DC100 DCMedia Development Guide

Release Version：V1.0.0

Release Date：20xx－xx－xx

Security Level： □Top-Secret □Secret □Internal ■Public

Overview

This document mainly describes the DCMedia media development reference.

Product Version

Chipset	Kernel Version
RV1126	Linux 4.19

Intended Audience

This document (this guide) is mainly intended for:

- Technical support engineers
- Software development engineers

Revision History

Version	Author	Date	Revision History
V1.0.0	LEE	20xx-xx-xx	Initial version

Contents

DC100 DCMedia Development Guide

1. Introduction

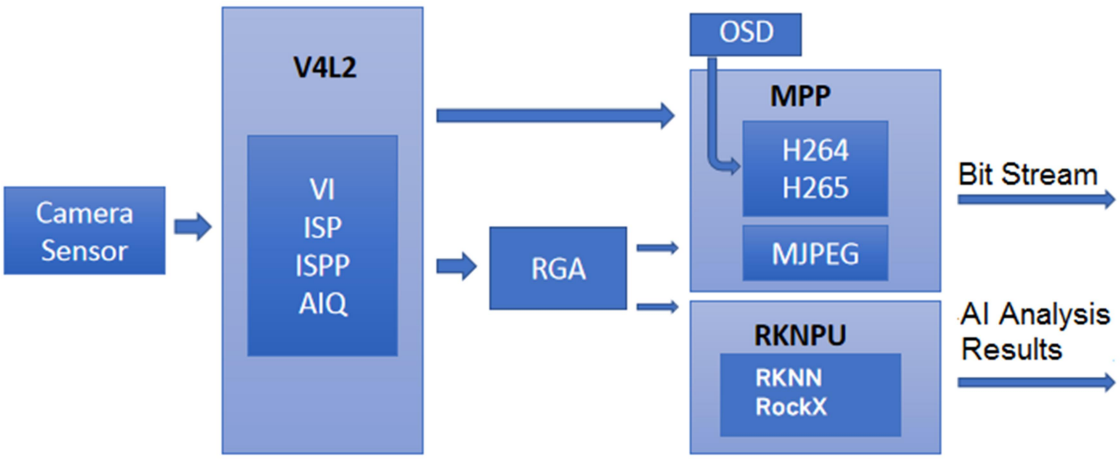
1.1 Overview

DCMedia provides a media processing solution that helps customers developing application software rapidly. DCMedia further package basic API of each module to simplify the difficulty of application development.

This platform supports the following functions:

- VI (input video capture)
- VENC (H.265/H.264/JPEG/MJPEG encoding)
- VDEC (H.265/H.264/JPEG, MJPEG decoding)
- RGA video processing (including rotation, scaling, cropping)
- MD (motion detection)
- OD (occlusion detection)

1.2 System Structure



1.3 System Resources Table

Module name	Number of channels
VI	8
VENC	16
VEDC	16
RGA	16
VMIX	16
MD	4
OD	4

2. System Control

2.1 Overview

The system controls the initialization work of the basic system, and is also responsible for initialization and de-initialization of each module, managing the binding relationship of each module, providing the current system version, and system log management.

2.2 Function Introduction

2.2.1 System Binding Expression

DCMedia provides a system binding interface (DC_MPI_SYS_Bind), in other words, data source is bound by data receiver to establish the relations between the two(data receiver is allowed to bind data source only). After binding, the data generated by the data source will be automatically sent to the receiver. Currently supported binding relationships are:

Data Source	Data Receiver
VI	RGA/VENC
VDEC	RGA/VENC
RGA	VENC

2.3 API Reference

2.3.1 DC_MPI_SYS_Init

【Description】

Initialize the system.

【Grammar】

DC_S32 DC_MPI_SYS_Init();

【Parameter】

No.

【Return value】

Return value	Description
0	Success.
Not 0	Failure, see Error Code for details.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.2 DC_MPI_SYS_DumpChn

【Description】

Print channel information.

【Grammar】

DC_VOID DC_MPI_SYS_DumpChn(MOD_ID_E enModId);

【Parameter】

Parameter name	Description	Input/Output
enModId	Module number	Input

【Return value】

No.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.3 DC_MPI_SYS_Bind

【Description】

The binding interface between data source and data receiver.

【Grammar】

```
DC_S32  DC_MPI_SYS_Bind(const MPP_CHN_S  *pstSrcChn, const MPP_CHN_S  *pstDestChn);
```

【Parameter】

Parameter name	Description	Input/Output
pstSrcChn	Source channel pointer	Input
pstDestChn	Target channel pointer	Input

【Return value】

Return value	Description
0	Success.
Not 0	Failure, see Error Code for details.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

For the binding relations currently supported by the system, please refer to 2.2.1 System Binding Expression. Before releasing the bound channel, you need to unbind it through DC_MPI_SYS_UnBind.

【Example】

No.

【Related topic】

DC_MPI_SYS_UnBind

2.3.4 DC_MPI_SYS_UnBind

【Description】

The Data source to data receiver unbinding interface.

【Grammar】

DC_S32 DC_MPI_SYS_UnBind(const MPP_CHN_S *pstSrcChn, const MPP_CHN_S *pstDestChn);

【Parameter】

Parameter name	Description	Input/Output
pstSrcChn	Source channel pointer	Input
pstDestChn	Target channel pointer	Input

【Return value】

Return value	Description
0	Success.
Not 0	Failure, see Error Code for details.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

DC_MPI_SYS_Bind

2.3.5 DC_MPI_SYS_RegisterEventCb

【Description】

Register an event callback, such as motion detection event.

【Grammar】

DC_S32 DC_MPI_SYS_RegisterEventCb(const MPP_CHN_S *pstChn, EventCbFunc cb);

【Parameter】

Parameter name	Description	Input/Output
pstChn	Specify the channel pointer	Input
cb	Event callback function	Output

【Return value】

Return value	Description
0	Success.
Not 0	Failure, see Error Code for details.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.6 DC_MPI_SYS_RegisterOutCb

【Description】

Register a data output callback. Compared with DC_MPI_SYS_GetMediaBuffer, there is no need to cache buffer waiting for users to request it, and save memories.

【Grammar】

DC_S32 DC_MPI_SYS_RegisterEventCb(const MPP_CHN_S *pstChn, OutCbFunc cb);

【Parameter】

Parameter name	Description	Input/Output
pstChn	Specify the channel pointer	Input
cb	Out callback function	Output

【Return value】

Return value	Description
0	Success.
Not 0	Failure, see Error Code for details.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

The callback function cannot handle time-consuming operations, otherwise the corresponding channel data stream will be blocked.

【Example】

cmos_rightcam_rtsp_test

【Related topic】

No.

2.3.7 DC_MPI_SYS_SendMediaBuffer

【Description】

Input data to the specified channel, such as sending the local yuv file to the encoder for encoding.

【Grammar】

DC_S32 DC_MPI_SYS_SendMediaBuffer(MODE_ID_E enModId, DC_S32 s32ChnId, MEDIA_BUFFER buffer);

【Parameter】

Parameter name	Description	Input/Output
enModId	Module number	Input

s32ChnId	Channel number	Input
buffer	Buffer	Input

【Return value】

Return value	Description
0	Success.
Not 0	Failure, see Error Code for details.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

DC_MPI_SYS_GetMediaBuffer

2.3.8 DC_MPI_SYS_DevSendMediaBuffer

【Description】

Input data to the specified channel of the specified device, such as sending the local yuv file to VMIX.

【Grammar】

```
DC_S32 DC_MPI_SYS_DevSendMediaBuffer(MODE_ID_E enModId, DC_S32 s32DevId,
                                     DC_S32 s32ChnId, MEDIA_BUFFER buffer);
```

【Parameter】

Parameter name	Description	Input/Output
enModId	Module number	Input
s32DevId	Device number	Input
s32ChnId	Channel number	Input
buffer	Buffer	Input

【Return value】

Return value	Description
0	Success.
Not 0	Failure, see Error Code for details.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

DC_MPI_SYS_GetMediaBuffer

2.3.9 DC_MPI_SYS_StartGetMediaBuffer

【Description】

Enable the receive buffer. After the receive buffer is enabled, even if the channel is bound, the MediaBuffer can be obtained through DC_MPI_SYS_GetMediaBuffer.

【Grammar】

DC_S32 DC_MPI_SYS_StartGetMediaBuffer(MOD_ID_E enModId, DC_S32 s32ChnId);

【Parameter】

Parameter name	Description	Input/Output
enModId	Module number	Input
s32ChnId	Channel number	Input

【Return value】

Return value	Description
0	Success.

Not 0	Failure, see Error Code for details.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

After enabling the receive buffer, call DC_MPI_SYS_GetMediaBuffer in time to remove MediaBuffer, otherwise a packet loss warning will be prompted.

【Example】

No.

【Related topic】

DC_MPI_SYS_GetMediaBuffer
DC_MPI_SYS_StopGetMediaBuffer

2.3.10 DC_MPI_SYS_StopGetMediaBuffer

【Description】

Close the receive buffer and clear the existing MediaBuffer in the buffer.

【Grammar】

DC_S32 DC_MPI_SYS_StopGetMediaBuffer(MODE_ID_E enModId, DC_S32 s32ChnId);

【Parameter】

Parameter name	Description	Input/Output
enModId	Module number	Input
s32ChnId	Channel number	Input

【Return value】

Return value	Description
0	Success.
Not 0	Failure, see Error Code for details.

【Requirement】

Header file：dcmadia_api.h

Library file：libdcmadia.so

【Notice】

After calling this interface, if you call RK_MPI_SYS_GetMediaBuffer again to get data, the receive buffer will be opened again.

【Example】

No.

【Related topic】

DC_MPI_SYS_GetMediaBuffer
DC_MPI_SYS_StartGetMediaBuffer

2.3.11 DC_MPI_SYS_GetMediaBuffer

【Description】

Obtain data from the specified channel.

【Grammar】

```
MEDIA_BUFFER DC_MPI_SYS_GetMediaBuffer(MODE_ID_E enModId, DC_S32 s32ChnId,
                                       DC_S32 s32MilliSec);
```

【Parameter】

Parameter name	Description	Input/Output
enModId	Module number	Input
s32ChnId	Channel number	Input
s32MilliSec	-1：block；>=0：blocking waiting time.	Input

【Return value】

Return value type	Description
MEDIA_BUFFER	Buffer pointer.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

This interface will trigger DC_MPI_SYS_StartGetMediaBuffer automatically.

【Example】

No.

【Related topic】

DC_MPI_SYS_StartGetMediaBuffer
DC_MPI_SYS_StopGetMediaBuffer

2.3.12 DC_MPI_SYS_SetFrameRate

【Description】

Set input frame rate of a channel.

【Grammar】

```
DC_S32 DC_MPI_SYS_SetFrameRate(MODE_ID_E enModId, DC_S32 s32ChnId,
                               MPP_FPS_ATTR_S *pstFpsAttr);
```

【Parameter】

Parameter name	Description	Input/Output
enModId	Module number	Input
s32ChnId	Channel number	Input
pstFpsAttr	Frame rate attribute	Input

【Return value】

Return value	Description
0	Success.
Not 0	Failure, see Error Code for details.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.13 DC_MPI_SYS_CreateBuffer

【Description】

Create a common buffer.

【Grammar】

MEDIA_BUFFER DC_MPI_SYS_CreateBuffer(DC_S32 u32Size, DC_BOOL boolHardWare,
DC_U8 u8Flag);

【Parameter】

Parameter name	Description	Input/Output
enModId	Module number	Input
s32ChnId	Channel number	Input
u8Flag	Additional flag for hardware type buffer, values: 0: create a hardware buffer with a buffer type MB_FLAG_NOCACHED: create a hardware buffer without a buffer type	Input

【Return value】

Return value type	Description
MEDIA_BUFFER	Buffer pointer.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

There is no image information structure in regular buffers.

【Example】

No.

【Related topic】

No.

2.3.14 DC_MPI_SYS_CreateImageBuffer

【Description】

Create an image buffer. Compared with regular buffers, it carries image information structure. During image handling, it is recommended to use this method to get buffer.

【Grammar】

```
MEDIA_BUFFER DC_MPI_SYS_CreateImageBuffer(MB_IMAGE_INFO_S *pstImageInfo,
                                          DC_BOOL boolHardWare,  DC_U8 u8Flag);
```

【Parameter】

Parameter name	Description	Input/Output
enModId	Module number	Input
s32ChnId	Channel number	Input
u8Flag	Additional flag for hardware type buffer, values: 0: create a hardware buffer with a buffer type MB_FLAG_NOCACHED: create a hardware buffer without a buffer type	Input

【Return value】

Return value type	Description
MEDIA_BUFFER	Buffer pointer.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.15 DC_MPI_MB_Copy

【Description】

MediaBuffer “zero copy” interface.

【Grammar】

MEDIA_BUFFER DC_MPI_MB_Copy(MEDIA_BUFFER mb, DC_BOOL bZeroCopy);

【Parameter】

Parameter name	Description	Input/Output
mb	Normal buffer pointer	Input
s32ChnId	“Zero Copy” is enabled	Input

【Return value】

Return value type	Description
MEDIA_BUFFER	Buffer pointer.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

Currently, only the "Zero Copy" (bZeroCopy=RK_TRUE) flag is supported, and the "Deep Copy" (bZeroCopy=RK_FALSE) flag is not yet supported.

【Example】

No.

【Related topic】

No.

2.3.16 DC_MPI_MB_ReleaseBuffer

【Description】

Release a buffer.

【Grammar】

void *DC_MPI_MB_ReleaseBuffer(MEDIA_BUFFER mb);

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer pointer	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.17 DC_MPI_MB_GetPtr

【Description】

Get the buffer data pointer from the specified MEDIA_BUFFER.

【Grammar】

```
void *DC_MPI_MB_Ptr(MEDIA_BUFFER mb);
```

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer pointer	Input

【Return value】

Return value type	Description
void *	Buffer data pointer.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.18 DC_MPI_MB_GetFD

【Description】

Get the file descriptor from the specified MEDIA_BUFFER.

【Grammar】

```
int DC_MPI_MB_GetFD(MEDIA_BUFFER mb);
```

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer pointer	Input

【Return value】

Return value type	Description
int	File descriptor

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.19 DC_MPI_MB_GetSize

【Description】

Get the buffer data size from the specified MEDIA_BUFFER.

【Grammar】

```
size_t DC_MPI_MB_GetSize(MEDIA_BUFFER mb);
```

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer	Input

【Return value】

Return value type	Description
size_t	Buffer data size

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.20 DC_MPI_MB_SetSize

【Description】

Set the date size of specified MEDIA_BUFFER.

【Grammar】

DC_S32 DC_MPI_MB_SetSize(MEDIA_BUFFER mb, size_t size);

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer pointer	Input
size	Buffer data size	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

The buffer data size has to been set by this function after operating the buffer and changing its size, otherwise DC_MPI_MB_GetSize will not be able to get the correct buffer data size.

【Example】

No.

【Related topic】

No.

2.3.21 DC_MPI_MB_GetModeID

【Description】

Get the module ID from the specified MEDIA_BUFFER.

【Grammar】

MOD_ID_E DC_MPI_MB_GetModeID(MEDIA_BUFFER mb);

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer pointer	Input

【Return value】

Return value type	Description
MOD_ID_E	Module ID

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.22 DC_MPI_MB_GetChannelID

【Description】

Get the channel ID from the specified MEDIA_BUFFER.

【Grammar】

DC_S16 DC_MPI_MB_GetChannelID(MEDIA_BUFFER mb);

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer	Input

【Return value】

Return value type	Description
DC_S16	Channel ID

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.23 DC_MPI_MB_GetTimestamp

【Description】

Get the timestamp from the specified MEDIA_BUFFER.

【Grammar】

DC_U64 DC_MPI_MB_GetTimestamp(MEDIA_BUFFER mb);

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer	Input

【Return value】

Return value type	Description
DC_U64	Timestamp

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.24 DC_MPI_MB_SetTimestamp

【Description】

Set timestamp of the specified MEDIA_BUFFER.

【Grammar】

DC_S32 DC_MPI_MB_SetTimestamp(MEDIA_BUFFER mb, DC_U64 timestamp);

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer	Input
timestamp	Timestamp	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

When calling the librga library (relative to RGA Channel) to handle MEDIA_BUFFER directly, users need to process the timestamp attribute manually. such as:

```
// Call librga api to deal with MediaBuffer0, the output timestamp attribute of MediaBuffer1 will be lost
// MediaBuffer0 --> RGA Crop --> MediaBuffer1
// Need to call this interface to copy the timestamp of MediaBuffer0 manually
DC_MPI_MB_SetTimestamp(MediaBuffer1, DC_MPI_MB_GetTimestamp(MediaBuffer0));
```

【Example】

No.

【Related topic】

No.

2.3.25 DC_MPI_MB_GetFlag

【Description】

Get Flag from the specified MEDIA_BUFFER. Flag is used to mark the special attributes of Buffer, such as frame type: I frame, P frame, etc

【Grammar】

DC_S32 DC_MPI_MB_GetFlag(MEDIA_BUFFER mb);

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer	Input

【Return value】

Return value type	Description
DC_S32	Additional flag for hardware type Buffer: 0: create hardware buffer with buffer type MB_FLAG_NOCACHED: create hardware buffer without buffer type

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.26 DC_MPI_MB_GetTsvcLevvel

【Description】

Get TSVC level from the specified MEDIA_BUFFER. TSVC represents SVC in time dimension. After TSVC function is enabled, the code stream will be divided into three levels: L0, L1, and L2. High-level coded frames are decoded depending on low-level coded frames (for example, L2 depends on L1, L0; L1 depends on L0), but low-level coded frames can be decoded independently (L0 can be decoded independently, L1, L0 can be decoded independently of L2). For example, for a 60fps video, if you decode L1 and L0 only, you will get a 30fps video, if you decode L0 only, you will get a 15fps video.

