HiMPP Media Processing Software

Development Reference

Issue **25**

Date **2015-04-16**

Copyright © HiSilicon Technologies Co., Ltd. 2011-2015. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of HiSilicon Technologies Co., Ltd.

Trademarks and Permissions

, HISILICON, and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

HiSilicon Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.hisilicon.com

Email: support@hisilicon.com

Contents

About This Document ii

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

About This Document

Purpose

This document provides reference information for the programmers that develop products or solutions by using the HiSilicon media processing platform (HiMPP). The information includes application programming interfaces (APIs), header files, and error codes.

This document also describes the method of calling each API, and the definitions of data types and error codes.

Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3531	V100
Hi3532	V100
Hi3521	V100
Hi3520A	V100
Hi3518	V100
Hi3516C	V100
Hi3520D	V100
Hi3515A	V100
Hi3515C	V100

Intended Audience

This document is intended for:

- Technical support personnel
- Software development engineers

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Conventions

Symbol Conventions

The symbols that may be found in this document are defined as follows:

Symbol	Description
DANGER	Indicates a hazard with a high level of risk which, if not avoided, will result in death or serious injury.
WARNING	Indicates a hazard with a medium or low level of risk that, if not avoided, could result in minor or moderate injury.
CAUTION	Indicates a potentially hazardous situation, which if not avoided, could result in equipment damage, data loss, performance degradation, or unexpected results.
TIP	Indicates a tip that may help you solve a problem or save time.
NOTE	Provides additional information to emphasize or supplement important points of the main text.

General Conventions

The general conventions that may be found in this document are defined as follows:

Convention	Description
Times New Roman	Normal paragraphs are in Times New Roman.
Boldface	Names of files, directories, folders, and users are in **boldface**. For example, log in as user **root**.
Italic	Book titles are in *italics*.
Courier New	Examples of information displayed on the screen are in Courier New.

Command Conventions

The command conventions that may be found in this document are defined as follows:

Convention	Description
Boldface	The keywords of a command line are in **boldface**.
Italic	Command arguments are in *italics*.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Convention	Description
[ ]	Items (keywords or arguments) in square brackets ([ ]) are optional.
{ x | y | ... }	Optional items are grouped in braces and separated by vertical bars. One item is selected.
[x | y | ...]	Optional items are grouped in brackets and separated by vertical bars. One item is selected or no item is selected.
{ x | y | ... } *	Optional items are grouped in braces and separated by vertical bars. A minimum of one item or a maximum of all items can be selected.
[x | y | ...] *	Optional items are grouped in brackets and separated by vertical bars. Several items or no item can be selected.

Numerical System

The expressions of data capacity, frequency, and data rate are described as follows:

Type	Symbol	Value
Data capacity (such as the RAM capacity)	K	1024
	M	1,048,576
	G	1,073,741,824
Frequency, data rate	k	1000
	M	1,000,000
	G	1,000,000,000

The expressions of addresses and data are described as follows:

Symbol	Example	Description
0x	0xFE04, 0x18	Address or data in hexadecimal
0b	0b000, 0b00 00000000	Data or sequence in binary (register description is excluded.)
X	00X, 1XX	In data expression, X indicates 0 or 1. For example, 00X indicates 000 or 001 and 1XX indicates 100, 101, 110, or 111.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Change History

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made in previous issues.

Issue 25 (2015-04-16)

This issue is the twenty-fifth official release, which incorporates the following changes:

Chapter 4 VO

In section 4.3, HI_MPI_VO_SetChnDisplayWindow and HI_MPI_VO_GetChnDisplayWindow are added.

Chapter 6 VENC

Section 6.2.13 is added.

In section 6.4, the descriptions in the **Member** fields of VENC_ATTR_H264_S, VENC_ATTR_MJPEG_S, VENC_ATTR_JPEG_S, and VENC_ATTR_MPEG4_S are updated.

Chapter 13 Proc Debugging Information

The contents in sections 13.4, 13.7, 13.8, 13.10, 13.22, and 13.23 are modified.

Issue 24 (2015-03-30)

This issue is the twenty-fourth official release, which incorporates the following changes:

Chapter 8 Region Management

The description in the **Member** field of COVER_CHN_ATTR_S is updated.

Chapter 9 Audio

The description in section 9.2.1.5 is updated.

In section 9.3.1, HI_MPI_AI_SetHpfAttr and HI_MPI_AI_GetHpfAttr are added.

In section 9.3.5.1, ACODEC_SET_INPUT_VOL to ACODEC_GET_OUTPUT_VOL are added.

In section 9.4.1, AUDIO_HPF_FREQ_E, AI_HPF_CONFIG_S and AI_HPF_ATTR_S are added.

Chapter10 VDEC

The description in section 10.2 is updated.

In section 10.3, the description in the **Note** field of HI_MPI_VDEC_SetPrtclParam is updated.

In section 10.4, the descriptions in the **Member** fields of VDEC_CHN_ATTR_S and VDEC_PRTCL_PARAM_S are updated.

Issue 23 (2015-02-10)

This issue is the twenty-third official release, which incorporates the following changes:

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chapter 3 VI

In section 3.4, HI_MPI_VI_SetChnScanMode and HI_MPI_VI_GetChnScanMode are added.

Chapter 5 VPSS

In section 5.3, HI_MPI_VPSS_EnableBackupFrame and HI_MPI_VPSS_DisableBackupFrame are added. The description in the **Note** field of HI_MPI_VPSS_SetCropCfg is updated.

Chapter 6 VENC

Sections 6.2.11 and 6.2.12 are added.

In section 6.3, HI_MPI_VENC_AttachVbPool, HI_MPI_VENC_DetachVbPool, HI_MPI_VENC_SetH264VideoSignal, and HI_MPI_VENC_GetH264VideoSignal are added.

In section 6.4, VENC_PARAM_H264_VIDEO_SIGNAL_S is added.

Chapter 9 Audio

In section 9.3.5.1, the description of ACODEC_ENABLE_BOOSTL is updated. ACODEC_ENABLE_BOOSTR is added.

Chapter 12 HDMI

In section 12.4, HI_HDMI_SPD_INFOFRAME_S, HI_HDMI_MPEGSOURCE_INFOFRAME_S, and HI_HDMI_VENDORSPEC_INFOFRAME_S are added.

Chapter 13 Proc Debugging Information

The descriptions in the **Debugging Information** and **Parameter Description** fields of sections 13.8, 13.9, 13.12, and 13.23 are updated.

Issue 22 (2014-09-21)

This issue is the twenty-second official release, which incorporates the following changes:

Chapter 4 VO

In section 4.4, the **Chip Difference** fields are added to VO_PUB_ATTR_S and VO_PUB_ATTR_S, and the description in the **Difference** field of VO_VIDEO_LAYER_ATTR_S is updated.

Chapter 5 VPSS

In section 5.3, the description in the **Note** field of HI_MPI_VPSS_SetGrpParam is updated.

In section 5.4, the description in the **Member** field of VPSS_GRP_ATTR_S is updated.

Chapter 9 Audio

Section 9.2.2.3 is updated.

In section 9.4.3, AUDIO_FRAME_INFO_S is added.

Chapter 12 HDMI

In section 12.4, the description of HI_HDMI_VIDEO_FMT_E is updated.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Issue 21 (2014-09-02)

This issue is the twenty-first official release, which incorporates the following changes:

Chapter 3 VI

In section 3.4, the HI_MPI_VI_SetPortMode, HI_MPI_VI_GetPortMode, HI_MPI_VI_EnableChn422to420, and HI_MPI_VI_DisableChn422to420 are added.

In section 3.5, VI_PORTMODE_E is added.

Chapter 9 Audio

In section 9.3.4, the HI_MPI_ADEC_GetFrame and HI_MPI_ADEC_ReleaseFrame are added.

Issue 20 (2014-08-13)

This issue is the twentieth official release, which incorporates the following changes:

Chapter 2 System Control

In section 2.3, HI_MPI_VB_GetVRegAddr is added.

Chapter 3 VI

In section 3.4, the VIU parameters are added to HI_MPI_VI_SetDevAttr, and the parameters of the CSC attribute are added to the **Note** field of HI_MPI_VI_SetCSCAttr.

Chapter 5 Video Processing Subsystem

In section 5.3, the description in the **Note** fields of HI_MPI_VPSS_CreateGrp and HI_MPI_VPSS_SetChnMode is updated.

Chapter 6 VENC

In Table 6-1, the description about SVC-T is added.

In section 6.3, the MPIs HI_MPI_VENC_SetRcPriority, HI_MPI_VENC_GetRcPriority, HI_MPI_VENC_SetLostFrameStrategy, HI_MPI_VENC_GetH264InterPred, and HI_MPI_VENC_GetLostFrameStrategy are added.

The description in the **Difference** field of HI_MPI_VENC_CreateChn is updated, and the description in the **Note** fields of HI_MPI_VENC_SetChnAttr and HI_MPI_VENC_SetRcPara is updated.

In section 6.4, VENC_RC_PRIORITY_E, VENC_PARAM_LOSTFRM_S, H264E_REF_TYPE_E, VENC_ATTR_H264_ABR_S, and VENC_LOSTFRM_MODE_E are added.

Chapter 9 Audio

In section 9.2.1.4, Figure 9-4 is modified. Section 9.2.1.5 is added.

In section 9.2.4.2, the description about BOOST is added.

In section 9.3.1, the MPIs HI_MPI_AI_SetVqeAttr, HI_MPI_AI_GetVqeAttr, HI_MPI_AI_EnableVqe, HI_MPI_AI_DisableVqe, HI_MPI_AI_EnableReSmpEx, HI_MPI_AI_DisableReSmpEx, HI_MPI_AI_ClrPubAttr, and HI_MPI_AI_Save_File are added. The descriptions in the **Note** fields of HI_MPI_AI_GetFrame, HI_MPI_AI_ReleaseFrame, HI_MPI_AI_SetChnParam, and HI_MPI_AI_GetChnParam are updated.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

In section 9.3.2, the MPIs HI_MPI_AO_EnableReSmpEx, HI_MPI_AO_DisableReSmpEx, and HI_MPI_AO_ ClrPubAttr are added.

The description in the **Note** field of HI_MPI_AO_SendFrame is updated.

In section 9.3.3, the HI_MPI_AENC_Save_File MPI is added, and the description in the **Note** field of HI_MPI_AENC_SendFrame is updated.

In section 9.3.5, the **ACODEC_ENABLE_BOOST** command is added.

In section 9.4.1, AUDIO_RESAMPLE_ATTR_EX_S, AUDIO_AEC_MODE_E, AI_AEC_CONFIG_S, AI_ANR_CONFIG_S, AI_ALC_CONFIG_S, and AI_VQE_CONFIG_S are added.

Chapter 10 VDEC

In section 10.2, the concept of "channel decoding capability" is added.

In section 10.3, the description in the **Note** field and **See Also** field of HI_MPI_VDEC_CreateChn is updated.

The description in the **Note** fields of HI_MPI_VDEC_SetChnParam, HI_MPI_VDEC_GetChnParam, HI_MPI_VDEC_SetPrtclParam, HI_MPI_VDEC_GetPrtclParam, and HI_MPI_VDEC_SetDisplayFrameNum is updated.

In section 10.4, the description in the **Note** field of VDEC_ATTR_VIDEO_S is updated. The description of the **u32RefFrameNum** parameter is updated in the **Member** field of VDEC_ATTR_VIDEO_S. The description of the **s32DisplayFrameNum** parameter is updated in the **Member** field of VDEC_PRTCL_PARAM_S.

Chapter 13 Proc Debugging Information

In section 13.10, the debugging information and parameter description of RC are updated.

In section 13.19, the debugging information and parameter description of AI are updated.

In section 13.20, the debugging information and parameter description of AO are updated.

In section 13.21, the debugging information and parameter description of AENC are updated.

Section 13.25 is added.

Issue 19 (2014-05-09)

This issue is the nineteenth official release, which incorporates the following changes:

Chapter 4 VO

In section 4.3, notes are added to the **Note** fields of HI_MPI_VO_EnableSdTdeBypass and HI_MPI_VO_DisableSdTdeBypass, and the MPIs HI_MPI_VO_EnableRecvFrameRateMatch and HI_MPI_VO_DisableRecvFrameRateMatch are added.

Chapter 6 VENC

Section 6.2.10 is added.

In section 6.3, HI_MPI_VENC_GetStreamBufInfo is added.

Chapter 7 VDA

In section 7.3, the notes in the **Note** field of HI_MPI_VDA_ReleaseData are updated.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chapter 8 Region Management

In section 8.2.1, the concepts of OSD color inversion and supported modules are added.

Chapter 9 Audio

In section 9.2.1.4, a caution is added.

In section 9.3, the notes in the **Note** fields of HI_MPI_AI_Disable, HI_MPI_AI_GetFrame, HI_MPI_AO_SetPubAttr are updated.

Chapter 10 VDEC

In section 10.3, HI_MPI_VDEC_SetDisplayFrameNum, HI_MPI_VDEC_GetDisplayFrameNum, HI_MPI_VDEC_SetChnPrevMode, and HI_MPI_VDEC_GetChnPrevMode are added.

In section 10.4, the default value of **s32DisplayFrameNum** for the H.264 channel is changed to **2** in the **Member** field, and a note is added to the **Note** field of VDEC_PRTCL_PARAM_S.

Chapter 13 Proc Debugging Information

In section 13.13, the **PREVMODE** parameter is added.

In section 13.15, the **bRcvMatch** parameter is added.

Issue 18 (2014-02-26)

This issue is the eighteenth official release, which incorporates the following changes:

Chapter 2 System Control

In section 2.3, the description in the **Note** field of HI_MPI_SYS_SetMemConf is updated.

Chapter 3 VI

In section 3.5, the description in the **Note** field of LDC_ATTR_S is updated.

Chapter 4 VO

In section 4.3, HI_MPI_VO_EnableSdTdeBypass and HI_MPI_VO_DisableSdTdeBypass are added.

Chapter 5 VPSS

In section 5.3, HI_MPI_VPSS_SetGrpField and HI_MPI_VPSS_GetGrpField are added.

Chapter 6 VENC

In section 6.3, HI_MPI_VENC_RequestIDRInst is added.

Chapter 7 VDA

In section 7.3, the description in the **Note** field of HI_MPI_VDA_UpdateRef is updated.

Chapter 9 Audio

In section 9.2.1.4, cautions are added below Figure 9-4.

Chapter 13 Proc Debugging Information

In section 13.8 and section 13.9, the **ContiBuffer** parameter is added.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

In section 13.12, the **BuffCache** parameter is added.

Issue 17 (2014-01-20)

This issue is the seventeenth official release, which incorporates the following changes:

The descriptions of the Hi3518E are added.

Issue 16 (2013-11-30)

This issue is the sixteenth official release, which incorporates the following changes:

Chapter 4 VO

In section 4.3, HI_MPI_VO_SetDevFramerate and HI_MPI_VO_GetDevFramerate are added.

In section 4.4, the description of "user-defined timing" is deleted from the **Note** field of VO_PUB_ATTR_S.

Chapter 13 Proc Debugging Information

In sections 13.15 and 13.16, the descriptions are updated.

Issue 15 (2013-09-25)

This issue is the fifteenth official release, which incorporates the following changes:

Chapter 2 System Control

In section 2.1, the description "The MPP does not support calling multiple processes" is added.

In Table 2-1, the descriptions related to VDEC and VO are updated.

The description "You only need to use one process to initialize the system" is removed from the **Note** field of HI_MPI_SYS_Init.

Chapter 3 VI

The member u32AutoDisInt and its description are added to VI_CHN_STAT_S.

Chapter 4 VO

HI_MPI_VO_EnableChnDoubleFrame and HI_MPI_VO_DisableChnDoubleFrame are added.

HI_MPI_VO_SetVtth and HI_MPI_VO_GetVtth are added.

VO_OUTPUT_1600x1200_60, VO_OUTPUT_1680x1050_60, VO_OUTPUT_1920x1200_60, and VO_OUTPUT_640x480_60 are added to **enIntfSync** of VO_PUB_ATTR_S. The maximum value of **enIntfSync** in the **Note** field is changed to **22**.

Chapter 5 VPSS

In section 5.2.3, chn1 is added and some descriptions of chn0 are updated.

Figure 5-3 is updated.

Chapter 6 VENC

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

The **Example** fields of HI_MPI_VENC_CreateGroup and HI_MPI_VENC_StartRecvPicEx are updated.

Stream feature information stMpeg4Info is added to the **Note** field of HI_MPI_VENC_GetStream.

The descriptions in the **Syntax** fields of HI_MPI_VENC_SetH264eRefParam and HI_MPI_VENC_GetH264eRefParam are updated.

In the **Note** field of HI_MPI_VENC_GetJpegSnapMode, "channel group" is changed to "channel".

VENC_PARAM_H264_CBRV2_S is added.

The descriptions in the Syntax fields of VENC_STREAM_INFO_H264_S, VENC_STREAM_INFO_JPEG_S, VENC_STREAM_INFO_MPEG4_S, VENC_RC_ATTR_S, VENC_RC_MODE_E, VENC_PARAM_H264_CBR_S, VENC_PARAM_H264_VBR_S, VENC_PARAM_MJPEG_CBR_S, VENC_PARAM_MJPEG_VBR_S, VENC_PARAM_MPEG4_CBR_S, VENC_PARAM_MPEG4_VBR_S, and VENC_RC_PARAM_S are updated.

Chapter 10 VDEC

VDEC_DECODE_ERROR_S is added.

The descriptions in the **Syntax** field of VDEC_CHN_STAT_S are updated.

Chapter 12 HDMI

HI_MPI_HDMI_SetInfoFrame and HI_MPI_HDMI_GetInfoFrame are added.

HI_HDMI_INFOFRAME_TYPE_E, HI_HDMI_AVI_INFOFRAME_VER2_S, HI_HDMI_AUD_INFOFRAME_VER1_S, HI_HMDI_INFORFRAME_UNIT_U, and HI_HDMI_INFOFRAME_S are added.

Chapter 13 Proc Debugging Information

In section 13.13 and section 13.15, the descriptions are updated.

Issue 14 (2013-07-31)

This issue is the fourteenth official release, which incorporates the following changes:

Chapter 9 Audio

In section 9.3.2, HI_MPI_AO_SetVolume and HI_MPI_AO_GetVolume are added.

Issue 13 (2013-07-03)

This issue is the thirteenth official release, which incorporates the following changes:

Chapter 13 Proc Debugging Information

In section 13.4, the descriptions are updated.

Issue 12 (2013-06-21)

This issue is the twelfth official release, which incorporates the following changes:

Chapter 5 VPSS

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

The descriptions of the Hi3515C are added.

In section 5.4, the value range and default value of the **u32TfStrength** parameter are changed in the **Member** field of VPSS_GRP_PARAM_S.

Chapter 8 Region Management

In section 8.4, notes are added to the **Note** fields of OVERLAY_ATTR_S, OVERLAY_CHN_ATTR_S, and COVEREX_CHN_ATTR_S.

Issue 11 (2013-05-21)

This issue is the eleventh official release, which incorporates the following changes:

Chapter 3 VI

The descriptions of the Hi3515A are added.

In section 3.3.4, the descriptions of Dev0 are updated.

In section 3.4, the descriptions are updated in the **Note** field of HI_MPI_VI_SetLDCAttr.

Chapter 4 VO

In Table 4-1, the descriptions of the Hi3520D and Hi3515A are updated.

In section 4.3, the descriptions are updated in the **Description** field of HI_MPI_VO_SetChnField, **Note** field of HI_MPI_VO_SetZoomInWindow, and **Difference** fields of HI_MPI_VO_SetDevCSC and HI_MPI_VO_SetVgaParam. The minimum value of **u32BufLen** is changed to **3** in the **Parameter** field of HI_MPI_VO_SetDispBufLen. Two notes are added to the **Note** field of HI_MPI_VO_SetChnFrameRate. HI_MPI_VO_SetChnDispThreshold and HI_MPI_VO_GetChnDispThreshold are added.

In section 4.4, the descriptions are updated in the **Note** field of VO_ZOOM_ATTR_S. The value ranges of members are added to the **Member** fields of VO_CSC_S and VO_VGA_PARAM_S.

Chapter 5 VPSS

In section 5.2.2, a note related to the VPSS channel and VENC module is added.

Chapter 8 Region Management

In section 8.3, HI_MPI_RGN_SetAttachField and HI_MPI_RGN_GetAttachField are added.

In section 8.4, RGN_ATTACH_FIELD_E is added, and the value range of **stRect** is changed in the **Member** fields of COVER_CHN_ATTR_S and COVEREX_CHN_ATTR_S.

Chapter 9 Audio

In section 9.3.5.1, the descriptions are updated in the **Note** fields of ACODEC_SET_MIXER_MIC, ACODEC_SET_GAIN_MICL, and ACODEC_SET_GAIN_MICR.

Chapter 12 HDMI

In section 12.3, HI_MPI_HDMI_SetCsc and HI_MPI_HDMI_GetCsc are added.

In section 12.4, HI_HDMI_CSC_S is added.

Chapter 13 Proc Debugging Information

In sections 13.11, 13.14, and 13.15, the descriptions are updated.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Issue 10 (2013-04-03)

This issue is the tenth official release, which incorporates the following changes:

Chapter 3 VI

In section 3.4, a note is added to the **Note** field of HI_MPI_VI_SetUserPic.

Chapter 4 VO

In section 4.4, the descriptions are updated in the **Note** fields of HI_MPI_VO_SetChnAttr, HI_MPI_VO_SetChnDispPos, HI_MPI_VO_SetDevCSC, HI_MPI_VO_GfxLayerBindDev, and HI_MPI_VO_SetGfxLayerCSC.

Chapter 5 VPSS

In Table 5-2, the scaling performance of channel 0 is updated.

In section 5.4, the descriptions of **u32MotionThresh** are updated, and the value range of **u32ChromaRange** is added to the **Member** field of VPSS_GRP_PARAM_S.

Chapter 6 VENC

In section 6.3, the descriptions of **ViFrmRate** are updated in the **Note** field of HI_MPI_VENC_CreateChn.

Chapter 10 VDEC

In section 10.3, the descriptions of Hi3520D and Hi3515A VDEC channels are added to the **Note** field of HI_MPI_VDEC_CreateChn.

Chapter 13 Proc Debugging Information

In section 13.13, the descriptions are updated.

Issue 09 (2013-03-31)

This issue is the ninth official release, which incorporates the following changes:

Chapter 5 VPSS

Figure 5-4 is updated.

Chapter 6 VENC

Section 6.2.5 is added.

In section 6.3, HI_MPI_VENC_SetH264eRefParam and HI_MPI_VENC_GetH264eRefParam are added.

In section 6.4, VENC_ATTR_H264_REF_PARAM_S is added.

Chapter 9 Audio

In section 9.2.1, Table 9-2 and the descriptions of the AIO interface, SIO interface, AI device, and AO device are added.

In section 9.2.3, Table 9-6, Figure 9-8, and the descriptions of the chip using AIO interfaces are added.

In section 9.3.1, the descriptions of the working mode and clock selection are updated in the **Note** field of HI_MPI_AI_SetPubAttr.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chapter 10 VDEC

In section 10.3, the error code HI_ERR_VDEC_BADADDR is added.

Chapter 13 Proc Debugging Information

In sections 13.5 to 13.16, 13.19, and 13.20, the descriptions are updated.

Issue 08 (2013-02-05)

This issue is the eighth official release, which incorporates the following changes:

Chapter 3 VI

In section 3.5, the **bDataRev** member is added to the **Syntax** and **Member** fields of VI_DEV_ATTR_S.

Chapter 4 VO

In section 4.3, the description "The range of the VO device ID is not supported by the Hi3518C or Hi3516C" is added to the **Difference** field of HI_MPI_VO_SetVgaParam and HI_MPI_VO_GetVgaParam.

In section 4.4, the **enDataSource** member is added to the **Syntax** and **Member** fields of VO_WBC_ATTR_S.

Chapter 6 VENC

In section 6.3, HI_MPI_VENC_StartRecvPicEx is added.

In section 6.3, descriptions of **u32LeftRecvPics** and **u32LeftEncPics** are added to the **Note** field of HI_MPI_VENC_Query.

In section 6.4, **u32LeftRecvPics** and **u32LeftEncPics** are added to the **Syntax** and **Member** fields of VENC_CHN_STAT_S.

In section 6.4, the maximum value of the **u32StatTime** member is changed from **10** to **16** in the **Member** field of VENC_ATTR_MJPEG_VBR_S.

Chapter 9 Audio

The sections 9.2.4, 9.3.5, and 9.4.4 are added.

Issue 07 (2013-01-15)

This issue is the seventh official release, which incorporates the following changes:

Chapter 9 Audio

In section 9.3.2, the HI_MPI_AO_QueryChnStat MPI is added. In section 9.4.1, the AO_CHN_STATE_S data structure is added.

Chapter 13 Proc Debugging Information

In section 13.5, the parameters **IntPro** and **IntCostL** are added.

In section 13.7, debugging information is updated.

In section 13.14, the parameters **detect_err_frame**, **MaxIntT**, **OverCnt**, **LintCnt**, and **ThrCnt** are added to VI debugging information.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

In section 13.19, the parameters **MaxFrmTime**, **MaxIsrTime**, **bAnr**, **bResmp**, **PoiNum**, **SampR**, and **ResmpType** are added.

In section 13.20, the parameters **FrmTime**, **MaxFrmTime**, **IsrTime**, **MaxIsrTime**, **bResmp**, **PoiNum**, **SampR**, and **ResmpType** are added.

Issue 06 (2012-12-26)

This issue is the sixth official release, which incorporates the following changes:

Hi3516C information is added.

Chapter 3 VI

In section 3.3.3, descriptions of the mask configuration and hardware pin are added.

Usage restrictions on **s32Y** and **u32Height** in the channel attribute parameter **stCapRect** are added to the **Difference** field of HI_MPI_VI_SetChnAttr. The description that **s32Y** and **u32Height** must be 4-pixel aligned in interlaced dual-field mode is deleted from the **Note** field.

Descriptions of **stDestSize** for the Hi3518 or Hi3516C are modified in the **Difference** field of VI_CHN_ATTR_S.

Chapter 4 VO

In section 4.3, descriptions are modified in the **Note** field of HI_MPI_VO_DisableChn.

In section 4.3, the description "If the HD device's timing is interlaced and the display format is SPYCbCr420, the height must be 4-pixel aligned" is added to the **Note** field of HI_MPI_VO_SetChnAttr.

In section 4.4, description of the **stRect** member "If the HD device's timing is interlaced and the display format is SPYCbCr420, the height must be 4-pixel aligned" is added to the **Member** field of VO_CHN_ATTR_S. The description "The Channel attribute stRect of the SD device must be 2-pixel aligned" is deleted from the **Note** field.

Chapter 5 VPSS

The **enCapSel** parameter and its description are added to the **Syntax** field of VPSS_CROP_INFO_S.

The maximum value of **u32MaxH** for the Hi3521, Hi3520A, Hi3531, and Hi3532 are updated to **4080** in the **Member** field of VPSS_GRP_ATTR_S.

Chapter 6 VENC

The maximum value for the **u32MCUPerECS** parameter is changed in the **Note** fields of HI_MPI_VENC_SetJpegParam and HI_MPI_VENC_SetMjpegParam.

In section 6.4, the minimum value for **u32BitRate** is changed to **2** in the **Member** field of VENC_ATTR_H264_CBR_S. Descriptions of **ViFrmRate** are updated in the **Note** field.

In section 6.4, the minimum value for **u32BitRate** is changed to **2** in the **Member** fields of VENC_ATTR_MPEG4_CBR_S and VENC_ATTR_MJPEG_CBR_S.

In section 6.4, the minimum value for **u32MaxBitRate** is changed to **2** in the **Member** fields of VENC_ATTR_H264_VBR_S, VENC_ATTR_MPEG4_VBR_S, and VENC_ATTR_MJPEG_VBR_S.

Chapter 9 Audio

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

In section 9.3, the descriptions "Call this MPI to enable noise reduction and echo cancellation if you want to use the functions after enabling an AI channel again" are added to the **Note** field of HI_MPI_AI_EnableAec and HI_MPI_AI_EnableAnr respectively.

In section 9.3 and section 9.4, the following MPIs and data types are added: HI_MPI_AENC_RegeisterEncoder, HI_MPI_AENC_UnRegisterEncoder, HI_MPI_DENC_RegeisterEncoder, HI_MPI_DENC_UnRegisterEncoder, AENC_ENCODER_S, and ADNC_DECODER_S.

In section 9.3, the minimum value of the buffer is modified in the **Note** field of HI_MPI_AENC_CreateChn.

In section 9.3, the **Syntax** field is updated for HI_MPI_ADEC_CreateChn.

In section 9.3, HI_MPI_DENC_GetFrame, HI_MPI_DENC_ReleaseFrame, and AUDIO_FRAME_INFO_S are deleted.

In section 9.4, the minimum buffer size is changed to **2** in the **Syntax** and **Member** fields of AENC_CHN_ATTR_S.

In section 9.4, the minimum value of **u32BufSize** is changed to **2** in the **Member** field of ADEC_CHN_ATTR_S.

The minimum value for **u32BufSize** is changed to **2** in the **Member** field of ADEC_CHN_ATTR_S.

In section 9.2.2, the description that "You are not advised to use the ADPCM_ORG_DVI4 packet encapsulation format in audio stream transmission applications" is added to Table 9-3.

In section 9.3.3, the example that an encoding protocol can be used to register only one encoder of the same type is modified in the **Note** field of HI_MPI_AENC_RegeisterEncoder.

Chapter 11 IVE

In section 11.2, the description "The histogram output stride is fixed at 256" is added.

Chapter 13 Proc Debugging Information

In section 13.24, descriptions of the **EDID Parse Status** parameter is updated.

Issue 05 (2012-11-30)

This issue is the fifth official release, which incorporates the following changes:

Chapter 5 VPSS

In section 5.2.1, descriptions of backup nodes are added to the Note fields of HI_MPI_VPSS_UserGetGrpFrame.

Chapter 13 Proc Debugging Information

In section 13.3, descriptions of the **MinFree** parameter are added to the debugging information.

In section 13.8, descriptions of the **slcmode** parameter are modified in the debugging information of the Syntax INFO.

In section 13.10, the HDMI debugging information is added.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Issue 04 (2012-11-25)

This issue is the fourth official release, which incorporates the following changes:

Chapter 2 System Control

In Table 2-1, the data receiver VDA is added for the data sources VPSS and VDEC.

Chapter 3 VI

In Table 3-1, the maximum picture size is changed to 2560x1600 for port 0, port 2, port 4, and port 6.

In section 3.4, the descriptions are updated in the **Note** fields of HI_MPI_VI_SetFlashConfig, HI_MPI_VI_SetExtChnAttr, and HI_MPI_VI_SetLDCAttr.

In section 3.5, VI_DATA_TYPE_E is added and the descriptions are updated in the **Note** field of VI_EXT_CHN_ATTR_S.

Chapter 4 VO

In section 4.3, the descriptions are updated in the **Difference** and **Note** fields of HI_MPI_VO_SetPubAttr. The descriptions are updated in the **Note** fields of HI_MPI_VO_SetChnAttr, HI_MPI_VO_SetChnDispPos, HI_MPI_VO_SetZoomInWindow, HI_MPI_VO_GetZoomInWindow, HI_MPI_VO_SetAttrBegin, HI_MPI_VO_ReleaseChnFrame, HI_MPI_VO_SetAttrEnd, HI_MPI_VO_ChnShow, HI_MPI_VO_QueryChnStat, HI_MPI_VO_GetScreenFrame, and HI_MPI_VO_ReleaseScreenFrame. The data types are updated in the **Syntax** fields of HI_MPI_VO_SetVgaParam and HI_MPI_VO_GetVgaParam. The maximum value of **u32Layer** is changed in the **Parameter** fields of HI_MPI_VO_SetGfxLayerCSC and HI_MPI_VO_GetGfxLayerCSC.

In section 4.4, the descriptions are updated in the **Difference** and **Note** fields of VO_VIDEO_LAYER_ATTR_S. The API name is changed in the **Syntax** field of HI_MPI_VPSS_CreateGrp.

Chapter 5 VPSS

In section 5.3, the **vpsschn** parameter is deleted from the **Syntax** field of HI_MPI_VPSS_SetGrpParam. The descriptions are updated in the **Syntax** and **Parameter** fields of HI_MPI_VPSS_SetDepth and the **Description** field of HI_MPI_VPSS_UserGetFrame. The MPIs HI_MPI_VPSS_UserGetGrpFrame and HI_MPI_VPSS_UserReleaseGrpFrame are added.

In section 5.4, the value range and default value of **u32SfStrength** (Hi3518) are changed and the description of **u32ChromaRange** (Hi3518) is updated in the **Member** field of VPSS_GRP_PARAM_S.

Chapter 6 VENC

In section 6.3, the descriptions are updated in the **Note** fields of HI_MPI_VENC_SetH264InterPred and HI_MPI_VENC_SetRcPara.

In section 6.4, the descriptions of **bVIField** are updated and **u32Profile** is added to the **Member** field of VENC_ATTR_H264_S. The descriptions of **bVIField** are added to the **Member** fields of VENC_ATTR_MJPEG_S and VENC_ATTR_JPEG_S. The descriptions of **u32MCUPerECS** are updated in the **Member** fields of VENC_PARAM_JPEG_S and VENC_PARAM_MJPEG_S. The maximum value of **u32Gop** is changed in the **Member** fields of VENC_ATTR_H264_CBR_S, VENC_ATTR_H264_VBR_S, VENC_ATTR_H264_FIXQP_S, VENC_ATTR_MPEG4_CBR_S,

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VENC_ATTR_MPEG4_VBR_S, and VENC_ATTR_MPEG4_FIXQP_S. The value ranges of **u32MinIprop**, **u32MaxIprop**, **u32MinQp**, **u32SuperIFrmBitsThr**, and **u32SuperPFrmBitsThr** are changed in the **Member** field of VENC_PARAM_H264_CBR_S. The value ranges of **u32SuperIFrmBitsThr** and **u32SuperPFrmBitsThr** are changed in the **Member** field of VENC_PARAM_H264_VBR_S. The value range of **u32SuperFrmBitsThr** is changed in the **Member** field of VENC_PARAM_MJPEG_CBR_S. The value range of **u32SuperFrmBitsThr** is changed in the **Member** field of VENC_PARAM_MJPEG_VBR_S. The value ranges of **u32SuperIFrmBitsThr** and **u32SuperPFrmBitsThr** are changed in the **Member** field of VENC_PARAM_MPEG4_CBR_S. The value ranges of **u32SuperIFrmBitsThr** and **u32SuperPFrmBitsThr** are changed in the **Member** field of VENC_PARAM_MPEG4_VBR_S.

Chapter 7 VDA

In section 7.4, the descriptions are updated in the **Syntax** field of VDA_MAX_HEIGHT.

Chapter 9 Audio

A note is added below Figure 9-7.

In section 9.3, the descriptions are updated in the **Note** fields of HI_MPI_AI_EnableAec, HI_MPI_AI_EnableReSmp, HI_MPI_AI_EnableAnr, HI_MPI_AI_DisableAnr, and HI_MPI_AO_EnableReSmp.

In section 9.4, AIO_MAX_CHN_NUM is added. The descriptions are updated in the **Note** fields of AUDIO_BIT_WIDTH_E and AIO_MODE_E.

The error code HI_ERR_AI_NOT_ENABLED is modified and 0xA015800B is deleted from Table 9-7.

Chapter 10 VDEC

In section 10.3, a note is added to the **Note** field of HI_MPI_VDEC_CreateChn.

Chapter 11 IVE

In section 11.2.1, the stride concept is added.

Chapter 12 HDMI

In section 12.3, HI_MPI_HDMI_GetDeepColor is added.

Issue 03 (2012-10-30)

This issue is the third official release, which incorporates the following changes:

Chapter 3 VI

In section 3.4, the descriptions are updated in the **Note** fields of HI_MPI_VI_SetExtChnAttr, HI_MPI_VI_SetCSCAttr, and HI_MPI_VI_SetRotate.

Chapter 4 VO

In section 4.3, the MPIs HI_MPI_VO_SetChnField, HI_MPI_VO_GetChnField, HI_MPI_VO_SetGfxLayerCSC, and HI_MPI_VO_GetGfxLayerCSC are added. HI_MPI_VPSS_SetClipCfg is changed to HI_MPI_VPSS_SetCropCfg in the **Note** field of HI_MPI_VO_ChnRefresh. The value range of the **enGfxLayer** parameter is changed to [0, VOU_GRAPHICS_LAYER_BUTT) in the **Parameter** fields of HI_MPI_VO_GfxLayerBindDev and HI_MPI_VO_GfxLayerUnBindDev.

Chapter 5 VPSS

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

In section 5.3, the MPIs HI_MPI_VPSS_SetDelay and HI_MPI_VPSS_GetDelay are added. The descriptions of the crop area are added to the **Note** field of HI_MPI_VPSS_SetCropCfg. The attribute of **pstVideoFrame** is changed to Output in the **Parameter** field of HI_MPI_VPSS_UserGetFrame.

In section 5.4, the parameters are described by chip in the **Member** field of VPSS_GRP_ATTR_S. The value ranges of **u32SfStrength** and **u32TfStrength** are changed in the **Member** field of VPSS_GRP_PARAM_S. The descriptions are updated in the **Syntax** and **Member** fields of VPSS_CAPSEL_E.

Chapter 6 VENC

The description of the JPEG snapshot mode supported by the Hi3518 is added below Figure 6-1.

Section 6.2.7 "JPEG Snapshot Modes" is added.

In section 6.3, the **u32MaxQp**, **u32MinQp**, and **u32MaxStartQp** parameters are added to the **Note** field of HI_MPI_VENC_SetRcPara. The description of the JPEG snapshot mode is added to the **Note** field of HI_MPI_VENC_SetJpegSnapMode.

In section 6.4, the value range of **u32MaxQp** is changed to (u32MinQp, 31] in the **Member** field of VENC_ATTR_MPEG4_VBR_S. The **u32MaxStartQp** parameter is added to the **Syntax** and **Member** fields of VENC_PARAM_H264_CBR_S. The value range of **u32MaxQp** is changed to (u32MinQp, 51] in the **Member** field of VENC_PARAM_H264_CBR_S. The **u32MaxStartQp** parameter is added to the **Syntax** and **Member** fields of VENC_PARAM_MPEG4_CBR_S. The value range of **u32MinIprop** is changed to (0, 100], the value range of **u32MaxIprop** is changed to (u32MinIprop, 100], and the value range of **u32MaxQp** is changed to (u32MinQp, 31] in the **Member** field of VENC_PARAM_MPEG4_CBR_S.

Chapter 9 Audio

The description of the internal audio CODEC used the Hi3518 is added below Figure 9-7.

Chapter 12 HDMI

In section 12.3, HI_MPI_HDMI_SetDeepColor is added.

Issue 02 (2012-09-20)

This issue is the second official release, which incorporates the following changes:

Chapter 5 VPSS

In Table 5-1, Table 5-2, and Table 5-3, the supported scaling directions are added. The descriptions of extended channels of the Hi3518 are added below Figure 5-3.

In section 5.5, the error code HI_ERR_VPSS_NOT_SUPPORT is added.

Chapter 6 VENC

In section 6.1, the input sources of the VENC module are updated.

In section 6.2.1, the descriptions of encoding protocols and encoding operations are updated.

In section 6.3, the descriptions of the MPEG-4 protocol and static attributes are updated, and the descriptions of bit rate fluctuation levels are added below Table 6-5.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

In section 6.4, VENC_ATTR_MPEG4_VBR_S, VENC_PARAM_MJPEG_CBR_S, VENC_PARAM_MJPEG_VBR_S, VENC_PARAM_MPEG4_CBR_S, and VENC_PARAM_MPEG4_VBR_S are added.

Chapter 7 VDA

In section 7.2.3, the input sources are updated.

In section 7.4, the data types VDA_MAX_WIDTH and VDA_MAX_HEIGHT are added.

Chapter 13 Proc Debugging Information

In section 13.18, the **VpssSd** parameter is added.

Issue 01 (2012-08-30)

This issue is the first official release, which incorporates the following changes:

Chapter 3 VI

In section 3.4, the MPIs HI_MPI_VI_EnableChnInterrupt and HI_MPI_VI_DisableChnInterrupt are added, and the limitations on the Hi3518 channel attribute parameters are added to the **Difference** field of HI_MPI_VI_SetChnAttr.

In section 3.5, the **Syntax** field is added to LDC_VIEW_TYPE_E.

Chapter 4 VO

In this chapter, the descriptions of the Hi3520A and Hi3521 are combined.

In section 4.2, the sentence "The total size of the PIP layer cannot be greater than D1" is deleted.

In section 4.3, the last error code and descriptions in the **Error Code** field of HI_MPI_VO_EnableChn are updated, and the descriptions in the Description and Note fields of HI_MPI_VO_GfxLayerBindDev and HI_MPI_VO_GfxLayerUnBindDev are updated.

In section 4.4, the definitions of values 1 and 2 for the Hi3520A are updated in the **Member** field of VO_DEV.

Chapter 5 VPSS

In Table 5-2, the descriptions of the scaling functions of channels 0−3 are updated.

In section 5.1, the descriptions are updated.

In section 5.2, the descriptions of the channel and group concepts are updated, and the differences of the Hi3518 are added.

In section 5.3, HI_MPI_VPSS_SetExtChnAttr and HI_MPI_VPSS_GetExtChnAttr are added, and the descriptions in the **Note** fields of HI_MPI_VPSS_EnableChn and HI_MPI_VPSS_DisableChn are updated.

In section 5.4, VPSS_MAX_PHY_CHN_NUM, VPSS_MAX_EXT_CHN_NUM, and VPSS_EXT_CHN_ATTR_S are added.

Chapter 6 VENC

In section 6.3, the advanced parameters related to quantization tables are modified in the **Note** and **Example** fields of HI_MPI_VENC_SetJpegParam and HI_MPI_VENC_SetMjpegParam.

Chapter 8 Region Management

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

In section, the descriptions of the color inversion trigger mode parameters are updated in the **Syntax** and **Member** fields of OVERLAY_INVERT_COLOR_S.

Chapter 9 Audio

In section 9.3, the MPIs HI_MPI_ADEC_GetFrame and HI_MPI_ADEC_ReleaseFrame are added, and the examples for obtaining audio frames from an ADEC channel and releasing audio frames are modified in the **Example** field of HI_MPI_ADEC_CreateChn.

Chapter 11 IVE

In section 11.2.1, the descriptions of flushing data are updated, and the descriptions of input and output data formats are added.

In section 11.3, a **Formula** field is added to HI_MPI_IVE_DMA, HI_MPI_IVE_FILTER, HI_MPI_IVE_CSC, HI_MPI_IVE_SOBEL, HI_MPI_IVE_CANNY, HI_MPI_IVE_DILATE, HI_MPI_IVE_ERODE, HI_MPI_IVE_THRESH, HI_MPI_IVE_AND, HI_MPI_IVE_SUB, HI_MPI_IVE_OR, HI_MPI_IVE_INTEG, and HI_MPI_IVE_HIST each.

Chapter 13 Proc Debugging Information

In section 13.10, the RC debugging information updated.

In section 13.14, the parameter descriptions of physical VI channels in status 1 are updated.

Issue 00B80 (2012-08-15)

This issue is the eleventh draft release, which incorporates the following changes:

Chapter 5 VPSS

The descriptions of the Hi3518 are added.

Issue 00B70 (2012-08-09)

This issue is the tenth draft release, which incorporates the following changes:

The descriptions of the Hi3518 are added.

Chapter 3 VI

In section 3.2, the concept of lens distortion correction is added.

In section 3.3.1, the descriptions of binding relationships are updated.

Section 3.3.3 is added.

In section 3.4, a note is added to the **Note** field of HI_MPI_VI_GetFrame, HI_MPI_VI_GetFrameTimeOut is added, the value range of the Hi3531/Hi3532 VI channel ID is changed to [0, VIU_MAX_CHN_NUM), and the MPIs HI_MPI_VI_SetFlashConfig, HI_MPI_VI_GetFlashConfig, HI_MPI_VI_FlashTrigger, HI_MPI_VI_SetExtChnAttr, HI_MPI_VI_GetExtChnAttr, HI_MPI_VI_SetLDCAttr, HI_MPI_VI_GetLDCAttr, HI_MPI_VI_SetCSCAttr, HI_MPI_VI_GetCSCAttr, HI_MPI_VI_SetRotate, HI_MPI_VI_GetRotate, and HI_MPI_VI_GetChnLuma are added.

In section 3.5, the **Syntax** fields of VIU_MAX_PHYCHN_NUM, VI_USERPIC_MODE_E are updated, and the data types ROTATE_E, VI_DATA_PATH_E, VI_EXT_CHN_ATTR_S, LDC_VIEW_TYPE_E, LDC_ATTR_SVI_LDC_ATTR_S, VI_CSC_TYPE_E, VI_CSC_ATTR_S, VIU_EXT_CHN_START, VIU_MAX_EXT_CHN_NUM,

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VIU_MAX_EXTCHN_BIND_PER_CHN, VI_FLASH_MODE_E, VI_FLASH_CONFIG_S, and VI_CHN_LUM_S are added.

Chapter 4 VO

The descriptions of the Hi3518 are added.

In section 4.3, the limitations on the width and height of SD device channels are added, the value range of the CSC matrix is added to the **Note** field of HI_MPI_VO_SetDevCSC, and the MPIs HI_MPI_VO_SetChnDispPos and HI_MPI_VO_GetChnDispPos are added.

In section 4.4, VO_CSC_MATRIX_E is added, and the **enCscMatrix** parameter is added to VO_CSC_S.

Chapter 5 VPSS

In section 5.2.1, the concept of Sizer is added.

In section 5.3, HI_MPI_VPSS_SetGrpSizer, HI_MPI_VPSS_GetGrpSizer, HI_MPI_VPSS_SetGrpFrameRate, and HI_MPI_VPSS_GetGrpFrameRate are added, and the descriptions in the **Note** field of HI_MPI_VPSS_SetPreScale are updated.

In section 5.4, VPSS_SIZER_INFO_S and VPSS_FRAME_RATE_S are added, the default values of **u32IeStrength** and **u32IeSharp** (Hi3521/Hi3520A) and the values of **u32SfStrength** and **u32TfStrength** (Hi3521/Hi3520A) are changed in the **Member** field of VPSS_GRP_PARAM_S, the value ranges of **u32Width** and **u32Height** are changed in the **Note** field of VPSS_CHN_MODE_S, and the default value of **u32SfStrength** is changed from **8** to **16** in the **Member** field of VPSS_CHN_NR_PARAM_S.

Chapter 6 VENC

The descriptions of the Hi3518 are added.

In section 6.3, HI_MPI_VENC_SetGrpCrop, HI_MPI_VENC_GetGrpCrop, HI_MPI_VENC_SetJpegSnapMode, and HI_MPI_VENC_GetJpegSnapMode are added, the descriptions in the **Note** field of HI_MPI_VENC_SendFrame are updated, the limitations on the horizontal and vertical search windows are added to the **Note** field of HI_MPI_VENC_SetH264InterPred, and the recommended values of **u32Thrd[12]** and **u32QpDelta** at a low bit rate and the default values of **u32MinIprop** and **u32MaxIprop** are changed in the **Note** field of HI_MPI_VENC_SetRcPara.

In section 6.4, VENC_JPEG_SNAP_MODE_EN is added.

Chapter 8 Region Management

The descriptions of the Hi3518 are added.

Chapter 9 Audio

The descriptions of the Hi3518 are added.

In section 9.2, the descriptions of the SIO master mode are updated.

In section 9.3, the value range of the AI channel ID is changed, and the descriptions in the **Note** fields of HI_MPI_AO_SetPubAttr and HI_MPI_AENC_DestroyChn are added.

In section 9.4, the descriptions of the dual-channel mode are added to the **Note** field of AUDIO_SOUND_MODE_E.

Chapter 10 VDEC

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

In section 10.4, the parameter u32Tmp is added from the **Syntax** field of VDEC_ATTR_JPEG_S.

Chapter 13 Proc Debugging Information

In sections 13.2, 13.5, 13.7, 13.10, 13.12, 13.12, 13.14, 13.15, 13.16, 13.21, the parameter descriptions are updated.

Issue 00B60 (2012-07-13)

This issue is the ninth draft release, which incorporates the following changes:

The descriptions of the Hi3520A are added.

Issue 00B50 (2012-06-08)

This issue is the eighth draft release, which incorporates the following changes:

The descriptions of the Hi3520A are added.

Issue 00B40 (2012-04-20)

This issue is the seventh draft release, which incorporates the following changes:

The descriptions of the Hi3521 are added.

Issue 00B30 (2012-03-16)

This issue is the sixth draft release, which incorporates the following changes:

Chapter 3 VI

In section 3.3, the **Parameter** field of HI_MPI_VI_SetChnAttr is updated, the **Note** field of HI_MPI_VI_SetChnAttr is updated, the MPI HI_MPI_VI_GetFrameDepth is added, and the **Example** fields of HI_MPI_VI_SetDevAttr, HI_MPI_VI_SetUserPic and HI_MPI_VI_EnableCascade are updated.

In section 3.4, the **Syntax**, **Member**, and **Note** fields of VI_CHN_ATTR_S are updated.

Chapter 4 VO

In section 4.2, the concept of virtual device is added.

In section 4.3, the **Example** fields of HI_MPI_VO_PipLayerBindDev, HI_MPI_VO_PipLayerUnBindDev, HI_MPI_VO_ClearChnBuffer, HI_MPI_VO_ClearChnBuffer, HI_MPI_VO_GetScreenFrame, HI_MPI_VO_GetDispBufLen, HI_MPI_VO_ReleaseScreenFrame, HI_MPI_VO_SetWbcDepth, and HI_MPI_VO_WbcGetScreenFrame are updated. The **Note** and **Example** fields of HI_MPI_VO_SetAttrBegin and HI_MPI_VO_EnableWbc are updated. The **Parameter**, **Note**, and **Example** fields of HI_MPI_VO_SetDispBufLen are updated. The **Note** field of HI_MPI_VO_SetWbcAttr and HI_MPI_VO_SetWbcMode are updated. The MPIs HI_MPI_VO_GetWbcMode, HI_MPI_VO_SetDevCSC, HI_MPI_VO_GetDevCSC, HI_MPI_VO_SetVgaParam, HI_MPI_VO_GetVgaParam, HI_MPI_VO_SetPlayToleration, and HI_MPI_VO_GetPlayToleration are added.

Chapter 6 VENC

Section 6.2.4 "Frame Skipping Reference Modes" is added.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

In section 6.3, the descriptions of **ViFrmRate** and **TargetFrmRate** below Table 6-2 are updated, and a note is added to the **Note** field of HI_MPI_VENC_SetChnAttr and HI_MPI_VENC_InsertUserData respectively.

In section 6.4, the **Member** fields of VENC_PARAM_H264_INTRA_PRED_S, VENC_PARAM_H264_TRANS_S, VENC_PARAM_H264_ENTROPY_S, VENC_PARAM_H264_VUI_S, VENC_PARAM_JPEG_S, VENC_PARAM_MPEG4_S, VENC_RC_PARAM_S are updated. Two notes are added to the **Note** field of VENC_ATTR_H264_CBR_S. A note is added to the **Note** fields of VENC_ATTR_H264_VBR_S, VENC_ATTR_H264_FIXQP_S, VENC_ATTR_MPEG4_CBR_S, VENC_ATTR_MPEG4_FIXQP_S, VENC_ATTR_MJPEG_FIXQP_S, and VENC_ATTR_MJPEG_VBR_S respectively.

Chapter 8 Region Management

In section 8.4, a note and a figure are added to **Note** field of OVERLAY_CHN_ATTR_S.

Chapter 9 Audio

In section 9.2.1.3, the descriptions of the SIO interface timing are updated.

In section 9.3.1, the **Note** fields of HI_MPI_AI_EnableReSmp and HI_MPI_AI_DisableReSmp are updated.

In section 9.3.2, the **Note** field of HI_MPI_AO_EnableReSmp is updated.

In section 9.3.3, the MPI HI_MPI_AENC_GetFd is added.

In section 9.4.1, the description of **u32EXFlag** is updated in the **Member** field of AIO_ATTR_S, and a note is added to the **Note** field of AUDIO_SOUND_MODE_E.

In section 9.5, the error codes 0xA0158005 and 0xA0168005 are added, and the error codes 0xA015800B and 0xA016800B are deleted.

Chapter 10 VDEC

In section 10.3, a note is added to the **Note** field of HI_MPI_VDEC_GetImage.

Chapter 11 IVE

In section 11.2.1, the stride requirements are updated.

In section 11.3, the **Note** fields of HI_MPI_IVE_DMA, HI_MPI_IVE_FILTER, HI_MPI_IVE_CSC, HI_MPI_IVE_FILTER_AND_CSC, HI_MPI_IVE_SOBEL, HI_MPI_IVE_CANNY, HI_MPI_IVE_DILATE, HI_MPI_IVE_ERODE, HI_MPI_IVE_THRESH, HI_MPI_IVE_AND, HI_MPI_IVE_SUB, HI_MPI_IVE_OR, HI_MPI_IVE_INTEG, and HI_MPI_IVE_HIST are updated.

Chapter 13 Proc Debugging Information

In section 13.5, the **hl(us)** and **he** parameters are added.

In section 13.7, the debugging information and descriptions of the **VoSend** and **VoSndOk** parameters are updated, and a parameter type is added.

In section 13.8, the analysis information and the descriptions of the **Lost** and **Disc** parameters are updated.

In section 13.9, the descriptions of the **NoBuf** and **FullInt** parameters are updated.

In section 13.12, the debugging information is updated, and the descriptions of the **UserRls**, **Interval**, and **FrameRate** parameters are added.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

In section 13.13, the **FrmDis** and **PrtclErr** parameters are added.

In section 13.14, the debugging information is updated, and the descriptions of the **CapW** and **CapH** parameters are added.

In section 13.15, the debugging information is updated, and the **RealRat** and **Mode** parameters are added.

In section 13.23, the analysis information and the descriptions of the **Lost** and **Disc** parameters are updated.

Section 13.24 is added.

Issue 00B20 (2012-02-15)

This issue is the fifth draft release, which incorporates the following changes:

Chapter 2 System Control

In section 2.3, the MPIs HI_MPI_SYS_MmzAlloc_Cached and HI_MPI_SYS_MmzFlushCache are added.

Chapter 3 VI

In section 3.3, the MPI HI_MPI_VI_Query is added.

In section 3.4, the data type VI_CHN_STAT_S is added.

Chapter 4 VO

In section 4.2, the concept of VGA compatibility is added.

In section 4.3, the MPIs HI_MPI_VO_SetDispBufLen, HI_MPI_VO_GetDispBufLen, and HI_MPI_VO_SetWbcMode are added, and the value range of the **VoDev** parameter is changed.

In section 4.4, the description of the **VO_DEV** parameter is changed, and the data type VO_WBC_MODE_E is added.

Chapter 5 VPSS

In section 5.2.2, the descriptions below Figure 5-2 are updated.

In section 5.3, the MPIs HI_MPI_VPSS_SetChnMode, HI_MPI_VPSS_GetChnMode, HI_MPI_VPSS_SetDepth, HI_MPI_VPSS_GetDepth, HI_MPI_VPSS_UserGetFrame, and HI_MPI_VPSS_UserRlsFrame are added.

In section 5.4, the data types VPSS_CHN_MODE_E and VPSS_CHN_MODE_S are added.

Chapter 6 VENC

In section 6.3, the MPIs HI_MPI_VENC_SetH264eRefMode and HI_MPI_VENC_GetH264eRefMode are added.

In section 6.4, the data type H264E_REFSLICE_TYPE_E is added, and the descriptions in the **Member** and **Note** fields of VENC_STREAM_INFO_H264_S are updated.

Chapter 7 VDA

In section 7.4, the value range of the **u32SadTh** parameter of VDA_OD_RGN_ATTR_S is changed.

Chapter 8 Region Management

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

In section 8.1, the sentence "the types and number of regions to be created vary depending on modules" is added.

In section 8.2.1, the descriptions are updated.

In this chapter, Vicover is changed to Cover, VicoverEx to CoverEx, and Vioverlay to OverlayEx.

Chapter 9 Audio

In section 9.2.2, the descriptions are updated.

Table 9-3 is modified.

Chapter 10 VDEC

In section 10.3, the MPIs HI_MPI_VDEC_SetPrtclParam and HI_MPI_VDEC_GetPrtclParam are added.

In section 10.4, the data type VDEC_PRTCL_PARAM_S is added.

Chapter 12 HDMI

This chapter is added.

Chapter 13 Proc Debugging Information

In section 13.2, the debugging information and parameter descriptions are updated.

In section 13.7, the **NotStart** parameter is added.

In section 13.14, the description of the **ccErrN** parameter is updated.

In section 13.16, the **WorkMode**, **Depth**, and **bDouble** parameters are added.

Issue 00B10 (2012-01-15)

This issue is the fourth draft release, which incorporates the following changes:

Chapter 3 VI

In section 3.2.4, the descriptions are updated.

Chapter 4 VO

In section 4.2, the descriptions of scaling and clipping are added.

In section 4.3, the descriptions in the **Note** field of HI_MPI_VO_ChnPause, HI_MPI_VO_ChnResume, HI_MPI_VO_ChnStep, HI_MPI_VO_SetAttrBegin, and HI_MPI_VO_SetWbcDepth are updated.

Chapter 5 VPSS

In section 5.4, the descriptions in the **Description** field of VPSS_SF_WINDOW_E and the descriptions in the **Member** field of VPSS_GRP_PARAM_S are updated.

Chapter 6 VENC

In section 6.3, the descriptions in the **Note** field of HI_MPI_VENC_SetH264InterPred and HI_MPI_VENC_SetRcPara are updated.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

In section 6.4, the descriptions in the **Syntax** field and **Member** field of MPEG4E_PACK_TYPE_E, VENC_STREAM_INFO_MPEG4_S, and VENC_PARAM_H264_INTRA_PRED_S are updated.

Chapter 9 Audio

In section 9.2.1.1, Figure 9-1 is updated.

In section 9.2.1.3, the descriptions of main clocks corresponding to different sampling rates are added.

Issue 00B03 (2011-12-07)

This issue is the third draft release, which incorporates the following changes:

Chapter 3 VI

In section 3.1, the description of two extended cascade channels is added below Figure 3-1.

Section 3.2.4 and section 3.2.5 are added.

In section 3.3, the value range of **ViChn** is changed, the **Note** fields of HI_MPI_VI_ChnBInd and HI_MPI_VI_ChnUnBind are updated, and the MPIs HI_MPI_VI_SetUserPic, HI_MPI_VI_EnableUserPic, HI_MPI_VI_DisableUserPic, HI_MPI_VI_EnableCascade, HI_MPI_VI_DisableCascade, HI_MPI_VI_EnableCascadeChn, and HI_MPI_VI_DisableCascadeChn are added.

In section 3.4, the notes of VIU_MAX_CHN_NUM and VI_DEV_ATTR_S are added, the data types VI_CAS_CHN_1, VI_CAS_CHN_2, VI_USERPIC_MODE_E, VI_USERPIC_BGC_S, and VI_USERPIC_ATTR_S are added, the description of the **s32AdChnId[4]** parameter of VI_DEV_ATTR_S is updated, the description of the **au32CompMask[2]** and **s32AdChnId[4]** parameters of VI_DEV_ATTR_EX_S are updated.

Chapter 4 VO

In section 4.2, the description of video cascade is added.

In section 4.3, the MPIs HI_MPI_VO_SendFrame, HI_MPI_VO_GetScreenFrame, HI_MPI_VO_ReleaseScreenFrame, HI_MPI_VO_SetCascadeAttr, HI_MPI_VO_GetCascadeAttr, HI_MPI_VO_EnableCascadeDev, HI_MPI_VO_DisableCascadeDev, HI_MPI_VO_SetCascadePattern, HI_MPI_VO_GetCascadePattern, HI_MPI_VO_CascadePosBindChn, HI_MPI_VO_CascadePosUnBindChn, HI_MPI_VO_EnableCascade, and HI_MPI_VO_DisableCascade are added.

In section 4.4, the data types VO_CAS_MODE_E, VO_CAS_RGN_E, and VO_CAS_ATTR_S are added, the **Member** and **Note** fields of VO_DEV is updated.

Chapter 5 VPSS

In section 5.3, the HI_MPI_VPSS_UserSendFrame MPI is added, the **Note** field of HI_MPI_VPSS_GetChnAttr is updated, and the **Note** field of HI_MPI_VPSS_SetClipCfg is updated.

In section 5.4, the data types VPSS_SF_WINDOW_E, VPSS_DISPLAY_MODE_E, and RECT_S are added, the **Member** and **Note** fields of VPSS_GRP_ATTR_S are updated, the **Note** field of VPSS_FRAME_S is updated, and the **Syntax** and **Member** fields of VPSS_GRP_PARAM_S are updated.

Chapter 7 VDA

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

This chapter is added.

Chapter 9 Audio

This chapter is added.

Chapter 10 VDEC

In section 10.2, the description of the mode of sending streams is updated.

In section 10.3, the description of HI_ERR_VDEC_SYS_NOTREADY is updated, the **Note** fields of HI_MPI_VDEC_CreateChn and HI_MPI_VDEC_GetUserData is updated, and the **Error Code** and **Note** fields of HI_MPI_VDEC_SetChnParam are updated.

In section 10.4, the **Syntax** and **Member** fields of VDEC_ATTR_JPEG_S are updated, and the **Member** field of VDEC_CHN_STAT_S is updated.

Chapter 12 Proc Debugging Information

In sections 12.3, 12.4, 12.7, 12.11, 12.14, 12.15, and 12.16, the debugging information and parameter descriptions are updated.

In section 12.13, the description of **RlsPic** is updated.

Section 12.17 to section 12.23 are added.

Issue 00B02 (2011-11-10)

This issue is the second draft release.

Issue 00B01 (2011-09-30)

This issue is the first draft release.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Contents

1 Introduction to the HiMPP 1-1
1.1 Overview 1-1
1.2 System Architecture 1-1
1.3 Architecture of the HiMPP 1-2

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figures

Figure 1-1 System architecture of the HiMPP 1-2

Figure 1-2 Internal workflow of the HiMPP 1-3

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

1 Introduction to the HiMPP

NOTE

In this document, the Hi35xx indicates the Hi3531, Hi3532, Hi3521, Hi3520A, Hi3518, Hi3516C, Hi3520D, or Hi3515A..

1.1 Overview

HiSilicon provides a media processing platform (HiMPP for short) for rapid application development. For applications, the HiMPP shields the complex processing procedures at the bottom layer related to the chip, and provides HiMPP programming interfaces (MPIs) for implementing various functions. The functions include:

- Capturing of input videos
- H.264, MJPEG, JPEG, or MPEG4 encoding
- H264, VC1, MPEG4, MPEG2, or AVS decoding
- Displaying of output videos
- Video and picture pre-processing, including noise reduction (NR), image enhancement (IE), sharpening (SP), and de-interlacing (DIE)
- On-screen display (OSD) overlaying for encoded streams
- Video detection and analysis
- Intelligent analysis
- Audio capturing and output
- Audio encoding and decoding

This document describes the architecture and modules of the HiMPP, and the MPIs of each module. It is intended for but not limited to application development engineers and technical support personnel.

1.2 System Architecture

As shown in Figure 1-1, the HiMPP provides the following layers:

- Hardware layer

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

It is comprised of the Hi35xx and peripherals, including the flash memory, double-date rate (DDR), video sensor or video analog-to-digital converter (VADC), and audio analog-to-digital converter (AADC).

- Operating system (OS) layer

  It is based on the Linux 3.0 OS.

- MPP

  Based on the OS layer, the MPP controls the media processing functions. That is, the MPP shields the details about hardware processing, and provides application programming interfaces (APIs) for the application layer.

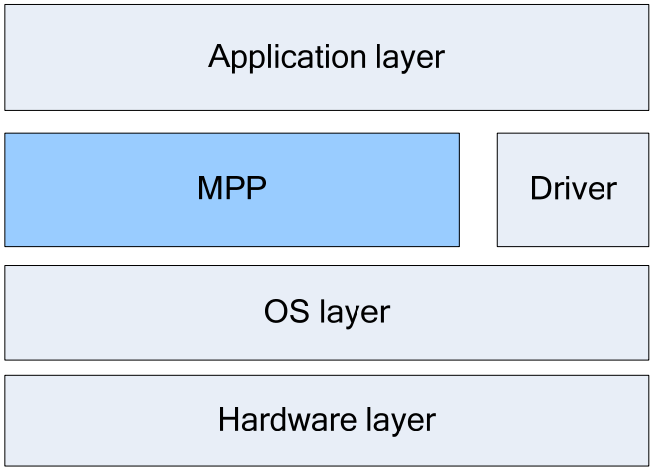
- Drivers

  HiSilicon provides drivers for the hardware processing units of the Hi35xx. The drivers include the gigabit media access controller (GMAC) driver, secure digital input/output (SDIO) driver, inter-integrated circuit (I^2C) driver, Universal Serial Bus (USB) driver, and synchronous serial port (SSP) driver.

- Application layer

  It is used for developing applications based on the MPP and drivers.

Figure 1-1 System architecture of the HiMPP



1.3 Architecture of the HiMPP

The HiMPP has the following modules:

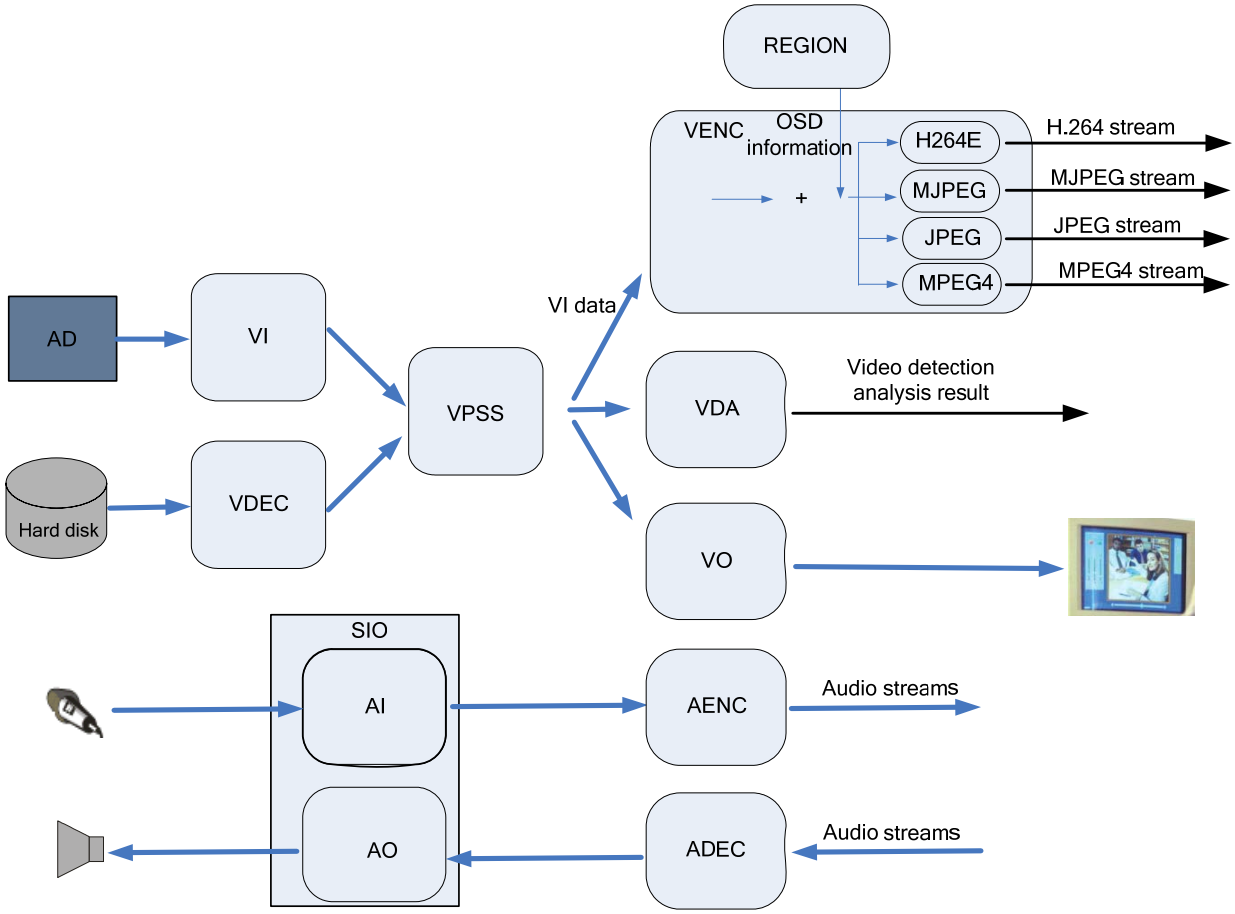
- Video input unit (VIU)
- Video processing subsystem (VPSS)
- Video encoding (VENC) module
- Video encoding (VDEC) module
- Video output unit (VOU)
- Image signal processor (ISP)
- Video detection analysis (VDA) module
- Audio input unit (AIU)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Audio output unit (AOU)
- Audio encoding (AENC) module
- Audio decoding (ADEC) module
- REGION module

Figure 1-2 shows the internal workflow of the HiMPP.

Figure 1-2 Internal workflow of the HiMPP



The following describes the module functions:

- The VIU captures video pictures, crops, scales, and mirrors the pictures, and outputs pictures with different resolutions.
- The VDEC module decodes the encoded video streams and transfers the decoded streams to the VPSS or VOU. The VDEC module can decode the video streams in the format of H.264, VC1, MPEG4, MPEG2, or AVS.
- The VPSS receives the pictures from the VIU or VDEC module and performs denoising, image enhancement, and sharpening on the pictures. In addition, the VPSS can output the pictures from the same source as multi-channel pictures with different resolutions for encoding, previewing, or snapshot.
- The VENC module receives the pictures that are captured by the VIU and processed by the VPSS, and overlays the OSD pictures configured by using the REGION module, and encodes the pictures and output streams complying with different protocols.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- The VDA module receives pictures from the VIU, performs motion detection analysis and cover detection analysis, and outputs results.
- The VOU receives the pictures processed by the VPSS, performs play control, and outputs pictures to external video devices based on the configured output protocols.
- The AIU captures the audio data, and the AENC module encodes the captured audio data complying with multiple audio protocols and outputs encoded audio streams.
- You can transmit the audio streams from the Internet or external storage devices to the ADEC module. The ADEC module can decode the streams in different audio formats, and transmit the decoded data to the AOU for playing.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Contents

2 System Control 2-1
2.1 Overview 2-1
2.2 Functions 2-1
2.2.1 VB Pool 2-1
2.2.2 Binding the System 2-2
2.3 API Reference 2-3
2.4 Data Type 2-38
2.4.1 Basic Data Types 2-38
2.4.2 System Control Data Type 2-50
2.4.3 Public Video Data Type 2-52
2.5 Error Codes 2-58
2.5.1 Error Codes for System Control APIs 2-58
2.5.2 Error Codes for VB Pool APIs 2-58

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figures

Figure 2-1 Typical data flows of public VB pools.....2-2

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Tables

Table 2-1 Binding relationships supported by the HiMPP....2-3
Table 2-2 Error codes for the system control APIs....2-58
Table 2-3 Error codes for VB pool APIs....2-58

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

2 System Control

2.1 Overview

Based on the features of the Hi35xx, the system control module supports the following operations:

- Resets and initializes hardware components
- Initializes and deinitializes HiMPP modules
- Manages the mass physical memory and the status of HiMPP modules
- Provides version information about the current HiMPP system.

The HiMPP must be initialized before the HiMPP services are enabled by the applications. Similarly, the HiMPP must be deinitialized to release sources after the HiMPP services are disabled by applications.

NOTE

The MPP does not support calling multiple processes.

2.2 Functions

2.2.1 VB Pool

The functions of the video buffer (VB) pool are as follows:

- Provides the mass physical memory for media services
- Allocates and releasing the memory to make full use of the buffer pool
- Ensures that the physical memory resources are fully used by each media processing module.

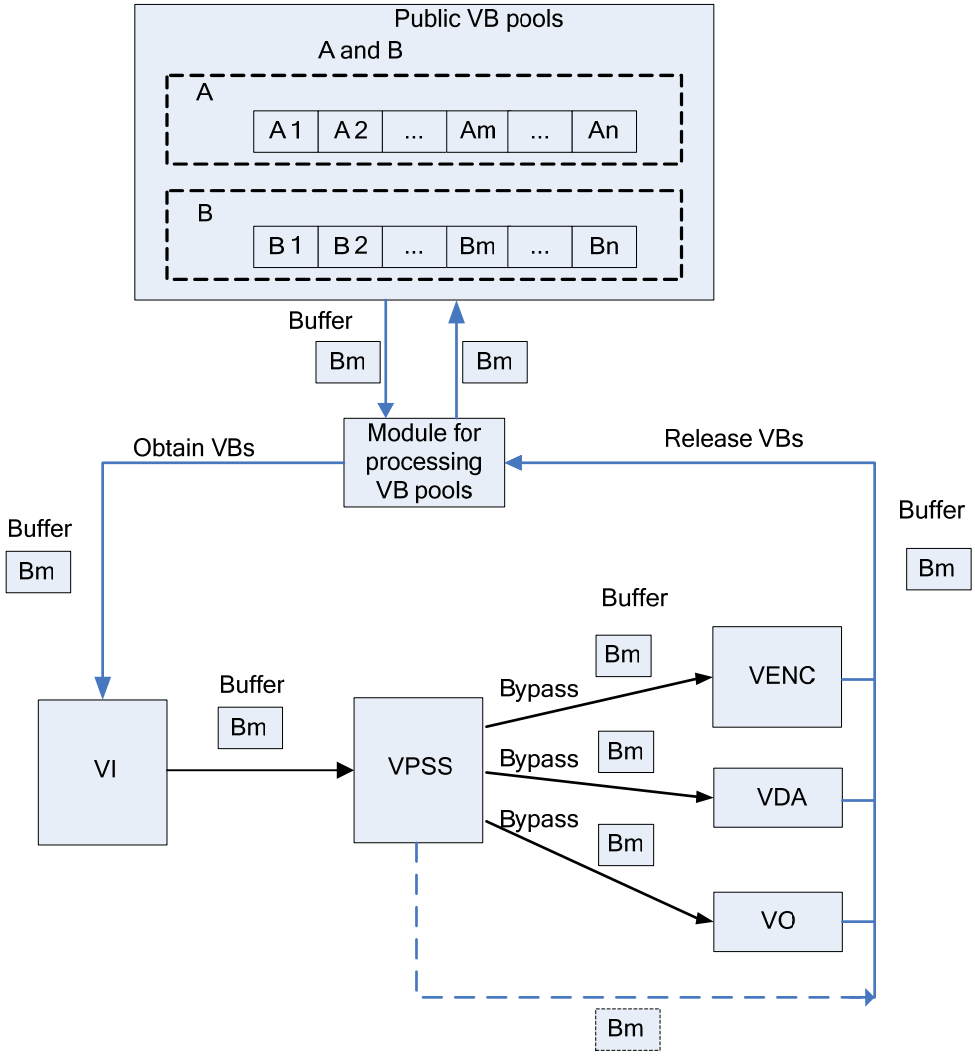
A VB pool consists of the buffers with the same size and continuous physical addresses.

For the VI channel, public VB pools are required. All VI channels can obtain VB buffers from public VB pools for storing captured pictures. For example, the VB block Bm can be obtained from the public VB pool A. Public VB pools must be configured for VI channels before the system is initialized, because VI channels do not support the functions of creating and destroying public VB pools. The number of public VB pools and the size and number of buffers vary according to services.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

The lifetime of a buffer is that the information of the buffer is transferred to other module through a bypass channel of the VPSS. See the solid line in Figure 2-1. If the buffer information is not transparently transferred to other modules through the bypass channel of the VPSS, the buffer is returned to the public VB pool after the VPSS performs operations. See the dotted line in Figure 2-1.

Figure 2-1 Typical data flows of public VB pools



2.2.2 Binding the System

The HiMPP provides a system binding MPI HI_MPI_SYS_Bind. By calling this MPI, the data receiver binds a data source to establish an association. Note that only the data receiver can bind the data source. After binding, the data generated by the data source is automatically transmitted to the data receiver. Table 2-1 shows the binding relationships supported by the HiMPP.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Table 2-1 Binding relationships supported by the HiMPP

Data Source	Data Receiver
VI	VO
	VENC
	VDA
	VPSS
VPSS	VO
	VENC
	VDA
VDEC	VPSS
	VO (SD or virtual device only)
	VDA
VO (WBC and virtual device)	VO
	VENC
	VPSS
AI	AENC
	AO
ADEC	AO

2.3 API Reference

The system control module provides the functions of initializing the HiMPP system, obtaining the version number of the HiMPP, initializing a VB pool, and creating a VB pool.

The module provides the following MPIs:

- HI_MPI_SYS_SetConf: Sets system control parameters.
- HI_MPI_SYS_GetConf: Obtains system control parameters.
- HI_MPI_SYS_Init: Initializes the HiMPP system.
- HI_MPI_SYS_Exit: Deinitializes the HiMPP system.
- HI_MPI_SYS_Bind: Binds a data source to a data receiver.
- HI_MPI_SYS_UnBind: Unbinds a data source from a data receiver.
- HI_MPI_SYS_GetBindbyDest: Obtains the information about a bound data source.
- HI_MPI_SYS_GetVersion: Obtains the version number of the HiMPP.
- HI_MPI_SYS_GetCurPts: Obtains the current presentation time stamp (PTS) of the HiMPP.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_SYS_InitPtsBase: Initializes the HiMPP PTS.
- HI_MPI_SYS_SyncPts: Synchronizes the HiMPP PTS.
- HI_MPI_SYS_Mmap: Maps the storage address.
- HI_MPI_SYS_Munmap: Unmaps the storage address.
- HI_MPI_SYS_SetReg: Sets the register values.
- HI_MPI_SYS_GetReg: Obtains the register values.
- HI_MPI_SYS_MmzAlloc: Allocates the media memory zone (MMZ) in the user state.
- HI_MPI_SYS_MmzAlloc_Cached: Allocates the MMZ supporting cache in the user state.
- HI_MPI_SYS_MmzFlushCache: Copies data from the cache to the MMZ and flushes the cache.
- HI_MPI_SYS_MmzFree: Releases the MMZ in the user state.
- HI_MPI_SYS_SetMemConf: Sets the DDR whose memory will be used.
- HI_MPI_SYS_GetMemConf: Obtains the name of the DDR whose memory is used.
- HI_MPI_SYS_CloseFd: Closes the file descriptor (FD) opened by the SYS module.
- HI_MPI_VB_SetConf: Sets the attributes of a HiMPP VB pool.
- HI_MPI_VB_GetConf: Obtains the attributes of a HiMPP VB pool.
- HI_MPI_VB_Init: Initializes a HiMPP VB pool.
- HI_MPI_VB_Exit: Deinitializes a HiMPP VB pool.
- HI_MPI_VB_CreatePool: Creates a VB pool.
- HI_MPI_VB_DestroyPool: Destroys a VB pool.
- HI_MPI_VB_GetBlock: Obtains a buffer.
- HI_MPI_VB_ReleaseBlock: Releases an obtained buffer.
- HI_MPI_VB_Handle2PhysAddr: Obtains the physical address of a buffer.
- HI_MPI_VB_Handle2PoolId: Obtains the ID of the VB pool where a buffer is located.
- HI_MPI_VB_MmapPool: Maps the user-state virtual address to a VB pool.
- HI_MPI_VB_MunmapPool: Unmaps the user-state address from a VB pool
- HI_MPI_VB_GetBlkVirAddr: Obtains the user-state virtual address of a buffer in a VB pool.
- HI_MPI_VB_GetVRegAddr: Obtains the physical address of an ISP register.

HI_MPI_SYS_SetConf

[Description]

Sets system control parameters.

[Syntax]

```
HI_S32 HI_MPI_SYS_SetConf(const MPP_SYS_CONF_S *pstSysConf);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
pstSysConf	Pointer to system control parameters. Static attribute. A static attribute can be set only before the system is initialized, the device is started, or the channel is enabled.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

This MPI can be called to configure the HiMPP system only before the entire HiMPP system is initialized. Otherwise, the HiMPP system fails to be configured.

[Example]

```
HI_S32 s32ret;
VB_CONF_S struVbConf;
MPP_SYS_CONF_S struSysConf;
memset(&struVbConf,0,sizeof(VB_CONF_S));
struVbConf.u32MaxPoolCnt              = 64;
struVbConf.astCommPool[0].u32BlkSize  = 1920*1088*2;
struVbConf.astCommPool[0].u32BlkCnt   = 15;
memset(struVbConf.astCommPool[0].acMmzName,0,sizeof(struVbConf.astCommPoo
l[0].acMmzName));
s32ret = HI_MPI_VB_SetConf(&struVbConf);
if (HI_SUCCESS != s32ret)
{
return s32ret;
}
s32ret = HI_MPI_VB_Init();
if (HI_SUCCESS != s32ret)
{
return s32ret;
}
struSysConf.u32AlignWidth = 16;
/* set config of mpp system*/
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
s32ret = HI_MPI_SYS_SetConf(&struSysConf);
if (HI_SUCCESS != s32ret)
{
printf("Set mpp sys config failed!\n");
return s32ret;
}

/* init system*/
s32ret = HI_MPI_SYS_Init();
if (HI_SUCCESS != s32ret)
{
printf("Mpi init failed!\n");
return s32ret;
}
/* ..... */

/* exit system*/
s32ret = HI_MPI_SYS_Exit();
if (HI_SUCCESS != s32ret)
{
printf("Mpi exit failed!\n");
return s32ret;
}

s32ret = HI_MPI_VB_Exit();
if (HI_SUCCESS != s32ret)
{
return s32ret;
}
```

[See Also]

HI_MPI_VB_GetConf

HI_MPI_SYS_GetConf

[Description]

Obtains system control parameters.

[Syntax]

```
HI_S32 HI_MPI_SYS_GetConf(MPP_SYS_CONF_S *pstSysConf);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
pstSysConf	Pointer to system control parameters. Static attribute.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

You can obtain system control parameters only after calling HI_MPI_VB_SetConf successfully.

[Example]

None

[See Also]

HI_MPI_SYS_SetConf

HI_MPI_SYS_Init

[Description]

Initializes the HiMPP system. Except the AENC module and ADEC module, the module including the AIU, AOU, VIU, VOU, VENC module, VDEC module, region management module, VDA module are initialized.

[Syntax]

```
HI_S32 HI_MPI_SYS_Init(HI_VOID);
```

[Parameter]

None

[Return Value]

Parameter	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

- You must call HI_MPI_SYS_SetConf before initializing the HiMPP system. Otherwise, the HiMPP system fails to be initialized.
- The running of the HiMPP system depends on the buffer pool. Therefore, you must call HI_MPI_VB_Init to initialize the buffer pool before initializing the HiMPP system.
- If the system is initialized for several times, a code indicating success is returned each time. This has no effect on the running status of the HiMPP.
- You only need to use one process to initialize the system.

[Example]

See the example of HI_MPI_SYS_SetConf.

