color can only be displayed in the region. It will only work if the inverted color is turned on and the setting parameters are legal.

The position and size of the inverted region is the same as the region. It is recommended to create a region according to the size of the area that needs to be reversed.

The number of cuttable blocks in the inverted color range is 1~2048. The maximum range of each cutting block is 64pixel x 64pixel. The product of u16WdivNum and u16HDivNum cannot exceed 2048. Can't be set too small, so that the extent of the cutting block exceeds the maximum limit. At the same time, it is necessary to satisfy the width and height divided by the horizontal cutting number and the vertical cutting number respectively, otherwise the inverted color has no effect. Bind multiple regions with the same Channel port. When multiple regions turn on inverted color, the inverted color mode and threshold of each region must be set to be consistent. If the inverted color mode and the threshold setting are inconsistent, the actual inverted color mode and its threshold are the same as the settings of the top layer displaying and turning on the inverted region.

➤ Relevant data types and interface

N/A.

3.10. MI_RGN_OsdAlphaAttr_t

➤ Description

The structure of the attribute of OSD 'alpha'.

➤ Definition

```
typedef struct MI_RGN_OsdAlphaAttr_s
{
    MI\_RGN\_AlphaMode\_e eAlphaMode;
    MI\_RGN\_AlphaModePara\_u stAlphaPara;
}MI_RGN_OsdAlphaAttr_t;
```

➤ Members

Member	Description
eAlphaMode	The OSD alpha mode enum.
stAlphaPara	The parameters of alpha mode.

➤ Note

N/A

➤ Relevant data types and interface

N/A.

3.11. MI_RGN_OsdInitParam_t

➤ Description

The structure of the attribute of overlaid region.

➤ Definition

```
typedef struct MI_RGN_OsdInitParam_s
{
    MI\_RGN\_PixelFormat\_e ePixelFmt;
    MI\_RGN\_Size\_t stSize;
}MI_RGN_OsdInitParam_t;
```

➤ Members

Member	Description
ePixelFmt	Pixel format
stSize	The width and height of region Range: Width: It is related to output port resolution(8pixel alignment).TBD Height: It is related to output port resolution(2pixel alignment).

- Note
 - ePixelFormat, stSize are meaningful only after calling [MI_RGN_AttachToChn](#).
 - ePixelFormat and stSize are static variables before calling [MI_RGN_DetachFromChn](#)
- Relevant data types and interface
N/A.

3.12. MI_RGN_PaletteElement_t

- Description
The structure of the palette element of region

- Definition


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

- Members

Member	Description
u8Alpha	Alpha
u8Red	Red
u8Green	Green
u8Blue	Blue

- Note
N/A.
- Relevant data types and interface
N/A.

3.13. MI_RGN_PaletteTable_t

- Description
The structure of Palette table.

➤ Definition

```
typedef struct MI_RGN_PaletteTable_s
{
    MI\_RGN\_PaletteElement\_t astElement[MI\_RGN\_MAX\_PALETTE\_TABLE\_NUM];
}MI_RGN_PaletteTable_t;
```

➤ Members

Member	Description
astElement	Color element

➤ Note

N/A.

➤ Relevant data types and interface

N/A.

3.14. MI_RGN_Attr_t

➤ Description

The structure of region attribute.

➤ Definition

```
typedef struct MI_RGN_Attr_s
{
    MI\_RGN\_Type\_e eType;
    MI\_RGN\_OsdInitParam\_t stOsdInitParam;
}MI_RGN_Attr_t;
```

➤ Members

Member	Description
eType	Region type
stOsdInitParam	Region attribute of OSD

➤ Note

N/A.

➤ Relevant data types and interface

N/A.

3.15. MI_RGN_Bitmap_t

➤ Description

The attribute of bitmap attribute.

➤ Definition

```
typedef struct MI_RGN_Bitmap_s
{
    MI\_RGN\_PixelFormat\_e ePixelFormat;
    MI\_RGN\_Size\_t stSize;
    void *pData;
} MI_RGN_Bitmap_t;
```

➤ Members

Member	Description
ePixelFormat	Pixel format
stSize	Bitmap width, height and stride
pData	The data of bitmap

➤ Note

N/A.

➤ Relevant data types and interface

N/A.

3.16. MI_RGN_ModId_e

➤ Description

The enumeration of module ID

➤ Definition

```
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;
```

➤ Members

Member	Description
E_MI_RGN_MODID_VPE	Module ID: VPE
E_MI_RGN_MODID_DIVP	Module ID: DIVP
E_MI_RGN_MODID_LDC	Module ID: LDC

➤ Note

N/A.

- Relevant data types and interface
N/A.

3.17. MI_RGN_ChnPort_t

- Description
The structure of channel port
- Definition

```
typedef struct MI_RGN_ChnPort_s
{
    MI\_RGN\_ModId\_e eModId;
    MI_S32 s32DevId;
    MI_S32 s32ChnId;
    MI_S32 s32OutputPortId;
}MI_RGN_ChnPort_t;
```

- Members

Member	Description
eModId	Module ID
s32DevId	Device ID
s32ChnId	Channel ID
s32OutputPortId	Output port ID

- Note
N/A.
- Relevant data types and interface
N/A.

3.18. MI_RGN_Point_t

- Description
The structure of coordinates.
- Definition

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

➤ Members

Member	Description
u32X	X-axis
u32Y	Y-axis

➤ Note

N/A.

➤ Relevant data types and interface

N/A.