【Grammar】

DC_S32 DC_MPI_MB_GetTsvcLevel(MEDIA_BUFFER mb);

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer	Input

【Return value】

Return value type	Description
DC_S32	TSVC level

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

If the encoder does not turn on the TSVC/SMARTP mode, the return value of this interface is invalid.

【Example】

No.

2.3.27 DC_MPI_MB_IsViFrame

【Description】

Determine whether the specified MEDIA_BUFFER is a VirtualIntra frame (virtual I frame).

【Grammar】

DC_BOOL DC_MPI_MB_IsViFrame(MEDIA_BUFFER mb);

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer	Input

【Return value】

Return value type	Description
DC_BOOL	VI frame인지 여부 확인.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

Only valid in smartp mode.

【Example】

No.

【Related topic】

No.

2.3.28 DC_MPI_MB_GetImageInfo

【Description】

Get image information from the specified image buffer MEDIA_BUFFER.

【Grammar】

DC_S32 DC_MPI_MB_GetImageInfo(MEDIA_BUFFER mb, MB_IMAGE_INFO_S *pstImageInfo);

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer	Input
bReadonly	Image information structure pointer in Buffer	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

This function can be used by the image buffer to get information only.

【Example】

No.

【Related topic】

No.

2.3.29 DC_MPI_MB_BeginCPUAccess

【Description】

Solve the synchronization problem caused by the operation of the same MEDIA_BUFFER between CPU and hardware module (such as ENCODER).

【Grammar】

DC_S32 DC_MPI_MB_BeginCPUAccess(MEDIA_BUFFER mb, DC_BOOL bReadonly);

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer	Input
pstImageInfo	Whether it is read-only.	Output

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

Need to be used together with DC_MPI_MB_EndCPUAccess.

【Example】

For example, in the case that after CPU filling buffer and then sending date for VENC encoding, the MediaBuffer created by the RK_MPI_MB_CreateImageBuffer interface is with cache by default (you can create a NoCache type MediaBuffer through flag=MB_FLAG_NOCACHED), so after CPU writing buffer and it will remain some data in the cache and fail to be synchronized to memory (DDR) in time. At this time, it will be sent for VENC coding immediately, which will cause the image to be abnormal (such as intermittent green short lines). This interface is used to ensure that after CPU operates buffer, the buffer is flushed to memory immediately.

```
MEDIA_BUFFER mb;
RK_MPI_MB_BeginCPUAccess(mb, RK_FALSE);
// CPU fill data to mb.
memset(RK_MPI_MB_GetPtr(mb),'F', size);
RK_MPI_MB_EndCPUAccess(mb, RK_FALSE);
// Send mb to VENC
RK_MPI_SYS_SendMediaBuffer(RK_ID_VENC, 0, mb);
```

2.3.30 DC_MPI_MB_EndCPUAccess

【Description】

Solve the synchronization problem caused by the operation of the same MEDIA_BUFFER between CPU and hardware module (such as ENCODER).

【Grammar】

DC_S32 DC_MPI_MB_EndCPUAccess(MEDIA_BUFFER mb, DC_BOOL bReadonly);

【Parameter】

Parameter name	Description	Input/Output
mb	Buffer	Input
pstImageInfo	읽기 전용인지 확인.	Output

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

Need to be used together with RK_MPI_MB_BeginCPUAccess.

【Example】

No.

【Related topic】

No.

2.3.31 DC_MPI_MB_POOL_Create

【Description】

Create BufferPool.

【Grammar】

MEDIA_BUFFER_POOL DC_MPI_MB_POOL_Create(MB_POOL_PARAM_S *pstPoolParam);

【Parameter】

Parameter name	Description	Input/Output
pstPoolParam	BufferPool attribute	Input

【Return value】

Return value	Description
NULL	Failure
Not NULL	Success

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.32 DC_MPI_MB_POOL_Destroy

【Description】

Destroy BufferPool.

【Grammar】

DC_S32 DC_MPI_MB_POOL_Destroy(MEDIA_BUFFER_POOL MBPHandle);

【Parameter】

Parameter name	Description	Input/Output
MBPHandle	MediaBufferPool object to be destroyed	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for its value.

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.33 DC_MPI_MB_POOL_GetBuffer

【Description】

Get MediaBuffer from BufferPool.

【Grammar】

```
MEDIA_BUFFER DC_MPI_MB_POOL_GetBuffer(MEDIA_BUFFER_POOL MBPHandle,
                                      DC_BOOL bIsBlock);
```

【Parameter】

Parameter name	Description	Input/Output
MBPHandle	MediaBufferPool object to be destroyed	Input
bIsBlock	Whether to block	Input

【Return value】

Return value	Description
Not NULL	Success
NULL	Failure

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.3.34 DC_MPI_LOG_SetLevelConf

【Description】

Set log level.

【Grammar】

DC_S32 DC_MPI_LOG_SetLevelConf(LOG_LEVEL_CONF_S *pstConf);

【Parameter】

Parameter name	Description	Input/Output
pstConf	Log level information structure	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

When the member cModName in pstConf is set to the string "all", log level of all modules will be set. Otherwise, set log level of the module specified by enModId only.

【Example】

No.

【Related topic】

No.

2.3.35 DC_MPI_LOG_GetLevelConf

【Description】

Get log level.

【Grammar】

DC_S32 DC_MPI_LOG_GetLevelConf(LOG_LEVEL_CONF_S *pstConf);

【Parameter】

Parameter name	Description	Input/Output
pstConf->enModID	The module ID whose log level needs to be obtained.	Input
pstConf->s32Level	Get log level.	Output
pstConf->cModName	The name of the module.	Output

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

2.4 Data Type

2.4.1 Basic Data Types

2.4.1.1 Regular Data Types

【Description】

Definition of basic data types.

【Definiton】

```
typedef unsigned char DC_U8;
typedef unsigned short DC_U16;
typedef unsigned int DC_U32;
typedef signed char DC_S8;
typedef short DC_S16;
typedef int DC _S32;
typedef unsigned long DC_UL;
typedef signed long DC_SL;
typedef float DC_FLOAT;
typedef double DC_DOUBLE;
typedef unsigned long long DC_U64;
typedef long long DC_S64;
typedef char DC_CHAR;
typedef unsigned int DC_HANDLE;
#define RK_VOID void

typedef enum {
    DC_FALSE = 0,
    DC_TRUE = 1,
} DC_BOOL;
```

2.4.1.2 IMAGE_TYPE_E

【Description】

Define an image format enumeration type.

【Definiton】

```
typedef enum rk_IMAGE_TYPE_E {
    IMAGE_TYPE_UNKNOW = 0,
    IMAGE_TYPE_GRAY8,
```

```
    IMAGE_TYPE_GRAY16,
    IMAGE_TYPE_YUV420P,
    IMAGE_TYPE_NV12,
    IMAGE_TYPE_NV21,
    IMAGE_TYPE_YV12,
    IMAGE_TYPE_FBC2,
    IMAGE_TYPE_FBC0,
    IMAGE_TYPE_YUV422P,
    IMAGE_TYPE_NV16,
    IMAGE_TYPE_NV61,
    IMAGE_TYPE_YV16,
    IMAGE_TYPE_YUYV422,
    IMAGE_TYPE_UYVY422,
    IMAGE_TYPE_RGB332,
    IMAGE_TYPE_RGB565,
    IMAGE_TYPE_BGR565,
    IMAGE_TYPE_RGB888,
    IMAGE_TYPE_BGR888,
    IMAGE_TYPE_ARGB8888,
    IMAGE_TYPE_ABGR8888,
    IMAGE_TYPE_JPEG,
    IMAGE_TYPE_BUTT
} IMAGE_TYPE_E;
```

2.4.1.3 CODEC_TYPE_E

【Description】

Define codec format enumeration type.

【Definiton】

```
typedef enum rk_CODEC_TYPE_E {
    DC_CODEC_TYPE_NONE = -1,
    DC_CODEC_TYPE_H264 = 6,
    DC_CODEC_TYPE_H265,
    DC_CODEC_TYPE_JPEG,
    DC_CODEC_TYPE_MJPEG,
    DC_CODEC_TYPE_NB
} CODEC_TYPE_E;
```

2.4.1.4 MOD_ID_E

【Description】

Define a module ID enumeration type.

【Definiton】

```
typedef enum rkMOD_ID_E {
    DC_ID_UNKNOW = 0,
    DC_ID_VB,
    DC_ID_SYS,
    DC_ID_VDEC,
    DC_ID_VENC,
    DC_ID_H264E,
    DC_ID_JPEGE,
    DC_ID_H265E,
    DC_ID_VO,
    DC_ID_VI,
    DC_ID_AIO,
    DC_ID_AI,
    DC_ID_AO,
    DC_ID_AENC,
    DC_ID_ADEC,
    DC_ID_ALGO_MD,
    DC_ID_ALGO_OD,
    DC_ID_RGA,
    DC_ID_BUTT,
} MOD_ID_E;
```

2.4.1.5 RECT_S

【Description】

Define the region attribute structure.

【Definiton】

```
typedef struct rkRECT_S {
    DC_S32 s32X;
    DC_S32 s32Y;
    DC_U32 u32Width;
```

```
    DC_U32 u32Height;
} RECT_S;
```

【Members】

Member name	Description
s32X	X coordinate of the region
s32Y	Y coordinate of the region
u32Width	The width of the region
u32Height	The height of the region

2.4.2 System control Data Type

The data types about system control are defined as follows:

- MPP_CHN_S: define the module device channel structure.
- EventCbFunc: event callback function pointer.
- MEDIA_BUFFER: data buffer pointer.
- OutCbFunc: data output callback function pointer.
- MB_IMAGE_INFO_S: image information structure.

2.4.2.1 MPP_CHN_S

【Description】

Define the module device channel structure.

【Definiton】

```
typedef struct rkMPP_CHN_S {
        MOD_ID_E enModId;
        DC_S32 s32DevId;
        DC_S32 s32ChnId;
} MPP_CHN_S;
```

【Members】

Member name	Description
enMode	Module number

s32DevId	Device ID
s32ChnId	Channel number

2.4.2.2 EventCbFunc

【Description】

Event callback function pointer.

【Definiton】

```
typedef enum rkEVENT_TYPE_E {
    DC_EVENT_ERR = 0,
    DC_EVENT_MD,      // Algo::Move detection event.
    DC_EVENT_OD,      // Algo::Occlusion detection event.
    DC_EVNET_BUT
} EVENT_TYPE_E;

typedef struct rkMD_EVENT_S {
    DC_U16 u16Cnt;
    DC_U32 u32Width;
    DC_U32 u32Height;
    RECT_S stRects[4096];
} MD_EVENT_S;

typedef struct rkOD_EVENT_S {
    DC_U16 u16Cnt;
    DC_U32 u32Width;
    DC_U32 u32Height;
    RECT_S stRects[10];
    DC_U16 u16Occlusion[10];
} OD_EVENT_S;

typedef struct rkEVENT_S {
    EVENT_TYPE_E type;
    MOD_ID_E mode_id;
    union {
        MD_EVENT_S md_event;
        OD_EVENT_S stOdEvent;
    };
} EVENT_S;

typedef void (*EventCbFunc)(EVENT_S *event);
```

【Members】

Member name	Description
type	Event type
mode_id	Module number
md_event	Motion detection event
stOdEvent	Occlusion detection event

2.4.2.3 MEDIA_BUFFER

【Description】

Data buffer pointer.

【Definiton】

```
typedef void *MEDIA_BUFFER;
```

2.4.2.4 OutCbFunc

【Description】

Data output callback function pointer.

【Definiton】

```
typedef void (*OutCbFunc)(MEDIA_BUFFER mb);
```

2.4.2.5 MB_IMAGE_INFO_S

【Description】

Image information structure.

【Definiton】

```
typedef struct rkMB_IMAGE_INFO {
    DC_U32 u32Width;
    DC_U32 u32Height;
```

```
    DC_U32 u32VerStride;
    DC_U32 u32HorStride;
    IMAGE_TYPE_E enImgType;
} MB_IMAGE_INFO_S;
```

【Members】

Member name	Description
u32width	Width
u32Height	Height
u32HorStride	Virtual width
u32VerStride	Virtual height
enImgType	Image format type

2.4.2.6 LOG_LEVEL_CONF_S

【Description】

Define the log level information structure.

【Definiton】

```
typedef struct rkLOG_LEVEL_CONF_S {
    MOD_ID_E enModId;
    DC_S32 s32Level;
    DC_CHAR cModName[16];
} LOG_LEVEL_CONF_S;
```

【Members】

Member name	Description
enModId	Module ID
s32Level	Log level
cModName	The name of the module

2.4.2.7 MPP_FPS_ATTR_S

【Description】

Channel input frame rate attributes.

【Definiton】

```
typedef struct rkMPP_FPS_ATTR_S {
    DC_S32 s32FpsInNum;
    DC_S32 s32FpsInDen;
    DC_S32 s32FpsOutNum;
    DC_S32 s32FpsOutDen;
} MPP_FPS_ATTR_S;
```

【Members】

Member name	Description
s32FpsInNum	Input the frame rate numerator
s32FpsInDen	Input the frame rate denominator
s32FpsOutNum	Output frame rate numerator
s32FpoOutDen	Output frame rate denominator

2.4.2.8 MB_POOL_PARAM_S

【Description】

Media BufferPool attribute structure.

【Definiton】

```
typedef enum rkMB_TYPE {
    MB_TYPE_COMMON = 0, // Original image, such as NV12, RGB
    MB_TYPE_IMAGE = MB_TYPE_IMAGE_MASK | 0x0000,
    // Encoded video data. Treat JPEG as a video data.
    MB_TYPE_VIDEO = MB_TYPE_VIDEO_MASK | 0x0000,
    MB_TYPE_H264 = MB_TYPE_VIDEO_MASK | 0x0001,
    MB_TYPE_H265 = MB_TYPE_VIDEO_MASK | 0x0002,
    MB_TYPE_JPEG = MB_TYPE_VIDEO_MASK | 0x0003,
    MB_TYPE_MJPEG = MB_TYPE_VIDEO_MASK | 0x0004,
} MB_TYPE_E;

typedef struct rkMB_POOL_PARAM_S {
    MB_TYPE_E enMediaType;
    DC_U32 u32Cnt;
    DC_U32 u32Size;
```

```
    DC_BOOL bHardWare;
    DC_U16 u16Flag;
    union {
        MB_IMAGE_INFO_S stImageInfo;
    };
} MB_POOL_PARAM_S;
```

【Members】

Member name	Description
enMediaType	Media type
u32Cnt	Number of Buffers in BufferPool.
u32Size	The memory size of each buffer.
bHardWare	Whether to allocate the type of hardware buffer.
u16Flag	Memory allocation flag, used to select the type of hardware buffer.
stImageInfo	Attribute information of image buffer, please refer to MB_IMAGE_INFO_S

2.5 Error Code

System control error code is as follows:

Error code	Macro definition	Description
1	DC_ERR_SYS_NULL_PTR	Null pointer error.
2	DC_ERR_SYS_NOTREADY	System control attributes are not configured.
3	DC_ERR_SYS_NOT_PERM	Operation not allowed.
4	DC_ERR_SYS_NOMEM	Failed to allocate memory, such as insufficient system memory.
5	DC_ERR_SYS_ILLEGAL_PARAM	Invalid parameter setting.
6	DC_ERR_SYS_BUSY	System is busy.
7	DC_ERR_SYS_NOT_SUPPORT	Unsupported function.

3. Video Input

3.1 Overview

Video Input (Vi for short) is used to read camera data. This module is an encapsulation of standard V4L2 interface and depends on Linux V4L2 driver architecture. The ISP/ISPP/VICAP driver provides file nodes (such as /dev/video0) to user layer through V4L2 architecture, and VI implements operations such as reading parameter configuration video frame by operating the file nodes.

3.2 Function Description

3.2.1 VI Channel Initialization

For DC100 platform, the dcaiq interface needs to be called to initialize hardware path. Please refer to the interface in app/dc100_test/common/sample_common.h for details.

Introduction to using dcaiq and dcmedia VI interface together:

1. Restrictions:

If all VI Channels are closed, you need to re-initialize the channel. In the case of multi-VI Channel, if only part of VI Channel is closed, there is no need to reinitialize the channel. Reinitialization is required when all VI Channels are closed, In the single VI Channel case, to close the VI Channel, you need to call ISP Stop logic to close the ISP channel; to open the VI, you need to initialize the VI channel again.

2. The interface calling sequence for channel initialization and de-initialization is as follows

Initialization:

① ISP Init // corresponding to dc_aiq_uapi_sysctl_init
② ISP Run // corresponding to dc_aiq_uapi_sysctl_prepare & dc_aiq_uapi_sysctl_start
③ VI Enable (single/multi)

Deinitialization:

① VI disable (single/multi)
② ISP Stop // corresponding to rk_aiq_uapi_sysctl_stop & rk_aiq_uapi_sysctl_deinit

3.2.2 VI Video Node

VI creation needs to specify the video node (VideoNode), such as "/dev/video0". Each video node corresponds to a video stream. A single camera can provide multiple resolution video streams. For example, ISPP of DC100 platform can provide 4 resolution video streams at the same time, because ISPP driver provides 4 video nodes to user layer.