[See Also]

HI_MPI_SYS_Exit

HI_MPI_SYS_Exit

[Description]

Deinitializes the HiMPP system. Except the AENC module and ADEC module, the modules including the AIU, AOU, VIU, VOU, VENC module, VDEC module, region management module, VDA module are destroyed or disabled.

[Syntax]

```
HI_S32 HI_MPI_SYS_Exit(HI_VOID);
```

[Parameter]

None

[Return Value]

Parameter	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

- If a process is blocked in the MPI, the HiMPP system fails to be deinitalized. The HiMPP system can be deinitialized only when the MPI is not blocked.
- This MPI can be called repeatedly and a code indicating success is returned.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- The AENC channel and ADEC channel are not destroyed after system deinitialization. Therefore, you need to destroy these channels. If the processes of creating these channels end, these channels are destroyed automatically.

[Example]

See the example of HI_MPI_SYS_SetConf.

[See Also]

HI_MPI_SYS_Init

HI_MPI_SYS_Bind

[Description]

Binds a data source and a data receiver.

[Syntax]

```
HI_S32  HI_MPI_SYS_Bind(const MPP_CHN_S *pstSrcChn, const MPP_CHN_S
*pstDestChn);
```

[Parameter]

Parameter	Description	Input/Output
pstSrcChn	Pointer to the source channel.	Input
pstDestChn	Pointer to the destination channel.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

- For details about the supported binding relationships, see Table 2-1.
- A data receiver can be bound only to one data source.
- The binding establishes an association between a data source and a data receiver. After binding, the data generated by the data source is automatically transmitted to the data receiver.
- When the VIU and VDEC module are data sources, related channels act as transmitters to transmit data to other modules. You can set the device ID to 0. Then the SDK does not check the entered device ID.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- When the VOU acts as the data source for transmitting writeback (WBC) data, the device transmits data to other modules. You can set the channel ID to 0. Then the SDK does not check the entered channel ID.
- When the VPSS and VDEC module receive data, the device (GROUP) receives the data from other modules. You can set the channel ID to 0. Then the SDK does not check the entered channel ID.
- The device ID and channel ID do not need to be specified in other cases.

[Example]

```
HI_S32 s32Ret;
MPP_CHN_S stBindSrc;
MPP_CHN_S stBindDest;

stBindDest.enModId = HI_ID_VPSS;
stBindDest.s32ChnId = 0;
stBindDest.s32DevId = 0;

stBindSrc.enModId = HI_ID_VIU;
stBindSrc.s32ChnId = 0;
stBindSrc.s32DevId = 0;

s32Ret = HI_MPI_SYS_Bind(&stBindSrc, &stBindDest);
if (s32Ret)
{
   return HI_FAILURE;
}
```

[See Also]

HI_MPI_SYS_UnBind

HI_MPI_SYS_UnBind

[Description]

Unbind a data source from a data receiver.

[Syntax]

```
HI_S32  HI_MPI_SYS_UnBind(const MPP_CHN_S *pstSrcChn, const MPP_CHN_S
*pstDestChn);
```

[Parameter]

Parameter	Description	Input/Output
pstSrcChn	Pointer to the source channel.	Input
pstDestChn	Pointer to the destination channel.	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

None

[Example]

See the related sample.

[See Also]

HI_MPI_SYS_Bind

HI_MPI_SYS_GetBindbyDest

[Description]

Obtains the information about a bound data source.

[Syntax]

```
HI_S32  HI_MPI_SYS_GetBindbyDest(const MPP_CHN_S *pstDestChn, MPP_CHN_S
*pstSrcChn);
```

[Parameter]

Parameter	Description	Input/Output
pstSrcChn	Pointer to the source channel.	Output
pstDestChn	Pointer to the destination channel.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

None

[Example]

See the related sample.

[See Also]

- HI_MPI_SYS_Bind
- HI_MPI_SYS_UnBind

HI_MPI_SYS_GetVersion

[Description]

Obtains the version number of the HiMPP.

[Syntax]

```
HI_S32 HI_MPI_SYS_GetVersion(MPP_VERSION_S *pstVersion);
```

[Parameter]

Parameter	Description	Input/Output
pstVersion	Pointer to the version number. Dynamic attribute. A dynamic attribute can be set at any time.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

None

[Example]

```
HI_S32 s32ret;
MPP_VERSION_S stVersion;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
s32ret = HI_MPI_SYS_GetVersion(&stVersion);
if (HI_SUCCESS != s32ret)
{
return s32ret;
}
printf("mpi version is %s\n", stVersion.aVersion);
```

[See Also]

None

HI_MPI_SYS_GetCurPts

[Description]

Obtains the current PTS of the HiMPP.

[Syntax]

```
HI_S32 HI_MPI_SYS_GetCurPts(HI_U64 *pu64CurPts);
```

[Parameter]

Parameter	Description	Input/Output
pu64CurPts	Pointer to the current PTS.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_SYS_InitPtsBase

[Description]

Initializes the HiMPP PTS.

[Syntax]

```
HI_S32 HI_MPI_SYS_InitPtsBase(HI_U64 u64PtsBase);
```

[Parameter]

Parameter	Description	Input/Output
u64PtsBase	PTS base.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

Regardless of the original system PTS, initializing the PTS base forces the current system PTS to u64PtsBase. Therefore, you are recommended to call this MPI before a media service is enabled. For example, you can call this MPI immediately when the OS starts. If a media service is enabled, you can call HI_MPI_SYS_SyncPts to tune the PTS.

[Example]

None

[See Also]

None

HI_MPI_SYS_SyncPts

[Description]

Synchronizes the HiMPP PTS.

[Syntax]

```
HI_S32 HI_MPI_SYS_SyncPts(HI_U64 u64PtsBase);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
u64PtsBase	PTS base.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

After the current system PTS is tuned, the PTS does not roll back. When multiple Hi3531 chips are synchronized, the difference between the clock sources of the boards may be significant. Therefore, you are recommended to tune the PTS once a second.

[Example]

None

[See Also]

None

HI_MPI_SYS_Mmap

[Description]

Maps the storage address.

[Syntax]

```
HI_VOID * HI_MPI_SYS_Mmap(HI_U32 u32PhyAddr, HI_U32 u32Size);
```

[Parameter]

Parameter	Description	Input/Output
u32PhyAddr	Start address of the memory to be mapped.	Input
u32Size	Number of mapped bytes.	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Invalid address.
Other values	Valid address.

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

- This MPI is equivalent to the mmap function in Linux.
- The address entered must be a valid physical address.

[Example]

None

[See Also]

HI_MPI_SYS_Munmap

HI_MPI_SYS_Munmap

[Description]

Unmaps the storage address.

[Syntax]

```
HI_S32 HI_MPI_SYS_Munmap(HI_VOID* pVirAddr, HI_U32 u32Size);
```

[Parameter]

Parameter	Description	Input/Output
pVirAddr	Address returned after mmap is called.	Input
u32Size	Length of mapped area, in bytes.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

This MPI is equivalent to the munmap function in Linux.

[Example]

None

[See Also]

HI_MPI_SYS_Mmap

HI_MPI_SYS_SetReg

[Description]

Sets register values.

[Syntax]

```
HI_S32 HI_MPI_SYS_SetReg(HI_U32 u32Addr, HI_U32 u32Value)
```

[Parameter]

Parameter	Description	Input/Output
u32Addr	Written address.	Input
u32Value	Written value.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

The address entered must be a valid physical address.

[Example]

None

[See Also]

HI_MPI_SYS_GetReg

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_SYS_GetReg

[Description]

Obtains register values.

[Syntax]

```
HI_S32 HI_MPI_SYS_GetReg(HI_U32 u32Addr, HI_U32 *pu32Value)
```

[Parameter]

Parameter	Description	Input/Output
u32Addr	Physical address.	Input
pu32Value	Value of a memory address.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

The address entered must be a valid physical address.

[Example]

None

[See Also]

HI_MPI_SYS_SetReg

HI_MPI_SYS_MmzAlloc

[Description]

Allocates the MMZ in the user state.

[Syntax]

```
HI_S32 HI_MPI_SYS_MmzAlloc(HI_U32 *pu32PhyAddr, void **ppVitAddr,const
HI_CHAR *pstrMmb, const HI_CHAR *pstrZone, HI_U32 u32Len);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
pu32PhyAddr	Pointer to the physical address.	Output
ppVirtAddr	Pointer to the virtual address.	Output
pstrMmb	String pointer to the media memory block (MMB) name.	Input
pstrZone	String pointer to the MMZ zone name.	Input
u32Len	Buffer size.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

The MMZ consists of multiple zones and each zone has multiple MMBs. You can call this MPI to allocate a memory block *pstrMmb with the size of u32Len in the *pstrZone zone of the MMZ. In this case, the pointers that point to the physical address and user-state virtual address are returned. If there is the anonymous zone in the MMZ, set *pstrZone to null. If *pstrMmb is set to null, the created MMB is named <null>.

[Example]

None

[See Also]

None

HI_MPI_SYS_MmzAlloc_Cached

[Description]

Allocates the MMZ supporting cache in the user state.

[Syntax]

```
HI_S32 HI_MPI_SYS_MmzAlloc_Cached(HI_U32 *pu32PhyAddr, HI_VOID
**ppVitAddr, const HI_CHAR *pstrMmb, const HI_CHAR *pstrZone, HI_U32
u32Len);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
pu32PhyAddr	Pointer to the physical address.	Output
ppVirtAddr	Pointer to the virtual address.	Output
pstrMmb	String pointer to the MMB name.	Input
pstrZone	String pointer to the MMZ name.	Input
u32Len	Buffer size.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

- The differences between HI_MPI_SYS_MmzAlloc_Cached and HI_MPI_SYS_MmzAlloc are as follows: The memory that is allocated by calling HI_MPI_SYS_MmzAlloc_Cached supports cache. HI_MPI_SYS_MmzAlloc_Cached is recommended if the memory to be allocated will be frequently used. This improves the CPU read/write efficiency and system performance. For example, when intelligent video engine (IVE) operators are being used, a large amount of data is frequently read and written. If you allocate the memory by calling HI_MPI_SYS_MmzAlloc_Cached, the working efficiency of the CPU is improved.
- When the CPU accesses the memory that is allocated by calling HI_MPI_SYS_MmzAlloc_Cached, the data in the memory will be stored in the cache. The hardware devices such as the IVE can access only the physical memory rather than the cache. In this case, HI_MPI_SYS_MmzFlushCache needs to be called to synchronize data.

[Note]

Note

[See Also]

- HI_MPI_SYS_MmzAlloc
- HI_MPI_SYS_MmzFlushCache

HI_MPI_SYS_MmzFlushCache

[Description]

Copies data from the cache to the MMZ and flushes the cache.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_SYS_MmzFlushCache(HI_U32 u32PhyAddr, HI_VOID *pVitAddr,
HI_U32 u32Size);
```

[Parameter]

Parameter	Description	Input/Output
u32PhyAddr	Start physical address for storing the data to be used.	Input
pVirtAddr	Pointer to the start virtual address for storing the data to be used.	Input
u32Size	Amount of the data to be used.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

- If the data in the cache is the latest data, you need to call this MPI to synchronize the data to the memory. This ensures that the hardware such as the IVE that cannot directly access the cache can obtain correct data.
- This MPI must be called if HI_MPI_SYS_MmzAlloc_Cached is called.
- If u32PhyAddr is set to 0, the entire cache is operated.

[Example]

Note

[See Also]

HI_MPI_SYS_MmzAlloc_Cached

HI_MPI_SYS_MmzFree

[Description]

Releases the MMZ in the user state.

[Syntax]

```
HI_S32 HI_MPI_SYS_MmzFree(HI_U32 u32PhyAddr, HI_VOID *pVirtAddr);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
u32PhyAddr	Physical address.	Input
pVirtAddr	Pointer to the virtual address.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

The entered address must be a valid physical address. The pointer that points to the virtual address can be set to null.

[Example]

None

[See Also]

HI_MPI_SYS_MmzAlloc

HI_MPI_SYS_SetMemConf

[Description]

Sets the DDR whose memory is used by the device channel of a module.

[Syntax]

```
HI_S32 HI_MPI_SYS_SetMemConf(MPP_CHN_S *pstMppChn,const HI_CHAR
*pcMmzName);
```

[Parameter]

Parameter	Description	Input/Output
pstMppChn	Information about a device channel of a module.	Input
pcMmzName	Name of the DDR where a memory is located.	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

- The DDR name must be the name of an existing DDR. Assume that there are two DDRs. One is unnamed and the other is ddr1. In this case, pcMmzName must be set to null or ddr1. This MPI is used to evenly allocate the memory.
- When the memory is evenly allocated for the VOU, ensure that the memories of the channels of the same device are from the same DDR. Otherwise, the SDK allocates memory from the DDR configured for channel 0 of this device by default.
- This MPI must be called before services run. That is, after the sys and VB are initialized, call this MPI if necessary and then configure other modules.

[Example]

```
HI_S32  i, s32Ret;
HI_CHAR   *pcMmzName;
MPP_CHN_S stMppChnVI;
/*Evenly allocate memory for 16D1 channels*/
    for(i=0;i<16;i++)
    {
        stMppChnVI.enModId = HI_ID_VIU;
        stMppChnVI.s32DevId = 0;
        stMppChnVI.s32ChnId = i;

        if(0 == (i%2))
        {
            pcMmzName = NULL;
        }
        else
        {
            pcMmzName = "ddr1";
        }

        s32Ret = HI_MPI_SYS_SetMemConf(&stMppChnVI,pcMmzName);
        if (HI_SUCCESS != s32ret)
        {
            printf("SetMemConf ERR !\n");
            return s32ret;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
        }
    }
```

[See Also]

HI_MPI_SYS_GetMemConf

HI_MPI_SYS_GetMemConf

[Description]

Obtains the name of the DDR whose memory is used by the device channel.

[Syntax]

```
HI_S32 HI_MPI_SYS_GetMemConf(MPP_CHN_S *pstMppChn, HI_CHAR *pcMmzName);
```

[Parameter]

Parameter	Description	Input/Output
pstMppChn	Information about a device channel of a module.	Input
pcMmzName	Name of the DDR where a memory is located.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_SYS_SetMemConf

HI_MPI_SYS_CloseFd

[Description]

Closes all the logs, system, and memory FDs that are opened by the SYS module.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_SYS_CloseFd(HI_VOID);
```

[Parameter]

None

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h, mpi_sys.h
- Library file: libmpi.a

[Note]

Before calling this MPI, ensure that all modules are disabled.

[Example]

None

[See Also]

None

HI_MPI_VB_SetConf

[Description]

Sets the attributes of a HiMPP VB pool.

[Syntax]

```
HI_S32 HI_MPI_VB_SetConf (const VB_CONF_S *pstVbConf);
```

[Parameter]

Parameter	Description	Input/Output
pstVbConf	Pointer to the attributes of a VB pool. Static attribute.	Input

[Return Value]

Return Value	Description
0	Success.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vb.h, mpi_vb.h
- Library file: libmpi.a

[Note]

- The attributes of a VB pool can be set only before the system is initialized. Otherwise, an error code indicating failure is returned.
- The video buffer must be configured based on the application scenario. For details about configuration rules, see section 2.2.1 "VB Pool."

[Example]

```
HI_S32 s32ret;
VB_CONF_S stVbConf;
memset(&stVbConf,0,sizeof(VB_CONF_S));
stVbConf.u32MaxPoolCnt = 128;
stVbConf.astCommPool[0].u32BlkSize = 768*576*2;
stVbConf.astCommPool[0].u32BlkCnt = 20;
stVbConf.astCommPool[1].u32BlkSize = 384*288*2;
stVbConf.astCommPool[1].u32BlkCnt = 40;
s32ret = HI_MPI_VB_SetConf(&stVbConf);
if (HI_SUCCESS != s32ret)
{
printf("set vb err:0x%x\n", s32ret);
return s32ret;
}

s32ret = HI_MPI_VB_Init();
if (HI_SUCCESS != s32ret)
{
    printf("init vb err:0x%x\n", s32ret);
    return s32ret;
}
/* ... ... */
(void)HI_MPI_VB_Exit();
```

[See Also]

HI_MPI_VB_GetConf

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VB_GetConf

[Description]

Obtains the attributes of a HiMPP VB pool.

[Syntax]

```
HI_S32 HI_MPI_VB_GetConf (VB_CONF_S *pstVbConf);
```

[Parameter]

Parameter	Description	Input/Output
pstVbConf	Pointer to the attributes of a VB pool. Static attribute.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vb.h, mpi_vb.h
- Library file: libmpi.a

[Note]

Before obtaining the attributes of a HiMPP VB pool, you must call HI_MPI_VB_SetConf to set the attributes of the VB pool.

[Example]

None

[See Also]

HI_MPI_VB_SetConf

HI_MPI_VB_Init

[Description]

Initializes a HiMPP VB pool.

[Syntax]

```
HI_S32 HI_MPI_VB_Init (HI_VOID);
```

[Parameter]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vb.h, mpi_vb.h
- Library file: libmpi.a

[Note]

- Before initializing a VB pool, you must call HI_MPI_VB_SetConf to set the attributes of the VB pool. Otherwise, the VB pool fails to be initialized.
- This MPI can be called repeatedly, and a code indicating success is returned.

[Example]

See the example of HI_MPI_VB_SetConf.

[See Also]

HI_MPI_VB_Exit

HI_MPI_VB_Exit

[Description]

Deinitializes a HiMPP VB pool.

[Syntax]

```
HI_S32 HI_MPI_VB_Exit (HI_VOID);
```

[Parameter]

None

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vb.h, mpi_vb.h
- Library file: libmpi.a

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Before deinitializing a HiMPP VB pool, you must call HI_MPI_SYS_Exit to deinitialize the HiMPP system. Otherwise, the VB pool fails to be deinitialized.
- This MPI can be called repeatedly, and a code indicating success is returned.
- After the VB pool is deinitialized, the configurations of the VB pool are retained.

[Example]

See the example of HI_MPI_VB_SetConf.

[See Also]

HI_MPI_VB_Init

HI_MPI_VB_CreatePool

[Description]

Creates a VB pool.

[Syntax]

```
VB_POOL HI_MPI_VB_CreatePool(HI_U32 u32BlkSize,HI_U32 u32BlkCnt, const
HI_CHAR *pcMmzName);
```

[Parameter]

Parameter	Description	Input/Output
u32BlkSize	Size of each buffer in a VB pool. Value range: (0, 2^{32}), in byte	Input
u32BlkCnt	Number of buffers in a VB pool. Value range: (0, 2^{32})	Input
pcMmzName	Name of the DDR where a VB pool is located.	Input

[Return Value]

Return Value	Description
Other values	The ID of a VB pool is valid.
VB_INVALID_POOLID	A VB pool fails to be created, because the parameter is invalid or the reserved memory is insufficient.

[Requirement]

- Header files: hi_comm_vb.h, mpi_vb.h
- Library file: libmpi.a

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- A VB pool is an area of the reserved memory. It consists of several buffers with the same size. If the size of a VB pool to be created is larger than the reserved memory, the VB pool fails to be created.
- Ensure that the name of an existing DDR is entered. If the DDR does not exist, no memory is allocated.

[Example]

```
VB_POOL VbPool;
VB_BLK VbBlk;
HI_U32 u32BlkSize = 768*576*2;
HI_U32 u32BlkCnt  = 15;
HI_U32 u32Addr;
/* create a video buffer pool*/
VbPool = HI_MPI_VB_CreatePool(u32BlkSize,u32BlkCnt,”ddr1”);
if ( VB_INVALID_POOLID == VbPool )
{
printf("create vb err\n");
return HI_FAILURE;
}

/* get a buffer from pool*/
VbBlk = HI_MPI_VB_GetBlock(VbPool, u32BlkSize,”ddr1”);
if (VB_INVALID_HANDLE == VbBlk )
{
printf("get vb block err\n");
(void)HI_MPI_VB_DestroyPool(VbPool);
return HI_FAILURE;
}

/* get the physical address of buffer*/
u32Addr = HI_MPI_VB_Handle2PhysAddr(VbBlk);
if (HI_NULL_PTR == u32Addr)
{
printf("blk to physaddr err\n");
(void)HI_MPI_VB_ReleaseBlock(VbBlk);
(void)HI_MPI_VB_DestroyPool(VbPool);
return HI_FAILURE;
}

/* use this address do something*/

/* then release the buffer*/
(void)HI_MPI_VB_ReleaseBlock(VbBlk);

/* destroy video buffer pool */
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
(void)HI_MPI_VB_DestroyPool(VbPool);
```

[See Also]

HI_MPI_VB_DestroyPool

HI_MPI_VB_DestroyPool

[Description]

Destroys a VB pool.

[Syntax]

```
HI_S32 HI_MPI_VB_DestroyPool(VB_POOL Pool);
```

[Parameter]

Parameter	Description	Input/Output
Pool	ID of a VB pool. Value range: [0, VB_MAX_POOLS)	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vb.h, mpi_vb.h
- Library file: libmpi.a

[Note]

- If the VB pool to be destroyed does not exist, HI_ERR_VB_UNEXIST is returned.
- When the HiMPP system is deinitialized, all buffer pools including user-state buffer pools are destroyed.

[Example]

See the example of HI_MPI_VB_CreatePool.

[See Also]

HI_MPI_VB_CreatePool

HI_MPI_VB_GetBlock

[Description]

Obtains a buffer.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
VB_BLK HI_MPI_VB_GetBlock(VB_POOL Pool, HI_U32 u32BlkSize ,const HI_CHAR
*pcMmzName);
```

[Parameter]

Parameter	Description	Input/Output
Pool	ID of a VB pool. Value range: [0, VB_MAX_POOLS)	Input
u32BlkSize	Size of a buffer. Value Range: (0, 2^{32}), in byte	Input
pcMmzName	Name of the DDR where a VB pool is located.	Input

[Return Value]

Return Value	Description
Other values	The buffer handle is valid.
VB_INVALID_HANDLE	A buffer fails to be obtained.

[Error Code]

None

[Requirement]

- Header files: hi_comm_vb.h, mpi_vb.h
- Library file: libmpi.a

[Note]

- After creating a VB pool, you can call this MPI to obtain a buffer from the VB pool. That is, you can set the first parameter Pool to the ID of the created VB pool, and left the second parameter u32BlkSize blank. In this way, the size of the obtained buffer is equal to the size of the buffer that is specified when you create the VB pool. The pcMmzName parameter is invalid when a buffer is obtained from a specified VB pool.
- If you want to obtain a buffer with a specified size from a public VB pool, you can set the first parameter Pool to an invalid ID (VB_INVALID_POOLID), set the second parameter u32BlkSize to a required size of the buffer, and specify the DDR from whose public VB pools that buffers are obtained. If the specified DDR does not have public VB pool, no buffer is obtained.
- The public VB pool is mainly used for storing the captured pictures of the VIU. Therefore, misoperations performed on the public VB pool (for example, occupying too many buffers) may affect the running of the HiMPP system.

[Example]

See the example of HI_MPI_VB_CreatePool.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

HI_MPI_VB_ReleaseBlock

HI_MPI_VB_ReleaseBlock

[Description]

Releases an obtained buffer.

[Syntax]

```
HI_S32 HI_MPI_VB_ReleaseBlock(VB_BLK Block);
```

[Parameter]

Parameter	Description	Input/Output
Block	Buffer handle.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vb.h, mpi_vb.h
- Library file: libmpi.a

[Note]

If no buffer is available, call this MPI to release buffers.

[Example]

See the example of HI_MPI_VB_CreatePool.

[See Also]

HI_MPI_VB_GetBlock

HI_MPI_VB_Handle2PhysAddr

[Description]

Obtains the physical address of a buffer.

[Syntax]

```
HI_U32 HI_MPI_VB_Handle2PhysAddr(VB_BLK Block);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
Block	Buffer handle.	Input

[Return Value]

Return Value	Description
0	Invalid value, indicating that the buffer handle is invalid.
Other values	The physical address is valid.

[Error Code]

None

[Requirement]

- Header files: hi_comm_vb.h, mpi_vb.h
- Library file: libmpi.a

[Note]

The specified buffer must be a valid buffer obtained from a HiMPP VB pool.

[Example]

See the example of HI_MPI_VB_CreatePool.

[See Also]

None

HI_MPI_VB_Handle2PoolId

[Description]

Obtains the ID of the buffer pool where a frame buffer is located.

[Syntax]

```
VB_POOL HI_MPI_VB_Handle2PoolId (VB_BLK Block);
```

[Parameter]

Parameter	Description	Input/Output
Block	Buffer handle.	Input

[Return Value]

Return Value	Description
Positive number or zero	The ID of a VB pool is valid.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Negative number	The ID of a VB pool is invalid.

[Error Code]

None

[Requirement]

- Header files: hi_comm_vb.h, mpi_vb.h
- Library file: libmpi.a

[Note]

The specified buffer must be a valid buffer obtained from a HiMPP VB pool.

[Example]

```
VB_POOL VbPool;
VB_BLK VbBlk;/* get vb blk id from somewhere*/
/* get pool id */
VbPool = HI_MPI_VB_Handle2PoolId(VbBlk);
if ( VB_INVALID_POOLID != VbPool )
{
printf("pool id is %d\n", VbPool);
/* use pool id do something*/
}
```

[See Also]

None

HI_MPI_VB_MmapPool

[Description]

Maps the user-state virtual address to a VB pool.

[Syntax]

```
HI_S32 HI_MPI_VB_MmapPool(VB_POOL Pool);
```

[Parameter]

Parameter	Description	Input/Output
Pool	ID of a VB pool. Value range: [0, VB_MAX_POOLS)	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vb.h, mpi_vb.h
- Library file: libmpi.a

[Note]

- The ID of the buffer pool must be valid.
- A code indicating success is returned if you call this MPI repeatedly.

[Example]

None

[See Also]

- HI_MPI_VB_MunmapPool
- HI_MPI_VB_GetBlkVirAddr

HI_MPI_VB_MunmapPool

[Description]

Unmaps the user-state address from a VB pool

[Syntax]

```
HI_S32 HI_MPI_VB_MunmapPool(VB_POOL Pool);
```

[Parameter]

Parameter	Description	Input/Output
Pool	ID of a VB pool. Value range: [0, VB_MAX_POOLS)	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vb.h, mpi_vb.h

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Library file: libmpi.a

[Note]

- The ID of the buffer pool must be valid.
- The VB pool must be mapped.
- The buffers in the VB pool are not used by the MPI layer. If the buffers are used by the MPI layer, the system considers that the mapped user-state virtual address is used, and a code indicating failure is returned.

[Example]

None

[See Also]

- HI_MPI_VB_MmapPool
- HI_MPI_VB_GetBlkVirAddr

HI_MPI_VB_GetBlkVirAddr

[Description]

Obtains the user-state virtual address of a buffer in a VB pool.

[Syntax]

```
HI_S32 HI_MPI_VB_GetBlkVirAddr(VB_POOL Pool, HI_U32 u32PhyAddr, HI_VOID
**ppVirAddr);
```

[Parameter]

Parameter	Description	Input/Output
Pool	ID of a VB pool. Value range: [0, VB_MAX_POOLS)	Input
u32PhyAddr	Physical address of a buffer.	Input
ppVirAddr	User-state virtual address.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vb.h, mpi_vb.h
- Library file: libmpi.a

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- The ID of the buffer pool and the physical address of the buffer must be valid.
- HI_MPI_VB_MmapPool must be called.

[Example]

None

[See Also]

- HI_MPI_VB_MmapPool
- HI_MPI_VB_MunmapPool

HI_MPI_VB_GetVRegAddr

[Description]

Obtains the physical address of an ISP register.

[Syntax]

```
HI_S32 HI_MPI_SYS_GetVRegAddr(HI_U32 *pu32Addr);
```

[Parameter]

Parameter	Description	Input/Output
pu32Addr	Pointer to the physical address.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 2.5 "Error Codes."

[Requirement]

- Header files: hi_comm_sys.h、mpi_sys.h
- Library file: libmpi.a

[Note]

The MPI is used only in the SDK.

[Example]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

2.4 Data Type

2.4.1 Basic Data Types

The basic data types are defined as follows:

Common Data Types

```
typedef unsigned char           HI_U8;
typedef unsigned short          HI_U16;
typedef unsigned int            HI_U32;
typedef signed char             HI_S8;
typedef short                   HI_S16;
typedef int                     HI_S32;
#ifndef _M_IX86
   typedef unsigned long long  HI_U64;
   typedef long long            HI_S64;
#else
   typedef __int64              HI_U64;
   typedef __int64              HI_S64;
#endif

typedef char                    HI_CHAR;
#define HI_VOID                 void

/*----------------------------------------------*
 * const definition                             *
 *----------------------------------------------*/
typedef enum {
   HI_FALSE = 0,
   HI_TRUE  = 1,
} HI_BOOL;

#ifndef NULL
   #define NULL    0L
#endif

#define HI_NULL     0L
#define HI_SUCCESS  0
#define HI_FAILURE  (-1)

typedef HI_S32 AI_CHN;
typedef HI_S32 AO_CHN;
typedef HI_S32 AENC_CHN;
typedef HI_S32 ADEC_CHN;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
typedef HI_S32 AUDIO_DEV;
typedef HI_S32 AVENC_CHN;
typedef HI_S32 VI_DEV;
typedef HI_S32 VI_WAY;
typedef HI_S32 VI_CHN;
typedef HI_S32 VO_DEV;
typedef HI_S32 VO_CHN;
typedef HI_S32 VENC_CHN;
typedef HI_S32 VDEC_CHN;
typedef HI_S32 VENC_GRP;
typedef HI_S32 VO_GRP;
typedef HI_S32 VDA_CHN;

/*Invalid channel ID, invalid way ID, invalid device ID, and invalid
handle ID*/
#define HI_INVALID_CHN (-1)
#define HI_INVALID_WAY (-1)
#define HI_INVALID_DEV (-1)
#define HI_INVALID_HANDLE (-1)

/*Maximum number of VB pools*/
#define VB_MAX_POOLS               256

/*Maximum number of public VB pools*/
#define VB_MAX_COMM_POOLS          16
/*Length of the MMZ name*/
#define MAX_MMZ_NAME_LEN           16
```

Basic Data Types for the Hi3531

```
/*Maximum number of VIU devices*/
#define VIU_MAX_DEV_NUM            8

/*Maximum number of channels supported by each VI device*/
#define VIU_MAX_CHN_NUM_PER_DEV    4

/*Maximum number of VI channels including software channels and hardware
channels*/
#define VIU_MAX_CHN_NUM            34
#define VIU_MAX_PHYCHN_NUM         32

/*Maximum number of VO devices*/
#define VO_MAX_DEV_NUM             14
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
/*Maximum number of VO video layers*/
#define VO_MAX_LAYER_NUM              14

/*Maximum number of VO channels*/
#define VO_MAX_CHN_NUM            64

/*Maximum number of video layers supported by a VO device*/
#define VO_MAX_LAYER_IN_DEV    2

/*Maximum number of cascaded VO devices*/
#define VO_MAX_CAS_DEV_NUM     2

/*Macro for defining VO cascade device 0. The device ID is 8.*/
#define VO_CAS_DEV_1           8
/*Macro for defining VO cascade device 2. The device ID is 9.*/
#define VO_CAS_DEV_2           9

/*Maximum number of VO cascade patterns*/
#define VO_CAS_MAX_PAT        128

/*Maximum number of regions in 32-region VO cascade mode*/
#define VO_CAS_MAX_POS_32RGN    32
/*Maximum number of regions in 64-region VO cascade mode*/
#define VO_CAS_MAX_POS_64RGN    64

/*Maximum number of virtual VO devices*/
#define VO_MAX_VIRT_DEV_NUM    4

/*Macro for defining virtual VO device 0. The device ID is 10.*/
#define VO_VIRT_DEV_0         10
/*Macro for defining virtual VO device 1. The device ID is 11.*/
#define VO_VIRT_DEV_1         11
/*Macro for defining virtual VO device 2. The device ID is 12.*/
#define VO_VIRT_DEV_2         12
/*Macro for defining virtual VO device 3. The device ID is 13.*/
#define VO_VIRT_DEV_3         13

/*Maximum number of VO synchronous groups*/
#define VO_SYNC_MAX_GRP           16

/*Maximum number of channels in a VO synchronous group*/
#define VO_SYNC_MAX_CHN           32

/*Minimum number of VO display buffers*/
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
#define VO_MIN_DISP_BUF           5

/*Maximum number of VO display buffers*/
#define VO_MAX_DISP_BUF           15

/*Minimum number of virtual VO device buffers*/
#define VO_MIN_VIRT_BUF           3

/*Maximum number of virtual VO buffers*/
#define VO_MAX_VIRT_BUF           15

/*Maximum number of VENC groups*/
#define VENC_MAX_GRP_NUM          64

/*Maximum number of VENC channels*/
#define VENC_MAX_CHN_NUM          64

/*Maximum number of VDEC channels*/
#define VDEC_MAX_CHN_NUM          32

/* Number of SIO devices*/
#define SIO_MAX_NUM               6

/*Maximum number of AIO channels*/
#define AIO_MAX_CHN_NUM           16

/*Maximum number of AENC channels*/
#define AENC_MAX_CHN_NUM          32

/*Maximum number of ADEC channels*/
#define ADEC_MAX_CHN_NUM          32

/*Maximum number of audio frame buffers*/
#define MAX_AUDIO_FRAME_NUM       50
```

Basic Data Types for the Hi3521

```
/*Maximum number of VIU devices*/
#define VIU_MAX_DEV_NUM           4

/*Maximum number of channels supported by each VI device*/
#define VIU_MAX_CHN_NUM_PER_DEV   4

/*Maximum number of VI channels including software channels and hardware
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
channels*/
#define VIU_MAX_PHYCHN_NUM        16

/*Maximum number of VO devices*/
#define VO_MAX_DEV_NUM            7

/*Maximum number of VO video layers*/
#define VO_MAX_LAYER_NUM          7

/*Maximum number of VO channels*/
#define VO_MAX_CHN_NUM            32

/*Maximum number of video layers supported by a VO device*/
#define VO_MAX_LAYER_IN_DEV    2

/*Maximum number of virtual VO devices*/
#define VO_MAX_VIRT_DEV_NUM    4

/*Macro for defining virtual VO device 0. The device ID is 3.*/
#define VO_VIRT_DEV_0         3
/*Macro for defining virtual VO device 1. The device ID is 4.*/
#define VO_VIRT_DEV_1         4
/*Macro for defining virtual VO device 2. The device ID is 5.*/
#define VO_VIRT_DEV_2         5
/*Macro for defining virtual VO device 3. The device ID is 6.*/
#define VO_VIRT_DEV_3         6

/*Maximum number of VENC groups*/
#define VENC_MAX_GRP_NUM          64

/*Maximum number of VENC channels*/
#define VENC_MAX_CHN_NUM          64

/*Maximum number of VDEC channels*/
#define VDEC_MAX_CHN_NUM          64

/* Number of SIO devices*/
#define SIO_MAX_NUM               4

/*Maximum number of AIO channels*/
#define AIO_MAX_CHN_NUM           16

/*Maximum number of AENC channels*/
#define AENC_MAX_CHN_NUM          32
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
/*Maximum number of ADEC channels*/
#define ADEC_MAX_CHN_NUM          32

/*Maximum number of audio frame buffers*/
#define MAX_AUDIO_FRAME_NUM       50
```

Besides the preceding data types, the following basic data types are provided:

- POINT_S: Defines the coordinate information.
- SIZE_S: Defines the dimension information.
- RECT_S: Defines the rectangle information.
- MOD_ID_E: Defines the module ID enumeration.
- MPP_CHN_S: Defines the module, device, and channel.
- PAYLOAD_TYPE_E: Defines the audio or video payload type.

POINT_S

[Description]

Defines the coordinate information.

[Syntax]

```
typedef struct hiPOINT_S
{
  HI_S32    s32X;
  HI_S32    s32Y;
}POINT_S;
```

[Member]

Member	Description
s32X	Horizontal coordinate.
s32Y	Vertical coordinate.

[Note]

None

[See Also]

REGION_CTRL_PARAM_U

SIZE_S

[Description]

Defines the dimension information.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
typedef struct hiSIZE_S
{
    HI_U32 u32Width;
    HI_U32 u32Height;
} SIZE_S;
```

[Member]

Member	Description
u32Width	Width.
u32Height	Height.

[Note]

None

[See Also]

None

RECT_S

[Description]

Defines the rectangle information.

[Syntax]

```
typedef struct hiRECT_S
{
   HI_S32   s32X;
   HI_S32   s32Y;
   HI_U32   u32Width;
   HI_U32   u32Height;
}RECT_S;
```

[Member]

Member	Description
s32X	Horizontal coordinate.
s32Y	Vertical coordinate.
u32Width	Width.
u32Height	Height.

[Note]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

- REGION_ATTR_S
- OVERLAY_ATTR_S
- VI_CHM_ATTR_S
- VI_PUB_ATTR_S
- VO_CHN_ATTR_S

MOD_ID_E

[Description]

Defines the module ID enumeration.

[Syntax]

```
typedef enum hiMOD_ID_E
{
    HI_ID_CMPI    = 0,
    HI_ID_VB      = 1,
    HI_ID_SYS     = 2,
    HI_ID_VALG    = 3,

    HI_ID_CHNL    = 4,
    HI_ID_GROUP   = 5,
    HI_ID_VENC    = 6,
    HI_ID_VPSS     = 7,

    HI_ID_VDA      = 8,

    HI_ID_H264E   = 9,
    HI_ID_JPEGE   = 10,
    HI_ID_MPEG4E  = 11,

    HI_ID_VDEC    = 12,
    HI_ID_H264D   = 13,
    HI_ID_JPEGD   = 14,
    HI_ID_VOU     = 15,

    HI_ID_VIU     = 16,
    HI_ID_DSU     = 17,
HI_ID_RGN     = 18,
HI_ID_RC      = 19,

    HI_ID_SIO     = 20,
    HI_ID_AI      = 21,
    HI_ID_AO      = 22,
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_ID_AENC    = 23,
    HI_ID_ADEC    = 24,

    HI_ID_AVENC   = 25,

    HI_ID_PCIV    = 26,
    HI_ID_PCIVFMW = 27,

    /* there is a hole */

    HI_ID_DCCM    = 31,
    HI_ID_DCCS    = 32,

    HI_ID_PROC    = 33,
    HI_ID_LOG     = 34,
    HI_ID_MST_LOG = 35,
    HI_ID_VD      = 36,

/* there is a hole */

HI_ID_VCMP    = 38,
HI_ID_FB      = 39,

    HI_ID_USR     = 40,

    HI_ID_BUTT,
} MOD_ID_E;
```

[Member]

None

[Note]

None

[See Also]

MPP_CHN_S

MPP_CHN_S

[Description]

Defines the module, device, and channel.

[Syntax]

```
typedef struct hiMPP_CHN_S
{
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    MOD_ID_E    enModId;
    HI_S32      s32DevId;
    HI_S32      s32ChnId;
} MPP_CHN_S;
```

[Member]

Member	Description
enModId	Module ID.
s32DevId	Device ID.
s32ChnId	Channel ID.

[Note]

None

[See Also]

- HI_MPI_SYS_Bind
- HI_MPI_SYS_UnBind
- HI_MPI_SYS_GetBindbyDest

PAYLOAD_TYPE_E

[Description]

Defines the audio or video payload type.

[Syntax]

```
typedef enum
{
    PT_PCMU = 0,
    PT_1016 = 1,
    PT_G721 = 2,
    PT_GSM = 3,
    PT_G723 = 4,
    PT_DVI4_8K = 5,
    PT_DVI4_16K = 6,
    PT_LPC = 7,
    PT_PCMA = 8,
    PT_G722 = 9,
    PT_S16BE_STEREO,
    PT_S16BE_MONO = 11,
    PT_QCELP = 12,
    PT_CN = 13,
    PT_MPEGAUDIO = 14,
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
PT_G728 = 15,
PT_DVI4_3 = 16,
PT_DVI4_4 = 17,
PT_G729 = 18,
PT_G711A = 19,
PT_G711U = 20,
PT_G726 = 21,
PT_G729A = 22,
PT_LPCM = 23,
PT_CelB = 25,
PT_JPEG = 26,
PT_CUSM = 27,
PT_NV = 28,
PT_PICW = 29,
PT_CPV = 30,
PT_H261 = 31,
PT_MPEGVIDEO = 32,
PT_MPEG2TS = 33,
PT_H263 = 34,
PT_SPEG = 35,
PT_MPEG2VIDEO = 36,
PT_AAC = 37,
PT_WMA9STD = 38,
PT_HEAAC = 39,
PT_PCM_VOICE = 40,
PT_PCM_AUDIO = 41,
PT_AACLC = 42,
PT_MP3 = 43,
PT_ADPCMA = 49,
PT_AEC = 50,
PT_X_LD = 95,
PT_H264 = 96,
PT_D_GSM_HR = 200,
PT_D_GSM_EFR = 201,
PT_D_L8 = 202,
PT_D_RED = 203,
PT_D_VDVI = 204,
PT_D_BT656 = 220,
PT_D_H263_1998 = 221,
PT_D_MP1S = 222,
PT_D_MP2P = 223,
PT_D_BMPEG = 224,
PT_MP4VIDEO = 230,
PT_MP4AUDIO = 237,
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    PT_VC1 = 238,
    PT_JVC_ASF = 255,
    PT_D_AVI = 256,
    PT_MAX = 257,
    PT_AMR = 1001, /* add by mpp */
    PT_MJPEG = 1002,
}PAYLOAD_TYPE_E;
```

[Member]

None

[Note]

None

[See Also]

- VENC_CHN_ATTR_S
- VDEC_CHN_ATTR_S
- AENC_CHN_ATTR_S
- ADEC_CHN_ATTR_S

2.4.2 System Control Data Type

The system control data types are as follows:

- MPP_SYS_CONF_S: Defines HiMPP system control attributes.
- VB_CONF_S: Defines VB pool attributes.
- MPP_VERSION_S: Defines the structure of the HiMPP version information.

MPP_SYS_CONF_S

[Description]

Defines HiMPP system control attributes.

[Syntax]

```
typedef struct hiMPP_SYS_CONF_S
{
    /* stride of picture buffer must be aligned with this value.
    * you can choose a value from 1 to 1024,
    * and it except 1 must be multiple of 16.*/
    HI_U32  u32AlignWidth;
}MPP_SYS_CONF_S;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u32AlignWidth	Stride for aligning the pictures used in the system, in bytes. Value range: [1, 1024]. 16-byte alignment is recommended. Static attribute.

[Note]

None

[See Also]

HI_MPI_SYS_SetConf

VB_CONF_S

[Description]

Defines VB pool attributes.

[Syntax]

```
typedef struct hiVB_CONF_S
{
HI_U32  u32MaxPoolCnt;  /* max count of pools, (0,VB_MAX_POOLS] */
Struct  hiVB_CPOOL_S
{
HI_U32  u32BlkSize;
HI_U32  u32BlkCnt;
HI_CHAR acMmzName[MAX_MMZ_NAME_LEN];
}astCommPool[VB_MAX_COMM_POOLS];
```

[Member]

Member	Description
u32MaxPoolCnt	Maximum number of VB pools in the entire system. Value range: (0, VB_MAX_POOLS] Static attribute. The value is fixed at VB_MAX_POOLS at present.
astCommPool	Attributes of the public VB pool. The members include the size (in bytes) of each buffer, number of buffers in the public VB pool, and the name of the DDR where the VB pool is located. Static attribute.

[Note]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

HI_MPI_SYS_SetConf

MPP_VERSION_S

[Description]

Defines the structure of the HiMPP version information.

[Syntax]

```
typedef struct hiMPP_VERSION_S
{
HI_CHAR aVersion[64];
}MPP_VERSION_S;
```

[Member]

Member	Description
aVersion	String for describing the version information, for example, HI_VERSION=Hi35xx_MPP_V1.0.4.0.

[Note]

None

[See Also]

HI_MPI_SYS_GetVersion

2.4.3 Public Video Data Type

The public video data types are as follows:

- VIDEO_NORM_E: Defines the video input norm.
- PIXEL_FORMAT_E: Defines the pixel format.
- VIDEO_FIELD_E: Defines the frame or field type of a video picture.
- VIDEO_FRAME_S: Defines the original video picture frame.
- VIDEO_FRAME_INFO_S: Defines the video picture information.
- BITMAP_S: Defines the bitmap information.
- VIDEO_VBI_INFO_S: Defines the video vertical blanking interval (VBI) information.

VIDEO_NORM_E

[Description]

Defines the video input norm.

[Syntax]

```
typedef enum hiVIDEO_NORM_E
{
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    VIDEO_ENCODING_MODE_PAL=0,
    VIDEO_ENCODING_MODE_NTSC,
    VIDEO_ENCODING_MODE_AUTO,
    VIDEO_ENCODING_MODE_BUTT,
} VIDEO_NORM_E;
```

[Member]

Member	Description
VIDEO_ENCODING_MODE_PAL	Phase alternating line (PAL) norm.
VIDEO_ENCODING_MODE_NTSC	National Television Systems Committee (NTSC) norm.
VIDEO_ENCODING_MODE_AUTO	Automatic mode.

[Note]

The automatic mode is not supported.

[See Also]

VI_PUB_ATTR_S

PIXEL_FORMAT_E

[Description]

Defines the pixel format.

[Syntax]

```
typedef enum hiPIXEL_FORMAT_E
{
    PIXEL_FORMAT_RGB_1BPP = 0,
    PIXEL_FORMAT_RGB_2BPP,
    PIXEL_FORMAT_RGB_4BPP,
    PIXEL_FORMAT_RGB_8BPP,
    PIXEL_FORMAT_RGB_444,
    PIXEL_FORMAT_RGB_4444,
    PIXEL_FORMAT_RGB_555,
    PIXEL_FORMAT_RGB_565,
    PIXEL_FORMAT_RGB_1555,
    PIXEL_FORMAT_RGB_888,
    PIXEL_FORMAT_RGB_8888,
    PIXEL_FORMAT_RGB_PLANAR_888,
    PIXEL_FORMAT_RGB_BAYER,
    PIXEL_FORMAT_YUV_A422,
    PIXEL_FORMAT_YUV_A444,
    PIXEL_FORMAT_YUV_PLANAR_422,
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    PIXEL_FORMAT_YUV_PLANAR_420,
    PIXEL_FORMAT_YUV_PLANAR_444,
    PIXEL_FORMAT_YUV_SEMIPLANAR_422,
    PIXEL_FORMAT_YUV_SEMIPLANAR_420,
    PIXEL_FORMAT_YUV_SEMIPLANAR_444,
    PIXEL_FORMAT_UYVY_PACKAGE_422,
    PIXEL_FORMAT_YUYV_PACKAGE_422,
    PIXEL_FORMAT_VYUY_PACKAGE_422,
    PIXEL_FORMAT_YCbCr_PLANAR,
    PIXEL_FORMAT_BUTT
} PIXEL_FORMAT_E;
```

[Member]

None

[Note]

None

[See Also]

- VI_CHN_ATTR_S
- OVERLAY_ATTR_S

VIDEO_FIELD_E

[Description]

Defines the frame/field type of a video picture.

[Syntax]

```
typedef enum hiVIDEO_FIELD_E
{
    VIDEO_FIELD_TOP        = 0x01,    /* even field */
    VIDEO_FIELD_BOTTOM     = 0x02,    /* odd field */
    VIDEO_FIELD_INTERLACED  = 0x03,    /* two interlaced fields */
    VIDEO_FIELD_FRAME      = 0x04,    /* frame */
    VIDEO_FIELD_BUTT
} VIDEO_FIELD_E;
```

[Member]

Member	Description
VIDEO_FIELD_TOP	Top field.
VIDEO_FIELD_BOTTOM	Bottom field.
VIDEO_FIELD_INTERLACED	Interlaced.
VIDEO_FIELD_FRAME	Frame.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

None

[See Also]

VIDEO_FRAME_S

VIDEO_FRAME_S

[Description]

Defines the original video picture frame.

[Syntax]

```
typedef struct hiVIDEO_FRAME_S
{
   PIXEL_FORMAT_E   enPixelFormat;
   HI_U32           u32Width;
   HI_U32           u32Height;

   VIDEO_FIELD_E    u32Field;
   HI_U32   u32PhyAddr[3];
   HI_VOID  *pVirAddr[3];
   HI_U32   u32Stride[3];

   HI_U16   u16OffsetTop;       /* top offset of show area */
   HI_U16   u16OffsetBottom;    /* bottom offset of show area */
   HI_U16   u16OffsetLeft;      /* left offset of show area */
   HI_U16   u16OffsetRight;     /* right offset of show area */

   HI_U64   u64pts;
   HI_U32   u32TimeRef;

   HI_U32   u32PrivateData;
}VIDEO_FRAME_S;
```

[Member]

Member	Description
enPixelFormat	Pixel format of a video picture.
u32Width	Picture width.
u32Height	Picture height.
u32Field	Frame/field mode.
u32PhyAddr	Physical address.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
pVirAddr	Virtual address.
u32Stride	Picture stride.
u16OffsetTop	Top offset of the displayed area.
u16OffsetBottom	Bottom offset of the displayed area.
u16OffsetLeft	Left offset of the displayed area.
u16OffsetRight	Right offset of the displayed area.
u64pts	Picture PTS.
u32TimeRef	Frame sequence number of a picture.
u32PrivateData	Private data.

[Note]

None

[See Also]

VI_PUB_ATTR_S

VIDEO_FRAME_INFO_S

[Description]

Defines the video picture information.

[Syntax]

```
typedef struct hiVIDEO_FRAME_INFO_S
{
   VIDEO_FRAME_S    stVFrame;
   HI_U32           u32PoolId;
}VIDEO_FRAME_INFO_S;
```

[Member]

Member	Description
stVFrame	Video picture frame.
u32PoolId	ID of a VB pool.

[Note]

None

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VIDEO_FRAME_S

BITMAP_S

[Description]

Defines the bitmap information.

[Syntax]

```
typedef struct hiBITMAP_S
{
    PIXEL_FORMAT_E  enPixelFormat;
    HI_U32          u32Width;
    HI_U32          u32Height;
    HI_VOID         *pData;
} BITMAP_S;
```

[Member]

Member	Description
enPixelFormat	Pixel format of a bitmap.
u32Width	Bitmap width.
u32Height	Bitmap height.
pData	Bitmap data.

[Note]

None

[See Also]

REGION_CTRL_PARAM_U

VIDEO_VBI_INFO_S

[Description]

Defines the video VBI information.