3.19. MI_RGN_CoverChnPortParam_t

➤ Description

The structure of the covered channel port

➤ Definition

```
typedef struct MI_RGN_CoverChnPortParam_s
{
    MI_U32 u32Layer;
    MI\_RGN\_Size\_t stSize;
    MI_U32 u32Color;
}MI_RGN_CoverChnPortParam_t;
```

➤ Members

Member	Description
u32Layer	Layer. Low value at the bottom.
stSize	Width, height and stride
u32Color	Color , VYU444

➤ Note

N/A.

➤ Relevant data types and interface

N/A.

3.20. MI_RGN_OsdChnPortParam_t

➤ Description

The structure of the OSD channel port

➤ Definition

```
typedef struct MI_RGN_OsdChnPortParam_s
{
    MI_U32 u32Layer;
    MI\_RGN\_OsdAlphaAttr\_t stOsdAlphaAttr;
    MI\_RGN\_OsdInvertColorAttr\_t stColorInvertAttr;
}MI_RGN_OsdChnPortParam_t;
```

➤ Members

Member	Description
u32Layer	Layer.Low value at the bottom.

➤ Note

N/A.

➤ Relevant data types and interface

N/A.

3.21. [MI_RGN_OsdArgb1555Alpha_t](#)

➤ Description

Argb1555 only has one bit to describe the alpha, which divide two settings, the foreground alpha with alpha bit equal '1' and background alpha with alpha bit equal '0'.

➤ Definition

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

➤ Members

Member	Description
u8BgAlpha	The alpha value to display in background. Range is from '0' to '0xFF'.
u8FgAlpha	The alpha value to display in foreground. Range is from '0' to '0xFF'.

➤ Note

N/A.

➤ Relevant data types and interface

N/A.

3.22. MI_RGN_ChnPortParamUnion_u

➤ Description

The union of the channel port

➤ Definition

```
typedef union
{
    MI\_RGN\_CoverChnPortParam\_t stCoverChnPort;
    MI\_RGN\_OsdChnPortParam\_t stOsdChnPort;
} MI_RGN_ChnPortParamUnion_u;
```

➤ Members

Member	Description
stCoverChnPort	The attribute of covered channel port
stOsdChnPort	The attribute of OSD region on channel port.

➤ Note

N/A.

➤ Relevant data types and interface

N/A.

3.23. MI_RGN_AlphaModePara_u

➤ Description

The union of the alpha mode parameters.

➤ Definition

```
typedef union
{
    MI\_RGN\_OsdArgb1555Alpha\_t stArgb1555Alpha;
    MI_U8 u8ConstantAlpha;
} MI_RGN_AlphaModePara_u;
```

➤ Members

Member	Description
stArgb1555Alpha	The setting of argb1555 format foreground and background alpha.
u8ConstantAlpha	The setting in constant alpha case, and value range is from 0 to 0xFF.

- Note
N/A.
- Relevant data types and interface
N/A.

3.24. MI_RGN_ChnPortParam_t

- Description
The structure of the attribute of channel port.
- Definition

```
typedef struct MI_RGN_ChnPortParam_s
{
    MI_BOOL bShow;
    MI\_RGN\_Point\_t stPoint;
    MI\_RGN\_ChnPortParamUnion\_u unPara;
} MI_RGN_ChnPortParam_t;
```

- Members

Member	Description
bShow	Whether the area is displayed. Range: MI_TRUE or MI_FALSE. Dynamic attribute.
stPoint	Starting point coordinate
unPara	The display attribute of channel port.

- Note
N/A.
- Relevant data types and interface
N/A.

3.25. MI_RGN_CanvasInfo_t

- Description
The structure of canvas information

➤ Definition

```
typedef struct MI_RGN_CanvasInfo_s
{
    MI_PHY phyAddr;
    MI_VIRT virtAddr;
    MI\_RGN\_Size\_t stSize;
    MI_U32 u32Stride;
    MI\_RGN\_PixelFormat\_e ePixelFormat;
} MI_RGN_CanvasInfo_t;
```

➤ Members

Member	Description
phyAddr	The physical address of canvas
virtAddr	The virtual address of canvas
stSize	The size of canvas
u32Stride	The stride of canvas
ePixelFormat	The pixel format of canvas

➤ Note

N/A.

➤ Relevant data types and interface

N/A.

4. RETURN VALUE

The returned values of region module is shown in the following table.

Table 1: Return Values of Region Module

Marco definition	Description
MI_RGN_OK	Success.
MI_NOTICE_RGN_BUFFER_CHANGE	Buffer changed. This happens when the attribute is set and needs to be remapped again
MI_ERR_RGN_INVALID_HANDLE	Invalid handle.
MI_ERR_RGN_INVALID_DEVID	Invalid Device ID.
MI_ERR_RGN_INVALID_CHNID	Invalid channel ID or the control code
MI_ERR_RGN_ILLEGAL_PARAM	Illegal parameters.
MI_ERR_RGN_EXIST	Repeat creating the existing devices, channels, or resources.
MI_ERR_RGN_UNEXIST	Trying to use or destroy a device, channel, or resource that does not exist.
MI_ERR_RGN_NULL_PTR	Null pointer
MI_ERR_RGN_NOT_CONFIG	Not configured
MI_ERR_RGN_NOT_SUPPORT	Not supported
MI_ERR_RGN_NOT_PERM	This operation is not allowed, such as trying to modify static configuration parameters.
MI_ERR_RGN_NOMEM	Failed to allocate memory, such as insufficient system memory.
MI_ERR_RGN_NOBUF	The allocation for cache failed, such as the requested data buffer is too large.
MI_ERR_RGN_BUF_EMPTY	Buffer is empty.
MI_ERR_RGN_BUF_FULL	Buffer is full.
MI_ERR_RGN_BADADDR	Bad address
MI_ERR_RGN_BUSY	System is busy.
MI_ERR_RGN_NOTREADY	The system is not initialized or the corresponding module is not loaded.