For platforms with RKISP, each camera connected to ISPP will provide users with 4 video nodes, as shown in the following table. The name starting with "rkispp_" is an alias mechanism provided by driver, which will be translated into the corresponding /dev/videoX node inside VI. Users only need to use these 4 fixed names to obtain video streams of different resolutions.

ISPP node name	Maximum width	Scale	Description
rkispp_m_bypass	maximum width of sensors	Not Supported	NV12/NV16/YUYV/FBC0/FBC2
rkispp_scale0	3264	1~8 times	NV12/NV16/YUYV
rkispp_scale1	1280	2~8 times	NV12/NV16/YUYV
rkispp_scale2	1280	2~8 times	NV12/NV16/YUYV

Note: rkispp_m_bypass does not support scaling, and the resolution can only maintain the maximum resolution of sensor. After the resolution of rkispp_scale0 exceeds 2K, NV16 format is needed.

3.2.3 VI Working Mode

There are two VI working modes, as shown in the following table:

Mode name	Macro definition name	Function description
Normal mode	VI_WORK_MODE_NORMAL	Compared with "Brightness mode", camera data is read normally in this mode and sent to the subsequent stage.
Brightness mode	VI_WORK_MODE_LUMA_ONLY	In brightness mode, VI is used for brightness statistics only. At this time, VI module cannot get data through callback function or RK_MPI_SYS_GetMediaBuffer.

3.3 API Reference

3.3.1 DC_MPI_VI_EnableChn

【Description】

Enable the VI channel.

【Grammar】

DC_S32 DC_MPI_VI_EnableChn(VI_PIPE ViPipe, VI_CHN ViChn);

【Parameter】

Parameter name	Description	Input/Output
ViPipe	VI pipe number.	Input
ViChn	VI channel number. (0 ~ 8)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

3.3.2 DC_MPI_VI_DisableChn

【Description】

Close the VI channel.

【Grammar】

DC_S32 DC_MPI_VI_DisableChn(VI_PIPE ViPipe, VI_CHN ViChn);

【Parameter】

Parameter name	Description	Input/Output
ViPipe	VI pipe number.	Input
ViChn	VI channel number. (0 ~ 8)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

3.3.3 DC_MPI_VI_SetChnAttr

【Description】

Set VI channel properties.

【Grammar】

DC_S32 DC_MPI_VI_SetChnAttr(VI_PIPE ViPipe, VI_CHN ViChn, VI_CHN_ATTR_S *pstChnAttr);

【Parameter】

Parameter name	Description	Input/Output
ViPipe	VI pipe number.	Input
ViChn	VI channel number. (0 ~ 8)	Input
pstChnAttr	VI channel attribute structure pointer.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

If the channel to be set has been bound to other channels through DC_MPI_SYS_Bind, you need to unbind it through DC_MPI_SYS_UnBind before using this function setting. If the channel to be set has been enabled with DC_MPI_VI_EnableChn, you need to disable the channel through DC_MPI_VI_DisableChn before using this function setting.

【Example】

No.

【Related topic】

No.

3.3.4 DC_MPI_VI_StartRegionLuma

【Description】

Turn on VI brightness statistics.

【Grammar】

DC_S32 DC_MPI_VI_StartRegionLuma (VI_CHN ViChn);

【Parameter】

Parameter name	Description	Input/Output
ViChn	VI channel number. (0 ~ 8)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

It will take effect when VI_WORK_MODE is set to VI_WORK_MODE_LUMA_ONLY. In this mode, the number of buffers (u32BufCnt) needs to be greater than or equal to 3.

【Example】

No.

【Related topic】

No.

3.3.5 DC_MPI_VI_StopRegionLuma

【Description】

Stop VI brightness counting.

【Grammar】

DC_S32 DC_MPI_VI_StopRegionLuma (VI_CHN ViChn);

【Parameter】

Parameter name	Description	Input/Output
ViChn	VI channel number. (0 ~ 8)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

It will take effect when VI_WORK_MODE is set to VI_WORK_MODE_LUMA_ONLY only.

【Example】

No.

【Related topic】

No.

3.3.6 DC_MPI_VI_GetChnRegionLuma

【Description】

Get the region brightness information. It can be used to reverse color of VENC OSD.

【Grammar】

```
DC_S32 DC_MPI_VI_GetChnRegionLuma (VI_PIPE ViPipe , VI_CHN ViChn,
                                   const VIDEO_REGION_INFO_S *pstRegionInfo, DC_U64 *pu64LumaData,
                                   DC_S32 s32MilliSec);
```

【Parameter】

Parameter name	Description	Input/Output
ViPipe	VI pipe number.	Input
ViChn	VI channel number. (0 ~ 8)	Input
pstRegionInfo	Region information. pstRegionInfo->pstRegion is region attribute of the statistical region, that is, the starting position, width, and height; pstRegionInfo->u32RegionNum is the number of the statistical region.	Input
pu64LumaData	Memory pointer for receiving region brightness and statistical information. The memory size should be greater than or equal to sizeof(RK_U64)×pstRegionInfo->u32RegionNum.	Output
s32MilliSec	Timeout parameter s32MilliSec: -1 means blocking mode; 0 means non-blocking mode; greater than 0 means timeout mode, and the unit of timeout time is milliseconds (ms).	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

This interface does not support FBC0/FBC2 compression format.

【Example】

No.

【Related topic】

No.

3.3.7 DC_MPI_VI_StartStream

【Description】

Start video stream.

【Grammar】

DC_S32 DC_MPI_VI_StartStream(VI_PIPE ViPipe , VI_CHN ViChn);

【Parameter】

Parameter name	Description	Input/Output
ViPipe	Vi pipe number	Input
ViChn	Vi channel number (0 ~ 8)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

3.3.8 DC_MPI_VI_SetUserPic

【Description】

Insert user pictures.

【Grammar】

DC_S32 DC_MPI_VI_SetUserPic(VI_CHN ViChn, VI_USERPIC_ATTR_S *pstUserPicAttr);

【Parameter】

Parameter name	Description	Input/Output
ViChn	Vi channel number (0 ~ 8)	Input
pstUserPicAttr	User pictures atrribute	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

After inserting, you need to actively call DC_MPI_VI_EnableUserPic. At this time, VI will output user inserted.

【Example】

No.

【Related topic】

No.

3.3.9 DC_MPI_VI_EnableUserPic

【Description】

Enable users to insert pictures.

【Grammar】

DC_S32 DC_MPI_VI_EnableUserPic(VI_CHN ViChn);

【Parameter】

Parameter name	Description	Input/Output
ViChn	Vi channel number	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

3.3.10 DC_MPI_VI_DisableUserPic

【Description】

Disable users for inserting pictures.

【Grammar】

DC_S32 DC_MPI_VI_DisableUserPic(VI_CHN ViChn);

【Parameter】

Parameter name	Description	Input/Output
ViChn	Vi channel number	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

3.4 Type of data

The video input data types are defined as follows:

- VI_MAX_CHN_NUM：define the total of VI physical channels.
- VI_PIPE：VI pipe numbet.
- VI_CHN：VI chammel number.
- VI_CHN_ATTR_S：VI channel attribute structure pointer.
- VIDEO_REGION_INFO_S：define video region information.

3.4.1 VI_MAX_CHN_NUM

【Description】

Define the total number of VI physical channels. The typical case of RV1126/RV1109 platform is to connect 1~2 sensor modules, and each sensor can provide a maximum of 4 video channels (corresponding to the 4 video nodes of ISPP), so the maximum number is 8.

【Definiton】

```
#defime VI_MAX_CHN_NUM 8
```

3.4.2 VI_PIPE

【Description】

VI pipe number, corresponding to the number of sensors. PIPE is named because the back end of sensor is connected to a series of processing units such as ISP/ISPP to form a data PIPE.

【Definiton】

```
typedef DC_S32 VI_PIPE;
```

3.4.3 VI_CHN

【Description】

VI channel number. It determine the data of a channel of a certain camera together with VI_PIPE.

【Definiton】

```
typedef DC_S32 VI_CHN;
```

3.4.4 VI_CHN_ATTR_S

【Description】

VI channel attribute structure pointer.

【Definiton】

```
typedef char DC_CHAR;

typedef enum rkVI_CHN_WORK_MODE {
    VI_WORK_MODE_NORMAL = 0,
    VI_WORK_MODE_LUMA_ONLY
} VI_CHN_WORK_MODE;

typedef enum rkVI_CHN_BUF_TYPE {
VI_CHN_BUF_TYPE_DMA = 0, // Default
VI_CHN_BUF_TYPE_MMAP,
} VI_CHN_BUF_TYPE;

typedef struct rkVI_CHN_ATTR_S {
    const DC_CHAR *pcVideoNode;
```

```
        DC_U32 u32Width;
        DC_U32 u32Height;
        IMAGE_TYPE_E enPixFmt;
        DC_U32 u32BufCnt;
        VI_CHN_BUF_TYPE enBufType;
        VI_CHN_WORK_MODE enWorkMode;
    } VI_CHN_ATTR_S;
```

【Members】

Member name	Description
pcVideoNode	Video node path.
u32Width	Video widht
u32Height	Video height.
enPixFmt	Video format
u32BufCnt	VI capture video buffer count.
enWorkMode	VI channel working mode.

【Notice】

VI_WORK_MODE_LUMA_ONLY mode is used for VI brightness counting. In this mode, VI has no output and cannot obtain data from VI.

3.4.5 VIDEO_REGION_INFO_S

【Description】

Define the video region information.

【Definiton】

```
typedef structure rkVIDEO_REGION_INFO_S {
    DC_U32 u32RegionNum;      /* count of the region */
    RECT_S *pstRegion;        /* region attribute */
} VIDEO_REGION_INFO_S;
```

【Members】

Member name	Description
u32RegionNum	The number of video regions.
pstRegion	A pointer to the position information of a video region.

【Notice】

No.

3.4.5 VI_USERPIC_ATTR_S

【Description】

User picture attribute information.

【Definiton】

```
typedef struct rkVI_USERPIC_ATTR_S {
    IMAGE_TYPE_E enPixFmt;
    DC_U32 u32Width;
    DC_U32 u32Height;
    DC_VOID *pvPicPtr;
} VI_USERPIC_ATTR_S;
```

【Members】

Member name	Description
enPixFmt	The format of pictures.
u32Width	The width of the Picture.
u32Height	The height of the Picture.
pvPicptr	Picture's data.

【Notice】

No.

3.5 Error Code

Video input API error codes are as follows:

Error code	Macro definition	Description
10	DC_ ERR_VI_INVALID_CHNID	Invalid video input channel number
11	DC_ ERR_VI_BUSY	Video input system is busy
12	DC_ ERR_VI_EXIST	Video input channel already exists
13	DC_ ERR_VI_NOT_CONFIG	Video input is not configured
14	DC_ ERR_VI_TIMEOUT	Video input timeout
15	DC_ ERR_VI_BUF_EMPTY	Video input buffer is empty
16	DC_ ERR_VI_ILLEGAL_PARAM	Video input parameter setting is invalid
17	DC_ERR_VI_NOTREADY	The video input system is not initialized

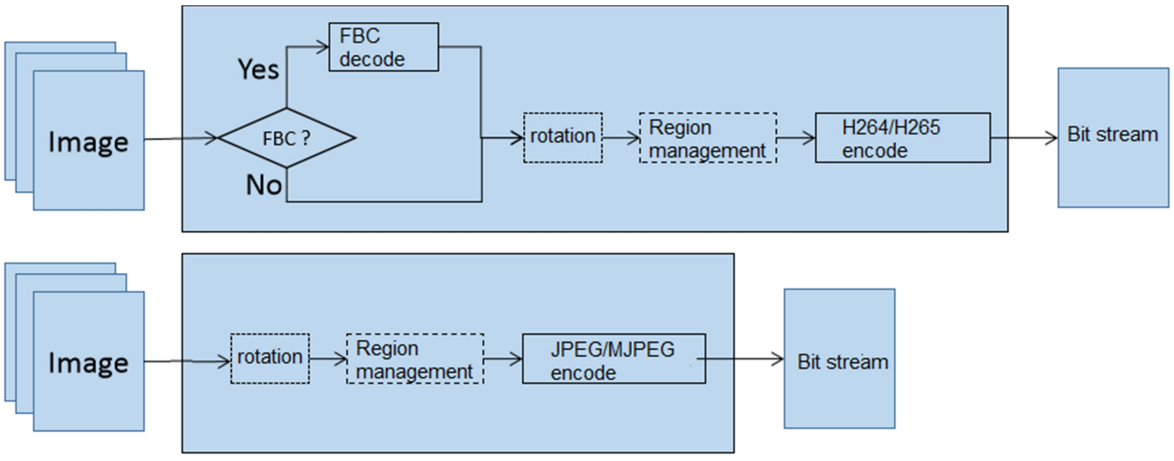
4. Video Encoding

4.1 Overview

VENC module is also called video encoding module. This module supports multi-channel real-time encoding, and each channel encodes independently, and the encoding protocol and encoding profile can be different. While supporting video encoding, the Region module is scheduled to overlay and cover the encoded image contents, supporting H264/H265/MJPEG/JPEG encoding.

4.2 Function Description

4.2.1 Data Flow Chart



Note: The functions described in the dotted rectangle are optional and will only be triggered when the encoder is configured accordingly.

4.2.2 Rate Control

Encoder type	Supported code control type
H264	CBR/VBR
H265	CBR/VBR
MJPEG	CBR/VBR

4.2.3 GOP Mode

GOP Mode is used to customize the dependency of reference frame, and the following modes are supported currently. Note: can be customized accordingly.

Name	Macro Definition	Description
Normal mode	VENC_GOPMODE_NORMALP	The most common scene, one I frame every GopSize
Smart P frame mode	VENC_GOPMODE_SMARTP	One Virtual I frame every GopSize and one I frame every BgInterval
Multi-layer time domain referende mode	VENC_GOPMODE_TSVC	The coding dependency relationship is divided into multiple layers, and layers information can be obtained according to DC_MPI_MB_GetTsvcLevel, and the code stream can be customized. For example, only play the 0th layer stream, which can realize fast preview.

4.2.4 Region of Interest

By configuring the region of interest of encoder, QP can be customized for the specified region. For example, a lens facing corridor, users are really interested in the center of corridor. ROI can be configured to make the coding quality in the center of the corridor higher and the image clearer, and the image quality of the non-interest region of the corridor (wall, ceiling, etc.) will be lower. In this way, user's region of interest is highlighted while keeping the bit rate nearly unchanged.

Each VENC channel provides 8 regions of interest, and the priority increases from REGION_ID_0 to REGION_ID_7. In regions where multiple ROIs overlap, the QP strategy will be configured according to the regions with high priority.

```
REGION_ID_0
REGION_ID_1
REGION_ID_2
REGION_ID_3
REGION_ID_4
REGION_ID_5
REGION_ID_6
REGION_ID_7
```

4.2.4 Region of Interest

The encoder supports 4 types of rotation, 0° , 90° , 180° , 270° . Encoder rotation does not support FBC format currently, and FBC format rotation needs to be achieved through ISPP rotation.

4.3 API Reference

4.3.1 DC_MPI_VENC_CreateChn

【Description】

Create an encoding channel.

【Grammar】

DC_S32 DC_MPI_VENC_CreateChn(VENC_CHN VeChn, VENC_CHN_ATTR_S *stVencChnAttr);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
stVencChnAttr	Encoding channel attribute pointer	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

Limitations of JPEG/MJPEG encoder：

① If there is FBC format or the zoom function is enabled, the encoder channel created at this time does not support dynamic resolution switching.
② Only 90 degree and 180 degree rotation are supported.
③ OSD will change the original buffer content of the input JPEG/MJPEG. There may be problems in the following cases:

VI[0] is bound to VENC[H264] and VENC[JPEG] at the same time, and JPEG is configured with OSD. At this time, the JPEG will directly add OSD on the original image output by VI, so the input data of H264

encoder will also probably have this OSD effect. At this time, you can add an RGA type channel before VENC[JPEG] to avoid directly add OSD on the VI output original image.

【Example】

No.

【Related topic】

No.

4.3.2 DC_MPI_VENC_GetVenc ChnAttr

【Description】

Get encoding channel Parameters.

【Grammar】

DC_S32 DC_MPI_VENC_GetVencChnAttr(VENC_CHN VeChn, VENC_CHN_ATTR_S *stVencChnAttr);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
stVencChnAttr	Encoding channel attribute pointer	Output

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

This function can get the parameter configuration of the created channels only.

【Example】

No.

【Related topic】

No.

4.3.3 DC_MPI_VENC_SetVencChnAttr

【Description】

Get encoding channel Parameters.

【Grammar】

DC_S32 DC_MPI_VENC_SetVencChnAttr(VENC_CHN VeChn, VENC_CHN_ATTR_S *stVencChnAttr);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
stVencChnAttr	Encoding channel attribute pointer	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

This function can configure parameters of the created channels only. Currently, it supports encoding complexity (H264 only), resolution, bit rate, frame rate, GOP dynamic settings. The other configuration changes need to be destroyed and then recreated.

The resolution dynamic configuration needs to keep the width and height information of the encoder input buffer consistent with dynamic configuration, otherwise it will cause the risk of memory access overrun. This interface is only recommended to be used when VENC is not bound. And make sure that the input buffer of VENC channel has been cleared before changing resolution (see RK_MPI_VENC_QueryStatus)

【Example】

No.