[Syntax]

```
typedef struct hiVIDEO_VBI_INFO_S
{
  HI_U32 au32Data[VIU_MAX_VBI_LEN];
  HI_U32 u32Len;
}VIDEO_VBI_INFO_S;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
au32Data	VBI data.
u32Len	VBI data length.

[Note]

None

2.5 Error Codes

2.5.1 Error Codes for System Control APIs

Table 2-2 describes the error codes for system control APIs.

Table 2-2 Error codes for the system control APIs

Error Code	Macro Definition	Description
0xA0028003	HI_ERR_SYS_ILLEGAL_PARAM	The parameter configuration is invalid.
0xA0028006	HI_ERR_SYS_NULL_PTR	The pointer is null.
0xA0028009	HI_ERR_SYS_NOT_PERM	The operation is forbidden.
0xA0028010	HI_ERR_SYS_NOTREADY	The system control attributes are not configured.
0xA0028012	HI_ERR_SYS_BUSY	The system is busy.
0xA002800C	HI_ERR_SYS_NOMEM	The memory fails to be allocated due to some causes such as insufficient system memory.

2.5.2 Error Codes for VB Pool APIs

Table 2-3 describes the error codes for VB pool APIs.

Table 2-3 Error codes for VB pool APIs

Error Code	Macro Definition	Description
0xA0018003	HI_ERR_VB_ILLEGAL_PARAM	The parameter configuration is invalid.
0xA0018005	HI_ERR_VB_UNEXIST	The VB pool does not exist.
0xA0018006	HI_ERR_VB_NULL_PTR	The pointer is null.
0xA0018009	HI_ERR_VB_NOT_PERM	The operation is forbidden.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Macro Definition	Description
0xA001800C	HI_ERR_VB_NOMEM	The memory fails to be allocated.
0xA001800D	HI_ERR_VB_NOBUF	The buffer fails to be allocated.
0xA0018010	HI_ERR_VB_NOTREADY	The system control attributes are not configured.
0xA0018012	HI_ERR_VB_BUSY	The system is busy.
0xA0018040	HI_ERR_VB_2MPOOLS	Too much buffer pools are created.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Contents

3 VI 3-1
3.1 Overview 3-1
3.2 Concepts 3-1
3.3 Functions 3-2
3.3.1 Hi3531/Hi3532 3-2
3.3.2 Hi3521/Hi3520A 3-16
3.3.3 Hi3520D/Hi3515A/Hi3515C 3-19
3.3.4 Hi3518/Hi3516C 3-22
3.4 API Reference 3-23
3.5 Data Types 3-97
3.6 Error Codes 3-137

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figures

Figure 3-1 Functional block diagram of the Hi3531/Hi3532 VIU 3-3
Figure 3-2 Functional block diagram of a VI channel 3-5
Figure 3-3 Device ports and channels in a simplified manner 3-7
Figure 3-4 Device ports, ways, and channels in a comprehensive manner 3-8
Figure 3-5 16xD1/960H in 4-channel multiplexed mode 3-10
Figure 3-6 16xD1/960H in 2-channel multiplexed mode 3-11
Figure 3-7 8x720p or 8xD1/960H in 1-channel multiplexed mode 3-12
Figure 3-8 4x720p/1080p in 1-channel multiplexed mode 3-13
Figure 3-9 Data transfer in VI cascade mode 3-15
Figure 3-10 Functional block diagram of the Hi3521/Hi3520A VIU 3-17
Figure 3-11 Functional block diagram of the Hi3520D VIU 3-20
Figure 3-12 Functional block diagram of the Hi3518/Hi3516C VIU 3-22
Figure 3-13 Workflow of Hi3518/Hi3516C VI channels 3-22

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Tables

Table 3-1 Comparison between odd device ports and even device ports....3-4
Table 3-2 Specifications of each primary VI channel....3-5
Table 3-3 Typical application scenarios....3-6
Table 3-4 Multiplexing mode for default VI pins and HD inputs....3-14
Table 3-5 Multiplexing mode for other pins and 16xD1 inputs....3-14
Table 3-6 Mask configuration in 8x720p input scenario....3-15
Table 3-7 Mask configuration in the 4x720p or 1080p input scenario....3-15
Table 3-8 Mask configuration in 16-channel input scenario....3-18
Table 3-9 Mask configuration in 4x720p input mode....3-18
Table 3-10 Mask configuration in 2x720p or 1080p input mode....3-19
Table 3-11 Mask configuration in 8-channel input scenario....3-21
Table 3-12 Mask configuration in 2x720p mode....3-21
Table 3-13 Mask configuration in 1-channle 720p or 1x1080p mode....3-21
Table 3-14 Mask configuration in 1x720p mode (12 bits)....3-23
Table 3-15 Mask configuration in 1xD1 input mode....3-23
Table 3-16 Parameters for the primary attribute and secondary attribute....3-40
Table 3-17 Error codes of VI APIs....3-137

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

3 VI

3.1 Overview

The video input unit (VIU) stores the external video data to the specified address of the DDR over the ITU-R BT.656 interface, BT.601 interface, BT.1120 interface, or the digital camera (DC) interface. During this process, the VIU can process the source picture data including cropping, horizontal down scaling, mirroring, and flipping. The VIU also supports 1-channel source picture input and 1-channel or multi-channel picture outputs. Figure 3-1 shows the functional block diagram of the VIU.

3.2 Concepts

Note the following:

- VI device

  A VI device supports multiple input timings and parses them.
- Physical VI channel

  A physical VI channel transfers the data obtained after a VI device parses timings to the DDR. Before the data arrives in the DDR, cropping, covering, horizontal down scaling, mirroring, and flipping are supported. For details, see the related data sheet.
- Mask

  Mask indicates the video data source of a VI device.
- Cascade

  It is a mode in which the video output end of a chip transfers video data to the video input end of another chip complying with the BT.1120 timing.
- Lens distortion correction (LDC)

  The pictures captured by some low-end lenses are easily distorted. This function is used to correct distorted pictures.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

3.3 Functions

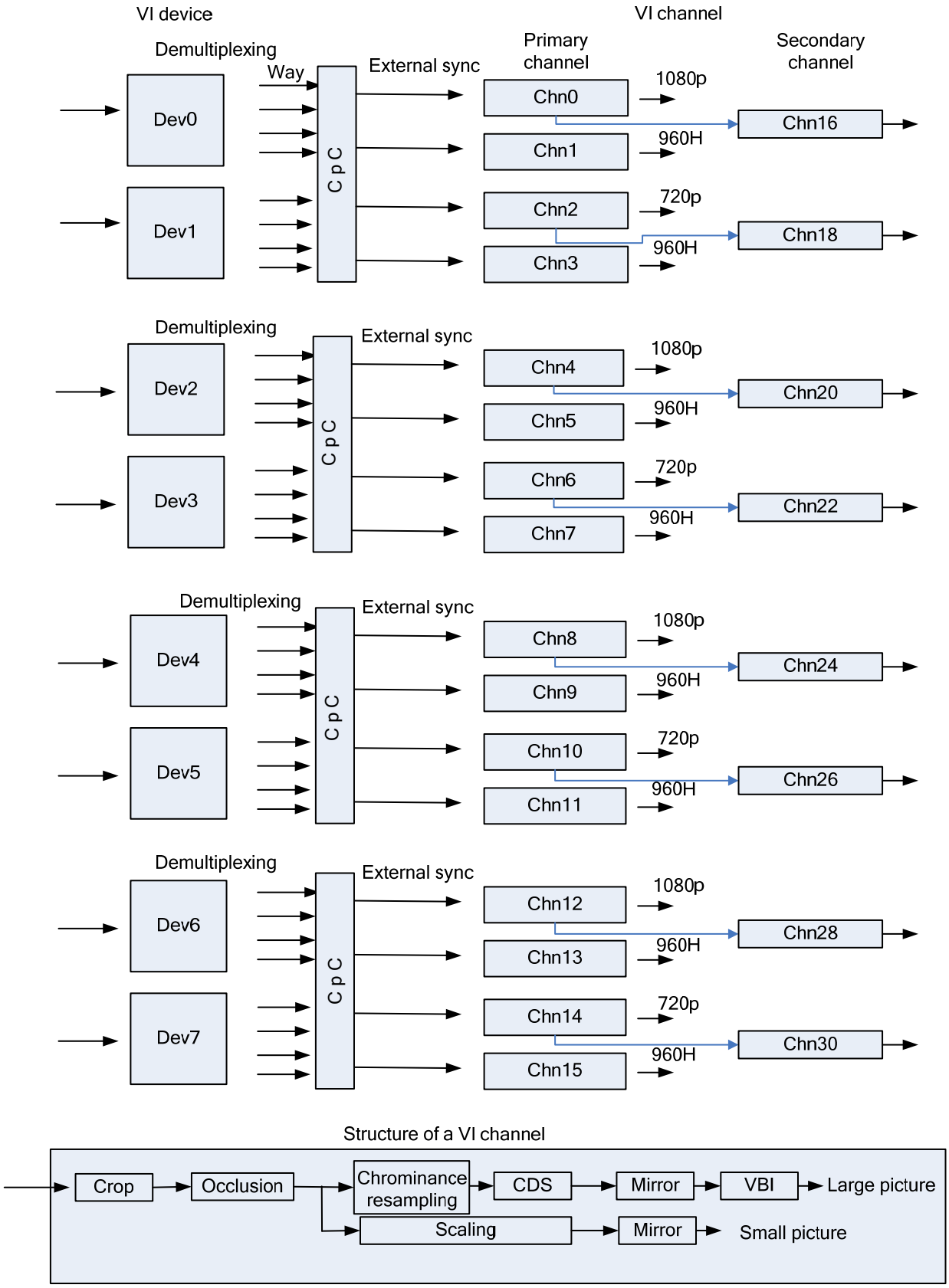
3.3.1 Hi3531/Hi3532

Functional Block Diagram

Figure 3-1 shows the functional block diagram of the Hi3531/Hi3532 VIU.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 3-1 Functional block diagram of the Hi3531/Hi3532 VIU



HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VI Device Ports

The Hi3531 has four BT.1120 interfaces, and the Hi3532 has five BT.1120 interfaces. Typically, each BT.1120 interface corresponds to two VI devices. To be specific, the first BT.1120 interface corresponds to the VI devices Dev0 and Dev1, the second BT.1120 interface corresponds to the VI devices Dev2 and Dev3, and so on. Note that the fifth BT.1120 interface of the Hi3532 is used only for transparent transmission in cascade mode and does not correspond to any VI device. In the 16xD1 or 16x960H scenario, Dev2 of the Hi3531 can receive video data from the lower eight bits of the first BT.1120 interface, and Dev6 of the Hi3531 can receive video data from the lower eight bits of the third BT.1120 interface. In this way, 16-channel video data can be captured by using only two BT.1120 interfaces.

The Hi3531 has eight VI device ports and each VI device port supports 2xD1 multiplexed inputs (complying with the BT.656 protocol) and 1x720p interleaving. Only Dev0, Dev2, Dev4, or Dev6 supports 4xD1 multiplexed inputs (complying with the BT.656 protocol) or 1x720p/1080p input (complying with the BT.1120 protocol). In this case, Dev1, Dev3, Dev5, and Dev7 are unavailable. The functions of odd device ports and even device ports are different. For details, see Table 3-1.

Table 3-1 Comparison between odd device ports and even device ports

Function	Dev0, Dev2, Dev4, or Dev6	Dev1, Dev3, Dev5, or Dev7
Maximum picture input performance	2560 x1600 megapixels	720p
4xD1/960H multiplexed	BT.656 timing supporting 4-channel multiplexing, 108 MHz (4x4D1 input scenario) or 144 MHz (4x960H input scenario)	Not supported
2xD1/960H multiplexed	BT.656 timing supporting 2-channel multiplexing, 54 MHz (8x2D1 input scenario) or 72 MHz (8x960H input scenario)	BT.656 timing supporting 2-channel multiplexing, 54 MHz (8x2D1 input scenario) or 72 MHz (8x960H input scenario)
1xD1/960H multiplexed	BT.656 timing, 27 MHz (8x1D1 input scenario) or 36 MHz (8x960H input scenario)	BT.656 timing, 27 MHz (8x1D1 input scenario) or 36 MHz (8x960H input scenario)
Timings of the connected peripheral	• BT.656 timing (480i/576i) • BT.601 timing • SMPTE 296M timing (720p) • SMPTE293M or ITU-R BT.1358 timing (480p/576p) • SMPTE 274M or BT.1120 timing (1080i/1080p) • Universal DC interface timing	• BT.656 timing (480i/576i) • BT.601 timing • SMPTE 296M timing (720p) • SMPTE293M or ITU-R BT.1358 timing (480p/576p) • Universal DC interface timing
Interleave mode, 8-bit	Interleave mode for BT.1120 timing 148.5 MHz (8x720p input scenario)	Interleave mode for BT.1120 timing 148.5 MHz (8x720p input scenario)
Data type of the	YUV data	YUV data

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Function	Dev0, Dev2, Dev4, or Dev6	Dev1, Dev3, Dev5, or Dev7
peripheral		

Physical VI Channels

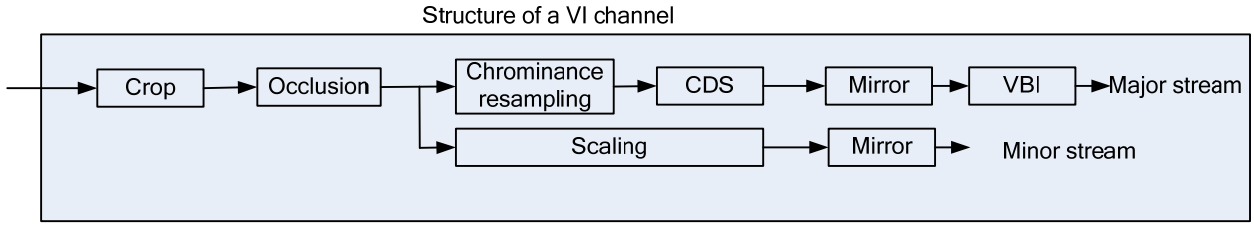
The Hi3531 or Hi3532 VIU has 32 VI hardware channels that are called physical VI channels in the software development kit (SDK). The 32 physical VI channels are classified into primary channels (chn0–chn15) and secondary channels (chn16–chn31). As only the secondary channels corresponding to the primary channels with even channel IDs are valid, the VIU supports 24 valid VI channels. The picture sources of primary channels and secondary channels are the same. The odd secondary channel ID is calculated as follows: Secondary channel ID = Primary channel ID + 16. The secondary channel ID can be obtained by using the SUBCHN macro that is provided by calling the corresponding API.

The Hi3531 VI channels process the received pictures in sequence as follows:

- For the primary channels: cropping, occluding, CDS, mirroring and flipping, vertical blanking interval (VBI), output of processed pictures in the DDR.
- For the secondary channels: cropping, occluding, zooming out horizontally and vertically, mirroring and flipping, output of processed pictures in the DDR

The pictures of the secondary channels are the pictures that are cropped and occluded by the primary channels. See Figure 3-2.

Figure 3-2 Functional block diagram of a VI channel



All primary channels support D1 and 960H video inputs and all even primary channels support the 720p pictures input in GV7601 10bit multiplexed mode. Only chn0, chn4, chn8, and chn12 support standard BT1120 720p or 1080p video inputs. Table 3-2 describes the specifications of each primary VI channel.

Table 3-2 Specifications of each primary VI channel

Specification	Odd Channel	Chn2, Chn6, Chn10, or Chn14	Chn0, Chn4, Chn8, or Chn12
Crop	Supported	Supported	Supported
Horizontal and vertical zoom out	Not supported	Supported At most x1/16 zoom out	Supported At most x1/16 zoom out
Maximum input	960x576	1280x720 (the	2560x1600 (the

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Specification	Odd Channel	Chn2, Chn6, Chn10, or Chn14	Chn0, Chn4, Chn8, or Chn12
resolution		maximum input resolution of the corresponding secondary channel is 640x720)	maximum input resolution of the corresponding secondary channel is 960x1600)
Mirror and flip	Supported	Supported	Supported
Pixel format	Semi-planar420 and semi-planar422	Semi-planar420 and semi-planar422	Semi-planar420 and semi-planar422
GV7601 10-bit multiplexed input mode	Not supported	Supported	Supported
SMPTE 296M (720p)	Not supported	Not supported	Supported
SMPTE 274M (1080i/1080p)	Not supported	Not supported	Supported

Typical Application Scenarios

Table 3-3 describes the typical application scenarios.

Table 3-3 Typical application scenarios

Typical Application Scenario	Description
16xD1 capture	4xD1 inputs (BT.656/108 MHz) for Dev0, Dev2, Dev4, and Dev6 respectively 1xD1 output for chn0–chn15 respectively and outputs of eight secondary channels
	2xD1 inputs (BT.656/54 MHz) for Dev0–Dev9 respectively 1xD1 output for chn0–chn15 respectively and outputs of eight secondary channels
16x960H capture	4xD1 inputs (BT.656/144 MHz) for Dev0, Dev2, Dev4, and Dev6 respectively 1x960H output for chn0–chn15 respectively and outputs of eight secondary channels
	2x960H inputs (BT.656/72 MHz) for Dev0–Dev7 respectively 1x960H output for chn0–chn15 respectively and outputs of eight secondary channels
8xD1 capture	1xD1 input (BT.656/27 MHz) for Dev0–Dev7 respectively

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

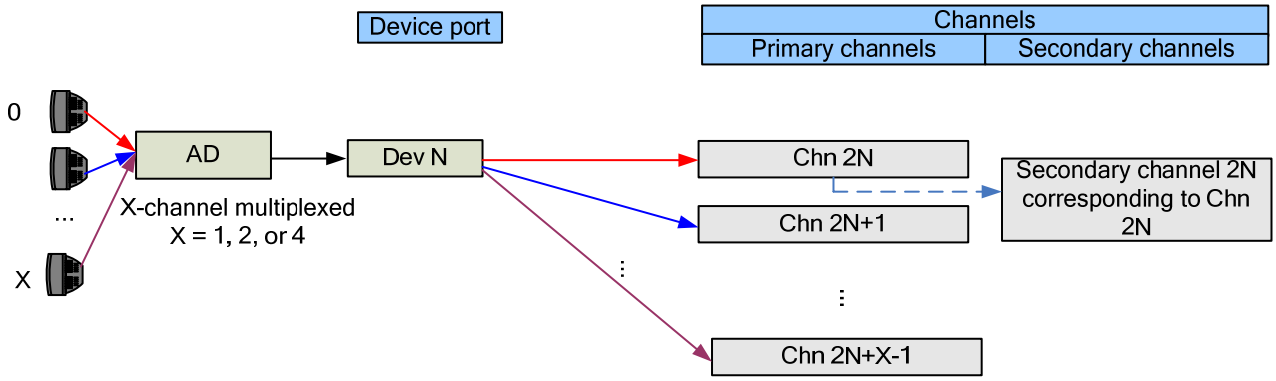
Typical Application Scenario	Description
	Outputs for even channels and outputs for eight secondary channels
8x960H capture	1x960H input (BT.656/36 MHz) for Dev0–Dev7 respectively Outputs for even channels and outputs for eight secondary channels
8x720p/30 fps inputs	BT.656/148.5 MHz for eight ports (each port connects to an 8-bit data line, in GV7601 10-bit multiplexed mode) Outputs for even channels and outputs for eight secondary channels
4x720p/30 fps inputs	1x720p input for Dev0, Dev2, Dev4, and Dev6 respectively (each port connects to a 16-bit data line, BT.1120/74.25 MHz) Outputs for chn0, chn4, chn8 and chn12 (channel ID = 2 x Device port ID) and output for four secondary channels
4x1080p/30 fps inputs	1x1080p input for Dev0, Dev2, Dev4, and Dev6 respectively (each port connects to a 16-bit data line, BT.1120/74.25 MHz) Outputs for chn0, chn4, chn8 and chn12 (channel ID = 2 x Device port ID) and output for four secondary channels

Binding Relationships

The VI device ports and VI channels are divided into four groups. That is, group 1 includes Dev0, Dev1, and chn0–chn3, group 2 includes Dev2, Dev3, and chn4–chn7, and so on. For details, see Figure 3-1. For each group, the first two channels must be bound to the first device port, and the other two channels can be bound to either of the device ports. However, a channel can be bound only to one device port and only primary channels can be bound. When device attributes are set, the system sets the binding relationship between channels and device ports by default. Therefore, the binding relationship does not need to be set. The default binding relationship meets the requirements in most application scenarios. You can change the binding relationship as required.

Figure 3-3 shows device ports and channels in a simplified manner.

Figure 3-3 Device ports and channels in a simplified manner



HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

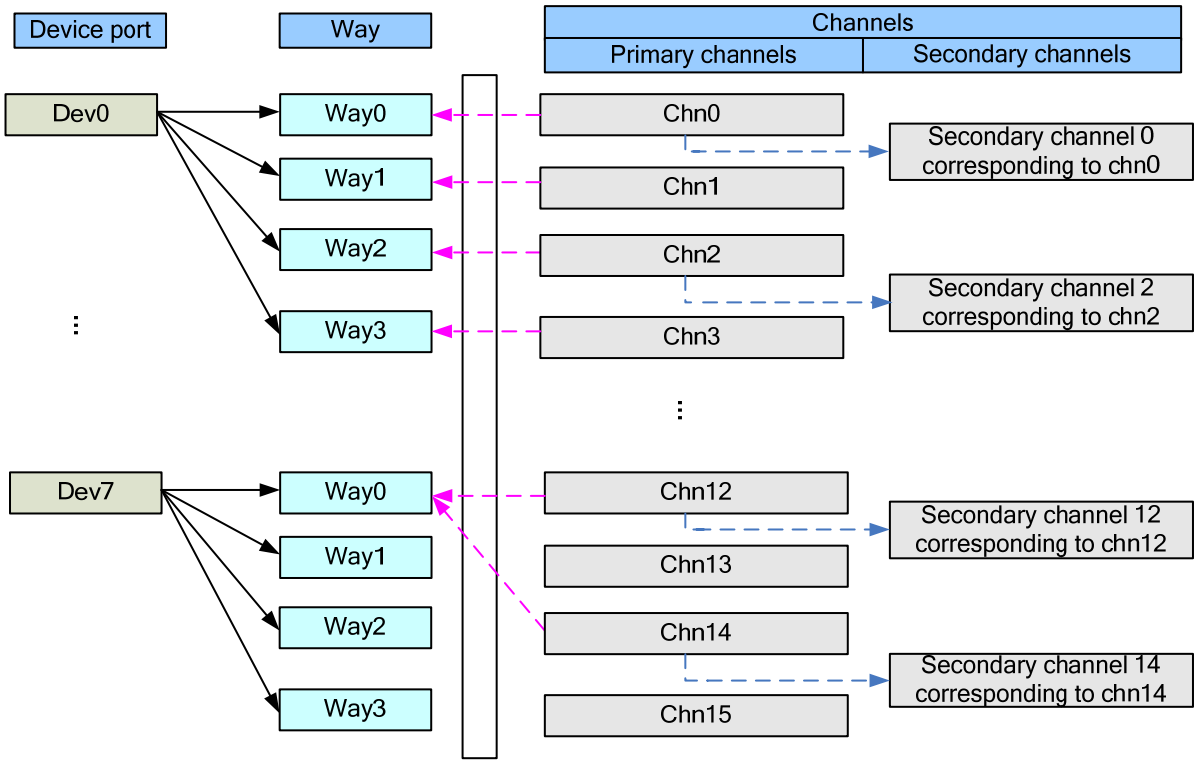
You only need to know the concepts of device ports and channels and binding relationship between them. That is, you do not need to explicitly bind device ports to channels. As shown in Figure 3-3, the color arrows before channels correspond to those before the analog-to-digital converter (ADC) respectively. When Dev N is set to X-channel multiplexed mode (X = 1, 2, or 4), the number of bound channels is X, and the bound channels are chn 2N to chn (2N + X – 1) by default. For example, if Dev0 is set to 1-channel multiplexed mode, chn0 is bound to Dev0 by default; if Dev0 is set to 2-channel multiplexed mode, chn0 and chn1 are bound to Dev0 by default; if Dev0 is set to 4-channel multiplexed mode, chn0 to chn3 are bound to Dev0 by default.

CAUTION

When Dev N (N = 0, 2, 4, or 6) is set to 4-channel multiplexed mode or 720p/1080p (standard BT.1120) input mode, Dev (N+1) is unavailable. Only primary channels need to be bound, because the secondary channels share the binding relationship with primary channels.

Figure 3-4 shows device ports, ways, and channels in a comprehensive manner.

Figure 3-4 Device ports, ways, and channels in a comprehensive manner



If you want to display the pictures from the same VI source through multiple channels, you can change the default binding relationship. That is, you can bind multiple channels to the same group of device port and way. As shown in Figure 3-4, each device port has four data ways, way0–way3. A way corresponds to the input of a camera. The ways to be used depend on the configured X-channel multiplexed mode. By default, each group of channels is bound

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

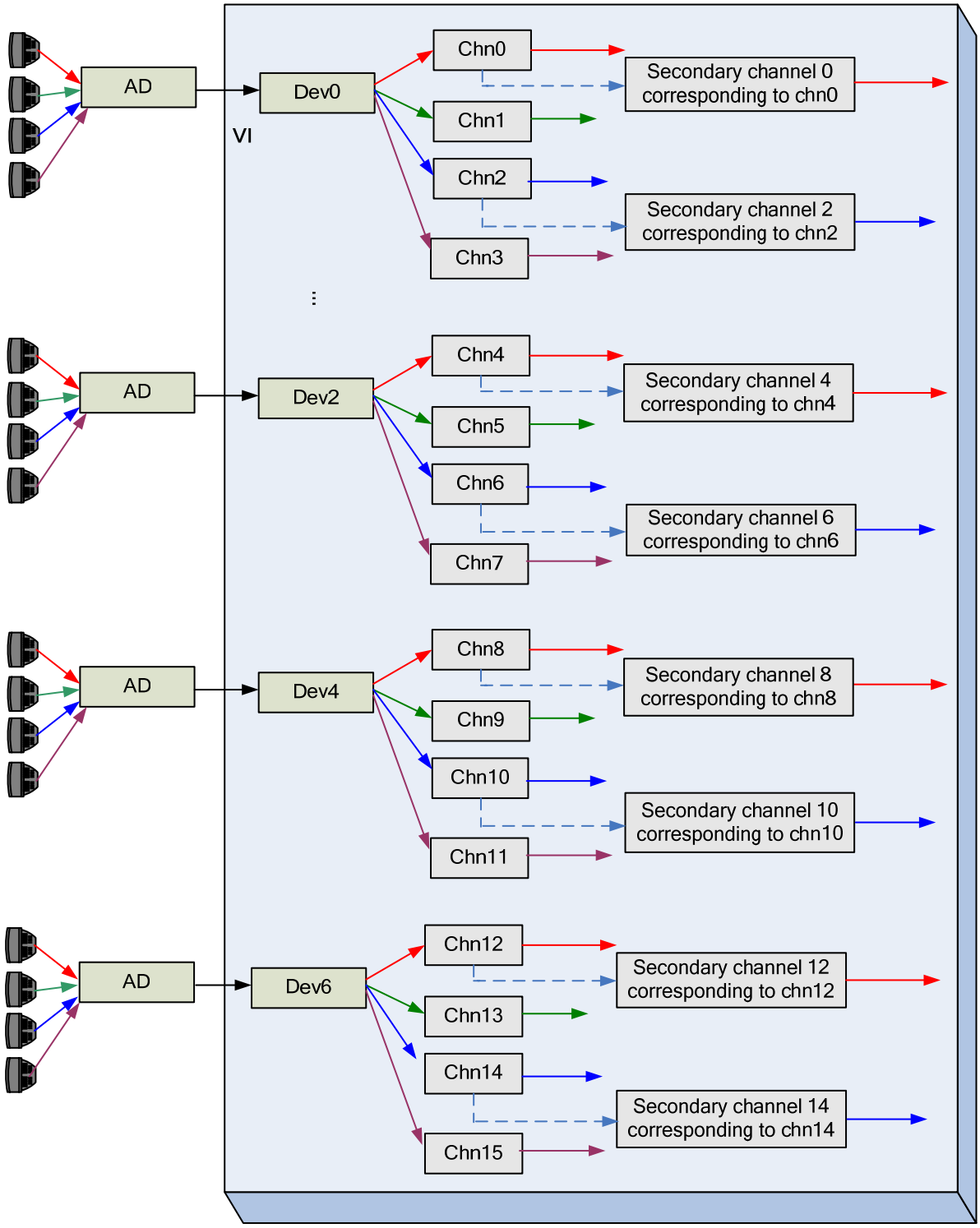
to the enabled ways in sequence based on the X-channel multiplexed mode. For example, if Dev0 and Dev1 are set to 2-channel multiplexed mode, chn0 and chn1 are bound to way0 and way1 of Dev0 respectively and the pictures from camera0 and camera1 are output by default; chn2 and chn3 are bound to way0 and chn1 of Dev1 respectively and the pictures from camera0 and camera 1 are output by default. If Dev0 is set to 4-channel multiplexed mode, chn0–chn3 are bound to way0–way3 of Dev0 respectively and the pictures from camera0–camera3 are output. You can change the default binding relationship. As shown in Figure 3-4, you can bind chn12 and chn14 to way0 of Dev7. In this way, the VI sources of chn12 and chn14 are the same, which enables pictures with different resolutions to be output.

Default Binding Relationships

Figure 3-5 to Figure 3-8 show default binding relationships.

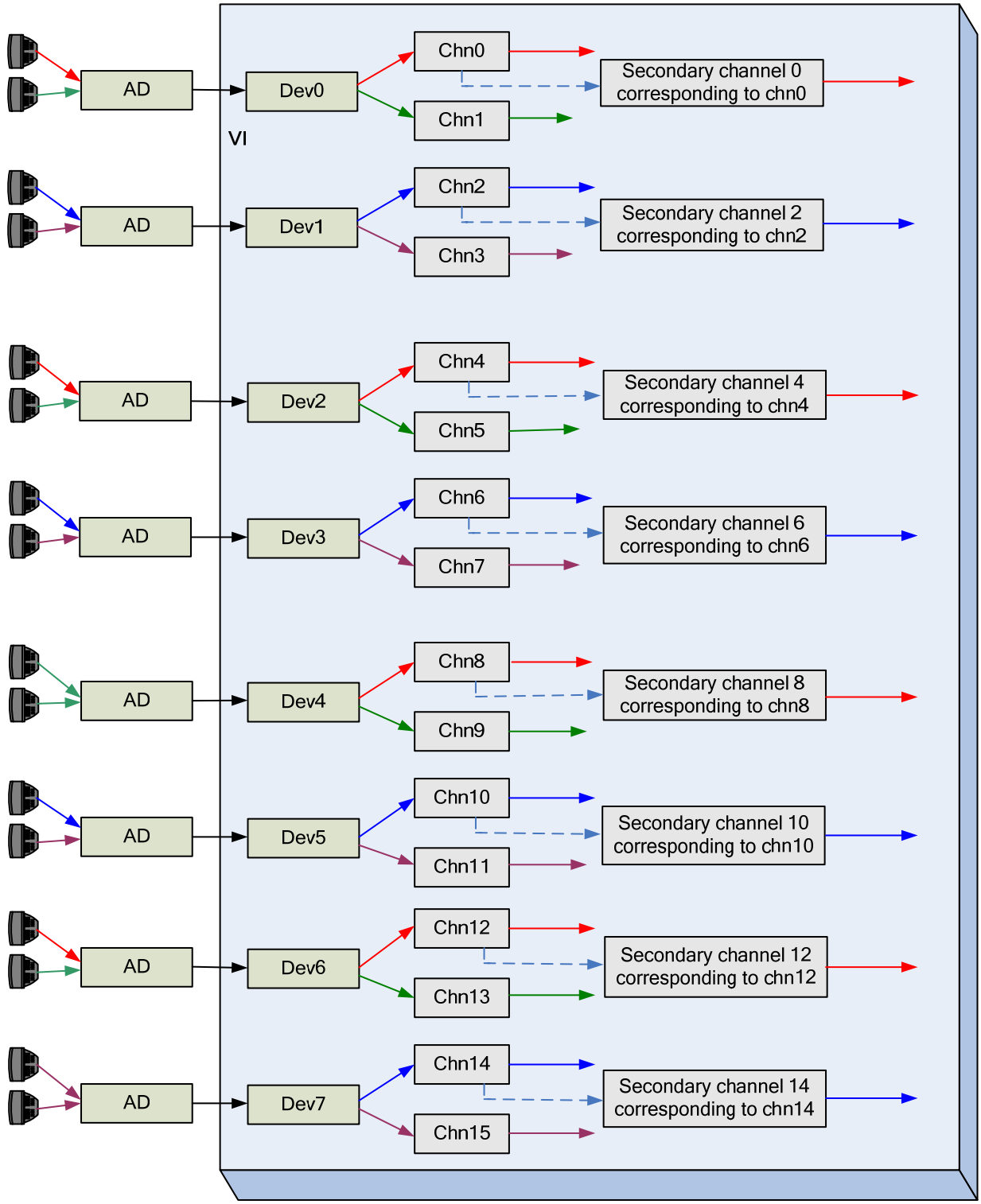
HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 3-5 16xD1/960H in 4-channel multiplexed mode



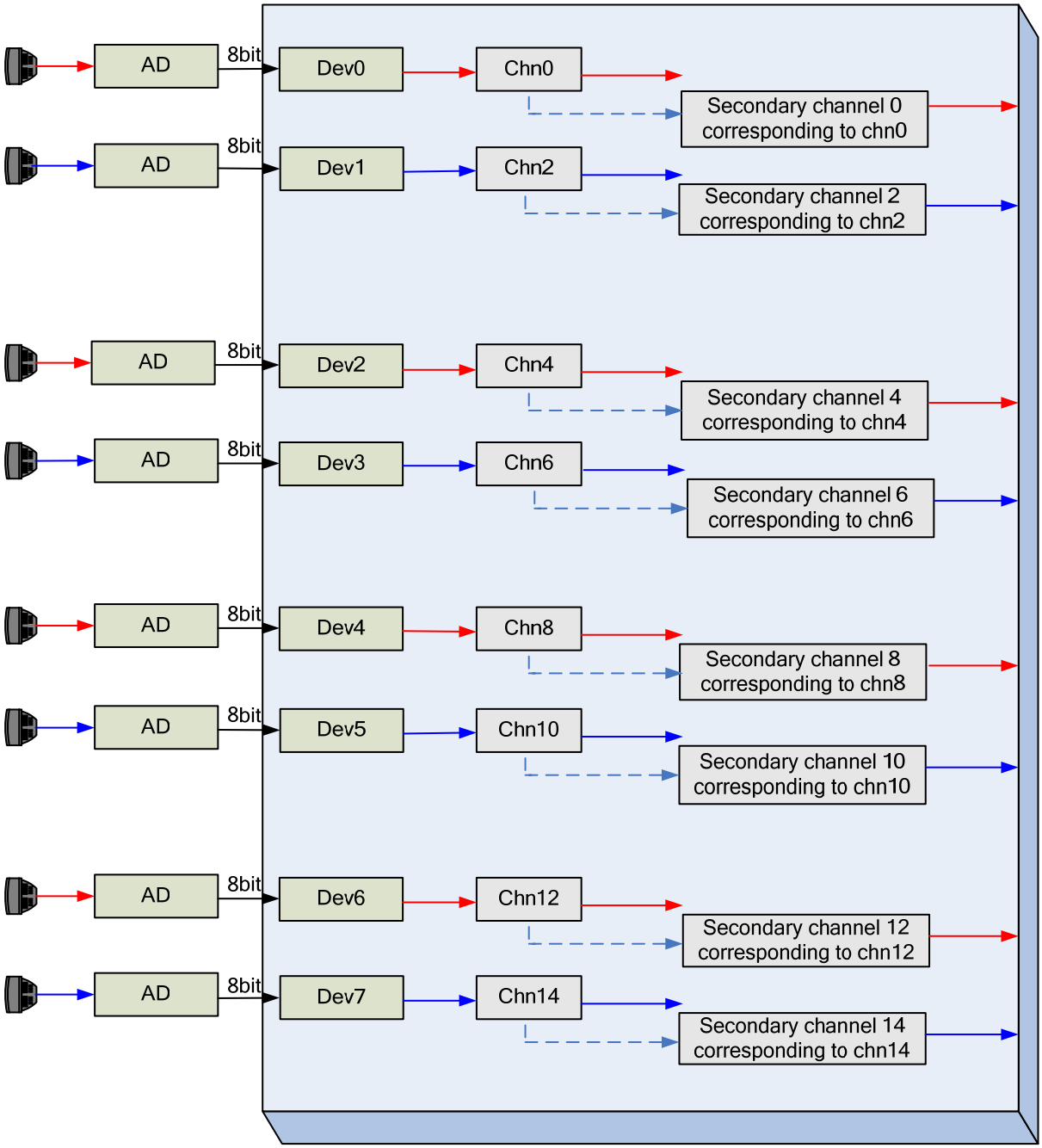
HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 3-6 16xD1/960H in 2-channel multiplexed mode



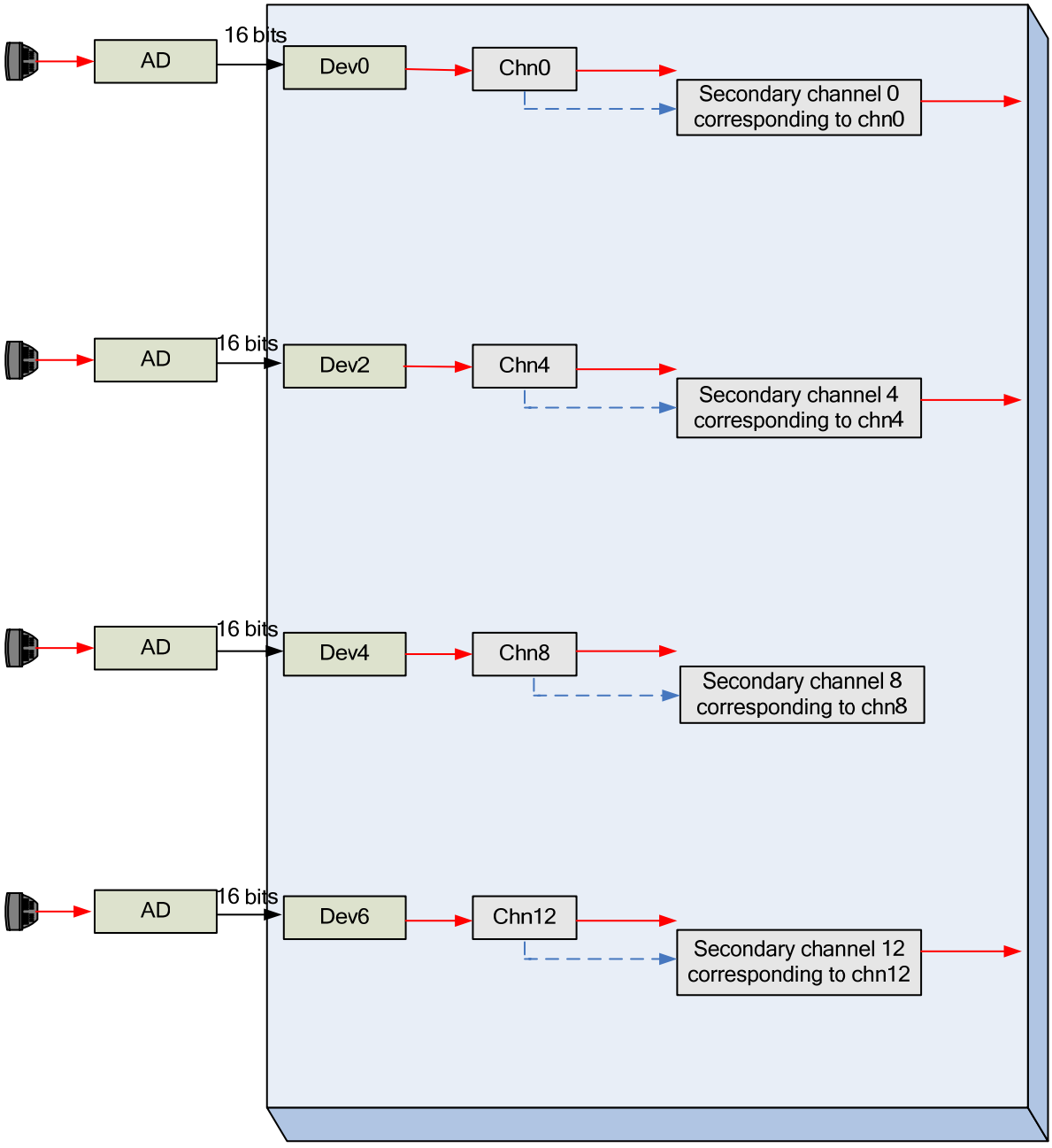
HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 3-7 8x720p or 8xD1/960H in 1-channel multiplexed mode



HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 3-8 4x720p/1080p in 1-channel multiplexed mode



As shown in Figure 3-8, the arrows in the same color indicate the inputs of the same camera. Note that every two device ports and corresponding four channels constitute a group. That is, the Hi3531 VIU has four {Dev, Chn} groups that have the same functions, as shown in Figure 3-1. In each {Dev, Chn} group, only the first device port supports the BT1120_STANDARD mode (4x720p/1080p) and 4-channel multiplexed mode, and only the first channel can output 1080p pictures. In addition, the X-channel multiplexed mode (X = 1, 2, or 4) determines the number of channels that are bound to device ports.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Mask Configuration

When a VI device starts to work, the data source is selected based on the mask configuration.

- 16xD1 or 16x960H

  In 16xD1 or 16x960H scenario, the chip is connected in either of the following schemes:

  - Scheme 1: The upper eight bits of four BT.1120 interfaces (VIU0 to VIU3) are used to receive video data from the ADC.
  - Scheme 2: The upper eight bits and lower eight bits of the first and the third BT.1120 interfaces (VIU0 and VIU2) are used to receive video data from the ADC.

  The masks must be configured based on Table 3-4 and Table 3-5.

Table 3-4 Multiplexing mode for default VI pins and HD inputs

Device ID	Mask 0	Mask 1
0	0xFF000000	0x0
1	0x00FF0000	0x0
2	0x0000FF00	0x0
3	0x000000FF	0x0
4	0xFF000000	0x0
5	0x00FF0000	0x0
6	0x0000FF00	0x0
7	0x000000FF	0x0

CAUTION

The Hi3531 or Hi3532 can receive 16xD1 inputs over two BT.1120 interfaces. For details, see the *Hi3531 H.264 CODEC Processor Data Sheet* or *Hi3532 H.264 Encoding Processor Data Sheet*. When the pin multiplexing mode is configured to receive 16xD1 inputs over two BT.1120 interfaces, the mask configuration is shown in Figure 3-5.

Table 3-5 Multiplexing mode for other pins and 16xD1 inputs

Device ID	Mask 0	Mask 1
0	0xFF000000	0x0
2	0x00FF0000	0x0
4	0xFF000000	0x0
6	0x00FF0000	0x0

- 8x720p

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

In the 8x720p scenario, the masks must be configured based on Table 3-6.

Table 3-6 Mask configuration in 8x720p input scenario

Device ID	Mask 0	Mask 1
0	0xFF000000	0x0
1	0x00FF0000	0x0
2	0x0000FF00	0x0
3	0x000000FF	0x0
4	0xFF000000	0x0
5	0x00FF0000	0x0
6	0x0000FF00	0x0
7	0x000000FF	0x0

- 4x720p or 4x1080p

  In the 4x720p or 4x1080p scenario, the masks must be configured based on Table 3-7.

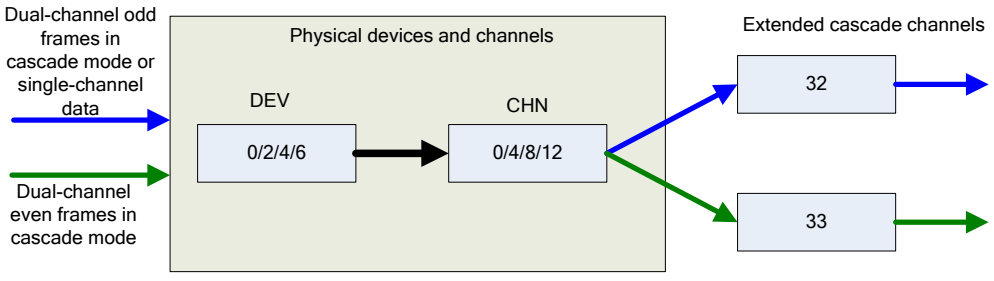
Table 3-7 Mask configuration in the 4x720p or 1080p input scenario

Device ID	Mask 0	Mask 1
0	0xFF000000	0x00FF0000
2	0x0000FF00	0x000000FF
4	0xFF000000	0x00FF0000
6	0x0000FF00	0x000000FF

Cascade Mode

Figure 3-9 shows data transfer in VI cascade mode.

Figure 3-9 Data transfer in VI cascade mode



HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

The VI devices Dev0, Dev2, Dev4, and Dev6 and physical channels chn0, chn4, chn8, and chn12 support the cascade mode. After the cascade function is enabled, the VIU captures cascade pictures by using the original physical devices and physical channels. The two extended cascade channels (virtual channels) transmit and differentiate captured 2-channel cascade pictures. Note the following:

- In single-channel cascade mode, chn32 transmits captured pictures.
- In dual-channel cascade mode, chan32 transmits the pictures (odd frames) transferred by extended VO cascade device 1, and chn33 transmits the pictures (even frames) transferred by extended VO cascade device 2.

Note that the picture ID configured at the VO end determines whether a frame is an odd frame or even frame. In dual-channel cascade mode, the picture ID for extended VO cascade device 1 is fixed at odd frame, and the picture ID for extended VO cascade device 2 is fixed at even frame. The SDK ensures that odd and even frames are transferred in interleaved mode during dual-data transfer. The VIU checks whether the current frame is an odd frame or even frame by reading the picture VBI information.

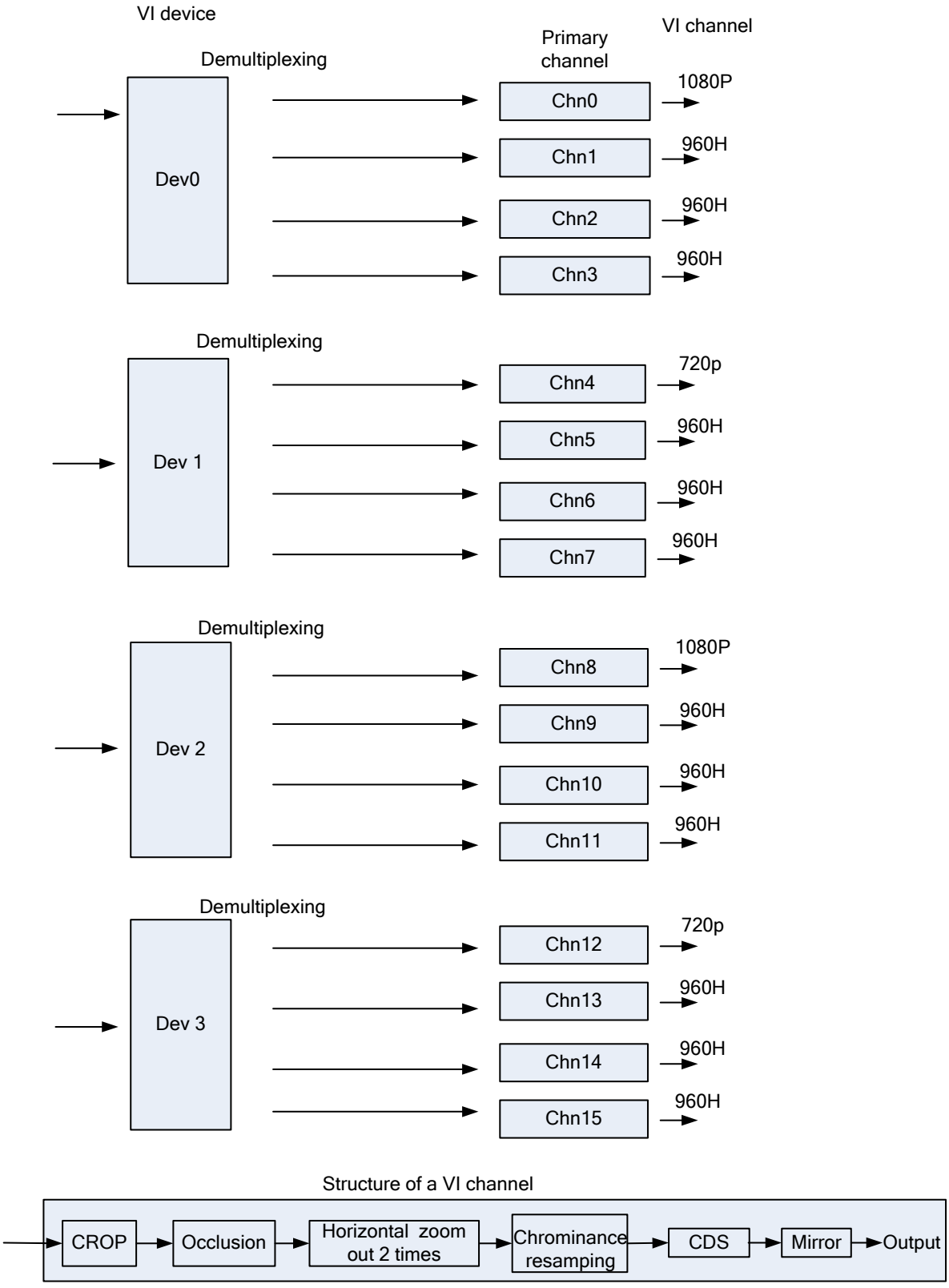
3.3.2 Hi3521/Hi3520A

Functional Block Diagram

Figure 3-10 shows the functional block diagram of the Hi3521/Hi3520A VIU.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 3-10 Functional block diagram of the Hi3521/Hi3520A VIU



HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VI Device Ports

The Hi3521/Hi3520A has two BT.1120 interfaces, and each BT.1120 interface corresponds to two VI devices. To be specific, the first BT.1120 interface corresponds to the VI devices Dev0 and Dev1, the second BT.1120 interface corresponds to the VI devices Dev2 and Dev3.

The Hi3521/Hi3520A has four VI device ports, and each VI device port supports 4xD1 multiplexed inputs (complying with the BT.656 protocol) and 1x720p interleaving. Only Dev0 or Dev2 supports 1x720p/1080p input (complying with the BT.1120 protocol). In this case, Dev1 and Dev3 are unavailable.

Physical VI Channels

The Hi3521/Hi3520A VIU has 16 VI hardware channels that are called physical VI channels in the SDK. The 16 physical VI channels are classified into chn0–chn15 and no secondary channels.

All channels support D1 and 960H video inputs and chn0, chn4, chn8 or chn12 support the 720p pictures input in GV7601 10bit multiplexed mode. Only chn0 and chn8 support standard BT1120 720p or 1080p video inputs.

Binding Relationships

The VI devices and physical VI channels are divided into four groups. Each group has one device and four channels. For example, Dev0 and Chn0–Chn3 belong to a group, and so on. In each group, the physical VI channels are always bound to the VI device. Their binding relationships cannot be changed.

Mask Configuration

- 16-channel inputs

  In the 16-channel input scenario, the masks must be configured based on Table 3-8.

Table 3-8 Mask configuration in 16-channel input scenario

Device ID	Mask 0	Mask 1
0	0xFF000000	0x0
1	0x00FF0000	0x0
2	0x0000FF00	0x0
3	0x000000FF	0x0

- 4x720p

  In 4x720p scenario, the masks must be configured based on Table 3-9.

Table 3-9 Mask configuration in 4x720p input mode

Device ID	Mask 0	Mask 1
0	0xFF000000	0x0

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Device ID	Mask 0	Mask 1
1	0x00FF0000	0x0
2	0x0000FF00	0x0
3	0x000000FF	0x0

- 2x720p or 2x1080p
  In 2x720p or 2x1080p scenario, the masks must be configured based on Table 3-10.

Table 3-10 Mask configuration in 2x720p or 1080p input mode

Device ID	Mask 0	Mask 1
0	0xFF000000	0x00FF0000
2	0x0000FF00	0x000000FF

Cascade Mode

The Hi3521/Hi3520A does not support the cascade mode.

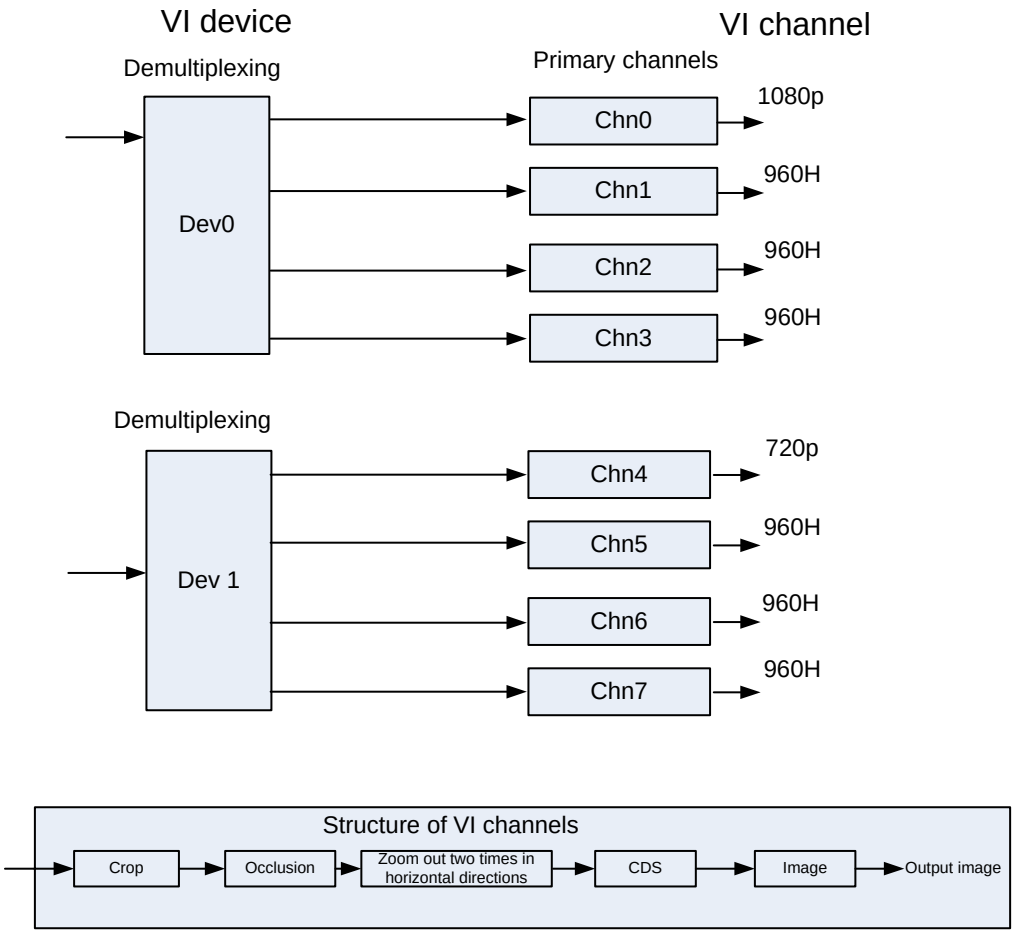
3.3.3 Hi3520D/Hi3515A/Hi3515C

Functional Block Diagram

Figure 3-11 shows the functional block diagram of the Hi3520D VIU. The Hi3515A/Hi3515C supports only one VI device and four physical VI channels, and channel 0 supports at most 720p inputs.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 3-11 Functional block diagram of the Hi3520D VIU



VI Devices

- The Hi3520D has one BT.1120 interface, which corresponds to two VI devices: Dev0 and Dev1.

  Each VI device supports 4xD1 multiplexed inputs (complying with the BT.656 protocol) and 1x720p interleaved input. Only Dev0 supports 1x720p/1080p high definition (HD) input (complying with the BT.1120 protocol). In this case, the devices with odd ID cannot be used.
- The Hi3515A/Hi3515C has only one BT.656 interface.

  The VI device supports 4xD1 multiplexed inputs (complying with the BT.656 protocol) and 1x720p interleaved input.

Physical VI Channels

- The Hi3520D VIU supports eight physical VI channels Chn0–Chn7. All channels act as primary channels and support D1 and 960H video inputs. The channels whose ID is an integer of 4 support 720p inputs in GV7601 10-bit multiplexed mode. Only Chn0 supports standard BT.1120 720p or 1080p video inputs.
- The Hi3515A/Hi3515C VIU supports four physical VI channels Chn0−Chn3. All channels act as primary channels and support D1 and 960H video inputs. The channels

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

whose ID is a multiple of 4 support 720p inputs in GV7601 10-bit multiplexed mode. The standard BT.1120 720p/1080p video inputs are not supported.

Binding Relationship

The VI devices and physical VI channels are divided into two groups. Each group has one device and four channels. For example, Dev0 and Chn0–Chn3 belong to a group, and so on. The physical VI channels are always bound to the VI device in the same group, and the binding relationships cannot be changed.

Mask Configuration

- 8-channel input

  In the 8-channel input scenario, the masks must be configured based on Table 3-11.

Table 3-11 Mask configuration in 8-channel input scenario

Device ID	Mask 0	Mask 1
0	0xFF000000	0x0
1	0x00FF0000	0x0

- 2x720p (interleaved) scenario

  In the 2x720p scenario, the masks must be configured based on Table 3-12 when you set the VI device attributes.

Table 3-12 Mask configuration in 2x720p mode

Device ID	Mask 0	Mask 1
0	0xFF000000	0x0
1	0x00FF0000	0x0

- 1x720p or 1x1080p scenario

  In the 1x720p or 1x1080p scenario, the masks must be configured based on Table 3-13 when you set the VI device attributes.

Table 3-13 Mask configuration in 1-channle 720p or 1x1080p mode

Device ID	Mask 0	Mask 1
0	0xFF000000	0x00FF0000

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Cascade Mode

The Hi3520D or Hi3515A/Hi3515C does not support the cascade mode.

3.3.4 Hi3518/Hi3516C

Functional Block Diagram

Figure 3-12 shows the functional block diagram of the Hi3518/Hi3516C VIU.

Figure 3-12 Functional block diagram of the Hi3518/Hi3516C VIU

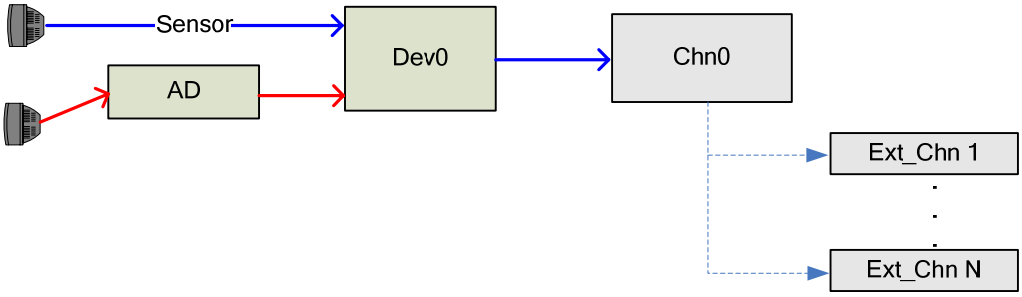
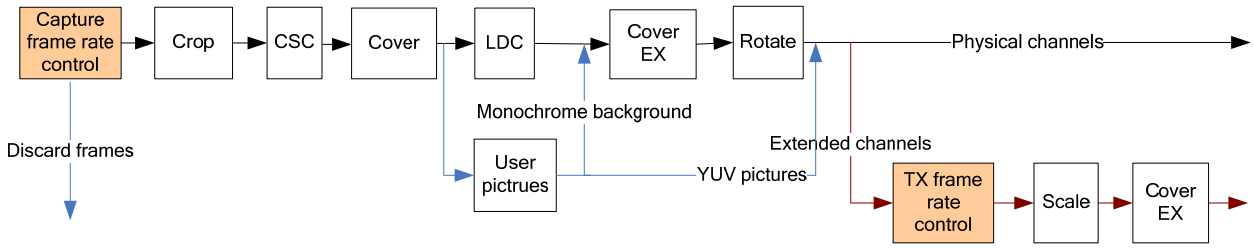


Figure 3-13 shows the workflow of Hi3518/Hi3516C VI channels.

Figure 3-13 Workflow of Hi3518/Hi3516C VI channels



VI Device Port

The Hi3518/Hi3516C provides only one VI device port Dev0.

Dev0 supports BT.656, BT.601, and DC inputs but not BT.1120 inputs. In addition, Dev0 does not support inconsecutive pixel clock timings.

Physical VI Channel

The Hi3518/Hi3516C VIU provides only one physical VI channel Chn0. This is no secondary channel for the Hi3518/Hi3516C. The Hi3518/Hi3516C supports extended channels.

Chn0 supports various resolutions such as D1, 720p@30, 720p@60, and 1080p@30.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Extended VI Channels

Extended VI channels are used for scaling, and their data source is the physical VI channel. The Hi3518/Hi3516C supports a maximum of 16 extended VI channels.

Binding Relationship

The binding relationship between the Hi3518/Hi3516C physical VI channel and the device is fixed. The binding relationship cannot be changed.

Mask Configuration

The upper 12 bits of the mask correspond to the 12 pins (D0–D11) on the hardware circuit. Configure the mask appropriately based on the actual connection. The most upper bit of the mask corresponds to D11 pin. For example, if the pins (D2–D11) connect to a 10-bit sensor, configure the mask to 0xFFC00000.

- 1x720p inputs (12-bit inputs)

  In the 1x720p input scenario, the masks must be configured based on Table 3-14.

Table 3-14 Mask configuration in 1x720p mode (12 bits)

Device ID	Mask 0	Mask 1
0	0xFFF00000	0x0

- 1xD1 inputs (8-bit inputs)

  In the 1xD1 input scenario, the masks must be configured based on Table 3-15.

Table 3-15 Mask configuration in 1xD1 input mode

Device ID	Mask 0	Mask 1
0	0xFF000000	0x0

3.4 API Reference

The VIU enables VI devices and VI channels and binds VI channels. The VIU provides the following MPP programming interfaces (MPIs):

- HI_MPI_VI_SetDevAttr: Sets the attributes of a VI device.
- HI_MPI_VI_GetDevAttr: Obtains the attributes of a VI device.
- HI_MPI_VI_SetDevAttrEx: Sets the attributes of a VI device (this advanced MPI is called for special timings).
- HI_MPI_VI_GetDevAttrEx: obtains the attributes of a VI device.
- HI_MPI_VI_EnableDev: Enables a VI device.
- HI_MPI_VI_DisableDev: Disables a VI device.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_VI_SetChnAttr: Sets the attributes of a VI channel.
- HI_MPI_VI_GetChnAttr: Obtains the attributes of a VI channel.
- HI_MPI_VI_SetChnMinorAttr: Sets the secondary attribute of a VI channel.
- HI_MPI_VI_GetChnMinorAttr: Obtains the secondary attribute of a VI channel.
- HI_MPI_VI_ClearChnMinorAttr: Clears the secondary attribute settings of a VI channel.
- HI_MPI_VI_EnableChn: Enables a VI channel.
- HI_MPI_VI_DisableChn: Disables a VI channel.
- HI_MPI_VI_EnableChnInterrupt: Enables a VI channel to respond to the hardware interrupt handler.
- HI_MPI_VI_DisableChnInterrupt: Forbids a VI channel to respond to the hardware interrupt handler.
- HI_MPI_VI_SetFrameDepth: Sets the maximum depth of the buffer for storing the obtained VI pictures.
- HI_MPI_VI_GetFrameDepth: Obtains the maximum depth of the buffer for storing the obtained VI pictures.
- HI_MPI_VI_GetFrame: Obtains pictures captured by a VI channel.
- HI_MPI_VI_GetFrameTimeOut: Obtains pictures captured by a VI channel within the configured period.
- HI_MPI_VI_ReleaseFrame: Releases the buffer occupied by VI pictures.
- HI_MPI_VI_SetUserPic: Sets a user picture that is used as the picture inserted when there is no video signal.
- HI_MPI_VI_EnableUserPic: Enables the function of inserting a user picture.
- HI_MPI_VI_DisableUserPic: Disables the function of inserting a user picture.
- HI_MPI_VI_EnableCascade: Enables the cascade mode of a VI device.
- HI_MPI_VI_DisableCascade: Disables the cascade mode of a VI device.
- HI_MPI_VI_EnableCascadeChn: Enables an extended VI cascade channel.
- HI_MPI_VI_DisableCascadeChn: Disables an extended VI cascade channel.
- HI_MPI_VI_ChnBind: Binds a VI channel.
- HI_MPI_VI_ChnUnBind: Unbinds a VI channel.
- HI_MPI_VI_GetChnBind: Obtains the binding relationship of a VI channel.
- HI_MPI_VI_GetFd: Obtains the device file descriptor (FD) corresponding to a VI channel.
- HI_MPI_VI_Query: Queries the information about a VI channel such as the interrupt count and average frame rate.
- HI_MPI_VI_SetFlashConfig: Sets the camera flash.
- HI_MPI_VI_GetFlashConfig: Obtains camera flash settings.
- HI_MPI_VI_FlashTrigger: Enables or disables the camera flash.
- HI_MPI_VI_SetExtChnAttr: Sets the attributes of an extended channel.
- HI_MPI_VI_GetExtChnAttr: Obtains the attributes of an extended VI channel.
- HI_MPI_VI_SetLDCAttr: Sets the LDC attribute.
- HI_MPI_VI_GetLDCAttr: Obtains the LDC attribute.
- HI_MPI_VI_SetCSCAttr: Sets the CSC attribute of a VI device.
- HI_MPI_VI_GetCSCAttr: Obtains the CSC attribute of a VI device.
- HI_MPI_VI_SetRotate: Sets the rotation attribute of VI pictures.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_VI_GetRotate: Obtains the rotation attribute of VI pictures.
- HI_MPI_VI_GetChnLuma: Obtains the channel luminance statistics.
- HI_MPI_VI_SetPortMode: Sets the VI port mode.
- HI_MPI_VI_GetPortMode: Obtains the VI port mode.
- HI_MPI_VI_EnableChn422to420: Enables the function of converting a captured YUV422 image into a YUV420 image for the VI channel.
- HI_MPI_VI_DisableChn422to420: Disables the function of converting a captured YUV422 image into a YUV420 image for the VI channel.
- HI_MPI_VI_SetChnScanMode: Sets the scanning mode of a VI channel.
- HI_MPI_VI_GetChnScanMode: Obtains the scanning mode of a VI channel.

HI_MPI_VI_SetDevAttr

[Description]

Sets the attributes of a VI device. As basic device attributes support some chip configurations by default, the interconnection requirements of most ADCs can be met. For some special timings, the advanced MPI HI_MPI_VI_SetDevAttrEx can be called to set all attributes of a VI device.

[Syntax]

```
HI_S32 HI_MPI_VI_SetDevAttr(VI_DEV ViDev, const VI_DEV_ATTR_S *pstDevAttr)
```

[Parameter]

Parameter	Description	Input/Output
ViDev	ID of a VI device Value range: [0, VIU_MAX_DEV_NUM)	Input
pstDevAttr	Pointer to the attributes of a VI device Static attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	After this MPI is called, the system automatically configures the default binding relationships between channels and device ports based on the interface mode and working mode. To change the

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Description
	binding relationship, you can call this MPI, call HI_MPI_VI_ChnUnBind to unbind channels, and call HI_MPI_VI_ChnBind to configure new binding relationships.
Hi3521/Hi3520A/Hi3520D/Hi3515A/ Hi3515C	The Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C does not allow you to change the binding relationships between devices and channels.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support the BT.1120 timing and does not allow you to change the binding relationship between the physical channel and the device. The Hi3518/Hi3516C supports only 1-channel multiplexing mode.

[VI Parameter]

Parameter	Description
detect_err_frame	This parameter indicates whether the user detects the VIU timing exceptions in real time. 0: Default value. If **detect_err_frame** is **0**, the real-time detection is controlled by the SDK, and the detected error frames are discarded. If **detect_err_frame** is **−1**, the function of discarding error frames is disabled. If **detect_err_frame** is greater than **0**, the real-time detection is controlled by the user.
drop_err_frame	This parameter indicates that consecutive *n* frames are discarded when the system detects a picture error no matter whether the subsequent frames are correct. If **drop_err_frame** is **0** or **1**, the current frame is discarded. The default value is **0**.
int_gap_time	In the PAL standard, the value of **int_gap_time** ranges from 20000 to 40000, and the unit is μs. If the interval between two interrupts exceeds 40000 μs, pictures are lost in the VIU.
stop_int_level	When the VIU detects too many interrupts, interrupt response is automatically disabled, ensuring that applications are responded. The default value of **stop_int_level** is **0**, and the unit is μs. The value of **stop_int_level** cannot be too large because when the service traffic volume is large, the interrupt is delayed. In this case, the interval between two interrupts is short, but the VIU can still handle this.
discard_int	This parameter indicates that first *n* interrupts are discarded. That is, you can enable the channel capturing function after interrupt configurations take effect to prevent logic errors. The default value of **discard_int** is **0**.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- Before calling HI_MPI_VI_SetDevAttr, ensure that the VI device is disabled. If the VI device is enabled, you can disable it by calling HI_MPI_VI_DisableDev.
- The pstDevAttr parameter is used to configure the video interface mode of a specified VI device to connect it to an external camera, sensor, or codec. The supported interface modes include BT.656 mode, BT.601 mode, DC mode, BT.1120 progressive mode, and BT.1120 interlaced mode. You need to configure the following information. For details, see section 3.5 "Data Types."
  - Interface mode information: BT.656 mode, BT.601 mode, DC mode, or BT.1120 mode
  - Working mode information: 1-, 2-, or 4-channel multiplexed mode
  - Data layout information: layout of valid data
  - Data information: interlaced or progressive input and YUV data input sequence
  - Sync timing information: attributes of the vertical or horizontal sync signal

[Example]

```
HI_S32 s32Ret;
VI_DEV ViDev = 0;
VI_CHN ViChn = 0;
VI_DEV_ATTR_S stDevAttr;
VI_CHN_ATTR_S stChnAttr;
stDevAttr.enIntfMode = VI_MODE_BT656;
stDevAttr.enWorkMode = VI_WORK_MODE_1Multiplex;
stDevAttr.au32CompMask[0] = 0xFF000000;
stDevAttr.au32CompMask[1] = 0x0;
stDevAttr.enScanMode = VI_SCAN_INTERLACED;
stDevAttr.s32AdChnId[0] = -1;
stDevAttr.s32AdChnId[1] = -1;
stDevAttr.s32AdChnId[2] = -1;
stDevAttr.s32AdChnId[3] = -1;

s32Ret = HI_MPI_VI_SetDevAttr(ViDev, &stDevAttr);
if (s32Ret != HI_SUCCESS)
{
printf("Set dev attributes failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}
s32Ret = HI_MPI_VI_EnableDev(ViDev);
if (s32Ret != HI_SUCCESS)
{
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
printf("Enable dev failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}

stChnAttr.stCapRest.s32X = 0;
stChnAttr.stCapRect.s32Y = 0;
stChnAttr.stCapRect.u32Width = 720;
stChnAttr.stCapRect.u32Height = 576;
stChnAttr.stDestSize.u32Width = 720;
stChnAttr.stDestSize.u32Height = 576;
stChnAttr.enCapSel = VI_CAPSEL_BOTH;
stChnAttr.enPixFormat = PIXEL_FORMAT_YUV_SEMIPLANAR_422;
stChnAttr.bMirror = HI_FALSE;
stChnAttr.bFlip = HI_FALSE;
stChnAttr.bChromaResample = HI_FALSE
stChnAttr.s32SrcFrameRate = -1;
stChnAttr.s32FrameRate = -1;
s32Ret = HI_MPI_VI_SetChnAttr(ViChn,&stChnAttr);
if (s32Ret != HI_SUCCESS)
{
printf("Set chn attributes failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}
s32Ret = HI_MPI_VI_EnableChn(ViChn);
if (s32Ret != HI_SUCCESS)
{
printf("Enable chn failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}
/* now, vi is capturing images, you can do something else*/

s32Ret = HI_MPI_VI_DisableChn(ViChn);
if (s32Ret != HI_SUCCESS)
{
printf("Disable chn failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}
s32Ret = HI_MPI_VI_DisableDev(ViDev);
if (s32Ret != HI_SUCCESS)
{
printf("Disable dev failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

- HI_MPI_VI_GetDevAttr
- HI_MPI_VI_SetDevAttrEx

HI_MPI_VI_GetDevAttr

[Description]

Obtains the attributes of a VI device.

[Syntax]

```
HI_S32 HI_MPI_VI_GetDevAttr(VI_DEV ViDev, VI_DEV_ATTR_S *pstDevAttr)
```

[Parameter]

Parameter	Description	Input/Output
ViDev	ID of a VI device Value range: [0, VIU_MAX_DEV_NUM)	Input
pstDevAttr	Pointer to the attributes of a VI device	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

If the attributes of a VI device are not set, a code indicating failure is returned after this MPI is called.

[Example]

None

[See Also]

HI_MPI_VI_GetDevAttr

HI_MPI_VI_SetDevAttrEx

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Sets the attributes of a VI device. This MPI is called to set all attributes of a VI device only when HI_MPI_VI_GetDevAttr cannot meet the requirements of some special timings.

[Syntax]

```
HI_S32 HI_MPI_VI_SetDevAttrEx(VI_DEV ViDev, const VI_DEV_ATTR_EX_S
*pstDevAttrEx)
```

[Parameter]

Parameter	Description	Input/Output
ViDev	ID of a VI device Value range: [0, VIU_MAX_DEV_NUM)	Input
pstDevAttrEx	Pointer to the advanced attributes of a VI device Static attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	After this MPI is called, the system automatically configures the default binding relationships between channels and device ports based on the interface mode and working mode. To change the binding relationship, you can call this MPI, call an advanced MPI to unbind channels, and configure new binding relationships.
Hi3521/Hi3520A/Hi3520D/Hi3515A/ Hi3515C	The Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C does not allow you to change the binding relationships between devices and channels.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support the BT.1120 timing and does not allow you to change the binding relationship between the physical channel and the device. The Hi3518/Hi3516C supports only 1-channel multiplexing mode.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- Before calling HI_MPI_VI_SetDevAttr, ensure that the VI device is disabled. If the VI device is enabled, you can disable it by calling HI_MPI_VI_DisableDev.
- The pstDevAttrEx parameter is used to configure the video interface mode of a specified VI device to connect it to an external camera, sensor, or codec. The supported interface modes include BT.656 mode, BT.601 mode, DC mode, BT.1120 progressive mode, and BT.1120 interlaced mode. You need to configure the following information. For details, see section 3.5 "Data Types."
  - Interface mode information: BT.656 mode, BT.601 mode, DC mode, or interleave mode
  - Working mode information: 1-, 2-, or 4-channel multiplexed mode
  - Data layout information: layout of valid data
  - Data information: interlaced or progressive input, YUV data input sequence, composite or separation mode, and component mode
  - Clock information: sampling on the rising edge or falling edge
  - Sync timing information: attributes of the vertical or horizontal sync signal

[Example]

None

[See Also]

HI_MPI_VI_GetDevAttrEx

HI_MPI_VI_GetDevAttrEx

[Description]

Obtains the attributes of a VI device.

[Syntax]

```
HI_S32 HI_MPI_VI_GetDevAttrEx(VI_DEV ViDev, VI_DEV_ATTR_EX_S
*pstDevAttrEx)
```

[Parameter]

Parameter	Description	Input/Output
ViDev	ID of a VI device Value range: [0, VIU_MAX_DEV_NUM)	Input
pstDevAttrEx	Pointer to the advanced attributes of a VI device	Output

[Return Value]

Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

If the attributes of a VI device are not set, a code indicating failure is returned after this MPI is called.

[Example]

None

[See Also]

HI_MPI_VI_SetDevAttrEx

HI_MPI_VI_EnableDev

[Description]

Enables a VI device.

[Syntax]

```
HI_S32 HI_MPI_VI_EnableDev(VI_DEV ViDev);
```

[Parameter]

Parameter	Description	Input/Output
ViDev	ID of a VI device Value range: [0, VIU_MAX_DEV_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Library file: libmpi.a

[Note]

- Ensure that the attributes of a VI device are set before you enable it. Otherwise, a code indicating failure is returned.
- The VI device can be enabled repeatedly without any failure.

[Example]

See the example of HI_MPI_VI_SetDevAttr.

[See Also]

HI_MPI_VI_DisableDev

HI_MPI_VI_DisableDev

[Description]

Disables a VI device.

[Syntax]

```
HI_S32 HI_MPI_VI_DisableDev(VI_DEV ViDev);
```

[Parameter]

Parameter	Description	Input/Output
ViDev	ID of a VI device Value range: [0, VIU_MAX_DEV_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	After this MPI is called, the system automatically cancels the default binding relationships between channels and device ports. If you change the default binding relationships by calling an advanced MPI, you must cancel the binding relationships by calling the corresponding advanced MPI. Otherwise, the system unbinds all channels bound to the device port by default.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Description
Hi3521/Hi3520A/Hi3520D/Hi3515A /Hi3515C	The Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C does not allow you to change the binding relationships between devices and channels.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not allow you to change the binding relationships between devices and channels.

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- A VI device can be disabled only after all VI channels bound to the device are disabled.
- The VI device can be disabled repeatedly without any failure.
- The SDK supports the low-power mode. A VI device is closed after being disabled. You must set the attributes of the VI device before enabling it.

[Example]

See the example of HI_MPI_VI_SetDevAttr.

[See Also]

HI_MPI_VI_EnableDev

HI_MPI_VI_SetChnAttr

[Description]

Sets the attributes of a VI channel.

[Syntax]

```
HI_S32 HI_MPI_VI_SetChnAttr(VI_CHN ViChn,const VI_CHN_ATTR_S *pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input
pstAttr	Pointer to the attributes of a VI channel	Input

[Return Value]

Return Value	Description
0	Success

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Description	
Hi3531/Hi3532	ID of a VI channel	Value range: [0, VIU_MAX_CHN_NUM – 2). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM – 2) are invalid.
	Attributes of a VI channel	• The attributes of both primary and secondary channels can be set. The attributes of secondary channels can be set only after the attributes of primary channels are set. In this case, the primary channels are not required to be enabled. The attributes of a secondary channel can be set dynamically. That is, this MPI can be called when a VI secondary channel is enabled and is working. • In interlaced single-field capture mode, s32Y and u32Height must be 2-pixel aligned. In interlaced dual-field capture, s32Y and u32Height must be 4-pixel aligned. • stCapRect is valid only for primary channels. For primary channels, stCapRect (width and height), stDestSize, and enCapSel are static attributes. Before changing their values, you must disable primary and secondary channels. Ensure that the width and height that are set by configuring stDestSize are the same as those that are set by configuring stCapRect. After stDestSize of a primary channel is changed, the attributes of corresponding secondary channel must be reconfigured. Otherwise, the scaling coefficient is not updated, which affects the picture quality. • If you change the default binding relationship by calling an advanced MPI, you must configure the binding relationship again.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	ID of a VI channel	[0, VIU_MAX_CHN_NUM)
	Attributes of a VI channel	• For primary attributes, the width and height in stCapRect are static attributes, and others are dynamic attributes. For secondary attributes, it is meaningless to set stCapRect and control the frame rate. Other attributes

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Description	
		are dynamic attributes. • stCapRect is based on the source picture. The width of stDestSize can be the same as or half of the width of stCapRect. The height of stDestSize must be half of the height of stCapRect in single-field capture mode. In dual-field capture mode, the heights must be the same. • In interlaced capture mode, s32Y and u32Height must be 4-pixel aligned.
Hi3518/Hi3516C	ID of a VI channel	[0, VIU_MAX_PHYCHN_NUM)
	Attributes of a VI channel	• The width and height of stCapRect are static attributes, and other attributes are dynamic attributes. There is no secondary attribute for the Hi3518/Hi3516C. • stCapRect is used to crop the source picture. The widths defined by stDestSize and stCapRect must be the same. In single-field capture mode, the height defined by stDestSize must be half of the height defined by stCapRect. In dual-field capture mode, the heights defined by stDestSize and stCapRect must be the same. • In interlaced capture mode, s32Y and u32Height must be 4-pixel aligned. • The value of enPixFormat (output picture format) can be dynamically changed. The rotation and LDC functions of the Hi3518/Hi3516C are available only for semi-planar420 pictures. If enPixFormat is dynamically changed to semi-planar422 when the rotation or LDC function is enabled, no pictures are output. • You are advised not to control the frame rate when taking pictures. Otherwise, flash signal triggering is delayed, and pictures cannot be taken in a timely manner.

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

The limitations on each channel attribute item are as follows:

- Capture region (stCapRect)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

  - stCapStart is used to configure the position of the rectangle to be captured relative to the start point of the original picture. For the start point, the horizontal coordinate is in the unit of pixel and the vertical coordinate is in the unit of line.
  - s32X and u32Width must be 2-pixel aligned, and s32Y and u32Height must be 2-pixel aligned.
- Target picture size (stDestSize)
  - stDestSize must be set and its value must be within the region of the picture output by the external ADC. Otherwise, the VI device cannot work properly.
  - When the input picture is in interlaced mode, the capture height is set to the height of a field or two fields based on the field mode and the capture height must be a multiple of 4 in two-field capture mode.
  - The minimum target picture size is 32x32.
- Field capture selection (enCapSel)
  - enCapSel is used to capture only one field when the original pictures are input in interlaced mode.
  - Only two fields or the bottom field can be captured to avoid screen flicker.
- Pixel format (enPixFormat): The pixel formats semi-planar422 and semi-planar420 are supported.
- Chrominance resampling (bChromaResample): It is used to determine whether chrominance resampling is performed.
- Source frame rate (s32SrcFrameRate): If you do not want to control the frame rate, set s32SrcFrameRate to –1.
- Target frame rate (s32FrameRate): If you do not want to control the frame rate, set s32FrameRate to –1. If you want to control the frame rate, you must set the source frame rate and target frame rate in sequence, and the target frame rate must be lower than the source frame rate.
- Interlaced pictures do not support mirror and flip. If the function is enabled, the sequence of capturing top and bottom fields is reversed, which may affect the picture quality when pictures are processed by other modules. As the MPP does not check whether the function is enabled, you need to disable the function in interlaced mode.

[Example]

See the example of HI_MPI_VI_SetDevAttr.

[See Also]

HI_MPI_VI_GetChnAttr

HI_MPI_VI_GetChnAttr

[Description]

Obtains the attributes of a VI channel.

[Syntax]

```
HI_S32 HI_MPI_VI_GetChnAttr(VI_CHN ViChn,VI_CHN_ATTR_S *pstAttr);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input
pstAttr	Pointer to the attributes of a VI channel Dynamic attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Channel ID Range
Hi3531/Hi3532	[0, VIU_MAX_CHN_NUM – 2). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM – 2) are invalid.
Hi3521/Hi3520A/Hi3520D/ Hi3515A/Hi3515C	[0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	[0, VIU_MAX_PHYCHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

You must set the attributes of a VI channel before calling this MPI. Otherwise, a code indicating failure is returned.

[Example]

```
HI_S32 s32ret;
VI_CHN ViChn = 0;
VI_CHN_ATTR_S stChnAttr;

/* first enable vi device and vi chn */

/* get channel attribute for vi chn */
s32ret = HI_MPI_VI_GetChnAttr(ViChn, &stChnAttr);
if (HI_SUCCESS != s32ret)
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
{
printf("get vi chn attr err:0x%x\n", s32ret);
return s32ret;
}
```

[See Also]

HI_MPI_VI_SetChnAttr

HI_MPI_VI_SetChnMinorAttr

[Description]

Sets the secondary attribute of a VI channel.

[Syntax]

```
HI_S32 HI_MPI_VI_SetChnMinorAttr (VI_CHN ViChn,const VI_CHN_ATTR_S
*pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel Value range: [0, VIU_MAX_CHN_NUM)	Input
pstAttr	Pointer to the secondary attribute of VI channels	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Secondary Attribute Supported	Limitation
Hi3531/Hi3532	No	Not supported
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	Yes	It is meaningless to set the secondary attribute stCapRect. The width defined in stDestSize can be the same as or half of the width defined in the primary attribute stCapRect. In single-field capture mode, the height defined in stDestSize must be half of the height defined in the primary

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Secondary Attribute Supported	Limitation
		attribute stCapRect. In dual-field capture mode, the heights defined in stDestSize and stCapRect must be the same.
Hi3518/Hi3516C	No	Not supported

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- By default, the VI channel captures consecutive video frames based on the resolution configured by calling HI_MPI_VI_SetChnAttr. HI_MPI_VI_SetChnMinorAttr is used to set another resolution. Based on the configured frame rate, the VI channel captures pictures with two resolutions in turn.
- The channel secondary attribute is defined relative to the channel primary attribute. The primary attribute is configured by calling HI_MPI_VI_SetChnAttr. The primary and secondary attributes take effect only when the frame rate of the VI channel is controlled. The VI channel captures the pictures with the size defined in the major attribute based on the configured target frame rate, and then captures other frames with the size defined in the secondary attribute.

  The following example assumes that the source frame rate is 25 fps, the target frame rate is 5 fps, the picture size defined in the primary attribute is D1, and the picture size defined in the secondary attribute is CIF. In this case, the VI channel captures five frames of D1 pictures and 20 frames of CIF pictures. Table 3-16 describes the parameters for the primary attribute and secondary attribute.

Table 3-16 Parameters for the primary attribute and secondary attribute

Attribute	AD	Primary Attribute	Secondary Attribute
Resolution	N/A	SetChnAttr	SetChnMinorAttr
Frame rate	SrcFrameRate	FrameRate	SrcFrameRate − FrameRate

- If the VI frame rate is set but the channel secondary attribute is not set, only the pictures with the size defined in the primary attribute are output based on the target frame rate.
- If the channel secondary attribute is set and you want to stop capturing the pictures with the size defined in the secondary attribute, you can call HI_MPI_VI_ClearChnMinorAttr to clear the secondary attribute settings.
- If the source frame rate defined in the primary attribute is greater than 0 and the target frame rate is 0, only the pictures with the size defined in the secondary attribute are output.
- If FrameRate is the same as SrcFrameRate, only the pictures with the size defined in the primary attribute are output.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- You can dynamically modify the secondary attribute of a VI channel after the channel is enabled.

[Example]

```
VI_CHN_ATTR_S stAttr;
/* set public attribute of VI device*/
/* ... ... */

/* enable VI device*/
/* ... ... */
StAttr.s32SrcFrameRate = 25;
StAttr.s32FrameRate = 5;

/* set main attribute of VI channel, size is D1 */
/* ... ... */

HI_MPI_VI_GetChnAttr(ViChn,&stAttr);
stAttr.enCapSel = VI_CAPSEL_BOTTOM;
stAttr.bDownScale = HI_TRUE;

/* set secondary attribute of VI channel, size is CIF */
if (HI_MPI_VI_SetChnMinorAttr(ViChn,&stAttr))
{
BBIT_ERR("set chn attr ex fail\n");
return -1;
}
/* enable VI channel*/
/* ... ... */

/* if you not need minor attr capture */
if (HI_MPI_VI_ClearChnMinorAttr(ViChn))
{
BBIT_ERR("clear chn minor attr fail\n");
return -1;
}
```

[See Also]

HI_MPI_VI_GetChnMinorAttr

HI_MPI_VI_GetChnMinorAttr

[Description]

Obtains the secondary attribute of a VI channel.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
HI_S32 HI_MPI_VI_GetChnMinorAttr(VI_CHN ViChn, VIDEO_FRAME_INFO_S
*pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel Value range: [0, VIU_MAX_CHN_NUM)	Input
pstAttr	Pointer to the attribute of a VI channel Dynamic attribute.	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Secondary Attribute Supported
Hi3531/Hi3532	No
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	Yes
Hi3518/Hi3516C	No

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

Before obtaining the attribute of a VI channel, you must set the attribute. Otherwise, HI_ERR_VI_FAILED_NOTCONFIG is returned.

[Example]

See the example of HI_MPI_VI_SetChnMinorAttr.

[See Also]

HI_MPI_VI_SetChnMinorAttr

HI_MPI_VI_ClearChnMinorAttr

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Clears the secondary attribute settings of a VI channel.

[Syntax]

```
HI_S32 HI_MPI_VI_ClearChnMinorAttr(VI_CHN ViChn);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel Value range: [0, VIU_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Secondary Attribute Supported
Hi3531/Hi3532	No
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	Yes
Hi3518/Hi3516C	No

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- You can call this MPI to clear the secondary attribute settings of a VI channel.
- This MPI can be called when a VI channel is enabled.

[Example]

See the example of HI_MPI_VI_SetChnMinorAttr.

[See Also]

- HI_MPI_VI_SetChnMinorAttr
- HI_MPI_VI_GetChnMinorAttr

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VI_EnableChn

[Description]

Enables a VI channel.

[Syntax]

```
HI_S32 HI_MPI_VI_EnableChn(VI_CHN ViChn);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Channel ID Range
Hi3531/Hi3532	[0, VIU_MAX_CHN_NUM – 2). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM – 2) are invalid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	[0, VIU_MAX_CHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- You must set the attributes of a VI channel and enable the VI device bound to the VI channel before enabling the VI channel.
- If a VI channel is bound to a VO channel or VENC channel, video data is obtained after this MPI is called successfully.
- A VI channel can be enabled repeatedly without any failure.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

See the example of HI_MPI_VI_SetDevAttr.

[See Also]

HI_MPI_VI_DisableChn

HI_MPI_VI_DisableChn

[Description]

Disables a VI channel.

[Syntax]

```
HI_S32 HI_MPI_VI_DisableChn(VI_CHN ViChn);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Channel ID Range
Hi3531/Hi3532	[0, VIU_MAX_CHN_NUM – 2). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM – 2) are invalid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	[0, VIU_MAX_CHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- After a VI channel is disabled, it stops capturing VI data. If the VI channel is bound to a VO channel or a VENC channel, the VO channel or VENC channel does not receive pictures.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- A VI channel can be disabled repeatedly without any failure.

[Example]

See the example of HI_MPI_VI_SetDevAttr.

[See Also]

HI_MPI_VI_EnableChn

HI_MPI_VI_EnableChnInterrupt

[Description]

Enables a VI channel to respond to the hardware interrupt handler.

[Syntax]

```
HI_S32 HI_MPI_VI_EnableChnInterrupt(VI_CHN ViChn);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	• As primary and secondary channels share the interrupt mask configuration, if a primary channel and its secondary channel are enabled and any of them is enabled to respond to the hardware interrupt handler, the other channel is also enabled to respond to the hardware interrupt handler. • Channel ID range: [0, VIU_MAX_CHN_NUM − 2). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM − 2) are invalid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	Channel ID range: [0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	Channel ID range: [0, VIU_MAX_PHYCHN_NUM)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- HI_MPI_VI_EnableChnInterrupt is an advanced interface, and it is not called typically.
- When an application detects exceptions in front-end ADC timings, HI_MPI_VI_DisableChnInterrupt is called to forbid the VI channel to respond to the hardware interrupt handler while the function of inserting user pictures is enabled. This reduces the system resources that are consumed when hardware reports a large amount of interrupts due to timing exceptions. When it is detected that the front-end ADC timings are normal, call HI_MPI_VI_EnableChnInterrupt to enable the VI channel to respond to the hardware interrupt handler again.
- Enable a VI channel before calling HI_MPI_VI_EnableChnInterrupt. Otherwise, an error is returned.
- HI_MPI_VI_EnableChnInterrupt can be called repeatedly, and no code indicating failure is returned.

[Example]

None

[See Also]

HI_MPI_VI_DisableChnInterrupt

HI_MPI_VI_DisableChnInterrupt

[Description]

Forbids a VI channel to respond to the hardware interrupt handler.

[Syntax]

```
HI_S32 HI_MPI_VI_DisableChnInterrupt(VI_CHN ViChn);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Description
Hi3531/Hi3532	• As primary and secondary channels share the interrupt mask configuration, if a primary channel and its secondary channel are enabled and any of them is forbidden to respond to the hardware interrupt handler, the other channel is also forbidden to respond to the hardware interrupt handler. • Channel ID range: [0, VIU_MAX_CHN_NUM − 2). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM − 2) are invalid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/ Hi3515C	Channel ID range: [0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	Channel ID range: [0, VIU_MAX_PHYCHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- HI_MPI_VI_DisableChnInterrupt is an advanced interface, and it is not called typically.
- When an application detects exceptions in front-end ADC timings, HI_MPI_VI_DisableChnInterrupt is called to forbid the VI channel to respond to the hardware interrupt handler while the function of inserting user pictures is enabled. This reduces the system resources that are consumed when hardware reports a large amount of interrupts due to timing exceptions. When it is detected that the front-end ADC timings are normal, call HI_MPI_VI_EnableChnInterrupt to enable the VI channel to respond to the hardware interrupt handler again.
- If no VI channel is enabled, you do not need to forbid the VI channel to respond to the hardware interrupt handler. In this case, a code indicating success is returned if you call HI_MPI_VI_DisableChnInterrupt.
- HI_MPI_VI_DisableChnInterrupt can be called repeatedly, and no code indicating failure is returned.

[Example]

None

[See Also]

HI_MPI_VI_EnableChnInterrupt

HI_MPI_VI_SetFrameDepth

[Description]

Sets the maximum depth of the buffer for storing the obtained VI pictures.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_VI_SetFrameDepth(VI_CHN ViChn, HI_U32 u32Depth);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input
u32Depth	Maximum depth of the VI picture to be obtained The default value is 0. Value range: [0, 8]	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Channel ID Range
Hi3531/Hi3532	[0, VIU_MAX_CHN_NUM). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM – 2) are invalid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	[0, VIU_MAX_CHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- This MPI is used to set the number of video frames buffered in a VI channel. If the number of buffered video frames is not set to 0, consecutive video frames can be obtained.
- If u32Depth is set to 0, the system does not buffer pictures for the VI channel. Therefore, you cannot obtain pictures from this VI channel. By default, the system does not buffer pictures for the VI channel. That is, the default value of u32Depth is 0.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- If u32Depth is set to an integer greater than 0, the system buffers u32Depth pictures for the VI channel. You can obtain the pictures by calling HI_MPI_VI_GetFrame. Note the following cases:
  - Pictures are not fetched.

    The system automatically replaces old pictures with latest pictures. This ensures that users can obtain the latest u32Depth pictures.
  - Pictures are fetched for u32Depth consecutive times and the video buffer (VB) is not released.

    The system automatically stops buffering new VI picture because the system cannot obtain the VB. As a result, users cannot obtain new VI pictures. You are advised to call the release MPI after you call the obtain MPI.
  - The speed of obtaining and releasing pictures is lower than the speed of generating pictures in the VI channel.

    The system automatically updates the old pictures that are not fetched. This ensures that the buffered pictures are the latest ones. As the speed of obtaining pictures cannot be ensured, the obtained pictures may be inconsecutive.
- When the system buffers u32Depth pictures for each VI channel, the system occupies the VBs in the MPP. You need to set sufficient VBs by calling HI_MPI_VB_SetConf. Otherwise, the VIU cannot capture pictures properly because the system may occupy too many VBs for buffering pictures. As a result, users cannot obtain VI pictures. You are advised to set u32Depth dynamically. If you do not need to obtain pictures from VI channels, you can set u32Depth to 0 to reduce the number of VBs occupied by VI channels. When you want to obtain pictures, you can set u32Depth. Then you can obtain consecutive VI pictures generated after the time when you set u32Depth.

[Example]

See the example of HI_MPI_VI_GetFrame.

[See Also]

HI_MPI_VI_GetFrameDepth

HI_MPI_VI_GetFrameDepth

[Description]

Obtains the maximum depth of the buffer for storing the obtained VI pictures.

[Syntax]

```
HI_S32 HI_MPI_VI_GetFrameDepth(VI_CHN ViChn, HI_U32 *pu32Depth);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input
pu32Depth	U32 pointer	Output

[Return Value]

Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Channel ID Range
Hi3531/Hi3532	[0, VIU_MAX_CHN_NUM). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM – 2) are invalid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	[0, VIU_MAX_CHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

None

HI_MPI_VI_GetFrame

[Description]

Obtains pictures captured by a VI channel.

[Syntax]

```
HI_S32 HI_MPI_VI_GetFrame(VI_CHN ViChn, VIDEO_FRAME_INFO_S *pstFrameInfo);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input
pstFrameInfo	Pointer to the structure of the VI frame information	Output

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Channel ID Range
Hi3531/Hi3532	[0, VIU_MAX_CHN_NUM). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM – 2) are invalid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	[0, VIU_MAX_CHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the information about pictures in a specified VI channel. The picture information includes the picture width, height, pixel format, PTS, and physical addresses of YUV components.
- HI_MPI_VI_GetFrame is available only after a channel is enabled.
- The occupied buffer can be released after HI_MPI_VI_GetFrame is called for multiple times. You are advised to call the release MPI after you call the obtain MPI.
- The physical address information is obtained from the VB used by the MPP. Therefore, the VB must be released after use by calling HI_MPI_VI_ReleaseFrame.
- pstFrameInfo -> stVFrame .u32PhyAddr[0] points to the physical address of the Y component and pstFrameInfo -> stVFrame .u32PhyAddr[1] points to the physical address of the C component.
- The default timeout period for this MPI is 2s. That is, this MPI is returned if no pictures are obtained in 2s.

[Example]

```
HI_S32 s32ret;
VI_CHN ViChn = 0;
VIDEO_FRAME_INFO_S stFrame;
HI_U32 u32Depth;
/* set max depth */
u32Depth = 10;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
s32ret = HI_MPI_VI_SetFrameDepth(ViChn, u32Depth);
if (HI_SUCCESS != s32ret)
{
printf("set max depth err:0x%x\n", s32ret);
return s32ret;
}

/* get video frame from vi chn */
s32ret = HI_MPI_VI_GetFrame(ViChn, &stFrame)
if (HI_SUCCESS != s32ret)
{
    printf("get vi frame err:0x%x\n", s32ret);
    return s32ret;
}
/* deal with video frame ... */
/* release video frame */
(void)HI_MPI_VI_ReleaseFrame(ViChn, &stFrame);
```

[See Also]

- HI_MPI_VI_ReleaseFrame
- HI_MPI_VI_GetFrameTimeOut

HI_MPI_VI_GetFrameTimeOut

[Description]

Obtains pictures captured by a VI channel within the configured period.

[Syntax]

```
HI_S32 HI_MPI_VI_GetFrameTimeOut(VI_CHN ViChn, VIDEO_FRAME_INFO_S
*pstFrameInfo, HI_U32 u32MilliSec);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input
pstFrameInfo	Pointer to the structure of the VI frame information	Output
u32MilliSec	Timeout period, in the unit of ms	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
	details, see section 3.6 "Error Codes."

[Difference]

Chip	Channel ID Range
Hi3531/Hi3532	[0, VIU_MAX_CHN_NUM). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM – 2) are invalid.
Hi3521/Hi3520A/Hi3520D/ Hi3515A/Hi3515C	[0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	[0, VIU_MAX_CHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the information about pictures in a specified VI channel. The picture information includes the picture width, height, pixel format, PTS, and physical addresses for YUV components.
- This MPI is available only after a channel is enabled.
- The occupied buffer can be released after HI_MPI_VI_GetFrameTimeOut is called for multiple times. You are advised to call the release MPI after you call the obtain MPI.
- The physical address information is obtained from the VB used by the MPP. Therefore, the VB must be released after use by calling HI_MPI_VI_ReleaseFrame.
- pstFrameInfo -> stVFrame .u32PhyAddr[0] points to the physical address of the Y component and pstFrameInfo -> stVFrame .u32PhyAddr[1] points to the physical address of the C component.
- The u32MilliSec parameter indicates the timeout period and its unit is ms. The MPI is returned if no pictures are obtained within the period defined by u32MilliSec. If u32MilliSec is set to 0, the block mode is used. In this case, the MPI is returned only after pictures are obtained.

[Example]

None

[See Also]

- HI_MPI_VI_ReleaseFrame
- HI_MPI_VI_GetFrame

HI_MPI_VI_ReleaseFrame

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Releases the buffer occupied by VI pictures.

[Syntax]

```
HI_S32 HI_MPI_VI_ReleaseFrame(VI_CHN ViChn, VIDEO_FRAME_INFO_S
*pstFrameInfo);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input
pstFrameInfo	Pointer to the storage structure of VI frames	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Channel ID Range
Hi3531/Hi3532	[0, VIU_MAX_CHN_NUM – 2). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM – 2) are invalid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	[0, VIU_MAX_CHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- Ensure that the information about the pstFrameInfo structure is the same as the obtained information. Otherwise, the buffer fails to be released.
- After obtaining original pictures by calling HI_MPI_VI_GetFrame, you must call HI_MPI_VI_ReleaseFrame to release the buffer for storing the original pictures. That is, HI_MPI_VI_ReleaseFrame must be called if HI_MPI_VI_GetFrame or HI_MPI_VI_GetFrameTimeOut is called.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

See the example of HI_MPI_VI_GetFrame.

[See Also]

- HI_MPI_VI_GetFrame
- HI_MPI_VI_GetFrameTimeOut

HI_MPI_VI_SetUserPic

[Description]

Sets a user picture that is used as the picture inserted when there is no video signal.

[Syntax]

```
HI_S32 HI_MPI_VI_SetUserPic(VI_CHN ViChn, VI_USERPIC_ATTR_S *pstUsrPic);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input
pstUsrPic	Pointer to the structure of the user picture information	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Channel ID Range
Hi3531/Hi3532	[0, VIU_MAX_CHN_NUM – 2). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM – 2) are invalid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	[0, VIU_MAX_PHYCHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- This MPI supports two user picture types: YUV picture and background picture only with a color.
- This MPI is used to replace the video picture output by the VI channel with a user-defined YUV picture or a background picture only with a color rather than the video data from the codec. This MPI must work with HI_MPI_VI_EnableUserPic and HI_MPI_VI_DisableUserPic.

  Currently, the IDs of the VI device and the VI channel are not used. That is, when you set a user picture, the setting is valid for all VI channels rather than a specified channel.
- After setting a user picture, you can insert a user picture into a specified VI channel. Then the VI channel outputs the configured YUV picture or a background picture only with a color. Typically, the inserted picture is displayed and indicates no video when video signals are lost.

  The width and height of the user picture must be within [32, 4096]. Typically, the size of the user picture should be the same as that of the VI channel. If the sizes are inconsistent, the user picture is automatically scaled to fit into the VI channel. If you want to capture pictures with two resolutions from VI channels, you are advised to set the user picture size to the larger resolution and capture mode to interlaced mode. Otherwise, the screen flickers when pictures are previewed on a VO device.
- For the video frame information structure of a user picture, you need to set the picture width, picture height, picture stride, YUV format, and physical addresses of the Y component and C component. You can obtain a video buffer from the public buffer pool of the MPP, obtain the physical address for storing the YUV data by reading the buffer information, and map the physical address into the user space. Then you can fill YUV data into the buffer. Note that only semi-planar YUV420 or semi-planarYUV422 data is supported. Therefore, you must fill the Y component and interleaved data of U and V components in sequence. In little endian mode, the V and U components are filled in sequence.

  When the function of inserting a user picture is enabled, the VIU uses the physical address in the user picture frame information. Therefore, after setting a user picture, do not release or destroy the corresponding buffer unless the buffer is no longer used. To modify the information about the picture, you can call HI_MPI_VI_SetUserPi again to configure another video buffer.
- You can dynamically call this MPI when you have enabled a VI channel or the function of inserting a user picture.

[Example]

```
HI_S32  s32ret;
   HI_U32  u32Width;
   HI_U32  u32Height;
   HI_U32  u32LStride;
   HI_U32  u32CStride;
   HI_U32  u32LumaSize;
   HI_U32  u32ChrmSize;
   HI_U32  u32Size;
   VB_BLK  VbBlk;
   HI_U32  u32PhyAddr;
   HI_U8   *pVirAddr;
    /*You need to obtain the picture width and height.*/
   u32LumaSize = (u32LStride * u32Height);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    u32ChrmSize = (u32CStride * u32Height) >> 2;/* 420*/
    u32Size = u32LumaSize + (u32ChrmSize << 1);

    /*Obtain video buffer block form common pool.*/
    VbBlk = HI_MPI_VB_GetBlock(VB_INVALID_POOLID, u32Size);
    if (VB_INVALID_HANDLE == VbBlk)
    {
    return -1;
    }
    /*Obtain physical address*/
    u32PhyAddr = HI_MPI_VB_Handle2PhysAddr(VbBlk);
    if (0 == u32PhyAddr)
    {
return -1;
    }

    /* mmap physical address to virtual address*/
    /* ... ... */

    /*Obtain the pool ID.*/
    pstVFrameInfo->u32PoolId = HI_MPI_VB_Handle2PoolId(VbBlk);
    if (VB_INVALID_POOLID == pstVFrameInfo->u32PoolId)
    {
    return -1;
    }

    pstVFrameInfo->stVFrame.u32PhyAddr[0] = u32PhyAddr;
    pstVFrameInfo->stVFrame.u32PhyAddr[1] = pstVFrameInfo-
>stVFrame.u32PhyAddr[0] + u32LumaSize;
    pstVFrameInfo->stVFrame.u32PhyAddr[2] = pstVFrameInfo-
>stVFrame.u32PhyAddr[1] + u32ChrmSize;

    pstVFrameInfo->stVFrame.pVirAddr[0] = pVirAddr;
    pstVFrameInfo->stVFrame.pVirAddr[1] = pstVFrameInfo-
>stVFrame.pVirAddr[0] + u32LumaSize;
    pstVFrameInfo->stVFrame.pVirAddr[2] = pstVFrameInfo-
>stVFrame.pVirAddr[1] + u32ChrmSize;

    pstVFrameInfo->stVFrame.u32Width  = u32Width;
    pstVFrameInfo->stVFrame.u32Height = u32Height;
    pstVFrameInfo->stVFrame.u32Stride[0] = u32LStride;
    pstVFrameInfo->stVFrame.u32Stride[1] = u32CStride;
    pstVFrameInfo->stVFrame.u32Stride[2] = u32CStride;
    pstVFrameInfo->stVFrame.enPixelFormat =
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
PIXEL_FORMAT_YUV_SEMIPLANAR_420;

   /*Now you need obtain YUV Semi Planar Data, fill them into the virtual
address */
   /* ... ... */
   /* ... ... */

   /*Enable VI channel ... ... */

   /*First set user pic info*/
   s32ret = HI_MPI_VI_SetUserPic(0, pstVFrameInfo);
   if (s32ret)
   {
return -1;
   }

   /* ... ... */

   /*Enable the function of inserting user pictures as required.*/
   s32ret = HI_MPI_VI_EnableUserPic(0);
   if (s32ret)
   {
   return -1;
   }

   /* ... ... */

   /*Disable insert user pic if you do not need.*/
   s32ret = HI_MPI_VI_DisableUserPic(0, 0);
   if (s32ret)
   {
   return -1;
   }
```

[See Also]

None

HI_MPI_VI_EnableUserPic

[Description]

Enables the function of inserting a user picture.

[Syntax]

```
HI_S32 HI_MPI_VI_EnableUserPic(VI_CHN ViChn);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Channel ID Range
Hi3531/Hi3532	[0, VIU_MAX_CHN_NUM – 2). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM – 2) are invalid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	[0, VIU_MAX_PHYCHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- Before enabling the function of inserting a user picture, ensure that the frame information about the user picture is set.
- After the function of inserting user pictures is enabled, the current VI channel immediately outputs the configured user picture.
- This MPI can be repeatedly called.

[Example]

See the example of HI_MPI_VI_SetUserPic.

[See Also]

None

HI_MPI_VI_DisableUserPic

[Description]

Copyright © HiSilicon Technologies Co., Ltd

Disables the function of inserting a user picture.

[Syntax]

```
HI_S32 HI_MPI_VI_DisableUserPic(VI_CHN ViChn);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Channel ID Range
Hi3531/Hi3532	[0, VIU_MAX_CHN_NUM – 2). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM – 2) are invalid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	[0, VIU_MAX_PHYCHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- When the user picture is not required, you can call this MPI to restore the output of the original video data from the ADC.
- This MPI can be repeatedly called.

[Example]

See the example of HI_MPI_VI_SetUserPic.

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VI_EnableCascade

[Description]

Enables the cascade mode of a VI device.

[Syntax]

```
HI_S32 HI_MPI_VI_EnableCascade(VI_DEV ViDev);
```

[Parameter]

Parameter	Description	Input/Output
ViDev	ID of a VI device Value range: [0, VIU_MAX_DEV_NUM). **ViDev** is an even number.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	Only Dev0, Dev2, Dev4, and Dev6 and corresponding physical channels chn0, chn4, chn8, and chn12 support the cascade mode.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	The Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C does not support the cascade mode. If you call this MPI, an error occurs.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support the cascade mode. If you call this MPI, an error occurs.

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- Before enable the cascade mode of a device, you must set the device attributes. Otherwise, an error code indicating failure is returned.
- Only the BT.1120 progressive timing is supported in cascade mode.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- The cascade mode can be repeatedly enabled and no error code indicating failure is returned.

[Example]

```
HI_S32 s32Ret;
VI_DEV ViDev = 2;
VI_CHN ViChn = 4, ViCasChn = VI_CAS_CHN_1;
VI_DEV_ATTR_S stDevAttr;
VI_CHN_ATTR_S stChnAttr;

memset(&stDevAttr, 0, sizeof(VI_DEV_ATTR_S));
stDevAttr.enIntfMode = VI_MODE_BT1120_STANDARD;
stDevAttr.enWorkMode = VI_WORK_MODE_1Multiplex;
stDevAttr.au32CompMask[0] = 0xFF00;
stDevAttr.au32CompMask[1] = 0xFF;
stDevAttr.enScanMode = VI_SCAN_PROGRESSIVE;
stDevAttr.s32AdChnId[0] = -1;
stDevAttr.s32AdChnId[1] = -1;
stDevAttr.s32AdChnId[2] = -1;
stDevAttr.s32AdChnId[3] = -1;

s32Ret = HI_MPI_VI_SetDevAttr(ViDev, &stDevAttr);
if (s32Ret != HI_SUCCESS)
{
printf("Set dev attributes failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}
s32Ret = HI_MPI_VI_EnableDev(ViDev);
if (s32Ret != HI_SUCCESS)
{
printf("Enable dev failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}

s32Ret = HI_MPI_VI_EnableCascade(ViDev);
if (s32Ret != HI_SUCCESS)
{
printf("Enable dev cascade mode failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}

stChnAttr.stCapRect.s32X = 0;
stChnAttr.stCapRect.s32Y = 0;
stChnAttr.stCapRect.u32Width = 1920;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
stChnAttr.stCapRect.u32Height = 1080;
stChnAttr.stDestSize.u32Width = 1920;
stChnAttr.stDestSize.u32Height = 1080;
stChnAttr.enCapSel = VI_CAPSEL_BOTH;
stChnAttr.enPixFormat = PIXEL_FORMAT_YUV_SEMIPLANAR_422;
stChnAttr.bMirror = HI_FALSE;
stChnAttr.bFlip = HI_FALSE;
stChnAttr.bChromaResample = HI_FALSE
stChnAttr.s32SrcFrameRate = -1;
stChnAttr.s32FrameRate = -1;
s32Ret = HI_MPI_VI_SetChnAttr(ViChn,&stChnAttr);
if (s32Ret != HI_SUCCESS)
{
printf("Set chn attributes failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}
s32Ret = HI_MPI_VI_EnableChn(ViChn);
if (s32Ret != HI_SUCCESS)
{
printf("Enable chn failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}
/* here, we suppose it's VO_CAS_MODE_SINGLE cascade mode */
s32Ret = HI_MPI_VI_EnableCascadeChn(ViCasChn);
if (s32Ret != HI_SUCCESS)
{
printf("Enable cascade chn failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}
/* now, vi is capturing images, you can do something else*/

s32Ret = HI_MPI_VI_DisableCascadeChn(ViCasChn);
if (s32Ret != HI_SUCCESS)
{
printf("Disable cascade chn failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}
s32Ret = HI_MPI_VI_DisableChn(ViChn);
if (s32Ret != HI_SUCCESS)
{
printf("Disable chn failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}
s32Ret = HI_MPI_VI_DisableCascade(ViDev);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
if (s32Ret != HI_SUCCESS)
{
printf("Disable dev cascade mode failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}
s32Ret = HI_MPI_VI_DisableDev(ViDev);
if (s32Ret != HI_SUCCESS)
{
printf("Disable dev failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}
```

[See Also]

HI_MPI_VI_DisableCascade

HI_MPI_VI_DisableCascade

[Description]

Disables the cascade mode of a VI device.

[Syntax]

```
HI_S32 HI_MPI_VI_DisableCascade(VI_DEV ViDev);
```

[Parameter]

Parameter	Description	Input/Output
ViDev	ID of a VI device Value range: [0, VIU_MAX_DEV_NUM). **ViDev** is an even number.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Cascade Mode Supported
Hi3531/Hi3532	Yes
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	No

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Cascade Mode Supported
Hi3518/Hi3516C	No

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- You can disable the cascade mode of a VI device by calling this MPI only after all enabled extended cascade channels and physical channels are disabled.
- The cascade mode can be repeatedly disabled without any failure.

[Example]

See the sample of HI_MPI_VI_EnableCascade.

[See Also]

HI_MPI_VI_EnableDev

HI_MPI_VI_EnableCascadeChn

[Description]

Enables an extended VI cascade channel.

[Syntax]

```
HI_S32 HI_MPI_VI_EnableCascadeChn(VI_CHN ViChn);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of an extended VI cascade channel Value range: {VI_CAS_CHN_1, VI_CAS_CHN_2}	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Cascade Mode Supported
Hi3531/Hi3532	Yes
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	No
Hi3518/Hi3516C	No

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- Before calling HI_MPI_VI_EnableCascadeChn, you must call HI_MPI_VI_EnableCascade to enable the cascade mode.
- The extended cascade channel is a virtual software channel. You must enable a physical cascade channel before enabling an extended cascade channel by calling I_MPI_VI_EnableCascadeChn.
- If the cascade mode is enabled but no extended cascade channel is enabled, the VIU does not capture and transmit pictures.
- The extended cascade channel VI_CAS_CHN_1 always receives the pictures in single-channel mode or odd frames in dual-channel cascade mode. The extended cascade channel VI_CAS_CHN_2 always receives even frames in dual-channel cascade mode.
- If dual-channel cascade video data is input when the cascade mode is disabled, the VIU does not differentiate the data from dual channels. That is, the data from dual channels is captured and transmitted as the data from a channel.
- An extended cascade channel can be repeatedly enabled and no error code indicating failure is returned.

[Example]

See the sample of HI_MPI_VI_EnableCascade.

[See Also]

HI_MPI_VI_DisableCascadeChn

HI_MPI_VI_DisableCascadeChn

[Description]

Disables an extended VI cascade channel.

[Syntax]

```
HI_S32 HI_MPI_VI_DisableCascadeChn(VI_CHN ViChn);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of an extended VI cascade channel Value range: {VI_CAS_CHN_1, VI_CAS_CHN_2}	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Cascade Mode Supported
Hi3531/Hi3532	Yes
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	No
Hi3518/Hi3516C	No

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- After an extended VI cascade channel is disabled, the corresponding physical VI channel stops capturing even frames from this extended VI cascade channel. If a VO or VDEC channel is bound to the extended cascaded channel, the VO or VDEC channel stops cannot receive pictures.
- An extended cascade channel can be repeatedly disabled and no error code indicating failure is returned.

[Example]

See the sample of HI_MPI_VI_EnableCascade.

[See Also]

HI_MPI_VI_EnableChn

HI_MPI_VI_ChnBind

[Description]

Binds a VI channel. This MPI is an advanced interface.

[Syntax]

```
HI_S32 HI_MPI_VI_ChnBind(VI_CHN ViChn, const VI_CHN_BIND_ATTR_S
*pstChnBindAttr);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input
pstChnBindAttr	Pointer to the binding attribute of a VI channel Static attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Binding Relationship Setting Supported	Channel ID Range
Hi3531/Hi3532	Yes	[0, (VIU_MAX_CHN_NUM –2)/2)
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	No	N/A
Hi3518/Hi3516C	No	N/A

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- This MPI is called only when you want to change the default binding relationship or bind multiple channels to the same {Dev, Way}. Before calling this MPI, you must cancel the default binding relationship by calling HI_MPI_VI_ChnBind.
- You must set the attributes of a VI device before binding the device to a VI channel.
- A VI channel cannot be bound repeatedly. If a VI channel is bound, you can bind it again only after it is unbound.
- The binding relationship is a static attribute. You must disable primary and secondary channels before configuring the binding relationship.

[Example]

None

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

HI_MPI_VI_ChnUnBind

[Description]

Unbinds a VI channel. This MPI is an advanced interface.

[Syntax]

```
HI_S32 HI_MPI_VI_ChnUnBind(VI_CHN ViChn);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Binding Relationship Setting Supported	Channel ID Range
Hi3531/Hi3532	Yes	[0, (VIU_MAX_CHN_NUM – 2)/2)
Hi3521/Hi3520A/Hi3520D/ Hi3515A/Hi3515C	No	N/A
Hi3518/Hi3516C	No	N/A

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- This MPI is called to cancel the binding relationship when you want to change the default binding relationship.
- The binding relationship can be canceled only when primary and secondary channels are disabled.
- Only the primary channel ID can be used to cancel the binding relationship.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Example]

None

[See Also]

None

HI_MPI_VI_GetChnBind

[Description]

Obtains the binding relationship of a VI channel.

[Syntax]

```
HI_S32 HI_MPI_VI_GetChnBind(VI_CHN ViChn, VI_CHN_BIND_ATTR_S
*pstChnBindAttr);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input
pstChnBindAttr	Pointer to the binding attribute of a VI channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Binding Relationship Setting Supported	Channel ID Range
Hi3531/Hi3532	Yes	[0, (VIU_MAX_CHN_NUM – 2)/2)
Hi3521/Hi3520A/Hi3520D /Hi3515A/Hi3515C	No	N/A
Hi3518/Hi3516C	No	N/A

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

The default binding relationship of a VI channel can be obtained only after you set the attributes of the device port corresponding to the VI channel. Note that only the primary channel ID can be used to obtain the binding relationship.

[Example]

None

[See Also]

None

HI_MPI_VI_GetFd

[Description]

Obtains the device FD corresponding to a VI channel.

[Syntax]

```
HI_S32 HI_MPI_VI_GetFd(VI_CHN ViChn);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input

[Return Value]

Return Value	Description
Positive number	Valid return value
Zero or negative number	Invalid return value

[Difference]

Chip	Channel ID Range
Hi3531/Hi3532	Value range: [0, VIU_MAX_CHN_NUM – 2). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM – 2) are invalid.
Hi3521/Hi3520A/Hi3520D/Hi3515A /Hi3515C	[0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	[0, VIU_MAX_CHN_NUM)

[Error Code]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

After obtaining the FD, you can obtain the video frames from multiple channels by calling the select function.

[Example]

```
VI_CHN ViChn = 0;
HI_S32 s32ViFd = 0;

/* enable vi dev and vi chn */
/* get video frame from vi chn */
s32ViFd = HI_MPI_VI_GetFd(ViChn);
if (s32ViFd <= 0)
{
return HI_FAILURE;
}
```

[See Also]

None

HI_MPI_VI_Query

[Description]

Queries the information about a VI channel such as the interrupt count and average frame rate.

[Syntax]

```
HI_S32 HI_MPI_VI_Query(VI_CHN ViChn, VI_CHN_STAT_S *pstStat);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a VI channel	Input
pstStat	Pointer to the structure of the channel information	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Difference]

Chip	Channel ID Range
Hi3531/Hi3532	Value range: [0, VIU_MAX_CHN_NUM – 2). The odd values within [VIU_MAX_CHN_NUM/2, VIU_MAX_CHN_NUM – 2) are invalid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/ Hi3515C	[0, VIU_MAX_CHN_NUM)
Hi3518/Hi3516C	[0, VIU_MAX_PHYCHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- By calling this MPI, you can query the interrupt count, channel enable status, average frame rate, lost interrupt count, number of times that VBs fail to be obtained, picture width, and picture height.
- You can also query the current interrupt count by calling this MPI to check whether interrupts are generated.
- The frame rate queried by calling this MPI is the average frame rate that is calculated every 10 seconds. Therefore, the average frame rate is inaccurate.
- If the queried lost interrupt count continuously increases, an exception occurs in the VI channel.

[Example]

None

[See Also]

None

HI_MPI_VI_SetFlashConfig

[Description]

Sets the camera flash.

[Syntax]

```
HI_S32 HI_MPI_VI_SetFlashConfig(VI_DEV ViDev,const VI_FLASH_CONFIG_S
*pstFlashConfig);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
ViDev	ID of a VI device	Input
pstFlashConfig	Pointer to camera flash settings	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Camera Flash Support	Device ID Range
Hi3531/Hi3532	No	None
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	No	None
Hi3518/Hi3516C	Yes	[0, VIU_MAX_DEV_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- The camera flash works in single-blink or multi-blink mode. In multi-blink mode, u32Interval must be greater than u32CapFrmIndex.
- For details about camera flash settings, see VI_FLASH_CONFIG_S.

[Example]

```
HI_S32 s32Ret;
VI_DEV ViDev = 0;
VI_FLASH_CONFIG_S stFlashConfig;

/* first enable vi device and vi chn */

/* Init flash config */
stFlashConfig.enFlashMode = VI_FLASH_ONCE;
stFlashConfig.u32StartTime = 100;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
stFlashConfig.u32Duration = 400;
stFlashConfig.u32CapFrmIndex = 2;

/* Set configuration for vi flash */
s32Ret = HI_MPI_VI_SetFlashConfig(ViDev, &stFlashConfig);
if (s32Ret != HI_SUCCESS)
{
   printf("Set flash config failed with error code %#x!\n", s32Ret);
   return HI_FAILURE;
}

/* Get configuration for vi flash */
s32Ret = HI_MPI_VI_GetFlashConfig(ViDev, &stFlashConfig);
if (s32Ret != HI_SUCCESS)
{
   printf("Get flash config failed with error code %#x!\n", s32Ret);
   return HI_FAILURE;
}
s32Ret = HI_MPI_VI_FlashTrigger(ViDev, HI_TRUE);
if (s32Ret != HI_SUCCESS)
{
   printf("Trigger flash failed with error code %#x!\n", s32Ret);
   return HI_FAILURE;
}
```

[See Also]

- HI_MPI_VI_GetFlashConfig
- HI_MPI_VI_FlashTrigger

HI_MPI_VI_GetFlashConfig

[Description]

Obtains camera flash settings.

[Syntax]

```
HI_S32 HI_MPI_VI_GetFlashConfig(VI_DEV ViDev,VI_FLASH_CONFIG_S
*pstFlashConfig);
```

[Parameter]

Parameter	Description	Input/Output
ViDev	ID of a VI device	Input
pstFlashConfig	Pointer to camera flash settings	Output

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Camera Flash Support	Device ID Range
Hi3531/Hi3532	No	None
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	No	None
Hi3518/Hi3516C	Yes	[0, VIU_MAX_DEV_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

Before obtaining camera flash settings, you must set the camera flash. Otherwise, an error code indicating failure is returned.

[Example]

None

[See Also]

- HI_MPI_VI_SetFlashConfig
- HI_MPI_VI_FlashTrigger

HI_MPI_VI_FlashTrigger

[Description]

Enables or disables the camera flash.

[Syntax]

```
HI_S32 HI_MPI_VI_FlashTrigger(VI_DEV ViDev, HI_BOOL bEnable);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
ViDev	ID of a VI device	Input
bEnable	Camera flash enable	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Camera Flash Support	Device ID Range
Hi3531/Hi3532	No	None
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	No	None
Hi3518/Hi3516C	Yes	[0, VIU_MAX_DEV_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

Before enabling or disabling the camera flash, you must set it. Otherwise, an error code indicating failure is returned.

[Example]

None

[See Also]

- HI_MPI_VI_SetFlashConfig
- HI_MPI_VI_GetFlashConfig

HI_MPI_VI_SetExtChnAttr

[Description]

Sets the attributes of an extended channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_VI_SetExtChnAttr(VI_CHN ViChn, const VI_EXT_CHN_ATTR_S
*pstExtChnAttr);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of an extended VI channel	Input
pstExtChnAttr	Pointer to extended channel attributes	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Extended Channel Support	Range of the Extended Channel ID
Hi3531/Hi3532	No	None
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	No	None
Hi3518/Hi3516C	Yes	[VIU_EXT_CHN_START, VIU_MAX_CHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

The following describes extended channel attributes:

- Target picture size (stDestSize)
  - This attribute must be set and its value cannot be less than the minimum picture size or greater than the maximum picture size supported by the extended channel after scaling. Otherwise, the extended channel has no output. The minification for the extended channel is less than 13. When the input data format is semi-planar422 and output data format is semi-planar420, the extended channel supports at most x1/8 zoom out in the vertical direction. The width or height of the output picture of the extended channel is less than 4096 pixels.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

  - u32Width must be 2-pixel-aligned. In progressive capture mode, u32Height must be 2-pixel-aligned; in interlaced dual-field capture mode, u32Height must be 4-pixel-aligned.
  - Pixel format (enPixFormat): The pixel formats semi-planar422 and semi-planar420 are supported.
- Source frame rate (s32SrcFrameRate): If you do not want to control the frame rate, set s32SrcFrameRate to –1.
- Target frame rate (s32FrameRate): If you do not want to control the frame rate, set s32FrameRate to –1. If you want to control the frame rate, you must set the source frame rate and target frame rate in sequence, and the target frame rate must be lower than the source frame rate.
- The output frame rate of an extended channel is obtained by referring to its control frame rates and the frame rates of its parent channel. Assume that the source frame rate of a sensor is 30 frame/s, the parent channel of the extended channel is physical channel 0, the source frame rate and target frame rate of channel 0 are 30 frame/s and 10 frame/s respectively, and the source frame rate and target frame rate of the extended channel is 10 frame/s and 2 frame/s. The actual frame rate of the extended channel is 2 frame/s.
- The number of extended channels bound to a parent channel cannot exceed the allowed value. Otherwise, binding fails.

[Example]

```
HI_S32 s32Ret;
VI_CHN ViChn = VIU_EXT_CHN_START;
VI_EXT_CHN_ATTR_S stExtChnAttr;

/*First enable VI devices and VI channels.*/

/*Initialize extended channel attributes.*/
stExtChnAttr.enPixFormat = PIXEL_FORMAT_YUV_SEMIPLANAR_420;
stExtChnAttr.s32BindChn = 0;
stExtChnAttr.s32FrameRate = -1;
stExtChnAttr.s32SrcFrameRate = -1;
stExtChnAttr.stDestSize.u32Width = 720;
stExtChnAttr.stDestSize.u32Height = 576;

/*Set attributes of extended VI channels.*/
HI_MPI_VI_SetExtChnAttr(ViExtChn, &stExtChnAttr);
if (HI_SUCCESS != s32ret)
{
    printf("Set vi ext-chn attr err:0x%x\n", s32ret);
    return s32Ret;
}

/*Obtain attributes of extended VI channels.*/
s32Ret = HI_MPI_VI_GetExtChnAttr (ViChn, & stExtChnAttr);
if (HI_SUCCESS != s32ret)
{
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    printf("Get vi ext-chn attr err:0x%x\n", s32ret);
    return s32Ret;
}
```

[See Also]

HI_MPI_VI_GetExtChnAttr

HI_MPI_VI_GetExtChnAttr

[Description]

Obtains the attributes of an extended VI channel.

[Syntax]

```
HI_S32 HI_MPI_VI_GetExtChnAttr(VI_CHN ViChn, VI_EXT_CHN_ATTR_S
*pstExtChnAttr);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of an extended VI channel	Input
pstExtChnAttr	Pointer to extended channel attributes	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Extended Channel Support	Range of the Extended Channel ID
Hi3531/Hi3532	No	None
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	No	None
Hi3518/Hi3516C	Yes	[VIU_EXT_CHN_START, VIU_MAX_CHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Library file: libmpi.a

[Note]

Before obtaining extended channel attributes, you must set them. Otherwise, an error code indicating failure is returned.

[Example]

None

[See Also]

HI_MPI_VI_SetExtChnAttr

HI_MPI_VI_SetLDCAttr

[Description]

Sets the LDC attribute.

[Syntax]

```
HI_S32 HI_MPI_VI_SetLDCAttr(VI_CHN ViChn, const VI_LDC_ATTR_S
*pstLDCAttr);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a physical VI channel	Input
pstLDCAttr	Pointer to the LDC attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	LDC Support	Range of the Physical Channel ID
Hi3531/Hi3532	No	None
Hi3521/Hi3520A/Hi3520D/Hi3515A/ Hi3515C	No	None
Hi3518/Hi3516C	Yes	[0, VIU_MAX_PHYCHN_NUM)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- Call this MPI only after setting channel attributes.
- The LDC attributes can be dynamically changed.
- Only the semi-planar420 pictures output in non-field mode are supported.
- Only the LDC attributes of physical channels can be set.
- When LDC is enabled, you are advised to use the CoverEx region to replace the Cover region, because the Cover region may be distorted.

[Example]

```
HI_S32 s32Ret;
VI_LDC_ATTR_S stLDCAttr;

/*First enable VI devices and VI channel.*/

/*Initialize LDC attributes.*/
stLDCAttr.bEnable = HI_TRUE;
stLDCAttr.stAttr.enViewType = LDC_VIEW_TYPE_ALL;   //LDC_VIEW_TYPE_CROP;
stLDCAttr.stAttr.s32CenterXOffset = 0;
stLDCAttr.stAttr.s32CenterYOffset = 0;
stLDCAttr.stAttr.s32Ratio = 255;

/*Set LDC attributes.*/
s32Ret = HI_MPI_VI_SetLDCAttr(ViChn, &stLDCAttr);
if (HI_SUCCESS != s32ret)
{
    printf("Set vi LDC attr err:0x%x\n", s32ret);
    return s32Ret;
}

/*Obtain LDC attributes.*/
s32Ret = HI_MPI_VI_GetLDCAttr (ViChn, &stLDCAttr);
if (HI_SUCCESS != s32ret)
{
    printf("Get vi LDC attr err:0x%x\n", s32ret);
    return s32Ret;
}
```

[See Also]

HI_MPI_VI_GetLDCAttr

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VI_GetLDCAttr

[Description]

Obtains the LDC attribute.

[Syntax]

```
HI_S32 HI_MPI_VI_GetLDCAttr(VI_CHN ViChn, VI_LDC_ATTR_S *pstLDCAttr);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a physical VI channel	Input
pstLDCAttr	Pointer to the LDC attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	LDC Support	Range of the Physical Channel ID
Hi3531/Hi3532	No	None
Hi3521/Hi3520A/Hi3520D/Hi3515A/ Hi3515C	No	None
Hi3518/Hi3516C	Yes	[0, VIU_MAX_PHYCHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VI_SetLDCAttr

HI_MPI_VI_SetCSCAttr

[Description]

Sets the CSC attribute of a VI device.

[Syntax]

```
HI_S32 HI_MPI_VI_SetCSCAttr(VI_DEV ViDev, const VI_CSC_ATTR_S
*pstCSCAttr);
```

[Parameter]

Parameter	Description	Input/Output
ViDev	ID of a VI device	Input
pstCSCAttr	Pointer to CSC attributes	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	CSC Support	Device ID Range
Hi3531/Hi3532	No	None
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	No	None
Hi3518/Hi3516C	Yes	[0, VIU_MAX_DEV_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- Call this MPI after setting the device attributes.
- This MPI is used to adjust the luminance, hue, contrast, and saturation.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- For the HD sensor, set enViCscType to VI_CSC_TYPE_709; for the standard definition (SD) sensor, set enViCscType to VI_CSC_TYPE_601.
- The value range of luminance, hue, contrast, or saturation is [0, 100], and the default value is 50.
- Parameters of the CSC attribute are as follows:

Parameter	Description	Remarks
ext_csc_en	0: Default value. The adjustable luminance range is [0, 100]. 1: The adjustable luminance range is [−128, 128].	Applicable to the Hi3518 and Hi3516C
csc_ct_mode	0: Default value. The luminance changes according to the contrast adjustment. 1: The luminance does not change according to the contrast adjustment.	Applicable to the Hi3518 and Hi3516C
csc_tv_en	0: Default value. The value range of the CSC matrix coefficients is [0, 255]. 1: The value range of the CSC matrix coefficients is [16, 235].	

[Example]

```
HI_S32 s32Ret;
VI_CSC_ATTR_S stCscAttr;

/*First enable VI devices and VI channels.*/

/*Initialize CSC attributes.*/
stCscAttr.enViCscType = VI_CSC_TYPE_709;
stCscAttr.u32ContrVal = 50;
stCscAttr.u32HueVal = 50;
stCscAttr.u32LumaVal = 50;
stCscAttr.u32SatuVal = 50;

/*Set CSC attributes.*/
s32Ret = HI_MPI_VI_SetCSCAttr(ViDev, &stCscAttr);
if (HI_SUCCESS != s32ret)
{
    printf("Set vi CSC attr err:0x%x\n", s32ret);
    return s32Ret;
}

/*Obtain CSC attributes.*/
s32Ret = HI_MPI_VI_GetCSCAttr(ViDev, &stCscAttr);
if (HI_SUCCESS != s32ret)
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
{
    printf("Get vi CSC attr err:0x%x\n", s32ret);
    return s32Ret;
}
```

[See Also]

None

HI_MPI_VI_GetCSCAttr

[Description]

Obtains the CSC attribute of a VI device.

[Syntax]

```
HI_S32 HI_MPI_VI_GetCSCAttr(VI_DEV ViDev, VI_CSC_ATTR_S *pstCSCAttr);
```

[Parameter]

Parameter	Description	Input/Output
ViDev	ID of a VI device	Input
pstCSCAttr	Pointer to CSC attributes	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	CSC Support	Device ID Range
Hi3531/Hi3532	No	None
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	No	None
Hi3518/Hi3516C	Yes	[0, VIU_MAX_DEV_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

None

[Example]

None

[See Also]

None

HI_MPI_VI_SetRotate

[Description]

Sets the rotation attribute of VI pictures.

[Syntax]

```
HI_S32 HI_MPI_VI_SetRotate(VI_CHN ViChn, const ROTATE_E enRotate);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a physical VI channel	Input
enRotate	Rotation attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Rotation Support	Range of the Physical Channel ID
Hi3531/Hi3532	No	None
Hi3521/Hi3520A/Hi3520D /Hi3515A/Hi3515C	No	None
Hi3518A/Hi3516C	Yes	[0, VIU_MAX_PHYCHN_NUM)
Hi3518C/Hi3518E	No	None

[Requirement]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- The rotation attribute is a static attribute and cannot be dynamically modified. Set the rotation attribute after you set the channel attributes but before you enable a channel.
- Only the semi-planar420 pictures output in non-field mode can be rotated.
- Pictures can be rotated only by 90º, 180º, or 270º.
- Only the pictures in a physical channel can be rotated.
- To rotate a picture by 180º, you are advised to use the flip and mirror functions. This reduces the bandwidth usage and improves performance.
- After the rotation attribute is set, the width and height of the output pictures from a physical channel may change. However, the width and height obtained from the channel attributes are still configured values. For example, if a 720p input picture is rotated by 90º or 270º, the actual width and height of the output picture is (720, 1280), but the width and height (DestSize) obtained by calling HI_MPI_VI_GetChnAttr is (1280, 720).
- After the rotation attribute is set, the user picture size must be set based on the picture size after rotation.

[Example]

```
HI_S32 s32Ret;
VI_CHN ViChn = 0;
ROTATE_E enRotate;

/*First enable VI devices.*/

/*Set VI channel attributes.*/
s32Ret = HI_MPI_VI_SetChnAttr(ViChn,&stChnAttr);
if (s32Ret != HI_SUCCESS)
{
   printf("Set chn attributes failed with error code %#x!\n", s32Ret);
   return HI_FAILURE;
}

/*Set the rotation angle to 90°.*/
s32Ret = HI_MPI_VI_SetRotate(ViChn, ROTATE_90);
if (HI_SUCCESS != s32ret)
{
    printf("Set vi rotate err:0x%x\n", s32ret);
    return s32Ret;
}
s32Ret = HI_MPI_VI_EnableChn(ViChn);
if (s32Ret != HI_SUCCESS)
{
   printf("Enable chn failed with error code %#x!\n", s32Ret);
   return HI_FAILURE;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
}
/*Obtain rotation attributes.*/
s32Ret = HI_MPI_VI_GetRotate(ViChn, &enRotate);
if (HI_SUCCESS != s32ret)
{
    printf("Get vi rotate err:0x%x\n", s32ret);
    return s32Ret;
}
```

[See Also]

None

HI_MPI_VI_GetRotate

[Description]

Obtains the rotation attribute of VI pictures.

[Syntax]

```
HI_S32 HI_MPI_VI_GetRotate(VI_CHN ViChn, ROTATE_E *penRotate);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a physical VI channel	Input
penRotate	Pointer to the rotation attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Rotation Support	Range of the Physical Channel ID
Hi3531/Hi3532	No	None
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	No	None
Hi3518A/Hi3516C	Yes	[0, VIU_MAX_PHYCHN_NUM)
Hi3518C/Hi3518E	No	None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_VI_SetRotate

HI_MPI_VI_GetChnLuma

[Description]

Obtains the channel luminance statistics.

[Syntax]

```
HI_S32 HI_MPI_VI_GetChnLuma(VI_CHN ViChn, VI_CHN_LUM_S *pstLuma);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	ID of a physical VI channel	Input
pstLuma	Pointer to the luminance statistics	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	Luminance Statistics Support	Range of the Physical Channel ID
Hi3531/Hi3532	No	None
Hi3521/Hi3520A	No	None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Luminance Statistics Support	Range of the Physical Channel ID
Hi3518/Hi3516C/Hi3520D/Hi3515A /Hi3515C	Yes	[0, VIU_MAX_PHYCHN_NUM)

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

None

HI_MPI_VI_SetPortMode

[Description]

Sets the VI port mode.

[Syntax]

```
HI_S32 HI_MPI_VI_SetPortMode(VI_DEV ViDev, const VI_PORTMODE_E
enPortMode);
```

[Parameter]

Parameter	Description	Input/Output
ViDev	ID of a VI device	Input
enPortMode	Port mode	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Copyright © HiSilicon Technologies Co., Ltd

Chip	VI Port Mode Support	Range of the VI Device ID
Hi3531/Hi3532/ Hi3521/Hi3520A	No	-
Hi3520Dv200	Yes	[0, 1]
Hi3518A/Hi3516C	No	
Hi3518C/Hi3518E	No	-

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

- The default VI port mode is D1 or 960H.
- The VI port mode must be set to the 720p mode only when Hi3520D V200 runs the 4-channel 720p services.
- This MPI must be called after HI_MPI_VI_SetDevAttr is called and before HI_MPI_VI_EnableDev is called.

[Example]

None

[See Also]

None

HI_MPI_VI_GetPortMode

[Description]

Obtains the VI port mode.

[Syntax]

```
HI_S32 HI_MPI_VI_GetPortMode(VI_DEV ViDev, VI_PORTMODE_E *penPortMode);
```

[Parameter]

Parameter	Description	Input/Output
ViDev	ID of a VI device	Input
penRotate	Pointer to the VI port attribute	Output

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Difference]

Chip	VI Port Mode Support	Range of the VI Device ID
Hi3531/Hi3532/ Hi3521/Hi3520A	No	-
Hi3520Dv200	Yes	[0, 1]
Hi3518A/Hi3516C	No	
Hi3518C/Hi3518E	No	-

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_VI_SetPortMode

HI_MPI_VI_EnableChn422to420

[Description]

Enables the function of converting a captured YUV422 image into a YUV420 image for the VI channel.

[Syntax]

```
HI_S32 HI_MPI_VI_EnableChn422to420(VI_CHN ViChn);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	VI channel ID	Input

Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

Typical application scenario: When Hi3520D V200 runs 4-channel 720p services, two channels or six channels can capture only YUV422 images. In this case, this function can be used to convert a YUV422 image into a YUV420 image to reduce the required bandwidth.

[Example]

None

[See Also]

None

HI_MPI_VI_DisableChn422to420

[Description]

Disables the function of converting a captured YUV422 image into a YUV420 image for the VI channel.

[Syntax]

```
HI_S32 HI_MPI_VI_DisableChn422to420(VI_CHN ViChn);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	VI channel ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_VI_EnableChn422to420

HI_MPI_VI_SetChnScanMode

[Description]

Sets the scanning mode of a VI channel.

[Syntax]

```
HI_S32 HI_MPI_VI_SetChnScanMode(VI_CHN ViChn, VI_SCAN_MODE_E enScanMode);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	VI channel ID	Input
enScanMode	Scanning mode	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

When the one-channel SD video signal is multiplexed with the one-channel HD video signal of the frontend ADC, the scanning mode of the corresponding VI channel needs to be configured.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[See Also]

None

HI_MPI_VI_GetChnScanMode

[Description]

Obtains the scanning mode of a VI channel.

[Syntax]

```
HI_S32 HI_MPI_VI_GetChnScanMode(VI_CHN ViChn, VI_SCAN_MODE_E
*penScanMode);
```

[Parameter]

Parameter	Description	Input/Output
ViChn	VI channel ID	Input
penScanMode	Pointer to the scanning mode	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 3.6 "Error Codes."

[Requirement]

- Header files: hi_comm_vi.h, mpi_vi.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

None

3.5 Data Types

The VI data types are as follows:

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- VIU_MAX_DEV_NUM: Defines the maximum number of VI device ports.
- VIU_MAX_WAY_NUM_PER_DEV: Defines the maximum number of data ways for a VI device.
- VIU_MAX_PHYCHN_NUM: Defines the maximum number of physical VI channels.
- VIU_EXT_CHN_START: Defines the initial value of the extended VI channel ID.
- VIU_MAX_EXT_CHN_NUM: Defines the maximum number of extended VI channels.
- VIU_MAX_EXTCHN_BIND_PER_CHN: Defines the maximum number of extended channels bound to a VI channel.
- VIU_MAX_CHN_NUM: Defines the maximum number of VI channels.
- SUBCHN: Defines the ID of a secondary VI channel.
- VI_CAS_CHN_1: Defines the channel ID 32 for an extended VI cascade channel.
- VI_CAS_CHN_2: Defines the channel ID 33 for an extended VI cascade channel.
- VI_INTF_MODE_E: Defines the interface mode of a VI device.
- VI_INPUT_MODE_E: Defines the input mode of a VI device.
- VI_WORK_MODE_E: Defines the multiplexed working mode of a VI device.
- VI_SCAN_MODE_E: Defines whether interlaced or progressive input pictures are received by a VI device.
- VI_DATA_YUV_SEQ_E: Defines the arrangement sequence of the YUV data received by a VI device.
- VI_CLK_EDGE_E: Defines the type of the clock received by a VI device.
- VI_COMP_MODE_E: Defines the component type of the data received by a VI device.
- VI_COMBINE_MODE_E: Defines the mode of the data received by a VI device.
- VI_VSYNC_E: Defines the vertical sync signal type of the input data of a VI device.
- VI_VSYNC_NEG_E: Defines the vertical sync signal polarity of the input data of a VI device.
- VI_HSYNC_E: Defines the horizontal sync signal type of the input data of a VI device.
- VI_HSYNC_NEG_E: Defines the horizontal sync signal polarity of the input data of a VI device.
- VI_VSYNC_VALID_E: Defines the vertical valid sync signal type of the input data of a VI device.
- VI_VSYNC_VALID_NEG_E: Defines the vertical valid sync signal polarity of the input data of a VI device.
- VI_TIMING_BLANK_S: Defines the blanking information about the input timing of a VI device.
- VI_SYNC_CFG_S: Defines the sync information about the BT.601 timing or DC timing received by a VI device.
- VI_BT656_SYNC_CFG_S: Defines the sync information about the BT.656 timing received by a VI device.
- BT656_FIXCODE_E: Defines the most significant bit (MSB) of the BT.656 timing reference code.
- BT656_FIELD_POLAR_E: Defines the polarity of the field indication bit (F) of the BT.656 timing reference code.
- VI_DEV_ATTR_S: Defines the attributes of a VI device.
- VI_DEV_ATTR_EX_S: Defines the extended attributes of a VI device.
- VI_CHN_BIND_ATTR_S: Defines the binding attributes of a VI channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- VI_CAPSEL_E: Defines the frame/field selection of the captured VI pictures.
- VI_CHN_ATTR_S: Defines the attributes of a VI channel.
- VI_USERPIC_MODE_E: Defines the user picture mode.
- VI_USERPIC_BGC_S: Defines the information about a background picture only with a color.
- VI_USERPIC_ATTR_S: Defines the information about a user picture.
- VI_CHN_STAT_S: Defines the structure of VI channel information.
- ROTATE_E: Defines the rotation enumeration.
- VI_DATA_PATH_E: Defines the VI data channel enumeration.
- VI_DATA_TYPE_E: Defines the VI data type enumeration.
- VI_EXT_CHN_ATTR_S: Defines the attributes of an extended VI channel.
- LDC_VIEW_TYPE_E: Defines the LDC mode.
- LDC_ATTR_S: Defines the LDC attribute.
- VI_LDC_ATTR_S: Defines the VI LDC attribute.
- VI_CSC_TYPE_E: Defines the YUV CSC standard.
- VI_CSC_ATTR_S: Defines the CSC attribute.
- VI_FLASH_MODE_E: Defines the camera flash mode.
- VI_FLASH_CONFIG_S: Defines camera flash settings.
- VI_CHN_LUM_S: Defines the channel luminance statistics.
- VI_PORTMODE_E: Defines the VI port mode.

VIU_MAX_DEV_NUM

[Description]

Defines the maximum number of VI device ports.

[Syntax]

For Hi3531/Hi3532:

```
#define VIU_MAX_DEV_NUM 8
```

For Hi3521/Hi3520A:

```
#define VIU_MAX_DEV_NUM 4
```

For Hi3520D:

```
#define VIU_MAX_DEV_NUM       2
```

For Hi3518/Hi3516C/Hi3515A/Hi3515C:

```
#define VIU_MAX_DEV_NUM       1
```

[Note]

For details, see the descriptions of VI devices for each chip.

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VIU_MAX_WAY_NUM_PER_DEV

[Description]

Defines the maximum number of data ways for a VI device.

[Syntax]

For Hi3531/Hi3532/Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C:

```
#define VIU_MAX_WAY_NUM_PER_DEV 4
```

For Hi3518/Hi3516C:

```
#define VIU_MAX_WAY_NUM_PER_DEV        1
```

[Note]

None

[See Also]

None

VIU_MAX_PHYCHN_NUM

[Description]

Defines the maximum number of physical VI channels.

[Syntax]

For Hi3531/Hi3532:

```
#define VIU_MAX_PHYCHN_NUM 32
```

For Hi3521/Hi3520A:

```
#define VIU_MAX_PHYCHN_NUM 16
```

For Hi3520D:

```
#define VIU_MAX_PHYCHN_NUM 8
```

For Hi3515A/Hi3515C:

```
#define VIU_MAX_PHYCHN_NUM 4
```

For Hi3518/Hi3516C:

```
#define VIU_MAX_PHYCHN_NUM 1
```

[Note]

For details, see the descriptions physical channels for each chip.

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VIU_EXT_CHN_START

[Description]

Defines the initial value of the extended VI channel ID.

[Syntax]

```
#define VIU_EXT_CHN_START          VIU_MAX_PHYCHN_NUM
```

[Note]

None

[See Also]

None

VIU_MAX_EXT_CHN_NUM

[Description]

Defines the maximum number of extended VI channels.

[Syntax]

For Hi3531/Hi3532/Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C:

```
#define VIU_MAX_EXT_CHN_NUM         0
```

For Hi3518/Hi3516C

```
#define VIU_MAX_EXT_CHN_NUM         16
```

[Note]

None

[See Also]

None

VIU_MAX_EXTCHN_BIND_PER_CHN

[Description]

Defines the maximum number of extended channels bound to a VI channel.

[Syntax]

For Hi3531/Hi3532/Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C:

```
#define VIU_MAX_EXTCHN_BIND_PER_CHN         0
```

For Hi3518/Hi3516C

```
#define VIU_MAX_EXTCHN_BIND_PER_CHN         8
```

[Note]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

None

VIU_MAX_CHN_NUM

[Description]

Defines the maximum number of VI channels.

[Syntax]

For the Hi3531/Hi3532:

```
#define VIU_MAX_CHN_NUM 34
```

For the Hi3521/Hi3520A:

```
#define VIU_MAX_CHN_NUM 16
```

For Hi3520D:

```
#define VIU_MAX_CHN_NUM  8
```

For the Hi3515A/Hi3515C:

```
#define VIU_MAX_CHN_NUM 4
```

For the Hi3518/Hi3516C:

```
#define VIU_MAX_CHN_NUM (VIU_MAX_PHYCHN_NUM + VIU_MAX_EXT_CHN_NUM)
```

[Note]

For the Hi3531/Hi3532:

- Chn0 to chn15 are primary channels.
- Chn16 to chn31 are secondary channels corresponding to primary channels 0–15 respectively. Note that odd channels are invalid.
- Chn32 and chn33 are extended cascade channels. The two channels are virtual software channels.

For the Hi3518/Hi3516C:

- Chn0 is a physical channel.
- Chn1 to chn16 are extended channels.

[See Also]

None

SUBCHN

[Description]

Defines the ID of a secondary VI channel.

[Syntax]

```
#define  SUBCHN(ViChn)   (ViChn + 16)
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Difference]

Chip	Description
Hi3531/Hi3532	The macro is valid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	The macro is invalid.
Hi3518/Hi3516C	The macro is invalid.

[Note]

The value range of ViChn is {0, 15}.

[See Also]

None

VI_CAS_CHN_1

[Description]

Defines the channel ID 32 for an extended VI cascade channel.

[Syntax]

```
#define VI_CAS_CHN_1   32
```

[Difference]

Chip	Description
Hi3531/Hi3532	The macro is valid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	The macro is invalid.
Hi3518/Hi3516C	The macro is invalid.

[Note]

The channel ID 32 is valid only when you call HI_MPI_VI_EnableCascadeChn or HI_MPI_VI_DisableCascadeChn.

[See Also]

None

VI_CAS_CHN_2

[Description]

Defines the channel ID 33 for an extended VI cascade channel.

[Syntax]

```
#define VI_CAS_CHN_1   33
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Difference]

Chip	Description
Hi3531/Hi3532	The macro is valid.
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	The macro is invalid.
Hi3518/Hi3516C	The macro is invalid.

[Note]

The channel ID 33 is valid only when you call HI_MPI_VI_EnableCascadeChn or HI_MPI_VI_DisableCascadeChn.

[See Also]

None

VI_INTF_MODE_E

[Description]

Defines the interface mode of a VI device.

[Syntax]

```
typedef enum hiVI_INTF_MODE_E
{
  VI_MODE_BT656 = 0,
  VI_MODE_BT601,
  VI_MODE_DIGITAL_CAMERA,
  VI_MODE_BT1120_STANDARD,
  VI_MODE_BT1120_INTERLEAVED,
  VI_MODE_BUTT
} VI_INTF_MODE_E;
```

[Member]

Member	Description
VI_MODE_BT656	The data input protocol is the standard BT.656 protocol, the port data input mode is the Y/C composite mode, and the component mode is the single-component mode.
VI_MODE_BT601	The data input protocol is the standard BT.601 protocol, the port data input mode is the Y/C composite mode, and the component mode is the single-component mode.
VI_MODE_DIGITAL_CAMERA	The data input protocol is the DC protocol, the port data input mode is the Y/C composite mode, and the component mode is the single-

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	component mode.
VI_MODE_BT1120_STANDARD	The data input protocol is the standard BT.1120 protocol (BT.656+dual components), the port data input mode is the Y/C separation mode, and the component mode is the dual-component mode.
VI_MODE_BT1120_INTERLEAVED	The data input protocol is the BT.1120 interleave protocol, the port data input mode is the Y/C separation mode, and the component mode is the dual-component mode.

[Note]

The interface mode is VI_MODE_DIGITAL_CAMERA for the sensor.

[See Also]

- VI_DEV_ATTR_S
- HI_MPI_VI_SetDevAttr

VI_INPUT_MODE_E

[Description]

Defines the input mode of a VI device.

[Syntax]

```
typedef enum hiVI_INPUT_MODE_E
{
  VI_MODE_BT656 = 0,
  VI_MODE_BT601,
  VI_MODE_DIGITAL_CAMERA,
  VI_INPUT_MODE_INTERLEAVED,
  VI_MODE_BUTT
} VI_INPUT_MODE_E;
```

[Member]

Member	Description
VI_MODE_BT656	The data input protocol is the standard BT.656 protocol.
VI_MODE_BT601	The data input protocol is the standard BT.601 protocol.
VI_MODE_DIGITAL_CAMERA	The data input protocol is the DC protocol.
VI_INPUT_MODE_INTERLEAVED	The data input protocol is the BT.1120

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	interleave protocol.

[Note]

The input mode is VI_MODE_DIGITAL_CAMERA for the sensor. If the device input complies with the standard BT.1120 protocol, the input mode is set to VI_MODE_BT656 and VI_COMP_MODE_E is set to the dual-component mode. If the device input complies with the BT.1120 interleave protocol, the input mode is set to VI_INPUT_MODE_INTERLEAVED.

[See Also]

- VI_DEV_ATTR_EX_S
- HI_MPI_VI_SetDevAttrEx

VI_WORK_MODE_E

[Description]

Defines the multiplexed working mode of a VI device.

[Syntax]

```
typedef enum hiVI_WORK_MODE_E
{
   VI_WORK_MODE_1Multiplex = 0,
   VI_WORK_MODE_2Multiplex,
   VI_WORK_MODE_4Multiplex,
   VI_WORK_MODE_BUTT
} VI_WORK_MODE_E;
```

[Member]

Member	Description
VI_WORK_MODE_1Multiplex	1-channel multiplexed mode
VI_WORK_MODE_2Multiplex	2-channel multiplexed mode. The data input protocol must be the standard BT.656 protocol.
VI_WORK_MODE_4Multiplex	4-channel multiplexed mode. The data input protocol must be the standard BT.656 protocol.

[Note]

When the multiplexed mode is set to 2-channel multiplexed mode or 4-channel multiplexed mode, the device input protocol must be the BT.656 protocol. There is no such limitation in 1-channel multiplexed mode.

[See Also]

- VI_DEV_ATTR_S

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_VI_SetDevAttr

VI_SCAN_MODE_E

[Description]

Defines whether interlaced or progressive input pictures are received by a VI device.

[Syntax]

```
typedef enum hiVI_SCAN_MODE_E
{
  VI_SCAN_INTERLACED  = 0,/*Interlaced*/
  VI_SCAN_PROGRESSIVE,/*Progressive*/
  VI_SCAN_BUTT,
} VI_SCAN_MODE_E;
```

[Member]

Member	Description
VI_SCAN_INTERLACED	Interlaced picture
VI_SCAN_PROGRESSIVE	Progressive picture

[Note]

None

[See Also]

- VI_DEV_ATTR_S
- HI_MPI_VI_SetDevAttrEx

VI_DATA_YUV_SEQ_E

[Description]

Defines the arrangement sequence of the YUV data received by a VI device.

[Syntax]

```
typedef enum hiVI_DATA_YUV_SEQ_E
{
  VI_INPUT_DATA_VUVU = 0,
  VI_INPUT_DATA_UVUV,

  VI_INPUT_DATA_UYVY = 0,
  VI_INPUT_DATA_VYUY,
  VI_INPUT_DATA_YUYV,
  VI_INPUT_DATA_YVYU,
  VI_DATA_YUV_BUTT
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
} VI_DATA_YUV_SEQ_E;
```

[Member]

Member	Description
VI_INPUT_DATA_VUVU	When the YUV data is input in separation mode, the input sequence of the C component is VUVU.
VI_INPUT_DATA_UVUV	When the YUV data is input in separation mode, the input sequence of the C component is UVUV.
VI_INPUT_DATA_UYVY	When the YUV data is input in composite mode, the data input sequence is UYVY.
VI_INPUT_DATA_VYUY	When the YUV data is input in composite mode, the data input sequence is VYUY.
VI_INPUT_DATA_YUYV	When the YUV data is input in composite mode, the data input sequence is YUYV.
VI_INPUT_DATA_YVYU	When the YUV data is input in composite mode, the data input sequence is YVYU.

[Note]

None

[See Also]

- VI_DEV_ATTR_S
- HI_MPI_VI_SetDevAttr
- VI_DEV_ATTR_EX_S
- HI_MPI_VI_SetDevAttrEx

VI_CLK_EDGE_E

[Description]

Defines the type of the clock received by a VI device.

[Syntax]

```
typedef enum hiVI_CLK_EDGE_E
{
   VI_CLK_EDGE_SINGLE_UP = 0,
   VI_CLK_EDGE_SINGLE_DOWN,
   VI_CLK_EDGE_BUTT
} VI_CLK_EDGE_E;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
VI_CLK_EDGE_SINGLE_UP	Clock single-edge mode. The VI device samples clocks on the rising edge.
VI_CLK_EDGE_SINGLE_DOWN	Clock single-edge mode. The VI device samples clocks on the falling edge.

[Note]

None

[See Also]

- VI_DEV_ATTR_S
- HI_MPI_VI_SetDevAttrEx

VI_COMP_MODE_E

[Description]

Defines the component type of the data received by a VI device.

[Syntax]

```
typedef enum hiVI_COMP_MODE_E
{
   VI_COMP_MODE_SINGLE = 0,
   VI_COMP_MODE_DOUBLE = 1,
   VI_COMP_MODE_BUTT,
}VI_COMP_MODE_E;
```

[Member]

Member	Description
VI_COMP_MODE_SINGL	The component type of the input data is single-component.
VI_COMP_MODE_DOUBLE	The component type of the input data is dual-component.

[Note]

- This enumeration and the component mask au32CompMask[2] of VI_DEV_ATTR_S or VI_DEV_ATTR_EX_S must be configured at the same time.
- Dev0–Dev3 or Dev4–Dev7 (for the Hi3531/Hi3532) share 32-bit data lines. The data lines are used to connect to external video sources and only a part of data lines are used. The used data lines depend on the hardware connection. For example, Dev0 inputs 8-bit YUV data and the camera connects to the VI channel by using the first 8-bit data lines (data lines corresponding to bit0 to bit7). In this case, the component type of the input data is set to single-component and the component mask is set to 0xFF000000.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- If the video data is input in dual-component mode, the device attribute must be set to dual-component and the component mask of each component must be specified based on the pin connection. For example, if Dev0 is set to BT.1120 dual-component input mode (eight bits for the Y component and C component respectively), the component mask of the Y component is set to 0xFF000000 and the component mask of the C component is set to 0xFF0000.

[See Also]

- VI_DEV_ATTR_S
- VI_DEV_ATTR_EX_S
- HI_MPI_VI_SetDevAttrEx
- HI_MPI_VI_SetDevAttr

VI_COMBINE_MODE_E

[Description]

Defines the mode of the data received by a VI device.

[Syntax]

```
typedef enum hiVI_COMBINE_MODE_E
{
  VI_COMBINE_COMPOSITE = 0,
  VI_COMBINE_SEPARATE,
  VI_COMBINE_BUTT,
} VI_COMBINE_MODE_E;
```

[Member]

Member	Description
VI_COMBINE_COMPOSITE	Composite mode
VI_COMBINE_SEPARATE	Separation mode

[Note]

If the Y component and C component are input by using the same data line, the data mode is set to VI_COMBINE_COMPOSITE; if the Y component and C component are input by using different data lines, the data mode is set to VI_COMBINE_SEPARATE such as BT.1120.

[See Also]

- VI_DEV_ATTR_EX_S
- HI_MPI_VI_ChnBind

VI_VSYNC_E

[Description]

Defines the vertical sync signal type of the input data of a VI device.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
typedef enum hiVI_VSYNC_E
{
  VI_VSYNC_FIELD = 0,
  VI_VSYNC_PULSE,
} VI_VSYNC_E;
```

[Member]

Member	Description
VI_VSYNC_FIELD	Vertical sync invert mode. A field occurs when the level is inverted once. This member indicates the field number in BT.601 mode and line active signal in DC mode.
VI_VSYNC_PULSE	Vertical sync pulse mode. That is, when a pulse arrives, a new frame or field starts.

[Note]

None

[See Also]

- VI_VSYNC_NEG_E
- VI_SYNC_CFG_S

VI_VSYNC_NEG_E

[Description]

Defines the vertical sync signal polarity of the input data of a VI device.

[Syntax]

```
typedef enum hiVI_VSYNC_NEG_E
{
  VI_VSYNC_NEG_HIGH = 0,
  VI_VSYNC_NEG_LOW
} VI_VSYNC_NEG_E;
```

[Member]

Member	Description
VI_VSYNC_NEG_HIGH	If VI_VSYNC_E is set to VI_VSYNC_FIELD, the level of vsync signal of the even field is high. If VI_VSYNC_E is set to VI_VSYNC_PULSE, the positive pulse indicates vsync pulse.
VI_VSYNC_NEG_LOW	If VI_VSYNC_E is set to VI_VSYNC_FIELD, the level of vsync signal of the even field is low.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	If VI_VSYNC_E is set to VI_VSYNC_PULSE, the negative pulse indicates vsync pulse.

[Note]

None

[See Also]

- VI_VSYNC_E
- VI_SYNC_CFG_S

VI_HSYNC_E

[Description]

Defines the horizontal sync signal type of the input data of a VI device.

[Syntax]

```
typedef enum hiVI_HSYNC_E
{
  VI_HSYNC_VALID_SIGNAL = 0,
  VI_HSYNC_PULSE,
} VI_HSYNC_E;
```

[Member]

Member	Description
VI_HSYNC_VALID_SIGNAL	Horizontal sync data valid signal
VI_HSYNC_PULSE	Horizontal sync pulse signal

[Note]

None

[See Also]

- VI_HSYNC_NEG_E
- VI_SYNC_CFG_S

VI_HSYNC_NEG_E

[Description]

Defines the horizontal sync signal polarity of the input data of a VI device.

[Syntax]

```
typedef enum hiVI_HSYNC_NEG_E
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
{
  VI_HSYNC_NEG_HIGH = 0,
  VI_HSYNC_NEG_LOW
} VI_HSYNC_NEG_E;
```

[Member]

Member	Description
VI_HSYNC_NEG_HIGH	If VI_HSYNC_E is set to VI_HSYNC_VALID_SIGNAL, the high level indicates valid data. If VI_HSYNC_E is set to VI_HSYNC_PULSE, the positive pulse indicates sync pulse.
VI_HSYNC_NEG_LOW	If VI_HSYNC_E is set to VI_HSYNC_VALID_SIGNAL, the low level indicates valid data. If VI_HSYNC_E is set to VI_HSYNC_PULSE, the negative pulse indicates sync pulse.

[Note]

None

[See Also]

- VI_HSYNC_E
- VI_SYNC_CFG_S

VI_VSYNC_VALID_E

[Description]

Defines the vertical valid sync signal type of the input data of a VI device.

[Syntax]

```
typedef enum hiVI_VSYNC_VALID_E
{
  VI_VSYNC_NORM_PULSE = 0,
  VI_VSYNC_VALID_SINGAL,
} VI_VSYNC_VALID_E;
```

[Member]

Member	Description
VI_VSYNC_NORM_PULSE	Vertical valid sync flag. It is used in DC mode.
VI_VSYNC_VALID_SINGAL	Line active signal of the vertical sync timing. It is used in DC mode.

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

This data type must be configured in DC mode.

[See Also]

- VI_VSYNC_E
- VI_VSYNC_VALID_NEG_E
- VI_SYNC_CFG_S

VI_VSYNC_VALID_NEG_E

[Description]

Defines the vertical valid sync signal polarity of the input data of a VI device.

[Syntax]

```
typedef enum hiVI_VSYNC_VALID_NEG_E
{
  VI_VSYNC_VALID_NEG_HIGH = 0,
  VI_VSYNC_VALID_NEG_LOW
} VI_VSYNC_VALID_NEG_E;
```

[Member]

Member	Description
VI_VSYNC_VALID_NEG_HIGH	If VI_VSYNC_VALID_E is set to VI_VSYNC_NORM_SINGAL, the high level indicates valid signal.
VI_VSYNC_VALID_NEG_LOW	If VI_VSYNC_VALID_E is set to VI_VSYNC_NORM_SINGAL, the low level indicates valid signal.

[Note]

This data type is valid only when VI_VSYNC_VALID_E is set to VI_VSYNC_NORM_SINGAL.

[See Also]

- VI_VSYNC_VALID_E
- VI_SYNC_CFG_S

VI_TIMING_BLANK_S

[Description]

Defines the blanking information about the input timing of a VI device.

[Syntax]

```
typedef struct hiVI_TIMING_BLANK_S
{
  HI_U32 u32HsyncHfb ;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
  HI_U32 u32HsyncAct ;
  HI_U32 u32HsyncHbb ;
  HI_U32 u32VsyncVfb ;
  HI_U32 u32VsyncVact ;
  HI_U32 u32VsyncVbb ;
  HI_U32 u32VsyncVbfb ;
  HI_U32 u32VsyncVbact ;
  HI_U32 u32VsyncVbbb ;
}VI_TIMING_BLANK_S;
```

[Member]

Member	Description
u32HsyncHfb	Horizontal front blanking width
u32HsyncAct	Horizontal active width
u32HsyncHbb	Horizontal back blanking width
u32VsyncVfb	Vertical front blanking height of the frame picture or the odd vertical front blanking height during frame input
u32VsyncVact	Vertical active height of the frame picture or the odd vertical active width during frame input
u32VsyncVbb	Vertical back blanking height of the frame picture or the odd vertical back blanking height during frame input
u32VsyncVbfb	Even vertical front blanking height during interlaced input (invalid during frame input)
u32VsyncVbact	Even vertical active height during interlaced input (invalid during frame input
u32VsyncVbbb	Even vertical back blanking height during interlaced input (invalid during frame input)

[Note]

None

[See Also]

VI_SYNC_CFG_S

VI_SYNC_CFG_S

[Description]

Defines the sync information about the BT.601 timing or DC timing received by a VI device.

[Syntax]

```
typedef struct hiVI_SYNC_CFG_S
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
{
  VI_VSYNC_E  enVsync;
  VI_VSYNC_NEG_E enVsyncNeg;
  VI_HSYNC_E  enHsync;
  VI_HSYNC_NEG_E enHsyncNeg;
  VI_VSYNC_VALID_E   enVsyncValid;
  VI_VSYNC_VALID_NEG_E  enVsyncValidNeg;
  VI_TIMING_BLANK_S  stTimingBlank;
} VI_SYNC_CFG_S;
```

[Member]

Member	Description
enVsync	Type of the vertical sync signal
enVsyncNeg	Polarity of the vertical sync signal
enHsync	Type of the horizontal sync signal
enHsyncNeg	Polarity of the horizontal sync signal
enVsyncValid	Type of the vertical valid sync signal
enVsyncValidNeg	Polarity of the vertical valid sync signal
stTimingBlank	Blanking information about the input timing

[Note]

- In BT.601 mode, only the vertical sync signals enVsync and enVsyncNeg need to be configured. enVsync indicates the type of the vertical sync signal of the VI_P_VSYNC_FIELD pin, and enVsyncNeg indicates the polarity of the vertical sync signal of the VI_P_VSYNC_FIELD pin. The two signals determine the vertical sync timing to be used.
  - enVsync is set to VI_VSYNC_FIELD, which indicates the field number in BT.601 mode. In this case, if enVsyncNeg is set to VI_VSYNC_NEG_HIGH, the high level indicates even field and the low level indicates odd field; if enVsyncNeg is set to VI_VSYNC_NEG_LOW, the low level indicates even field and the high level indicates odd field
  - enVsync is set to VI_VSYNC_PULSE, which indicates the vertical sync pulse. That is, when a new pulse arrives, a new field or frame starts. In this case, if enVsyncNeg is set to VI_VSYNC_NEG_HIGH, the positive pulse indicates vertical sync pulse; if enVsyncNeg is set to VI_VSYNC_NEG_LOW, the negative pulse indicates vertical sync pulse.
- In DC mode, the vertical sync signals enVsync, enVsyncValid, and enVsyncValidNeg need to be configured. enVsync indicates the type of the vertical sync signal of the VI_P_VSYNC_FIELD pin, enVsyncValid indicates the vertical sync signal flag of the VI_P_VSYNC_FIELD pin, and enVsyncValidNeg indicates the polarity of the vertical valid signal of the VI_P_VSYNC_FIELD pin. enVsyncValidNeg is valid only when enVsyncValid is set to VI_VSYNC_VALID_SINGAL.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

  - enVsync is set to VI_VSYNC_PULSE, which indicates the vertical timing pulse mode. If the hsync signal is valid in the vertical blanking area, enVsyncValid must be set to VI_VSYNC_VALID_SINGAL to select the vertical valid flag and enVsyncValidNeg must be set to VI_VSYNC_VALID_NEG_HIGH (active high) or VI_VSYNC_VALID_NEG_LOW (active low). In other cases, enVsyncValidNeg can be ignored.
  - If the input timing is in vertical timing line active mode, enVsync must be set to VI_VSYNC_FIELD.
- enHsync indicates the type of the horizontal sync signal of the VI_P_HSYNC_VD pin, and enHsyncNeg indicates the polarity of the horizontal sync signal of the VI_P_HSYNC_VD pin. enHsync and enHsyncNeg determine the horizontal sync timing to be used.
  - enHsync is set to VI_HSYNC_VALID_SIGNAL, which indicates the data valid signal. In this case, if enHsyncNeg is set to VI_HSYNC_NEG_HIGH, the high level indicates data valid; if enHsyncNeg is set to VI_HSYNC_NEG_LOW, the low level indicates data valid.
  - enHsync is set to VI_HSYNC_PULSE, which indicates the horizontal sync pulse. That is, when a new pulse arrives, a new line starts. In this case, if enHsyncNeg is set to VI_HSYNC_NEG_HIGH, the positive pulse indicates horizontal sync pulse; if enHsyncNeg is set to VI_HSYNC_NEG_LOW, the negative pulse indicates horizontal sync pulse.

Note that the blanking area does not need to be configured for the horizontal or vertical valid data.

For details about the timings supported by the Hi3531, see section 11.1.3.2 "Function Principle" in the *Hi3531 H.264 CODEC Processor Data Sheet*.

[See Also]

- VI_DEV_ATTR_S
- HI_MPI_VI_SetDevAttr
- VI_DEV_ATTR_EX_S
- HI_MPI_VI_SetDevAttrEx

BT656_FIXCODE_E

[Description]

Defines the most significant bit (MSB) of the BT.656 timing reference code.

[Syntax]

```
typedef enum hiBT656_FIXCODE_E
{
  BT656_FIXCODE_1 = 0,
  BT656_FIXCODE_0
}BT656_FIXCODE_E;
```

[Member]

Member	Description
BT656_FIXCODE_1	The MSB of the start of active video (SAV) or end of active

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	video (EAV) of the BT.656 protocol is fixed at 1.
BT656_FIXCODE_0	The MSB of the SAV or EAV of the BT.656 protocol is fixed at 0.

[Note]

None

[See Also]

- VI_DEV_ATTR_EX_S
- HI_MPI_VI_SetDevAttrEx

BT656_FIELD_POLAR_E

[Description]

Defines the polarity of the field indication bit (F) of the BT.656 timing reference code.

[Syntax]

```
typedef enum hiBT656_FIELD_POLAR_E
{
  BT656_FIELD_POLAR_STD = 0,
  BT656_FIELD_POLAR_NSTD
}BT656_FIELD_POLAR_E;
```

[Member]

Member	Description
BT656_FIELD_POLAR_STD	In standard mode, F is 0 in the first field and 1 in the second field.
BT656_FIELD_POLAR_NSTD	In non-standard mode, F is 1 in the first field and 0 in the second field.

[Note]

None

[See Also]

- VI_DEV_ATTR_EX_S
- HI_MPI_VI_SetDevAttrEx

VI_BT656_SYNC_CFG_S

[Description]

Defines the sync information about the BT.656 timing received by a VI device.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
typedef struct hiVI_BT656_SYNC_CFG_S
{
   BT656_FIXCODE_E   enFixCode;
   BT656_FIELD_POLAR_E  enFieldPolar;
}VI_BT656_SYNC_CFG_S;
```

[Member]

Member	Description
enFixCode	MSB of the BT.656 timing reference code
enFieldPolar	Polarity of the field indication bit (F) of the BT.656 timing reference code

[Note]

For details about the definitions related to BT.656, see related protocols.

[See Also]

- VI_DEV_ATTR_EX_S
- HI_MPI_VI_SetDevAttrEx

VI_DEV_ATTR_S

[Description]

Defines the attributes of a VI device.

[Syntax]

```
typedef struct hiVI_DEV_ATTR_S
{
    VI_INTF_MODE_E        enIntfMode;
    VI_WORK_MODE_E        enWorkMode;
    HI_U32                au32CompMask[2];
    VI_SCAN_MODE_E        enScanMode;
    HI_S32                s32AdChnId[4];
    /*The preceding parameters must be set in BT.601 mode or DC mode and
are invalid in other modes.*/
    VI_DATA_YUV_SEQ_E     enDataSeq;
    VI_SYNC_CFG_S         stSynCfg;
    VI_DATA_PATH_E        enDataPath;
    VI_DATA_TYPE_E        enInputDataType;
    HI_BOOL               bDataRev;
} VI_DEV_ATTR_S;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
enIntfMode	Interface mode
enWorkMode	1-, 2-, or 4-channel multiplexed mode
au32CompMask[2]	Component mask When enIntfMode is set to VI_MODE_BT1120_STANDARD, the component masks of the Y component and C component must be configured. In other cases, only single-component mask needs to be configured.
enScanMode	Input scanning mode (progressive or interlaced)
s32AdChnId[4]	Its value range is [–1, +3]. Typically, the default value –1 is recommended.
enDataSeq	Input data sequence (only the YUV format is supported). This member must be configured in BT.601 mode or DC mode and is invalid in other modes.
stSynCfg	Sync timing. This member must be configured in BT.601 mode or DC mode and is invalid in other modes.
enDataPath	Input data path. This member is valid only for the Hi3518 or Hi3516C. If data is from a sensor without ISP, select the ISP channel. If data is from a sensor with an ISP or from the ADC, select the bypass channel (in this case, the build-in ISP will be disabled). The value Raw Data is used only for debugging. If enDataPath is set to a raw data channel, the raw data from the sensor can be captured.
enInputDataType	Input data type. This member is valid only for the Hi3518 or Hi3516C. Typically, the input data from the sensor is RGB data, and the input data from the ADC is YUV data.
bDataRev	This member is valid only for the Hi3520D, Hi3515A, Hi3515C, Hi3518, or Hi3516C. The data lines of the AD/sensor may reversely connect to those of the VI due to hardware routing restrictions. For example, the VIU_DATA0 pin on the VIU may connect to the DATA7 pin on the AD, and the VIU_DATA7 pin on the VIU may connect to the DATA0 pin on the AD. If the pins on the AD or sensor connect to the same pins on the VIU, **bDataRev** is **HI_FALSE**. Otherwise, **bDataRev** is **HI_TRUE**.

[Note]

- When s32AdChnId[i] is set to –1, the MPP detects the AdId numbered i by default. An AdId cannot be repeatedly detected. That is, s32AdChnId cannot be set to –1, 0, 2, or 3 because the values of s32AdChnId[0] and s32AdChnId[1] conflict when the value –1 of s32AdChnId[0] is converted into 0.
- When enWorkMode is set to 1-channel multiplexing mode, s32AdChnId[0] must be set to –1 or 0; when WorkMode is set to 2-channel multiplexing mode, s32AdChnId[0] or s32AdChnId[1] must be set to –1, 0, or +1.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- You are advised to set s32AdChnId to –1. That is, the AdIds 0, 1, 2, and 3 are detected in sequence in default binding configurations.

[See Also]

HI_MPI_VI_SetDevAttr

VI_DEV_ATTR_EX_S

[Description]

Defines the extended attributes of a VI device.

[Syntax]

```
typedef struct hiVI_DEV_ATTR_EX_S
{
    VI_INPUT_MODE_E       enInputMode;
    VI_WORK_MODE_E        enWorkMode;
    VI_COMBINE_MODE_E     enCombineMode;
    VI_COMP_MODE_E        enCompMode;
    HI_U32                au32CompMask[2];
    VI_CLK_EDGE_E         enClkEdge;
    VI_SCAN_MODE_E        enScanMode;
    VI_DATA_YUV_SEQ_E     enDataSeq;
    HI_S32                s32AdChnId[4];
    VI_SYNC_CFG_S         stSynCfg;
    VI_BT656_SYNC_CFG_S   stBT656SynCfg;
    VI_DATA_PATH_E        enDataPath;
VI_DATA_TYPE_E         enInputDataType;
HI_BOOL                  bDataRev;
} VI_DEV_ATTR_EX_S;
```

[Member]

Member	Description
enInputMode	Interface mode
enWorkMode	1-, 2-, or 4-channel multiplexed mode
enCombineMode	Y/C composite or separation mode
enCompMode	Component mode (single-component or dual-component), corresponding to enCombineMode
au32CompMask[2]	Component mask When enInputMode is set to VI_INPUT_MODE_BT656, enCompMode is set to VI_COMP_MODE_DOUBLE, and enCombineMode is set to VI_COMBINE_SEPARATE, VI_MODE_BT1120_STANDARD is supported. In this case, the component masks of the Y component and C component must be configured. In other cases, only single-component mask needs to be

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	configured.
enClkEdge	Clock edge mode (sampling on the rising or falling edge)
enScanMode	Input scanning mode (progressive or interlaced)
enDataSeq	Input data sequence (only the YUV format is supported)
s32AdChnId[4]	For details, see the description of VI_DEV_ATTR_S.
stSynCfg	Sync timing. This member must be configured in BT.601 mode or DC mode and is invalid in other modes.
stBT656SynCfg	Sync timing. This member must be configured in BT.656 mode.
enDataPath	Input data path. This member is valid only for the Hi3518 or Hi3516C. If data is from a sensor without ISP, select the ISP channel. If data is from a sensor with an ISP or from the ADC, select the bypass channel (in this case, the build-in ISP will be disabled). The value Raw Data is used only for debugging. If enDataPath is set to a raw data channel, the raw data from the sensor can be captured.
enInputDataType	Input data type. This member is valid only for the Hi3518 or Hi3516C. Typically, the input data from the sensor is RGB data, and the input data from the ADC is YUV data.
bDataRev	This member is valid only for the Hi3520D, Hi3515A, Hi3515C, Hi3518, or Hi3516C. The data lines of the AD/sensor may reversely connect to those of the VI due to hardware routing restrictions. For example, the VIU_DATA0 pin on the VIU may connect to the DATA7 pin on the AD, and the VIU_DATA7 pin on the VIU may connect to the DATA0 pin on the AD. If the pins on the AD or sensor connect to the same pins on the VIU, **bDataRev** is **HI_FALSE**. Otherwise, **bDataRev** is **HI_TRUE**.

[Note]

None

[See Also]

HI_MPI_VI_SetDevAttrEx

VI_CHN_BIND_ATTR_S

[Description]

Defines the binding attributes of a VI channel.

[Syntax]

```
typedef struct hiVI_CHN_BIND_ATTR_S
{
  VI_DEV ViDev;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
  VI_WAY ViWay;
} VI_CHN_BIND_ATTR_S;
```

[Member]

Member	Description
ViDev	Device port bound to a VI channel
ViWay	Way in which the device port is bound to a VI channel

[Note]

None

[See Also]

HI_MPI_VI_ChnBind

VI_CAPSEL_E

[Description]

Defines the frame/field selection of the captured VI pictures.

[Syntax]

```
typedef enum hiVI_CAPSEL_E
{
  VI_CAPSEL_TOP=0,
  VI_CAPSEL_BOTTOM,
  VI_CAPSEL_BOTH,
  VI_CAPSEL_BUTT
} VI_CAPSEL_E;
```

[Member]

Member	Description
VI_CAPSEL_TOP	The top field is selected.
VI_CAPSEL_BOTTOM	The bottom field is selected (recommended when a single field is captured).
VI_CAPSEL_BOTH	Both the top field and bottom field are selected.

[Note]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VI_CHN_ATTR_S

[Description]

Defines the attributes of a VI channel.

[Syntax]

```
typedef struct hiVI_CHN_ATTR_S
{
   RECT_S  stCapRect;
   SIZE_S  stDestSize;
   VI_CAPSEL_E enCapSel;
   PIXEL_FORMAT_E  enPixFormat;
   HI_BOOL bMirror;
   HI_BOOL bFlip;
   HI_BOOL bChromaResample;
   HI_S32  s32SrcFrameRate;
   HI_S32  s32FrameRate;
} VI_CHN_ATTR_S;
```

[Member]

Member	Description
stCapRect	Start coordinates of the capture region relative to the device picture size, and width and height of the capture region
stDestSize	Target picture size
enCapSel	Frame/field select It is used only in interlaced mode. You are advised to capture the bottom field in single-field mode. In progressive mode, this parameter must be set to VI_CAPSEL_BOTH.
enPixFormat	Pixel storage format The formats semi-planar420 and semi-planar422 are supported.
bMirror	Whether to mirror
bFlip	Whether to flip
bChromaResample	Whether to perform chrominance resampling
s32SrcFrameRate	Source frame rate It cannot be less than –1. You are advised to set the source frame rate to the frame rate of the connected camera.
s32FrameRate	Target frame rate It must be lower than or equal to the source frame rate, and it cannot be less than –1. The source frame rate and target frame rate must be both –1 or neither is –1. If both the source frame rate and target frame rate are –1, the frame rate is not controlled.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Difference]

Member	Hi3531/Hi3532	Hi3521/Hi3520A/ Hi3520D/Hi3515A/ Hi3515A	Hi3518/Hi3516C
stCapRect	This member is valid only for primary channels, and the width and height defined in this member are static attributes.	This member is valid only for primary attributes, and the width and height defined in this member are static attributes. For secondary attributes, the attributes of this member are static.	This member is valid only for physical channels, and the width and height defined in this member are static attributes.
stDestSize	As primary channels do not support scaling, the width and height defined in this member must be the same as the width and height defined in stCapRect. The width and height defined in this member are static attributes. For secondary channels, the attributes of this member are dynamic.	For both primary and secondary attributes, the attributes of this member are dynamic.	As VI channels do not support scaling, the width and height defined in this member must be the same as the width and height defined in stCapRect in dual-field capture mode. The height defined in this member must be half of the height defined in stCapRect in single-field capture mode. The height defined in this member is a static attribute.
enCapSel	For primary channels, the attributes of this member are static. For secondary channels, the attributes of this member are dynamic.	For both primary and secondary attributes, the attributes of this member are dynamic.	For both primary and secondary attributes, the attributes of this member are dynamic.
bChromaResample	This member is valid only for primary channels.	This member is invalid.	This member is invalid.
s32SrcFrameRate and s32FrameRate	The members are valid for both primary and secondary channels.	The members are valid only for primary attributes.	The members are valid only for primary attributes.

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- The attributes are shared by primary and secondary channels. You must set the attributes of primary channels before setting the attributes of secondary channels.
- The attributes are shared by primary and secondary attributes. You must set primary attributes before setting secondary attributes.
- Static attributes can be modified only after primary and secondary channels are disabled.
- Interlaced pictures do not support mirror and flip. If the function is enabled, the sequence of capturing top and bottom fields is reversed, which may affect the picture quality when pictures are processed by other modules. As the MPP does not check whether the function is enabled, you need to disable the function in interlaced mode.

[See Also]

- HI_MPI_VI_SetChnAttr
- HI_MPI_VI_GetChnAttr

VI_USERPIC_MODE_E

[Description]

Defines the user picture mode.

[Syntax]

```
typedef enum hi_VI_USERPIC_MODE_E
{
   VI_USERPIC_MODE_PIC = 0,
   VI_USERPIC_MODE_BGC,
   VI_USERPIC_MODE_BUTT,
} VI_USERPIC_MODE_E;
```

[Member]

Member	Description
VI_USERPIC_MODE_PIC	YUV picture
VI_USERPIC_MODE_BGC	Background picture with only one color

[Note]

None

[See Also]

None

VI_USERPIC_BGC_S

[Description]

Defines the information about a background picture only with a color.

[Syntax]

```
typedef struct hiVI_USERPIC_BGC_S
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
{
    HI_U32  u32BgColor;
} VI_USERPIC_BGC_S;
```

[Member]

Member	Description
u32BgColor	Filled data

[Note]

None

[See Also]

None

VI_USERPIC_ATTR_S

[Description]

Defines the information about a user picture.

[Syntax]

```
typedef struct hiVI_USERPIC_ATTR_S
{
   HI_BOOL bPub;
   VI_USERPIC_MODE_E   enUsrPicMode;
   union
   {
   VIDEO_FRAME_INFO_S  stUsrPicFrm;
   VI_USERPIC_BGC_SstUsrPicBg;
    }unUsrPic;
} VI_USERPIC_ATTR_S;
```

[Member]

Member	Description
bPub	Whether the user picture information is shared by all VI devices and channels
enUsrPicMode	User picture mode
stUsrPicFrm	Information about a YUV picture
stUsrPicBg	Information about a background picture only with a color

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Currently, the user picture information is shared by all devices and channels. That is, bPub must be set to HI_TRUE. Otherwise, the kernel returns an error.

[See Also]

None

VI_CHN_STAT_S

[Description]

Defines the structure of VI channel information.

[Syntax]

```
typedef struct hiVI_CHN_STAT_S
{
  HI_BOOL bEnable;   /* Whether this channel is enabled */
  HI_U32 u32IntCnt;  /* The video frame interrupt count */
  HI_U32 u32FrmRate; /* current frame rate */
  HI_U32 u32LostInt; /* The interrupt is received but nobody care */
  HI_U32 u32VbFail;  /* Video buffer malloc failure */
  HI_U32 u32PicWidth;/* current picture width */
  HI_U32 u32PicHeight;/* current picture height */
  HI_U32 u32AutoDisInt; * auto disable interrupt count, when VIU detected
  too many interrupts */
} VI_CHN_STAT_S;
```

[Member]

Member	Description
bEnable	Channel enable
u32IntCnt	Interrupt count
u32FrmRate	Average frame rate that is calculated every 10 seconds. This value may be inaccurate.
u32LostInt	Lost interrupt count
u32VbFail	Number of times that VBs fail to be obtained
u32PicWidth	Picture width
u32PicHeight	Picture height
u32AutoDisInt	Number of times of automatically disabling interruption when too many interruption exceptions occur This parameter is related to stop_int_level. For example, if stop_int_level is set to 500 μs and the interval between two neighboring interruption exceptions is smaller than 500 μs, too many interruption exceptions occur. In this case, the SDK automatically disables interruption for 10s. Ten seconds later, the SDK enables interruption.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

- The interrupt count is used to check whether interrupts are generated.
- The frame rate is the average frame rate that is calculated every 10 seconds. Therefore, the average frame rate is inaccurate.
- If the queried lost interrupt count continuously increases, an exception occurs in the VI channel.

[See Also]

None

ROTATE_E

[Description]

Defines the rotation enumeration.

[Syntax]

```
typedef enum hiROTATE_E
{
  ROTATE_NONE = 0,
  ROTATE_90  = 1,
  ROTATE_180= 2,
  ROTATE_270= 3,
  ROTATE_BUTT
} ROTATE_E;;
```

[Member]

Member	Description
ROTATE_NONE	No rotation
ROTATE_90	Clockwise rotation by 90º
ROTATE_180	Clockwise rotation by 180º
ROTATE_270	Clockwise rotation by 270º

[Note]

None

[See Also]

- HI_MPI_VI_SetRotate
- HI_MPI_VI_GetRotate

VI_DATA_PATH_E

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Defines the VI data channel enumeration.

[Syntax]

```
typedef enum hiVI_DATA_PATH_E
{
  VI_PATH_BYPASS= 0,   /* ISP bypass */
  VI_PATH_ISP   = 1,/* ISP enable */
  VI_PATH_RAW   = 2,/* Capture raw data, for debug */
  VI_PATH_BUTT
}VI_DATA_PATH_E;
```

[Member]

Member	Description
VI_PATH_BYPASS	VIU internal ISP bypass
VI_PATH_ISP	VIU internal ISP enable
VI_PATH_RAW	Disable for internal ISP, CSC, and LDC modules of the VIU. This member is used when the raw data from the sensor is captured.

[Note]

- When the input YUV data is from the sensor with an ISP, set this data type to VI_PATH_BYPASS.
- When the input data is from the sensor without ISP, set this data type to VI_PATH_ISP.
- VI_PATH_RAW is used to capture the raw data from the sensor, and is used only for debugging.

[See Also]

- VI_DEV_ATTR_S
- VI_DEV_ATTR_EX_S

VI_DATA_TYPE_E

[Description]

Defines the VI data type enumeration.

[Syntax]

```
typedef enum hiVI_DATA_TYPE_E
{
  VI_DATA_TYPE_YUV = 0,
  VI_DATA_TYPE_RGB = 1,
  VI_DATA_TYPE_BUTT
} VI_DATA_TYPE_E;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
VI_DATA_TYPE_YUV	The input data format is YUV and the ADC is connected at the VI front-end.
VI_DATA_TYPE_RGB	The input data format is RGB and the sensor is connected at the VI front-end.

[Note]

None

[See Also]

- VI_DEV_ATTR_S
- VI_DEV_ATTR_EX_S

VI_EXT_CHN_ATTR_S

[Description]

Defines the attributes of an extended VI channel.

[Syntax]

```
typedef struct hiVI_EXT_CHN_ATTR_S
{
   VI_CHN   s32BindChn;
   SIZE_S   stDestSize;
   HI_S32  s32SrcFrameRate;
   HI_S32    s32FrameRate;
   PIXEL_FORMAT_E  enPixFormat;
}VI_EXT_CHN_ATTR_S;
```

[Member]

Member	Description
s32BindChn	Channel to which an extended channel is bound
stDestSize	Target picture size
s32SrcFrameRate	Source frame rate. It cannot be less than −1. You are advised to set it the same as the frame rate of the interconnected camera.
s32FrameRate	Target frame rate. It must be lower than or equal to the source frame rate, and it cannot be less than –1. The source frame rate and target frame rate must be both –1 or neither is –1. If both the source frame rate and target frame rate are –1, the frame rate is not controlled.
enPixFormat	Pixel storage format. The formats semi-planar420 and semi-planar422 are supported.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

The target picture size is the width and height of the output picture of the extended channel. The width or height is less than 4096 pixels. The minification for the extended channel is less than 13.

[See Also]

- HI_MPI_VI_SetChnAttr
- HI_MPI_VI_GetChnAttr

LDC_VIEW_TYPE_E

[Description]

Defines the LDC mode.

[Syntax]

```
typedef enum hiLDC_VIEW_TYPE_E
{
    LDC_VIEW_TYPE_ALL = 0,
    LDC_VIEW_TYPE_CROP,
    LDC_VIEW_TYPE_BUTT,
} LDC_VIEW_TYPE_E;
```

[Member]

Member	Description
LDC_VIEW_TYPE_ALL	Full mode. The maximum rectangle is obtained based on the edges of the corrected picture, and then the rectangle is zoomed out to the source picture size. Picture edges may be retained.
LDC_VIEW_TYPE_CROP	Crop mode. The corrected picture is cropped based on the source picture size. Picture edges may be lost.

[Note]

None

[See Also]

- LDC_ATTR_S
- VI_LDC_ATTR_S

LDC_ATTR_S

[Description]

Defines the LDC attribute.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
typedef struct hiLDC_ATTR_S
{
    LDC_VIEW_TYPE_E enViewType;
    HI_S32 s32CenterXOffset;
    HI_S32 s32CenterYOffset;
    HI_S32 s32Ratio;
} LDC_ATTR_S;
```

[Member]

Member	Description
enViewType	LDC mode, including crop mode and full mode
s32CenterXOffset	Horizontal offset of the distortion center relative to the picture center. The value range is [−28, +28].
s32CenterYOffset	Vertical offset of the distortion center relative to the picture center. The value range is [−14, +14].
s32Ratio	Distortion ratio. The value range is [0, 511].

[Note]

When the resolution of the captured VI picture is greater than 720p, the range of **s32Ratio** is [0, 192].

[See Also]

LDC_VIEW_TYPE_E

VI_LDC_ATTR_S

[Description]

Defines the VI LDC attribute.

[Syntax]

```
typedef struct hiVI_LDC_ATTR_S
{
    HI_BOOL bEnable;
    LDC_ATTR_S stAttr;
}VI_LDC_ATTR_S;
```

[Member]

Member	Description
bEnable	LDC enable
stAttr	LDC attribute

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

None

[See Also]

- LDC_VIEW_TYPE_E
- LDC_ATTR_S
- HI_MPI_VI_SetLDCAttr
- HI_MPI_VI_GetLDCAttr

VI_CSC_TYPE_E

[Description]

Defines the YUV CSC standard.

[Syntax]

```
typedef enum hiVI_CSC_TYPE_E
{
    VI_CSC_TYPE_601 = 0,
    VI_CSC_TYPE_709,
    VI_CSC_TYPE_BUTT,
} VI_CSC_TYPE_E;
```

[Member]

Member	Description
VI_CSC_TYPE_601	BT.601 CSC standard
VI_CSC_TYPE_709	BT.709 CSC standard

[Note]

The YUV CSC standards include BT.601 and BT.709 standards. This enumeration defines the CSC standard that is used when RGB pictures are converted into YUV pictures. Typically, the BT.601 standard is used for the SD device, and the BT.709 standard is used for the HD device.

[See Also]

None

VI_CSC_ATTR_S

[Description]

Defines the CSC attribute.

[Syntax]

```
typedef struct hiVI_CSC_ATTR_S
{
    VI_CSC_TYPE_E enViCscType;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_U32 u32LumaVal;                  /*Luminance:   [0-100] */
    HI_U32 u32ContrVal                  /*Contrast :   [0-100] */;
    HI_U32 u32HueVal;                   /*Hue      :   [0-100] */
    HI_U32 u32SatuVal;                  /*Saturation:  [0-100] */
} VI_CSC_ATTR_S;
```

[Member]

Member	Description
enViCscType	YUV CSC standard
u32LumaVal	Luminance adjustment The default value is 50. The value range is [0, 100].
u32ContrVal	Contrast adjustment The default value is 50. The value range is [0, 100].
u32HueVal	Hue adjustment The default value is 50. The value range is [0, 100].
u32SatuVal	Saturation adjustment The default value is 50. The value range is [0, 100].

[Note]

None

[See Also]

- VI_CSC_TYPE_E
- HI_MPI_VI_SetCSCAttr
- HI_MPI_VI_GetCSCAttr

VI_FLASH_MODE_E

[Description]

Defines the camera flash mode.

[Syntax]

```
typedef enum hiVI_FLASH_MODE_E
{
    VI_FLASH_ONCE = 0,          /*Blink once*/
    VI_FLASH_FREQ = 1,         /*Blink frequently*/
    VI_FLASH_MODE_BUTT
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
}VI_FLASH_MODE_E;
```

[Member]

Member	Description
VI_FLASH_ONCE	Single-blink mode. The camera flash blinks only once.
VI_FLASH_FREQ	Multi-blink mode. The camera flash blinks periodically.

[Note]

None

[See Also]

VI_FLASH_CONFIG_S

VI_FLASH_CONFIG_S

[Description]

Defines camera flash settings.

[Syntax]

```
typedef struct hiVI_FlASH_CONFIG_S
{
    VI_FLASH_MODE_E enFlashMode;
    HI_U32 u32StartTime;
    HI_U32 u32Duration;
    HI_U32 u32CapFrmIndex;
    HI_U32 u32Interval;
}VI_FLASH_CONFIG_S;
```

[Member]

Member	Description
enFlashMode	Camera flash mode
u32StartTime	Start time of the camera flash. Its unit is clock cycle.
u32Duration	Duration in which the camera flash is on. Its unit is clock cycle.
u32CapFrmIndex	Frame to be captured. This member is used to set the frame to be captured after the camera flash blinks. The value 0 indicates the current frame is the one to be captured. Value range: [0, 5]
u32Interval	Blink interval. Its unit is frame.

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- When the next frame is received after HI_MPI_VI_FlashTrigger is called, the VI device starts to trigger the camera flash signal u32StartTime clock cycles later after the field start interrupt is generated. The camera flash is on for u32Duration clock cycles. When the camera flash is set based on sensor timings, u32StartTime must include the clock cycle of the blanking area. The clock cycles are equal to the number of pixels.
- The sum of u32StartTime and u32Duration cannot be greater than the total clock cycles of a frame.
- In multi-blink mode, the camera flash signal is triggered for multiple times based on the interval defined in u32Interval. In single-blink mode, the camera flash signal is triggered only once, and u32Interval is invalid.
- In multi-blink mode, u32Interval must be greater than u32CapFrmIndex.

[See Also]

- VI_CSC_ATTR_S
- HI_MPI_VI_SetFlashConfig
- HI_MPI_VI_GetFlashConfig

VI_CHN_LUM_S

[Description]

Defines the channel luminance statistics.

[Syntax]

```
typedef struct hiVI_CHN_LUM_S
{
    HI_U32 u32Lum;
    HI_U64 u64Pts;
} VI_CHN_LUM_S;
```

[Member]

Member	Description
u32Lum	Luminance statistics
u64Pts	PTS of the obtained frame

[Note]

None

[See Also]

HI_MPI_VI_GetChnLuma

VI_PORTMODE_E

[Description]

Defines the VI port mode.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
typedef enum hiVI_PORTMODE_E
{
    VI_PORTMODE_D1 = 0,              /* D1/960H mode */
    VI_PORTMODE_720P,                /* 720P mode */
    VI_PORTMODE_BUTT,
} VI_PORTMODE_E;
```

[Member]

Member	Description
VI_PORTMODE_D1	D1 or 960H mode
VI_PORTMODE_720P	720p mode

[Note]

None

[See Also]

- HI_MPI_VI_SetPortMode
- HI_MPI_VI_GetPortMode

3.6 Error Codes

Table 3-17 describes the error codes of VI APIs.