【Related topic】

No.

4.3.4 DC_MPI_VENC_GetVencChnParam

【Description】

To get parameters of encoding channel.

【Grammar】

DC_S32 DC_MPI_VENC_GetVencChnParam(VENC_CHN VeChn, VENC_CHN_PARAM_S *stVencChnParam);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
stVencChnParam	Encoding channel parameter pointer	Output

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

This function can only get the parameter configuration of the created channel.

【Example】

No.

【Related topic】

No.

4.3.5 DC_MPI_VENC_SetVencChnParam

【Description】

Set parameters of encoding channel.

【Grammar】

DC_S32 DC_MPI_VENC_SetVencChnParam(VENC_CHN VeChn, VENC_CHN_PARAM_S *stVencChnParam);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
stVencChnParam	Encoding channel parameter pointer	Output

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

This function can only configure the parameters of the created channels. Currently supports encoding complexity (H264 only), resolution, bit rate, frame rate, GOP dynamic settings. The rest of the configuration modification need to be destroyed and then recreated.

The resolution dynamic configuration needs to keep the width and height information of the encoder input buffer consistent with the dynamic configuration, otherwise it will cause the risk of memory access crossing the boundary. This interface is only recommended for the case which VENC does not use Bind. And make sure that the input buffer of VENC Channel has been cleared before changing the resolution (please refer to DC_MPI_VENC_QueryStatus).

【Example】

No.

【Related topic】

No.

4.3.6 DC_MPI_VENC_DestroyChn

【Description】

Destroy the encoding channel.

【Grammar】

DC_S32 DC_MPI_VENC_DestroyChn(VENC_CHN VeChn);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

4.3.7 DC_MPI_VENC_GetRcParam

【Description】

Get the bit rate control parameters.

【Grammar】

DC_S32 DC_MPI_VENC_GetRcParam(VENC_CHN VeChn, VENC_RC_PARAM_S *pstRcParam);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
pstRcParam	The advanced parameters of the code rate controller	Output

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

4.3.8 DC_MPI_VENC_SetRcParam

【Description】

Set bit rate control parameters.

【Grammar】

DC_S32 DC_MPI_VENC_SetRcParam(VENC_CHN VeChn, VENC_CHN_PARAM_S *pstRcParam);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
pstRcParam	The advanced parameters of the code rate controller	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

There are default values of the advanced parameters of the encoding channel rate controller, instead of having to call this interface to start the encoding channel.

It is recommended that users call the DC_MPI_VENC_GetRcParam interface to obtain RC advanced parameters firstly, secondly, modify the corresponding parameters, and then call this interface to set the advanced parameters.

RC advanced parameters now only support parameters setting of rows level and macroblock level code control under CBR/VBR rate control mode of H.264/H.265/Mjpeg. The advanced parameters of the code rate controller including the following parameters:

① u32ThrdI[RC_TEXTURE_THR_SIZE], u32ThrdP[RC_TEXTURE_THR_SIZE] ： a group of thresholds for measuring the complexity of macroblocks of I frame and P frame respectively. This group of thresholds are arranged in order from small to large, and the value range of each threshold is [0, 255]. This group of Parameter name Description Input/Output VeChn Encoding channel number. Input RcMode Rate control mode. Input thresholds is used to appropriately adjust the Qp of each macroblock according to the image complexity when performing macroblock level rate control.

② u32RowQpDeltaI, u32RowQpDeltaP：when in macroblock level code rate control, the fluctuation amplitude value of the starting Qp of each row of macroblocks relative to the starting Qp of frames. For scenes with strict code rate fluctuations, you can try to increase this parameter to achieve more precise code rate control, but it may cause differences in image quality within certain frames. When the bit rate is high, the value is recommended to be 0; when the bit rate is medium, the value is recommended to be 0 or 1; when the bit rate is low, the value is recommended to be 2~5.

③ s32FirstFrameStartQp ： the starting Qp value of the first frame, CBR/VBR/AVBR are valid. If s32FirstFrameStartQp is −1, the starting QP of the first frame is calculated internally by the encoder, and the value is −1 by default. The meaning of the first frame here is: channel creation, after Gop mode switching, RC mode switching, or resolution switching, the first IDR frame of the sequence.

H264/H265/MJPEG CBR/VBR/AVBR advanced parameter settings:

① u32StepQp : reserved, meaningless.

② u32MaxQp and u32MinQp represent the maximum Qp and minimum Qp of the current frames. This clamping effect is the strongest. All other adjustments to the image Qp, such as macroblock-level rate control, will eventually be constrained to this maximum Qp and minimum Qp. The default value u32MinQp is 8 and u32MaxQp is 48. If there is no special requirement for quality, it is recommended not to change this group of parameters.

③ u32MaxIQp and u32MinIQp represent the maximum Qp and minimum Qp of the current sequence IDR frames. This clamping effect is the strongest. All other adjustments to the image Qp, such as macroblocklevel rate control, will eventually be constrained to this maximum Qp and minimum Qp. The default value u32MinIQp is 8, u32MaxIQp is 48. If there is no special requirement for quality, it is recommended not to change this group of parameters.

【Example】

No.

【Related topic】

No.

4.3.9 DC_MPI_VENC_SetRcMode

【Description】

Set the bit rate control mode.

【Grammar】

DC_S32 DC_MPI_VENC_SetRcMode(VENC_CHN VeChn, VENC_RC_MODE_E RcMode);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
RcMode	Rate control mode.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

4.3.10 DC_MPI_VENC_SetRcQuality

【Description】

Set the encoding quality for H264/H265 encoder.

【Grammar】

DC_S32 DC_MPI_VENC_SetRcQuality(VENC_CHN VeChn, VENC_RC_QUALITY_E RcQuality);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
RcQuality	Encoding quality.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

4.3.11 DC_MPI_VENC_SetBitrate

【Description】

Set the bit rate.

【Grammar】

DC_S32 DC_MPI_VENC_SetBitrate(VENC_CHN VeChn, DC_U32 u32BitRate, DC_U32 u32MinBitRate, DC_U32 u32MaxBitRate);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
u32BitRate	Target bit rate.	Input
u32MinBitRate	The minimum bit rate. Unit bps	Input
u32MaxBitRate	The maximum bit rate. Unit bps	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

4.3.12 DC_MPI_VENC_RequestIDR

【Description】

Request IDR frame. After calling this interface, the encoder refreshes IDR frame immediately.

【Grammar】

DC_S32 DC_MPI_VENC_RequestIDR(VENC_CHN VeChn, DC_BOOL bInstant);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
bInstant	Whether to enable IDR frame encoding immediately.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

4.3.13 DC_MPI_VENC_SetFps

【Description】

Set the encoding frame rate.

【Grammar】

DC_S32 DC_MPI_VENC_RequestIDR(VENC_CHN VeChn, DC_U8 u8OutNum, DC_U8 u8OutDen, DC_U8 u8InNum, DC_U8 u8InDen);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
u8OutNum	Denorminator of the encoding output frame rate.	Input
u8OutDen	Numerator of the encoding output frame rate.	Input
u8InNum	Denominater of the encoding input frame rate.	Input
u8InDen	Numerator of the encoding input frame rate.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

The output frame rate cannot be greater than the input frame rate.

【Example】

No.

【Related topic】

No.

4.3.14 DC_MPI_VENC_SetGop

【Description】

Set GOP for H264/H265 encoder.

【Grammar】

DC_S32 DC_MPI_VENC_SetGop(VENC_CHN VeChn, DC_U32 u32Gop);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
u32Gop	GOP	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

4.3.15 DC_MPI_VENC_SetAvcProfile

【Description】

Set profile for H264 encoder.

【Grammar】

DC_S32 DC_MPI_VENC_SetAvcProfile(VENC_CHN VeChn, DC_U32 u32Profile, DC_U32 u32Level);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
u32Profile	Profile IDC value.	Input
u32Level	Level IDC value.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

The supported u32Profile are 66, 77, 100 only, corresponding to Baseline Main Profile, and High Profile respectively at present.

【Example】

No.

【Related topic】

No.

4.3.16 DC_MPI_VENC_InsertUserData

【Description】

Inserted user data which will be reflected in the SEI packet of code stream, and it is used for H264/265 edcoder.

【Grammar】

DC_S32 DC_MPI_VENC_InsertUserdata(VENC_CHN VeChn, DC_U8 *pu8Data, DC_U32 u32Len);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
pu8Data	User data pointer.	Input
u32Len	User data length.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

4.3.17 DC_MPI_VENC_SetResolution

【Description】

Set VENC channel resolution.

【Grammar】

DC_S32 DC_MPI_VENC_SetResolution(VENC_CHN VeChn, VENC_RESOLUTION_PARAM_S *stResolutionParam);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input

stResolutionParam	Resolution parameter structure.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

4.3.18 DC_MPI_VENC_GetRoiAttr

【Description】

Get ROI configuration parameters of the specified index value, it is used for H264/265 edcoder.

【Grammar】

DC_S32 DC_MPI_VENC_GetRoiAttr(VENC_CHN VeChn, VENC_ROI_ATTR_S *pstRoiAttr, DC_S32 roi_index);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
pstRoiAttr	ROI region parameters.	Input
roi_index	ROI region index value. (0 ~ 7).	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

4.3.19 DC_MPI_VENC_SetRoiAttr

【Description】

Set ROI coding region of interest, it is used for H264/265 edcoder.

【Grammar】

```
DC_S32 DC_MPI_VENC_SetRoiAttr(VENC_CHN VeChn, cosnt VENC_ROI_ATTR_S *pstRoiAttr,
                              DC_S32 region_cnt);
```

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
pstRoiAttr	ROI region parameters.	Input
roi_index	The number of ROI regions. (1 ~ 8).	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

4.3.20 DC_MPI_VENC_SetGopMode

【Description】

Set GopMode, it is used for H264/265 edcoder.

【Grammar】

DC_S32 DC_MPI_VENC_SetGopMode(VENC_CHN VeChn, cosnt VENC_GOP_ATTR_S GopMode);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
GopMode	GOP attribute structure	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

4.3.21 DC_MPI_VENC_RGN_Init

【Description】

Initialize the VENC RGN module. Each VENC_CHN needs call function for initialization before using the VENC RGN interface.

【Grammar】

DC_S32 DC_MPI_VENC_RGN_Init(VENC_CHN VeChn, VENC_COLOR_TBL_S *stColorTbl);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
stColorTbl	256 color palette, supports ARGB8888 format only, setting NULL to use the default palette.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

Before calling DC_MPI_VENC_RGN_SetBitMap or DC_MPI_VENC_RGN_SetCover, this interface must be called first, and each encoding channel can only be called once..

【Example】

No.

4.3.22 DC_MPI_VENC_RGN_SetBitMap

【Description】

Set OSD bitmap. Supports ARGB8888 format bitmap Only.

【Grammar】

DC_S32 DC_MPI_VENC_RGN_SetBitMap(VENC_CHN VeChn, const OSD_REGION_INFO_S *pstRgnInfo, const BITMAP_S *pstBitmap);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
pstRgnInfo	OSD region information.	Input
pstBitmap	Bitmap information and data.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

Before calling this interface, you must call DC_MPI_VENC_RGN_Init firstly. This interface shares 8 layers of the encoder with DC_MPI_VENC_RGN_SetCover, please see OSD_REGION_INFO_S for details.

【Example】

No.

【Related topic】

No.

4.3.23 DC_MPI_VENC_RGN_SetCover

【Description】

Set privacy cover. the efficiency of monochrome cover with RGB8888 is higher than RK_MPI_VENC_RGN_SetBitMap.

【Grammar】

```
DC_S32 DC_MPI_VENC_RGN_SetCover(VENC_CHN VeChn, const OSD_REGION_INFO_S *pstRgnInfo,
                                const COVER_INFO_S *pstCoverInfo);
```

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
pstRgnInfo	RGN region information.	Input
pstCoverInfo	Privacy covering information.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

Before calling this interface, you must call DC_MPI_VENC_RGN_Init firstly. This interface shares 8 layers of the encoder with DC_MPI_VENC_RGN_SetBitmap, please refer to OSD_REGION_INFO_S.

【Example】

No.

【Related topic】

No.

4.3.24 DC_MPI_VENC_RGN_SetPaletteId

【Description】

Palette index is used to build buffer for OSD overlay. It will be higher efficiency without matching color palette. In addition, compared with argb8888 format buffer, the memory consumption of using index to build buffer is reduced to 1/4.

【Grammar】

```
DC_S32 DC_MPI_VENC_RGN_SetPaletteId(VENC_CHN VeChn, const OSD_REGION_INFO_S *pstRgnInfo,
                                    const OSD_COLOR_PALETTE_BUF_S *pstColPalBuf);
```

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
pstRgnInfo	RGN region information.	Input
pstColPalBuf	OSD buffer constructed by palette index.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

Before calling this interface, you must call DC_MPI_VENC_RGN_Init firstly. This interface shares 8 layers of the encoder with DC_MPI_VENC_RGN_SetBitmap, please refer to OSD_REGION_INFO_S.

【Example】

No.

【Related topic】

No.

4.3.25 DC_MPI_VENC_RGN_SetJpegParam

【Description】

Set JPEG encoding parameters.

【Grammar】

DC_S32 DC_MPI_VENC_RGN_SetJpegParam(VENC_CHN VeChn, const VENC_JPEG_PARAM_S *pstJpegParam);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
pstRgnInfo	RGN region information.	Input
pstColPalBuf	OSD buffer constructed by palette index.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

Before calling this interface, you must call DC_MPI_VENC_RGN_Init firstly. This interface shares 8 layers of the encoder with DC_MPI_VENC_RGN_SetBitmap, please refer to OSD_REGION_INFO_S.

【Example】

No.

【Related topic】

No.

4.3.26 DC_MPI_VENC_StartRecvFrame

【Description】

Set the number of frames received by encoder. The encoder created by default will receive VI data continuously. The number of received frames can be set by the DC_MPI_VENC_StartRecvFrame interface. After reaching specified number, the encoder will go to sleep until next time the interface is called to change the number of received frames.

【Grammar】

```
DC_S32 DC_MPI_VENC_RGN_StartRecvFrame(VENC_CHN VeChn,
                                      const VENC_RECV_PIC_PARAM_S *pstRecvParam);
```

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
pstRecvParam	Receive image parameter structure pointer which is used to specify the number of image frames to be received.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

4.3.27 DC_MPI_VENC_GetFd

【Description】

Get the file descriptor of the encoder channel.

【Grammar】

DC_S32 DC_MPI_VENC_RGN_GetFd(VENC_CHN VeChn);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input

【Return value】

Return value type	Description
DC_S32	File Descriptor

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

4.3.28 DC_MPI_VENC_QueryStatus

【Description】

Get the encoder channel status.

【Grammar】

DC_S32 DC_MPI_VENC_RGN_QueryStatus(VENC_CHN VeChn, VENC_RECV_PIC_PARAM_S *pstStatus);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
pstStatus	Encoder status structure.	Output

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

4.3.29 DC_MPI_VENC_SetSuperFrameStrategy

【Description】

Set the super frame related configuration of the encoding channel.

【Grammar】

DC_S32 DC_MPI_VENC_RGN_SetSuperFrameStrategy(VENC_CHN VeChn,

cosnt VENC_SUPERFRAME_CFG_S *pstSuperFrmParam);

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
pstSuperFrmParam	Super frame configuration structure of a coding channel.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

If the channel is not created, it returns failure. This interface is a high-level interface, and users can call it optionally. The system default value, please refer to VENC_SUPERFRAME_CFG_S. During the disabling period, if the encoder detects that the code rate is greater than 1.5 times the set code rate, it will restart encoding once. This interface is set after the encoding channel is created and before the encoding channel is destroyed.

【Example】

No.

【Related topic】

No.

4.3.30 DC_MPI_VENC_GetSuperFrameStrategy

【Description】

Obtain the configurations related to the super frame of the encoding channel.

【Grammar】

```
DC_S32 DC_MPI_VENC_RGN_SetSuperFrameStrategy(VENC_CHN VeChn,
                                             cosnt VENC_SUPERFRAME_CFG_S *pstSuperFrmParam);
```

【Parameter】

Parameter name	Description	Input/Output
VeChn	Encoding channel number. (0 ~ 16)	Input
pstSuperFrmParam	Super frame configuration structure of a coding channel.	Output

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

This interface is meaningful only after calling RK_MPI_VENC_SetSuperFrameStrategy.

【Example】

No.

【Related topic】

No.

4.4 Type of Data

The video coding related data types are defined as follows:

- VENC_MAX_CHN_NUM: define the total number of VENC physical channels and extended channels.
- VENC_CHN: VENC channel number.
- VENC_ATTR_JPEG_S: define the attribute structure of JPEG capture encoder.
- VENC_ATTR_MJPEG_S: define the attribute structure of MJPEG encoder.
- VENC_ATTR_H264_S: define H.264 encoder attribute structure.
- VENC_ATTR_H265_S: define H.265 encoder attribute structure.
- VENC_ATTR_S: define the encoder attribute structure.
- VENC_MJPEG_CBR_S: define CBR attribute structure of MJPEG encoding channel.
- VENC_MJPEG_VBR_S: define VBR attribute structure of MJPEG encoding channel.
- VENC_H264_CBR_S: define CBR attribute structure of H.264 encoding cha nnel.
- VENC_H264_VBR_S: define VBR attribute structure of H.264 encoding channel.
- VENC_H265_CBR_S: define CBR attribute structure of H.265 encoding channel.
- VENC_H265_VBR_S: define VBR attribute structure of H.265 encoding channel.
- VENC_RC_MODE_E: define the code rate controller mode of the encoding channel.
- VENC_RC_ATTR_S: define the code rate controller attributes of the encoding channel.
- VENC_GOP_MODE_E: define Gop mode type.