Table 3-17 Error codes of VI APIs

Error Code	Macro Definition	Description
0xA0108001	HI_ERR_VI_INVALID_DEVID	The VI device ID is invalid.
0xA0108002	HI_ERR_VI_INVALID_CHNID	The VI channel ID is invalid.
0xA0108003	HI_ERR_VI_INVALID_PARA	The VI parameter is invalid.
0xA0108006	HI_ERR_VI_INVALID_NULL_PTR	The pointer of the input parameter is null.
0xA0108007	HI_ERR_VI_FAILED_NOTCONFIG	The attributes of the video device are not set.
0xA0108008	HI_ERR_VI_NOT_SUPPORT	The operation is not supported.
0xA0108009	HI_ERR_VI_NOT_PERM	The operation is forbidden.
0xA010800C	HI_ERR_VI_NOMEM	The memory fails to be allocated.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Macro Definition	Description
0xA010800E	HI_ERR_VI_BUF_EMPTY	The VI buffer is empty.
0xA010800F	HI_ERR_VI_BUF_FULL	The VI buffer is full.
0xA0108010	HI_ERR_VI_SYS_NOTREADY	The VI system is not initialized.
0xA0108012	HI_ERR_VI_BUSY	The VI system is busy.
0xA0108040	HI_ERR_VI_FAILED_NOTENABLE	The VI device or VI channel is not enabled.
0xA0108041	HI_ERR_VI_FAILED_NOTDISABLE	The VI device or VI channel is not disabled.
0xA0108042	HI_ERR_VI_FAILED_CHNOTDISABLE	The VI channel is not disabled.
0xA0108043	HI_ERR_VI_CFG_TIMEOUT	The video attribute configuration times out.
0xA0108045	HI_ERR_VI_INVALID_WAYID	The video channel ID is invalid.
0xA0108046	HI_ERR_VI_INVALID_PHYCHNID	The physical video channel ID is invalid.
0xA0108047	HI_ERR_VI_FAILED_NOTBIND	The video channel is not bound.
0xA0108048	HI_ERR_VI_FAILED_BINDED	The video channel is bound.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Contents

4 VO 4-1
4.1 Overview 4-1
4.2 Concepts 4-2
4.3 API Reference 4-7
4.4 Data Types 4-143
4.5 Error Codes 4-165

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figures

Figure 4-1 Concepts related to video layer attributes 4-150

Figure 4-2 Principle of partial zoom-in 4-157

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Tables

Table 4-1 Supported display devices, graphics layers, and cursor layers 4-1

Table 4-2 Differences between HD and SD display devices 4-2

Table 4-3 Error codes of VO APIs 4-165

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

4 VO

4.1 Overview

The video output unit (VOU) reads video and graphics data from the DDR and then outputs the data through display devices. Table 4-1 describes the display devices, graphics layers, and cursor layers supported by the Hi3531, Hi3532, and Hi3521.

Table 4-1 Supported display devices, graphics layers, and cursor layers

Item	Display Devices		Graphics Layer	Cursor Layer
	HD Device	SD Device		
Hi3531	Two high-definition (HD) display devices (DHD0 and DHD1) The two HD display devices support picture-in-picture (PIP) overlay.	Six standard-definition (SD) display devices (DSD0–DSD5) The six SD devices support channel overlay by priority.	Five graphics layers (G0–G4)	Two cursor layers (HC0 and HC1)
	Each display device can display the pictures from a maximum of 64 channels.			
Hi3532	One HD display devices (DHD0) DHD0 does not support PIP overlay.	N/A	One graphics layer (G0)	One cursor layer (HC0)
Hi3521/Hi3520A	One HD display device (DHD0) DHD0 supports PIP overlay. The PIP layer is multiplexed with the DSD0 layer. When the PIP layer is used,	Two SD display devices (DSD0 and DSD1) The two SD devices support channel overlay by priority.	Three graphics layers (G0–G2)	One cursor layer (HC0)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Item	Display Devices		Graphics Layer	Cursor Layer
	HD Device	SD Device		
	DSD0 has no output.			
	Each display device can display the pictures from a maximum of 32 channels.			
Hi3518/Hi3516C	One SD display device (DSD1). It can display the pictures from a maximum of 32 channels.		One graphics layer (G3)	No cursor layer
Hi3520D/Hi3515A/Hi3515C	One HD display device (DHD0) DHD0 supports PIP. The PIP layer is multiplexed with the DSD0 layer. When the PIP layer is used, DSD0 has no output.	Two SD devices (DSD0 and DSD1) DSD0 and DSD1 supports channel overlay by priority.	Three graphics layer (G0–G2)	One cursor layer (HC0)
	Each HD device can display the pictures from a maximum of 16 channels, and each SD device can display the pictures from a maximum of 32 channels.			

4.2 Concepts

Note the following:

- HD and SD display devices

  In the SDK, each HD display device is named DHDx, and each SD display device is named DSDx. *x* is numbered from 0. For example, HD display device 0 is DHD0, and SD display device 0 is DSD0. For details about the display devices supported by each chip, see Table 4-1.

  Table 4-2 describes the differences between HD and SD display devices.

Table 4-2 Differences between HD and SD display devices

Chip		Maximum Resolution	Output Interface	Scaling	Number of Supported Channels	PIP Display	Color Space Conversion Adjustment
Hi3531	HD	DHD0: 1920x1080 DHD1: 2560x1600	BT.1120 /HDMI /YPbPr /VGA	Zoom in only	64	Eight channels are supported (PIP layer).	Supported

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip		Maximum Resolution	Output Interface	Scaling	Number of Supported Channels	PIP Display	Color Space Conversion Adjustment
	SD	720x576	BT.656 /CVBS	None	64	Supported (TDE overlay)	Supported
Hi3532	HD	1920x1080	BT.1120	Not supported	64	Not supported	Supported
	SD	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
Hi3521/Hi3520A	HD	1920x1080	BT.1120 /HDMI /LCD /VGA	Zoom in only	32	Eight channels are supported (PIP layer).	Supported
	SD	720x576	BT.656 /CVBS	Not supported	32	Supported (TDE overlay)	Supported
Hi3518/Hi3516C	HD	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
	SD	1920x1080	BT.1120 /CVBS	Not supported	32	Not supported	Supported
Hi3520D/Hi3515A/Hi3515C	HD	1920x2048	HDMI /VGA	Zoom in only	16	One channel is supported (hardware PIP layer).	Supported
	SD	720x576	CVBS	Not supported	32	TDE overlay	Supported

- Scaling and displaying

  Each HD device supports a maximum of x channels. For details about the value of x, see Table 4-1. The priority in the channel attributes determines whether a channel belongs to the device video layer or PIP layer. The channels at the video layer or PIP layer cannot be overlaid. As the channels of HD devices do not support scaling, the external video process subsystem (VPSS) is required. After the source pictures in the video input (VI) channel or video decoding (VDEC) channel are scaled by the VPSS, the pictures are output to a VO channel. If the output picture size is greater than the region size, the picture is cropped. The HD device can zoom in on the entire video layer, but cannot crop it. Note that the PIP layer cannot be scaled or cropped.
- Each SD device supports a maximum of x channels. For details about the value of x, see Table 4-1. The channels are scaled by using the TDE. The pictures in the channels are overlaid as one picture and output to the display device. The SD device cannot zoom in

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

on the entire video layer, but can crop it. For details about the scaling and display functions of each chip, see Table 4-2.

CAUTION

- Cropping indicates that when the width or height of the canvas (stImageSize) is greater than the width or height of the display device resolution (stDispRect), the canvas is cropped to fit into the display device resolution.
- Scaling indicates that when the width or height of the display device resolution (stDispRect) is greater than the width or height of the canvas (stImageSize), the canvas is scaled to fit into the display device resolution.
- The size of a private VB pool is subject to the canvas size.

- Channel priority

  The HD device allows multiple channels to output pictures at the same time. The channel priorities include priority 0 and priority 1. The priority takes effect only after the PIP function is enabled. The SD device allows multiple channels to output pictures at the same time and overlays output pictures by priority. When the pictures of channels overlap, the picture with the highest priority is displayed on the top. If the channel priorities are the same, the channel with the largest channel ID takes priority over other channels by default.

  The channel priority of the Hi3532 can be set only to 0.

- PIP overlay

  The PIP function enables the picture at the PIP layer of the HD device to be overlaid with the picture of the original video layer. The original video layer of the HD device and PIP layer do not support channel overlay. The channel priorities of the HD device include priority 0 and priority 1. The picture of the channel with priority 0 is displayed at the original video layer, and the picture of the channel with priority 1 is displayed at the PIP layer. The picture at the original video layer is overlaid with the picture at the PIP layer. The pictures of a maximum of eight channels can be displayed. For details about the PIP overlay function of each chip and the number of supported PIP channels, see Table 4-2.

- Resolution

  Resolutions are categorized into:

  - Device resolution. It is the number of valid output pixels of a device that depends on the device timing.
  - Display resolution. It is the valid display region on a display device.
  - Picture resolution. It is the number of active pixels of a picture.

- Partial enlargement

  The SD device supports partial enlargement on a picture. The enlarged original area is clipped from the original picture and the size of the enlarged target area is the size of the display channel. The HD device can partially enlarge the picture of a channel only when it works with the VPSS, because only the VPSS supports scaling.

- Binding graphics layers

  It indicates that a graphics layer or cursor layer is bound to a device.

  The Hi3531 supports five graphics layers (G0–G4) and two cursor layers (HC0 and HC1). G0, G1, G2, and G3 are always bound to DHD0, DHD1, DSD0, and DSD1

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

respectively. G4, HC0, or HC1 can be bound to DHD0, DHD1, DSD0, or DSD1 as required.

Note:

- G4, HC0, or HC1 can be bound only to a device. For example, if G4 is bound to DHD0, it cannot be bound to DHD1, DSD0, or DSD1.
- HC0 and HC1 cannot be bound to DHD0 at the same time. If HC0 is bound to DHD0, HC1 must be bound to DHD1, DSD0, or DSD1; if HC1 is bound to DHD0, HC0 must be bound to DHD1, DSD0, or DSD1.
- G4, HC0, or HC1 can be bound to DHD0, DHD1, DSD0, or DSD1 as required. However, the bound device cannot be dynamically switched. That is, the bound device can be bound only after the bound device and the device to be bound are disabled. For example, if G4 is bound to DHD0 and you want to bind G4 to DHD1, you must disable G4 before binding G4 to DHD1.

  The Hi3532 supports one graphics layer (G0) and one cursor layer (HC0). Both Hi3532 G0 and HC0 are always bound to DHD0.

  The Hi3521, Hi3520A, Hi3520D, Hi3515A, or Hi3515C supports three graphics layers (G0–G2) and one cursor layer (HC0). G0, G1, and G2 are always bound to DHD0, DSD0, and DSD1 respectively. HC0 can be bound to DHD0 or DSD0 as required.

  The Hi3518/Hi3516C supports one graphics layer (G3) and always bound to DSD1.

- Binding the PIP layer

  It indicates that the video layer of an SD device is bound to the video layer of an HD device. Then the two video layers are overlaid with each other. The overlaid layer can be the PIP layer of the HD device or the video layer of the SD device at a time.

  If the PIP layer is bound to DHD0 and you want to bind the PIP layer to DHD1, you must disable the PIP layer, unbind the PIP layer from DHD0, and then bind the PIP layer to DHD1. If the PIP layer is not bound to any HD device, the layer acts as the video layer of an SD device.

  - The Hi3531 VOU allows the PIP layer to be overlaid with the video layer of an HD device. The HD device is DHD0 or DHD1.
  - The Hi3532/ Hi3518/Hi3516C VOU does not support the PIP layer.
  - The Hi3521/Hi3520A/Hi3520D VOU allows the video layer of DSD0 to act as the PIP layer that is overlaid with the video layer of DHD0.

- Video WBC

  The video WBC function is that the pictures displayed on the screen of an HD device are written to the DDR after being scaled. The pictures include the pictures obtained after the video layer, graphics layer, and cursor layer are overlaid. The WBC module can be considered as a video source. When the video source is bound to an SD device, the written pictures are re-transmitted to the SD device. This enables pictures from the same source to be displayed on an HD device and an SD device at the same time. The WBC module supports only zoom-out. The resolution of an HD device must be greater than or equal to the WBC resolution. The WBC module supports semi-plannar422 and semi-plannar420 data formats. The source data format can be different from the WBC data format.

  - The Hi3531 VOU supports DHD1 that provides the video WBC function. When the system binding API is called, the device ID corresponding to the WBC video source is DHD1, the channel ID is 0, and the module is VOU.

    When the WBC data format is semi-planar422, the picture can be zoomed out at most 16 times; when the data format is semi-planar420, the picture can be zoomed out at most eight times.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- The Hi3532/ Hi3518/Hi3516C VOU does not support WBC.
- The Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C VOU supports DHD0 that provides the video WBC function. Before calling the system binding API, note that the device ID corresponding to the WBC video source is DHD0, the channel ID is 0, and the module is VOU. When the WBC data format is semi-planar422, the picture can be zoomed out at most 16 times; when the data format is semi-planar420, the picture can be zoomed out at most eight times.

- Video cascade

  The output end of a slave chip connects to the VI end of the other chip to output BT.1120 data. In this way, multiple chips are connected and the pictures of slave chips are transferred to the master chip for displaying. The HD device of the Hi3531/Hi3532 supports video cascade. For details about the video cascade function, see the *Hi3531/Hi3532 Video Cascade Application Notes*.

  The Hi3521/Hi3520A/ Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C VOU does not support Video cascade.

- VGA compatibility

  The VGA interface can be connected as follows:

  The VGA interface has I^2C lines. You can simulate the I^2C function by using GPIO pins or direct use the I^2C lines. When the VGA monitor is connected, you can read the contents at addresses 0x0 to 0x70 of device 0xA0 over the I^2C bus to obtain the extended display identification data (EDID) information. The device addresses are fixed and are irrelevant to hardware.

  After the EDID information is parsed, the supported timings can be obtained. The methods are as follows:

  - Read the contents at 0x23 of device 0xA0 over the I^2C bus to obtain setup timing 1 of the monitor. Bit 7 to bit 0 correspond to 720x400@70 Hz, 720x400@88 Hz, 640x480@60 Hz, 640x480@67 Hz, 640x480@72 Hz, 640x480@75 Hz, 800x600@56 Hz, and 800x600@60 Hz respectively.
  - Read the contents at 0x24 of device 0xA0 over the I^2C bus to obtain setup timing 2 of the monitor. Bit 7 to bit 0 correspond to 800x600@72 Hz, 800x600@75 Hz, 832x624@75 Hz, 1024x768@87 Hz (interleaved or interlaced), 1024x768@60 Hz, 1024x768@70 Hz, 1024x768@75 Hz, 1280x1024@75 Hz respectively.
  - Read the contents at 0x25 of device 0xA0 over the I^2C bus to obtain the timings reserved by the manufacturer. The value 00h indicates that no timings are reserved, and bit 7 corresponds to 1152x870@ 75 Hz (Mac II, Apple).
  - Read the contents at 0x26 to 0x27 of device 0xA0 over the I^2C bus to obtain the identification results for standard timings. Two bytes such as 0x26 and 0x27 or 0x28 and 0x29 constitute a group. The first byte such as 0x26 indicates the horizontal result. The horizontal value is calculated as follows: (Read value + 31) x 8Bit 5 to bit 0 of the second byte such as 0x27 indicate the vertical frequency. It is the read value added by 60. Bit 7 and bit 6 of the second byte indicate the aspect ratio. For example, the value 00 indicates 16:10, 01 indicates 4:3, 10 indicates 5:4, and 11 indicates 16:9.

  For details, see the *E-EDID Standard*.

  If the return analysis result indicates that the contents are valid, you need to check whether the monitor supports the timing to be configured. If the return analysis result indicates that the contents are invalid, the monitor may not support the EDID protocol. In this case, you need to configure the VGA timings supported by the VOU until VGA output is normal.

- Virtual device

  Each virtual device has no corresponding physical device.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- The internal processing mode of virtual devices is the same as that of SD devices. To be specific, the picture from the channels of SD devices or virtual devices are scaled by the TDE, and then overlaid as a picture.
- Virtual devices allow you to obtain pictures from them by calling HI_MPI_VO_GetScreenFrame in user state, and then transmit the pictures to other modules. The pictures from virtual devices can also be transmitted to other modules by calling HI_MPI_SYS_Bind in kernel state. However, a virtual VO device cannot be bound to its corresponding VO device by calling HI_MPI_SYS_Bind.
- The timings of virtual devices are not restricted to phase alternating line (PAL) norm or National Television Systems Committee (NTSC) norm. The configured timing determines the size of the overlaid picture from the TDE. For example, if the timing of a virtual device is set to 1080p60, it will transmit 1920x1080 pictures to other modules. When virtual devices are used, the TDE performance and total service volume must be considered.
- A VO virtual device can be bound to the VPSS channel specified by VPSS_BYPASS_CHN or a VPSS channel whose mode is set to VPSS_CHN_MODE_USER by calling HI_MPI_VPSS_SetChnMode.

The Hi3531 or Hi3532 supports four virtual devices: device 10, device 11, device 12, and device 13.

The Hi3521, Hi3520A, Hi3520D, Hi3515A, or Hi3515C supports four virtual devices: device 3, device 4, device 5, and device 6.

The Hi3518 or Hi3516C does not support virtual devices.

CAUTION

- Do not bind HD or cascade devices to the VDEC directly. Otherwise, decoding and playback lose control.
- SD or virtual devices cannot be bound to VPSS physical channels in AUTO mode.

4.3 API Reference

The VOU supports multiple functions including enabling the VO device or VO channel and transmitting video data to the output channel.

The VOU provides the following MPP programming interfaces (MPIs).

The MPIs related to device operations are as follows:

- HI_MPI_VO_Enable: Enables a VO device.
- HI_MPI_VO_Disable: Disables a VO device.
- HI_MPI_VO_SetPubAttr: Sets the public attributes of a VO device.
- HI_MPI_VO_GetPubAttr: Obtains the public attributes of a VO device.
- HI_MPI_VO_CloseFd: Disables the file descriptors (FDs) of all the VO devices.
- HI_MPI_VO_SetDevFramerate: Sets the frame rate of a device when the user timing is used.
- HI_MPI_VO_GetDevFramerate: Obtains the frame rate of a device.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_VO_EnableSdTdeBypass: Enables the TDE bypass function of an SD device.
- HI_MPI_VO_DisableSdTdeBypass: Disables the TDE bypass function of an SD device.

The MPIs related to video layer operations are as follows:

- HI_MPI_VO_EnableVideoLayer: Enables a video layer.
- HI_MPI_VO_DisableVideoLayer: Disables a video layer.
- HI_MPI_VO_SetVideoLayerAttr: Sets the attributes of a video layer.
- HI_MPI_VO_GetVideoLayerAttr: Obtains the attributes of a video layer.

The MPIs related to PIP operations are as follows:

- HI_MPI_VO_PipLayerBindDev: Binds the PIP layer to a specified device.
- HI_MPI_VO_PipLayerUnBindDev: Unbinds the PIP layer from a specified device.
- HI_MPI_VO_SetPipLayerAttr: Sets the attributes of the PIP layer.
- HI_MPI_VO_GetPipLayerAttr: Obtains the attributes of the PIP layer.
- HI_MPI_VO_EnablePipLayer: Enables the PIP layer.
- HI_MPI_VO_DisablePipLayer: Disables the PIP layer.

The MPIs related to channel operations are as follows:

- HI_MPI_VO_EnableChn: Enables a specified VO channel.
- HI_MPI_VO_DisableChn: Disables a specified VO channel.
- HI_MPI_VO_SetChnAttr: Sets the attributes of a specified VO channel.
- HI_MPI_VO_GetChnAttr: Obtains the attributes of a specified VO channel.
- HI_MPI_VO_SetChnDisplayWindow: Sets the display window of a specified VO channel.
- HI_MPI_VO_GetChnDisplayWindow: Obtains the display window of a specified VO channel.
- HI_MPI_VO_SetChnDispPos: Sets the display position of a specified VO channel.
- HI_MPI_VO_GetChnDispPos Obtains the display position of a specified VO channel.
- HI_MPI_VO_SetChnField: Sets the frame/field display policy of a specified VO channel.
- HI_MPI_VO_GetChnField: Obtains the frame/field display policy of a specified VO channel.
- HI_MPI_VO_SendFrame: Sends video pictures to a specified VO channel.
- HI_MPI_VO_SetChnFrameRate: Sets the display frame rate of a specified VO channel.
- HI_MPI_VO_GetChnFrameRate: Obtains the display frame rate of a specified VO channel.
- HI_MPI_VO_ChnPause: Pauses a specified VO channel.
- HI_MPI_VO_ChnResume: Resumes a specified VO channel.
- HI_MPI_VO_ChnStep: Plays the streams in a specified VO channel by frame.
- HI_MPI_VO_ChnRefresh: Refreshes a specified VO channel.
- HI_MPI_VO_EnableRecvFrameRateMatch: Enables the frame rate match function of a specified VO channel.
- HI_MPI_VO_DisableRecvFrameRateMatch: Disables the frame rate match function of a specified VO channel.
- HI_MPI_VO_SetZoomInWindow: Sets a VO zoom-in window.
- HI_MPI_VO_GetZoomInWindow: Obtains the parameters of a VO zoom-in window.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_VO_GetChnPts: Obtains the time stamp of the current picture in a specified VO channel.
- HI_MPI_VO_SetAttrBegin: Starts attribute setting.
- HI_MPI_VO_SetAttrEnd: Ends attribute setting.
- HI_MPI_VO_ChnShow: Shows a specified channel.
- HI_MPI_VO_ChnHide: Hides a specified channel.
- HI_MPI_VO_QueryChnStat: Queries the status of a VO channel.
- HI_MPI_VO_GetChnFrame: Obtains the pictures of a VO channel.
- HI_MPI_VO_ReleaseChnFrame: Releases the buffer for storing the pictures of a VO channel.
- HI_MPI_VO_ClearChnBuffer: Clears the data in a specified VO channel buffer.
- HI_MPI_VO_SetChnDispThreshold: Sets the display threshold for a VO channel.
- HI_MPI_VO_GetChnDispThreshold: Obtains the display threshold for a VO channel.
- HI_MPI_VO_EnableChnDoubleFrame: Enables the double-frame-rate function of a channel.
- HI_MPI_VO_DisableChnDoubleFrame: Disables the double-frame-rate function of a channel.

The MPIs related to display operations are as follows:

- HI_MPI_VO_GetScreenFrame: Obtains the pictures displayed on the screen.
- HI_MPI_VO_ReleaseScreenFrame: Releases the buffer for storing the pictures displayed on the screen.
- 0HI_MPI_VO_SetDevCSC: Sets the effect of the output pictures from a device.
- HI_MPI_VO_GetDevCSC: Obtains the effect of the output pictures from a device.
- HI_MPI_VO_SetVgaParam: Sets the effect of the VGA output pictures from a device.
- HI_MPI_VO_GetVgaParam: Obtains the effect of the VGA output pictures from a device.
- HI_MPI_VO_SetDispBufLen: Sets the size of a display buffer.
- HI_MPI_VO_GetDispBufLen: Obtains the size of a display buffer.
- HI_MPI_VO_SetPlayToleration: Sets the play tolerance.
- HI_MPI_VO_GetPlayToleration: Obtains the play tolerance.
- HI_MPI_VO_SetVtth: Sets the vertical time sequence interruption threshold of a device.
- HI_MPI_VO_GetVtth: Obtains the vertical time sequence interruption threshold of a device.

The MPIs related to writeback are as follows:

- HI_MPI_VO_EnableWbc: Enables WBC.
- HI_MPI_VO_DisableWbc: Disables WBC.
- HI_MPI_VO_SetWbcAttr: Sets the WBC attribute.
- HI_MPI_VO_GetWbcAttr: Obtains the WBC attribute.
- HI_MPI_VO_SetWbcMode: Sets the WBC mode.
- HI_MPI_VO_GetWbcMode: Obtains the WBC mode.
- HI_MPI_VO_SetWbcDepth: Sets the depth of the buffer for storing WBC pictures.
- HI_MPI_VO_GetWbcDepth: Obtains the depth of the buffer for storing WBC pictures.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_VO_WbcGetScreenFrame: Obtains WBC pictures.
- HI_MPI_VO_WbcReleaseScreenFrame: Releases the buffer for storing WBC pictures.

The MPIs related to graphics layer operations are as follows:

- HI_MPI_VO_GfxLayerBindDev: Binds a graphics layer to a device.
- HI_MPI_VO_GfxLayerUnBindDev: Unbinds a graphics layer from a device.
- HI_MPI_VO_SetGfxLayerCSC: Sets the picture output effect of a graphics layer.
- HI_MPI_VO_GetGfxLayerCSC: Obtains the picture output effect of a graphics layer.

The MPIs related to cascade are as follows:

- HI_MPI_VO_SetCascadeAttr: Sets the cascade attribute.
- HI_MPI_VO_GetCascadeAttr: Obtains the cascade attribute.
- HI_MPI_VO_EnableCascadeDev: Enables an extended cascade device.
- HI_MPI_VO_DisableCascadeDev: Disables an extended cascade device.
- HI_MPI_VO_SetCascadePattern: Sets the picture pattern during video cascading.
- HI_MPI_VO_GetCascadePattern: Obtains the picture pattern during video cascading.
- HI_MPI_VO_CascadePosBindChn: Binds a cascade region to a VO channel.
- HI_MPI_VO_CascadePosUnBindChn: Unbinds a cascade region from a VO channel.
- HI_MPI_VO_EnableCascade: Enables the video cascade function.
- HI_MPI_VO_DisableCascade: Disables the video cascade function.

HI_MPI_VO_Enable

[Description]

Enables a VO device.

[Syntax]

```
HI_S32 HI_MPI_VO_Enable (VO_DEV VoDev);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Range of the VO Device ID
Hi3531	[0, VO_MAX_DEV_NUM), excluding 8 and 9
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A /Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_SYS_NOTREADY	The system is not ready, that is, the device FD is not opened.
HI_ERR_VO_DEV_NOT_CONFIG	The VO device is not configured.
HI_ERR_VO_DEV_HAS_ENABLED	The VO device has been enabled.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- As the VO device is not enabled through system initialization, you must enable the VO device before using the video output function.
- Before calling this MPI, you must configure the public attributes of the VO device. Otherwise, an error message is returned, indicating that the VO device is not configured.
- To ensure that the startup screen and the operation screen are smoothly switched, you need to check whether the VO device is enabled. If the VO device is enabled, SUCCESS is returned and the existing timing configuration is used. If you want to change the VO timing configuration, you need to call HI_MPI_VO_Disable to disable the VO device and then call HI_MPI_VO_Enable to enable the VO device again.
- For details about each device, see the description of VO_DEV.

[Example]

```
HI_S32 s32Ret;
VO_DEV VoDev = 0;
VO_PUB_ATTR_S stPubAttr;

s32Ret = HI_MPI_VO_GetPubAttr(VoDev, &stPubAttr);
if (s32Ret != HI_SUCCESS)
{
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
printf("Get device attributes failed with error code %#x!\n", s32Ret);
return HI_FAILURE;
}

stPubAttr.u32BgColor = 0xff;
stPubAttr.enIntfType = VO_INTF_VGA;
stPubAttr.enIntfSync = VO_OUTPUT_1280x1024_60;

s32Ret = HI_MPI_VO_SetPubAttr(VoDev, &stPubAttr);
if (s32Ret != HI_SUCCESS)
{
printf("Set device attributes failed with errno %#x!\n", s32Ret);
return HI_FAILURE;
}

s32Ret = HI_MPI_VO_Enable(VoDev);
if (s32Ret != HI_SUCCESS)
{
   printf("Enable vo dev %d failed with errno %#x!\n", VoDev, s32Ret);
   return HI_FAILURE;
}

s32Ret = HI_MPI_VO_Disable(VoDev);
if (s32Ret != HI_SUCCESS)
{
   printf("Enable vo dev %d failed with errno %#x!\n", VoDev, s32Ret);
   return HI_FAILURE;
}

s32Ret = HI_MPI_VO_CloseFd();
if (s32Ret != HI_SUCCESS)
{
   printf("Some device is not disable with errno %#x!\n", VoDev, s32Ret);
   return HI_FAILURE;
}
```

[See Also]

HI_MPI_VO_Disable

HI_MPI_VO_Disable

[Description]

Disables a VO device.

[Syntax]

Copyright © HiSilicon Technologies Co., Ltd

```
HI_S32 HI_MPI_VO_Disable(VO_DEV VoDev);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Range of the VO Device ID
Hi3531	[0, VO_MAX_DEV_NUM), excluding 8 and 9
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D /Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_SYS_NOTREADY	The system is not ready, that is, the device FD is not opened.
HI_ERR_VO_VIDEO_NOT_DISABLE	The video layer is not disabled.
HI_ERR_VO_WBC_NOT_DISABLE	The WBC is not disabled

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Before disabling a device, you must disable its video layer.
- If the WBC is enabled for a device is disabled, disable the WBC.
- If you do not call HI_MPI_VO_Disable after calling HI_MPI_VO_Enable, the VO device is always enabled, and the device attributes that are set next time do not take effect.
- If a device is disabled, it can be enabled only after public attributes are set again.

[Example]

See the example of HI_MPI_VO_Enable.

[See Also]

HI_MPI_VO_Enable

HI_MPI_VO_SetPubAttr

[Description]

Sets the public attributes of a VO device.

[Syntax]

```
HI_S32 HI_MPI_VO_SetPubAttr(VO_DEV VoDev, const VO_PUB_ATTR_S
*pstPubAttr);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pstPubAttr	Pointer to the public attributes of a VO device Static attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531	Automatic load detection is not supported during CVBS output. Value range of VoDev: [0, VO_MAX_DEV_NUM), excluding VO_CAS_DEV_1 and VO_CAS_DEV_2
Hi3532	Automatic load detection is not supported during CVBS output.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Description
	Value range of VoDev: 0 and [10, 13]
Hi3521/Hi3520A /Hi3520D/Hi3515A/Hi3515C	Automatic load detection is not supported during CVBS output. Value range of VoDev: [0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	Automatic load detection is not supported during CVBS output. For details, see the **Note** field. Value range of VoDev: [0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_NULL_PTR	The parameter pointer is null.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_SYS_NOTREADY	The system is not ready, that is, the device FD is not opened.
HI_ERR_VO_ILLEGAL_PARAM	The parameter is invalid.
HI_ERR_VO_NOT_SUPPORT	The VO attributes are not supported.
HI_ERR_VO_DEV_HAS_ENABLED	The VO device has been enabled.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- The attributes of a VO device are static attributes. Therefore, you must set the attributes before calling HI_MPI_VO_Enable.
- For details about each device, see the description of VO_DEV.
- For details about the attributes of a VO device, see the description of VO_PUB_ATTR_S.
- When the interface type enIntfType (one of VO device attributes) is set to VO_INTF_CVBS, the Hi3518/Hi3516C supports automatic load detection.
  - Working principle: The system periodically checks whether the CVBS interface is connected by using a cable. If the CVBS interface is not connected, the driver automatically powers off the DAC to reduce power consumption.
  - Detection cycle: The load detection cycle can be configured by setting VO parameters. The detection cycle is calculated based on frame interrupts. The following example assumes that insmod hi3518_vou.ko detectCycle is 50. If the output timing enIntfSync (one of VO device attributes) is VO_OUTPUT_PAL, the detection cycle is 2s (50/25). If the output timing enIntfSync is VO_OUTPUT_NTSC, the detection cycle is 1.67s (50/30). If the VO driver is loaded when module

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

parameters are not set, the default load detection cycle is 30. If the VO module parameter detectCycle is set to 0, automatic load detection is disabled.

[Example]

See the example of HI_MPI_VO_Enable.

[See Also]

HI_MPI_VO_GetPubAttr

HI_MPI_VO_GetPubAttr

[Description]

Obtains the public attributes of a VO device.

[Syntax]

```
HI_S32 HI_MPI_VO_GetPubAttr(VO_DEV VoDev, VO_PUB_ATTR_S *pstPubAttr);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pstPubAttr	Pointer to the public attributes of a VO device	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Range of the VO Device ID
Hi3531	[0, VO_MAX_DEV_NUM), excluding VO_CAS_DEV_1 and VO_CAS_DEV_2
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/ Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_NULL_PTR	The parameter pointer is null.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_NOT_PERMIT	The cascade devices are not supported.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

You are advised to obtain the attributes of a VO device before setting public attributes. In this way, you only need to set the items to be changed.

[Example]

See the example of HI_MPI_VO_Enable.

[See Also]

HI_MPI_VO_SetPubAttr

HI_MPI_VO_CloseFd

[Description]

Disables the FDs of all the VO devices.

[Syntax]

```
HI_S32 HI_MPI_VO_CloseFd(HI_VOID);
```

[Parameter]

None

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Definition
HI_ERR_VO_BUSY	The device is busy. That is, the device is not disabled.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

This MPI must be called when all the VO devices are disabled.

[Example]

See the example of HI_MPI_VO_Enable.

[See Also]

None

HI_MPI_VO_SetDevFramerate

[Description]

Sets the frame rate of a device when the user timing is used.

[Syntax]

```
HI_S32 HI_MPI_VO_SetDevFramerate(VO_DEV VoDev, HI_U32 u32Framerate);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
u32Framerate	Frame rate of a device The frame rate must be 25, 30, 50, or 60.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Library file: libmpi.a

[Note]

- This MPI can be called only when the user timing is used.
- This MPI must be called after HI_MPI_VO_SetPubAttr is called and before HI_MPI_VO_Enable is called.

[Example]

None

[See Also]

HI_MPI_VO_GetDevFramerate

HI_MPI_VO_GetDevFramerate

[Description]

Obtains the frame rate of a device.

[Syntax]

```
HI_S32 HI_MPI_VO_GetDevFramerate(VO_DEV VoDev, HI_U32 *pu32Framerate);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pu32Framerate	u32 pointer	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VO_SetDevFramerate

HI_MPI_VO_EnableSdTdeBypass

[Description]

Enables the TDE bypass function of an SD device.

[Syntax]

```
HI_S32 HI_MPI_VO_EnableSdTdeBypass (VO_DEV VoDev);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- This MPI applies only to the SD device.
- This MPI can be used only when one channel of the current SD device is enabled and the size of the input picture is the same as that of the displayed picture.

[Example]

None

[See Also]

None

HI_MPI_VO_DisableSdTdeBypass

[Description]

Disables the TDE bypass function of an SD device.

[Syntax]

```
HI_S32 HI_MPI_VO_DisableSdTdeBypass(VO_DEV VoDev);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

None

HI_MPI_VO_EnableVideoLayer

[Description]

Enables a video layer.

[Syntax]

```
HI_S32 HI_MPI_VO_EnableVideoLayer (VO_DEV VoDev);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input

[Return Value]

Return Value	Description
0	Success

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Range of the VO Device ID
Hi3531	[0, VO_MAX_DEV_NUM), excluding 8 and 9
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/ Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_SYS_NOTREADY	The system is not ready, that is, the device FD is not opened.
HI_ERR_VO_DEV_NOT_ENABLE	The VO device is not enabled.
HI_ERR_VO_VIDEO_NOT_CONFIG	The video layer is not configured.
HI_ERR_VO_NO_MEM	There is no available memory.
HI_ERR_VO_NOT_PERMIT	The cascade devices are not supported.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before enabling a video layer, ensure that the device where the video layer is located is enabled.
- Before enabling a video layer, ensure that the video layer is configured.

[Example]

```
HI_S32 s32Ret;
VO_DEV VoDev = 0;
RECT_S stRect;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
VO_VIDEO_LAYER_ATTR_S stLayerAttr;
s32Ret = HI_MPI_VO_GetVideoLayerAttr(VoDev, &stLayerAttr);
if (s32Ret != HI_SUCCESS)
{
   printf("Get video layer attributes failed with errno %#x!\n", s32Ret);
   return HI_FAILURE;
}
stLayerAttr.stDispRect.s32X = 0;
stLayerAttr.stDispRect.s32Y = 0;
stLayerAttr.stDispRect.u32Width = 720;
stLayerAttr.stDispRect.u32Height = 576;
stLayerAttr.stImageSize.u32Width = 720;
stLayerAttr.stImageSize.u32Height = 576;
stLayerAttr. u32DispFrmRt = 25;
stLayerAttr. enPixFormat = PIXEL_FORMAT_YUV_SEMIPLANAR_422;

s32Ret = HI_MPI_VO_SetVideoLayerAttr(VoDev, &stLayerAttr);
if (s32Ret != HI_SUCCESS)
{
   printf("Set video layer attributes failed with errno %#x!\n", s32Ret);
   return HI_FAILURE;
}

s32Ret = HI_MPI_VO_EnableVideoLayer(VoDev);
if (s32Ret != HI_SUCCESS)
{
   printf("Enable video layer failed with errno %#x!\n", s32Ret);
   return HI_FAILURE;
}

s32Ret = HI_MPI_VO_DisableVideoLayer(VoDev);
if (s32Ret != HI_SUCCESS)
{
   printf("Disable video layer failed with errno %#x!\n", s32Ret);
   return HI_FAILURE;
}
```

[See Also]

HI_MPI_VO_DisableVideoLayer

HI_MPI_VO_DisableVideoLayer

[Description]

Disables a video layer.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_VO_DisableVideoLayer(VO_DEV VoDev);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Range of the VO Device ID
Hi3531	[0, VO_MAX_DEV_NUM), excluding 8 and 9
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_SYS_NOTREADY	The system is not ready, that is, the device FD is not opened.
HI_ERR_VO_CHN_NOT_DISABLE	Some channels are not disabled at this video layer.
HI_ERR_VB_BUSY	The video buffer (VB) is not released.
HI_ERR_VO_WBC_NOT_DISABLE	The WBC is not disabled.

[Requirement]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before disabling a video layer, ensure that all the channels at the video layer are disabled.
- If you do not release the VB that stores the pictures obtained from the VO channel when disabling a video layer, the error code HI_ERR_VB_BUSY is returned, indicating that the VB created by the VOU is not released. This error generally occurs when you obtain screen pictures but the VB is not released.
- After the video layer is disabled, ensure that the WBC function of this device is disabled if the WBC data source is VO_WBC_DATASOURCE_VIDEO.

[Example]

See the example of HI_MPI_VO_EnableVideoLayer.

[See Also]

HI_MPI_VO_EnableVideoLayer

HI_MPI_VO_SetVideoLayerAttr

[Description]

Sets the attributes of a video layer.

[Syntax]

```
HI_S32 HI_MPI_VO_SetVideoLayerAttr(VO_DEV VoDev, const
VO_VIDEO_LAYER_ATTR_S *pstLayerAttr);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pstLayerAttr	Pointer to the attributes of a video layer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Range of the VO Device ID
Hi3531	[0, VO_MAX_DEV_NUM), excluding

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Range of the VO Device ID
	VO_CAS_DEV_1 and VO_CAS_DEV_2
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/ Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_SYS_NOTREADY	The system is not ready, that is, the device FD is not opened.
HI_ERR_VO_NOT_PERMIT	The cascade devices are not supported.
HI_ERR_VO_ILLEGAL_PARAM	The parameter is invalid.
HI_ERR_VO_VIDEO_NOT_DISABLE	The video layer is not disabled.
HI_ERR_VO_NULL_PTR	The parameter pointer is null.
HI_ERR_VO_DEV_NOT_CONFIG	The device is not configured.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- The attributes of a video layer can be set only when the video layer is disabled.
- For details about the attributes of a video layer, see the description of VO_VIDEO_LAYER_ATTR_S.

[Example]

See the example of HI_MPI_VO_EnableVideoLayer.

[See Also]

HI_MPI_VO_GetVideoLayerAttr

HI_MPI_VO_GetVideoLayerAttr

[Description]

Obtains the attributes of a video layer.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_VO_GetVideoLayerAttr(VO_DEV VoDev, VO_VIDEO_LAYER_ATTR_S
*pstLayerAttr);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pstLayerAttr	Pointer to the attributes of a video layer	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Range of the VO Device ID
Hi3531	[0, VO_MAX_DEV_NUM), excluding 8 and 9
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A /Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_SYS_NOTREADY	The system is not ready, that is, the device FD is not opened.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_NULL_PTR	The parameter pointer is null.
HI_ERR_VO_NOT_PERMIT	The cascade devices are not supported.

[Requirement]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- The attributes of the video layer can be obtained when HI_MPI_VO_EnableVideoLayer and HI_MPI_VO_SetVideoLayerAttr are not called.
- You are advised to obtain the attributes of a video layer before setting attributes.

[Example]

See the example of HI_MPI_VO_EnableVideoLayer.

[See Also]

HI_MPI_VO_SetVideoLayerAttr

HI_MPI_VO_PipLayerBindDev

[Description]

Binds the PIP layer to a specified device.

[Syntax]

```
HI_S32 HI_MPI_VO_PipLayerBindDev(VO_DEV VoTargetDev);
```

[Parameter]

Parameter	Description	Input/Output
VoTargetDev	ID of the device that is bound to the PIP layer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_SYS_NOTREADY	The system is not ready, that is, the device FD is not opened.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_DEV_HAS_ENABLED	The PIP layer has been enabled.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Definition
HI_ERR_VO_NOT_SUPPORT	The chip does not support the API.
HI_ERR_VO_DEV_HAS_BINDED	The PIP layer is bound to other devices.
HI_ERR_VO_NOT_PERMIT	The calling operation is forbidden. This MPI cannot be called when the PIP layer is bound to an SD device, an HD device in cascade mode is enabled, or a multiplexed SD device is enabled.

[Difference]

Chip	Description
Hi3531	The video layer of DSD5 can be used as the PIP layer of an HD device. The value of VoTargetDev is 0 or 1.
Hi3532	The Hi3532 does not support this MPI.
Hi3521/Hi3520A/Hi3520D/Hi3515A/ Hi3515C	The video layer of DSD0 can be used as the PIP layer of an HD device. The value of VoTargetDev is 0.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- If the PIP layer for an HD device is shared with the video layer of DSDx and DSDx is enabled, the error code HI_ERR_VO_DEV_HAS_ENABLED is returned if you bind the PIP layer to the HD device. Therefore, you must disable DSDx before binding the PIP layer.
- The PIP layer can be bound to a device repeatedly. During binding, the PIP layer must be disabled.
- Before binding the PIP layer to another device, you must unbind the PIP layer.
- If the PIP layer of an HD device is shared with the video layer of DSDx, the attributes of the same layer are set when you set the attributes of the PIP layer and the video layer of DSDx. The attributes are subject to the latest settings. You are advised to set the attributes of a video layer before enabling it.

[Example]

```
HI_S32 s32Ret;
VO_DEV VoDev = 0;
RECT_S stRect;
```


Copyright © HiSilicon Technologies Co., Ltd

```
VO_VIDEO_LAYER_ATTR_S stLayerAttr;

s32Ret = HI_MPI_VO_PipLayerBindDev(VoDev);
if (s32Ret != HI_SUCCESS)
{
   printf("Pip video layer bind to dev %d failed with errno %#x!\n",
VoDev, s32Ret);
   return HI_FAILURE;
}

s32Ret = HI_MPI_VO_GetPipLayerAttr(&stLayerAttr);
if (s32Ret != HI_SUCCESS)
{
   printf("Get pip video layer attributes failed with errno %#x!\n",
s32Ret);
   return HI_FAILURE;
}
stLayerAttr.stDispRect.s32X = 0;
stLayerAttr.stDispRect.s32Y = 0;
stLayerAttr.stDispRect.u32Width = 720;
stLayerAttr.stDispRect.u32Height = 576;

stLayerAttr.stImageSize.u32Width = 720;
stLayerAttr.stImageSize.u32Height = 576;
stLayerAttr. u32DispFrmRt = 25;
stLayerAttr. enPixFormat = PIXEL_FORMAT_YUV_SEMIPLANAR_422;

s32Ret = HI_MPI_VO_SetPipLayerAttr(&stLayerAttr);
if (s32Ret != HI_SUCCESS)
{
   printf("Set pip video layer attributes failed with errno %#x!\n",
s32Ret);
   return HI_FAILURE;
}

s32Ret = HI_MPI_VO_EnablePipLayer();
if (s32Ret != HI_SUCCESS)
{
   printf("Enable pip video layer failed with errno %#x!\n", s32Ret);
   return HI_FAILURE;
}

s32Ret = HI_MPI_VO_DisablePipLayer();
if (s32Ret != HI_SUCCESS)
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
{
    printf("Disable pip video layer failed with errno %#x!\n", s32Ret);
    return HI_FAILURE;
}

s32Ret = HI_MPI_VO_PipLayerUnBindDev(VoDev);
if (s32Ret != HI_SUCCESS)
{
    printf("Pip video layer unbind to dev %d failed with errno %#x!\n",
VoDev, s32Ret);
    return HI_FAILURE;
}
```

[See Also]

HI_MPI_VO_PipLayerUnBindDev

HI_MPI_VO_PipLayerUnBindDev

[Description]

Unbinds the PIP layer from a specified device.

[Syntax]

```
HI_S32 HI_MPI_VO_PipLayerUnBindDev(VO_DEV VoTargetDev);
```

[Parameter]

Parameter	Description	Input/Output
VoTargetDev	ID of the device that is bound to the PIP layer Value range: [0, VO_MAX_DEV_NUM]	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_SYS_NOTREADY	The system is not ready, that is, the device

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Definition
	FD is not opened.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_VIDEO_NOT_DISABLE	The PIP layer is not disabled.
HI_ERR_VO_NOT_SUPPORT	The chip does not support the API.

[Difference]

Chip	Description
Hi3531	The video layer of DSD5 can be used as the PIP layer of an HD device.
Hi3532	The Hi3532 does not support this MPI.
Hi3521/Hi3520A/Hi3520D/Hi3515A /Hi3515C	The video layer of DSD0 can be used as the PIP layer of an HD device.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before unbinding the PIP layer from an HD device, you must call HI_MPI_VO_DisablePipLayer to disable the PIP layer.
- A code indicating success is returned when this MPI is repeatedly called. The value of VoTargetDev can be the ID of the device that is not bound to the PIP layer before. Calling this MPI cancels the binding relationship between the PIP layer and devices.

[Example]

See the example of HI_MPI_VO_PipLayerBindDev.

[See Also]

HI_MPI_VO_SetPipLayerAttr

HI_MPI_VO_SetPipLayerAttr

[Description]

Sets the attributes of the PIP layer.

[Syntax]

```
HI_S32 HI_MPI_VO_SetPipLayerAttr(const VO_VIDEO_LAYER_ATTR_S
*pstLayerAttr);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
pstLayerAttr	Pointer to the attributes of the PIP layer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_SYS_NOTREADY	The system is not ready, that is, the device FD is not opened.
HI_ERR_VO_ILLEGAL_PARAM	The parameter is invalid.
HI_ERR_VO_VIDEO_NOT_DISABLE	The PIP layer is not disabled.
HI_ERR_VO_NULL_PTR	The parameter pointer is null.
HI_ERR_VO_DEV_NOT_BINDED	The PIP video layer is not bound.
HI_ERR_VO_NOT_SUPPORT	The chip does not support the API.

[Difference]

Chip	Description
Hi3531	The video layer of DSD5 can be used as the PIP layer of an HD device.
Hi3532	The Hi3532 does not support this MPI.
Hi3521/Hi3520A/Hi3520D/Hi3515A /Hi3515C	The video layer of DSD0 can be used as the PIP layer of an HD device.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.

[Requirement]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before setting the attributes of the PIP layer, you must bind it to an HD device.
- You are advised to obtain the attributes of the PIP layer before setting attributes.

[Example]

See the example of HI_MPI_VO_PipLayerBindDev.

[See Also]

HI_MPI_VO_GetPipLayerAttr

HI_MPI_VO_GetPipLayerAttr

[Description]

Obtains the attributes of the PIP layer.

[Syntax]

```
HI_S32 HI_MPI_VO_GetPipLayerAttr(VO_VIDEO_LAYER_ATTR_S *pstLayerAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstLayerAttr	Pointer to the attributes of the PIP layer	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_SYS_NOTREADY	The system is not ready, that is, the device FD is not opened.
HI_ERR_VO_NULL_PTR	The parameter pointer is null.
HI_ERR_VO_NOT_SUPPORT	The chip does not support the API.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Difference]

Chip	Description
Hi3531	The video layer of DSD5 can be used as the PIP layer of an HD device.
Hi3532	The Hi3532 does not support this MPI.
Hi3521/Hi3520A/Hi3520D/ Hi3515A/Hi3515C	The video layer of DSD0 can be used as the PIP layer of an HD device.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

Regardless of whether the attributes of the PIP layer are set or the PIP layer is enabled, the attributes of the PIP layer can be obtained.

[Example]

See the example of HI_MPI_VO_PipLayerBindDev.

[See Also]

HI_MPI_VO_SetPipLayerAttr

HI_MPI_VO_EnablePipLayer

[Description]

Enables the PIP layer.

[Syntax]

```
HI_S32 HI_MPI_VO_EnablePipLayer (HI_VOID);
```

[Parameter]

None

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

Copyright © HiSilicon Technologies Co., Ltd

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_SYS_NOTREADY	The system is not ready, that is, the device FD is not opened.
HI_ERR_VO_DEV_NOT_ENABLE	The VO device is not enabled.
HI_ERR_VO_DEV_NOT_CONFIG	The VO device is not configured.
HI_ERR_VO_NO_MEM	There is no available memory.
HI_ERR_VO_NOT_SUPPORT	The chip does not support the API.
HI_ERR_VO_DEV_NOT_BINDED	The PIP video layer is not bound.

[Difference]

Chip	Description
Hi3531	The video layer of DSD5 can be used as the PIP layer of an HD device.
Hi3532	The Hi3532 does not support this MPI.
Hi3521/Hi3520A/Hi3520D /Hi3515A/Hi3515C	The video layer of DSD0 can be used as the PIP layer of an HD device.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before enabling the PIP layer, ensure that the device where the PIP layer is located is enabled.
- Before enabling the PIP layer, ensure that the PIP layer is configured.
- Before enabling the PIP video layer, ensure that the video layer is bound.

[Example]

See the example of HI_MPI_VO_PipLayerBindDev.

[See Also]

HI_MPI_VO_DisablePipLayer

HI_MPI_VO_DisablePipLayer

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Binds the PIP layer to a specified device.

[Syntax]

```
HI_S32 HI_MPI_VO_DisablePipLayer(HI_VOID);
```

[Parameter]

None

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_SYS_NOTREADY	The system is not ready, that is, the device FD is not opened.
HI_ERR_VO_CHN_NOT_DISABLE	Some channels are not disabled at this video layer.
HI_ERR_VB_BUSY	The VB is not released.
HI_ERR_VO_NOT_SUPPORT	The chip does not support the API.
HI_ERR_VO_DEV_NOT_BINDED	The PIP video layer is not bound.

[Difference]

Chip	Description
Hi3531	The video layer of DSD5 can be used as the PIP layer of an HD device.
Hi3532	The Hi3532 does not support this MPI.
Hi3521/Hi3520A/Hi3520D/Hi3515A /Hi3515C	The video layer of DSD0 can be used as the PIP layer of an HD device.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.

[Requirement]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before disabling the PIP layer, ensure that all the channels at the PIP layer are disabled.
- Before disabling the PIP layer, ensure that the video layer is bound.
- If you do not release the VB that stores the pictures obtained from the VO channel when disabling the PIP layer, the error code HI_ERR_VB_BUSY is returned, indicating that the VB created by the VOU is not released. This error generally occurs when you obtain screen pictures but the VB is not released.

[Example]

See the example of HI_MPI_VO_PipLayerBindDev.

[See Also]

HI_MPI_VO_EnablePipLayer

HI_MPI_VO_EnableChn

[Description]

Enables a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_EnableChn(VO_DEV VoDev, VO_CHN VoChn);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Value Range of VoDev
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_SYS_NOTREADY	The system is not ready, that is, the device FD is not opened.
HI_ERR_VO_VIDEO_NOT_ENABLE	The video layer is not enabled.
HI_ERR_VO_CHN_NOT_ALLOC	No channel is allocated.
HI_ERR_VO_CHN_NOT_CONFIG	The VO channel is not configured.
HI_FAILURE	The display channel at the PIP layer is overlapped.
HI_ERR_VO_ILLEGAL_PARAM	The total size of the display channels at the PIP layer exceeds the limit.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before calling the MPI, you must enable a VO device. Otherwise, an error code indicating failure is returned.
- Before calling the MPI, you must enable the video layer of the corresponding device.
- Before enabling a channel, you must configure it. Otherwise, an error is returned, indicating that the channel is not configured.
- A VO channel can be enabled repeatedly, and no error code indicating failure is returned.

[Example]

```
VO_DEV VoDev = 0;
VO_CHN VoChn = 0;
VO_CHN_ATTR_S stChnAttr;
s32Ret = HI_MPI_VO_GetChnAttr(VoDev, VoChn, &stChnAttr);
if (s32Ret != HI_SUCCESS)
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
{
    printf("Get channel attr failed with errno %#x!\n", s32Ret);
    return HI_FAILURE;
}
stChnAttr.u32Priority = 0;
stChnAttr.stRect.s32X = 0;
stChnAttr.stRect.s32Y = 0;
stChnAttr.stRect.u32Width = 720;
stChnAttr.stRect.u32Height = 576;

s32Ret = HI_MPI_VO_SetChnAttr(VoDev, VoChn, &stChnAttr);
if (s32Ret != HI_SUCCESS)
{
    printf("Set channel attr failed with errno %#x!\n", s32Ret);
    return HI_FAILURE;
}
s32Ret = HI_MPI_VO_EnableChn(VoDev, VoChn);
if (s32Ret != HI_SUCCESS)
{
    printf("Enable channel failed with errno %#x!\n", s32Ret);
    return HI_FAILURE;
}
s32Ret = HI_MPI_VO_DisableChn(VoDev, VoChn);
if (s32Ret != HI_SUCCESS)
{
    printf("Disable channel failed with errno %#x!\n", s32Ret);
    return HI_FAILURE;
}
```

[See Also]

HI_MPI_VO_DisableChn

HI_MPI_VO_DisableChn

[Description]

Disables a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_DisableChn(VO_DEV VoDev,VO_CHN VoChn);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_VIDEO_NOT_ENABLE	The video layer is not enabled.
HI_ERR_VO_CHN_NOT_ALLOC	No channel is allocated.
HI_ERR_VO_BUSY	Timeout occurs when the VOU waits for the pictures sent by the VPSS.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- A VO channel can be disabled repeatedly, and no error code indicating failure is returned.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- If an HD device's channel is bound to a VPSS channel, you are advised to disable this channel by calling this MPI and then cancel the binding relationship between the VO channel and the VPSS channel. Otherwise, the HI_ERR_VO_BUSY error code is returned, indicating that VO waiting for the pictures sent by the VPSS timeouts.

[Example]

See the example of HI_MPI_VO_EnableChn.

[See Also]

HI_MPI_VO_EnableChn

HI_MPI_VO_SetChnAttr

[Description]

Sets the attributes of a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_SetChnAttr(VO_DEV VoDev,VO_CHN VoChn, VO_CHN_ATTR_S
*pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
pstAttr	Pointer to the attributes of a VO channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531	Value range of VoDev: [0, VO_MAX_DEV_NUM) Each HD device supports channel priorities 0 and 1, and each SD device supports channel priorities 0–VO_MAX_CHN_NUM – 1. The width or height of an SD device channel must be greater than or equal to 16 pixels.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Description
Hi3532	Value range of VoDev: 0 and [10, 13] The channel priority of the Hi3532 can be set only to 0. The width or height of an SD device channel must be greater than or equal to 16 pixels.
Hi3521/Hi3520A/Hi3520D /Hi3515A/Hi3515C	Value range of VoDev: [0, VO_MAX_DEV_NUM) Each HD device supports channel priorities 0 and 1, and each SD device supports channel priorities 0– VO_MAX_CHN_NUM – 1. The width or height of an SD device channel must be greater than or equal to 16 pixels.
Hi3518/Hi3516C	Value range of VoDev: [0, VO_MAX_DEV_NUM) SD device supports channel priorities 0– VO_MAX_CHN_NUM – 1. The width or height of an SD device channel must be greater than or equal to 32 pixels.

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_ILLEGAL_PARAM	The parameter is invalid.
HI_ERR_VO_NULL_PTR	The parameter pointer is null.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- A larger priority value indicates a higher priority.
  - For the HD device, the channel priorities are 0 and 1. Priority 0 indicates that the channel is at the original video layer of the device, and priority 1 indicates that the channel is at the PIP layer. The original video layer is always overlaid with the PIP layer. The channels of the video layer of an HD device or the channels of the PIP layer cannot overlap each other.
  - For the SD device, the channel priorities are 0 to (VO_MAX_CHN_NUM – 1). When the display regions of multiple channels overlap, the picture from the channel with a lower priority is overlaid with the picture from the channel with a higher priority. When the display regions of multiple channels with the same priority are

Copyright © HiSilicon Technologies Co., Ltd

overlaid, the pictures from the channel with small ID are overlaid with the pictures from the channel with large ID.

- The channel display region cannot exceed the display resolution stImageSize (one of video layer attributes).
- HI_MPI_VO_SetChnAttr is a dynamic MPI. That is, you can call it after a VO device is enabled and the corresponding video layer is configured.
- The channel attribute stRect of the HD device must be 2-pixel aligned, the width or height of an HD channel must be greater than or equal to 32 pixels. Note that the height must be 4-pixel aligned if the HD device's timing is interlaced and the display format is SPYCbCr420. The channel attribute stRect of the HD device must be 2-pixel aligned. The width and height cannot be greater than or equal to 32. In interlaced mode, the height of stRect must be 4-pixel aligned if the pixel format is SEMIPLANAR_420. Note that the vertical coordinate of the start point should be 4-pixel aligned. Otherwise, unexpected picture quality issues may occur. In addition, the width and height of an HD channel must be greater than or equal to 32 pixels.
- If the video layer of an HD device is zoomed in, the channel start position, width, and height included in the stRect attribute are the values before the video layer is zoomed in. After the video layer is zoomed in, the channel start position shifts and the channel width and height may increase by ratio.
- For the channel attribute stRect of the SD device, the channel width and channel height must be 2-pixel aligned.

[Example]

See the example of HI_MPI_VO_EnableChn.

[See Also]

HI_MPI_VO_GetChnAttr

HI_MPI_VO_GetChnAttr

[Description]

Obtains the attributes of a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_GetChnAttr(VO_DEV VoDev,VO_CHN VoChn,
VO_CHN_ATTR_S *pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
pstAttr	Pointer to the attributes of a VO channel	Output

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_NULL_PTR	The parameter pointer is null.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

None

[Example]

See the example of HI_MPI_VO_EnableChn.

[See Also]

HI_MPI_VO_SetChnAttr

HI_MPI_VO_SetChnDisplayWindow

[Description]

Sets the display window of a specified VO channel.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
HI_S32 HI_MPI_VO_SetChnDisplayWindow(VO_DEV VoDev, VO_CHN VoChn, const
RECT_S  *pstWindow);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
pstWindow	Pointer to the display window of a VO channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- This MPI is used to change only the size of the channel display window while the picture pattern remains unchanged during VO HD multi-picture display.
- This MPI is used to set the position of the channel display window only at the VHD or PIP layer.
- If the display position of the channel at the PIP layer is set by calling HI_MPI_VO_SetChnDispPos, the actual display position is the one set by calling HI_MPI_VO_SetChnDispPos.
- The position of the channel display window set by calling this MPI is valid only when the VPSS is bound to the VOU in VPSS_CHN_MODE_AUTO mode.
- You must call HI_MPI_VO_SetChnAttr to set the channel attributes before calling this MPI. After the channel attributes are reconfigured, the position of the display window is restored to the position configured in the channel attributes.
- X and Y in the display window attribute indicate the relative coordinates when the coordinates of the upper left corner in the current channel attribute serve as the coordinate origin. The configured display window size cannot exceed the region size configured in the channel attributes, and the alignment requirement configured in the display window attribute must be the same as that configured in the channel attributes.
- The position of the channel display window can be dynamically configured, and the positions of multiple channel display windows can be configured in batch.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[See Also]

HI_MPI_VO_GetChnDisplayWindow

HI_MPI_VO_GetChnDisplayWindow

[Description]

Obtains the display window of a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_GetChnDisplayWindow(VO_DEV VoDev, VO_CHN VoChn, RECT_S
*pstWindow);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
pstWindow	Pointer to the display window of a VO channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_VO_SetChnDisplayWindow

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VO_SetChnDispPos

[Description]

Sets the display position of a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_SetChnDispPos(VO_DEV VoDev, VO_CHN VoChn, const POINT_S
*pstDispPos);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
pstDispPos	Pointer to the channel display position	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531	The value of VoDev is 0 or 1.
Hi3532	The Hi3532 does not support this MPI.
Hi3521/Hi3520A/Hi3520D/Hi3515A/ Hi3515C	The value of VoDev is 0.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.

[Error Code]

Error Code	Description
HI_SUCCESS	Success
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_ILLEGAL_PARAM	The parameter is invalid.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Description
HI_ERR_VO_NULL_PTR	The parameter pointer is null.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_DEV_NOT_CONFIG	The attributes of a channel are not configured.
HI_ERR_VO_NOT_SUPPORT	The operation is not supported.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- This MPI is used to set the actual display position of only the channel at the PIP layer of the HD device.
  - The VO device displays the channel on the video layer canvas from the upper left corner of the video layer by default. When channel attributes are set, the channel position on the canvas is specified. This position is the default channel display position.
  - To reduce the PIP layer memory, set the canvas size of the PIP layer (stImageSize) to the memory size required for displaying pictures, and set the display area size (stDispRect) to the output resolution of the VO device. In this way, the PIP layer does not zoom in on the canvas to the entire display area. However, the common video layer of the HD device will zoom in on the canvas to the entire display area. The canvas size of the SD device or virtual device must be the same as the display area size.
  - After the channel display position is set by calling this MPI, the default position is replaced with the configured position, and the configured position is the actual channel display position on the VO device.
- The channel display position must be within the area defined in stDispRect.
- Before calling this MPI, you must set channel attributes.
- The horizontal and vertical coordinates (x, y) of the channel display position must be 2-pixel-aligned.
- The channel display position can be dynamically set, and the positions of multiple channels can be set in batch. You are advised to set the actual display positions of multiple channels in batch.
- If the horizontal and vertical coordinates (x, y) of the channel display position are set to (−1, −1), the default display position is restored. When you restore the default display position of a channel, ensure that the position does not overlap the display positions of other channels. You are advised to restore the default display positions of multiple channels in batch.
- If you call this MPI to set the display position of a channel to (x, y), the setting is retained when the channel is disabled and then enabled. To restore the default display position, manually set (x, y) to (−1, −1).

[Example]

```
VO_DEV VoDev = 0;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
VO_CHN VoChn = 0;
VO_CHN_ATTR_S stChnAttr;
VO_VIDEO_LAYER_ATTR_S stLayerAttr;
POINT_S stDispPos;
……
stLayerAttr->stDispRect.s32X = 0;
stLayerAttr->stDispRect.s32Y = 0;
stLayerAttr->stDispRect.u32Width = 1920;
stLayerAttr->stDispRect.u32Height = 1080;
stLayerAttr->stImageSize.u32Width = 720;
stLayerAttr->stImageSize.u32Height = 576;
stLayerAttr->u32DispFrmRt = 25;
stLayerAttr->enPixFormat = PIXEL_FORMAT_YUV_SEMIPLANAR_422;

s32Ret = HI_MPI_VO_PipLayerBindDev(VoDev);
if (s32Ret != HI_SUCCESS)
{
   printf("PPI Layer Bind Dev failed with errno %#x!\n", s32Ret);
   return HI_FAILURE;
}
s32Ret = HI_MPI_VO_SetPipLayerAttr(&stLayerAttr);
if (s32Ret != HI_SUCCESS)
{
   printf("Set Pip Layer Attr failed with errno %#x!\n", s32Ret);
   return HI_FAILURE;
}
s32Ret = HI_MPI_VO_EnablePipLayer();
if (s32Ret != HI_SUCCESS)
{
   printf("Enable Pip Layer failed with errno %#x!\n", s32Ret);
   return HI_FAILURE;
}

stChnAttr.u32Priority = 1;
stChnAttr.stRect.s32X = 0;
stChnAttr.stRect.s32Y = 0;
stChnAttr.stRect.u32Width = 720;
stChnAttr.stRect.u32Height = 576;
stChnAttr.bDeflicker = HI_FALSE;

s32Ret = HI_MPI_VO_SetChnAttr(VoDev, VoChn, &stChnAttr);
if (s32Ret != HI_SUCCESS)
{
   printf("Set channel attr failed with errno %#x!\n", s32Ret);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    return HI_FAILURE;
}

s32Ret = HI_MPI_VO_EnableChn(VoDev, VoChn);
if (s32Ret != HI_SUCCESS)
{
    printf("Enable channel failed with errno %#x!\n", s32Ret);
    return HI_FAILURE;
}

stDispPos.s32X = 500;
stDispPos.s32Y = 500;
s32Ret = HI_MPI_VO_SetChnDispPos (VoDev, VoChn, & stDispPos);
if (s32Ret != HI_SUCCESS)
{
    printf("Set channel display position failed with errno %#x!\n",
s32Ret);
    return HI_FAILURE;
}

s32Ret = HI_MPI_VO_DisableChn(VoDev, VoChn);
if (s32Ret != HI_SUCCESS)
{
    printf("Disable channel failed with errno %#x!\n", s32Ret);
    return HI_FAILURE;
}
```

[See Also]

HI_MPI_VO_GetChnDispPos

HI_MPI_VO_GetChnDispPos

[Description]

Obtains the display position of a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_GetChnDispPos(VO_DEV VoDev, VO_CHN VoChn, POINT_S
*pstDispPos);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID Value range: [0, VO_MAX_DEV_NUM)	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
pstDispPos	Pointer to the channel display position	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

Error Code	Description
HI_SUCCESS	Success
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_NULL_PTR	The parameter pointer is null.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_VO_SetChnDispPos

HI_MPI_VO_SetChnField

[Description]

Sets the frame/field display policy of a specified VO channel.

This MPI applies to the following application scenarios:

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- If you set the VI format to 2-field CIF when you preview CIF pictures, the display effect is smoother in 2-field VO display mode than in 1-field VO display mode.
- If you preview 2-field pictures at a low frame rate (such as D1 previewing at a low frame rate), you must set the display mode of the VO channel to 1-field mode. Otherwise, the screen flickers.
- If VI channels capture D1 and CIF pictures at the same time, you need to call this MPI to capture fields of D1 pictures. Otherwise, the screen flickers upwards and downwards.

[Syntax]

```
HI_S32 HI_MPI_VO_SetChnField(VO_DEV VoDev, VO_CHN VoChn, const
VO_DISPLAY_FIELD_E enField);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID Value range: [0, VO_MAX_DEV_NUM)	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
enField	Frame/Field display policy of a video channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

Error Code	Description
HI_SUCCESS	Success
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_ILLEGAL_PARAM	The parameter is invalid.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_NOT_SUPPORT	The operation is not supported.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

- If the MPI is not called explicitly, the VO channel displays two fields (VO_FIELD_BOTH) by default.
- If the picture that is sent to a VO channel is a frame picture such as a single-field CIF picture), calling the MPI has no effect on picture display.
- This MPI is valid only for SD and virtual devices.

[Example]

```
/*The following is an example using low frame rate during previewing. The
VI frame rate is set to 8 frame/s, and the display mode of corresponding
VO channel is set to bottom field.*/
VO_DISPLAY_FIELD_E enField;
VI_CHN_ATTR_S stViChnAttr;
…
stViChnAttr.s32SrcFrameRate = 25;
stViChnAttr.s32FrameRate = 8;
…
if (HI_SUCCESS!= HI_MPI_VI_SetChnAttr(0, & stViChnAttr))
{
    printf("HI_MPI_VI_SetChnAttr failed !\n");
    return HI_FAILURE;
}
if (HI_SUCCESS!=HI_MPI_VO_SetChnField(2, 0, VO_FIELD_BOTTOM))
{
    printf("HI_MPI_VO_SetChnField failed !\n");
    return HI_FAILURE;
}
if (HI_SUCCESS!=HI_MPI_VO_GetChnField(2, 0, &enField))
{
    printf("HI_MPI_VO_GetChnField failed !\n");
    return HI_FAILURE;
}
```

[See Also]

HI_MPI_VO_GetChnField

HI_MPI_VO_GetChnField

[Description]

Obtains the frame/field display policy of a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_GetChnField(VO_DEV VoDev, VO_CHN VoChn,
VO_DISPLAY_FIELD_E *pField);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID Value range: [0, VO_MAX_DEV_NUM)	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
pField	Pointer to the frame/field display attribute of a video channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

Error Code	Description
HI_SUCCESS	Success
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_NULL_PTR	The parameter pointer is null.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

If the display policy is not set, the default policy VO_FIELD_BOTH is returned.

[Example]

See the example of HI_MPI_VO_SetChnField.

[See Also]

HI_MPI_VO_SetChnField

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VO_SendFrame

[Description]

Sends video pictures to a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_SendFrame(VO_DEV VoDev, VO_CHN VoChn,
VIDEO_FRAME_INFO_S *pstVFrame);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
pstVFrame	Pointer to the information about video data	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/ Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

Before calling the MPI, you must enable a VO channel.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[See Also]

None

HI_MPI_VO_SendFrameTimeOut

[Description]

Sends pictures to a specified VO channel in timeout mode.

[Syntax]

```
HI_S32 HI_MPI_VO_SendFrame(VO_DEV, VO_CHN VoChn,
VIDEO_FRAME_INFO_S *pstVFrame, HI_U32 u32MilliSec);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
pstVFrame	Pointer to the information about video data	Input
u32MilliSec	Timeout period, in the unit of ms	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/ Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Requirement]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before calling the MPI, you must enable a channel.
- The MPI supports the block mode. The u32MilliSec parameter is used to set the timeout period. If u32MilliSec is set to 0, the MPI is blocked until pictures are sent successfully.

[Example]

None

[See Also]

None

HI_MPI_VO_SetChnFrameRate

[Description]

Sets the display frame rate of a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_SetChnFrameRate (VO_DEV VoDev,VO_CHN VoChn, HI_S32
s32VoFramerate);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
s32VoFramerate	Display frame rate of a video channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Value Range of VoDev
Hi3521/Hi3520A/Hi3520D/Hi3515A/ Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The device ID is invalid.
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_ILLEGAL_PARAM	The parameter is invalid.
HI_ERR_VO_CHN_NOT_CONFIG	The VO channel is not configured.
HI_ERR_VO_BUSY	The system is busy. It generally indicates that the system initialization function is not called before.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- This MPI can be called only after the related VO channel attributes are set. Otherwise, an error code indicating failure is returned.
- The display frame rate can be set to Nx. Where, *N* is an integer ranging from –64 to +64 and x is the frame rate in PAL mode or NTSC mode. The negative multiple is used for the rewind operation. In this case, you need to send the pictures to the VO channel in an inverted sequence, that is, send the pictures by descending timestamp.
- Calling HI_MPI_VO_EnableChn enables the channel to play streams at the normal speed. Therefore, you are advised to call HI_MPI_VO_SetChnFrameRate to control playback after the VO channel is enabled.
- If you call HI_MPI_VO_SetChnAttr when the VO channel is disabled, the channel frame rate is reset to the display frame rate.

[Example]

```
VO_DEV VoDev = 0;
VO_CHN VoChn = 0;
HI_S32 s32ChnFrmRate = 30;
s32Ret = HI_MPI_VO_SetChnFrameRate(VoDev, VoChn, s32ChnFrmRate);
if (s32Ret != HI_SUCCESS)
{
   printf("Set channel %d frame rate failed with errno %#x!\n", VoChn,
```


Copyright © HiSilicon Technologies Co., Ltd

```
s32Ret);
    return HI_FAILURE;
}
```

[See Also]

None

HI_MPI_VO_GetChnFrameRate

[Description]

Obtains the display frame rate of a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_GetChnFrameRate(VO_DEV VoDev, VO_CHN VoChn, HI_S32
*ps32VoFramerate);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
ps32VoFramerate	Display frame rate of a video channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A /Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_NULL_PTR	The parameter pointer is null.
HI_ERR_VO_BUSY	The system is busy. It generally indicates that the system initialization function is not called before.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

The obtained frame rate is the configured value. If no frame rate is configured, the full frame rate is returned.

[Example]

```
VO_DEV VoDev = 0;
VO_CHN VoChn = 0;
HI_S32 s32ChnFrmRate;

s32Ret = HI_MPI_VO_GetChnFrameRate(VoDev, VoChn, &s32ChnFrmRate);
if (s32Ret != HI_SUCCESS)
{
   printf("Set channel %d frame rate failed with errno %#x!\n", VoChn,
s32Ret);
   return HI_FAILURE;
}
printf("Channel %d frame rate is %d.\n", VoChn, s32ChnFrmRate);.
```

[See Also]

None

HI_MPI_VO_ChnPause

[Description]

Pauses a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_ChnPause(VO_DEV VoDev, VO_CHN VoChn);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- This MPI is not recommended during when the VIU connects to the VOU for preview.
- A channel can be paused repeatedly and no error code indicating failure is returned.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
HI_S32 s32ret;
VO_DEV VoDev = 0;
VO_CHN VoChn = 0;
...
/* enable vo chn */
s32ret = HI_MPI_VO_EnableChn(VoDev, VoChn);
if (HI_SUCCESS != s32ret)
{
printf("enable vo chn failed! \n");
return s32ret;
}

/* pause current vo channel */
s32ret = HI_MPI_VO_ChnPause(VoDev, VoChn);
if (HI_SUCCESS != s32ret)
{
printf("pause vo chn failed! \n");
return s32ret;
}

/* resume current vo channel */
s32ret = HI_MPI_VO_ChnResume(VoDev, VoChn);
if (HI_SUCCESS != s32ret)
{
    printf("resume vo chn failed! \n");
    return s32ret;
}

while (getchar() != 'q')
{
/* step forward current vo channel */
s32ret = HI_MPI_VO_ChnStep(VoDev, VoChn);
if (HI_SUCCESS != s32ret)
{
printf("step play vo chn failed! \n");
return s32ret;
}
}

/* resume current vo channel */
s32ret = HI_MPI_VO_ChnResume(VoDev, VoChn);
if (HI_SUCCESS != s32ret)
{
printf("resume vo chn failed! \n");
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
return s32ret;
}
(void)HI_MPI_VO_DisableChn(VoDev, VoChn);
```

[See Also]

HI_MPI_VO_ChnResume

HI_MPI_VO_ChnResume

[Description]

Resumes a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_ChnResume(VO_DEV VoDev, VO_CHN VoChn);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

A channel can be resumed repeatedly and no error code indicating failure is returned.

[Example]

See the example of HI_MPI_VO_ChnPause.

[See Also]

HI_MPI_VO_ChnPause

HI_MPI_VO_ChnStep

[Description]

Plays the streams in a specified VO channel by frame.

[Syntax]

```
HI_S32 HI_MPI_VO_ChnStep(VO_DEV VoDev, VO_CHN VoChn);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- To resume normal play, you can call HI_MPI_VO_ChnResume.
- This MPI is not recommended when the VIU connects to the VOU for preview.

[Example]

See the example of HI_MPI_VO_ChnPause.

[See Also]

HI_MPI_VO_ChnResume

HI_MPI_VO_ChnRefresh

[Description]

Refreshes a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_ChnRefresh(VO_DEV VoDev, VO_CHN VoChn);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
	Value range: [0, VO_MAX_CHN_NUM)	

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_NOT_SUPPORT	The operation is not supported.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- This MPI applies to electronic magnification during playback when the HD device is paused. Electronic magnification for the HD device is implemented by calling HI_MPI_VPSS_SetCropCfg. When the HD device is paused, the pictures clipped by the VPSS cannot be sent to the VOU. You need to refresh the VO channel by calling HI_MPI_VO_ChnRefresh.
- This MPI applies to the electronic magnification scenario even the frame rate for decoding is low.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- As the settings by calling HI_MPI_VO_ChnRefresh take effect immediately, you must call HI_MPI_VO_ChnRefresh after calling HI_MPI_VPSS_SetCropCfg. Otherwise, the pictures before clipping are obtained.
- This MPI does not take effect in normal play mode including forward and slow play.
- This MPI does not apply to the SD device and VI-VO previewing.

[Example]

```
HI_S32 s32ret;
VO_DEV VoDev = 0;
VO_CHN VoChn = 0;
...

/* enable vo chn */
s32ret = HI_MPI_VO_EnableChn(VoDev, VoChn);
if (HI_SUCCESS != s32ret)
{
    printf("enable vo chn failed! \n");
    return s32ret;
}

/* pause current vo channel */
s32ret = HI_MPI_VO_ChnPause(VoDev, VoChn);
if (HI_SUCCESS != s32ret)
{
    printf("pause vo chn failed! \n");
    return s32ret;
}

/* set vpss clip config */
...

/* refresh current vo channel */
s32ret = HI_MPI_VO_ChnRefresh(VoDev, VoChn);
if (HI_SUCCESS != s32ret)
{
    printf("refresh vo chn failed! \n");
    return s32ret;
}
(void)HI_MPI_VO_DisableChn(VoDev, VoChn);
```

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VO_EnableRecvFrameRateMatch

[Description]

Enables the frame rate match function of a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_EnableRecvFrameRateMatch (VO_DEV VoDev, VO_CHN VoChn);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- This MPI supports only HD channels.
- The following is a typical application scenario: During VI-VO cascading, the frame rate doubling function is enabled for the master and slave chips, the display frame rate is set to 30 fps to ensure smooth picture displaying, and the slave chip transfers data to the master chip at 60 fps. In this case, frames are discarded. This issue can be resolved by calling HI_MPI_VO_EnableRecvFrameRateMatch.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[See Also]

HI_MPI_VO_DisableRecvFrameRateMatch

HI_MPI_VO_DisableRecvFrameRateMatch

[Description]

Disables the frame rate match function of a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_DisableRecvFrameRateMatch (VO_DEV VoDev, VO_CHN VoChn);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

This MPI supports only HD channels.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[See Also]

HI_MPI_VO_EnableRecvFrameRateMatch

HI_MPI_VO_SetZoomInWindow

[Description]

Sets a VO zoom-in window.

[Syntax]

```
HI_S32 HI_MPI_VO_SetZoomInWindow(VO_DEV VoDev, VO_CHN VoChn,
const VO_ZOOM_ATTR_S *pstZoomAttr);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
pstZoomAttr	Structure of the partial zoom-in attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_INVALID_RECT_PARA	The parameter related to the rectangular structure is invalid.
HI_ERR_VO_VIDEO_NOT_CONFIG	The video layer is not configured.
HI_ERR_VO_NOT_SUPPORT	The chip does not allow you to set the HD channel priority to 1.
HI_ERR_VO_ILLEGAL_PARAM	The input parameter is invalid.
HI_ERR_VO_NULL_PTR	The input parameter pointer is null.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before calling the MPI, you must enable a VO device and set public VO attributes.
- Before calling the MPI, you must configure the video layer.
- This MPI has different meanings for the HD device and SD device.
- The HD device scales the entire video layer rather than the picture of a single channel. For example, if the size of a source VI picture is D1 and the size of the corresponding VO channel is CIF, the picture is scaled by the VPSS. For the HD device, this MPI is used to clip the picture of the VO channel and display the clipped part on the display region without scaling. The vacancy region is displayed in background color.

  The SD device can scale the picture of a single channel by using the TDE. For the SD device, this MPI is used to clip the picture of the VO channel and scale the clipped picture to fit into the channel size, implementing the partial zoom-in function.
- The source data of a VO channel can be clipped by rectangle or ratio. For details, see the description of VO_ZOOM_ATTR_S.
- For SD devices, if video inputs are D1 and 2CIF (two fields) pictures, you need to set pstZoomAttr based on the small picture. The SDK automatically processes the large picture, ensuring that the fields of vision of large and small pictures are consistent. If pstZoomAttr is set by ratio, ensure that the fields of vision of the positions that are obtained by converting the large and small pictures by ratio are consistent. Otherwise, the screen flickers leftwards and rightwards.
- In video cascade mode, the master chip can select regions from the picture obtained from the cascaded slave chip and combine the regions by using the partial zoom-in function.

[Example]

```
/*Define input parameter */
HI_S32 s32Ret;
VO_DEV VoDev = 0;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
VO_CHN VoChn = 0;
VO_ZOOM_ATTR_S
stZoomWindow, stZoomWindowGet;

/*
*You must enable VO and VO channel first!
* TODO: enable operation.
*/

stZoomAttr.enZoomType = VOU_ZOOM_IN_RECT;
stZoomWindow.stZoomRect.s32X = 128;
stZoomWindow.stZoomRect.s32Y = 128;
stZoomWindow.stZoomRect.u32Width = 200;
stZoomWindow.stZoomRect.u32Height = 200;

/*Set zoom window */
s32Ret = HI_MPI_VO_SetZoomInWindow(VoDev, VoChn, &stZoomWindow);
if (s32Ret != HI_SUCCESS)
{
printf("Set zoom attribute failed, ret = %#x.\n", s32Ret);
return HI_FAILURE;
}

/*Obtain the current zoom window parameter */
s32Ret = HI_MPI_VO_GetZoomInWindow(VoDev, VoChn, &stZoomWindowGet);
if (s32Ret != HI_SUCCESS)
{
   printf("Get zoom attribute failed, ret = %#x.\n", s32Ret);
   return HI_FAILURE;
}
else
{
   printf("Current zoom window is (%d,%d,%d,%d)!\n",
          stZoomWindowGet.stZoomRect.s32X,
          stZoomWindowGet.stZoomRect.s32Y,
          stZoomWindowGet.stZoomRect.u32Width,
          stZoomWindowGet.stZoomRect.u32Height);
}

/*
 * TODO: something else
 */
stZoomWindow.stZoomRect.s32X = 0;
stZoomWindow.stZoomRect.s32Y = 0;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
stZoomWindow.stZoomRect.u32Width = 0;
stZoomWindow.stZoomRect.u32Height = 0;

/*Cancel zoom window, we use (0,0,0,0) to recover */
s32Ret = HI_MPI_VO_SetZoomInWindow(VoDev, VoChn, &stZoomWindow);
if (s32Ret != HI_SUCCESS)
{
    printf("Recover zoom attribute failed, ret = %#x.\n", s32Ret);
    return HI_FAILURE;
}
```

[See Also]

HI_MPI_VO_GetZoomInWindow

HI_MPI_VO_GetZoomInWindow

[Description]

Obtains the parameters of a VO zoom-in window.

[Syntax]

```
HI_S32 HI_MPI_VO_GetZoomInWindow(VO_DEV VoDev, VO_CHN VoChn,
VO_WBC_ATTR_S *pstZoomAttr);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
pstZoomAttr	Pointer to partial zoom-in attributes	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Value Range of VoDev
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_NULL_PTR	The parameter pointer is null.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

If the zoom-in window is not set, the setting (0, 0, 0, 0) is obtained by default.

[Example]

See the example of HI_MPI_VO_SetZoomInWindow.

[See Also]

HI_MPI_VO_SetZoomInWindow

HI_MPI_VO_GetChnPts

[Description]

Obtains the time stamp of the current picture in a specified VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_GetChnPts(VO_DEV VoDev, VO_CHN VoChn, HI_U64
*pu64VoChnPts);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
	Value range: [0, VO_MAX_CHN_NUM)	
pu64VoChnPts	Pointer to the time stamp of the channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_NULL_PTR	The parameter pointer is null.
HI_ERR_VO_CHN_NOT_ENABLE	No VO channel is enabled.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before calling the MPI, ensure that the VO device and VO channel are enabled.
- When the decoding module is bound to the VOU by the system for transmitting video frames and the PTS of the decoded frame is 0, the VOU marks frames with PTSs starting

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

from 0 based on play interval. This ensures that every frame is played once. In other cases, the PTSs of original videos are not changed.

[Example]

```
VO_DEV VoDev = 0;
VO_CHN VoChn = 0;
HI_U64 u64ChnPts;

s32Ret = HI_MPI_VO_GetChnPts(VoDev, VoChn, &u64ChnPts);
if (s32Ret != HI_SUCCESS)
{
    printf("Get channel %d pts failed with errno %#x!\n", VoChn, s32Ret);
    return HI_FAILURE;
}
printf("Channel %d pts is %llu.\n", VoChn, u64ChnPts);
```

[See Also]

None

HI_MPI_VO_SetAttrBegin

[Description]

Starts attribute setting.

[Syntax]

```
HI_S32 HI_MPI_VO_SetAttrBegin(VO_DEV VoDev);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Value Range of VoDev
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The device ID is invalid.
HI_ERR_VO_SETBEGIN_ALREADY	The attribute of the VO device starts to be set.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- If HI_MPI_VO_SetAttrBegin is called, HI_MPI_VO_SetAttrEnd must be called. Otherwise, the channel attributes that are set by calling HI_MPI_VO_SetAttrBegin do not take effect.
- HI_MPI_VO_SetAttrBegin can be used to process dynamic operations on the channel in batches. The dynamic operations include setting the channel attribute, displaying a channel, hiding a channel, enabling a channel, disabling a channel, binding a cascade region to a channel, and unbinding a cascade region from a channel. When the picture display mode is switched or a VO channel is displayed or hidden, a buffer needs to be obtained again and the picture display mode needs to be switched in a timely manner. Therefore, you are advised to call HI_MPI_VO_SetAttrBegin, MPIs supporting the operations in batches (such as the MPIs for setting the channel attribute, displaying a channel, and hiding a channel), and HI_MPI_VO_SetAttrEnd in sequence.
- The batch channel operations take effect only after HI_MPI_VO_SetAttrEnd is called. You need to call appropriate MPIs between HI_MPI_VO_SetAttrBegin and HI_MPI_VO_SetAttrEnd. For example, if you want to disable the video layer while disabling channels by calling HI_MPI_VO_SetAttrBegin, HI_MPI_VO_DisableChn, HI_MPI_VO_DisableVideoLayer, and HI_MPI_VO_SetAttrEnd in sequence, the video layer fails to be disabled. This is because the video layer can be disabled only after channels are disabled, and channels can be disabled only after HI_MPI_VO_SetAttrEnd is called.

[Example]

```
HI_S32 s32ret;
HI_U32 i;
VO_DEV VoDev = 0;
VO_PUB_ATTR_S VoAttr;
VO_CHN_ATTR_S VoChnAttr[4];
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
memset(&VoAttr, 0, sizeof(VO_PUB_ATTR_S));
VoAttr.enIntfType = VO_INTF_CVBS;  /*CVBS */
VoAttr.enIntfSync = VO_OUTPUT_PAL ; /*PAL Mode*/
VoAttr.u32BgColor = 0x0;

/* set public attribute of vo device */
s32ret = HI_MPI_VO_SetPubAttr(VoDev, &VoAttr);
if (HI_SUCCESS != s32ret)
{
    printf("vo set pub attr failed! \n");
    return s32ret;
}

/* enable vo device */
s32ret = HI_MPI_VO_Enable(VoDev);
if (HI_SUCCESS != s32ret)
{
printf("enable vo device failed! \n");
return s32ret;
}

/* configure and enable vo channel                                    */
for (i = 0; i < 4; i++)
{
   VoChnAttr[i].bDeflicker = HI_FALSE;
   VoChnAttr[i].u32Priority = 1;
   VoChnAttr[i].stRect.s32X = 360*(i%2);
   VoChnAttr[i].stRect.s32Y = 288*(i/2);
   VoChnAttr[i].stRect.u32Width = 360;
   VoChnAttr[i].stRect.u32Height = 288;

   /* set attribute of vo chn*/
   s32ret = HI_MPI_VO_SetChnAttr(VoDev, i, &VoChnAttr[i]);
   if (HI_SUCCESS != s32ret)
   {
   printf("vo set dev %d chn %d attr failed! \n", VoDev , i);
   return s32ret;
   }

   /* enable vo chn */
   s32ret = HI_MPI_VO_EnableChn(VoDev, i);
   if (HI_SUCCESS != s32ret)
   {
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    printf("enable vo dev %d chn %d failed! \n", VoDev,i);
    return s32ret;
    }
}

/* do something else                                                    */
sleep(20);

/* set channel attributes begin                                         */
s32ret = HI_MPI_VO_SetAttrBegin(VoDev);
if (HI_SUCCESS != s32ret)
{
    printf("set attributes begin failed!\n");
    return s32ret;
}

/* reset channel attributes
 * we just show channel 0 and 1 on screen, and scale them as vertical
* half D1
 */
for (i = 0; i < 2; i++)
{
    VoChnAttr[i].bDeflicker = HI_FALSE;
    VoChnAttr[i].u32Priority = 1;
    VoChnAttr[i].stRect.s32X = 352*(i%2);
    VoChnAttr[i].stRect.s32Y = 288*(i/2);
    VoChnAttr[i].stRect.u32Width = 360;
    VoChnAttr[i].stRect.u32Height = 576;

    /* set attribute of vo chn*/
    s32ret = HI_MPI_VO_SetChnAttr(VoDev, i, &VoChnAttr[i]);
    if (HI_SUCCESS != s32ret)
    {
    printf("vo set chn %d attr failed! \n", i);
    return s32ret;
    }
}

/* hide channel 2 and 3
 * we do this just for saving bandwidth
 */
for (i = 2; i < 4; i++)
{
    s32ret = HI_MPI_VO_ChnHide(VoDev,i);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    if (s32ret != HI_SUCCESS)
    {
        printf("vo hide channel %d failed!\n", i);
        return HI_FAILURE;
    }
}

/* set channel attributes end                                          */
s32ret = HI_MPI_VO_SetAttrEnd(VoDev);
if (HI_SUCCESS != s32ret)
{
    printf("set attributes end failed!\n");
    return s32ret;
}

/* do something else                                                   */
sleep(20);

/* set channel attributes begin                                        */
s32ret = HI_MPI_VO_SetAttrBegin(VoDev);
if (HI_SUCCESS != s32ret)
{
    printf("set attributes begin failed!\n");
    return s32ret;
}

/* reset channel attributes
 * now we set them back as 4 cif on screen
 */
for (i = 0; i < 4; i++)
{
    VoChnAttr[i].bDeflicker = HI_FALSE;
    VoChnAttr[i].u32Priority = 1;
    VoChnAttr[i].stRect.s32X = 360*(i%2);
    VoChnAttr[i].stRect.s32Y = 288*(i/2);
    VoChnAttr[i].stRect.u32Width = 360;
    VoChnAttr[i].stRect.u32Height = 288;

    /* set attribute of vo chn*/
    s32ret = HI_MPI_VO_SetChnAttr(VoDev,i, &VoChnAttr[i]);
    if (HI_SUCCESS != s32ret)
    {
    printf("vo set chn %d attr failed! \n", i);
    return s32ret;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    }
}

/* show channel 2 and 3                                                  */
for (i = 2; i < 4; i++)
{
    s32ret = HI_MPI_VO_ChnShow(VoDev,i);
    if (s32ret != HI_SUCCESS)
    {
        printf("vo show channel %d failed!\n", i);
        return HI_FAILURE;
    }
}

/* set channel attributes end                                             */
s32ret = HI_MPI_VO_SetAttrEnd(VoDev);
if (HI_SUCCESS != s32ret)
{
    printf("set attributes end failed!\n");
    return s32ret;
}

/* do something else                                                      */
sleep(20);

/* disable all channels                                                  */
for (i = 0; i < 4; i++)
{
    s32ret = HI_MPI_VO_DisableChn(VoDev , i);
    if (HI_SUCCESS != s32ret)
    {
        printf("vo disable chn %d failed!\n", i);
        return s32ret;
    }
}

/* disable vou                                                           */
(void)HI_MPI_VO_Disable(VoDev);
```

[See Also]

HI_MPI_VO_SetAttrEnd

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VO_SetAttrEnd

[Description]

Ends attribute setting.

[Syntax]

```
HI_S32 HI_MPI_VO_SetAttrEnd(VO_DEV VoDev);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_SETBEGIN_NOTYET	Attribute setting of the VO device does not start.
HI_ERR_VO_SETEND_ALREADY	Attribute setting of the VO device is not complete.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before calling the MPI, you must enable the video layer where the VO channel is located.
- If the BEGIN MPI is called, the corresponding END MPI must be called. Otherwise, the channel attributes that are set by calling the BEGIN MPI do not take effect.

[Example]

See the example of HI_MPI_VO_SetAttrBegin.

[See Also]

HI_MPI_VO_SetAttrBegin

HI_MPI_VO_ChnShow

[Description]

Shows a specified channel.

[Syntax]

```
HI_S32 HI_MPI_VO_ChnShow(VO_DEV VoDev, VO_CHN VoChn);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Value Range of VoDev
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The device ID is invalid.
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_VIDEO_NOT_ENABLE	The VO video layer is not enabled.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before calling the MPI, you must enable a VO device.
- The channel is displayed by default.

[Example]

See the example of HI_MPI_VO_SetAttrBegin.

[See Also]

HI_MPI_VO_ChnHide

HI_MPI_VO_ChnHide

[Description]

Hides a specified channel.

[Syntax]

```
HI_S32 HI_MPI_VO_ChnHide(VO_DEV VoDev, VO_CHN VoChn);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The device ID is invalid.
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

When a VO channel is bound to a VPSS channel, you are advised to call this MPI to hide the VO channel, and then unbind the VO channel from the VPSS channel. Otherwise, the error code HI_ERR_VO_BUSY is returned.

[Example]

See the example of HI_MPI_VO_SetAttrBegin.

[See Also]

HI_MPI_VO_ChnShow

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VO_QueryChnStat

[Description]

Queries the status of a VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_QueryChnStat (VO_DEV VoDev, VO_CHN VoChn,
VO_QUERY_STATUS_S *pstStatus);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
pstStatus	Pointer to the structure of the channel status	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_CHN_NOT_ENABLE	No VO channel is enabled.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Definition
HI_ERR_VO_INVALID_DEVID	The VO device ID is invalid.
HI_ERR_VO_CHN_NOT_ALLOC	No channel is allocated.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before calling the MPI, you must enable a VO device.
- Before calling the MPI, you must enable the VO channel to be queried.
- You can repeatedly call the MPI for obtaining the channel status.

[Example]

```
HI_S32 s32Ret;
VO_DEV VoDev = 0;
VO_CHN VoChn = 0;
VO_QUERY_STATUS_S stVoQueryStatus;

/* enable vo device and vo channel */
…
s32Ret = HI_MPI_VO_QueryChnStat(VoDev, VoChn, &stVoQueryStatus);
if (HI_SUCCESS != s32Ret)
{
printf("Query channel status failed with errorcode %#x!\n", s32Ret);
return HI_FAILURE;
}
printf("Current vo dev %d channel %d has %d VB occupied!\n",
VoDev, VoChn, stVoQueryStatus .u32ChnBufUsed );
```

[See Also]

None

HI_MPI_VO_GetChnFrame

[Description]

Obtains the pictures of a VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_GetChnFrame(VO_DEV VoDev, VO_CHN VoChn,
VIDEO_FRAME_INFO_S *pstFrame);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
pstFrame	Pointer to the obtained information about the pictures of a VO channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The device ID is invalid.
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_CHN_NOT_ENABLE	No VO channel is enabled.
HI_ERR_VO_NULL_PTR	The VO parameter pointer is null.
HI_ERR_VO_SYS_NOTREADY	There is no picture in the channel or the channel is hidden.
HI_ERR_VO_NOT_SUPPORT	The pictures of a VO channel cannot be obtained repeatedly.

[Requirement]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before calling the MPI, you must enable a VO device.
- Before calling the MPI, you must enable a video layer.
- The pictures of a VO channel cannot be obtained repeatedly.
- You must release the buffer for storing the pictures in time.

[Example]

```
...
VIDEO_FRAME_INFO_S *pstFrame =
(VIDEO_FRAME_INFO_S*)malloc(sizeof(VIDEO_FRAME_INFO_S));

if (HI_SUCCESS != HI_MPI_VO_GetScreenFrame(VoDev, pstFrame))
{
   printf("Get screen frame failed!\n");
   return HI_FAILURE;
}
/* Perform processing, for example, storing files or performing JPEG
encoding*/

if (HI_SUCCESS != HI_MPI_VO_ReleaseScreenFrame(VoDev, pstFrame))
{
   printf("Release screen frame failed!\n");
   return HI_FAILURE;
}
/* Enable vo chn */
s32ret = HI_MPI_VO_EnableChn(VoDev, VoChn);
if (HI_SUCCESS != s32ret)
{
    printf("enable vo chn failed! \n");
    return s32ret;
}

if (HI_SUCCESS != HI_MPI_VO_GetChnFrame(VoDev, VoChn, pstFrame))
{
   printf("Get channel frame failed!\n");
   return HI_FAILURE;
}

/* Perform processing, for example, storing files or performing JPEG
encoding.*/
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
if (HI_SUCCESS != HI_MPI_VO_ReleaseChnFrame(VoDev, VoChn, pstFrame))
{
    printf("Release screen frame failed!\n");
    return HI_FAILURE;
}
(void)HI_MPI_VO_DisableChn(VoDev,VoChn);
```

[See Also]

HI_MPI_VO_ReleaseChnFrame

HI_MPI_VO_ReleaseChnFrame

[Description]

Releases the buffer for storing the pictures of a VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_ReleaseChnFrame(VO_DEV VoDev, VO_CHN
VoChn, VIDEO_FRAME_INFO_S *pstFrame);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
pstFrame	Pointer to the information about the released pictures of a VO channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Value Range of VoDev
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VO_INVALID_DEVID	The device ID is invalid.
HI_ERR_VO_INVALID_CHNID	The VO channel ID is invalid.
HI_ERR_VO_NOT_PERMIT	The operation is forbidden.
HI_ERR_VO_NULL_PTR	The input parameter pointer is null.
HI_ERR_VO_ILLEGAL_PARAM	The parameter is invalid.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- After calling HI_MPI_VO_GetChnFrame, you must call HI_MPI_VO_ReleaseChnFrame.
- This MPI cannot be called repeatedly.

[Example]

See the example of HI_MPI_VO_GetChnFrame.

[See Also]

HI_MPI_VO_GetChnFrame

HI_MPI_VO_ClearChnBuffer

[Description]

Clears the data in a specified VO channel buffer.

[Syntax]

```
HI_S32 HI_MPI_VO_ClearChnBuffer(VO_DEV VoDev, VO_CHN VoChn,
HI_BOOL bClrAll)
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
bClrAll	Whether to clear all the data in the channel buffer Value: • HI_TRUE: clears all the data in the channel buffer. In this case, no picture is displayed in the channel area corresponding to this channel until new pictures are received. • HI_FALSE: retains a picture in the channel buffer and clears other data.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

You can call this MPI to clear a VO channel buffer.

[Example]

```
VO_DEV VoDev = 0;
VO_CHN VoChn = 0;
HI_BOOL bClearAll = HI_TRUE;

s32Ret = HI_MPI_VO_ClearChnBuffer (VoDev, VoChn, & bClearAll);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
if (s32Ret != HI_SUCCESS)
{
    printf("Clear channel %d buf failed with errno %#x!\n", VoChn, s32Ret);
    return HI_FAILURE;
}
```

[See Also]

None

HI_MPI_VO_SetChnDispThreshold

[Description]

Sets the display threshold for a VO channel.

In PCIV cascade mode, when the slave chip transfers pictures to the master chip for previewing or displaying, the number of pictures to be displayed in the VO channel may be greater than the default threshold because the transfer interval is uneven. As a result, frames are lost or intermittence occurs during previewing or playback. In this case, this MPI is used to set the threshold to a larger value.

[Syntax]

```
HI_S32 HI_MPI_VO_SetChnDispThreshold(VO_DEV VoDev, VO_CHN VoChn, HI_U32
u32Threshold);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
u32Threshold	Display threshold for a VO channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Range of the VO Device ID
Hi3531	[0, VO_MAX_DEV_NUM)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Range of the VO Device ID
Hi3532	[0] and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- The display threshold range of the HD device is [3, 8]. The default threshold 3 indicates that there are three frames to be displayed in the VO channel. The display threshold range of the SD device is [2, 8]. The default threshold 2 indicates that the channel buffer queue can receive at most two frames. Because there is a displaying frame or a frame to be displayed, three frames are to be displayed in total.
- A smaller threshold indicates a smaller preview delay, and vice versa. Therefore, a larger threshold is not recommended typically.
- The threshold can be dynamically adjusted. It is recommended that the threshold is set before the VO channel is enabled. You can call this MPI to restore the threshold to the default value.

[Example]

None

[See Also]

HI_MPI_VO_GetChnDispThreshold

HI_MPI_VO_GetChnDispThreshold

[Description]

Obtains the display threshold for a VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_GetChnDispThreshold(VO_DEV VoDev, VO_CHN VoChn, HI_U32
*pu32Threshold);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input
u32Threshold	Display threshold for a VO channel	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Range of the VO Device ID
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	[0] and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

If you call HI_MPI_VO_GetChnDispThreshold before calling HI_MPI_VO_SetChnDispThreshold, the default threshold is obtained. If you call HI_MPI_VO_GetChnDispThreshold after calling HI_MPI_VO_SetChnDispThreshold, the configured threshold is obtained.

[Example]

None

[See Also]

HI_MPI_VO_SetChnDispThreshold

HI_MPI_VO_EnableChnDoubleFrame

[Description]

Enables the double-frame-rate function of a channel.

[Syntax]

```
HI_S32 HI_MPI_VO_EnableChnDoubleFrame(VO_DEV VoDev, VO_CHN VoChn);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- This interface is an optional advanced interface used to enable/disable the double-frame-rate function of VO channels dynamically. If bDoubleFrame in VO_PUB_ATTR_S is not enabled, this interface is invalid. That is, even after you call this interface to enable the double-frame-rate function, the frame rate is not doubled.
- Only HD and cascade devices support the double-frame-rate function.
- After the HI_MPI_VO_EnableChn interface is called, the program enables or disables the double-frame-rate function of the channel based on the value of bDoubleFrame in VO_PUB_ATTR_S and the ID of the current device. For example, if bDoubleFrame is set to TRUE and the device is a HD or cascade device, the double-frame-rate function is enabled by default. Otherwise, this function is disabled.

[Example]

None

[See Also]

HI_MPI_VO_DisableChnDoubleFrame

HI_MPI_VO_DisableChnDoubleFrame

[Description]

Disables the double-frame-rate function of a channel.

[Syntax]

```
HI_S32 HI_MPI_VO_DisableChnDoubleFrame(VO_DEV VoDev, VO_CHN VoChn);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VoDev	VO device ID	Input
VoChn	VO channel ID Value range: [0, VO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

Refer to HI_MPI_VO_EnableChnDoubleFrame.

[Example]

None

[See Also]

HI_MPI_VO_EnableChnDoubleFrame

HI_MPI_VO_GetScreenFrame

[Description]

Obtains the pictures displayed on the screen.

[Syntax]

```
HI_S32 HI_MPI_VO_GetScreenFrame(VO_DEV VoDev, VIDEO_FRAME_INFO_S
*pstFrame);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pstFrame	Pointer to the obtained information about the pictures displayed on the screen	Output

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Range of the VO Device ID
Hi3531	[2, 7] and [10, VO_MAX_DEV_NUM)
Hi3532	[10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[1, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before calling the MPI, you must enable a VO device.
- The MPI can be repeatedly called. After calling HI_MPI_VO_GetScreenFrame, you must call HI_MPI_VO_ReleaseScreenFrame in time.
- If the buffer is not released after screen pictures are obtained, an error code indicating busy is returned when you disable the video layer. In this case, you can successfully disable the video layer only after you release the buffer for storing obtained pictures.
- The pictures obtained by calling HI_MPI_VO_GetScreenFrame contain time stamps. The time stamps show the time for combining captured pictures. You can modify the time stamps for encoding.

[Example]

See the example of HI_MPI_VO_GetChnFrame.

[See Also]

HI_MPI_VO_ReleaseScreenFrame

HI_MPI_VO_ReleaseScreenFrame

[Description]

Releases the buffer for storing the pictures displayed on the screen.

[Syntax]

```
HI_S32 HI_MPI_VO_ReleaseScreenFrame(VO_DEV VoDev, VIDEO_FRAME_INFO_S
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
*pstFrame);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pstFrame	Pointer to the information about the released pictures displayed on the screen	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Range of the VO Device ID
Hi3531	[2, 7] and [10, VO_MAX_DEV_NUM)
Hi3532	[10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[1, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before disabling the video layer, you must release the buffer for storing the obtained screen pictures.
- The MPI can be repeatedly called. After calling HI_MPI_VO_GetScreenFrame, you must call HI_MPI_VO_ReleaseScreenFrame.

[Example]

See the example of HI_MPI_VO_GetChnFrame.

[See Also]

HI_MPI_VO_GetScreenFrame

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VO_SetDevCSC

[Description]

Sets the effect of the output pictures from a device.

[Syntax]

```
HI_S32 HI_MPI_VO_SetDevCSC(VO_DEV VoDev, VO_CSC_S *pstDevCSC);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pstDevCSC	Pointer to the effect of output pictures	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531	• The value range of VoDev is [0, 7]. • The value range of u32Luma, u32Contrast, u32Hue, or u32Satuature is [0, 100], and the default value is 50. • The default value of enCscMatrix is identity matrix.
Hi3532	• The value of VoDev is 0. • The value range of u32Luma, u32Contrast, u32Hue, or u32Satuature is [0, 100], and the default value is 50. • The default value of enCscMatrix is identity matrix.
Hi3521/Hi3520A	• The value range of VoDev is [0, 2]. • The value range of u32Luma, u32Contrast, u32Hue, or u32Satuature is [0, 100], and the default value is 50. • The default value of enCscMatrix is identity matrix.
Hi3520D/Hi3515A/ Hi3515C	• The value range of VoDev is [0, 2]. • The value range of u32Luma, u32Contrast, u32Hue, or u32Satuature is [0, 100]. Note that the default values of u32Luma, u32Contrast, u32Hue, and u32Satuature are 50 for SD devices, and the default values of u32Luma, u32Contrast, and u32Satuature are 50, 50, 50, and 59 respectively for HD devices.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Description
	• The default value of enCscMatrix is identity matrix for SD devices or VO_CSC_MATRIX_BT601_TO_BT709 for HD devices.
Hi3518/Hi3516C	• The value range of VoDev is [0, VO_MAX_DEV_NUM). • The value range of u32Luma, u32Contrast, u32Hue, or u32Satuature is [0, 100], and the default value is 50. • The default value of enCscMatrix is identity matrix.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- This MPI is used to adjust the effect of output pictures, including the luminance, contrast, hue, and saturation. The effect value ranges from 0 to 100.
- The color space conversion (CSC) matrix must be VO_CSC_MATRIX_IDENTITY, VO_CSC_MATRIX_BT601_TO_BT709, or VO_CSC_MATRIX_BT709_TO_BT601.
- Never select VO_CSC_MATRIX_BT709_TO_BT601 for VGA outputs. Otherwise, the picture quality is abnormal.
- If the video layer is disabled, the luminance, contrast, hue, and saturation are restored to the default value, and the CSC matrix is also restored to the default value.

[Example]

None

[See Also]

HI_MPI_VO_GetDevCSC

HI_MPI_VO_GetDevCSC

[Description]

Obtains the effect of the output pictures from a device.

[Syntax]

```
HI_S32 HI_MPI_VO_GetDevCSC(VO_DEV VoDev, VO_CSC_S *pstDevCSC);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pstDevCSC	Pointer to the effect of output pictures	Output

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Range of the VO Device ID
Hi3531	[0, 7]
Hi3532	0
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, 2]
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

This MPI is used to obtain the effect of output pictures, including the luminance, contrast, hue, and saturation. The effect value ranges from 0 to 100.

[Example]

None

[See Also]

HI_MPI_VO_SetDevCSC

HI_MPI_VO_SetVgaParam

[Description]

Sets the effect of the VGA output pictures from a device.

[Syntax]

```
HI_S32 HI_MPI_VO_SetVgaParam (VO_DEV VoDev, VO_VGA_PARAM_S *pstVgaParam);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pstVgaParam	Pointer to the effect of VGA output pictures	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531	• The value range of VoDev is [0, 7]. • The value range of u32Luma, u32Contrast, u32Hue, or u32Satuature is [0, 100], and the default value is 50.
Hi3532	• The value of VoDev is 0. • The value range of u32Luma, u32Contrast, u32Hue, or u32Satuature is [0, 100], and the default value is 50.
Hi3521/Hi3520A	• The value range of VoDev is [0, 2]. • The value range of u32Luma, u32Contrast, u32Hue, or u32Satuature is [0, 100], the default value of u32Satuature is 59, and the default value of u32Luma, u32Contrast, or u32Hue is 50.
Hi3518/Hi3516C/Hi3520D/Hi3515A /Hi3515C	The VoDev, u32Luma, u32Contrast, u32Hue, and u32Satuature parameters are not supported.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- This MPI is used to adjust the effect of VGA output pictures, including the luminance, contrast, hue, saturation, and DAC gain. The value range of the luminance, contrast, hue, or saturation is [0, 100], and the value range of the DAC gain is [0, 63].
- The CSC matrix must be VO_CSC_MATRIX_BT601_TO_RGB_PC or VO_CSC_MATRIX_BT709_TO_RGB_PC.
- When the device is disabled, the luminance, contrast, hue, and saturation are restored to the default value, the DAC gain is restored to the default value 10, and the CSC matrix is restored to VO_CSC_MATRIX_BT601_TO_RGB_PC.
- When the luminance, contrast, hue, and saturation are adjusted by calling HI_MPI_VO_SetDevCsc, the output pictures from all interfaces of the device are affected. When the luminance, contrast, hue, and saturation are adjusted by calling HI_MPI_VO_SetVgaParam, only the output pictures from the VGA interface of the

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

device are affected. If both HI_MPI_VO_SetDevCsc and HI_MPI_VO_SetVgaParam are called, VGA output pictures are affected by all settings.

[Example]

None

[See Also]

HI_MPI_VO_GetDevCSC

HI_MPI_VO_GetVgaParam

[Description]

Obtains the effect of the VGA output pictures from a device.

[Syntax]

```
HI_S32 HI_MPI_VO_GetVgaParam(VO_DEV VoDev, VO_VGA_PARAM_S *pstVgaParam);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pstVgaParam	Pointer to the effect of VGA output pictures	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Range of the VO Device ID
Hi3531	[0, 7]
Hi3532	0
Hi3521/Hi3520A	[0, 2]
Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C	not supported

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

None

[Example]

None

[See Also]

HI_MPI_VO_SetDevCSC

HI_MPI_VO_SetDispBufLen

[Description]

Sets the size of a display buffer.

[Syntax]

```
HI_S32 HI_MPI_VO_SetDispBufLen(VO_DEV VoDev, HI_U32 u32BufLen);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
u32BufLen	Display buffer size Value range: [3, 15]	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Requirement]

Copyright © HiSilicon Technologies Co., Ltd

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, ensure that the VO devices are not enabled.
- The default buffer size is 3 for SD devices or virtual devices and 6 for HD devices and cascade display devices. The default buffer size is 3 for SD devices or virtual devices. The default buffer size is 6 for HD devices and cascade display devices of the Hi3531 and Hi3532, the default value is 4 for HD devices of Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C/Hi3518/Hi3516C.
- The minimum buffer size is 3 for SD devices, virtual devices, HD devices, or cascade display devices.

[Example]

```
HI_S32 s32Ret;
VO_DEV VoDev = 10;
VO_PUB_ATTR_S stPubAttr;
HI_U32 u32DispBufLen = 10;

s32Ret = HI_MPI_VO_SetDispBufLen(VoDev, u32DispBufLen);
if (s32Ret != HI_SUCCESS)
{
   printf("Set display buf len failed with error code %#x!\n", s32Ret);
   return HI_FAILURE;
}

s32Ret = HI_MPI_VO_GetPubAttr(VoDev, &stPubAttr);
if (s32Ret != HI_SUCCESS)
{
   printf("Get device attributes failed with error code %#x!\n", s32Ret);
   return HI_FAILURE;
}

stPubAttr.u32BgColor = 0xff;
stPubAttr.enIntfType = VO_INTF_VGA;
stPubAttr.enIntfSync = VO_OUTPUT_1280x1024_60;

s32Ret = HI_MPI_VO_SetPubAttr(VoDev, &stPubAttr);
if (s32Ret != HI_SUCCESS)
{
   printf("Set device attributes failed with errno %#x!\n", s32Ret);
   return HI_FAILURE;
}

s32Ret = HI_MPI_VO_Enable(VoDev);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
if (s32Ret != HI_SUCCESS)
{
    printf("Enable vo dev %d failed with errno %#x!\n", VoDev, s32Ret);
    return HI_FAILURE;
}
```

[See Also]

None

HI_MPI_VO_GetDispBufLen

[Description]

Obtains the size of a display buffer.

[Syntax]

```
HI_S32 HI_MPI_VO_GetDispBufLen(VO_DEV VoDev, HI_U32 *pu32BufLen);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pu32BufLen	u32 pointer	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Library file: libmpi.a

[Note]

None

[Example]

```
HI_S32 s32Ret;
VO_DEV VoDev = 10;
VO_PUB_ATTR_S stPubAttr;
HI_U32 u32DispBufLen;
s32Ret = HI_MPI_VO_GetDispBufLen(VoDev, &u32DispBufLen);
if (s32Ret != HI_SUCCESS)
{
   printf("Get display buf len failed with error code %#x!\n", s32Ret);
   return HI_FAILURE;
}

printf("Get dev %d disp buf len is %d.\n", VoDev, u32DispBufLen);
```

[See Also]

None

HI_MPI_VO_SetPlayToleration

[Description]

Sets the play tolerance.

[Syntax]

```
HI_S32 HI_MPI_VO_SetPlayToleration(VO_DEV VoDev, HI_U32 u32Toleration);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
u32Toleration	Play tolerance Value range: [1, 100000]	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Difference]

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before calling the MPI, you must enable a VO device.
- The play tolerance is in the unit of millisecond, and the default value is 10000.
- If the absolute value of the time tolerance of two frames of decoded data is greater than the play tolerance, the system will reset the time stamp and control the frame rate for data decoding based on the time stamp of the current frame. The play tolerance is valid for play control during picture decoding rather than picture previewing.

[Example]

```
HI_S32 s32Ret;
VO_DEV VoDev = 1;
VO_PUB_ATTR_S stPubAttr;
HI_U32 u32Toleration = 5000;

s32Ret = HI_MPI_VO_GetPubAttr(VoDev, &stPubAttr);
if (s32Ret != HI_SUCCESS)
{
   printf("Get device attributes failed with error code %#x!\n", s32Ret);
   return HI_FAILURE;
}

stPubAttr.u32BgColor = 0xff;
stPubAttr.enIntfType = VO_INTF_VGA;
stPubAttr.enIntfSync = VO_OUTPUT_1280x1024_60;

s32Ret = HI_MPI_VO_SetPubAttr(VoDev, &stPubAttr);
if (s32Ret != HI_SUCCESS)
{
   printf("Set device attributes failed with errno %#x!\n", s32Ret);
   return HI_FAILURE;
}
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
s32Ret = HI_MPI_VO_Enable(VoDev);
if (s32Ret != HI_SUCCESS)
{
    printf("Enable vo dev %d failed with errno %#x!\n", VoDev, s32Ret);
    return HI_FAILURE;
}

s32Ret = HI_MPI_VO_ SetPlayToleration (VoDev, u32Toleration);
if (s32Ret != HI_SUCCESS)
{
    printf("Set play toleration failed with error code %#x!\n", s32Ret);
    return HI_FAILURE;
}
```

[See Also]

None

HI_MPI_VO_GetPlayToleration

[Description]

Obtains the play tolerance.

[Syntax]

```
HI_S32 HI_MPI_VO_GetPlayToleration(VO_DEV VoDev, HI_U32 *pu32Toleration);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pu32Toleration	u32 pointer	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Value Range of VoDev
Hi3531	[0, VO_MAX_DEV_NUM)
Hi3532	0 and [10, 13]
Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C	[0, VO_MAX_DEV_NUM)
Hi3518/Hi3516C	[0, VO_MAX_DEV_NUM)

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

Before calling the MPI, you must enable a VO device.

[Example]

```
HI_S32 s32Ret;
VO_DEV VoDev = 0;
HI_U32 u32Toleration;

s32Ret = HI_MPI_VO_ GetPlayToleration (VoDev, & u32Toleration);
if (s32Ret != HI_SUCCESS)
{
    printf("Get play toleration failed with error code %#x!\n", s32Ret);
    return HI_FAILURE;
}
printf("Get dev %d play toleration is %d.\n", VoDev, u32Toleration);
```

[See Also]

None

HI_MPI_VO_SetVtth

[Description]

Sets the vertical time sequence interruption threshold of a device.

[Syntax]

```
HI_S32 HI_MPI_VO_SetVtth(VO_DEV VoDev, HI_U32 u32Vtth);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
u32Vtth	Threshold, indicating that vertical time sequence	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
	interruption is reported at the last but *X* valid pixel line (*X* = value of this parameter − 1) Value range: [240, 8191] for HD devices; [100, 8191] for SD devices	

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Generally, when interruption processing response is delayed due to large service volume in the scenario where multiple pieces of videos are cascaded, this interface can be used to set the threshold to a larger value to make vertical time sequence interruption be reported in advance. It is not recommended that this interface be used in other scenarios.
- Ensure that the device is disabled before calling this interface.
- Only physical devices support setting the threshold.
- The default vertical time sequence interruption thresholds of HD and SD devices are respectively 240 and 100. Ensure that the threshold set is not larger than the number of valid pixel lines of the current time sequence.

[Example]

None

[See Also]

HI_MPI_VO_GetVtth

HI_MPI_VO_GetVtth

[Description]

Obtains the vertical time sequence interruption threshold of a device.

[Syntax]

```
HI_S32 HI_MPI_VO_GetVtth(VO_DEV VoDev, HI_U32 *pu32Vtth);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pu32Vtth	Pointer of u32 type	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

Only physical devices support obtaining the threshold.

[Example]

None

[See Also]

HI_MPI_VO_SetVtth

HI_MPI_VO_EnableWbc

[Description]

Enables WBC.

[Syntax]

```
HI_S32 HI_MPI_VO_EnableWbc(VO_DEV VoDev);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input

[Return Value]

Return Value	Description
0	Success

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531	DHD1 supports WBC. The value of VoDeV is 1.
Hi3532/ Hi3518/Hi3516C	The Hi3532/Hi3518/Hi3516C does not support WBC.
Hi3521/Hi3520A/Hi3520D/Hi3515A/ Hi3515C	DHD0 supports WBC. The value of VoDeV is 0.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, you must enable DHDx.
- Before calling this MPI, you must enable DHDx.
- WBC can be enabled only after the WBC attribute is set.
- If this MPI is repeatedly called, a code indicating success is returned.
- If the WBC function is enabled immediately after DHDx is enabled, DHDx displays the background color and the WBC module writes back the background color, because DHDx may not receive pictures from the VIU or other modules. In this case, if the WBC mode is VO_WBC_MODE_PROG_TO_INTL, a frame may be the background color picture, and the other frame may be the data picture. After the two frames are written back and combined, a field of the combined picture is the background color, and the other field is the picture data. Therefore, you are advised to enable the WBC function after DHDx displays the picture data.

[Example]

```
……
VO_WBC_ATTR_S stWbcAttr;
VO_WBC_MODE_E enWbcMode;
stWbcAttr.stTargetSize.u32Width = 720;
stWbcAttr.stTargetSize.u32Height = 576;
stWbcAttr.enPixelFormat = PIXEL_FORMAT_YUV_SEMIPLANAR_422;
stWbcAttr.u32FrameRate = 25;
stWbcAttr.enDataSource = VO_WBC_DATASOURCE_MIXER;
}
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
if (HI_SUCCESS != HI_MPI_VO_SetWbcAttr(VoDev,&stWbcAttr))
{
    printf("Set wbc attr failed!\n");
    return HI_FAILURE;
}
if (HI_SUCCESS != HI_MPI_VO_GetWbcMode(VoDev, VO_WBC_MODE_NOMAL))
{
    printf("Get wbc mode failed!\n");
    return HI_FAILURE;
}
printf("Get wbc mode %d\n", enWbcMode);
enWbcMode = VO_WBC_MODE_NOMAL;

if (HI_SUCCESS != HI_MPI_VO_SetWbcMode(VoDev, enWbcMode))
{
    printf("Set wbc mode failed!\n");
    return HI_FAILURE;
}

/* enable wbc */
if (HI_SUCCESS != HI_MPI_VO_EnableWbc(VoDev))
{
    printf("Enable wbc failed!\n");
    return HI_FAILURE;
}
if (HI_SUCCESS != HI_MPI_VO_GetWbcAttr(VoDev,&stWbcAttr))
{
    printf("Get wbc attr failed!\n");
    return HI_FAILURE;
}

/* do some process*/

/* disable wbc */
if (HI_SUCCESS != HI_MPI_VO_DisableWbc (VoDev))
{
    printf("Disable wbc failed!\n");
    return HI_FAILURE;
}
```

[See Also]

HI_MPI_VO_DisableWbc

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VO_DisableWbc

[Description]

Disables WBC.

[Syntax]

```
HI_S32 HI_MPI_VO_DisableWbc(VO_DEV VoDev);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531	DHD1 supports WBC. The value of VoDeV is 1.
Hi3532/ Hi3518/Hi3516C	The Hi3532/Hi3518/Hi3516C does not support WBC.
Hi3521/Hi3520A/Hi3520D/ Hi3515A/Hi3515C	DHD0 supports WBC. The value of VoDeV is 0.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

If this MPI is repeatedly called, a code indicating success is returned.

[Example]

See the example of HI_MPI_VO_EnableWbc.

[See Also]

HI_MPI_VO_EnableWbc

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VO_SetWbcAttr

[Description]

Sets the WBC attribute.

[Syntax]

```
HI_S32 HI_MPI_VO_SetWbcAttr(VO_DEV VoDev, const VO_WBC_ATTR_S
*pstWbcAttr);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pstWbcAttr	Pointer to the WBC attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531	DHD1 supports WBC. The value of VoDeV is 1.
Hi3532/ Hi3518/Hi3516C	The Hi3532/Hi3518/Hi3516C does not support WBC.
Hi3521/Hi3520A/Hi3520D/ Hi3515A/Hi3515C	DHD0 supports WBC. The value of VoDeV is 0.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- The WBC picture size cannot be greater than the display size of a WBC device, and the minification must be within the value range supported by the WBC module. An SPYCbCr422 picture supports at most 1/16 scaling (excluding 1/16 scaling), and an SPYCbCr420 picture supports at most 1/8 scaling (excluding 1/8 scaling).
- If the WBC mode is set to VO_WBC_MODE_PROG_TO_INTL, an SPYCbCr422 picture supports at most 1/8 scaling (excluding 1/8 scaling), and an SPYCbCr420 picture supports at most 1/4 scaling (excluding 1/4 scaling).

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- This MPI is a dynamic MPI. That is, you can call it when WBC is enabled.
- The WBC write-back data source can be set to VO_WBC_DATASOURCE_VIDEO or tVO_WBC_DATASOURCE_MIXER. If the WBC write-back data source is set to VO_WBC_DATASOURCE_VIDEO, the data is not written back to the PIP layer of this device. VO_WBC_DATASOURCE_VIDEO is valid only for VO_WBC_DATASOURCE_E of the Hi3520D/Hi3515A/Hi3515C. For other chips, VO_WBC_DATASOURCE_E is set to VO_WBC_DATASOURCE_MIXER by default, and other values are invalid.
- The width and height must be greater than 32 pixels and 2-pixel aligned.

[Example]

See the example of HI_MPI_VO_EnableWbc.

[See Also]

HI_MPI_VO_GetWbcAttr

HI_MPI_VO_GetWbcAttr

[Description]

Obtains the WBC attribute.

[Syntax]

```
HI_S32 HI_MPI_VO_GetWbcAttr(VO_DEV VoDev, VO_WBC_ATTR_S *pstWbcAttr);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pstWbcAttr	Pointer to the WBC attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531	DHD1 supports WBC. The value of VoDeV is 1.
Hi3532/ Hi3518/Hi3516C	The Hi3532/Hi3518/Hi3516C does not support WBC.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Description
Hi3521/Hi3520A/Hi3520D /Hi3515A/Hi3515C	DHD0 supports WBC. The value of VoDev is 0.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

None

[Example]

See the example of HI_MPI_VO_EnableWbc.

[See Also]

HI_MPI_VO_SetWbcAttr

HI_MPI_VO_SetWbcMode

[Description]

Sets the WBC mode.

[Syntax]

```
HI_S32 HI_MPI_VO_SetWbcMode(VO_DEV VoDev, VO_WBC_MODE_E enWbcMode);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
enWbcMode	WBC mode	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Description
Hi3531	DHD1 supports WBC. The value of VoDeV is 1.
Hi3532/ Hi3518/Hi3516C	The Hi3532/ Hi3518/Hi3516C does not support WBC.
Hi3521/Hi3520A/Hi3520D /Hi3515A/Hi3515C	DHD0 supports WBC. The value of VoDev is 0.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- To achieve the optimal WBC display effect, you can call this MPI to set the WBC mode.
- This MPI is a static interface. That is, the WBC mode can be changed only after the WBC function is disabled.
- Before calling this MPI, you must set WBC attributes. If you disable the WBC function or call this MPI to change the WBC mode, and then enable the WBC function again, you must set WBC attributes again.
- If the WBC mode is set to VO_WBC_MODE_PROG_TO_INTL, an SPYCbCr422 picture supports at most 1/8 scaling (excluding 1/8 scaling), and an SPYCbCr420 picture supports at most 1/4 scaling (excluding 1/4 scaling).

[Example]

See the example of HI_MPI_VO_EnableWbc.

[See Also]

None

HI_MPI_VO_GetWbcMode

[Description]

Obtains the WBC mode.

[Syntax]

```
HI_S32 HI_MPI_VO_GetWbcMode(VO_DEV VoDev, VO_WBC_MODE_E *penWbcMode);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
penWbcMode	Pointer to the WBC mode	Output

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531	DHD1 supports WBC. The value of VoDeV is 1.
Hi3532/ Hi3518/Hi3516C	The Hi3532/ Hi3518/Hi3516C does not support WBC.
Hi3521/Hi3520A/Hi3520D /Hi3515A/Hi3515C	DHD0 supports WBC. The value of VoDev is 0.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

None

[Example]

See the example of HI_MPI_VO_EnableWbc.

[See Also]

None

HI_MPI_VO_SetWbcDepth

[Description]

Sets the depth of the buffer for storing WBC pictures.

[Syntax]

```
HI_S32 HI_MPI_VO_SetWbcDepth(VO_DEV VoDev, HI_U32 u32Depth);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
u32Depth	Depth of the buffer for storing WBC pictures Value range: [0, VO_DEF_WBC_DEPTH_LEN]	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531	DHD1 supports WBC. The value of VoDeV is 1.
Hi3532/ Hi3518/Hi3516C	The Hi3532/ Hi3518/Hi3516C does not support WBC.
Hi3521/Hi3520A/Hi3520D /Hi3515A/Hi3515C	DHD0 supports WBC. The value of VoDeV is 0.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

You can obtain WBC pictures for encoding, displaying pictures in SD mode, or other purposes. In addition, you can set the depth of the buffer for storing WBC pictures. This ensures that no video frames are lost when you obtain WBC pictures.

[Example]

```
HI_S32 s32Ret;
VO_DEV VoDev = 1;
HI_U32 u32Depth;
VIDEO_FRAME_INFO_S *pstFrame =
(VIDEO_FRAME_INFO_S*)malloc(sizeof(VIDEO_FRAME_INFO_S));

s32Ret = HI_MPI_VO_GetWbcDepth (VoDev, &u32Depth);
if (s32Ret != HI_SUCCESS)
{
    printf("Get wbc depth failed with error code %#x!\n", s32Ret);
    return HI_FAILURE;
}
u32Depth = 10;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
s32Ret = HI_MPI_VO_SetWbcDepth (VoDev, u32Depth);
if (s32Ret != HI_SUCCESS)
{
    printf("Set wbc depth failed with error code %#x!\n", s32Ret);
    return HI_FAILURE;
}

if (HI_SUCCESS != HI_MPI_VO_WbcGetScreenFrame(VoDev, pstFrame))
{
    printf("Get wbc screen frame failed!\n");
    return HI_FAILURE;
}

/*Perform processing, for example, store file or perform JPEG encoding*/

if (HI_SUCCESS != HI_MPI_VO_WbcReleaseScreenFrame(VoDev, pstFrame))
{
    printf("Release wbc screen frame failed!\n");
    return HI_FAILURE;
}
```

[See Also]

HI_MPI_VO_WbcGetScreenFrame

HI_MPI_VO_GetWbcDepth

[Description]

Gets the depth of the buffer for storing WBC pictures.

[Syntax]

```
HI_S32 HI_MPI_VO_GetWbcDepth(VO_DEV VoDev, HI_U32 *pu32Depth);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pu32Depth	Depth of the buffer for storing WBC pictures	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
	details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531	DHD1 supports WBC. The value of VoDev is 1.
Hi3532/ Hi3518/Hi3516C	The Hi3532/ Hi3518/Hi3516C does not support WBC.
Hi3521/Hi3520A/Hi3520D /Hi3515A/Hi3515C	DHD0 supports WBC. The value of VoDev is 0.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

None

[Example]

See the example of HI_MPI_VO_SetWbcDepth.

[See Also]

HI_MPI_VO_WbcGetScreenFrame

HI_MPI_VO_WbcGetScreenFrame

[Description]

Obtains WBC pictures.

[Syntax]

```
HI_S32 HI_MPI_VO_WbcGetScreenFrame(VO_DEV VoDev, VIDEO_FRAME_INFO_S
*pstVFrame);
```

[Parameter]

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pstFrame	Pointer to the obtained information about the WBC pictures	Output

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531	DHD1 supports WBC. The value of VoDev is 1.
Hi3532/ Hi3518/Hi3516C	The Hi3532/ Hi3518/Hi3516C does not support WBC.
Hi3521/Hi3520A/Hi3520D/ Hi3515A/Hi3515C	DHD0 supports WBC. The value of VoDeV is 0.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, you must set the depth of the buffer for storing the WBC picture to a value greater than 0. Otherwise, no pictures can be obtained.
- Before calling this MPI, you must enable WBC. Otherwise, no pictures can be obtained.
- You must release the buffer for storing the pictures in time.

[Example]

See the example of HI_MPI_VO_SetWbcDepth.

[See Also]

HI_MPI_VO_WbcReleaseScreenFrame

HI_MPI_VO_WbcReleaseScreenFrame

[Description]

Releases the buffer for storing WBC pictures.

[Syntax]

```
HI_S32 HI_MPI_VO_WbcReleaseScreenFrame(VO_DEV VoDev, VIDEO_FRAME_INFO_S
*pstVFrame);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VoDev	VO device ID	Input
pstFrame	Pointer to the information about the released pictures of a VO channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531	DHD1 supports WBC. The value of VoDev is 1.
Hi3532/ Hi3518/Hi3516C	The Hi3532/ Hi3518/Hi3516C does not support WBC.
Hi3521/Hi3520A/Hi3520D/Hi3515A/ Hi3515C	DHD0 supports WBC. The value of VoDeV is 0.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

After calling HI_MPI_VO_WbcGetScreenFrame, you must call HI_MPI_VO_WbcReleaseScreenFrame.

[Example]

See the example of HI_MPI_VO_SetWbcDepth.

[See Also]

HI_MPI_VO_WbcGetScreenFrame

HI_MPI_VO_GfxLayerBindDev

[Description]

Binds a graphics layer to a VO device.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_VO_GfxLayerBindDev(VOU_GFX_BIND_LAYER_E enGfxLayer, VO_DEV
VoTargetDev);
```

[Parameter]

Parameter	Description	Input/Output
enGfxLayer	ID of a graphics layer Value range: [0, VOU_GRAPHICS_LAYER_BUTT)	Input
VoTargetDev	VO device ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

Error Code	Description
HI_SUCCESS	Success
HI_ERR_VO_GFX_NOT_DISABLE	The graphics layer is not disabled.
HI_ERR_VO_GFX_NOT_UNBIND	The graphics layer is still bound.

[Difference]

Chip	Description
Hi3531	G4, HC0, and HC1 can be dynamically bound. The value range of VoTargetDev is [0, 3].
Hi3532/ Hi3518/Hi3516C	The Hi3532/ Hi3518/Hi3516C does not support this MPI.
Hi3521/Hi3520A/Hi3520D/ Hi3515A/Hi3515C	Only HC0 can be dynamically bound. The value range of VoTargetDev is [0, 2].

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

- Before calling this MPI, ensure that the graphics layer to be bound is enabled.
- A graphics layer cannot be repeatedly bound. Before rebinding a graphics layer, you must unbind it.
- Two cursor layers cannot be bound to the same VO device at the same time.
- For the Hi3520D/Hi3515A/Hi3515C, when a cursor layer is dynamically bound to another device, the CSC mode of the cursor layer does not automatically change. You are advised to set the CSC mode as follows:
  - When the cursor layer is bound to an HD device, set the CSC mode to VO_CSC_MATRIX_RGB_TO_BT709_PC.
  - When the cursor mode is bound to an SD device, set the CSC mode to VO_CSC_MATRIX_RGB_TO_BT601_PC.

[Example]

None

[See Also]

HI_MPI_VO_GfxLayerUnBindDev

HI_MPI_VO_GfxLayerUnBindDev

[Description]

Unbinds a graphics layer from a VO device.

[Syntax]

```
HI_S32 HI_MPI_VO_GfxLayerUnBindDev(VOU_GFX_BIND_LAYER_E enGfxLayer,
VO_DEV VoTargetDev);
```

[Parameter]

Parameter	Description	Input/Output
enGfxLayer	ID of a graphics layer Value range: [0, VOU_GRAPHICS_LAYER_BUTT)	Input
VoTargetDev	VO device ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Error Code]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Description
HI_SUCCESS	Success
HI_ERR_VO_GFX_NOT_DISABLE	The graphics layer is not disabled.

[Difference]

Chip	Description
Hi3531	G4, HC0, and HC1 can be dynamically bound. The value range of VoTargetDev is [0, 3].
Hi3532/ Hi3518/Hi3516C	The Hi3532/ Hi3518/Hi3516C does not support this MPI.
Hi3521/Hi3520A/Hi3520D /Hi3515A/Hi3515C	Only HC0 can be dynamically bound. The value range of VoTargetDev is [0, 2].

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, ensure that the graphics layer to be bound is disabled.
- VoTargetDev is meaningless. You can set it to 0.
- If you unbind a graphics layer that is not bound before, a code indicating success is returned. That is, a graphics layer can be unbound for multiple times.

[Example]

None

[See Also]

HI_MPI_VO_GfxLayerBindDev

HI_MPI_VO_SetGfxLayerCSC

[Description]

Sets the picture output effect of a graphics layer.

[Syntax]

```
HI_S32 HI_MPI_VO_SetGfxLayerCSC(HI_U32 u32Layer, const VO_CSC_S *pstCSC);
```

[Parameter]

Parameter	Description	Input/Output
u32Layer	ID of a graphics layer Value range: [0, VOU_GRAPHICS_LAYER_ NUM)	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
pstCSC	Pointer to the picture output effect of a graphics layer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532/ Hi3521/Hi3520A	The Hi3531, Hi3532, Hi3521, or Hi3520A allows you to adjust the luminance, contrast, hue, and saturation and select the CSC matrix.
Hi3518/Hi3516C	The Hi3518/Hi3516C only allows you to select the CSC matrix.
Hi3520D/Hi3515A /Hi3515C	G0/G1/G2 supports the adjustment of the luminance, contrast, chrominance, and saturation and the selection of the CSC conversion matrix. The HC0 cursor layer supports only the selection of the conversion matrix. You can randomly set other four attribute values. The images are not affected. If the attributes are not set when you obtain the values of these attributes, the default values are returned. Otherwise, the configured values are returned.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- This MPI is used to adjust the picture output effect, including the luminance, contrast, hue, and saturation. The effect value ranges from 0 to 100.
- The CSC matrix must be VO_CSC_MATRIX_RGB_TO_BT601_PC or VO_CSC_MATRIX_RGB_TO_BT709_PC.
- If the graphics layer is disabled, the luminance, contrast, hue, and saturation are restored to the default value.

[Example]

None

[See Also]

HI_MPI_VO_GetGfxLayerCSC

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VO_GetGfxLayerCSC

[Description]

Obtains the picture output effect of a graphics layer.

[Syntax]

```
HI_S32 HI_MPI_VO_GetGfxLayerCSC(HI_U32 u32Layer, VO_CSC_S *pstCSC);
```

[Parameter]

Parameter	Description	Input/Output
u32Layer	ID of a graphics layer Value range: [0, VOU_GRAPHICS_LAYER_ NUM))	Input
pstCSC	Pointer to the picture output effect of a graphics layer	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

This MPI is used to obtain the picture output effect, including the luminance, contrast, hue, and saturation. The effect value ranges from 0 to 100.

[Example]

None

[See Also]

HI_MPI_VO_SetGfxLayerCSC

HI_MPI_VO_SetCascadeAttr

[Description]

Sets the cascade attribute.

[Syntax]

```
HI_S32 HI_MPI_VO_SetCascadeAttr(const VO_CAS_ATTR_S *pstCasAttr);
```


Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
pstCasAttr	Pointer to the VO cascade attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	When the Hi3531 is cascaded with the Hi3532, the 64- or 32-region mode is supported. When the Hi3520 is cascaded with the Hi3531 or Hi3532, only the 32-region mode is supported. When the Hi3531 is cascaded with the Hi3532, 1080p60 data is transmitted as odd frames and even frames through dual channels. When the Hi3520 is cascaded with the Hi3531 or Hi3532, enCasMode must be set to single-channel cascade mode, and the data must be 1080p30 cascade data.
Hi3521/Hi3520A/Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C	The Hi3521/Hi3520A/Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C does not support this MPI.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Set the video cascade attribute before enabling a device. That is, ensure that the device is disabled before setting the video cascade attribute.
- The bSlave parameter indicates the master/slave mode. The first cascaded chip is set to master mode and the other cascaded chips are set to slave mode.
- The enCasMode parameter indicates the single- or dual-channel mode.
  - In dual-channel cascade mode, the 1080p60 data is transmitted as odd frames and even frames through dual channels. That is, dual-channel 1080p30 data is transferred.
  - In single-channel cascade mode, the 1080p60 or 1080p30 data is supported.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Example]

None

[See Also]

HI_MPI_VO_GetCascadeAttr

HI_MPI_VO_GetCascadeAttr

[Description]

Obtains the cascade attribute.

[Syntax]

```
HI_S32 HI_MPI_VO_GetCascadeAttr(VO_CAS_ATTR_S *pstCasAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstCasAttr	Pointer to the VO cascade attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531 or Hi3532 supports this MPI.
Hi3521/Hi3520A/Hi3518/Hi3516C /Hi3520D/Hi3515A/Hi3515C	The Hi3521/Hi3520A/Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C does not support this MPI.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

None

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[See Also]

HI_MPI_VO_SetCascadeAttr

HI_MPI_VO_EnableCascadeDev

[Description]

Enables an extended cascade device.

[Syntax]

```
HI_S32 HI_MPI_VO_EnableCascadeDev (VO_DEV VoCasDev);
```

[Parameter]

Parameter	Description	Input/Output
VoCasDev	Extended cascade device	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531 or Hi3532 support this MPI.
Hi3521/Hi3520A/Hi3518/Hi3516C /Hi3520D/Hi3515A/Hi3515C	The Hi3521/Hi3520A/Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C does not support this MPI.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- The extended cascade device is a virtual software device rather than an actual hardware device.
- When the cascade mode is set to dual-channel mode, the two extended cascade devices VO_CAS_DEV_1 and VO_CAS_DEV_2 must be enabled to transfer cascade data.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- When the cascade mode is set to single-channel mode, only VO_CAS_DEV_1 needs to be enabled.
- An extended cascade device can be enabled by calling this MPI only after the video layer of the cascade device Dev0 is enabled.
- For an extended cascade device, you do not need to set the hardware device attributes such as the public attribute and video layer attribute. That is, you only need to set the attributes of Dev0.
- In cascade mode, the channel of Dev0 is unavailable. Other modules must be bound to the channel of an extended cascade device to transfer data.

[Example]

None

[See Also]

HI_MPI_VO_DisableCascadeDev

HI_MPI_VO_DisableCascadeDev

[Description]

Disables an extended cascade device.

[Syntax]

```
HI_S32 HI_MPI_VO_DisableCascadeDev (VO_DEV VoCasDev);
```

[Parameter]

Parameter	Description	Input/Output
VoCasDev	Extended cascade device	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531 or Hi3532 supports this MPI.
Hi3521/Hi3520A/Hi3518/Hi3516C /Hi3520D/Hi3515A/Hi3515C	The Hi3521/Hi3520A/Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C does not support this MPI.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- This MPI can be repeatedly called and a code indicating success is returned.
- Before calling this MPI, you must disable all channels of the extended cascade devices.

[Example]

None

[See Also]

HI_MPI_VO_EnableCascadeDev

HI_MPI_VO_SetCascadePattern

[Description]

Sets the picture pattern during video cascading.

[Syntax]

```
HI_S32 HI_MPI_VO_SetCascadePattern(VO_DEV VoCasDev, HI_U32 u32Pattern);
```

[Parameter]

Parameter	Description	Input/Output
VoCasDev	Extended cascade device	Input
u32Pattern	Picture pattern during video cascading Value range: [0, 127]	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531 or Hi3532 supports this MPI.
Hi3521/Hi3520A/Hi3518/Hi3516C /Hi3520D/Hi3515A/Hi3515C	The Hi3521/Hi3520A/Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C does not support this MPI.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

The value of the picture pattern parameter u32Pattern is a positive integer smaller than 128. The value is used to check whether the pictures transmitted between the chips match. Pictures can be properly transmitted only when picture patterns are consistent.

- The pattern value indicates the picture pattern. For example, you can set the pattern value to 4, 9, and 16 to represent the 4-picture mode, 9-picture mode, and 16-picture mode respectively. The pattern values of the cascaded chips must be the same. However, the pattern value can be different from the picture mode value. For example, the value 1 can be used to express the 16-picture mode for each cascaded chip.
- The pattern value also ensures picture synchronization when the picture mode is switched. Assume that all the cascaded chips are in 4-picture mode. When a chip is switched to 9-picture mode and then other chips are switched to 9-picture pattern, incorrect frames occur if the pattern value is retained. If you set the pattern value to 4 in 4-picture mode and set the pattern value to 9 in 9-picture pattern, incorrect pictures will be dropped. This ensures that the picture mode is properly switched.

[Example]

None

[See Also]

HI_MPI_VO_GetCascadePattern

HI_MPI_VO_GetCascadePattern

[Description]

Obtains the picture pattern during video cascading.

[Syntax]

```
HI_S32 HI_MPI_VO_GetCascadePattern(VO_DEV VoCasDev, HI_U32 *pu32Pattern);
```

[Parameter]

Parameter	Description	Input/Output
VoCasDev	Extended cascade device	Input
pu32Pattern	Pointer to the picture pattern during video cascading	Output

[Return Value]

Return Value	Description
0	Success

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531 or Hi3532 supports this MPI.
Hi3521/Hi3520A/Hi3518/Hi3516C /Hi3520D/Hi3515A/Hi3515C	The Hi3521/Hi3520A/Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C does not support this MPI.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_VO_SetCascadePattern

HI_MPI_VO_CascadePosBindChn

[Description]

Binds a cascade region to a VO channel.

[Syntax]

```
HI_S32 HI_MPI_VO_CascadePosBindChn (HI_U32 u32Pos, VO_DEV VoCasDev,
VO_CHN VoChn);
```

[Parameter]

Parameter	Description	Input/Output
u32Pos	Video cascade position ID Value range: [0, 31] or [0, 63]	Input
VoCasDev	Extended cascade device	Input
VoChn	ID of the VO channel Value range: [0, 63]	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531 or Hi3532 supports this MPI.
Hi3521/Hi3520A/Hi3518/Hi3516C /Hi3520D/Hi3515A/Hi3515C	The Hi3521/Hi3520A/Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C does not support this MPI.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- The video cascade position ID must be unique.
- When the cascade region mode is 64-region mode, the value range of u32Pos is [0, 63]; when the cascade region mode is 32-region mode, the value range of u32Pos is [0, 31].
- The u32Pos value must be unique. If the u32Pos values of two or more channels are the same, an error may occur when pictures are transferred. The u32Pos value is the alias of a channel. All the channels to be cascaded must have unique names. This ensures that the pictures are properly transferred.
- You can bind cascade regions to VO channels in batches by calling HI_MPI_VO_CascadePosBindChn. You are advised to call HI_MPI_VO_SetAttrBegin, HI_MPI_VO_CascadePosBindChn, and HI_MPI_VO_SetAttrEnd in sequence.

[Example]

None

[See Also]

HI_MPI_VO_CascadePosUnBindChn

HI_MPI_VO_CascadePosUnBindChn

[Description]

Unbinds a cascade region from a VO channel.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
HI_S32 HI_MPI_VO_CascadePosUnBindChn(HI_U32 u32Pos, VO_DEV VoCasDev,
VO_CHN VoChn);
```

[Parameter]

Parameter	Description	Input/Output
u32Pos	Video cascade position ID Value range: [0, 31] or [0, 63]	Input
VoCasDev	Extended cascade device	Input
VoChn	ID of the VO channel Value range: [0, 63]	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531 or Hi3532 supports this MPI.
Hi3521/Hi3520A/Hi3518/Hi3516C /Hi3520D/Hi3515A/Hi3515C	The Hi3521/Hi3520A/Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C does not support this MPI.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- The video cascade position ID must be unique.
- When the cascade region mode is 64-region mode, the value range of Pos is [0, 63]; when the cascade region mode is 32-region mode, the value range of Pos is [0, 31].
- You can unbind cascade regions from VO channels in batches by calling HI_MPI_VO_CascadePosUnBindChn. You are advised to call HI_MPI_VO_SetAttrBegin, HI_MPI_VO_CascadePosUnBindChn, and HI_MPI_VO_SetAttrEnd in sequence.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[See Also]

HI_MPI_VO_CascadePosBindChn

HI_MPI_VO_EnableCascade

[Description]

Enables the video cascade function.

[Syntax]

```
HI_S32 HI_MPI_VO_EnableCascade (HI_VOID);
```

[Parameter]

None

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531 or Hi3532 supports this MPI.
Hi3521/Hi3520A/Hi3518/Hi3516C /Hi3520D/Hi3515A/Hi3515C	The Hi3521/Hi3520A/Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C does not support this MPI.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

- Only Dev0 supports the video cascade function.
- Before enabling the video cascade function, you must set the cascade attribute and enable Dev0 and its video layer.

[Example]

None

[See Also]

HI_MPI_VO_DisableCascade

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VO_DisableCascade

[Description]

Disables the video cascade function.

[Syntax]

```
HI_S32 HI_MPI_VO_DisableCascade(HI_VOID);
```

[Parameter]

None

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 4.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531 or Hi3532 supports this MPI.
Hi3521/Hi3520A/Hi3518/Hi3516C /Hi3520D/Hi3515A/Hi3515C	The Hi3521/Hi3520A/Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C does not support this MPI.

[Requirement]

- Header files: mpi_vo.h, hi_comm_vo.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_VO_EnableCascade

4.4 Data Types

The VO data types are as follows:

- VO_DEV: Defines the device ID.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- VO_PUB_ATTR_S: Defines the attributes of a VO device.
- VO_VIDEO_LAYER_ATTR_S: Defines the attributes of a video layer.
- VO_CHN_ATTR_S: Defines the attributes of a VO channel.
- VO_DISPLAY_FIELD_E: Defines the field mode during video output.
- VO_QUERY_STATUS_S: Defines the status of a VO channel.
- VO_ZOOM_RATIO_S: Defines the partial zoom-in ratio.
- VO_ZOOM_ATTR_S: Defines the partial zoom-in attributes.
- VO_CSC_S: Defines the picture output effect.
- VO_VGA_PARAM_S: Defines the VGA picture output effect.
- VO_CSC_MATRIX_E: Defines the CSC matrix.
- VO_WBC_ATTR_S: Defines the WBC attribute.
- VO_WBC_MODE_E: Defines the video WBC mode.
- VO_CAS_MODE_E: Defines the cascade channel mode.
- VO_CAS_RGN_E: Define the cascade region mode.
- VO_CAS_ATTR_S: Defines the cascade attribute.

VO_DEV

[Description]

Defines the device ID.

[Syntax]

```
typedef HI_S32 VO_DEV;
```

[Member]

Chip	Description
Hi3531	The VOU supports the following 14 VO devices: 0: DHD0 1: DHD1 2: DSD0 3: DSD1 4: DSD2 5: DSD3 6: DSD4 7: DSD5 8: VO_CAS_DEV_1 (extended cascade device 1) 9: VO_CAS_DEV_2 (extended cascade device 2) 10: VO_VIRT_DEV_0 device (virtual device 0) 11: VO_VIRT_DEV_1 device (virtual device 1) 12: VO_VIRT_DEV_2 device (virtual device 2) 13: VO_VIRT_DEV_3 device (virtual device 3)
Hi3532	The VOU supports the following 14 VO devices:

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Description
	0: DHD0 1−7: reserved 8: VO_CAS_DEV_1 (extended cascade device 1) 9: VO_CAS_DEV_2 (extended cascade device 2) 10: VO_VIRT_DEV_0 device (virtual device 0) 11: VO_VIRT_DEV_1 device (virtual device 1) 12: VO_VIRT_DEV_2 device (virtual device 2) 13: VO_VIRT_DEV_3 device (virtual device 3)
Hi3521/Hi3520A/Hi3520D /Hi3515A/Hi3515C	The VOU supports the following 7 VO devices: 0: DHD0 1: DSD0 2: DSD1 3: VO_VIRT_DEV_0 device (virtual device 0) 4: VO_VIRT_DEV_1 device (virtual device 1) 5: VO_VIRT_DEV_2 device (virtual device 2) 6: VO_VIRT_DEV_3 device (virtual device 3)
Hi3518/Hi3516C	The VOU supports the following VO device: 0: DSD1

[Difference]

Chip	Description
Hi3531	• The HD device can output HD digital and analog signals such as BT.1120, HDMI, and VGA (YPbPr) signals at the same time. • Two HD devices support only 1-channel BT.1120, HDMI, or VGA (YPbPr) signal at the same time. • The VGA signal and YPbPr signal cannot be output at the same time. • The HD device supports BT.656 output. When DHD0 outputs the BT.656 signal, other output interfaces of DHD0 are unavailable; when DHD1 outputs the BT.656 signal, other output interfaces of DHD1 are unavailable. • DSD0 or DSD1 can output the BT.656 signal and CVBS signal (PAL norm or NTSC norm) at the same time. DSD2 to DSD5 can output only the BT.656 signal. The video layer of DSD5 is multiplexed with the PIP layer of the HD device. That is, the video layer of DSD5 and PIP layer cannot be used at the same time. • Each device supports a maximum of 64 channels. • As the extended cascade devices are not actual hardware devices, their device attributes and video layer attributes

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Description
	cannot be set. The Hi3532 also supports the two extended cascade devices.
Hi3532	• The Hi3532 supports only one HD device (DHD0), and only BT.1120 data is output. • As the extended cascade devices are not actual hardware devices, their device attributes and video layer attributes cannot be set. The Hi3532 also supports the two extended cascade devices.
Hi3521/Hi3520A	• The HD device can output HD digital and analog signals such as BT.1120 (LCD), HDMI, and VGA signals at the same time. • The LCD, BT.1120, and VGA signals cannot be output at the same time. If the LCD interface is enabled, the BT.656 output interface of the SD device is unavailable. • The SD device can output BT.656 and CVBS signals at the same time. The BT.656 output interface of the SD device can output the data from DSD0 or DSD1. • The upper eight bits or lower eight bits of the BT.1120 interface can act as the BT.656 output interface for the SD device. The BT.656 interface can output the video data from DSD0 or DSD1. This function is supported only when the hardware connection is prepared. In this case, the BT.1120 interface is unavailable. The BT.656 interface formed by using special connections and the original BT.656 interface of the SD device cannot output the video data from a device at the same time.
Hi3518/Hi3516C	• HD devices are not supported. • The SD device can output BT.1120 or CVBS signals, but BT.1120 and CVBS signals cannot be output at the same time.
Hi3520D/Hi3515A/Hi3515C	• The HD device can output HD digital and analog signals such as HDMI and VGA signals at the same time. • The SD device outputs only CVBS signals.

[Note]

None

[See Also]

VO_PUB_ATTR_S

VO_PUB_ATTR_S

[Description]

Defines the public attributes of a VO device.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
typedef struct hiVO_PUB_ATTR_S
{
   HI_U32                u32BgColor;        /*Background color of a
device, in RGB format.*/
   VO_INTF_TYPE_E        enIntfType;        /*Type of a VO interface*/
   VO_INTF_SYNC_E        enIntfSync;        /*Type of a VO interface
timing*/
   VO_SYNC_INFO_S        stSyncInfo;        /*Information about VO
interface timings*/
   HI_BOOL               bDoubleFrame;      /*Whether to double frames*/
} VO_PUB_ATTR_S;
```

[Member]

Member	Description
u32BgColor	Background color of a VO device, in RGB888 format

enIntfType	Typical configuration of the interface type. The prototype is defined as follows:
	`Typedef HI_S32 VO_INTF_TYPE_E;`
	`#define VO_INTF_CVBS (0x01L<<0)`
	`#define VO_INTF_YPBPR (0x01L<<1)`
	`#define VO_INTF_VGA (0x01L<<2)`
	`#define VO_INTF_BT656 (0x01L<<3)`
	`#define VO_INTF_BT1120 (0x01L<<4)`
	`#define VO_INTF_HDMI (0x01L<<5)`
	`#define VO_INTF_LCD (0x01L<<6)`
	`#define VO_INTF_BT656_H (0x01L<<7)`
	`#define VO_INTF_BT656_L (0x01L<<8)`

enIntfSync	Typical configuration of the interface timing. The prototype is defined as follows:
	`typedef enum hiVO_INTF_SYNC_E`
	`{`
	`VO_OUTPUT_PAL = 0,`
	`VO_OUTPUT_NTSC,`
	`VO_OUTPUT_1080P24,`
	`VO_OUTPUT_1080P25,`
	`VO_OUTPUT_1080P30,`
	`VO_OUTPUT_720P50,`
	`VO_OUTPUT_720P60,`
	`VO_OUTPUT_1080I50,`
	`VO_OUTPUT_1080I60,`
	`VO_OUTPUT_1080P50,`

Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	VO_OUTPUT_1080P60, VO_OUTPUT_576P50, VO_OUTPUT_480P60, VO_OUTPUT_800x600_60, VO_OUTPUT_1024x768_60, VO_OUTPUT_1280x1024_60, VO_OUTPUT_1366x768_60, VO_OUTPUT_1440x900_60, VO_OUTPUT_1280x800_60, VO_OUTPUT_1600x1200_60, VO_OUTPUT_1680x1050_60, VO_OUTPUT_1920x1200_60, VO_OUTPUT_640x480_60, VO_OUTPUT_1920x2160_30, VO_OUTPUT_2560x1440_30, VO_OUTPUT_USER, VO_OUTPUT_BUTT } VO_INTF_SYNC_E;
stSyncInfo	Structure of the interface timing. The prototype is defined as follows: typedef struct tagVO_SYNC_INFO_S { HI_BOOL bSynm; HI_BOOL bIop; HI_U8 u8Intfb; HI_U16 u16Vact ; HI_U16 u16Vbb; HI_U16 u16Vfb; HI_U16 u16Hact; HI_U16 u16Hbb; HI_U16 u16Hfb; HI_U16 u16Hmid; HI_U16 u16Bvact; HI_U16 u16Bvbb; HI_U16 u16Bvfb; HI_U16 u16Hpw; HI_U16 u16Vpw; HI_BOOL bIdv;

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	HI_BOOL bIhs; HI_BOOL bIvs; } VO_SYNC_INFO_S;

[Difference]

Chip	Description
Hi3531	The Hi3531 supports the following timings: VO_INTF_CVBS, VO_INTF_YPBPR, VO_INTF_VGA, VO_INTF_BT656, VO_INTF_BT1120, and VO_INTF_HDMI. When the interface timing is VO_OUTPUT_1920x2160_30, the rate of the timings of the video layer, PIP, and WBC must be set to 60 fps (otherwise, frames are discarded), and the HDMI video format is set to HI_HDMI_VIDEO_FMT_1080P_60. When the interface timing is VO_OUTPUT_2560x1440_30, the rate of the timings of the video layer, PIP, and WBC must be set to 60 fps (otherwise, frames are discarded), and the HDMI video format is set to HI_HDMI_VIDEO_FMT_VESA_2560X1440_30.
Hi3532	The Hi3532 supports only the VO_INTF_BT1120 timing.
Hi3521/Hi3520A	The Hi3521 supports the following timings: VO_INTF_CVBS, VO_INTF_VGA, VO_INTF_BT656, VO_INTF_BT1120, VO_INTF_HDMI, VO_INTF_LCD, VO_INTF_BT656_H, and VO_INTF_BT656_L. VO_INTF_LCD cannot be used with BT1120, VGA, BT656, BT656_H, and BT656_L at the same time. BT1120 cannot be used with BT656_H and BT656_L at the same time.
Hi3518/Hi3516C	The Hi3518/Hi3516C supports the VO_INTF_BT1120 and VO_INTF_CVBS timings. VO_INTF_BT1120 and VO_INTF_CVBS cannot be used at the same time.
Hi3520D/Hi3515A/Hi3515C	The Hi3520D/Hi3515A/Hi3515C supports the VO_INTF_CVBS, VO_INTF_VGA, and VO_INTF_HDMI timings.

[Note]

- The timing structure defined by stSyncInfo is valid only when the interface timing is set to VO_OUTPUT_USER, which indicates the user-defined timing.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- If you want to use multiple interface types at the same time, you can set enIntfSync as follows: enIntfSync = VO_INTF_BT1120 | VO_INTF_HDMI
- When the interface type is set to VO_INTF_CVBS or VO_INTF_BT656, the value of enIntfSync is 0 or 1.
- When the interface type is set to VO_INTF_BT1120, VO_INTF_HDMI, or VO_INTF_VGA, the value range of enIntfSync is [2, 22]. When the interface type is set to VO_INTF_YPBPR, the value range of enIntfSync is [5, 10].
- The configured device attributes take effect after the device is restarted.

[See Also]

HI_MPI_VO_SetPubAttr

VO_VIDEO_LAYER_ATTR_S

[Description]

Defines the attributes of a video layer.

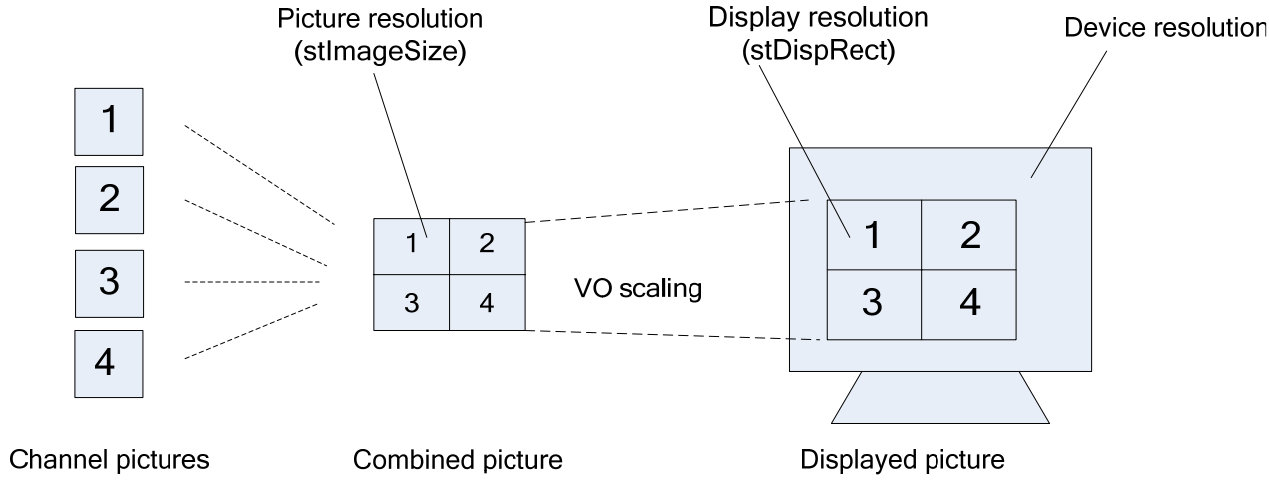
The attributes of a video layer include the device resolution, display resolution, and picture resolution. Figure 4-1 illustrates each resolution.

The picture resolution is the size of the canvas for placing the picture of each channel.

The display resolution is the size of the display region that is obtained after the VOU scales the canvas.

The device resolution is the same as the resolution defined in the device timing. For example, if the resolution defined in the timing is 1920x1080, the device resolution is 1920x1080.

Figure 4-1 Concepts related to video layer attributes



[Syntax]

```
typedef struct hiVO_VIDEO_LAYER_ATTR_S
{
RECT_S stDispRect;                /*Display resolution*/
SIZE_S stImageSize;               /*Canvas size of the video layer*/
```

```
HI_U32 u32DispFrmRt;              /*Display frame rate*/
PIXEL_FORMAT_E enPixFormat;       /*Pixel format of the video layer*/
} VO_VIDEO_LAYER_ATTR_S;
```

[Member]

Member	Description
stDispRect	Rectangular video display region
stImageSize	Picture resolution, that is, the size of the combined picture
u32DispFrmRt	Frame rate for video display
enPixFormat	Input pixel format of the video layer. The value is SPYCbCr420 or SPYCbCr422.

[Difference]

Chip	Description
Hi3531	The maximum resolution of the video layer of DHD0 is 2560 x 1440. The value of stDispRect cannot be greater than 2560 x 1440. The maximum resolution of the video layer of DHD1 is 2560x1600, and the maximum resolution of the video layer of DSD0–DSD5 is 720x576. There are restrictions on HD scaling. To be specific, DHD0 does not allow the width between 1280 pixels and 1920 pixels to be scaled to 1920 pixels. When the width exceeds 1920 pixels, only YUV422 pictures can be displayed. DHD1 does not allow the width between 2048 pixels and 2560 pixels to be scaled to 2560 pixels.
Hi3532	The maximum resolution of the video layer of DHD0 is 1920x1080. The value of stDispRect cannot be greater than 1920x1080. The video layer cannot be scaled.
Hi3521/Hi3520A	The maximum resolution of the video layer of DHD0 is 1920x1080. The value of stDispRect cannot be greater than 1920x1080. The maximum resolution of the video layer of DSD0 and DSD1 is 720x576. There are restrictions on HD scaling. To be specific, DHD0 does not allow the width between 1280 pixels and 1920 pixels to be scaled to 1920 pixels.
Hi3518/Hi3516C	The maximum resolution of the video layer of DSD1 is 1920x1080. The value of stDispRect cannot be greater than 1920x1080.
Hi3520D/Hi3515A /Hi3515C	The maximum resolution of DHD0 is 1920x2048. The value of stDispRect cannot be greater than 1920x2048. The maximum resolution of the video layer of DSD0 or DSD1 is 720x576. There is limit on the HD scaling. DHD0 cannot zoom in the resolution whose width is greater than 1440.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

- The value of stDispRect cannot greater than the device resolution. The values of stImageSize and stDispRect must be greater than or equal to the minimum resolution 32x32.
- The HD device supports scaling rather than clipping. If the value of stImageSize is less than the value of stDispRect, the picture at the video layer is zoomed in from stImageSize to stDispRect. Note that that value of stImageSize cannot be greater than the value of stDispRect.
- The SD device supports clipping rather than scaling. If the value of stImageSize is greater than the value of stDispRect, the combined picture is clipped. If the value of stImageSize is less than the value of stDispRect.
- For the SD device, the vertical coordinate of the start point (Y) and height (H) of stImageSize or stDispRect must be 4-pixel aligned, because the SD device writes the information about the picture overlaid by the TDE to the hardware and the hardware has such requirement. The resolution of the overlaid picture depends on stImageSize and stDispRect. In addition, the horizontal coordinate of start point (X) and width (W) must be 2-pixel aligned.
- For the HD device, the values of stImageSize and stDispRect must be 2-pixel aligned.
- u32DispFrmRt is the frame rate for video display. The value range of u32DispFrmRt is (0, VO_DISP_MAX_FRMRATE]. It is generally set to the full frame rate of the channel.
  - u32DispFrmRt is set to 25 if the source is in PAL format.
  - u32DispFrmRt is set to 30 if the source is in NTSC format.

  If the configured frame rate is greater than the source frame rate of the channel, the performance is not fully used.
- The SD device does not support VO scaling. When the SD device outputs pictures, the picture resolution is the same as the display resolution in general.
- The HD device supports only VO zoom-in. That is, the picture resolution cannot be greater than the display resolution.
- The start position (s32X, s32Y) of stDispRect for the SD device or HD device is invalid. The start coordinates of the video layer are fixed at (0, 0). You can specify any display position by setting the start position of each channel.

[See Also]

HI_MPI_VO_SetPubAttr

VO_CHN_ATTR_S

[Description]

Defines the attributes of a VO channel.

[Syntax]

```
typedef struct hiVO_CHN_ATTR_S
{
  HI_U32  u32Priority;              /*Channel priority*/
  RECT_S  stRect;                  /*Channel display region*/
  HI_BOOL bDeflicker;              /*Anti-flicker enable*/
}VO_CHN_ATTR_S;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Member]

Member	Description
u32Priority	Overlay priority of the video channel. The channel with low priority is overlaid with the one with high priority. Priority range for the HD device: [0, 1] Priority range for the SD device: [0, VO_MAX_CHN_NUM − 1] Dynamic attribute
stRect	Rectangular display region of the channel. The upper left corner of the screen acts as the coordinate origin. The value must be 2-pixel aligned, and the rectangular display region must fall within the screen. Note that the height must be 4-pixel aligned if the HD device's timing is interlaced and the display format is SPYCbCr420. Dynamic attribute
bDeflicker	Anti-flicker enable • HI_TRUE: enabled • HI_FALSE: disabled Anti-flicker is valid only for the device channels whose pictures can be scaled by the TDE. That is, this parameter is valid only for SD devices and virtual devices. Dynamic attribute

[Difference]

Chip	Description
Hi3531	• For the HD device, the channel priorities are 0 and 1. • For the SD device, the channel priorities are 0 to (VO_MAX_CHN_NUM – 1). • The width or height of an SD device channel must be greater than or equal to 16 pixels.
Hi3532	• The channel priority for the HD device can be set only to 0. • The width or height of an SD device channel must be greater than or equal to 16 pixels.
Hi3521/Hi3520A/ Hi3520D/Hi3515A /Hi3515C	• For the HD device, the channel priorities are 0 and 1. • For the SD device, the channel priorities are 0 to (VO_MAX_CHN_NUM – 1). • The width or height of an SD device channel must be greater than or equal to 16 pixels.
Hi3518/Hi3516C	• For the SD device, the channel priorities are 0 to (VO_MAX_CHN_NUM – 1). • The width or height of an SD device channel must be greater than or equal to 32 pixels.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

- The greater the priority value is, the higher the priority.
  - For the HD device, the channel priorities are 0 and 1. Priority 0 indicates that the channel is at the original video layer of the device and priority 1 indicates that the channel is at the PIP layer. The original video layer is always overlaid with the PIP layer. The channels of the video layer of an HD device or the channels of the PIP layer cannot overlap each other.
  - For the SD device, the channel priorities are 0–VO_MAX_CHN_NUM-1. When the display regions of multiple channels overlap, the picture from the channel with a lower priority is overlaid with the picture from the channel with a higher priority. When the display regions of multiple channels with the same priority are overlaid, the pictures from the channel with small ID are overlaid with the pictures from the channel with large ID.
- The channel display region cannot exceed the canvas size (stImageSize).
- The channel attribute stRect of the HD device must be 2-pixel aligned. In interlaced mode, the width of stRect must be 4-aligned if the pixel format is SEMIPLANAR_420. In addition, the width or height of an HD channel must be greater than or equal to 32 pixels.
- If the video layer of an HD device is zoomed in, the start position, width, and height included in the stRect attribute are the values before the video layer is zoomed in. After the video layer is zoomed in, the start position shifts and the width and height may increase by ratio.
- For the channel attribute stRect of the SD device, the channel width and channel height must be 2-pixel aligned.