- VENC_GOP_ATTR_S: define the encoder GOP attribute structure.
- VENC_CHN_ATTR_S: VENC channel attribute structure.
- VENC_PARAM_MJPEG_S: MJPEG channel parameters.
- VENC_PARAM_H264_S: H.264 channel parameters.
- VENC_PARAM_H265_S: H.265 channel parameters.
- VENC_RC_PARAM_S: the advanced parameters of the code rate controller of the encoding channel.
- VENC_RC_QUALITY_E: encoding quality.
- VENC_ROI_ATTR_S: ROI attribute structure.
- OSD_REGION_ID_E: OSD region ID enumeration type.
- OSD_REGION_INFO_S: OSD region information.
- OSD_PIXEL_FORMAT_E: OSD pixel format type enumeration.
- BITMAP_S: bitmap information and data.
- COVER_INFO_S: privacy cover information.
- VENC_RECV_PIC_PARAM_S: receive image parameter structure pointer, is used to specify the number of image frames to be received.
- VENC_JPEG_PARAM_S: advanced parameters of JPEG protocol encoding channel.

4.4.1 VENC_MAX_CHN_NUM

【Description】

The total number of VENC physical channels and expansion channels.

【Definiton】

```
#defime VENC_MAX_CHN_NUM 16
```

4.4.2 VENC_CHN

【Description】

VENC channel number.

【Definiton】

```
typedef DC_S32 VENC_CHN
```

4.4.3 VENC_ATTR_JPEG_S

【Description】

Define the attribute structure of JPEG capture encoder.

【Definiton】

```
typedef struct rkVENC_ATTR_JPEG_S {
    DC_U32 u32ZoomWidth;     // Zoom to specified width
    DC_U32 u32ZoomHeight;    // Zoom to specified height
    DC_U32 u32ZoomVirWidth;
    DC_U32 u32ZoomVirHeight;
} VENC_ATTR_JPEG_S;
```

【Members】

Member name	Description
u32ZoomWidth	Zoom to the specified width.
u32ZoomHeight	Zoom to the specified height
u32ZoomVirWidth	Zoom to the virtual width.
u32ZoomVirHeight	Zoom to the virtual height.

4.4.4 VENC_ATTR_MJPEG_S

【Description】

Define the attribute structure of MJPEG encoder.

【Definiton】

```
typedef struct rkVENC_ATTR_MJPEG_S {
    DC_U32 u32ZoomWidth;     // Zoom to specified width
    DC_U32 u32ZoomHeight;    // Zoom to specified height
    DC_U32 u32ZoomVirWidth;
    DC_U32 u32ZoomVirHeight;
} VENC_ATTR_MJPEG_S;
```

【Members】

Member name	Description
u32ZoomWidth	Zoom to the specified width.
u32ZoomHeight	Zoom to the specified height
u32ZoomVirWidth	Zoom to the virtual width.
u32ZoomVirHeight	Zoom to the virtual height.

4.4.5 VENC_ATTR_H264_S

【Description】

Define the attribute structure of H.264 encoder.

【Definiton】

```
typedef struct rkVENC_ATTR_H264_S {
    DC_U32 u32Level;
} VENC_ATTR_H264_S;
```

【Members】

Member name	Description
u32Level	Profile IDC value.

4.4.6 VENC_ATTR_H265_S

【Description】

Define the attribute structure of H.265 encoder.

【Definiton】

```
typedef struct rkVENC_ATTR_H264_S {
    DC_BOOL bScaleList;
} VENC_ATTR_H264_S;
```

【Members】

Member name	Description
bScaleList	Scale List.

4.4.7 VENC_ATTR_S

【Description】

Define the encoder attribute structure.

【Definiton】

```
typedef struct rkVENC_ATTR_S {
    CODEC_TYPE_E enType;        // RW; the type of encodec
    IMAGE_TYPE_E imageType;     // the type of input image
    DC_U32 u32VirWidth;         // stride width, same to buffer_width,
    DC_U32 u32VirHeight;        // stride height, same to buffer_height,
    DC_U32 u32Profile;          // RW;
                                // H.264: 66: baseline; 77:MP; 100:HP;
                                // H.265: default:Main;
                                // Jpege/MJpege: default:Baseline
    DC_BOOL bByFrame; // reserve
    DC_U32 u32PicWidth; // RW; width of a picture to be encoded, in pixel
    DC_U32 u32PicHeight; // RW; height of a picture to be encoded, in pixel
    VENC_ROTATION_E enRotation;
    union {
        VENC_ATTR_H264_S stAttrH264e; // attributes of H264e
        VENC_ATTR_H265_S stAttrH265e; // attributes of H265e
        VENC_ATTR_MJPEG_S stAttrMjpege; // attributes of Mjpeg
        VENC_ATTR_JPEG_S stAttrJpege; // attributes of jpeg
    };
} VENC_ATTR_S;
```

【Members】

Member name	Description
enType	Encoding protocol type.
imageType	Input image type.
u32VirWidth	stride width, usually is the same as buffer_width. If u32VirWidth is greater than buffer width, it must be 16 alignment.
u32VirHeight	stride height, usually is the same as buffer_height. If u32VirHeight is greater than buffer height, it must be16 alignment.
u32Profile	The level of encoding. H.264: 66: Baseline; 77: Main Profile; 100: High Profile; H.265: default:Main; Jpege/MJpege: default:Baseline
bByFrame	Reserved parameter, not supported currently.

u32PicWidth	Encoded image width. In pixels.
u32PicHeight	The height of the encoded image. In pixels.
stAttrH264e/stAttrH265e/stAttrMjpege/stAttrJpege	Encoder attributes of one protocol.

4.4.8 VENC_MJPEG_CBR_S

【Description】

Define the CBR attribute structure of MJPEG encoding channel.

【Definiton】

```
typedef struct rkVENC_MJPEG_CBR_S {
    DC_U32 u32SrcFrameRateNum;
    DC_U32 u32SrcFrameRateDen;
    DC_FR32 fr32DstFrameRateNum;
    DC_FR32 fr32DstFrameRateDen;
    DC_U32 u32BitRate;      // RW; Range:[2000, 98000000]; average bitrate
} VENC_MJPEG_CBR_S;
```

【Members】

Member name	Description
u32SrcFrameRateNum	Numerator of data source frame rate.
u32SrcFrameRateDen	Denominator of data source frame rate.
fr32DstFrameRateNum	Numerator of target frame rate.
Numeratorfr32DstFrameRateDen	Denominator of target frame rate.
u32BitRate	Average bit rate, Values range: [2000, 98000000].

4.4.9 VENC_MJPEG_VBR_S

【Description】

Defines the VBR attribute structure of MJPEG encoding channel.

【Definiton】

```
typedef struct rkVENC_MJPEG_VBR_S {
    DC_U32 u32SrcFrameRateNum;
    DC_U32 u32SrcFrameRateDen;
    DC_FR32 fr32DstFrameRateNum;
    DC_FR32 fr32DstFrameRateDen;
    DC_U32 u32BitRate;      // RW; Range:[2000, 98000000]; average bitrate
} VENC_MJPEG_VBR_S;
```

【Members】

Member name	Description
u32SrcFrameRateNum	Numerator of data source frame rate.
u32SrcFrameRateDen	Denominator of data source frame rate.
fr32DstFrameRateNum	Numerator of target frame rate.
Numeratorfr32DstFrameRateDen	Denominator of target frame rate.
u32BitRate	Average bit rate, Values range: [2000, 98000000].

4.4.10 VENC_H264_CBR_S

【Description】

Defines the CBR attribute structure of H.264 encoding channel.

【Definiton】

```
typedef struct rkVENC_H264_CBR_S {
    DC_U32 u32Gop;          // RW; Range:[1, 65536]; the interval of I Frame.
    DC_U32 u32SrcFrameRateNum;
    DC_U32 u32SrcFrameRateDen;
    DC_FR32 fr32DstFrameRateNum;
    DC_FR32 fr32DstFrameRateDen;
    DC_U32 u32BitRate;      // RW; Range:[2000, 98000000]; average bitrate
} VENC_H264_CBR_S;
```

【Members】

Member name	Description
u32Gop	I frame interval, Values range: [1, 65536].
u32SrcFrameRateNum	Numerator of data source frame rate.
u32SrcFrameRateDen	Denominator of data source frame rate.
fr32DstFrameRateNum	Numerator of target frame rate.
Numeratorfr32DstFrameRateDen	Denominator of target frame rate.
u32BitRate	Average bit rate, Values range: [2000, 98000000].

4.4.11 VENC_H264_VBR_S

【Description】

Defines the VBR attribute structure of H.264 encoding channel.

【Definiton】

```
typedef struct rkVENC_H264_VBR_S {
    DC_U32 u32Gop;              // RW; Range:[1, 65536]; the interval of I Frame.
    DC_U32 u32SrcFrameRateNum;
    DC_U32 u32SrcFrameRateDen;
    DC_FR32 fr32DstFrameRateNum;
    DC_FR32 fr32DstFrameRateDen;
    DC_U32 u32MaxBitRate;       // RW; Range:[2000, 98000000]; the max bitrate
} VENC_H264_VBR_S;
```

【Members】

Member name	Description
u32Gop	I frame interval, Values range: [1, 65536].
u32SrcFrameRateNum	Numerator of data source frame rate.
u32SrcFrameRateDen	Denominator of data source frame rate.
fr32DstFrameRateNum	Numerator of target frame rate.
Numeratorfr32DstFrameRateDen	Denominator of target frame rate.
u32MaxBitRate	Maximum bit rate, Values range: [2000, 98000000].

4.4.12 VENC_H265_CBR_S

【Description】

Define the CBR attribute structure of H.265 encoding channel.

【Definiton】

```
typedef struct rkVENC_H264_CBR_S VENC_H265_CBR_S;
```

4.4.13 VENC_H265_VBR_S

【Description】

Define the VBR attribute structure of H.265 encoding channel.

【Definiton】

```
typedef struct rkVENC_H264_VBR_S VENC_H265_VBR_S;
```

4.4.14 VENC_RC_MODE_E

【Description】

Define code rate controller mode of the encoding channel.

【Definiton】

```
typedef enum rkVENC_RC_MODE_E {
    // H264
    VENC_RC_MODE_H264CBR = 1,
    VENC_RC_MODE_H264VBR,
    VENC_RC_MODE_H264AVBR,
    // MJPEG
    VENC_RC_MODE_MJPEGCBR,
    VENC_RC_MODE_MJPEGVBR,
    // H265
    VENC_RC_MODE_H265CBR,
    VENC_RC_MODE_H265VBR,
    VENC_RC_MODE_H265AVBR,
    VENC_RC_MODE_BUTT,
} VENC_RC_MODE_E;
```

4.4.15 VENC_RC_ATTR_S

【Description】

Define code channel rate controller properties.

【Definiton】

```
typedef struct rkVENC_RC_ATTR_S {
    /* RW; the type of rc*/
    VENC_RC_MODE_E enRcMode;
    union {
        VENC_H264_CBR_S stH264Cbr;
        VENC_H264_VBR_S stH264Vbr;
        VENC_H264_AVBR_S stH264Avbr;
        VENC_MJPEG_CBR_S stMjpegCbr;
        VENC_MJPEG_VBR_S stMjpegVbr;
        VENC_H265_CBR_S stH265Cbr;
        VENC_H265_VBR_S stH265Vbr;
        VENC_H265_AVBR_S stH265Avbr;
    };
} VENC_RC_ATTR_S;
```

【Members】

Member name	Description
enRcMode	Encoding protocol type.
stH264Cbr	Cbr mode attribute of H.264 protocol encoding channel.
stH264Vbr	Vbr mode attribute of H.264 protocol encoding channel.
stMjpegCbr	Cbr mode attribute of MJPEG protocol encoding channel.
stMjpegVbr	Vbr mode attribute of MJPEG protocol encoding channel.
stH265Cbr	Cbr mode attribute of H.265 protocol encoding channel.
stH265Vbr	Vbr mode attributes of H.265 protocol encoding channel.

4.4.16 VENC_GOP_MODE_E

【Description】

Define code rate controller mode of the encoding channel.

【Definiton】

```
typedef enum rkVENC_GOP_MODE_E {
    VENC_GOPMODE_NORMALP = 0,
    VENC_GOPMODE_TSVC,
    VENC_GOPMODE_SMARTP,
    VENC_GOPMODE_BUTT,
} VENC_GOP_MODE_E;
```

【Notice】

For detailed mode description, please refer to GOP Mode..

4.4.17 VENC_GOP_ATTR_S

【Description】

Define encoder GOP attribute structure.

【Definiton】

```
typedef struct rkVENC_GOP_ATTR_S {
    VENC_GOP_MODE_E enGopMode;
    DC_U32 u32GopSize;
    DC_S32 s32IPQpDelta;
    DC_U32 u32BgInterval;
    DC_S32 s32ViQpDelta;
} VENC_GOP_ATTR_S;
```

【Members】

Member name	Description
enGopMode	Encoding GOP type.
u32GopSize	Encoding GOP size.
s32IPQpDelta	QP difference between I frame and P frame.
u32BgInterval	Long-term reference frame interval.
s32ViQpDelta	The QP difference between virtual I frame and regular P frame.

4.4.18 VENC_GOP_ATTR_S

【Description】

VENC channel attribute structure.

【Definiton】

```
typedef struct rkVENC_CHN_ATTR_S {
    VENC_ATTR_S stVencAttr;         // the attribute of video encoder
    VENC_RC_ATTR_S stRcAttr;        // the attribute of rate ctrl
    VENC_GOP_ATTR_S stGopAttr;      // the attribute of gop
} VENC_CHN_ATTR_S;
```

【Members】

Member name	Description
stVencAttr	Encoder attributes.
stRcAttr	Bit rate controller attributes.
stGopAttr	GOP attributes.

4.4.19 VENC_CHN_PARAM_S

【Description】

VENC channel parameter structure.

【Definiton】

```
typedef struct rkVENC_CHN_PARAM_S {
    DC_BOOL bColor2Grey;
    DC_U32 u32Priority;
    DC_U32 u32MaxStrmCnt;
    DC_U32 u32PollWakeUpFrmCnt;
    VENC_CROP_INFO_S stCropCfg;
    VENC_FRAME_RATE_S stFrameRate;
} VENC_CHN_PARAM_S;
```

【Members】

Member name	Description
bColor2Grey	Color to gray is enabled.
u32Priority	Channel priority, 0 1 2 3, non-preemptive.
u32MaxStrmCnt	Maximum number of stream frames.
u32PollWakeUpFrmCnt	The timeout threshold of channel obtaining code stream, the unit is the number of frames.
stCropCfg	Crop parameters.
stFrameRate	Frame rate control.

4.4.20 VENC_CROP_INFO_S

【Description】

VENC cropping parameter structure.

【Definiton】

```
typedef struct rkVENC_CROP_INFO_S {
    DC_BOOL bEnable;
    RECT_S stRect;
} VENC_CROP_INFO_S;
```

【Members】

Member name	Description
bEnable	Enable cropping.
stRect	Rectangular frame.

4.4.21 VENC_FRAME_RATE_S

【Description】

VENC frame rate information structure.

【Definiton】

```
typedef struct rkVENC_FRAME_RATE_S {
    DC_S32 s32SrcFrmRate;
    DC_S32 s32DstFrmRate;
} VENC_FRAME_RATE_S;
```

【Members】

Member name	Description
s32SrcFrmRate	Source frame rate.
s32DstFrmRate	Target frame rate.

4.4.22 VENC_PARAM_MJPEG_S

【Description】

MJPEG channel parameters.

【Definiton】

```
typedef struct rkVENC_PARAM_MJPEG_S {
    // reserved
} VENC_PARAM_MJPEG_S;
```

4.4.23 VENC_PARAM_H264_S

【Description】

H.264 channel parameters.

【Definiton】

```
typedef struct rkVENC_PARAM_H264_S {
    DC_U32 u32StepQp;
    DC_U32 u32MaxQp;
    DC_U32 u32MinQp;
    DC_U32 u32MaxIQp;
    DC_U32 u32MinIQp;
} VENC_PARAM_H264_S;
```

【Members】

Member name	Description
u32StepQp	The step value of QP.
u32MaxQp	QP maximum value, values range [8, 51].
u32MinQp	QP minimum value, values range [0, 48], cannot be greater than u32MaxQp.
u32MaxIQp	QP maximum value of I frame.
u32MinIQp	QP minimum value of I frame.

4.4.24 VENC_PARAM_H265_S

【Description】

H.265 channel parameters.

【Definiton】

```
typedef struct rkVENC_PARAM_H265_S {
    DC_U32 u32StepQp;
    DC_U32 u32MaxQp;
    DC_U32 u32MinQp;
    DC_U32 u32MaxIQp;
    DC_U32 u32MinIQp;
} VENC_PARAM_H265_S;
```

【Members】

Member name	Description
u32StepQp	The step value of QP.
u32MaxQp	QP maximum value, values range [8, 51].
u32MinQp	QP minimum value, values range [0, 48], cannot be greater than u32MaxQp.
u32MaxIQp	QP maximum value of I frame.
u32MinIQp	QP minimum value of I frame.