[See Also]

HI_MPI_VO_SetChnAttr

VO_DISPLAY_FIELD_E

[Description]

Defines the field mode during video output.

[Syntax]

```
typedef enum hiVO_DISPLAY_FIELD_E
{
  VO_FIELD_TOP,     /*Top field*/
  VO_FIELD_BOTTOM,  /*Bottom field*/
  VO_FIELD_BOTH,    /*Top and bottom fields*/
  VO_FIELD_BUTT
} VO_DISPLAY_FIELD_E;
```

[Member]

Member	Description
VO_FIELD_TOP	Processes only the top field during video output.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
VO_FIELD_BOTTOM	Processes only the bottom field during video output.
VO_FIELD_BOTH	Processes both the bottom and top fields during video output.

[Note]

None

[See Also]

None

VO_QUERY_STATUS_S

[Description]

Defines the status of a VO channel.

[Syntax]

```
typedef struct hiVO_QUERY_STATUS_S
{
HI_U32 u32ChnBufUsed;      /* channel buffer that been occupied */
} VO_QUERY_STATUS_S;
```

[Member]

Member	Description
u32ChnBufUsed	Number of VBs occupied by the VO channel

[Note]

None

[See Also]

None

VO_ZOOM_RATIO_S

[Description]

Defines the partial zoom-in ratio.

[Syntax]

```
typedef struct hiVO_ZOOM_RATIO_S
{
    HI_U32 u32XRatio;
    HI_U32 u32YRatio;
    HI_U32 u32WRatio;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_U32 u32HRatio;
} VO_ZOOM_RATIO_S;
```

[Member]

Member	Description
u32XRatio	Based on the screen coordinates, the ratio of the horizontal coordinate of the start point of the area to be zoomed to the width of the display channel
u32YRatio	Based on the screen coordinates, the ratio of the vertical coordinate of the start point of the area to be zoomed to the height of the display channel
u32WRatio	Based on the screen coordinates, the ratio of the width of the area to be zoomed to the width of the display channel
u32HRatio	Based on the screen coordinates, the ratio of the height of the area to be zoomed to the height of the display channel

[Note]

- The values of the preceding members range from 0 to 1000.
- The member value is 1000 times of the ratio. For example, if the ratio of the width of the area to be zoomed to the width of the display channel is 0.6, the value of u32WRatio is 600.
- To disable the partial zoom-in function, you only need to set the values of the preceding members to 0.

[See Also]

HI_MPI_VO_SetZoomInWindow

VO_ZOOM_ATTR_S

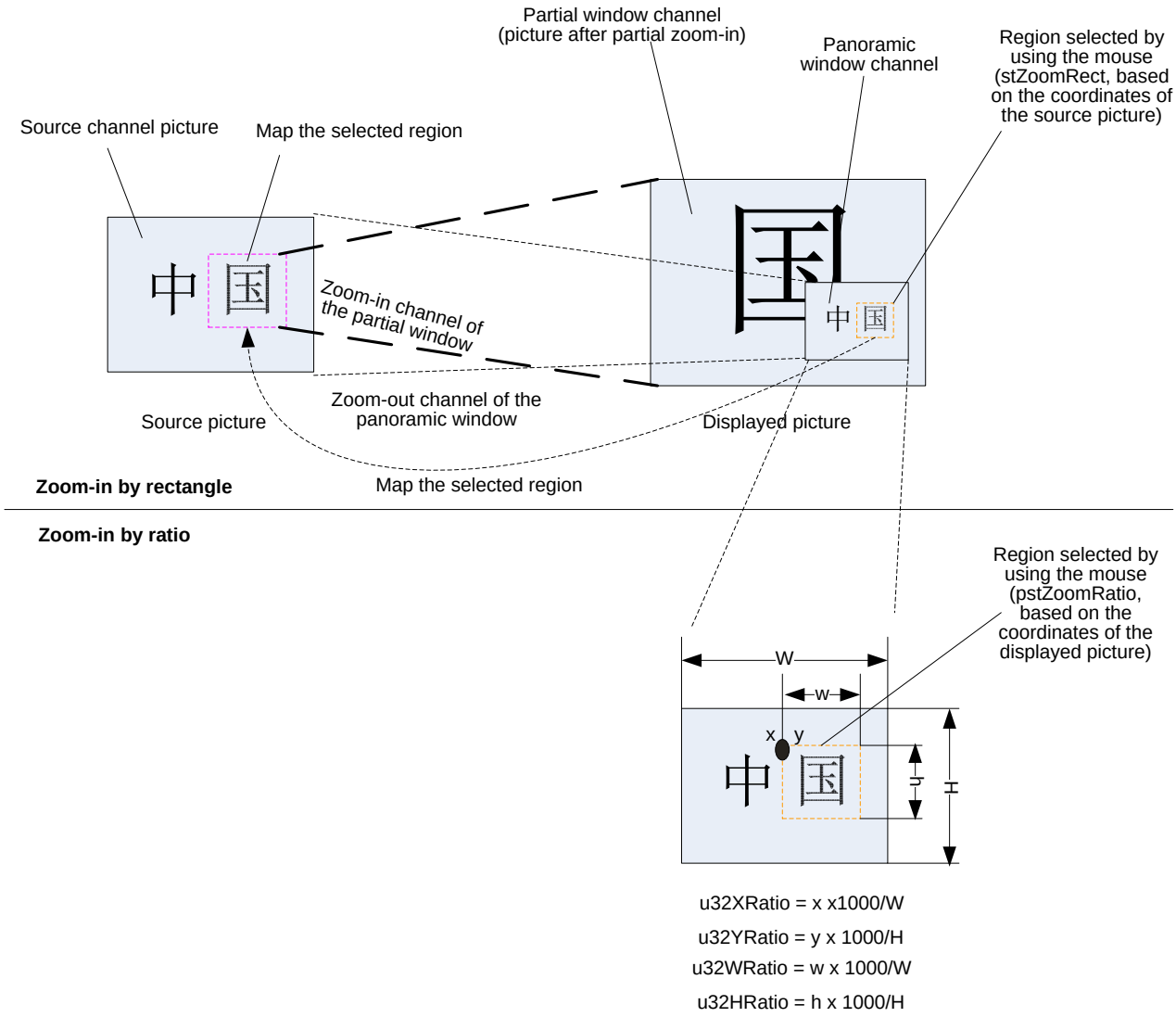
[Description]

Defines the partial zoom-in attributes.

Figure 4-2 shows the principle of partial zoom-in.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 4-2 Principle of partial zoom-in



[Syntax]

```
typedef struct hiVO_ZOOM_ATTR_S
{
    VOU_ZOOM_IN_E   enZoomType;        /*Choose the type of zoom in*/
    union
    {
        RECT_S         stZoomRect;     /*Zoom in by rectangle*/
        VO_ZOOM_RATIO_S stZoomRatio;    /*Zoom in by ratio*/
    };
} VO_ZOOM_ATTR_S;
```

[Member]

Member	Description
enZoomType	Partial zoom-in type
stZoomRect	Coordinates of the partial zoom-in window
stZoomRatio	Ratio of the partial zoom-in window

[Difference]

Chip	Description
Hi3531/Hi3532 /Hi3521/Hi3520A	For the SD device, its partial enlargement window size must be greater than or equal to 16x16. That is, the width or height of the window must be greater than or equal to 16 pixels.
Hi3518/Hi3516C	For the SD device, its partial enlargement window size must be greater than or equal to 32x32. That is, the width or height of the window must be greater than or equal to 32 pixels.

[Note]

- The source data of a VO channel can be clipped by rectangle or ratio during partial zoom-in.
- When the source data is clipped by rectangle, stZoomRect must be set to a non-negative number and 2-pixel-aligned. For the HD device, its partial enlargement window size must be greater than or equal to 32x32.
- When the source data is clipped by ratio, stZoomRatio needs to be configured. The value range of stZoomRatio is [0, 1000]. The value 1000 indicates the ratio of 1:1. The width, height, and coordinates of the partial enlargement window that are calculated by using the configured ratio and input picture resolution may be not 2-pixel-aligned. The SDK automatically adapts the window by using smaller values to ensure that the width, height, and coordinates of the window are 2-pixel-aligned. For the SD device, if the inputs are the pictures with various resolutions, ensure that the fields of vision of the windows that are obtained by calculating the large and small pictures by ratio are consistent. Otherwise, the screen flickers leftwards and rightwards
- The preceding two modes are based on the input screen rather the screen coordinates.
- To disable the partial zoom-in function, you can set stZoomRect or stZoomRatio to (0, 0, 0, 0).

[See Also]

HI_MPI_VO_SetZoomInWindow

VO_CSC_S

[Description]

Defines the picture output effect.

[Syntax]

```
typedef struct hiVO_CSC_S
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
{
    VO_CSC_MATRIX_E enCscMatrix;
    HI_U32 u32Luma;                 /*Luminance: 0-100*/
    HI_U32 u32Contrast;             /*Contrast: 0-100*/
    HI_U32 u32Hue;                  /*Hue: 0-100*/
    HI_U32 u32Satuature;            /*Saturation: 0-100*/
} VO_CSC_S;
```

[Member]

Member	Description
enCscMatrix	CSC matrix
u32Luma	Luminance Value range: [0, 100]
u32Contrast	Contrast Value range: [0, 100]
u32Hue	Hue Value range: [0, 100]
u32Satuature	Saturation Value range: [0, 100]

[Note]

None

[See Also]

HI_MPI_VO_SetDevCSC

VO_VGA_PARAM_S

[Description]

Defines the VGA picture output effect.

[Syntax]

```
typedef struct hiVO_VGA_PARAM_S
{
    VO_CSC_MATRIX_E enCscMatrix;
    HI_U32 u32Luma;                 /*Luminance: 0-100*/
    HI_U32 u32Contrast;             /*Contrast: 0-100*/
    HI_U32 u32Hue;                  /*Hue: 0-100*/
HI_U32 u32Satuature;            /*Saturation: 0-100*/
HI_U32 u32Gain;
} VO_VGA_PARAM_S;
```


Copyright © HiSilicon Technologies Co., Ltd

[Member]

Member	Description
enCscMatrix	CSC matrix
u32Luma	Luminance Value range: [0, 100]
u32Contrast	Contrast Value range: [0, 100]
u32Hue	Hue Value range: [0, 100]
u32Satuature	Saturation Value range: [0, 100]
u32Gain	DAC gain Value range: [0, 64)

[Note]

None

[See Also]

HI_MPI_VO_SetVgaParam

VO_CSC_MATRIX_E

[Description]

Defines the CSC matrix.

[Syntax]

```
typedef enum hiVO_CSC_MATRIX_E
{
    VO_CSC_MATRIX_IDENTITY = 0,
    VO_CSC_MATRIX_BT601_TO_BT709,
    VO_CSC_MATRIX_BT709_TO_BT601,
    VO_CSC_MATRIX_BT601_TO_RGB_PC,
VO_CSC_MATRIX_BT709_TO_RGB_PC,
VO_CSC_MATRIX_RGB_TO_BT601_PC,
VO_CSC_MATRIX_RGB_TO_BT709_PC,
    VO_CSC_MATRIX_BUTT
} VO_CSC_MATRIX_E;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
VO_CSC_MATRIX_IDENTITY	Identity matrix
VO_CSC_MATRIX_BT601_TO_BT709	CSC matrix from BT.601 to BT.709
VO_CSC_MATRIX_BT709_TO_BT601	CSC matrix from BT.709 to BT.601
VO_CSC_MATRIX_BT601_TO_RGB_PC	CSC matrix from BT.601 to RGB
VO_CSC_MATRIX_BT709_TO_RGB_PC	CSC matrix from BT.709 to RGB
VO_CSC_MATRIX_RGB_TO_BT601_PC	CSC matrix from RGB to BT.601
VO_CSC_MATRIX_RGB_TO_BT709_PC	CSC matrix from RGB to BT.709

[Note]

None

[See Also]

- HI_MPI_VO_SetDevCSC
- HI_MPI_VO_SetVgaParam

VO_WBC_ATTR_S

[Description]

Defines the WBC attributes.

[Syntax]

```
typedef struct hiVO_WBC_ATTR_S
{
SIZE_S                stTargetSize;
PIXEL_FORMAT_E        enPixelFormat;
HI_U32                u32FrameRate;
VO_WBC_DATASOURCE_E   enDataSource;
} VO_WBC_ATTR_S;
```

[Member]

Member	Description
stTargetSize	WBC picture size. The maximum size is 720x576, and the minimum size is 32x32. Dynamic attribute
enPixelFormat	Data format, SPYCbCr420 or SPYCbCr422 Dynamic attribute
u32FrameRate	Frame rate control Dynamic attribute

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
enDataSource	Writeback data source, VO_WBC_DATASOURCE_MIXER or VO_WBC_DATASOURCE_VIDEO This member is not supported.

[Note]

None

[See Also]

HI_MPI_VO_SetWbcAttr

VO_WBC_MODE_E

[Description]

Defines the video WBC mode.

[Syntax]

```
typedef enum hiVO_WBC_MODE_E
{
VO_WBC_MODE_NOMAL = 0,
VO_WBC_MODE_DROP_REPEAT,
VO_WBC_MODE_PROG_TO_INTL,

VO_WBC_MODE_BUTT,
} VO_WBC_MODE_E;
```

[Member]

Member	Description
VO_WBC_MODE_NOMAL	Normal video WBC mode. Only the device frame rate and WBC frame rate are used to control picture collection.
VO_WBC_MODE_DROP_REPEAT	Mode in which the repeated frames are dropped. For the high-definition device DHD, the WBC module writes back the repeated frames only once, drops the frame interrupt if the same frame is displayed for twice or more, and controls the frame rate based on the display frame rate of the video layer and the WBC frame rate. This mode is valid only when DHD is set to progressive display mode.
VO_WBC_MODE_PROG_TO_INTL	Mode in which progressive pictures are converted into interlaced pictures. For DHD, the WBC module converts the adjacent two frames in progressive display mode into an interlaced frame and drops the

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	frame interrupt if the same frame is displayed for three times or more. This mode is valid only when DHD is set to progressive display mode.

[Note]

- If the WBC mode is VO_WBC_MODE_PROG_TO_INTL, the HD display mode is 1080p@60 fps, and actual capture frame rate of the VI channel is 30 fps, the WBC module converts 60 progressive frames into 30 interlaced frames. This ensures that the display effect of the pictures after WBC processing is better. Therefore, when the HD display mode is 1080p@30 fps, the maximum WBC frame rate is 15 fps, but the configured frame rate does not take effect. The following describes the actual WBC frame rate:
  - If DispFrmRt is less than or equal to 1/2 x DevFrmRt, the actual WBC frame rate is DispFrmRt.
  - If DispFrmRt is equal to DevFrmRt, the actual WBC frame rate is 1/2 x DevFrmRt.
  - If DevFrmRt is greater than DispFrmRt and DispFrmRt is greater than 1/2 x DevFrmRt, the actual WBC frame rate varies according to dispFrmRt.
- You are advised to set DispFrmRt to 1/2 x DevFrmRt.
- If the WBC mode is set to VO_WBC_MODE_PROG_TO_INTL, an SPYCbCr422 picture supports at most 1/8 scaling (excluding 1/8 scaling), and an SPYCbCr420 picture supports at most 1/4 scaling (excluding 1/4 scaling).
- If the WBC mode is set to VO_WBC_MODE_DROP_REPEAT, the display mode of HD is 1080p@60 fps, and the actual capture frame rate of the VI channel is 30 fps, the WBC module writes back only the captured 30 frames and drops the frames that are repeatedly displayed on HD. That is, the maximum WBC frame rate is less than or equal to the actual capture frame rate of the VI channel. For example, if the display mode of HD is 1080p@60 fps and the actual capture frame rate of the VI channel is 10 fps, the maximum WBC frame rate is 10 fps.

[See Also]

HI_MPI_VO_SetWbcMode

VO_CAS_MODE_E

[Description]

Defines the cascade channel mode.

[Syntax]

```
typedef enum hiVO_CAS_MODE_E
{
    VO_CAS_MODE_SINGLE = 0,
    VO_CAS_MODE_DUAL,
    VO_CAS_MODE_BUTT,
} VO_CAS_MODE_E;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Member]

Member	Description
VO_CAS_MODE_SINGLE	Single-channel cascade mode
VO_CAS_MODE_DUAL	Dual-channel cascade mode

[Note]

None

[See Also]

None

VO_CAS_RGN_E

[Description]

Defines the cascade region mode.

[Syntax]

```
typedef enum hiVO_CAS_RGN_E
{
    VO_CAS_64_RGN = 0,
    VO_CAS_32_RGN,
    VO_CAS_RGN_BUTT,
} VO_CAS_RGN_E;
```

[Member]

Member	Description
VO_CAS_64_RGN	64-region mode
VO_CAS_32_RGN	32-region mode

[Note]

None

[See Also]

None

VO_CAS_ATTR_S

[Description]

Defines the cascade attribute.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
typedef struct hiVO_CAS_ATTR_S
{
    HI_BOOL        bSlave;
    VO_CAS_RGN_E    enCasRgn;
    VO_CAS_MODE_E   enCasMode;
} VO_CAS_ATTR_S;
```

[Member]

Member	Description
bSlave	Master/slave mode during video cascading. It is a static attribute.
enCasRgn	Region mode during video cascading. It is a static attribute.
enCasMode	Cascade channel mode during video cascading. It is a static attribute.

[Note]

None

[See Also]

HI_MPI_VO_SetCascadeAttr

4.5 Error Codes

Table 4-3 describes the error codes of VO APIs.

Table 4-3 Error codes of VO APIs

Error Code	Macro Definition	Description
0xA00F8001	HI_ERR_VO_INVALID_DEVID	The device ID exceeds the valid range.
0xA00F8002	HI_ERR_VO_INVALID_CHNID	The channel ID exceeds the valid range.
0xA00F8003	HI_ERR_VO_ILLEGAL_PARAM	The parameter value exceeds the valid range.
0xA00F8006	HI_ERR_VO_NULL_PTR	The pointer is null.
0xA00F8008	HI_ERR_VO_NOT_SUPPORT	The operation is not supported.
0xA00F8009	HI_ERR_VO_NOT_PERMIT	The operation is forbidden.
0xA00F800C	HI_ERR_VO_NO_MEM	The memory is insufficient.
0xA00F8010	HI_ERR_VO_SYS_NOTREADY	The system is not initialized.
0xA00F8012	HI_ERR_VO_BUSY	The resources are unavailable.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Macro Definition	Description
0xA00F8040	HI_ERR_VO_DEV_NOT_CONFIG	The VO device is not configured.
0xA00F8041	HI_ERR_VO_DEV_NOT_ENABLE	The VO device is not enabled.
0xA00F8042	HI_ERR_VO_DEV_HAS_ENABLED	The device has been enabled.
0xA00F8043	HI_ERR_VO_DEV_HAS_BINDED	The device has been bound.
0xA00F8044	HI_ERR_VO_DEV_NOT_BINDED	The device is not bound.
0xA00F8045	HI_ERR_VO_VIDEO_NOT_ENABLE	The video layer is not enabled.
0xA00F8046	HI_ERR_VO_VIDEO_NOT_DISABLE	The video layer is not disabled.
0xA00F8047	HI_ERR_VO_VIDEO_NOT_CONFIG	The video layer is not configured.
0xA00F8048	HI_ERR_VO_CHN_NOT_DISABLE	The VO channel is not disabled.
0xA00F8049	HI_ERR_VO_CHN_NOT_ENABLE	No VO channel is enabled.
0xA00F804A	HI_ERR_VO_CHN_NOT_CONFIG	The VO channel is not configured.
0xA00F804B	HI_ERR_VO_CHN_NOT_ALLOC	No VO channel is allocated.
0xA00F804C	HI_ERR_VO_INVALID_PATTERN	The pattern is invalid.
0xA00F804D	HI_ERR_VO_INVALID_POSITION	The cascade position is invalid.
0xA00F804E	HI_ERR_VO_WAIT_TIMEOUT	The waiting times out.
0xA00F804F	HI_ERR_VO_INVALID_VFRAME	The video frame is invalid.
0xA00F8050	HI_ERR_VO_INVALID_RECT_PARA	The rectangle parameter is invalid.
0xA00F8051	HI_ERR_VO_SETBEGIN_ALREADY	The BEGIN has been configured.
0xA00F8052	HI_ERR_VO_SETBEGIN_NOTYET	The BEGIN is not configured.
0xA00F8053	HI_ERR_VO_SETEND_ALREADY	The END has been configured.
0xA00F8054	HI_ERR_VO_SETEND_NOTYET	The END is not configured.
0xA00F8055	HI_ERR_VO_GRP_INVALID_ID	The sync group ID is invalid.
0xA00F8056	HI_ERR_VO_GRP_NOT_CREATE	No sync group is created.
0xA00F8057	HI_ERR_VO_GRP_HAS_CREATED	The sync group has been created.
0xA00F8058	HI_ERR_VO_GRP_NOT_DESTROY	The sync group is not destroyed.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Macro Definition	Description
0xA00F8059	HI_ERR_VO_GRP_CHN_FULL	The maximum number of channels that can be registered in the sync group is reached.
0xA00F805A	HI_ERR_VO_GRP_CHN_EMPTY	No channel is registered in the sync group.
0xA00F805B	HI_ERR_VO_GRP_CHN_NOT_EMPTY	The channel of the sync group is not empty.
0xA00F805C	HI_ERR_VO_GRP_INVALID_SYN_MODE	The sync mode is invalid.
0xA00F805D	HI_ERR_VO_GRP_INVALID_BASE_PTS	The base PTS is invalid.
0xA00F805E	HI_ERR_VO_GRP_NOT_START	No sync group is started.
0xA00F805F	HI_ERR_VO_GRP_NOT_STOP	The sync group is not stopped.
0xA00F8060	HI_ERR_VO_GRP_INVALID_FRMRATE	The frame rate of the sync group is invalid.
0xA00F8061	HI_ERR_VO_GRP_CHN_HAS_REG	The channel has been registered.
0xA00F8062	HI_ERR_VO_GRP_CHN_NOT_REG	No channel is registered.
0xA00F8063	HI_ERR_VO_GRP_CHN_NOT_UNREG	The channel is unregistered.
0xA00F8064	HI_ERR_VO_GRP_BASE_NOT_CFG	No base PTS is configured.
0xA00F8065	HI_ERR_VO_GFX_NOT_DISABLE	The graphics layer is not disabled.
0xA00F8066	HI_ERR_VO_GFX_NOT_BIND	The graphics layer is not bound.
0xA00F8067	HI_ERR_VO_GFX_NOT_UNBIND	The graphics layer is still bound.
0xA00F8068	HI_ERR_VO_GFX_INVALID_ID	The graphics layer ID exceeds the range.
0xA00F8069	HI_ERR_VO_WBC_NOT_DISABLE	WBC is not disabled.
0xA00F806a	HI_ERR_VO_WBC_NOT_CONFIG	WBC attributes are not set.
0xA00F806b	HI_ERR_VO_CHN_AREA_OVERLAP	The VO channel areas overlap.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Contents

5 VPSS 5-1
5.1 Overview 5-1
5.2 Functions 5-1
5.2.1 Concepts 5-1
5.2.2 Function Description 5-3
5.2.3 Differences Among Chips 5-3
5.3 API Reference 5-10
5.4 Data Types 5-62
5.5 Error Codes 5-85

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figures

Figure 5-1 Context relationship of the VPSS 5-3
Figure 5-2 Data processing for the Hi3531/Hi3532 5-4
Figure 5-3 Data processing for the Hi3521/Hi3520A 5-5
Figure 5-4 Data processing for the Hi3518/Hi3516C 5-5
Figure 5-5 Data processing for the Hi3520D/Hi3515A/Hi3515C 5-6

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Tables

Table 5-1 VPSS channel specifications for the Hi3531/Hi3532 5-6
Table 5-2 VPSS channel specifications for the Hi3521/Hi3520A 5-7
Table 5-3 VPSS channel specifications for the Hi3518/Hi3516C 5-8
Table 5-4 VPSS channel specifications for the Hi3520D/Hi3515A/Hi3515C 5-9
Table 5-5 Error codes of VPSS APIs 5-85

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

5 VPSS

5.1 Overview

Video process subsystem (VPSS) is a video preprocessing unit that preprocesses an input picture (such as denoising and deinterlacing), processes the picture in each channel separately (such as scaling and sharpening), and then outputs multiple pictures in various resolutions.

The VPSS supports various picture processing functions including prescaling, de-ring (Dr)/de-blocking (Db), noise reduction (NR), image enhancement (IE), deinterlacing (DEI), and sharpen.

5.2 Functions

5.2.1 Concepts

Note the following:

- Group

  The VPSS provides the group concept for users. At most 128 groups are supported, and the VPSS hardware is multiplexed by groups in time-division-multiplexing (TDM) mode. Each group has multiple channels. The number of channels varies according to solution implementation. For details, see the channel description.

- Channel

  Channels are classified into physical channels, bypass channels, and extended channels. The VPSS hardware provides multiple physical channels. The channels have different functions (see Figure 5-1) and limitations. The VPSS also provides one bypass channel that supports only cropping. If video preprocessing is not required for a picture, it can be transparently transmitted to the receiver over the bypass channel. Extended channels support scaling. They use physical channels as inputs after being bound to physical channels, scale pictures based on target resolution configured by users, and then transmit the scaled pictures. Currently, only the Hi3518/Hi3516C supports extended channels. For details about channel specifications differences among chips, see section 5.2.3 "Differences Among Chips."

- Prescale

  It is pre-scaling for short. If the size of an input picture is greater than the preset size, the picture is scaled to the preset size.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Sizer

  It is a function that enables the VPSS to filter out pictures with specific sizes for further processing.

- De-ring/De-block

  De-ring effect is also called de-ringing effect. It compensates low-frequency component loss caused by video compression and removes the ringing at the edge of the pictures. De-block effect is also called de-blocking effect. It compensates DC component quantization loss caused by video compression by block.

- DEI

  Deinterlacing function. The de-interlace module restores the interlaced video source to the progressive video source.

- NR

  Noise reduction function. Gaussian noise can be removed from the pictures by configuring the parameters. The pictures become smooth and the encoding bit rate also can be reduced.

- IE

  Image enhancement function. The IE module can distinguish the detail areas of the graphics, enhance the graphics details. Therefore, the graphics become clearer and the contrast of the graphics is enhanced.

- Sharpen

  Graphics sharpening function: The sharpening module sharps the edge of the graphics and enhances the graphics details. The sharpening module also compensates or enhances the frequency of the scaled graphics. Therefore, the edges of the graphics are sharpened and the outline of the graphics becomes clear.

- FRC

  Frame rate control

- Backup node

  Backup node for an original picture. Each group has a backup node that is used for backing up an original picture to be processed by hardware. The VPSS stores the picture for the head node of the buffer queue in the backup node in the following situations:

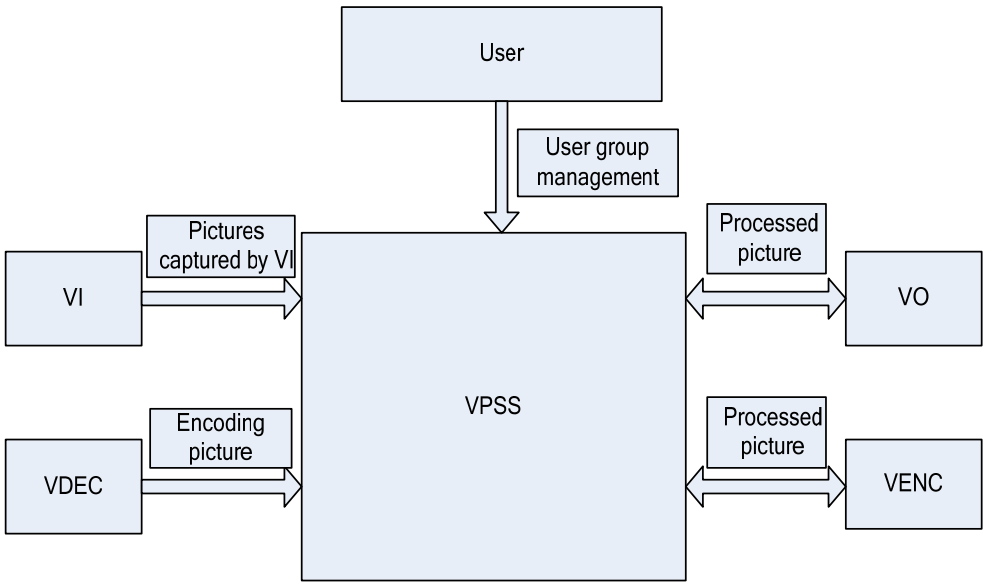
  - If the picture for the head node of the buffer queue needs to be processed by VPSS hardware, the VPSS stores this picture in the backup node to replace the original picture.
  - If the receiving module bound to the back end requires that the VPSS store the picture for the head node of the buffer queue in the backup node, the VPSS replaces the picture in the backup node even if this picture is not processed by hardware.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

5.2.2 Function Description

Figure 5-1 shows the position of the VPSS in the system.

Figure 5-1 Context relationship of the VPSS



The VPSS can be bound to the VI, VDEC, VO, and VENC modules by calling the binding interface of the SYS module. The VI and VDEC modules are input sources and the VO and VENC modules are receivers. You can manage the groups by calling media processing platform programming interfaces (MPIs). Each group can be bound only to one input source. The physical channels of each group support the automatic or user working mode. The working mode can be dynamically switched. By default, the automatic mode is used. In this mode, each channel can be bound only to one receiver. If you want to select the user mode, you need to call the related MPI and specify the size and format of the required picture. Note that the user mode is used when the pictures from a channel are encoded in multiple formats. In user mode, play control does not work.

CAUTION

When the VPSS channel is bound to the VENC module in automatic mode and the resolution of the source picture received by the VPSS channel is smaller than that of the picture to be encoded, the VENC module does not encode this picture even the VPSS channel supports the zoom-in function. If you want to zoom in on the picture and then encode it, set the mode of the VPSS channel to user mode.

5.2.3 Differences Among Chips

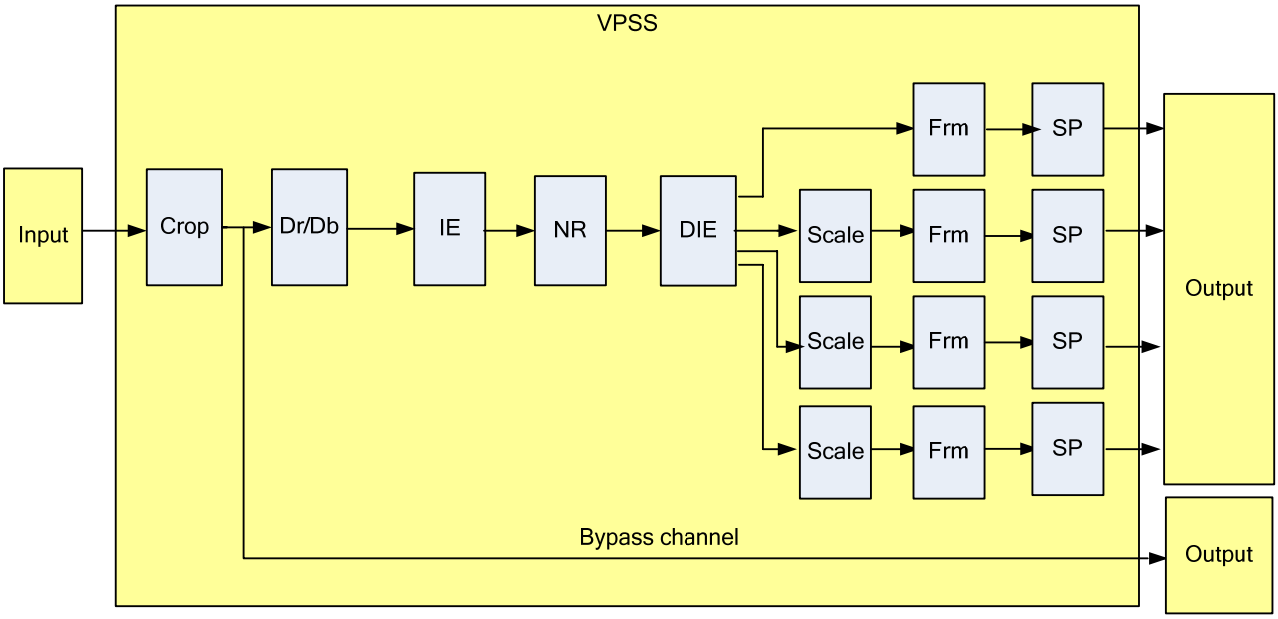
Compared with the Hi3521/Hi3520A, the Hi3531/Hi3532 processes data in a different way. See Figure 5-2 and Figure 5-3. The Hi3521/Hi3520A provides the pre-scaling function but does not support Dr/Db. The sizer and RFC functions are added to the Hi3520D/Hi3515A/Hi3515C and Hi3521 before pre-scaling, as shown in Figure 5-5. For the

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Hi3531/Hi3532, Dr/Db and NR or DEI cannot be enabled at the same time. You are advised to enable Dr/Db when processing decoded pictures and enable NR and DEI as required when processing the pictures captured by VI channels. For the Hi3521/Hi3520A, chn0/chn1 supports scaling and allows you to set the field mode (single-field or dual-field). However, chn0 does not support SP. The other four physical channels allow you to set the NR strength. The field mode (single-field or dual-field) selection function is added to chn0/chn1 of Hi3521/Hi3520A, but the sp function of chn0 is deleted. In addition, the four physical channels allow you to set the NR strength respectively. The channels of the Hi3520D/Hi3515A/Hi3515C are optimized. Four physical channels are the same and support the scaling and sharpening functions.

Compared with the Hi3521/Hi3520A, the Hi3518/Hi3516C does not support the IE and pre-scaling functions. As shown in Figure 5-4, after being processed by the NR and DEI modules, the input pictures are transmitted to the Scale and Sharpen modules respectively. Then target pictures are obtained. In addition, compared with the Hi3521/Hi3520A, the Hi3518/Hi3516C does not allow you to select single-field or dual-field mode. The functions of physical channels 0 and 1 are different. Channel 0 does not support the scaling function. The Hi3518/Hi3516C provides five extended channels that support scaling and frame rate control without the intervention of other hardware algorithm units. The binding relationships between extended channels and physical channels shown in Figure 5-4 are for reference. You can also customize the binding relationships.

Figure 5-2 Data processing for the Hi3531/Hi3532



HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 5-3 Data processing for the Hi3521/Hi3520A

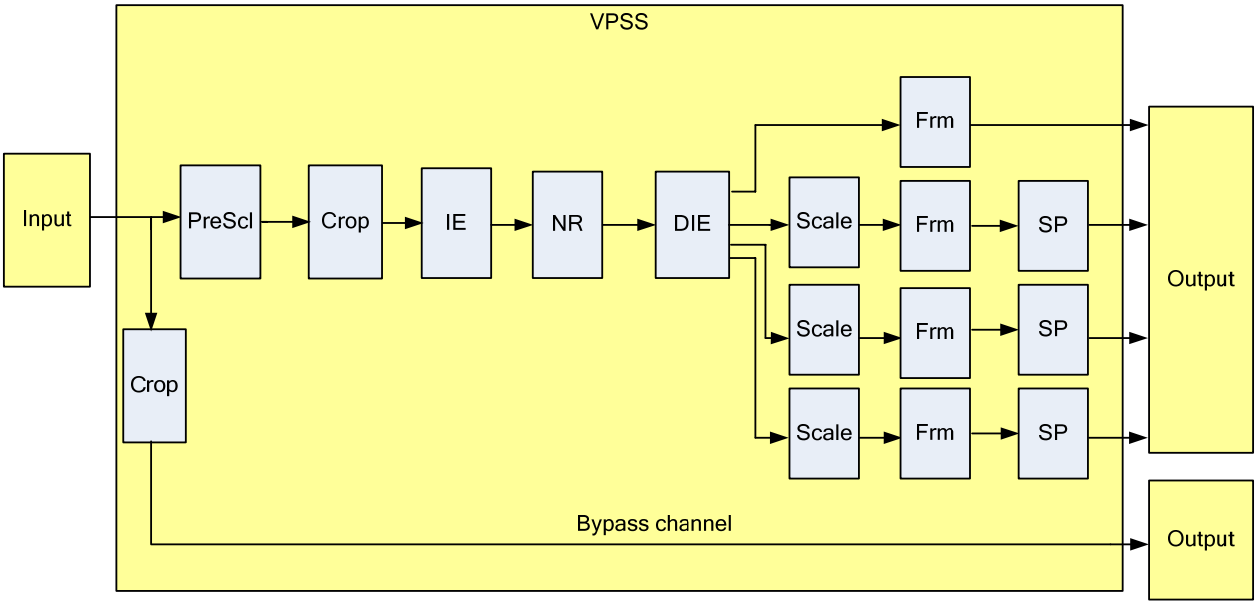
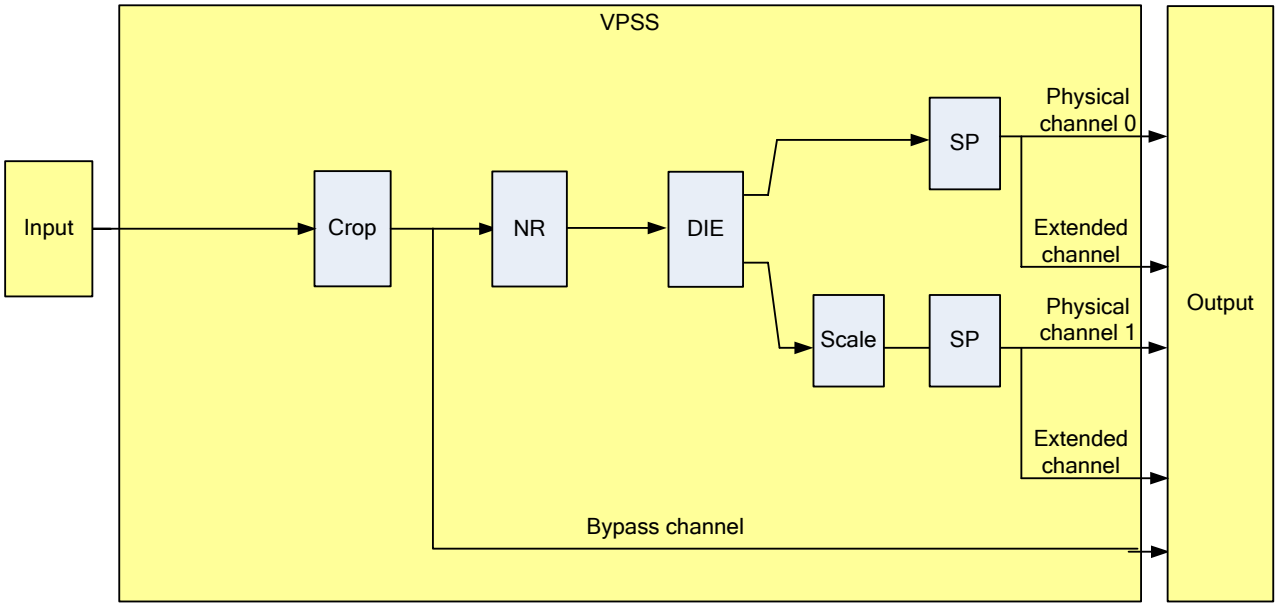


Figure 5-4 Data processing for the Hi3518/Hi3516C



Copyright © HiSilicon Technologies Co., Ltd

Figure 5-5 Data processing for the Hi3520D/Hi3515A/Hi3515C

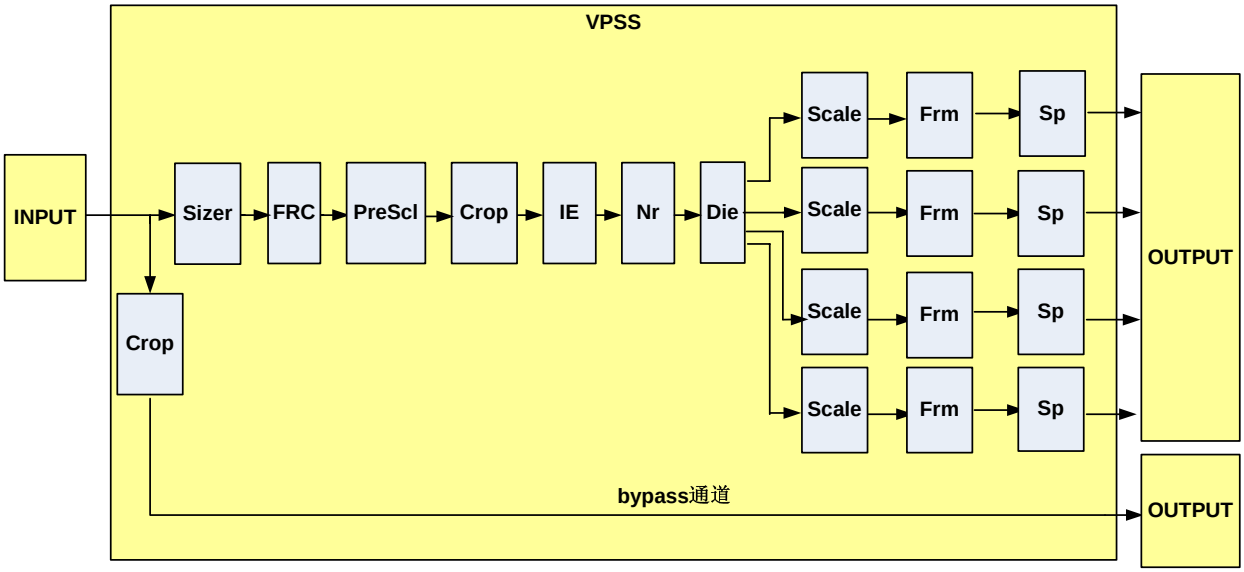


Table 5-1 VPSS channel specifications for the Hi3531/Hi3532

Channel ID	Channel	Scaling	Sharpen	Captured Field Select	Recommended Application Scenario
0	Channel 0	Not supported	Supported	Not supported	Large stream encoding, JPEG snapshot
1	Channel 1	At most zoom in or zoom out 16 times in both horizontal and vertical directions	Supported	Not supported	Small stream encoding
2	Channel 2	At most zoom in or zoom out 16 times in both horizontal and vertical directions Frame/field conversion supported	Supported	Not supported	High-definition (HD) preview
3	Channel 3	At most zoom in or zoom out 16 times in both horizontal and vertical directions	Supported	Not supported	HD preview

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Channel ID	Channel	Scaling	Sharpen	Captured Field Select	Recommended Application Scenario
4	Bypass channel	Not supported	Not supported	Not supported	D1 output required by the CVBS, MD, and IVE

Table 5-2 VPSS channel specifications for the Hi3521/Hi3520A

Channel ID	Channel	Scaling	Sharpen	Captured Field Select	Recommended Application Scenario
0	Channel 0	Not supported	Not supported	Supported	Large stream encoding, JPEG snapshot
1	Channel 1	At most zoom in 16 times or zoom out eight times in both horizontal and vertical directions	Supported	Supported	Small stream encoding
2	Channel 2	At most zoom in 16 times or zoom out eight times in both horizontal and vertical directions Frame/field conversion supported	Supported	Not supported	HD preview
3	Channel 3	At most zoom in 16 times or zoom out eight times in both horizontal and vertical directions	Supported	Not supported	HD preview
4	Bypass channel	Not supported	Not supported	Not supported	D1 output required by the CVBS, MD, and IVE

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

For channel 0, channel 1, channel 2, and channel 3, the application scenarios are the recommended schemes. You can allocate the functions of each channel as required. Note that only channel 2 supports frame/field conversion. If the conversion function is used, channel 3 has no output.

Table 5-3 VPSS channel specifications for the Hi3518/Hi3516C

Channel ID	Channel	Scaling	Sharpen	Captured Field Select	Recommended Application Scenario
0	Channel 0	Not supported	Supported	Not supported	Large stream encoding, JPEG snapshot
1	Channel 1	At most zoom in or zoom out eight times in both horizontal and vertical directions	Supported	Not supported	Small stream encoding, CVBS preview
2	Bypass channel	Not supported	Not supported	Not supported	JEPG snapshot
3	Extended channel 0	At most zoom out 15 times in both horizontal and vertical directions No limitations on zoom in	Not supported	Not supported	Small stream encoding or target picture output required by the MD and IVE
4	Extended channel 1	At most zoom out 15 times in both horizontal and vertical directions No limitations on zoom in	Not supported	Not supported	Small stream encoding or target picture output required by the MD and IVE
5	Extended channel 2	At most zoom out 15 times in both horizontal and vertical directions No limitations on zoom in	Not supported	Not supported	Small stream encoding or target picture output required by the MD and IVE
6	Extended channel 3	At most zoom out 15 times in both horizontal and vertical directions	Not supported	Not supported	Small stream encoding or target picture output required by the MD and IVE

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Channel ID	Channel	Scaling	Sharpen	Captured Field Select	Recommended Application Scenario
		No limitations on zoom in			
7	Extended channel 4	At most zoom out 15 times in both horizontal and vertical directions No limitations on zoom in	Not supported	Not supported	Small stream encoding or target picture output required by the MD and IVE

For the extended channels 0−4 of the Hi3518/Hi3516C, if the input data format is YUV422SP (the output data format of the bound physical channels is YUV422SP) and the output data format is YUV420SP, the extended channels support at most zoom out eight times in the vertical direction.

Table 5-4 VPSS channel specifications for the Hi3520D/Hi3515A/Hi3515C

Channel ID	Channel	Scaling	Sharpen	Captured Field Select	Recommended Application Scenario
0	Channel 0	At most zoom out eight times in both horizontal and vertical directions At most zoom in 16 times in both horizontal and vertical directions	Supported	Not supported	Large stream encoding, JPEG snapshot.
1	Channel 1	At most zoom out eight times in both horizontal and vertical directions At most zoom in 16 times in both horizontal and vertical directions	Supported	Not supported	Small stream encoding
2	Channel 2	At most zoom out eight times in both	Supported	Not supported	HD preview

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Channel ID	Channel	Scaling	Sharpen	Captured Field Select	Recommended Application Scenario
		horizontal and vertical directions At most zoom in 16 times in both horizontal and vertical directions Frame/Field conversion is supported.			
3	Channel 3	At most zoom out eight times in both horizontal and vertical directions At most zoom in 16 times in both horizontal and vertical directions	Supported	Not supported	HD preview
4	Bypass channel	Not supported	Not supported	Not supported	D1 picture output required by the CVBS, MD, and IVE

[Note]

For channel 0 to channel 3, the application scenarios are the recommended schemes. You can allocate the functions of each channel as required. Note that only channel 2 supports frame/field conversion. If the conversion function is used, channel 3 has no output.

5.3 API Reference

The VPSS provides the following MPIs:

- HI_MPI_VPSS_CreateGrp: Creates a VPSS group.
- HI_MPI_VPSS_DestroyGrp: Destroys a VPSS group.
- HI_MPI_VPSS_GetGrpAttr: Obtains VPSS group attributes.
- HI_MPI_VPSS_SetGrpAttr: Sets VPSS group attributes.
- HI_MPI_VPSS_StartGrp: Enables a VPSS group.
- HI_MPI_VPSS_StopGrp: Disables a VPSS group.
- HI_MPI_VPSS_ResetGrp: Resets a VPSS group.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_VPSS_EnableChn: Enables a VPSS channel.
- HI_MPI_VPSS_DisableChn: Disables a VPSS channel.
- HI_MPI_VPSS_EnableBackupFrame: Enables backup frames.
- HI_MPI_VPSS_DisableBackupFrame: Disables backup frames.
- HI_MPI_VPSS_GetChnAttr: Obtains the attributes of a VPSS channel.
- HI_MPI_VPSS_SetChnAttr: Sets the attributes of a VPSS channel.
- HI_MPI_VPSS_SetGrpParam: Sets the advanced parameters of a VPSS group.
- HI_MPI_VPSS_GetGrpParam: Obtains the advanced parameters of a VPSS group.
- HI_MPI_VPSS_SetCropCfg: Sets the VPSS crop attributes.
- HI_MPI_VPSS_GetCropCfg: Obtains the VPSS crop attributes.
- HI_MPI_VPSS_UserSendFrame: Transmits data to a VPSS group.
- HI_MPI_VPSS_UserSendFrameTimeout: Sends data to a VPSS group in timeout mode.
- HI_MPI_VPSS_SetChnMode: Sets the working mode of a VPSS channel.
- HI_MPI_VPSS_GetChnMode: Obtains the working mode of a VPSS channel.
- HI_MPI_VPSS_SetDepth: Sets the depth of the user picture queue.
- HI_MPI_VPSS_GetDepth: Obtains the depth of the user picture queue.
- HI_MPI_VPSS_UserGetFrame: Obtains a processed frame from a channel.
- HI_MPI_VPSS_UserReleaseFrame: Releases the buffer for storing a frame.
- HI_MPI_VPSS_UserGetGrpFrame: Obtains a source frame from a group.
- HI_MPI_VPSS_UserReleaseGrpFrame: Releases the buffer for storing a frame.
- HI_MPI_VPSS_SetChnNrParam: Sets the advanced NR attribute of a channel.
- HI_MPI_VPSS_GetChnNrParam: Obtains the advanced NR attribute of a channel.
- HI_MPI_VPSS_SetChnSpParam: Sets the advanced SP attribute of a channel.
- HI_MPI_VPSS_GetChnSpParam: Obtains the advanced SP attribute of a channel.
- HI_MPI_VPSS_SetPreScale: Sets the prescaling attribute of a VPSS group.
- HI_MPI_VPSS_GetPreScale: Obtains the prescaling attribute of a VPSS group.
- HI_MPI_VPSS_SetChnField: Sets the captured field attribute of a channel.
- HI_MPI_VPSS_GetChnField: Obtains the captured field attribute of a channel.
- HI_MPI_VPSS_SetGrpField: Sets the captured field attributes of a group.
- HI_MPI_VPSS_GetGrpField: Obtains the captured field attributes of a group.
- HI_MPI_VPSS_SetGrpSizer: Sets the picture size for the VPSS to filter out pictures.
- HI_MPI_VPSS_GetGrpSizer: Obtains the picture size for the VPSS to filter out pictures.
- HI_MPI_VPSS_SetGrpFrameRate: Sets the frame rate control information for the VPSS.
- HI_MPI_VPSS_GetGrpFrameRate: Obtains the frame rate control information for the VPSS.
- HI_MPI_VPSS_SetDelay: Sets the length of the delay queue that is used to delay starting VPSS processing.
- HI_MPI_VPSS_GetDelay: Obtains the length of the delay queue that is used to delay starting VPSS processing.
- HI_MPI_VPSS_SetExtChnAttr: Sets the attribute of an extended VPSS channel.
- HI_MPI_VPSS_GetExtChnAttr: Obtains the attribute of an extended VPSS channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VPSS_CreateGrp

[Description]

Creates a VPSS group.

[Syntax]

```
HI_S32 HI_MPI_VPSS_CreateGrp(VPSS_GRP VpssGrp, VpssGrp, VPSS_GRP_PARAM_S
*pstVpssGrpAttr)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstVpssGrpAttr	Pointer to the VPSS group attributes	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- Multiple groups can be created and the maximum number of groups is VPSS_MAX_GRP_NUM.
- A group can be created repeatedly.
- The attributes of the groups must be valid. For details, see VPSS_GRP_ATTR_S.
- Assume that [w, 64] is the 64-byte-aligned length. For the VPSS group with the width of W and the height of H, as long as the [w, 64] x h of the input picture is not greater than [W, 64] x H, the VPSS can process the picture.
- If the VPSS enables IE or the scaling function, the pixel value range of the output picture is [16, 235].

[Example]

```
VPSS_GRP_ATTR_S stGrpVpssAttr;
VPSS_CHN_ATTR_S stChnAttr;
VPSS_CROP_INFO_S stCropInfo;
HI_S32 s32Ret = HI_SUCCESS;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
VPSS_GRP VpssGrp;
VPSS_CHN VpssChn

VpssGrp = 0;
VpssChn = 0;

stGrpVpssAttr.u32MaxW = 720;
stGrpVpssAttr.u32MaxH = 576;
stGrpVpssAttr.bDrEn = HI_FALSE;
stGrpVpssAttr.bDbEn = HI_FALSE;
stGrpVpssAttr.bIeEn = HI_FALSE;
stGrpVpssAttr.bNrEn = HI_FALSE;
stGrpVpssAttr.bHistEn = HI_FALSE;
stGrpVpssAttr.enDieMode = VPSS_DIE_MODE_NODIE;
stGrpVpssAttr.enPixFmt = PIXEL_FORMAT_YUV_SEMIPLANAR_422;

s32Ret = HI_MPI_VPSS_CreateGrp(VpssGrp, &stGrpVpssAttr);
if(s32Ret != HI_SUCCESS)
{
    return s32Ret;
}

s32Ret = HI_MPI_VPSS_GetGrpAttr(VpssGrp, &stGrpVpssAttr);
if(s32Ret != HI_SUCCESS)
{
    return s32Ret;
}

stGrpVpssAttr.bIeEn = HI_TRUE;
stGrpVpssAttr.bNrEn = HI_TRUE;
s32Ret = HI_MPI_VPSS_SetGrpAttr(VpssGrp, &stGrpVpssAttr);
if(s32Ret != HI_SUCCESS)
{
    return s32Ret;
}

s32Ret = HI_MPI_VPSS_GetCropCfg(VpssGrp, &stCropInfo);
if(s32Ret != HI_SUCCESS)
{
return s32Ret;
}
stCropInfo.bEnable = 1;
stCropInfo.enCropCoordinate = VPSS_CROP_ABS_COOR;
stCropInfo.stCropRect.s32X = 180;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
stCropInfo.stCropRect.s32Y = 252;
stCropInfo.stCropRect.u32Width = 1920;
stCropInfo.stCropRect.u32Height = 1080;
s32Ret = HI_MPI_VPSS_SetCropCfg(VpssGrp, &stCropInfo);
if(s32Ret != HI_SUCCESS)
{
return s32Ret;
}

s32Ret = HI_MPI_VPSS_GetChnAttr(VpssGrp, VpssChn,&stChnAttr);
if(s32Ret != HI_SUCCESS)
{
    return s32Ret;
}

stChnAttr.bFrameEn = 0;
stChnAttr.bSpEn = 1;
s32Ret = HI_MPI_VPSS_SetChnAttr(VpssGrp, VpssChn,&stChnAttr);
if(s32Ret != HI_SUCCESS)
{
    return s32Ret;
}

s32Ret = HI_MPI_VPSS_EnableChn(VpssGrp, VpssChn);
if(s32Ret != HI_SUCCESS)
{
    return s32Ret;
}

s32Ret = HI_MPI_VPSS_StartGrp (VpssGrp);
if(s32Ret != HI_SUCCESS)
{
    return s32Ret;
}

/*****************************/
/*     call sys bind interface    */
/*****************************/

s32Ret = HI_MPI_VPSS_StopGrp (VpssGrp);
if(s32Ret != HI_SUCCESS)
{
    return s32Ret;
}
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
s32Ret = HI_MPI_VPSS_DisableChn(VpssGrp, VpssChn);
if(s32Ret != HI_SUCCESS)
{
return s32Ret;
}
/*****************************/
/*     call sys unbind interface*/
/*****************************/
s32Ret = HI_MPI_VPSS_DestroyGrp(VpssGrp);
if(s32Ret != HI_SUCCESS)
{
return s32Ret;
}
```

[See Also]

HI_MPI_VPSS_DestroyGrp

HI_MPI_VPSS_DestroyGrp

[Description]

Destroys a VPSS group.

[Syntax]

```
HI_S32 HI_MPI_VPSS_DestroyGrp(VPSS_GRP VpssGrp)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- You must disable this group by calling HI_MPI_VPSS_StopGrp before calling HI_MPI_VPSS_DestroyGrp.
- After this MPI is called, the VPSS group is destroyed only after the current task is completed.

[Example]

For details, see the description of HI_MPI_VPSS_CreateGrp.

[See Also]

HI_MPI_VPSS_CreateGrp

HI_MPI_VPSS_GetGrpAttr

[Description]

Obtains VPSS group attributes.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetGrpAttr (VPSS_GRP VpssGrp, VPSS_GRP_ATTR_S
*pstVpssGrpAttr)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstVpssGrpAttr	Pointer to the VPSS group attributes	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Before calling this MPI, you must create a group.
- The group attributes must be valid, and some static attributes cannot be set dynamically. For details, see VPSS_GRP_ATTR_S.

[Example]

For details, see the description of HI_MPI_VPSS_CreateGrp.

[See Also]

HI_MPI_VPSS_SetGrpAttr

HI_MPI_VPSS_SetGrpAttr

[Description]

Sets VPSS group attributes.

[Syntax]

```
HI_S32 HI_MPI_VPSS_SetGrpAttr (VPSS_GRP VpssGrp, VPSS_GRP_ATTR_S
*pstVpssGrpAttr)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstVpssGrpAttr	Pointer to the VPSS group attributes	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The group attributes must be valid, and some static attributes cannot be set dynamically. For details, see VPSS_GRP_ATTR_S.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

For details, see the description of HI_MPI_VPSS_CreateGrp.

[See Also]

HI_MPI_VPSS_GetGrpAttr

HI_MPI_VPSS_StartGrp

[Description]

Starts a VPSS group.

[Syntax]

```
HI_S32 HI_MPI_VPSS_StartGrp(VPSS_GRP VpssGrp)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.

[Example]

For details, see the description of HI_MPI_VPSS_CreateGrp.

[See Also]

HI_MPI_VPSS_StopGrp

HI_MPI_VPSS_StopGrp

[Description]

Disables a VPSS group.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_VPSS_StopGrp(VPSS_GRP VpssGrp)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.

[Example]

For details, see the description of HI_MPI_VPSS_CreateGrp.

[See Also]

HI_MPI_VPSS_StartGrp

HI_MPI_VPSS_ResetGrp

[Description]

Resets a VPSS group.

[Syntax]

```
HI_S32 HI_MPI_VPSS_ResetGrp (VPSS_GRP VpssGrp)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header file: mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.

[Example]

For details, see HI_MPI_VPSS_CreateGrp.

[See Also]

None

HI_MPI_VPSS_GetChnAttr

[Description]

Obtains the attributes of a VPSS channel.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetChnAttr(VPSS_GRP VpssGrp, VPSS_CHN VpssChn,
VPSS_CHN_ATTR_S *pstChnAttr)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_CHN_NUM)	Input
pstChnAttr	VPSS channel attributes	Output

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The channel ID must fall within [0, VPSS_MAX_CHN_NUM).
- This MPI is not available for bypass channels and extended channels. If this MPI is called when a bypass channel or an extended channel is used, the operation has no effect but no code indicating failure is returned.

[Example]

For details, see the description of HI_MPI_VPSS_CreateGrp.

[See Also]

HI_MPI_VPSS_SetChnAttr

HI_MPI_VPSS_SetChnAttr

[Description]

Sets VPSS channel attributes.

[Syntax]

```
HI_S32 HI_MPI_VPSS_SetChnAttr(VPSS_GRP VpssGrp, VPSS_CHN VpssChn,
VPSS_CHN_ATTR_S *pstChnAttr)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_CHN_NUM)	Input
pstChnAttr	VPSS channel attributes	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The channel ID must fall within [0, VPSS_MAX_CHN_NUM).
- This MPI is not available for bypass channels and extended channels. If this MPI is called when a bypass channel or an extended channel is used, the operation has no effect but no code indicating failure is returned.

[Example]

For details, see the description of HI_MPI_VPSS_CreateGrp.

[See Also]

HI_MPI_VPSS_GetChnAttr

HI_MPI_VPSS_EnableChn

[Description]

Enables a VPSS channel.

[Syntax]

```
HI_S32 HI_MPI_VPSS_EnableChn(VPSS_GRP VpssGrp, VPSS_CHN VpssChn)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_CHN_NUM)	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The channel ID must fall within [0, VPSS_MAX_CHN_NUM).
- Before a physical channel is bound to an extended channel, the physical channel must be enabled. Otherwise, an error code is returned.

[Example]

For details, see the description of HI_MPI_VPSS_CreateGrp.

[See Also]

HI_MPI_VPSS_DisableChn

HI_MPI_VPSS_DisableChn

[Description]

Disables a VPSS channel.

[Syntax]

```
HI_S32 HI_MPI_VPSS_DisableChn(VPSS_GRP VpssGrp, VPSS_CHN VpssChn)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The channel ID must fall within [0, VPSS_MAX_CHN_NUM).
- If extended channels are supported, ensure that the extended channels that are bound to a channel to be disabled are disabled before you disable the channel. Otherwise, an error code indicating failure is returned.

[Example]

For details, see the description of HI_MPI_VPSS_CreateGrp.

[See Also]

HI_MPI_VPSS_EnableChn

HI_MPI_VPSS_EnableBackupFrame

[Description]

Enables backup frames.

[Syntax]

```
HI_S32 HI_MPI_VPSS_EnableBackupFrame(VPSS_GRP VpssGrp)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: llibmpi.a

[Note]

- Before calling this MPI, you must create a group.
- Backup frames are enabled by default.

[Example]

None

[See Also]

None

HI_MPI_VPSS_DisableBackupFrame

[Description]

Disables backup frames.

[Syntax]

```
HI_S32 HI_MPI_VPSS_DisableBackupFrame(VPSS_GRP VpssGrp)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: llibmpi.a

[Note]

- Before calling this MPI, you must create a group.
- After backup frames are disabled, the memory required for storing one frame is spared for each VPSS group. However, the functions such as VO picture switching and

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

playback control are affected. Therefore, you need to disable backup frames with caution.

- If the available memory is insufficient, the memory is allocated to each module based on the minimum requirements. For example, when the number of display frames of the frontend decoding channel (**DisplayFrameNum**) is set to **0**, HI_MPI_VPSS_DisableBackupFrame must be called before HI_MPI_VPSS_StartGrp(). HI_MPI_VPSS_DisableBackupFrame can be dynamically called in other cases.

[Example]

None

[See Also]

None

HI_MPI_VPSS_SetGrpParam

[Description]

Sets the advanced parameters of a VPSS group.

[Syntax]

```
HI_S32 HI_MPI_VPSS_SetGrpParam(VPSS_GRP VpssGrp, VPSS_GRP_PARAM_S
*pstVpssParam)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstVpssParam	Advanced parameters configuration	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- For details about the limitations on the pstVpssParam members, see the descriptions of VPSS_GRP_PARAM_S.
- For the Hi3521/Hi3520A, the NR parameter value does not change after you call this MPI. To modify the NR parameter, you are advised to call HI_MPI_VPSS_SetChnNrParam( ).
- When the Hi3520D V200 is connected to the AHD frontend, noises are obvious and noises in stripe are jumping up and down. In this case, increasing the strength of spatial domain noise reduction is not enough, because spatial domain noise reduction can reduce the background noise but has no effect on the jumping noises in stripe. Therefore, the strength of time domain noise reduction also needs to be increased. Note that if the value of time domain noise reduction is too large, smearing effect occurs in motion scenarios. The recommended values for spatial domain noise reduction and time domain noise reduction are 32 and 10 respectively.

[Example]

None

[See Also]

None

HI_MPI_VPSS_GetGrpParam

[Description]

Obtains the advanced parameters of a VPSS group.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetGrpParam(VPSS_GRP VpssGrp, VPSS_GRP_PARAM_S
*pstVpssParam)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstVpssParam	Advanced attributes	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The channel ID must fall within [0, VPSS_MAX_CHN_NUM).

[Example]

None

[See Also]

HI_MPI_VPSS_SetChnAttr

HI_MPI_VPSS_SetCropCfg

[Description]

Sets the VPSS crop attributes.

[Syntax]

```
HI_S32 HI_MPI_VPSS_SetCropCfg(VPSS_GRP VpssGrp, VPSS_CROP_INFO_S
*pstCropInfo)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstCropInfo	Crop function parameters	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- When the coordinate type of the start point is VPSS_CROP_RITIO_COOR (relative coordinates), the value ranges of s32X and s32Y are [0, 999].
- When the coordinate type of the start point is VPSS_CROP_ABS_COOR (absolute coordinates), the value ranges of s32X and s32Y are [0, VPSS_MAX_IMAGE_WIDTH]. The value ranges of the width and height of the region to be cropped are [VPSS_MIN_IMAGE_WIDTH, VPSS_MAX_IMAGE_WIDTH]. If the configured start coordinates exceed the range of the actual picture, the picture is not cropped. If the configured width and height of the region to be cropped exceed the range of the actual picture, only the region that falls within the range of the picture is automatically cropped.
- For details about the maximum and minimum sizes of the region to be cropped, see the value ranges of **u32MaxW** and **u32MaxH** in VPSS_GRP_ATTR_S.
- The coordinates of the crop area start point are automatically rounded down to integral multiples of 4. The width or height of the crop area is automatically rounded down to an integral multiple of 16.
- The field capture function of the crop area is valid only for interlaced pictures. The crop area is based on the original input frame picture. If prescaling is enabled, the crop area is based on the size of the prescaled picture. For the bypass channel, the crop area is always based on the size of the original input picture no matter whether prescaling is enabled.

[Example]

For details, see HI_MPI_VPSS_CreateGrp.

[See Also]

None

HI_MPI_VPSS_GetCropCfg

[Description]

Obtains the VPSS crop attributes.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetCropCfg(VPSS_GRP VpssGrp, VPSS_CROP_INFO_S
*pstCropInfo)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstCropInfo	Crop function parameters	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
	details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

None

[Example]

For details, see HI_MPI_VPSS_CreateGrp.

[See Also]

None

HI_MPI_VPSS_UserSendFrame

[Description]

Transmits data to a VPSS group.

[Syntax]

```
HI_S32 HI_MPI_VPSS_UserSendFrame(VPSS_GRP VpssGrp, VIDEO_FRAME_INFO_S
*pstVideoFrame)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstVideoFrame	Information about the pictures to be transmitted	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- You need to control the frame rate when calling this MPI.
- Before calling this MPI, you must start a VPSS group and bind it to a back-end receiver.

[Example]

None

[See Also]

None

HI_MPI_VPSS_UserSendFrameTimeout

[Description]

Sends data to a VPSS group in timeout mode.

[Syntax]

```
HI_S32 HI_MPI_VPSS_UserSendFrameTimeout(VPSS_GRP VpssGrp,
VIDEO_FRAME_INFO_S *pstVideoFrame, HI_U32 u32MilliSec)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstVideoFrame	Information about the pictures to be transmitted	Input
u32MilliSec	Timeout period. The value 0 indicates no timeout.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- You need to control the frame rate when calling this MPI.
- Before calling this MPI, you must start a VPSS group and bind it to a back-end receiver.

[Example]

None

[See Also]

None

HI_MPI_VPSS_SetChnMode

[Description]

Sets the working mode of a VPSS channel.

[Syntax]

```
HI_S32 HI_MPI_VPSS_SetChnMode(VPSS_GRP VpssGrp, VPSS_CHN VpssChn,
VPSS_CHN_MODE_S *pstVpssMode)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_CHN_NUM)	Input
pstVpssMode	Channel working mode. For details, see VPSS_CHN_MODE_S.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Before calling this MPI, you must create a group.
- The channel ID must fall within [0, VPSS_MAX_CHN_NUM).
- The automatic mode is used by default. The mode can be dynamically switched.
- The working modes of bypass channels and extended channels cannot be set. If this MPI is called when a bypass channel or an extended channel is used, the operation has no effect but no code indicating failure is returned.
- If the VPSS enables the scaling function, the pixel value range of the output picture is [16, 235].

[Example]

```
VPSS_GRP_ATTR_S stGrpVpssAttr;
VPSS_CHN_ATTR_S stChnAttr;
VIDEO_FRAME_INFO_S stFrame;
HI_S32 s32Ret = HI_SUCCESS;
   VPSS_CHN_MODE_S stVpssMode;
   HI_U32 u32Depth = 8;
VPSS_GRP VpssGrp;
VPSS_CHN VpssChn

VpssGrp = 0;
VpssChn = 0;

stGrpVpssAttr.u32MaxW = 720;
stGrpVpssAttr.u32MaxH = 576;
stGrpVpssAttr.bDrEn = HI_FALSE;
stGrpVpssAttr.bDbEn = HI_FALSE;
stGrpVpssAttr.bIeEn = HI_FALSE;
stGrpVpssAttr.bNrEn = HI_FALSE;
stGrpVpssAttr.bHistEn = HI_FALSE;
stGrpVpssAttr.enDieMode = VPSS_DIE_MODE_NODIE;
stGrpVpssAttr.enPixFmt = PIXEL_FORMAT_YUV_SEMIPLANAR_422;

s32Ret = HI_MPI_VPSS_CreateGrp(VpssGrp, &stGrpVpssAttr);
if(s32Ret != HI_SUCCESS)
{
return s32Ret;
}

s32Ret = HI_MPI_VPSS_GetGrpAttr(VpssGrp, &stGrpVpssAttr);
if(s32Ret != HI_SUCCESS)
{
return s32Ret;
}

stGrpVpssAttr.bIeEn = HI_TRUE;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
stGrpVpssAttr.bNrEn = HI_TRUE;
s32Ret = HI_MPI_VPSS_SetGrpAttr(VpssGrp, &stGrpVpssAttr);
if(s32Ret != HI_SUCCESS)
{
return s32Ret;
}

s32Ret = HI_MPI_VPSS_GetChnMode(VpssGrp,VpssChn,&stVpssMode);
if(s32Ret != HI_SUCCESS)
{
return s32Ret;
}

stVpssMode.enChnMode = VPSS_CHN_MODE_USER;
stVpssMode.enPixelFormat = PIXEL_FORMAT_YUV_SEMIPLANAR_420;
stVpssMode.u32Width = 720;
stVpssMode.u32Height = 576;
s32Ret = HI_MPI_VPSS_SetChnMode(VpssGrp,VpssChn,&stVpssMode);
if(s32Ret != HI_SUCCESS)
{
return s32Ret;
}
HI_MPI_VPSS_SetDepth(VpssGrp,VpssChn,&u32Depth);
HI_MPI_VPSS_EnableChn(VpssGrp, VpssChn);
HI_MPI_VPSS_StartGrp(VpssGrp);
s32Ret = HI_MPI_VPSS_UserGetFrame(VpssGrp, VpssChn, &stFrame);
if(s32Ret == HI_SUCCESS)
{
HI_MPI_VPSS_UserRlsFrame(VpssGrp, VpssChn, &stFrame);
}
```

[See Also]

None

HI_MPI_VPSS_GetChnMode

[Description]

Obtains the working mode of a VPSS channel.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetChnMode(VPSS_GRP VpssGrp, VPSS_CHN VpssChn,
VPSS_CHN_MODE_S *pstVpssMode)
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_CHN_NUM)	Input
pstVpssMode	Channel working mode	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The channel ID must fall within [0, VPSS_MAX_CHN_NUM).
- The working modes of bypass channels and extended channels cannot be set. If this MPI is called when a bypass channel or an extended channel is used, the operation has no effect but no code indicating failure is returned.

[Example]

See the example of HI_MPI_VPSS_SetChnMode.

[See Also]

HI_MPI_VPSS_SetChnMode

HI_MPI_VPSS_SetDepth

[Description]

Sets the depth of the user picture queue.

[Syntax]

```
HI_S32 HI_MPI_VPSS_SetDepth(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, HI_U32
u32Depth)
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_CHN_NUM)	Input
u32Depth	Queue depth	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The channel ID must fall within [0, VPSS_MAX_CHN_NUM).
- The default user queue depth is 0, and the maximum user queue depth is 8.

[Example]

See the example of HI_MPI_VPSS_SetChnMode.

[See Also]

None

HI_MPI_VPSS_GetDepth

[Description]

Obtains the depth of the user picture queue.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetDepth(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, HI_U32 *pu32Depth)
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_CHN_NUM)	Input
pu32Depth	Queue depth	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The channel ID must fall within [0, VPSS_MAX_CHN_NUM).

[Example]

None

[See Also]

None

HI_MPI_VPSS_UserGetFrame

[Description]

Obtains a processed frame from a channel.

[Syntax]

```
HI_S32 HI_MPI_VPSS_UserGetFrame(VPSS_GRP VpssGrp, VPSS_CHN VpssChn,
VIDEO_FRAME_INFO_S *pstVideoFrame)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
	Value range: [0, VPSS_MAX_GRP_NUM)	
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_PHY_CHN_NUM)	Input
pstVideoFrame	Picture information	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- This API applies only to physical channels.
- Pictures can be obtained only when the channel working mode is user mode and the queue depth is not 0.

[Example]

See the example of HI_MPI_VPSS_SetChnMode.

[See Also]

None

HI_MPI_VPSS_UserGetGrpFrame

[Description]

Obtains a source frame from a group.

The following is the typical application scenario of HI_MPI_VPSS_UserGetGrpFrame: The same frame needs to be displayed in two channels at the PIP layer and common video layer when the pause and step operations are required during playback on an HD device. This function is supported when HI_MPI_VPSS_UserGetGrpFrame works with HI_MPI_VPSS_UserSendFrame.

[Syntax]

```
HI_S32 HI_MPI_VPSS_UserGetGrpFrame(VPSS_GRP VpssGrp, VIDEO_FRAME_INFO_S
*pstVideoFrame, HI_U32 u32FrameIndex)
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstVideoFrame	Picture information	Output
u32FrameIndex	Picture index ID. This parameter is invalid and is set to **0** currently.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The original picture in the backup node in a group can be obtained by calling this MPI. The VPSS stores the picture of the head node in the buffer queue in the backup node in the following situations:
- If the picture of the head node in the buffer queue requires to be processed by VPSS hardware, the VPSS stores this picture in the backup node to replace the source picture.
- If the receiving module bound to the back end requires that the VPSS store the picture of the head node in the buffer queue in the backup node, the VPSS replaces the picture in the backup node even if this picture is not processed by hardware.
- If the backup node is null, the HI_ERR_VPSS_NOTREADY error code will be returned after this MPI is called.
- Release the buffer for storing the obtained pictures in time. Otherwise, the VB is insufficient or playback stops. You are advised to call HI_MPI_VPSS_UserReleaseGrpFrame after calling HI_MPI_VPSS_UserGetGrpFrame.

[Example]

See the VDEC sample in the SDK.

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VPSS_UserReleaseGrpFrame

[Description]

Releases the buffer for storing a frame.

[Syntax]

```
HI_S32 HI_MPI_VPSS_UserReleaseGrpFrame(VPSS_GRP VpssGrp,
VIDEO_FRAME_INFO_S *pstVideoFrame)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstVideoFrame	Picture information	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

The VpssGrp parameter for this MPI is meaningless. You can set it to any value within the value range.

[Example]

None

[See Also]

None

HI_MPI_VPSS_UserReleaseFrame

[Description]

Releases the buffer for storing a frame.

[Syntax]

```
HI_S32 HI_MPI_VPSS_UserReleaseFrame (VPSS_GRP VpssGrp, VPSS_CHN VpssChn,
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
VIDEO_FRAME_INFO_S *pstVideoFrame)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_CHN_NUM)	Input
pstVideoFrame	Picture information	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

The VpssGrp and VpssChn parameters for this MPI are meaningless. You can set them to any values within value ranges.

[Example]

See the example of HI_MPI_VPSS_SetChnMode.

[See Also]

None

HI_MPI_VPSS_SetChnNrParam

[Description]

Sets the advanced NR attribute of a channel.

[Syntax]

```
HI_S32 HI_MPI_VPSS_SetChnNrParam(VPSS_GRP VpssGrp, VPSS_CHN VpssChn,
VPSS_CHN_NR_PARAM_S *pstChnNrParam)
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_CHN_NUM)	Input
pstChnNrParam	Settings of the advanced NR attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 does not support this MPI.
Hi3521	Chn0 and chn1 share a set of NR coefficients. When you call this MPI to change the NR settings of chn0, the NR settings of chn1 also change. This rule applies to chn2 and chn3.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.
Hi3520D/Hi3515A /Hi3515C	The Hi3520D/Hi3515A/Hi3515C does not support this MPI.

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The channel ID must fall within [0, VPSS_MAX_CHN_NUM).
- For details about the limitations on the pstChnNrParam members, see the descriptions of VPSS_CHN_NR_PARAM_S.

[Example]

None

[See Also]

Copyright © HiSilicon Technologies Co., Ltd

None

HI_MPI_VPSS_GetChnNrParam

[Description]

Obtains the advanced NR attribute of a channel.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetChnNrParam(VPSS_GRP VpssGrp, VPSS_CHN VpssChn,
VPSS_CHN_NR_PARAM_S *pstChnNrParam)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_CHN_NUM)	Input
pstChnNrParam	Settings of the advanced NR attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 does not support this MPI.
Hi3521	The Hi3521 supports this MPI.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.
Hi3520D/Hi3515A /Hi3515C	The Hi3520D/Hi3515A/Hi3515C does not support this MPI.

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The channel ID must fall within [0, VPSS_MAX_CHN_NUM).
- For details about the limitations on the pstChnNrParam members, see the descriptions of VPSS_CHN_NR_PARAM_S.

[Example]

None

[See Also]

None

HI_MPI_VPSS_SetChnSpParam

[Description]

Sets the advanced SP attribute of a channel.

[Syntax]

```
HI_S32 HI_MPI_VPSS_SetChnSpParam(VPSS_GRP VpssGrp, VPSS_CHN VpssChn,
VPSS_CHN_SP_PARAM_S *pstChnSpParam)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_CHN_NUM)	Input
pstChnSpParam	Settings of the advanced SP attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 does not support this MPI.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Description
Hi3521	Chn0 does not support SP. If you call this MPI to set the advanced SP attribute for chn0, the settings do not take effect and no error occurs.
Hi3518/Hi3516C	The Hi3518/Hi3516C supports this MPI.
Hi3520D/Hi3515A /Hi3515C	The Hi3520D/Hi3515A/Hi3515C supports this MPI.

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The channel ID must fall within [0, VPSS_MAX_CHN_NUM).
- For details about the limitations on the pstChnSpParam members, see the descriptions of VPSS_CHN_SP_PARAM_S.

[Example]

None

[See Also]

None

HI_MPI_VPSS_GetChnSpParam

[Description]

Obtains the advanced SP attribute of a channel.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetChnSpParam(VPSS_GRP VpssGrp, VPSS_CHN VpssChn,
VPSS_CHN_SP_PARAM_S *pstChnSpParam)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_CHN_NUM)	Input
pstChnSpParam	Settings of the advanced SP attribute	Output

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 does not support this MPI.
Hi3521/Hi3520A	Chn0 does not support SP. If you call this MPI to obtain the advanced SP attribute of chn0, no result is obtained and no error occurs.
Hi3518/Hi3516C	The Hi3518/Hi3516C supports this MPI.
Hi3520D/Hi3515A/ Hi3515C	The Hi3520D/Hi3515A/Hi3515C supports this MPI.

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The channel ID must fall within [0, VPSS_MAX_CHN_NUM).
- For details about the limitations on the pstChnSpParam members, see the descriptions of VPSS_CHN_SP_PARAM_S.

[Example]

None

[See Also]

None

HI_MPI_VPSS_SetPreScale

[Description]

Sets the prescaling attribute of a VPSS group.

[Syntax]

```
HI_S32 HI_MPI_VPSS_SetPreScale(VPSS_GRP VpssGrp,VPSS_PRESCALE_INFO_S
*pstPreScaleInfo)
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstPreScaleInfo	Setting of the prescaling attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 does not support this MPI.
Hi3521/Hi3520A	The target picture size cannot be greater than the maximum picture size allowed by a channel group. For details, see the descriptions of VPSS_PRESCALE_INFO_S.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- Prescaling is valid only for captured VI pictures.
- The prescaling attribute of a bypass channel cannot be set.
- For details about the limitations on the pstPreScaleInfo members, see the descriptions of VPSS_PRESCALE_INFO_S.
- The Hi3520D/Hi3515A/Hi3515C supports pre-scaling only two times of the original size. Images whose width is greater than 960 cannot be pre-scaled.

[Example]

None

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

HI_MPI_VPSS_GetPreScale

[Description]

Obtains the prescaling attribute of a VPSS group.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetPreScale(VPSS_GRP VpssGrp,VPSS_PRESCALE_INFO_S
*pstPreScaleInfo)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstPreScaleInfo	Setting of the prescaling attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 does not support this MPI.
Hi3521/Hi3520A	The Hi3521/Hi3520A supports this MPI.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.
Hi3520D/Hi3515A /Hi3515C	The Hi3520D/Hi3515A/Hi3515C supports this MPI.

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- For details about the limitations on the pstPreScaleInfo members, see the descriptions of VPSS_PRESCALE_INFO_S.

[Example]

None

[See Also]

None

HI_MPI_VPSS_SetChnField

[Description]

Sets the captured field attribute of a channel.