4.4.25 VENC_RC_PARAM_S

【Description】

The advanced parameters of the code rate controller of encoding channel.

【Definiton】

```
typedef struct rkVENC_RC_PARAM_S {
    DC_U32 u32ThrdI[RC_TEXTURE_THR_SIZE];       // [0, 255]
    DC_U32 u32ThrdP[RC_TEXTURE_THR_SIZE];       // [0, 255
    DC_U32 u32RowQpDeltaI;
    DC_U32 u32RowQpDeltaP;
    // hierachy qp cfg
    DC_BOOL bEnableHierQp;
    DC_S32 s32HierQpDelta[RC_HEIR_SIZE];
    DC_S32 s32HierFrameNum[RC_HEIR_SIZE]
    DC_U32 s32FirstFrameStartQp;        // RW; Start QP value of the first frame
    union {
        VENC_PARAM_H264_S stParamH264;
        VENC_PARAM_H265_S stParamH265;
        VENC_PARAM_MJPEG_S stParamMjpeg;
    };
} VENC_RC_PARAM_S;
```

【Members】

Member name	Description
u32ThrdI	I-frame macroblock QP threshold [0,255].
u32ThrdP	P-frame macroblock QP threshold [0,255].
u32RowQpDeltaI	In the macroblock-level code rate control, the fluctuation amplitude value of the start Qp of each row of the macro block of the I frame relative to the frame start Qp. Value range: [0, 10].
u32RowQpDeltaP	In the macroblock-level code rate control, the fluctuation amplitude value of the starting Qp of each line of the P frame relative to the starting Qp of the frame. Value range: [0, 10].
bEnableHierQp	Whether QP layering is enabled. RK_TRUE: enable; RK_FALSE: disable.
s32HierQpDelta	The Qp of each layer frame relative to the 0th layer P frame, the value range: [-10, 10]. Default value: 0.
s32HierFrameNum	The number of frames in each layer. Value range: [0, 5].

s32FirstFrameStartQp	The starting Qp value of the first frame is disabled by default, CBR/VBR/AVBR is valid, the value range: [u32MaxIQp, u32MinIQp] and -1; if s32FirstFrameStartQp is -1, the starting QP of the first frame calculated internally by the encoder
stParamH264	H.264 channel rate control mode advanced parameters.
stParamH265	H.265 channel rate control mode advanced parameters.
stParamMjpeg	MJPEG channel rate control mode advanced parameters.

4.4.26 VENC_RC_QUALITY_E

【Description】

Enumerated type of encoding quality.

【Definiton】

```
typedef enum rkVENC_RC_QUALITY_E {
    VENC_RC_QUALITY_HIGHEST,
    VENC_RC_QUALITY_HIGHER,
    VENC_RC_QUALITY_HIGH,
    VENC_RC_QUALITY_MEDIUM,
    VENC_RC_QUALITY_LOW,
    VENC_RC_QUALITY_LOWER,
    VENC_RC_QUALITY_LOWEST,
    VENC_RC_QUALITY_BUTT,
} VENC_RC_QUALITY_E;
```

4.4.27 VENC_ROI_ATTR_S

【Description】

ROI region parameters.

【Definiton】

```
typedef struct rkVENC_ROI_ATTR_S {
    DC_U32 u32Index;
    DC_BOOL bEnable;
    DC_BOOL bAbsQp;
    DC_S32 s32Qp;
```

```
    DC_BOOL bIntra;
    RECT_S stRect;
} VENC_ROI_ATTR_S;
```

【Members】

Member name	Description
u32Index	ROI index value, value range[0, 7]
bEnable	Whether to enable ROI.
bAbsQp	QP mode of ROI, Values range: [0, 1]. 1: absolute QP. 0: relative QP.
s32Qp	QP value, Values range: [-51, 51]. Only the relative mode can make QP value less than 0.
bIntra	The flag of macro block in mandatory frame.
stRect	ROI region.

4.4.28 OSD_REGION_ID_E

【Description】

OSD region ID enumeration type. The overlay priority increases from 0 to 7 gradually, and the OSD with the higher priority is located in the higher layer.

【Definiton】

```
typedef enum rkOSD_REGION_ID_E {
    REGION_ID_0 = 0,
    REGION_ID_1,
    REGION_ID_2,
    REGION_ID_3,
    REGION_ID_4,
    REGION_ID_5,
    REGION_ID_6,
    REGION_ID_7
} OSD_REGION_ID_E;
```

4.4.29 OSD_REGION_INFO_S

【Description】

OSD region information.

【Definiton】

```
typedef struct rkOSD_REGION_INFO_S {
    OSD_REGION_ID_E enRegionId;
    DC_U32 u32PosX;
    DC_U32 u32PosY;
    DC_U32 u32Width;
    DC_U32 u32Height;
    DC_U8 u8Inverse;
    DC_U8 u8Enable;
} OSD_REGION_INFO_S;
```

【Members】

Member name	Description
enRegionId	OSD region index value, value range [0, 7].
u32PosX	X-axis coordinate of OSD region. Must be 16 aligned.
u32PosY	Y-axis coordinate of OSD region. Must be 16 aligned.
u32Width	Width of OSD region. Must be 16 aligned.
u32Height	Height of OSD region. Must be 16 aligned.
u8Inverse	Whether OSD region is inverted.
u8Enable	Whether OSD region is enabled.

【Notice】

Each encoder channel (VENC CHN) supports 8 Regions (index: 0~7). Each Region can be configured as BitMap or Cover, but the two are mutually exclusive. For example, it is reasonable that Region[0] is BitMap and Region[1] is Cover; Region[0] cannot be configured as both BitMap and Cover.

4.4.30 OSD_PIXEL_FORMAT_E

【Description】

OSD pixel format type enumeration.

【Definiton】

```
typedef enum rkOSD_PIXEL_FORMAT_E {
    PIXEL_FORMAT_ARGB_8888 = 0,
    PIXEL_FORMAT_BUTT       // butt of enum
} OSD_PIXEL_FORMAT_E;
```

4.4.31 VENC_COLOR_TBL_S

【Description】

The palette structure.

【Definiton】

```
typedef struct rkVENC_COLOR_TBL {
    DC_U32 u32ArgbTbl[VENC_RGN_COLOR_NUM];
// Enabling dichotomy will speed up the search for the color table,
// but will sort the color table set by the user in ascending order.
    DC_BOOL bColorDichotomyEnable;
} VENC_COLOR_TBL_S;
```

【Members】

Member name	Description
u32ArgbTbl	Color palette, in ARGB8888 format, supported up to VENC_RGN_COLOR_NUM (256)
bColorDichotomyEnable	Turn on dichotomy to optimize query

4.4.32 OSD_COLOR_PALETTE_BUF_S

【Description】

OSD buffer constructed by palette index.

【Definiton】

```
typedef struct rkOSD_COLOR_PALETTE_BUF_S {
    DC_U32 u32Width;      /* buffer's width */
    DC_U32 u32Height;     /* buffer's height */
    DC_VOID *pIdBuf;      /* buffer of the color palette id */
} OSD_COLOR_PALETTE_BUF_S;
```

【Members】

Member name	Description
u32Width	Buffer width.
u32Height	Buffer height.
pIdBuf	Buffer pointer constructed by 8bit palette index.

4.4.33 BITMAP_S

【Description】

Bitmap information and data.

【Definiton】

```
typedef struct rkBITMAP_S {
    OSD_PIXEL_FORMAT_E enPixelFormat;      /* Bitmap's pixel format */
    DC_U32 u32Width;      /* Bitmap's width */
    DC_U32 u32Height;     /* Bitmap's height */
    DC_VOID *pData;       /* Address of Bitmap's data */
} BITMAP_S;
```

【Members】

Member name	Description
enPixelFormat	The format of Bitmap pixel.
u32Width	The width of Bitmap.
u32Height	The height of Bitmap.
pData	The address of Bitmap data.

4.4.34 COVER_INFO_S

【Description】

Privacy cover information.

【Definiton】

```
typedef struct rkCOVER_INFO_S {
    OSD_PIXEL_FORMAT_E enPixelFormat;    /* Bitmap's pixel format */
    DC_U32 u32Color;        /* Covered region color */
} COVER_INFO_S;
```

【Members】

Member name	Description
enPixelFormat	The format of Bitmap pixel.
u32Color	Color of the covered region.

4.4.35 VENC_RECV_PIC_PARAM_S

【Description】

Receive image parameter structure pointer, is used to specify the number of image frames to be received.

【Definiton】

```
typedef struct rkVENC_RECV_PIC_PARAM_S {
    DC_S32 s32RecvPicNum;
} VENC_RECV_PIC_PARAM_S;
```

【Members】

Member name	Description
s32RecvPicNum	The number of image frames to be received.

4.4.36 VENC_JPEG_PARAM_S

【Description】

The advanced parameters of JPEG protocol encoding channel.

【Definiton】

```
typedef struct rkVENC_JPEG_PARAM_S {
    DC_U32 u32Qfactor;      // 1-99
    DC_U8 u8YQt[64];      // reserve
    DC_U8 u8CbQt[64];     // reserve
    DC_U8 u8CrQt[64];      // reserve
    DC_U32 u32MCUPerECS;      // reserve
} VENC_JPEG_PARAM_S;
```

【Members】

Member name	Description
u32Qfactor	Please refer to the RFC2435 protocol for details, Values range: [1, 99].
u8YQt	Reserved parameter, not implemented yet.
u8CbQt	Reserved parameter, not implemented yet.
u8CrQt	Reserved parameter, not implemented yet.
u32MCUPerECS	Reserved parameter, not implemented yet.

4.4.37 VENC_RESOLUTION_PARAM_S

【Description】

VENC resolution configuration structure.

【Definiton】

```
typedef struct rkVENC_RESOLUTION_PARAM_S {
    DC_U32 u32Width;
    DC_U32 u32Height;
    DC_U32 u32VirWidth;
    DC_U32 u32VirHeight;
} VENC_RESOLUTION_PARAM_S;
```

【Members】

Member name	Description
u32Width	Width of buffer.
u32Height	Height of buffer.
u32VirWidth	Width of stride, usually it is the same as buffer_width. If u32VirWidth is greater than the buffer width, it must be16 alignment.
u32VirHeight	Height of stride, usually it is the same as buffer_height. If u32VirHeight is greater than the buffer height, it must be16 alignment.

4.4.38 VENC_CHN_STATUS_S

【Description】

Encoder status structure.

【Definiton】

```
typedef struct rkVENC_CHN_STATUS_S {
    DC_U32 u32LeftFrames;
    DC_U32 u32TotalFrames;
    DC_U32 u32LeftPackets;
    DC_U32 u32TotalPackets;
} VENC_CHN_STATUS_S;
```

【Members】

Member name	Description
u32LeftFrames	Number of frames haven't been handled.
u32TotalFrames	The total number of frames to be handled.
u32LeftPackets	The number of handled but not output packets.
u32TotalPackets	The total number of output packets.

4.4.39 VENC_SUPERFRAME_CFG_S

【Description】

Super frame configuration structure.

【Definiton】

```
typedef struct rkVENC_SUPERFRAME_CFG_S {
    VENC_SUPERFRM_MODE_E enSuperFrmMode;
    DC_U32 u32SuperIFrmBitsThr;
    DC_U32 u32SuperPFrmBitsThr;
    VENC_RC_PRIORITY_E enRcPriority;
} VENC_SUPERFRAME_CFG_S;
```

【Members】

Member name	Description
enSuperFrmMode	Super frame processing mode, the default mode is RKMEDIA_SUPERFRM_NONE RKMEDIA_SUPERFRM_NONE: turn off this function RKMEDIA_SUPERFRM_DISCARD: discard RKMEDIA_SUPERFRM_REENCODE: reprogram once
u32SuperIFrmBitsThr	I frame oversize threshold, the default is 0.
u32SuperPFrmBitsThr	P frame oversize threshold, the default is 0
enRcPriority	Bit rate control priority, the default is RKMEDIA_VENC_RC_PRIORITY_BITRATE_FIRST RKMEDIA_VENC_RC_PRIORITY_BITRATE_FIRST: target bit rate gets high priority RKMEDIA_VENC_RC_PRIORITY_FRAMEBITS_FIRST: super frame value gets high priority

4.5 Error Code

Video encoder API error codes are as follows:

Error code	Macro definition	Description
20	DC_ ERR_ VENC_INVALID_CHNID	Channel ID is out of legal range
21	DC_ ERR_ VENC_ILLEGAL_PARAM	Parameter is out of legal range
22	DC_ ERR_ VENC_EXIST	Attempt to apply for or create an existing device, channel or resource
23	DC_ ERR_ VENC_UNEXIST	Attempt to use or destroy a device, channel or resource that does not exist

24	DC_ ERR_ VENC_NULL_PTR	There is a null pointer in the function parameter
25	DC_ ERR_ VENC_NOT_CONFIG	Not configured before use
26	DC_ ERR_ VENC_NOT_SUPPORT	Unsupported parameter or function
27	DC_ERR_ VENC_NOT_PERM	This operation is not allowed, such as trying to modify static configuration parameters
28	DC_ ERR_ VENC_NOMEM	Virtual memory allocation fails, such as malloc fail
29	DC_ ERR_ VENC_NOBUF	Failed to allocate cache, such as the requested data buffer is too large
30	DC_ ERR_ VENC_BUF_EMPTY	No data in buffer
31	DC_ ERR_ VENC_BUF_FULL	Data in buffer is full
32	DC_ ERR_ VENC_NOTREADY	The system is not initialized or the corresponding module is not loaded
33	DC_ ERR_ VENC_BUSY	VENC system is busy

5. Video Decoding

5.1 Overview

VDEC module, is also called video decoding module. This module supports multi-channel decoding in real time, and each channel is decoded independently, and supports H264/H1265/MJPEG/JPEG decoding.

5.2 API Reference

5.2.1 DC_MPI_VDEC_CreateChn

【Description】

Create a decoding channel.

【Grammar】

DC_S32 DC_MPI_VDEC_CreateChn(VDEC_CHN VdChn, VDEC_CHN_ATTR_S *pstAttr);

【Parameter】

Parameter name	Description	Input/Output
VdChn	Decoding channel number. (0 ~ 16)	Input
pstAttr	Decoding channel attribute pointer	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

5.2.2 DC_MPI_VDEC_DestroyChn

【Description】

Destroy a decoding channel.

【Grammar】

DC_S32 DC_MPI_VDEC_DestroyChn(VDEC_CHN VdChn);

【Parameter】

Parameter name	Description	Input/Output
VdChn	Decoding channel number. (0 ~ 16)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

5.3 Type of Data

5.3.1 VDEC_MAX_CHN_NUM

【Description】

The total number of VDEC physical channels and extended channels.

【Definiton】

```
#defime VDEC_MAX_CHN_NUM 16
```

5.3.2 VDEC_CHN

【Description】

VDEC channel number.

【Definiton】

```
typedef DC_S32 VDEC_CHN
```

5.3.3 VDEC_CHN_ATTR_S

【Description】

Decoding channel attribute structure.

【Definiton】

```
typedef struct rkVDEC_CHN_ATTR_S {
    CODEC_TYPE_E enCodecType;       // RW; video type to be decoded
    IMAGE_TYPE_E enImageType;       // RW; image type to be outputed
    VIDEO_MODE_E enMode;            // RW; send by stream or by frame
    VIDEO_DECODEC_MODE_E enDecodecMode;       // RW; hardware or software
    union {
        VDEC_ATTR_VIDEO_S stVdecVideoAttr;       // RW; structure with video
    };
} VDEC_CHN_ATTR_S;
```

【Members】

Member name	Description
enCodecType	Decoding format.
enImageType	Output format after decoding.
enMode	Decoding input mode, support frame or stream.
enDecodecMode	Decoding mode, supports hardware or software decoding.
stVdecVideoAttr	Decoding video attribute structure. Reserved attribute and not supported currently.

5.3.4 VIDEO_MODE_E

【Description】

Input mode, support frame or stream input.

【Definiton】

```
typedef enum rkVIDEO_MODE_E {
    VIDEO_MODE_STREAM = 0,   // send by stream
    VIDEO_MODE_FRAME,      // send by frame
    VIDEO_MODE_COMPAT,   // Not Support now! One Frame supports multiple packets sending.
    // The current frame is considered to end when bEndOfFrame is equal to RK_TRUE
    VIDEO_MODE_BUTT
} VIDEO_MODE_E;
```

5.3.5 VIDEO_DECODEC_MODE_E

【Description】

Decoding mode.

【Definiton】

```
typedef enum rkVIDEO_DECODEC_MODE_E {
    VIDEO_DECODEC_SOFTWARE = 0,
    VIDEO_DECODEC_HADRWARE,
} VIDEO_DECODEC_MODE_E;
```

6. Motion Detection

6.1 Overview

The motion detection (MD) module will implement motion region detection and support up to 4096 regions.

6.2 Function Description

MD algorithm is implemented by software, and the input resolution should not be too large. The typical resolution is 640x480. The larger the resolution, the higher the CPU load.

6.3 API Reference

6.3.1 DC_MPI_ALGO_MD_CreateChn

【Description】

Create MD channel.

【Grammar】

DC_S32 DC_MPI_ALGO_MD_CreateChn(ALGO_MD_CHN MdChn, ALGO_MD_ATTR_S *pstChnAttr);

【Parameter】

Parameter name	Description	Input/Output
MdChn	Motion detection channel number. (0 ~ 8)	Input
pstChnAttr	Motion detection channel attributes.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

6.3.2 DC_MPI_ALGO_MD_DestroyChn

【Description】

Destroy MD channel.

【Grammar】

DC_S32 DC_MPI_ALGO_MD_DestroyChn(ALGO_MD_CHN MdChn);

【Parameter】

Parameter name	Description	Input/Output
MdChn	Motion detection channel number. (0 ~ 8)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

6.3.3 DC_MPI_ALGO_MD_EnableSwitch

【Description】

Under the condition that MD channel is kept opened, switch MD dynamically.

【Grammar】

DC_S32 DC_MPI_ALGO_MD_EnableSwitch(ALGO_MD_CHN MdChn, DC_BOOL *bEnable);

【Parameter】

Parameter name	Description	Input/Output
MdChn	Motion detection channel number. (0 ~ 8)	Input
bEnable	MD switch.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

6.4 Type of Data

The data types about motion detection are defined as follows:

- ALGO_MD_MAX_CHN_NUM: define the maximum number of motion detection channels.
- ALGO_MD_ROI_RET_MAX: define the maximum number of ROI regions for each channel of motion detection.
- ALGO_MD_CHN: motion detection channel number.
- ALGO_MD_ATTR_S: define the attribute structure of a motion detection channel.

6.4.1 ALGO_MD_MAX_CHN_NUM

【Description】

Define the maximum number of motion detection channels.

【Definiton】

```
#defime ALGO_MD_MAX_CHN_NUM 8
```

6.4.2 ALGO_MD_ROI_RET_MAX

【Description】

Define the maximum number of ROI regions for each channel of motion detection.

【Definiton】

```
#defime ALGO_MD_ROI_RET_MAX 4096
```

6.4.3 ALGO_MD_CHN

【Description】

Motion detection channel number.

【Definiton】

```
typedef DC_S32 ALGO_MD_CHN;
```

6.4.4 ALGO_MD_ATTR_S

【Description】

Define the attribute structure of the motion detection channel.

【Definiton】

```
typedef struct rkALGO_MD_ATTR_S {
    IMAGE_TYPE_E imageType;      //the type of input image
    DC_U32 u32Width;
    DC_U32 u32Height;
    DC_U16 u16RoiCnt;            //RW; Range:[0, ALGO_MD_ROI_RET_MAX].
    RECT_S stRoiRects[ALGO_MD_ROI_RET_MAX];
    DC_U16 u16Sensitivity;       //value 0(sys default) or [1~100].
} ALGO_MD_ATTR_S;
```

【Members】

Member name	Description
imageType	Input image type.
u32Width	The width of a motion detection region.
u32Height	The height of a motion detection region.
u16RoiCnt	Number of ROI regions
stRoiRects	Structure array of ROI region attributes.
u16Sensitivity	Motion detection sensitivity, values range: [1, 100].

6.5 Error Code

Motion detection API error codes are as follows:

Error code	Macro definition	Description
70	DC_ERR_ALGO_MD_INVALID_CHNID	Channel ID is out of legal range

71	DC_ ERR_ALGO_MD_BUSY	Motion detection system is busy
72	DC_ ERR_ALGO_MD_EXIST	Attempt to apply for or create an existing device, channel or resource
73	DC_ ERR_ALGO_MD_NOT_CONFIG	Not configured before use
74	DC_ ERR_ALGO_MD_ILLEGAL_PARAM	Parameter out of legal range

7. Occlusion Detection

7.1 Overview

The Occlusion Detection module implements occlusion alarms and supports up to 10 regions.

7.2 Function Description

The OD algorithm is implemented by software, and the input resolution should not be too large. The typical resolution is 640x480. The larger the resolution, the higher the CPU load.

7.3 API Reference

7.3.1 DC_MPI_ALGO_OD_CreateChn

【Description】

Create OD channel.

【Grammar】

DC_S32 DC_MPI_ALGO_OD_CreateChn(ALGO_OD_CHN OdChn, ALGO_OD_ATTR_S *pstChnAttr);

【Parameter】

Parameter name	Description	Input/Output
OdChn	Occlusion detection channel number. (0 ~ 8)	Input
pstChnAttr	Occlusion detection channel attributes.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

7.3.2 DC_MPI_ALGO_OD_DestroyChn

【Description】

Destroy an OD channel.

【Grammar】

DC_S32 DC_MPI_ALGO_OD_DestroyChn(ALGO_OD_CHN OdChn);

【Parameter】

Parameter name	Description	Input/Output
OdChn	Occlusion detection channel number. (0 ~ 8)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

7.3.3 DC_MPI_ALGO_OD_EnableSwitch

【Description】

Operate OD dynamic switching under the condition of keeping the OD channel open.

【Grammar】

DC_S32 DC_MPI_ALGO_OD_EnableSwitch(ALGO_OD_CHN OdChn, DC_BOOL bEnable);

【Parameter】

Parameter name	Description	Input/Output
OdChn	Occlusion detection channel number. (0 ~ 8)	Input
bEnable	OD switch.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

7.4 Type of Data

The data types about motion detection are defined as follows:

- ALGO_OD_MAX_CHN_NUM: define the maximum number of occlusion detection channels.
- ALGO_OD_ROI_RET_MAX: define the maximum number of ROI regions for each channel of occlusion detection.
- ALGO_OD_CHN: occlusion detection channel number.
- ALGO_OD_ATTR_S: define the attribute structure of a occlusion detection channel.

7.4.1 ALGO_OD_MAX_CHN_NUM

【Description】

Define the maximum number of occlusion detection channels.

【Definiton】

```
#defime ALGO_OD_MAX_CHN_NUM 8
```

7.4.2 ALGO_OD_ROI_RET_MAX

【Description】

Define the maximum number of ROI regions for each channel of occlusion detection.

【Definiton】

```
#defime ALGO_OD_ROI_RET_MAX 10
```

7.4.3 ALGO_OD_CHN

【Description】

Occlusion detection channel number.

【Definiton】

```
typedef DC_S32 ALGO_OD_CHN;
```

7.4.4 ALGO_OD_ATTR_S

【Description】

Define the occlusion detection channel attribute structure.

【Definiton】

```
typedef struct rkALGO_OD_ATTR_S {
    IMAGE_TYPE_E imageType;      //the type of input image
    DC_U32 u32Width;
    DC_U32 u32Height;
    DC_U16 u16RoiCnt;            //RW; Range:[0, ALGO_OD_ROI_RET_MAX].
    RECT_S stRoiRects[ALGO_OD_ROI_RET_MAX];
    DC_U16 u16Sensitivity;       //value 0(sys default) or [1-100].
} ALGO_MD_ATTR_S;
```

【Members】

Member name	Description
imageType	Input image type.
u32Width	The width of a occlusion detection region.
u32Height	The height of a occlusion detection region.
u16RoiCnt	Number of ROI regions
stRoiRects	Structure array of ROI region attributes.
u16Sensitivity	Occlusion detection sensitivity, values range: [1, 100].

7.5 Error Code

Occlusion detection API error codes are as follows:

Error code	Macro definition	Description
80	DC_ERR_ALGO_OD_INVALID_CHNID	Channel ID is out of legal range

81	DC_ ERR_ALGO_OD_BUSY	Occlusion detection system is busy
82	DC_ ERR_ALGO_OD_EXIST	Attempt to apply for or create an existing device, channel or resource
83	DC_ ERR_ALGO_OD_NOT_CONFIG	Not configured before use
84	DC_ ERR_ALGO_OD_ILLEGAL_PARAM	Parameter out of legal range

8. RGA

8.1 Overview

The RGA module is used for 2D image cropping, format conversion, scaling, rotation, image overlay, etc.

8.2 Function Description

The RGA channel in rkmedia only supports format conversion, zooming, cropping, and rotation functions. For image overlay, the librga.so library needs to be called separately.

8.3 API Reference

8.3.1 DC_MPI_RGA_CreateChn

【Description】

Create RGA channels.

【Grammar】

DC_S32 DC_MPI_RGA_CreateChn(RGA_CHN RgaChn, RGA_ATTR_S *pstRgaAttr);

【Parameter】

Parameter name	Description	Input/Output
RgaChn	RGA channel number. (0 ~ 16)	Input
pstRgaAttr	RGA channel attribute pointer.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

8.3.2 DC_MPI_RGA_DestroyChn

【Description】

Destroy an RGA channel.

【Grammar】

DC_S32 DC_MPI_RGA_DestroyChn(RGA_CHN RgaChn);

【Parameter】

Parameter name	Description	Input/Output
RgaChn	RGA channel number. (0 ~ 16)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

8.3.3 DC_MPI_RGA_RGN_SetBitMap

【Description】

Set the BitMap watermark.

【Grammar】

DC_S32 DC_MPI_RGA_RGN_SetBitMap (RGA_CHN RgaChn, const OSD_REGION_INFO_S *pstRgnInfo, const BITMAP_S *pstBitmap);

【Parameter】

Parameter name	Description	Input/Output
RgaChn	RGA channel number. (0 ~ 16)	Input
pstRgnInfo	OSD area information.	Input
pstBitmap	Bitmap information and data.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

8.3.4 DC_MPI_RGA_GetChnRegionLuma

【Description】

Get the brightness of the channel area.

【Grammar】

```
DC_S32 DC_MPI_RGA_GetChnRegionLuma (RGA_CHN RgaChn, const VIDEO_REGION_INFO_S *pstRegionInfo,
                                    DC_U64 *pu64LumaData, DC_S32 s32MIlliSec);
```

【Parameter】

Parameter name	Description	Input/Output
RgaChn	RGA channel number. (0 ~ 16)	Input
pstRegionInfo	Region information. pstRegionInfo->pstRegion is the area attribute of the statistical area, that is, the starting position, width, and height; pstRegionInfo->u32RegionNum is the number of the statistical area.	Input
pu64LumaDat	Memory pointer for receiving area brightness and statistical information. The memory size should be greater than or equal to sizeof(RK_U64)×pstRegionInfo->u32RegionNum.	Output
s32MilliSec	Timeout parameter s32MilliSec: less than or equal to 0 means blocking mode; greater than 0 means timeout mode, and the unit of timeout is milliseconds (ms).	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

8.3.5 DC_MPI_RGA_RGN_SetCover

【Description】

Set privacy cover.

【Grammar】

DC_S32 DC_MPI_RGA_GetChnRegionLuma (RGA_CHN RgaChn, const OSD_REGION_INFO_S *pstRgnInfo,
Const COVER_INFO_S *pstCoverInfo);

【Parameter】

Parameter name	Description	Input/Output
RgaChn	RGA channel number. (0 ~ 16)	Input
pstRgnInfo	RGN area information.	Input
pstCoverInfo	Privacy cover information.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

8.4 Type of Data

Data types related to RGA are defined as follows:

- RGA_MAX_CHN_NUM: the maximum number of RGA channels.
- RGA_CHN: RGA channel number.
- RGA_INFO_S: RGA region attribute structure.
- RGA_ATTR_S: RGA attribute structure.

8.4.1 RGA_MAX_CHN_NUM

【Description】

The maximum number of RGA channels.

【Definiton】

```
#defime RGA_MAX_CHN_NUM 16
```

8.4.2 RGA_CHN

【Description】

RGA channel number.

【Definiton】

```
typedef DC_S32 RGA_CHN;
```

8.4.3 RGA_INFO_S

【Description】

RGA region attribute structure.

【Definiton】

```
typedef struct rkRGA_INFO_S {
    IMAGE_TYPE_E imgType;
    DC_U32 u32X;
    DC_U32 u32Y;
    DC_U32 u32Width;
    DC_U32 u32Height;
    DC_U32 u32HorStride;    // horizontal stride
    DC_U32 u32VirStride;    // virtual stride
} RGA_INFO_S;
```

【Members】

Member name	Description
imgType	Image format type.
u32X	X-axis coordinate of RGA.
u32Y	Y-axis coordinate of RGA.
u32Width	The width of RGA.
u32Height	The height of RGA.
u32HorStride	Virtual width.
u32VirStride	Virtual height.

8.4.4 RGA_ATTR_S

【Description】

RGA attribute structure.

【Definiton】

```
typedef struct rkRGA_ATTR_S {
    RGA_INFO_S stImgIn;      // input image info
    RGA_INFO_S stImgOut;     // output image info
    DC_U16 u16Rotaion;       // support 0/90/180/270.
    DC_BOOL bEnBufPool;
    DC_U16 u16BufPoolCnt;
    RGA_FLIP_E enFlip;
} RGA_ATTR_S;
```

【Members】

Member name	Description
stImgIn	Input image information.
stImgOut	Output image information.
u16Rotaion	Rotation angle. Values: 0, 90, 180, 270.
bEnBufPool	Enable buffer pool.
u16BufPoolCnt	Buffer pool count.
enFlip	Mirror control. Support horizontal mirroring, vertical mirroring, horizontal and vertical mirroring.

8.5 Error Code

RGA API error codes are as follows:

Error code	Macro definition	Description
90	DC_ERR_RGA_INVALID_CHNID	RGA input device number is invalid
91	DC_ ERR_RGA_BUSY	RGA system is busy
92	DC_ERR_RGA_EXIST	Attempting to apply for or create an existing device, channel or resource
93	DC_ ERR_RGA_NOT_CONFIG	Not configured before use

94	DC_ ERR_RGA_ILLEGAL_PARAM	Illegal parameter
95	DC_ERR_RGA_NOTREADY	Device is not ready

9. Video Synthesis

9.1 Overview

The video synthesis VMIX module uses RGA to synthesize and splice multi-channel videos, and can bind the spliced video to VO display to realize multi-channel video synthesis display.

9.2 Function Description

The video synthesis VMIX module supports functions such as video synthesis, area drawing frame, sensitive area setting, channel display, channel hiding, and channel area brightness acquisition.

9.3 API Reference

9.3.1 DC_MPI_VMIX_CreateDev

【Description】

Create VMIX device.

【Grammar】

DC_S32 DC_MPI_VMIX_CreateDev(VMIX_DEV VmDev, VMIX_DEV_INFO_S *pstDevInfo);

【Parameter】

Parameter name	Description	Input/Output
VmDev	VMIX device number. (0 ~ 16)	Input
pstDevInfo	VMIX device attribute pointer.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

9.3.2 DC_MPI_VMIX_DestroyDev

【Description】

Destroy VMIX device.

【Grammar】

DC_S32 DC_MPI_VMIX_DestroyDev (VMIX_DEV VmDev);

【Parameter】

Parameter name	Description	Input/Output
VmDev	VMIX device number. (0 ~ 16)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

9.3.3 DC_MPI_VMIX_EnableChn

【Description】

Enable channels of VMIX devices.

【Grammar】

DC_S32 DC_MPI_VMIX_EnableChn(VMIX_DEV VmDev, VMIX_CHN VmChn);

【Parameter】

Parameter name	Description	Input/Output
VmDev	VMIX device number. (0 ~ 16)	Input
VmChn	VMIX device channel number. (0 ~ 16)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

9.3.4 DC_MPI_VMIX_DisableChn

【Description】

Disable channels of VMIX devices.

【Grammar】

DC_S32 DC_MPI_VMIX_DisableChn (VMIX_DEV VmDev, VMIX_CHN VmChn);

【Parameter】

Parameter name	Description	Input/Output
VmDev	VMIX device number. (0 ~ 16)	Input
VmChn	VMIX device channel number. (0 ~ 16)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

9.3.5 DC_MPI_VMIX_SetLineInfo

【Description】

Set VMIX frame drawing information.

【Grammar】

DC_S32 DC_MPI_VMIX_SetLineInfo(VMIX_DEV VmDev, VMIX_CHN VmChn,

VMIX_LINE_INFO_S VmLine);

【Parameter】

Parameter name	Description	Input/Output
VmDev	VMIX device number. (0 ~ 16)	Input
VmChn	VMIX device channel number. (0 ~ 16)	Input
VmLine	VMIX frame drawing information.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

9.3.6 DC_MPI_VMIX_ShowChn

【Description】

Display the channel of VMIX device.

【Grammar】

DC_S32 DC_MPI_VMIX_SetLineInfo(VMIX_DEV VmDev, VMIX_CHN VmChn);

【Parameter】

Parameter name	Description	Input/Output
VmDev	VMIX device number. (0 ~ 16)	Input
VmChn	VMIX device channel number. (0 ~ 16)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

9.3.7 DC_MPI_VMIX_HideChn

【Description】

Hide the channel of the VMIX device.

【Grammar】

DC_S32 DC_MPI_VMIX_HideChn(VMIX_DEV VmDev, VMIX_CHN VmChn);

【Parameter】

Parameter name	Description	Input/Output
VmDev	VMIX device number. (0 ~ 16)	Input
VmChn	VMIX device channel number. (0 ~ 16)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

9.3.8 DC_MPI_VMIX_RGN_SetBitMap

【Description】

Set the BitMap watermark.

【Grammar】

```
DC_S32 DC_MPI_VMIX_RGN_SetBitMap (VMIX_DEV VmDev, VMIX_CHN VmChn,
                                  const OSD_REGION_INFO_S *pstRgnInfo, const BITMAP_S *pstBitmap);
```

【Parameter】

Parameter name	Description	Input/Output
VmDev	VMIX device number. (0 ~ 16)	Input
VmChn	VMIX device channel number. (0 ~ 16)	Input
pstRgnInfo	OSD area information.	Input
pstBitmap	Bitmap information and data.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

9.3.9 DC_MPI_VMIX_GetRegionLuma

【Description】

Get region exposure of a VMIX device.

【Grammar】

DC_S32 DC_MPI_VMIX_GetRegionLuma (VMIX_DEV VmDev, const VIDEO_REGION_INFO_S *pstRegionInfo, DC_U64 *pu64LumaData, DC_S32 s32MilliSec);

【Parameter】

Parameter name	Description	Input/Output
VmDev	VMIX device number. (0 ~ 16)	Input
pstRegionInfo	Region information: pstRegionInfo->pstRegion is the region attribute of the statistical region , that is, the starting position, width, and height; pstRegionInfo->u32RegionNum is the number of the statistical region.	Input
pu64LumaData	Memory pointer for receiving area brightness and statistical information. The memory size should be greater than or equal to sizeof(RK_U64)×pstRegionInfo->u32RegionNum.	Output

s32MilliSec	Timeout parameter s32MilliSec: less than or equal to 0 means blocking mode; greater than 0 means timeout mode, and the unit of timeout is milliseconds (ms).	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

9.3.10 DC_MPI_VMIX_GetChnRegionLuma

【Description】

Get the channel region exposure of a VMIX device.

【Grammar】

```
DC_S32 DC_MPI_VMIX_GetChnRegionLuma(VMIX_DEV VmDev, VMIX_CHN VmChn,
                const VIDEO_REGION_INFO_S *pstRegionInfo, DC_U64 *pu64LumaData, DC_S32 s32MilliSec);
```

【Parameter】

Parameter name	Description	Input/Output
VmDev	VMIX device number. (0 ~ 16)	Input
VmChn	VMIX device channel number. (0 ~ 16)	Input
pstRegionInfo	Region information: pstRegionInfo->pstRegion is the region attribute of the statistical region , that is, the starting position, width, and height; pstRegionInfo->u32RegionNum is the number of the statistical region.	Input

pu64LumaData	Memory pointer for receiving area brightness and statistical information. The memory size should be greater than or equal to sizeof(RK_U64)×pstRegionInfo->u32RegionNum.	Output
s32MilliSec	Timeout parameter s32MilliSec: less than or equal to 0 means blocking mode; greater than 0 means timeout mode, and the unit of timeout is milliseconds (ms).	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

9.3.11 DC_MPI_VMIX_RGN_SetCover

【Description】

Privacy cover.

【Grammar】

```
DC_S32 DC_MPI_VMIX_RGN_SetCover (VMIX_DEV VmDev, VMIX_CHN VmChn,
                                 const OSD_REGION_INFO_S *pstRgnInfo, const COVER_INFO_S *pstCoverInfo);
```

【Parameter】

Parameter name	Description	Input/Output
VmDev	VMIX device number. (0 ~ 16)	Input
VmChn	VMIX device channel number. (0 ~ 16)	Input

pstRgnInfo	RGN area information.	Input
pstCove	Privacy covering information.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

9.4 Type of Data

VMIX related data types are defined as follows:

- VMIX_DEV: VMIX device number.
- VMIX_CHN: The channel number of the VMIX device.
- VMIX_DEV_INFO_S: VMIX device information structure.
- VMIX_CHN_INFO_S: VMIX device channel information structure.
- VMIX_LINE_INFO_S: VMIX frame drawing structure.
- VMIX_MAX_CHN_NUM: Maximum channel value of VMIX device.
- VMIX_MAX_DEV_NUM: Maximum value of VMIX device.
- VMIX_MAX_LINE_NUM: The maximum value of the channel frame of the VMIX device.

9.4.1 VMIX_DEV

【Description】

VMIX device number.

【Definiton】

```
typedef DC_S32 VMIX_DEV;
```

9.4.2 VMIX_CHN

【Description】

The channel number of the VMIX device.

【Definiton】

```
typedef DC_S32 VMIX_CHN;
```

9.4.3 VMIX_DEV_INFO_S

【Description】

VMIX device information structure.

【Definiton】

```
typedef struct rkVMIX_DEV_INFO_S {
    DC_U16 u16ChnCnt;
    DC_U16 u16Fps;
    DC_U32 u32ImgWidth;
    DC_U32 u32ImgHeight;
    IMAGE_TYPE_E enImgType;
    VMIX_CHN_INFO_S stChnInfo[VMIX_MAX_CHN_NUM];
} VMIX_DEV_INFO_S;
```

【Members】

Member name	Description
u16ChnCnt	Number of channels.
u16Fps	Frame rate.
u32ImgWidth	The width of synthetic image.
u32ImgHeight	The height of synthetic image.
enImgType	Image format type.
stChnInfo	Channel information.

9.4.4 VMIX_CHN_INFO_S

【Description】

VMIX device channel information structure.

【Definiton】

```
typedef struct rkVMIX_CHN_INFO_S {
    IMAGE_TYPE_E enImgInType;
    IMAGE_TYPE_E enImgOutType;
    RECT_S stInRect;
    RECT_S stOutRect;
} VMIX_CHN_INFO_S;
```

【Members】

Member name	Description
enImgInType	Format type of input image.
enImgOutType	Format type of output image.
stInRect	Input image region.
stOutRect	Output image region.

9.4.5 VMIX_LINE_INFO_S

【Description】

VMIX frames drawing structure.

【Definiton】

```
typedef struct rkVMIX_LINE_INFO_S {
    DC_U32 u32LineCnt;
    DC_U32 u32Color;
    RECT_S stLines[VMIX_MAX_LINE_NUM];
} VMIX_LINE_INFO_S;
```

【Members】

Member name	Description
u32LineCnt	Number of frames.
u32Color	Frame color.
stLines	Frame region.

9.4.6 VMIX_MAX_CHN_NUM

【Description】

Maximum channel value of VMIX device.

【Definiton】

```
#define VMIX_MAX_CHN_NUM 16
```

9.4.7 VMIX_MAX_DEV_NUM

【Description】

Maximum value of VMIX device.

【Definiton】

```
#define VMIX_MAX_DEV_NUM 16
```

9.4.8 VMIX_MAX_LINE_NUM

【Description】

The maximum value of the channel frame of the VMIX device.

【Definiton】

```
#define VMIX_MAX_LINE_NUM 64
```

9.5 Error Code

VMIX API error codes are as follows:

Error code	Macro definition	Description
130	DC_ERR_VMIX_INVALID_DEVID	VMIX input device number is invalid
131	DC_ERR_VMIX_INVALID_CHNID	VMIX input device channel number is invalid
132	DC_ERR_VMIX_BUSY	VMIX system is busy
133	DC_ERR_VMIX_EXIST	Attempting to apply for or create an existing device, channel or resource
134	DC_ERR_VMIX_ILLEGAL_PARAM	Illegal parameter
135	DC_ERR_VMIX_NOTREADY	VMIX device is not ready
136	DC_ERR_VMIX_NOTOPEN	The channel of the VMIX device is not opened

10. One in Four out Video

10.1 Overview

Video with one Input Four Output (VP) uses ISPP module to realize a video input from the rkispp_input_image node as video input, and four videos from the rkispp_m_bypass, rkispp_scale0, rkispp_scale1, rkispp_scale2 nodes as the video output. The node information can be found by the following command:

```
media-ctl -p -d /dev/media*
```

Use VP to complete the video from the rkispp_input_image node as the video input, and use the VI to complete the video from the rkispp_m_bypass, rkispp_scale0, rkispp_scale1, and rkispp_scale2 nodes as the video output, which can achieve one input four output.

rkispp_m_bypass, rkispp_scale0, rkispp_scale1, rkispp_scale2 usage requirements reference: 3.2.2 VI video node. rkispp_scale0, rkispp_scale1, rkispp_scale2 can achieve scaling, replace the scaling function of RGA, alleviate RGA hardware stress for related products such as DVR.

To use VP function, you need to configure the corresponding media node first, for example:

```
media-ctl -d /dev/media5 -l'"rkispp_input_image":0->"rkispp-subdev":0[1]'
media-ctl -d /dev/media5 --set-v4l2'"rkispp-subdev":0[fmt:YUYV8_2X8/1920x1080]'
media-ctl -d /dev/media5 --set-v4l2'"rkispp-subdev":2[fmt:YUYV8_2X8/1920x1080]'
```

After the configuration, you need to open 4 output nodes (1 to 4 nodes can be opened), and finally open the input nodes. The data of the input node can come from other VIs, RGAs, etc.

10.2 Function Description

To realize the video one in four out.

10.3 API Reference

10.3.1 DC_MPI_VP_EnableChn

【Description】

Enable the VP channel.

【Grammar】

DC_S32 DC_MPI_VP_EnableChn(VP_PIPE VpPipe, VP_CHN VpChn);

【Parameter】

Parameter name	Description	Input/Output
VpPipe	VP pipe number.	Input
VpChn	VP channel number. (0~8)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

10.3.2 DC_MPI_VP_DisableChn

【Description】

Close the VP channel.

【Grammar】

DC_S32 DC_MPI_VP_DisableChn (VP_PIPE VpPipe, VP_CHN VpChn);

【Parameter】

Parameter name	Description	Input/Output
VpPipe	VP pipe number.	Input
VpChn	VP channel number. (0~8)	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

10.3.3 DC_MPI_VP_SetChnAttr

【Description】

Close the VP channel.

【Grammar】

DC_S32 DC_MPI_VP_SetChnAttr (VP_PIPE VpPipe, VP_CHN VpChn, const VP_CHN_ATTR_S *pstChnAttr);

【Parameter】

Parameter name	Description	Input/Output
VpPipe	VP pipe number.	Input
VpChn	VP channel number. (0~8)	Input
pstChnAttr	VP Channel attribute structure pointer.	Input

【Return value】

Return value	Description
0	Success
Not 0	Failure, see Error Code for details

【Requirement】

Header file：dcmadia_api.h
Library file：libdcmadia.so

【Notice】

No.

【Example】

No.

【Related topic】

No.

10.4 Type of Data

The video input related data types are defined as follows:

- VP_MAX_CHN_NUM: Define the total number of VP physical channels.
- VP_PIPE: VP pipe number.
- VP_CHN: VP channel number.
- VP_CHN_ATTR_S: VP channel attribute structure pointer.

10.4.1 VP_MAX_CHN_NUM

【Description】

Define the total number of VP physical channels. For RV1126/RV1109 ISPP, it is up to 8 can be configured.

【Definiton】

```
#define VP_MAX_CHN_NUM 8
```

10.4.2 VP_PIPE

【Description】

The VP pipe number is not used yet.

【Definiton】

```
typedef DC_S32 VP_PIPE;
```

10.4.3 VP_CHN

【Description】

VP channel number.

【Definiton】

```
typedef DC_S32 VP_CHN;
```

10.4.3 VP_CHN_ATTR_S

【Description】

VP channel number.

【Definiton】

```
typedef enum rkVP_CHN_WORK_MODE {
    VP_WORK_MODE_NORMAL = 0,
    VP_WORK_MODE_BUTT
} VP_CHN_WORK_MODE;
```

```
typedef enum rkVP_CHN_BUF_TYPE {
    VP_CHN_BUF_TYPE_DMA = 0,        // Default
    VP_CHN_BUF_TYPE_MMAP,
} VP_CHN_BUF_TYPE;

typedef struct rkVP_CHN_ATTR_S {
    const RK_CHAR *pcVideoNode;
    RK_U32 u32Width;
    RK_U32 u32Height;
    IMAGE_TYPE_E enPixFmt;
    RK_U32 u32BufCnt;          // VP output video buffer cnt.
    VP_CHN_BUF_TYPE enBufType;          // VP output video buffer type.
    VP_CHN_WORK_MODE enWorkMode;
} VP_CHN_ATTR_S;
```

【Members】

Member name	Description
pcVideoNode	Video node path.
u32Width	Video width.
u32Height	Video height.
enPixFmt	Video format.
u32BufCnt	VP capture video buffer count.
enBufType	VP video buffer type.
enWorkMode	VP channel working mode.

【Description】

VI_WORK_MODE_LUMA_ONLY mode is used for VI brightness statistics. In this mode, VI has no output and cannot obtain data from VI.

10.5 Error Code

VP API error codes are as follows:

Error code	Macro definition	Description
150	DC_ERR_VP_INVALID_CHNID	VP input channel number is invalid.
151	DC_ERR_VP_BUSY	Device is occupied.

152	DC_ERR_VP_EXIST	VP channel has been opened.
153	DC_ERR_VP_NOT_CONFIG	The parameter is not configured.
154	DC_ERR_VP_TIMEOUT	Timeout.
155	DC_ERR_VP_BUF_EMPTY	Data is empty.
156	DC_ERR_VP_ILLEGAL_PARAM	Illegal parameter.
157	DC_ERR_VP_NOTREADY	VP channel has not been opened yet.

11. Notices

11.1 Channel Destruction Order

It is important to note that rkmedia has special requirements for destruction order of modules: the subsequent modules in the data flow pipeline must be destroyed before the previous modules. such as: VI --> RGA --> VENC The recommended order of destruction is as follows: destroy VENC destroy RGA destroy VI.

Take VI as an example. VI is a data generator, The generated buffer may be occupied by the subsequent stage when the data pipeline is destroyed, resulting in the resources managed by VI being occupied. You will encounter device busy error when you open it again. This problem may occur when frequently creating and destroying data channels.

11.2 Parameter Initialization

It is recommended to use memset to initialize the parameter to 0 to avoid the influences of random initialization of parameters.

12. Proc Debugging Information Description

12.1 VI

When VI has no data output, check the following node information to check where is abnormal.

【Debugging information】

```
# cif command
cat /proc/rkcif_mipi_lvds | grep "frame amount"; sleep 3; cat
/proc/rkcif_mipi_lvds | grep "frame amount"
```

```
# ouput
frame amount:1836735
frame amount:1836826

# isp command
cat /proc/rkisp* | grep Output; sleep 3; cat /proc/rkisp*| grep Output;

# output
Output rkispp0 ON Format:FBC420 Size:2688x1520 (frame:1837606 rate:32ms)
Output rkispp_m_bypass Format:NV12 Size:2688x1520 (frame:1837606 rate:31ms delay:29ms)
Output rkispp_scale0 Format:NV12 Size:1920x1080 (frame:1837606 rate:31ms delay:29ms)
Output rkispp_scale1 Format:NV12 Size:704x576 (frame:1837606 rate:31ms delay:29ms)
Output rkispp_scale2 Format:NV12 Size:1280x720 (frame:1837606 rate:31ms delay:29ms)
Output rkispp0 ON Format:FBC420 Size:2688x1520 (frame:1837698 rate:33ms)
Output rkispp_m_bypass Format:NV12 Size:2688x1520 (frame:1837697 rate:32ms delay:29ms)
Output rkispp_scale0 Format:NV12 Size:1920x1080 (frame:1837697 rate:32ms delay:29ms)
Output rkispp_scale1 Format:NV12 Size:704x576 (frame:1837697 rate:32ms delay:29ms)
Output rkispp_scale2 Format:NV12 Size:1280x720 (frame:1837697 rate:32ms delay:29ms)
```

【Analysis of the debugging information】

To get the difference between before and after frames, if frames increases normally, in means that the channel can transmit data normally. If the frame does not change, the channel may be abnormal and the data may be blocked.

【Parameter Description】

Member name	Description
frame amount/frame	The number of output frame.
rate	The rate of output frame.
Format	The format of output.
Size	The size of output frame.

Note: parameters which are not mentioned here, are not used in the debugging of DCMedia.

12.2 VENC

【Debugging information】

```
# command
cat /proc/mpp_service/session_summary

# output
------------------------------------------------------ ------------------
----- ------------------------- --------
| session| device| width| height| format| fps_in| fps_out| rc_mode|
bitrate|gop_size|fps_calc| profile|
|8cdb338a| RKVENC| 2688| 1520| avc| 25| 16| vbr| 7549747| 50| 19.49| high|
------------------------------------------------------ ------------------
------------------------------ --------
| session| device|
|0a1be0b6| VEPU2|
------------------------------------------------------ ------------------
----- ------------------------- --------
| session| device| width| height| format| fps_in| fps_out| rc_mode|
bitrate|gop_size|fps_calc| profile|
|6e6fd71b| RKVENC| 704| 576| avc| 25| 25| cbr| 943718| 50| 30.60| high|
------------------------------------------------------ ------------------
------------------------------ --------
| session| device| width| height| format| fps_in| fps_out| rc_mode|
bitrate|gop_size|fps_calc| profile|
|a87d7eac| RKVENC| 1920| 1080| hevc| 25| 25| cbr| 1887436| 50| 30.51| main|
```

【Analysis of the debugging information】

Check VENC parameters of each channel to make sure whether the parameters of channel creation/modification are normal.

【Parameter Description】

Member name	Description
width	The width of resolution
height	The high of resolution
format	Format
fps_in	The rate of input frame

fps_out	The rate of output frame
rc_mode	Rate control mode
bitrate	Bit rate
gop_size	I frame interval
fps_calc	Actual calculated frame rate
profile	Encoder profile

Note: parameters which are not mentioned here, are not used in the debugging of DCMedia.

13. LOG Debugging Level Description

It is supported to modify of the current debugging level of each module dynamically. For the echo command which is used to modify the debug level of a module, please refer to the following example:

```
echo "venc=3"> /tmp/loglevel
```

Modify the debugging level of all modules:

```
echo "all=3"> /tmp/loglevel
```

Modules listed in MOD_ID_E are all supported, with lowercase of module names. The numbers 0~3 correspond to the four levels of ERROR, WARN, INFO, and DEBUG respectively.