[Syntax]

```
HI_S32 HI_MPI_VPSS_SetChnField(VPSS_GRP VpssGrp, VPSS_CHN VpssChn,
VPSS_CAPSEL_E *enCapSel)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID	Input
*enCapSel	Channel captured field information	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 does not support this MPI.
Hi3521/Hi3520A	Only chn0 and chn1 support the captured field function, and their captured field attributes are set at the same time. After you call this MPI, both the captured field settings of chn0 and chn1 change.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.
Hi3520D/Hi3515A	The Hi3520D/Hi3515A/Hi3515C does not support this MPI.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Description
/Hi3515C	

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The channel ID must be valid. The captured field attribute of only channel 0 or channel 1 can be set.

[Example]

None

[See Also]

None

HI_MPI_VPSS_GetChnField

[Description]

Obtains the captured field attribute of a channel.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetChnField(VPSS_GRP VpssGrp, VPSS_CHN VpssChn,
VPSS_CAPSEL_E *enCapSel)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	VPSS channel ID Value range: [0, VPSS_MAX_CHN_NUM)	Input
*enCapSel	Channel captured field information	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
	details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 does not support this MPI.
Hi3521/Hi3520A	The Hi3521/Hi3520A supports this MPI.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.
Hi3520D/Hi3515A /Hi3515C	The Hi3520D/Hi3515A/Hi3515C does not support this MPI.

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The channel ID must fall within [0, VPSS_MAX_CHN_NUM).

[Example]

None

[See Also]

None

HI_MPI_VPSS_SetGrpField

[Description]

Sets the captured field attributes of a group.

[Syntax]

```
HI_S32 HI_MPI_VPSS_SetGrpField(VPSS_GRP VpssGrp, VPSS_CAPSEL_E enCapSel)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM]	Input
enCapSel	Group captured field information	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 supports this MPI.
Hi3521/Hi3520A	The Hi3521/Hi3520A supports this MPI.
Hi3518/Hi3516C	The Hi3518/Hi3516C supports this MPI.
Hi3520D/Hi3515A/Hi3515C	The Hi3520D/Hi3515A/Hi3515C supports this MPI.

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- A group must be created before this MPI is called.
- Calling this MPI forcibly captures the fields of frames.

[Example]

None

[See Also]

None

HI_MPI_VPSS_GetGrpField

[Description]

Obtains the captured field attributes of a group.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetGrpField(VPSS_GRP VpssGrp, VPSS_CAPSEL_E *penCapSel)
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM]	Input
*enCapSel	Group captured field information	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 supports this MPI.
Hi3521/Hi3520A	The Hi3521/Hi3520A supports this MPI.
Hi3518/Hi3516C	The Hi3518/Hi3516C supports this MPI.
Hi3520D/Hi3515A/Hi3515C	The Hi3520D/Hi3515A/Hi3515C supports this MPI.

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- A group must be created before this MPI is called.

[Example]

None

[See Also]

None

HI_MPI_VPSS_SetGrpSizer

[Description]

Sets the picture size for the VPSS to filter out pictures.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
HI_S32 HI_MPI_VPSS_SetGrpSizer(VPSS_GRP VpssGrp, VPSS_SIZER_INFO_S
*pstVpssSizerInfo)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstVpssSizerInfo	Picture size for the VPSS to filter out pictures	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 supports this MPI, but this MPI is not used typically.
Hi3521/Hi3520A	The Hi3521/Hi3520A supports this MPI.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.
Hi3520D/Hi3515A /Hi3515C	The Hi3520D/Hi3515A/Hi3515C supports this MPI.

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- Only the pictures captured by VI channels can be filtered based on the picture size.
- The bypass channel does not allow you to set the picture size for filtering out pictures.
- Pictures are filtered out based on the picture size at the front end of the VPSS, which affects subsequent processing.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[See Also]

None

HI_MPI_VPSS_GetGrpSizer

[Description]

Obtains the picture size for the VPSS to filter out pictures.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetGrpSizer(VPSS_GRP VpssGrp, VPSS_SIZER_INFO_S
*pstVpssSizerInfo)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstVpssSizerInfo	Picture size for the VPSS to filter out pictures	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 supports this MPI.
Hi3521/Hi3520A	The Hi3521/Hi3520A supports this MPI.
Hi3518/Hi3516C	The Hi3518/Hi3516C does not support this MPI.
Hi3520D/Hi3515A/ Hi3515C	The Hi3520D/Hi3515A/Hi3515C supports this MPI.

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.

[Example]

None

[See Also]

None

HI_MPI_VPSS_SetGrpFrameRate

[Description]

Sets the frame rate control information for the VPSS.

[Syntax]

```
HI_S32 HI_MPI_VPSS_SetGrpFrameRate(VPSS_GRP VpssGrp, VPSS_FRAME_RATE_S
*pstVpssFrameRate)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstVpssFrameRate	Frame rate control information	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 supports this MPI, but this MPI is not used typically.
Hi3521/Hi3520A	The Hi3521/Hi3520A supports this MPI.
Hi3518/Hi3516C	The Hi3518/Hi3516C supports this MPI.
Hi3520D/Hi3515A /Hi3515C	The Hi3520D/Hi3515A/Hi3515C supports this MPI.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- Frame rate control is valid only for the pictures captured by VI channels.
- The bypass channel does not support frame rate control.
- Frame rate control is performed after pictures are filtered out based on the picture size. Frame rate control affects subsequent processing.

[Example]

None

[See Also]

None

HI_MPI_VPSS_GetGrpFrameRate

[Description]

Obtains the frame rate control information for the VPSS.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetGrpFrameRate(VPSS_GRP VpssGrp, VPSS_FRAME_RATE_S
*pstVpssFrameRate)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pstVpssFrameRate	Frame rate control information	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 supports this MPI.
Hi3521/Hi3520A	The Hi3521/Hi3520A supports this MPI.
Hi3518/Hi3516C	The Hi3518/Hi3516C supports this MPI.
Hi3520D/Hi3515A/ Hi3515C	The Hi3520D/Hi3515A/Hi3515C supports this MPI.

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.

[Example]

None

[See Also]

None

HI_MPI_VPSS_SetDelay

[Description]

Sets the length of the delay queue that is used to delay starting VPSS processing.

This MPI applies to the following application scenarios:

- In an intelligent application, the pictures captured by VI channels must be processed by the TDE before they are transmitted to the VPSS. As the VPSS processes pictures at a high speed, ensure that the TDE has finished picture processing before VPSS processing starts. Therefore, the function of delaying starting VPSS processing is provided.
- In VI-VO cascade mode, the picture delay between the master chip and the slave chip can be adjusted by calling this MPI, reducing the time difference between chips.

[Syntax]

```
HI_S32 HI_MPI_VPSS_SetDelay(VPSS_GRP VpssGrp, HI_U32 u32Delay)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
u32Delay	Delay queue length	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
	Value range: [0, 5]	

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.
- The delay function is valid only for the pictures captured by VI channels. The delay period is calculated as follows: Delay period = Time of capturing a frame by VI channels x u32Delay. **u32Delay** indicates the delay queue length.
- The bypass channel does not support the delay function.

[Example]

None

[Dee Also]

HI_MPI_VPSS_GetDelay

HI_MPI_VPSS_GetDelay

[Description]

Obtains the length of the delay queue that is used to delay starting VPSS processing.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetDelay(VPSS_GRP VpssGrp, HI_U32 *pu32Delay)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
pu32Delay	Pointer to the obtained delay queue length	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The group ID must fall within [0, VPSS_MAX_GRP_NUM).
- Before calling this MPI, you must create a group.

[Example]

None

[See Also]

HI_MPI_VPSS_SetDelay

HI_MPI_VPSS_SetExtChnAttr

[Description]

Sets the attribute of an extended VPSS channel.

[Syntax]

```
HI_S32 HI_MPI_VPSS_SetExtChnAttr(VPSS_GRP VpssGrp, VPSS_CHN VpssChn,
VPSS_EXT_CHN_ATTR_S *pstExtChnAttr)
```

[Parameter]

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	ID of an extended VPSS channel Value range: (VPSS_MAX_PHY_CHN_NUM, VPSS_MAX_CHN_NUM)	Input
pExtChnAttr	Attribute of an extended VPSS channel	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 does not support this MPI.
Hi3521/Hi3520A	The Hi3521/Hi3520A does not support this MPI.
Hi3518/Hi3516C	The Hi3518/Hi3516C supports this MPI.
Hi3520D/Hi3515A/Hi3515C	The Hi3520D/Hi3515A/Hi3515C does not support this MPI.

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

- The extended channels can be bound only to physical channels. The channel width and height must be 2-pixel-aligned. SrcFrameRate must be greater than or equal to DstFrameRate. If both SrcFrameRate and DstFrameRate are set to −1, the frame rate is not controlled.
- Multiple channels can be bound to the same physical channel. If the maximum number of extended channels to be bound is reached, the attributes fail to be set.
- Before calling this MPI, you must create a group.

[Example]

None

[See Also]

None

HI_MPI_VPSS_GetExtChnAttr

[Description]

Obtains the attribute of an extended VPSS channel.

[Syntax]

```
HI_S32 HI_MPI_VPSS_GetExtChnAttr(VPSS_GRP VpssGrp, VPSS_CHN VpssChn,
VPSS_EXT_CHN_ATTR_S *pstExtChnAttr)
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VpssGrp	VPSS group ID Value range: [0, VPSS_MAX_GRP_NUM)	Input
VpssChn	ID of an extended VPSS channel Value range: (VPSS_MAX_PHY_CHN_NUM, VPSS_MAX_CHN_NUM)	Input
pExtChnAttr	Attribute of an extended VPSS channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 5.5 "Error Codes."

[Difference]

Chip	Description
Hi3531/Hi3532	The Hi3531/Hi3532 does not support this MPI.
Hi3521/Hi3520A	The Hi3521/Hi3520A does not support this MPI.
Hi3518/Hi3516C	The Hi3518/Hi3516C supports this MPI.
Hi3520D/Hi3515A/ Hi3515C	The Hi3520D/Hi3515A/Hi3515C does not support this MPI.

[Requirement]

- Header files: hi_comm_vpss.h, mpi_vpss.h
- Library file: libmpi.a

[Note]

Before calling this MPI, you must create a group.

[Example]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

5.4 Data Types

The data types of the VPSS are defined as follows:

- VPSS_MAX_GRP_NUM: Defines the maximum number of VPSS groups.
- VPSS_MAX_CHN_NUM: Defines the maximum number of VPSS channels.
- VPSS_MAX_PHY_CHN_NUM: Defines the maximum number of physical VPSS channels.
- VPSS_MAX_EXT_CHN_NUM: Defines the maximum number of extended VPSS channels.
- VPSS_BSTR_CHN: Defines the ID of a large stream channel.
- VPSS_LSTR_CHN: Defines the ID of a small stream channel.
- VPSS_PRE0_CHN: Defines the ID of preview channel 0.
- VPSS_PRE1_CHN: Defines the ID of preview channel 1.
- VPSS_BYPASS_CHN: Defines the bypass channel ID.
- VPSS_GRP: Defines the VPSS_GRP type.
- VPSS_CHN: Defines the VPSS_CHN type.
- VPSS_FRAME_WORK_E: Defines the video frames.
- VPSS_DIE_MODE_E: Defines DEI mode.
- VPSS_DISPLAY_MODE_E: Defines the display mode.
- VPSS_SF_WINDOW_E: Defines the size of the spatial domain filtering window.
- VPSS_CROP_COORDINATE_E: Defines the type of the coordinates of the crop start point.
- RECT_S: Defines a rectangle area.
- VPSS_CROP_INFO_S: Defines the information required by the crop function.
- VPSS_GRP_ATTR_S: Defines the static attributes of a VPSS group.
- VPSS_FRAME_S: Defines the attributes of a video frame.
- VPSS_CHN_ATTR_S: Defines the static attributes of a VPSS physical channel.
- VPSS_GRP_PARAM_S: Defines the advanced VPSS attributes.
- VPSS_CHN_MODE_E: Defines the working mode of a VPSS channel.
- VPSS_CHN_MODE_S: Defines the structure of the working mode of a VPSS channel.
- VPSS_CHN_NR_PARAM_S: Defines the advanced NR parameters for a channel.
- VPSS_CHN_SP_PARAM_S: Defines the advanced SP parameter for a channel.
- VPSS_CHN_PARAM_S: Defines VPSS channel advanced attributes.
- VPSS_CAPSEL_E: Defines the VPSS captured field type.
- VPSS_PRESCALE_INFO_S: Defines the VPSS prescaling attribute.
- VPSS_SIZER_INFO_S: Defines the VPSS size filtering attribute.
- VPSS_FRAME_RATE_S: Defines the VPSS frame rate control attribute.
- VPSS_EXT_CHN_ATTR_S: Defines the attribute of an extended VPSS channel.

VPSS_MAX_GRP_NUM

[Description]

Defines the maximum number of VPSS groups.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
#define VPSS_MAX_GRP_NUM   128
```

[Note]

None

[See Also]

None

VPSS_MAX_CHN_NUM

[Description]

Defines the maximum number of VPSS channels.

[Syntax]

```
#define VPSS_MAX_CHN_NUM   (VPSS_MAX_PHY_CHN_NUM + VPSS_MAX_EXT_CHN_NUM +1)
```

[Note]

The VPSS channels include physical channels, extended channels, and bypass channels.

[See Also]

None

VPSS_MAX_PHY_CHN_NUM

[Description]

Defines the maximum number of physical VPSS channels.

[Syntax]

```
#define VPSS_MAX_PHY_CHN_NUM 2
```

[Note]

The maximum number is related to hardware resources. The Hi3521/Hi3520A supports a maximum of four physical VPSS channels, and the Hi3518/Hi3516C supports a maximum of two physical VPSS channels.

[See Also]

None

VPSS_MAX_EXT_CHN_NUM

[Description]

Defines the maximum number of extended VPSS channels.

[Syntax]

```
#define VPSS_MAX_EXT_CHN_NUM 2
```

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

The maximum number depends on solution design and system performance. The Hi3518/Hi3516C supports a maximum of five extended VPSS channels.

[See Also]

None

VPSS_BSTR_CHN

[Description]

Defines the ID of a large stream channel.

[Syntax]

```
#define VPSS_BSTR_CHN   0
```

[Note]

None

[See Also]

None

VPSS_LSTR_CHN

[Description]

Defines the ID of a small stream channel.

[Syntax]

```
#define VPSS_LSTR_CHN   1
```

[Note]

None

[See Also]

None

VPSS_PRE0_CHN

[Description]

Defines the ID of preview channel 0.

[Syntax]

```
#define VPSS_PRE0_CHN   2
```

[Note]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VPSS_PRE1_CHN

[Description]

Defines the ID of preview channel 1.

[Syntax]

```
#define VPSS_PRE1_CHN   3
```

[Note]

No workaround is provided.

[See Also]

None

VPSS_BYPASS_CHN

[Description]

Defines the bypass channel ID.

[Syntax]

```
#define VPSS_BYPASS_CHN   4
```

[Note]

Because the bypass channel index is after physical channel indexes, the bypass channel ID is 4 for the Hi3531 or Hi3521 and is 2 for the Hi3518/Hi3516C.

[See Also]

None

VPSS_GRP

[Description]

Defines the VPSS_GRP type.

[Syntax]

```
typedef   HI_S32    VPSS_GRP;
```

[Note]

No workaround is provided.

[See Also]

None

VPSS_CHN

[Description]

Defines the VPSS_CHN type.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
typedef   HI_S32    VPSS_CHN;
```

[Note]

None

[See Also]

None

VPSS_FRAME_WORK_E

[Description]

Defines four video frames.

[Syntax]

```
typedef enum HI_VPSS_FRAME_WORK_E
{
    VPSS_FRAME_WORK_LEFT=  0,
    VPSS_FRAME_WORK_RIGHT    =  1,
    VPSS_FRAME_WORK_BOTTOM   =  2,
    VPSS_FRAME_WORK_TOP =  3,
    VPSS_FRAME_WORK_BUTT
}VPSS_FRAME_WORK_E;;
```

[Member]

Member	Description
VPSS_FRAME_WORK_LEFT	Left frame
VPSS_FRAME_WORK_RIGHT	Right frame
VPSS_FRAME_WORK_BOTTOM	Bottom frame
VPSS_FRAME_WORK_TOP	Top frame

[Note]

None

[See Also]

None

VPSS_DIE_MODE_E

[Description]

Defines DEI mode.

[Syntax]

```
typedef enum  HI_VPSS_DIE_MODE_E
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
{
    VPSS_DIE_MODE_AUTO      = 0,
    VPSS_DIE_MODE_NODIE     = 1,
    VPSS_DIE_MODE_DIE       = 2,
    VPSS_DIE_MODE_BUTT
}VPSS_DIE_MODE_E;
```

[Member]

Member	Description
VPSS_DIE_MODE_AUTO	Automatically perform DEI based on the current picture format.
VPSS_DIE_MODE_NODIE	Do not perform DEI forcibly.
VPSS_DIE_MODE_DIE	Perform DEI forcibly.

[Note]

The VPSS_DIE_MODE_AUTO mode is recommended.

[See Also]

VPSS_GRP_ATTR_S

VPSS_DISPLAY_MODE_E

[Description]

Defines the display mode.

[Syntax]

```
typedef enum  HI_VPSS_DISPLAY_MODE_E
{
    VPSS_DISPLAY_MODE_TV    = 0,
    VPSS_DISPLAY_MODE_PC    = 1,
    VPSS_DISPLAY_MODE
}VPSS_DISPLAY_MODE_E;
```

[Member]

Member	Description
VPSS_DISPLAY_MODE_TV	TV mode
VPSS_DISPLAY_MODE_PC	PC mode

[Note]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

VPSS_GRP_PARAM_S

VPSS_SF_WINDOW_E

[Description]

Defines the size of the spatial domain filtering window. A larger window leads to greater filtering strength and more blurred picture.

[Syntax]

```
typedef enum  HI_VPSS_SF_WINDOW_E
{
    VPSS_SF_WINDOW_3X3      = 0,
    VPSS_SF_WINDOW_5X5      = 1,
    VPSS_SF_WINDOW_7X7      = 2,
    VPSS_SF_WINDOW_9X9 = 3,
    VPSS_SF_WINDOW_BUTT
}VPSS_SF_WINDOW_E;
```

[Member]

Member	Description
VPSS_SF_WINDOW_3X3	3x3 control window
VPSS_SF_WINDOW_5X5	5x5 control window
VPSS_SF_WINDOW_7X7	7x7 control window
VPSS_SF_WINDOW_9X9	9x9 control window

[Note]

None

[See Also]

VPSS_GRP_PARAM_S

VPSS_CROP_COORDINATE_E

[Description]

Defines the type of the coordinates of the crop start point.

[Syntax]

```
typedef enum HI_VPSS_CROP_COORDINATE_E
{
    VPSS_CROP_RITIO_COOR = 0,
    VPSS_CROP_ABS_COOR
}VPSS_CROP_COORDINATE_E;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Member]

Member	Description
VPSS_CROP_RITIO_COOR	Relative coordinate
VPSS_CROP_ABS_COOR	Absolute coordinate

[Note]

The relative coordinates indicate that the coordinates of the start point are expressed by aspect ratio of the current picture. The relative coordinates must be converted before being used. For details, see VPSS_CROP_INFO_S.

[See Also]

VPSS_CROP_INFO_S

RECT_S

[Description]

Defines a rectangle area.

[Syntax]

```
typedef struct hiRECT_S
{
    HI_S32 s32X;
    HI_S32 s32Y;
    HI_U32 u32Width;
    HI_U32 u32Height;
}RECT_S;
```

[Member]

Member	Description
s32X	Horizontal coordinate of the start point
s32Y	Vertical coordinate of the start point
u32Width	Width of the rectangle
u32Height	Height of the rectangle

[Note]

None

[See Also]

VPSS_CROP_INFO_S

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VPSS_CROP_INFO_S

[Description]

Defines the information required by the crop function.

[Syntax]

```
typedef struct
{
    HI_BOOL bEnable;
    VPSS_CROP_COORDINATE_E  enCropCoordinate;
    RECT_S  stCropRect;
    VPSS_CAPSEL_E enCapSel;
}VPSS_CROP_INFO_S;
```

[Member]

Member	Description
bEnable	Crop enable
enCropCoordinate	Coordinate type of the crop start point
stCropRect	Crop rectangle area
enCapSel	Captured field selection

[Note]

If enCropCoordinate is VPSS_CROP_RITIO_COOR (relative coordinate mode), stCropRect must be converted when it is used. s32X is calculated as follows: s32X = coordinate of the start point/Actual image width x 1000. The value range is [0,999]. This formula is also applicable to the calculation of the vertical coordinate.

[See Also]

VPSS_CROP_COORDINATE_E

VPSS_GRP_ATTR_S

[Description]

Defines the static attributes of a VPSS groups.

[Syntax]

```
typedef struct HI_VPSS_GRP_ATTR_S
{
    HI_U32  u32MaxW;
    HI_U32  u32MaxH;
    PIXEL_FORMAT_E enPixFmt;
    VPSS_DIE_MODE_E enDieMode;
    HI_BOOL bDrEn;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_BOOL bDbEn;
    HI_BOOL bIeEn;
    HI_BOOL bNrEn;
HI_BOOL bHistEn;

}VPSS_GRP_ATTR_S;
```

[Member]

Member	Description			
	Hi3518/Hi3516C	**Hi3521/Hi3520A**	**Hi3531/ Hi3532**	**Hi3520D/Hi3515A/Hi3515C**
u32MaxW	Maximum picture width Value range: [32, 2048] It is a static attribute. It cannot be changed after it is set when a group is created.	Maximum picture width Value range: [32. 4096] It is a static attribute. It cannot be changed after it is set when a group is created.	Maximum picture width Value range: [32. 4096] It is a static attribute. It cannot be changed after it is set when a group is created.	Maximum picture width Value range: [48, 4080] It is a static attribute. It cannot be changed after it is set when a group is created.
u32MaxH	Maximum picture height Value range: [32, 2048] It is a static attribute. It cannot be changed after it is set when a group is created.	Maximum picture height Value range: [32, 4080] It is a static attribute. It cannot be changed after it is set when a group is created.	Maximum picture height Value range: [32, 4080] It is a static attribute. It cannot be changed after it is set when a group is created.	Maximum picture height Value range: [32, 4080] It is a static attribute. It cannot be changed after it is set when a group is created.
enPixFmt	Pixel format. Only semi-planar422 and semi-planar420 formats are supported. It is a static attribute. It cannot be changed after it is set when a group is created.			
enDieMode	DEI mode. For details, see VPSS_DIE_MODE_E.			
bDrEn	Reserved	Reserved	Dr enable	Reserved
bDbEn	Reserved	Reserved	Db enable	Reserved
bIeEn	Reserved	IE enable	IE enable	IE enable
bNrEn	NR enable			
bHistEn	Histogram enable			

[Note]

- The hardware for supporting Dr/Db, NR, and DEI is exclusive. Therefore, these functions cannot be enabled simultaneously when channel attributes are set.
- When the histogram and DEI are enabled, NR must be enabled.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- The Hi3520D/Hi3515A/Hi3515C does not perform the DEI operation for pictures whose width exceeds 960.
- u32MaxW, u32MaxH, and enPixFmt define the maximum size of the picture received by the current VPSS group. The values of u32MaxW, u32MaxH, and enPixFmt are set when a VPSS group is created and cannot be changed after they are set.
- If encoding is performed at the back end, the histogram switch must be enabled. Otherwise, the bit rate cannot be controlled accurately.

[See Also]

- PIXEL_FORMAT_E
- VPSS_DIE_MODE_E

VPSS_FRAME_S

[Description]

Defines the attributes of the video frames.

[Syntax]

```
typedef struct HI_VPSS_FRAME_S
{
    HI_U32  u32Width[4];
    HI_U32  u32Color[4];
}VPSS_FRAME_S;
```

[Member]

Member	Description
u32Width[4]	Frame width
u32Color[4]	Frame color

[Note]

- The following describes the mapping between the u32Width[4] subscript and four frames:

  u32Width[0]: left frame

  u32Width[1]: right frame

  u32Width[2]: bottom frame

  u32Width[3]: top frame
- The frame width must be an even number ranging from 0 to 14.
- The frame color is in RGB format. The R, G, and B components correspond to the following bits as follows

  31 23 1570

  |--------------|--------R------|--------G------|--------B------|
- The four frames of the Hi3520D/Hi3515A/Hi3515C must be in the same color.

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VPSS_CHN_ATTR_S

VPSS_CHN_ATTR_S

[Description]

Defines the static attributes of a VPSS physical channel.

[Syntax]

```
typedef struct HI_VPSS_CHN_ATTR_S
{
    HI_BOOL bSpEn;
    HI_BOOL bFrameEn;
    VPSS_FRAME_S  stFrame;
}VPSS_CHN_ATTR_S;
```

[Member]

Member	Description
bSpEn	SP enable
bFrameEn	Frame enable
stFrame	Frame attributes

[Note]

None

[See Also]

VPSS_FRAME_S

VPSS_GRP_PARAM_S

[Description]

Defines the advanced VPSS attributes.

[Syntax]

```
typedef struct HI_VPSS_GRP_PARAM_S
{
    HI_U32 u32Luminance;
    HI_U32 u32Contrast;
    HI_U32 u32DarkEnhance;
    HI_U32 u32BrightEnhance;
    HI_U32 u32IeStrength;
    HI_U32 u32IeSharp;
    HI_U32 u32SfStrength;
    HI_U32 u32TfStrength;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_U32 u32MotionThresh;
    HI_U32 u32DiStrength;
    HI_U32 u32ChromaRange;
    HI_U32 u32NrWforTsr;
    VPSS_SF_WINDOW_E enSfWindow;
    VPSS_DISPLAY_MODE_E enDisMode;
}VPSS_GRP_PARAM_S;
```

[Member]

Member	Description			
	Hi3518/Hi3516C	**Hi3521/Hi3520A**	**Hi3531/Hi3532**	**Hi3520D/Hi3515A/ Hi3515C**
u32Luminance	Reserved	Reserved	Luminance The value ranges from 0 to 48, and the default value is 32. This value affects the dark region range. A larger value indicates a narrower dark region range.	Reserved
u32Contrast	Reserved	Reserved	Contrast The value ranges from 0 to 48, and the default value is 8. This value affects the adjustment amplitude of the dark region and bright region. A larger value indicates more obvious adjustment effect.	Contrast The value ranges from 0 to 255, and the default value is 64. This value affects the adjustment amplitude of the dark region and bright region. A larger value indicates more obvious adjustment effect.
u32DarkEnhance	Reserved	Reserved	Dark region enhancement The value ranges from 0 to 48, and the default value is 16. This value affects the detail enhancement strength of the dark region. A larger value indicates more obvious enhancement effect.	Reserved
u32BrightEnhance	Reserved	Reserved	Bright region enhancement The value ranges from	Reserved

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description			
	Hi3518/Hi3516C	Hi3521/Hi3520A	Hi3531/Hi3532	Hi3520D/Hi3515A/ Hi3515C
			0 to 48, and the default value is 16. This value affects the detail enhancement strength of the bright region. A larger value indicates more obvious enhancement effect.	
u32IeStrength	Reserved	IE strength The value ranges from 0 to 8, and the default value is 2. This value affects the contrast. A larger value indicates more obvious contrast.	IE strength The value ranges from 0 to 255, and the default value is 32. This value affects the contrast. A larger value indicates more obvious contrast.	IE strength The value ranges from 0 to 255, and the default value is 6. This value affects the contrast. A larger value indicates more obvious contrast.
u32IeSharp	Reserved	IE sharpness The value ranges from 0 to 100, and the default value is 10. This value affects the contrast. A smaller value indicates more obvious contrast.	IE sharpness The value ranges from 0 to 7, and the default value is 7. This value affects the contrast. A larger value indicates more obvious contrast.	Reserved
u32SfStrength	Strength of spatial-domain noise reduction (NR) The value ranges from 0 to 255, and the default value is 32. This value affects the strength of the spatial-domain filtering strength. A larger value indicates greater filtering strength.	Reserved	Strength of spatial-domain NR The value ranges from 0 to 7, and the default value is 3. This value affects the strength of the spatial-domain filtering strength. A larger value indicates greater filtering strength.	Strength of spatial-domain NR The value ranges from 0 to 255, and the default value is 16. This value affects the strength of the spatial-domain filtering strength. A larger value indicates greater filtering strength.
u32TfStrength	Strength of time-domain NR	Reserved	Strength of time-domain NR	Strength of time-domain NR

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description			
	Hi3518/Hi3516C	Hi3521/Hi3520A	Hi3531/Hi3532	Hi3520D/Hi3515A/ Hi3515C
	The value ranges from 0 to 63, and the default value is 8. This value affects the strength of the time-domain filtering strength. A larger value indicates greater filtering strength.		The value ranges from 0 to 15, and the default value is 1. This value affects the strength of the time-domain filtering strength. A larger value indicates greater filtering strength.	The value ranges from 0 to 31, and the default value is 4. This value affects the strength of the time-domain filtering strength. A larger value indicates greater filtering strength.
u32MotionThresh	Reserved	Reserved	Motion judgment threshold The value ranges from 0 to 7, and the default value is 1. This value affects the motion regions involved in filtering. A larger value indicates that motion regions are involved in filtering more easily.	Filtering strength of channel 0 and channel 1 The value ranges from 0 and 8, and the default value is 2. This value affects the filtering strength of encoding channels. A larger value indicates greater noise reduction strength.
u32DiStrength	Reserved	DEI strength The value ranges from 0 to 15, and the default value is 0. This value affects the de-interlace effect. A larger value indicates that the de-interlace effect is more obvious, but the screen also flickers more obvious.	DEI strength The value ranges from 0 to 7, and the default value is 7. This value affects the de-interlace effect. A larger value indicates that the de-interlace effect is more obvious, but the screen also flickers more obvious.	Reserved
u32ChromaRange	Chrominance amplitude The value ranges from 0 to 255, and the default value is 8. This value affects the strength of the chrominance	Reserved	Chrominance amplitude The value ranges from 0 to 31, and the default value is 4. This value affects the strength of the chrominance filtering strength. A larger value	Reserved

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description			
	Hi3518/Hi3516C	Hi3521/Hi3520A	Hi3531/Hi3532	Hi3520D/Hi3515A/ Hi3515C
	filtering strength. A larger value indicates greater filtering strength.		indicates greater filtering strength.	
u32NrWforTsr	Reserved	Reserved	Time-domain filtering weight The value ranges from 0 to 8, and the default value is 8. This value affects the filtering strength. A larger value indicates larger time-domain filtering weight.	Reserved
enSfWindow	Reserved	Reserved	Size of the spatial domain filtering window The value ranges from 0 to 4, and the default value is 0. This value affects the filtering strength. A larger window size indicates greater filtering strength.	Reserved
enDisMode	Reserved	Reserved	Display mode The value is 0 or 1, and the default value is 0. This value affects display luminance output. In TV mode, the luminance value ranges from 16 to 235; in PC mode, the luminance value ranges from 0 to 255.	Reserved

[Note]

The member values vary depending to chip type. For details, see the member descriptions.

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VPSS_CHN_MODE_E

[Description]

Defines the working mode of a VPSS channel.

[Syntax]

```
typedef enum HI_VPSS_CHN_MODE_E
{
    VPSS_CHN_MODE_AUTO,
    VPSS_CHN_MODE_USER
}VPSS_CHN_MODE_E;
```

[Member]

Member	Description
VPSS_CHN_MODE_AUTO	Automatic mode. This mode is the default working mode of channels.
VPSS_CHN_MODE_USER	User-defined mode

[Note]

None

[See Also]

None

VPSS_CHN_MODE_S

[Description]

Defines the structure of the working mode of a VPSS channel.

[Syntax]

```
typedef struct HI_VPSS_CHN_MODE_S
{
    VPSS_CHN_MODE_E  enChnMode;
    HI_U32 u32Width;
    HI_U32 u32Height;
    HI_BOOL bDouble;
    PIXEL_FORMAT_E  enPixelFormat;
}VPSS_CHN_MODE_S;
```

[Member]

Member	Description
enChnMode	Working mode of a VPSS channel

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u32Width	Target picture width
u32Height	Target picture height
bDouble	Whether to double the frame rate. It is valid only for VPSS_PRE0_CHN.
enPixelFormat	Pixel format of the target picture

[Note]

The value ranges of u32Width and u32Height must comply with the VPSS processing capability. For details, see VPSS_GRP_ATTR_S. The values of u32Width and u32Height must be evens.

[See Also]

None

VPSS_CHN_NR_PARAM_S

[Description]

Defines the advanced NR parameters for a channel.

[Syntax]

```
typedef struct
{
    HI_U32 u32SfStrength;
    HI_U32 u32TfStrength;
    HI_U32 u32MotionThresh;
    HI_U32 u32ChromaRange;
    HI_U32 u32NrWforTsr;
}VPSS_CHN_NR_PARAM_S;
```

[Member]

Member	Description
u32SfStrength	Strength of spatial-domain filtering The value ranges from 0 to 16 and the default value is 16. The greater the value, the greater the filtering strength is.
u32TfStrength	Strength of time-domain filtering The value ranges from 0 to 16 and the default value is 8. The greater the value, the greater the filtering strength is.
u32MotionThresh	Reserved
u32ChromaRange	Reserved

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u32NrWforTsr	Reserved

[Note]

None

[See Also]

None

VPSS_CHN_SP_PARAM_S

[Description]

Defines the advanced SP parameter for a channel.

[Syntax]

```
typedef struct
{
  HI_U32 u32LumaGain;
}VPSS_CHN_SP_PARAM_S;
```

[Member]

Member	Description	
	Hi3521	Hi3520D/Hi3515A/Hi3515C
u32LumaGain	SP strength The value ranges from 0 to 255 and the default value is 40. The greater the value, the greater the SP strength is.	SP strength The value ranges from 0 to 255 and the default value is 64. The greater the value, the greater the SP strength is.

[Note]

None

[See Also]

None

VPSS_CHN_PARAM_S

[Description]

Defines VPSS channel advanced attributes.

[Syntax]

```
typedef struct
```


Copyright © HiSilicon Technologies Co., Ltd

```
{
  VPSS_CHN_SP_PARAM_S  stVpssChnSpParam;
  VPSS_CHN_NR_PARAM_S  stVpssChnNrParam;
} VPSS_CHN_PARAM_S;
```

[Member]

Member	Description
stVpssChnSpParam	Advanced SP attribute of a channel
stVpssChnNrParam	Advanced NR attribute of a channel

[Note]

None

[See Also]

- VPSS_CHN_SP_PARAM_S
- VPSS_CHN_NR_PARAM_S

VPSS_CAPSEL_E

[Description]

Defines the VPSS captured field type.

[Syntax]

```
typedef enum hiVPSS_CAPSEL_E
{
    VPSS_CAPSEL_BOTH = 0,
    VPSS_CAPSEL_TOP,
    VPSS_CAPSEL_BOTTOM,
    VPSS_CAPSEL_BUTT
} VPSS_CAPSEL_E;
```

[Member]

Member	Description
VPSS_CAPSEL_BOTH	Capture both top and bottom fields.
VPSS_CAPSEL_TOP	Capture the top field.
VPSS_CAPSEL_BOTTOM	Capture the bottom field.

[Note]

None

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

VPSS_PRESCALE_INFO_S

[Description]

Defines the VPSS prescaling attribute.

[Syntax]

```
typedef struct
{
    HI_BOOL bPreScale;
    VPSS_CAPSEL_E  enCapSel;
    SIZE_S stDestSize;
}VPSS_PRESCALE_INFO_S;
```

[Member]

Member	Description
bPreScale	Pre-scaling enable
enCapSel	Captured field select
stDestSize	Target picture size. The size cannot be greater than the maximum group size.

[Note]

None

[See Also]

None

VPSS_SIZER_INFO_S

[Description]

Defines the VPSS size filtering attribute.

[Syntax]

```
typedef struct
{
    HI_BOOL bSizer;
    SIZE_S  stSize;
}VPSS_SIZER_INFO_S;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
bSizer	Enable for picture filtering by size
SIZE_S	Picture size for filtering out pictures

[Note]

None

[See Also]

None

VPSS_FRAME_RATE_S

[Description]

Defines the VPSS frame rate control attribute.

[Syntax]

```
typedef struct
{
    HI_S32  s32SrcFrmRate;
    HI_S32  s32DstFrmRate;
} VPSS_FRAME_RATE_S;
```

[Member]

Member	Description
s32SrcFrmRate	Source frame rate
s32DstFrmRate	Target frame rate

[Note]

- If both the source frame rate and target frame rate is −1, the frame rate is not controlled.
- The target frame rate must be less than or equal to the source frame rate.

[See Also]

None

VPSS_EXT_CHN_ATTR_S

[Description]

Defines the attribute of an extended VPSS channel.

[Syntax]

```
typedef struct HI_VPSS_EXT_CHN_ATTR_S
```


Copyright © HiSilicon Technologies Co., Ltd

```
{
    VPSS_CHN  s32BindChn;
    HI_U32 u32Width;
    HI_U32 u32Height;
    HI_S32 s32SrcFrameRate;
    HI_S32 s32DstFrameRate;
    PIXEL_FORMAT_E enPixelFormat;
} VPSS_EXT_CHN_ATTR_S;
```

[Member]

Member	Description
s32BindChn	Physical channel to be bound
u32Width	Target output width of an extended channel
u32Height	Target output height of an extended channel
s32SrcFrameRate	Source input frame rate of an extended channel
s32DstFrameRate	Target output frame rate of an extended channel
enPixelFormat	Output picture format of an extended channel

[Note]

- The ID of the physical channel to be bound must fall within [0, VPSS_MAX_PHY_CHN_NUM).
- The channel width and height must be 2-pixel-aligned.
- The target frame rate must be less than or equal to the source frame rate. If both the target frame rate and source frame rate are −1, the frame rate is not controlled.
- The output data format can be semi-planar YUV420 or semi-planar YUV422.

[See Also]

None

5.5 Error Codes

Table 5-5 describes the error codes of the VPSS APIs.

Table 5-5 Error codes of VPSS APIs

Error Code	Macro Definition	Description
0xA0088001	HI_ERR_VPSS_INVALID_DEVID	The number of VPSS groups is invalid.
0xA0088002	HI_ERR_VPSS_INVALID_CHNID	The VPSS channel ID is invalid.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Macro Definition	Description
0xA0088003	HI_ERR_VPSS_ILLEGAL_PARAM	The VPSS parameters are invalid.
0xA0088004	HI_ERR_VPSS_EXIST	A VPSS group is created already.
0xA0088005	HI_ERR_VPSS_UNEXIST	No VPSS group is created.
0xA0088006	HI_ERR_VPSS_NULL_PTR	The pointer of the input parameter is null.
0xA0086008	HI_ERR_VPSS_NOT_SUPPORT	The operation is not supported.
0xA0088009	HI_ERR_VPSS_NOT_PERM	The operation is forbidden.
0xA008800C	HI_ERR_VPSS_NOMEM	The memory fails to be allocated.
0xA008800D	HI_ERR_VPSS_NOBUF	The buffer pool fails to be allocated.
0xA0088010	HI_ERR_VPSS_NOTREADY	The VPSS system is not initialized.
0xA0088012	HI_ERR_VPSS_BUSY	The VPSS system is busy.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Contents

6 VENC 6-1
6.1 Overview 6-1
6.2 Functions 6-2
6.2.1 Data Encoding Flowchart 6-2
6.2.2 VENC Channels and Channel Group 6-3
6.2.3 Bit Rate Control 6-3
6.2.4 Frame Skipping Reference Modes 6-4
6.2.5 Advanced Frame Skipping Reference Modes 6-5
6.2.6 Color-to-Gray 6-6
6.2.7 Clipping Encoding 6-6
6.2.8 ROI 6-7
6.2.9 JPEG Snapshot Modes 6-8
6.2.10 Configuration Mode of the Encoding Stream Buffer 6-8
6.2.11 Allocation Mode of the Buffers for Storing Encoded Frames 6-9
6.2.12 Memory Allocation Mode for the Reference Frame and Encoding Reconstruction Frame 6-10
6.2.13 Configuration Mode of the Encoding Stream Buffer 6-11
6.3 API Reference 6-12
6.4 Data Types 6-113
6.5 Error Codes 6-178

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figures

Figure 6-1 Data encoding of the VENC module 6-2
Figure 6-2 Function division of a VENC channel 6-3
Figure 6-3 Schematic diagram of frame skipping reference modes 6-5
Figure 6-4 Schematic diagram of advanced frame skipping reference modes 6-6
Figure 6-5 Clipping encoding 6-7
Figure 6-6 ROI 6-8
Figure 6-7 Configuration mode of the encoding stream buffer 6-9
Figure 6-8 Checking whether the stream buffer is wrapped 6-105

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Tables

Table 6-1 Encoding formats....6-1

Table 6-2 Number of frame buffers that are spared when the UserVB pool mode is used during *n*-channel encoding....6-10

Table 6-3 Operations when H264eVBSource and H264eRcnEqualRef are set to different values....6-11

Table 6-4 Widths and heights of VENC channels....6-19

Table 6-5 Limitations on encoder attributes....6-21

Table 6-6 Public attributes of three RC modes....6-21

Table 6-7 Typical configuration of the average bit rate....6-22

Table 6-8 Search widow size....6-52

Table 6-9 Sizes of the search window of the Hi3531/Hi3532 H.264 encoder under various resolutions....6-52

Table 6-10 Sizes of the search window of the Hi3521/Hi3520A H.264 encoder under various resolutions....6-53

Table 6-11 Sizes of the search window of the Hi3518/Hi3516C H.264 encoder under various resolutions....6-53

Table 6-12 Default sizes of the search windows of the Hi3520D/Hi3515A/Hi3515C H.264 encoder under various resolutions....6-53

Table 6-13 Default trans parameter values for different profile types....6-61

Table 6-14 Default entropy encoding parameter values for different profile types....6-64

Table 6-15 Error codes for the VENC APIs....6-178

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

6 VENC

6.1 Overview

The video encoding (VENC) module supports real-time encoding on multiple channels. Each VENC channel is independent. The encoding protocols and encoding profiles of VENC channels may be different. The VENC module can also schedule the region module to overlay and cover the contents of the encoded pictures. The input sources of the VENC module include:

- Pictures that are read in user state.
- Pictures that are collected by the VIU and then processed by the VPSS.
- Pictures that are collected by the VIU.

Table 6-1 lists the encoding formats supported by each chip.

Table 6-1 Encoding formats

Chip	H.264				JPEG	M JPEG	MPEG-4
	Baseline Profile	Main Profile	High Profile	SVC-T			
Hi3531/ Hi3532	Supported	Supported	Supported	Not supported	Supported	Supported	Supported
Hi3521/ Hi3520A	Supported	Supported	Not supported	Not supported	Supported	Supported	Not supported
Hi3518/Hi3516C	Supported	Supported	Supported	Supported	Supported	Supported	Not supported
Hi3520D/Hi3515A/Hi3515C	Supported	Supported	Supported	Not supported	Supported	Supported	Not supported

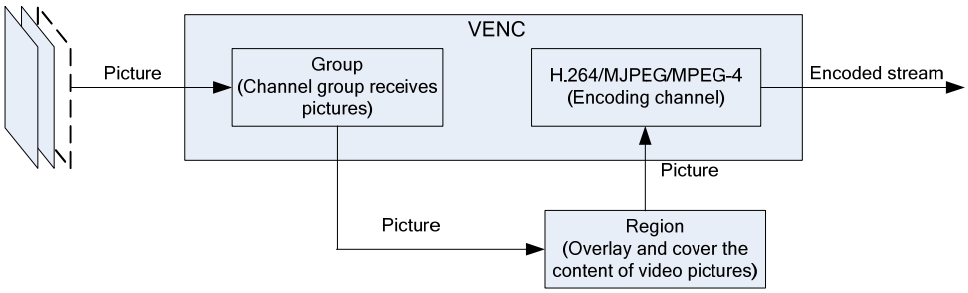
HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

6.2 Functions

6.2.1 Data Encoding Flowchart

Figure 6-1 shows data encoding of the VENC module.

Figure 6-1 Data encoding of the VENC module



A typical encoding procedure includes receiving input pictures, covering and overlaying pictures, encoding pictures, and outputting streams.

The VENC module consists of a VENC channel group submodule (GROUP) and an encoding protocol submodule (H.264, JPEG, MJPEG, and MPEG-4 are supported).

The channel group submodule can receive YUV pictures in the format of semi-planar YUV 4:2:0 or semi-planar YUV 4:2:2. For the H264E and MPEG-4 protocols, only the semi-planar YUV 4:2:0 format is supported; for the JPEG and MJPEG protocols, the semi-planar YUV 4:2:0 or semi-planar YUV 4:2:2 format is supported. The channel group submodule receives external raw picture data regardless of the picture source.

NOTE

A channel group allows only the VPSS, VIU, or users to transmit pictures. The Hi3518/Hi3516C supports the JPEG flash snapshot mode. When the Hi3518/Hi3516C JPEG channel is set to this mode, the channel group receives pictures only from the VIU (pictures are not transmitted to the VPSS). Otherwise, the JPEG channel cannot capture pictures.

After receiving a picture, the channel group compares the picture size with the VENC channel size. The VENC module performs operations in the following cases:

- If the input picture size is greater than the VENC channel size, the VENC module zooms out on the picture by using the dedicated scaling unit (DSU) to fit into the VENC channel, and then encodes the picture.
- If the input picture size is smaller than the VENC channel size, the VENC module discards the picture. The VENC module does not support zoom in.
- If the input picture size is the same as the VENC channel size, the VENC module encodes the picture directly.

NOTE

The frame rate control function of the channel group is disabled by default. You can enable this function by calling related MPIs. The rate controller (RC) also supports frame rate control. The frame rate control function of the RC is recommended, which ensures that the bit rate does not change severely.

A REGION module can cover and overlay pictures.

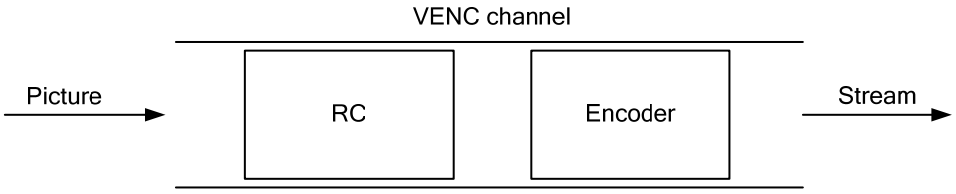
After video pre-processing and region management, the picture is sent to channel complying with a specific protocol, completing video encoding and stream output.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

6.2.2 VENC Channels and Channel Group

A VENC channel is a basic container that stores user configuration of various VENC channels and manages the internal resources on various VENC channels. A VENC channel converts pictures into streams. This operation is completed by an RC and an encoder. The encoder completes only the encoding function; therefore, it is an encoder in narrow sense. The RC controls and adjusts the encoding parameters, controlling the output bit rate.

Figure 6-2 Function division of a VENC channel



VENC channels can be categorized into primary VENC channels and secondary VENC channels. The VENC supports only one large stream encoding channel.

A VENC channel group is a set of VENC channels that operate concurrently. Only one large stream channel (H.264, JPEG, MJPEG, and MPEG-4 are supported) can be encoded when the VENC channel group is enabled once. Therefore, when VENC channels are created, only one large stream channel can be registered.

6.2.3 Bit Rate Control

An RC mainly controls the encoding bit rate.

From the aspect of informatics, lower compression rate of a picture obtains higher picture quality. In actual scenarios, when the picture quality is stable, the encoding bit rate may fluctuate; when the encoding bit rate is stable, the picture quality may change. The following is an example using H.264 encoding. The protocol evaluates the picture quality based on the quantizer parameter (QP). Typically, when the QP value becomes small, the picture quality is improved and the encoding bit rate increases; when the QP value becomes large, the picture quality deteriorates and the encoding bit rate decreases.

Rate control must be specific to consecutive encoding streams. Therefore, the JPEG encoding channel does not support bit rate control.

The RC provides three modes for the H.264, MPEG-4, or MJPEG encoding channel to adjust the picture quality and bit rate:

- Constant bit rate (CBR) mode
- Variable bit rate (VBR) mode
- FixQp mode

As the Hi3521, Hi3520A, and Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C do not support MPEG-4 encoding, they also cannot control the bit rate of MPEG-4 streams.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

CBR

CBR means that a stable encoding bit rate is ensured within the bit rate statistical time. A stable bit rate is evaluated by the following two indexes which can be specified by users when a VENC channel is created.

- Bit rate statistical time

  The statistical time unit is second. A longer statistical time indicates that the impact on bit rate adjustment exerted by the bit rate change of each frame is less, the bit rate is adjusted more slowly, and the picture quality changes more slightly.

- Fluctuation level for the bit rate

  The system provides six fluctuation levels for the bit rate. A higher level indicates a wider range of bit rate fluctuation. If the fluctuation level is high, the picture quality is more stable when the picture contents of adjacent frames vary significantly. The high level applies to the scenario in which the bandwidth margin is large. If the fluctuation level is low, the encoding bit rate is more stable. When the picture contents of adjacent frames vary significantly, the picture quality at a low level may be worse than that at a high level. The low level applies to the scenario in which the bandwidth margin is small.

VBR

VBR means that a stable picture quality is ensured within the bit rate statistical time when the bit rate changes. The following is an example using H.264 encoding. The VENC module allows you to set the MaxQp, MinQp, and MaxBitrate parameters. MaxQp and MinQp are used to control picture quality and MaxBitrate is used to clamp the maximum encoding bit rate within the bit rate statistical time. When the encoding bit rate is close to the maximum bit rate, the picture QP value is close to the value of MaxQp. When the encoding bit rate is far less than the maximum bit rate, the picture QP value is adjusted close to the value of MinQp. When the picture QP value reaches the value of MaxQp, the MaxBirate parameter becomes invalid. In this case, the encoding bit rate may exceed the value of MaxBirate. When the picture QP value reaches the value of MinQp, the encoding bit rate is the maximum, and the picture quality is the best.

FixQp

FixQp is a fixed QP value. Within the bit rate statistics, FixQp indicates that the QP values of all the macroblocks of encoded pictures are the same, the user-defined QP value is used, and the QP values of I frame and P frame can be set separately.

6.2.4 Frame Skipping Reference Modes

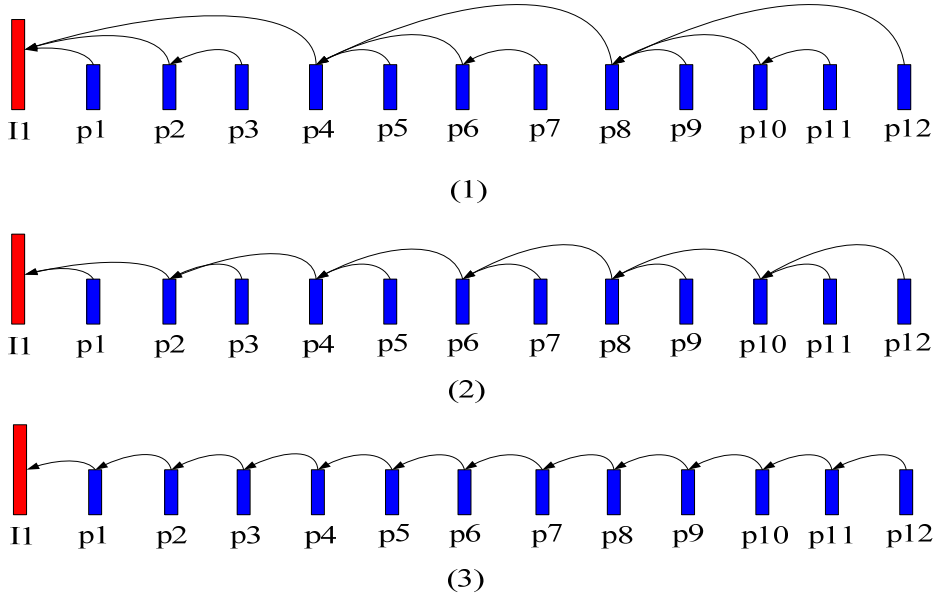
Three frame skipping reference modes are supported for H.264 encoding: 1x frame skipping reference mode, 2x frame skipping reference mode, and 4x frame skipping reference mode. As shown in Figure 6-3, note the following:

- Part (1) is the schematic diagram of the 4x frame skipping reference mode. I1 is an I frame, and P2 to P12 are P frames. The frames pointed by arrows such as I1, P2 and P4 are reference frames. The other frames such as P1 and P3 are non-reference frames. In this mode, you can save all reference frames, all frames, or only the frames that are skipped by four frames such as I1, P4, and P8.
- Part (2) is the schematic diagram of the 2x frame skipping reference mode. The frames pointed by arrows such as I1, P2 and P4 are reference frames. The other frames such as P1 and P3 are non-reference frames. In this mode, you can save all reference frames or all frames.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Part (3) is the schematic diagram of the 1x frame skipping reference mode. All frames are reference frames, and you must save all reference frames.

Figure 6-3 Schematic diagram of frame skipping reference modes

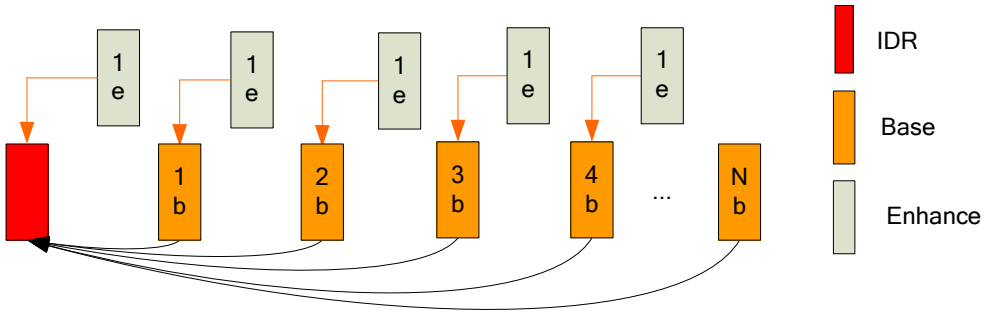


6.2.5 Advanced Frame Skipping Reference Modes

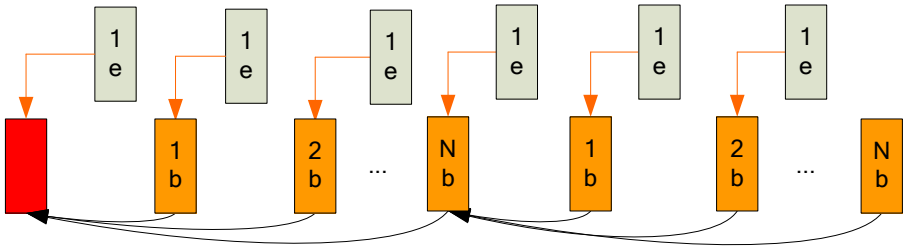
The u32Base, u32Enhance, and bEnablePred parameters are involved in advanced frame skipping reference modes. For details about these parameters, see the description of VENC_ATTR_H264_REF_PARAM_S. The advanced frame skipping reference modes are compatible with the reference modes described in section 6.2.4 "Frame Skipping Reference Modes." Figure 6-4 shows the schematic diagram of advanced frame skipping reference modes.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

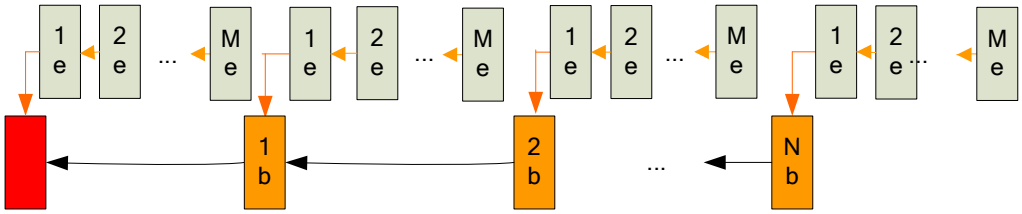
Figure 6-4 Schematic diagram of advanced frame skipping reference modes



(N, 1, HI_FALSE)



(N, 1, HI_TRUE)



(1, M, HI_TRUE)

6.2.6 Color-to-Gray

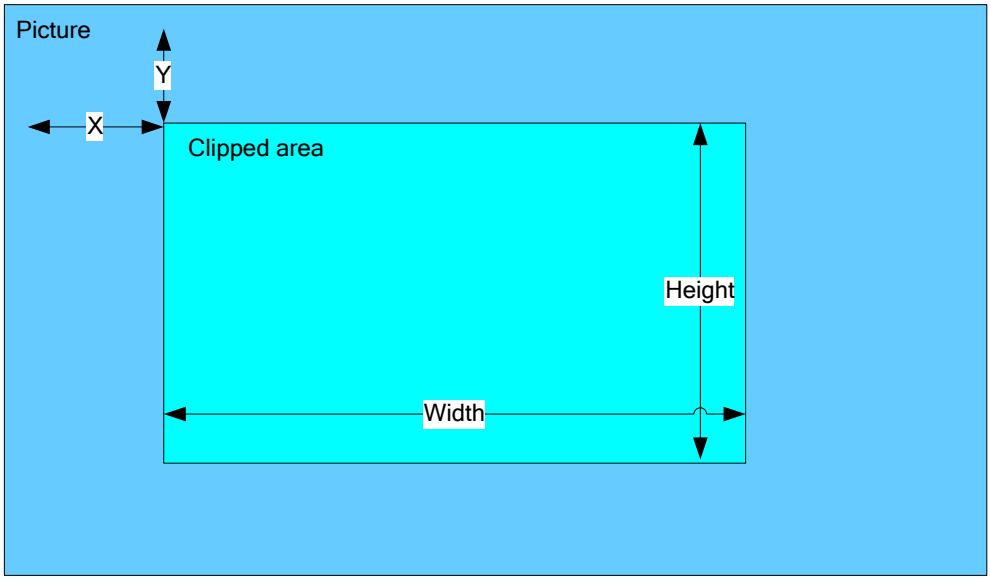
The color-to-gray function enables the VENC module to convert color pictures into gray pictures for encoding. For details, see the descriptions of HI_MPI_VENC_SetColor2GreyConf and HI_MPI_VENC_SetGrpColor2Grey.

6.2.7 Clipping Encoding

Clipping encoding indicates that the VENC clips a part of a picture for encoding. You can set the start position (X and Y), width, and height. For details, see the HI_MPI_VENC_GetGrpCrop and HI_MPI_VENC_SetGrpCrop APIs.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 6-5 Clipping encoding



6.2.8 ROI

You can set a region of interest (ROI) to control the picture QP value of the region, distinguishing this region from other regions. The system allows you to set the ROI only for the H.264 channel and provides eight ROIs.

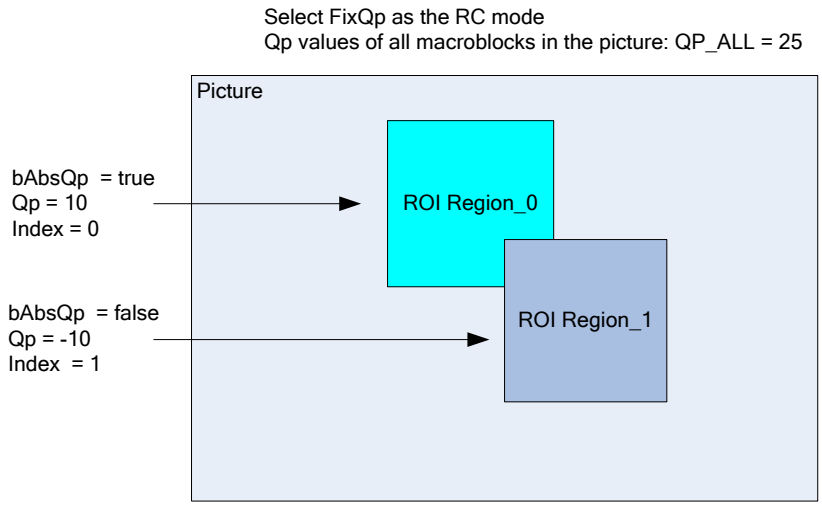
The eight regions can be overlaid with each other and the overlay priority ranges from 0 to 7. The overlay priority indicates the final QP value of picture regions when the regions are overlaid. The final QP value is set according to the region with the highest priority. An ROI can be configured with an absolute QP and a relative QP.

- Absolute QP: indicates the QP value of an ROI set by users.
- Relative QP: indicates the QP value generated during bit rate control plus the offset of the QP values set by users.

Figure 6-6 shows an encoded picture in FixQp mode. The picture QP value is 25, that is, the QP values of all macroblocks of the picture are 25. ROI 0 is configured with absolute QP mode, the QP value is 10, and the index is 0; ROI 1 is configured with relative QP mode, the QP value is –10, and the index is 1. When the two regions are overlaid, the QP is set according to the region with the highest priority (ROI 1). Except the overlaid part, the QP of ROI 0 is 10 and therefore the QP of ROI 1 is 15 (25 – 10).

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 6-6 ROI



6.2.9 JPEG Snapshot Modes

The JPEG VENC channels of the Hi3518/Hi3516C support two modes: snapshot all mode and flash snapshot mode. The JPEG VENC channels of the Hi3531, Hi3532, Hi3521, Hi3520A, Hi3520D, Hi3515A, or Hi3515C support only the snapshot all mode.

- Snapshot all mode: After the channels start to receive pictures, all received pictures are encoded.
- Flash snapshot mode: After the channels start to receive channels, only the pictures captured when the camera flash is on are encoded.

NOTE

When the Hi3518/Hi3516C JPEG channel is set to the flash snapshot mode, the channel group receives pictures only from the VIU (pictures are not transmitted to the VPSS). Otherwise, the JPEG channel cannot capture pictures.

6.2.10 Configuration Mode of the Encoding Stream Buffer

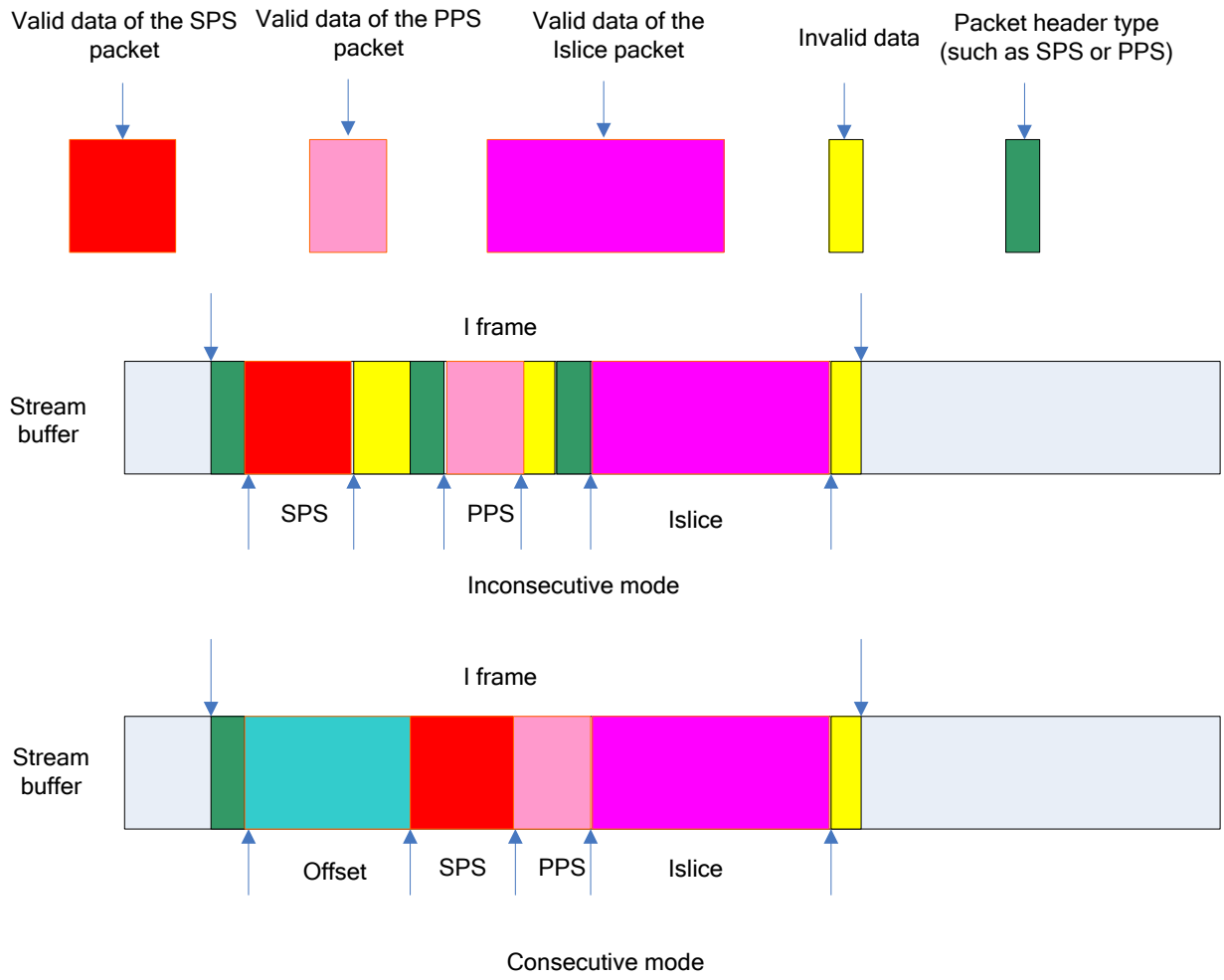
As shown in Figure 6-7, the encoding stream buffer supports the inconsecutive configuration mode and consecutive configuration mode.

- Inconsecutive mode: When frames are I frames, HI_MPI_GetStream() is called. An I frame contains a sequence parameter set (SPS) packet, a picture parameter set (PPS) packet, and an Islice packet (assuming that there is only one PPS packet). The addresses for packets are inconsecutive. When the stream buffer is wrapped, a frame has two inconsecutive addresses (pstPack[0].pu8Addr[0] + pstPack[0].u32Len[0] ≠ pstPack[0].pu8Addr[1]).
- Consecutive mode: When frames are I frames, HI_MPI_GetStream() is called. An I frame contains an Islice packet, and the frame addresses are consecutive (that is, pstPack[0].pu8Addr[0] + pstPack[0]. u32Len[0] = pstPack[0].pu8Addr[1] anytime).

The stream buffer configuration mode can be set by configuring the **ContiguousBuffer** parameter when .ko drivers are loaded. If **ContiguousBuffer** is **1**, the mode is consecutive mode; if **ContiguousBuffer** is **0**, the mode is inconsecutive mode.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 6-7 Configuration mode of the encoding stream buffer



6.2.11 Allocation Mode of the Buffers for Storing Encoded Frames

The buffers for storing encoded frames (reference frame and reconstruction frame) can be allocated in PrivateVB pool mode or UserVB pool mode.

- PrivateVB pool mode: When a VENC channel is created, the VENC module creates private video buffer (VB) pools as the buffers for reference frames and reconstruction frames.
- UserVB pool (supported only during H.264 encoding): When a VENC channel is created, the buffer for reference frames and reconstruction frames is not allocated. You need to create a VB pool by calling HI_MPI_VB_CreatePool, and then bind a VENC channel to a fixed VB pool by calling HI_MPI_VENC_AttachVbPool.

The allocation mode can be specified by setting **H264eVBSource** when **h264e.ko** is loaded. The PrivateVB pool mode is used by default. If **H264eVBSource** is set to **1**, the encoding PrivateVB pool mode is used; if **H264eVBSource** is set to **2**, the encoding UserVB pool mode is used.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

When the PrivateVB pool mode is used, the VENC channels are irrelevant to each other and destroying one VENC channel has no effect on other VENC channels.

Table 6-2 describes the number of frame buffers that are spared when the UserVB pool mode is used during *n*-channel encoding.

Table 6-2 Number of frame buffers that are spared when the UserVB pool mode is used during *n*-channel encoding

Chip	Number of VEDUs	Number of Spared Frame Buffers
Hi3531, Hi3532	2	$n - 4$ ($n > 4$)
Hi3521, Hi3520A, Hi3520D, Hi3515A, Hi3515C	1	$n - 2$ ($n > 2$)

If the UserVB pool mode is used, you need to create a VB pool, and then bind *n* channels to this VB pool.

The number of VBs is calculated as follows:

BlkCnt = *n* + 2 x VEDU. If the 4x frame skipping reference mode, advanced frame skipping reference mode, or SVC-T encoding is configured for a channel, an extra frame buffer is required for this channel.

The size of a VB is calculated as follows:

BlkSize = align(PicWidth,16) x align(PicHeight,16) x 3/2

6.2.12 Memory Allocation Mode for the Reference Frame and Encoding Reconstruction Frame

The reference frame can share the memory for storing the luminance data with the encoding reconstruction frame. The memory for storing the chrominance data needs to be allocated for the reconstruction frame and reference frame separately. The memory allocation mode for the reference frame and reconstruction frame can be specified by setting **H264eRcnEqualRef** when **h264e.ko** is loaded. If **H264eRcnEqualRef** is set to **0**, the memory is allocated separately for the reference frame and reconstruction frame by default. If **H264eRcnEqualRef** is set to **1**, the reference frame shares the memory for storing the luminance data with the reconstruction frame.

When the reference frame shares the memory for storing the luminance data with the reconstruction frame, the following issues occur:

- The reference frame is overwritten by the reconstruction frame during encoding. Therefore, only I frames can be inserted when the jumbo frame appears, bit rate overshoot occurs, or the stream buffer is full.
- One more memory for storing the luminance data is required in 2x frame skipping reference mode.

If **H264eVBSource** is set to **2** and **H264eRcnEqualRef** is set to **1**, the UserVB pool mode is used and the reference frame shares the memory for storing the luminance data with the reconstruction frame. In this case, you need to need to create a VB pool, and then bind *n* channels to this VB pool. For details, see section 6.2.11 "Allocation Mode of the Buffers for Storing Encoded Frames." The size of a VB is calculated as follows:

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Size = align(PicWidth,16) x align(PicHeight,16)/2

Table 6-3 describes the operations when **H264eVBSource** and **H264eRcnEqualRef** are set to different values.

Table 6-3 Operations when H264eVBSource and H264eRcnEqualRef are set to different values

H264eVBSource	H264eRcnEqualRef	Operation
1	0	Set the module parameters when .ko files are loaded.
1	1	Set the module parameters when .ko files are loaded.
2	0	• Step 1 Set the module parameters when .ko files are loaded. • Step 2 Create the VB pool. For details about the number and sizes of VB pools, see the preceding descriptions. • Step 3 Bind the VENC channel to a VB pool.
2	1	• Step 1 Set the module parameters when .ko files are loaded. • Step 2 Create the VB pool. For details about the number and sizes of VB pools, see the preceding descriptions. • Step 3 Bind the VENC channel to a VB pool.

6.2.13 Configuration Mode of the Encoding Stream Buffer

The encoding stream buffer supports two configuration modes: normal mode and memory reduction mode.

- Normal mode: In consideration of jumbo frames, the minimum size of the stream buffer is (Channel width x Channel height x 3/4) for H.264 and MPEG4 encoding and (Channel width x Channel height) for JPEG and MJPEG encoding.
- Memory reduction mode: The minimum size of the stream buffer is (32 x 1024) bytes. The stream buffer size must be set to an appropriate value in this mode. Otherwise, frames may be re-encoded continuously or discarded due to the insufficiency of the stream buffer.

The mode can be specified by setting corresponding module parameters when .ko drivers are loaded. If the parameter value is 1, the memory reduction mode is used; if the parameter value is 0, the normal mode is used.

The module parameter for **hi35xx_h264e.ko** is **H264eMiniBufMode**; the module parameter for **hi35xx_mpeg4e.ko** is **Mpeg4eMiniBufMode**; the module parameter for **hi35xx_jpege.ko** is **JpegeMiniBufMode**.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

6.3 API Reference

The VENC module mainly contains the VENC (video encoding) and GROUP (channel group management), and implements the following functions: creating and destroying a VENC channel group, creating and destroying a VENC channel, registering and deregistering a VENC channel, starting and stopping receiving pictures, setting and obtaining attributes of a VENC channel, obtaining and releasing streams, and setting and querying attributes of a VENC channel

The VENC module provides the following MPIs:

- HI_MPI_VENC_CreateGroup: Creates a VENC channel group.
- HI_MPI_VENC_DestroyGroup: Destroys a VENC channel group.
- HI_MPI_VENC_CreateChn: Creates a VENC channel.
- HI_MPI_VENC_DestroyChn: Destroys a VENC channel.
- HI_MPI_VENC_RegisterChn: Registers a VENC channel to a VENC channel group.
- HI_MPI_VENC_UnRegisterChn: Deregisters a VENC channel from a VENC channel group.
- HI_MPI_VENC_StartRecvPic: Enables a VENC channel to start receiving input pictures.
- HI_MPI_VENC_StartRecvPicEx: Enables a VENC channel to start receiving input pictures and automatically stop the VENC channel from receiving input pictures after the specified number of frames is exceeded.
- HI_MPI_VENC_StopRecvPic: Stops a VENC channel from receiving input pictures.
- HI_MPI_VENC_Query: Queries the status of a VENC channel.
- HI_MPI_VENC_SetChnAttr: Sets attributes of a VENC channel.
- HI_MPI_VENC_GetChnAttr: Obtains attributes of a VENC channel.
- HI_MPI_VENC_GetStream: Obtains encoded streams.
- HI_MPI_VENC_ReleaseStream: Releases a stream buffer.
- HI_MPI_VENC_InsertUserData: Inserts user data.
- HI_MPI_VENC_SendFrame: Sends raw pictures for encoding.
- HI_MPI_VENC_SetMaxStreamCnt: Sets the maximum number of frames in a stream buffer.
- HI_MPI_VENC_GetMaxStreamCnt: Obtains the maximum number of frames in a stream buffer.
- HI_MPI_VENC_RequestIDR: Requests an I frame.
- HI_MPI_VENC_RequestIDRInst: Requests an I frame and encodes an I frame immediately.
- HI_MPI_VENC_GetFd: Obtains the device file handle of a VENC channel.
- HI_MPI_VENC_SetRoiCfg: Sets the ROI configuration of a channel.
- HI_MPI_VENC_GetRoiCfg: Obtains the ROI configuration of a channel.
- HI_MPI_VENC_SetH264SliceSplit: Sets the slice attributes of an H.264 channel.
- HI_MPI_VENC_GetH264SliceSplit: Obtains the slice attributes of an H.264 channel.
- HI_MPI_VENC_SetH264InterPred: Sets the inter-prediction attributes of an H.264 channel.
- HI_MPI_VENC_GetH264InterPred: Obtains the inter-prediction attributes of an H.264 channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_VENC_SetH264IntraPred: Sets the intra-prediction attributes of an H.264 channel.
- HI_MPI_VENC_GetH264IntraPred: Obtains the intra-prediction attributes of an H.264 channel.
- HI_MPI_VENC_SetH264Trans: Sets the transformation and quantization attributes of an H.264 channel.
- HI_MPI_VENC_GetH264Trans: Obtains the transformation and quantization attributes of an H.264 channel.
- HI_MPI_VENC_SetH264Entropy: Sets the entropy encoding mode for an H.264 channel.
- HI_MPI_VENC_GetH264Entropy: Obtains the entropy encoding mode for an H.264 channel.
- HI_MPI_VENC_SetH264Poc: Sets the POC type of an H.264 channel.
- HI_MPI_VENC_GetH264Poc: Obtains the POC type of an H.264 channel.
- HI_MPI_VENC_SetH264Dblk: Sets the de-blocking type of an H.264 channel.
- HI_MPI_VENC_GetH264Dblk: Obtains the de-blocking type of an H.264 channel.
- HI_MPI_VENC_SetH264Vui: Sets the video usability information (VUI) parameters of an H.264 channel.
- HI_MPI_VENC_GetH264Vui: Obtains the VUI parameter settings of an H.264 channel.
- HI_MPI_VENC_SetJpegParam: Sets the parameter set of a JPEG channel.
- HI_MPI_VENC_GetJpegParam: Obtains the parameter set of a JPEG channel.
- HI_MPI_VENC_SetMpeg4Param: Sets the advanced parameters of an MPEG-4 encoding channel.
- HI_MPI_VENC_GetMpeg4Param: Obtains the configurations of the advanced parameters of an MPEG-4 encoding channel.
- HI_MPI_VENC_SetMjpegParam: Sets the advanced parameters of a JPEG encoding channel.
- HI_MPI_VENC_GetMjpegParam: Obtains the configurations of the advanced parameters of a JPEG encoding channel.
- HI_MPI_VENC_SetGrpFrmRate: Sets the frame rate control attributes of the channel group.
- HI_MPI_VENC_GetGrpFrmRate: Obtains the frame rate control attributes of the channel group.
- HI_MPI_VENC_SetRcPara: Sets the advanced parameters for the RC of a VENC channel.
- HI_MPI_VENC_GetRcPara: Obtains the configurations of the advanced parameters for the RC of a VENC channel.
- HI_MPI_VENC_SetH264eRefMode: Sets the frame skipping reference mode of an H.264 VENC channel.
- HI_MPI_VENC_GetH264eRefMode: Obtains the frame skipping reference mode of an H.264 VENC channel.
- HI_MPI_VENC_SetH264eRefParam: Sets the advanced frame skipping reference parameters for an H.264 VENC channel.
- HI_MPI_VENC_GetH264eRefParam: Obtains the advanced frame skipping reference parameters for an H.264 VENC channel.
- HI_MPI_VENC_SetColor2GreyConf: Configures the color-to-gray function globally. That is, all groups are affected after this MPI is called.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_VENC_GetColor2GreyConf: Obtains the global settings of the color-to-gray function.
- HI_MPI_VENC_SetGrpColor2Grey: Enables or disables the color-to-gray function of a group.
- HI_MPI_VENC_GetGrpColor2Grey: Obtains the enable status of the color-to-gray function of a group.
- HI_MPI_VENC_SetGrpCrop: Sets the crop attribute of a channel group. This MPI applies only to the Hi3518 or Hi3516C.
- HI_MPI_VENC_GetGrpCrop: Obtains the crop attribute of a channel group. This MPI applies only to the Hi3518 or Hi3516C.
- HI_MPI_VENC_SetJpegSnapMode: Sets the snapshot mode of a JPEG snapshot channel. This MPI applies only to the Hi3518 or Hi3516C.
- HI_MPI_VENC_GetJpegSnapMode: Obtains the snapshot mode of a JPEG snapshot channel. This MPI applies only to the Hi3518 or Hi3516C.
- HI_MPI_VENC_GetStreamBufInfo: Obtains the physical address and size of a stream buffer.
- HI_MPI_VENC_SetRcPriority: Sets the RC priority.
- HI_MPI_VENC_GetRcPriority: Obtains the RC priority.
- HI_MPI_VENC_SetLostFrameStrategy: Sets the frame discarding policy.
- HI_MPI_VENC_GetLostFrameStrategy: Obtains the frame discarding policy.
- HI_MPI_VENC_AttachVbPool: Binds a VENC channel to a VB pool.
- HI_MPI_VENC_DetachVbPool: Unbinds a VENC channel from a VB pool.
- HI_MPI_VENC_SetH264VideoSignal: Sets the VideoSignal parameter of an H.264 encoding channel.
- HI_MPI_VENC_GetH264VideoSignal: Obtains the VideoSignal parameter of an H.264 encoding channel.

HI_MPI_VENC_CreateGroup

[Description]

Creates a VENC channel group.

[Syntax]

```
HI_S32 HI_MPI_VENC_CreateGroup(VENC_GRP VeGroup);
```

[Parameter]

Parameter	Description	Input/Output
VeGroup	ID of a VENC channel group Value range: [0, AENC_MAX_ GRP_NUM)	Input

[Return Value]

Return Value	Description
0	Success

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- For the MPIs that contain the ID of a channel group, the value range of the channel group ID is 0–VENC_MAX_GRP_NUM.
- A VENC channel group is a set of VENC channels that can be processed by the Hi3531 concurrently. A VENC channel group can contain up to one large stream channel (H.264/MJPEG/MPEG-4) or one JPEG capture channel.
- If the channel group exists, an error code indicating failure is returned.

[Example]

```
HI_S32 StartVenc(HI_VOID)
{
    HI_S32 s32Ret;
    VI_CHN ViChn = 0;
    VENC_GRP VeGroup = 0;
    VENC_CHN VeChn = 0;
    VENC_CHN_ATTR_S stAttr;
    MPP_CHN_S stSrcChn, stDestChn;

    /* set h264 channel video encoding attribute */
    stAttr.stVeAttr.enType   = PT_H264;
    stAttr.stVeAttr.stAttrH264e.u32PicWidth = u32PicWidth;
    stAttr.stVeAttr.stAttrH264e.u32PicHeight = u32PicHeigh;
    stAttr.stVeAttr.stAttrH264e.u32MaxPicWidth = u32MaxPicWidth;
    stAttr.stVeAttr.stAttrH264e.u32MaxPicHeight = u32MaxPicHeigh;

    stAttr.stVeAttr.stAttrH264e.u32Profile = 2;

    …… // omit other video encode assignments here.
    /* set h264 channel rate control attribute */
    stAttr.stRcAttr.enRcMode          =  VENC_RC_MODE_H264CBR ;
    stAttr.stRcAttr.stAttrH264Cbr.u32BitRate            = 10*1024;
    stAttr.stRcAttr.stAttrH264Cbr.fr32TargetFrmRate     = 30;
    stAttr.stRcAttr.stAttrH264Cbr.u32ViFrmRate          = 30;
    stAttr.stRcAttr.stAttrH264Cbr.u32Gop                = 30;
    stAttr.stRcAttr.stAttrH264Cbr.u32FluctuateLevel       = 1;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    stAttr.stRcAttr.stAttrH264Cbr.u32StatTime            = 1;
    …… // omit other rate control assignments here.

    s32Ret = HI_MPI_VENC_CreateGroup(VeGroup);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_CreateGroup err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }

    s32Ret = HI_MPI_VENC_CreateChn(VeChn, &stAttr);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_CreateChn err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }

    s32Ret = HI_MPI_VENC_RegisterChn(VeGroup, VeChn);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_RegisterChn err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }
    s32Ret = HI_MPI_VENC_StartRecvPic(VeChn);
    if (s32Ret != HI_SUCCESS)
    {
        printf("HI_MPI_VENC_StartRecvPic err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }
    …… // omit other code here.
    return HI_SUCCESS;
}
```

For details, see the related sample codes.

[See Also]

None

HI_MPI_VENC_DestroyGroup

[Description]

Destroys a VENC channel group.

[Syntax]

```
HI_S32 HI_MPI_VENC_DestroyGroup(VENC_GRP VeGroup);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
VeGroup	ID of a VENC channel group Value range: [0, AENC_MAX_ GRP_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- Before destroying a channel group, you must ensure that the channel group is empty, namely, no channel registers in the channel group or channels registered in the channel group deregister. Otherwise, an error code indicating failure is returned.
- When you destroy a nonexistent channel group, an error code indicating failure is returned.

[Example]

```
HI_S32 StopVenc(HI_VOID)
{
    HI_S32 s32Ret;
    VENC_CHN VeChn = 0;
    VENC_GRP VeGroup = 0;
    MPP_CHN_S stDestChn;
    s32Ret = HI_MPI_VENC_StopRecvPic(VeChn);
    if (s32Ret != HI_SUCCESS)
    {
        printf("HI_MPI_VENC_StopRecvPic err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }

    s32Ret = HI_MPI_VENC_UnRegisterChn(VeChn);
    if (s32Ret != HI_SUCCESS)
    {
        printf("HI_MPI_VENC_UnRegisterChn err 0x%x\n",s32Ret);
        return HI_FAILURE;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    }

    s32Ret = HI_MPI_VENC_DestroyChn(VeChn);
    if (s32Ret != HI_SUCCESS)
    {
        printf("HI_MPI_VENC_DestroyChn err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }

    s32Ret =HI_MPI_VENC_DestroyGroup(VeGroup);
    if (s32Ret != HI_SUCCESS)
    {
        printf("HI_MPI_VENC_DestroyGroup err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }
    return HI_SUCCESS;
}
```

[See Also]

None

HI_MPI_VENC_CreateChn

[Description]

Creates a VENC channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_CreateChn(VENC_CHN VeChn, const VENC_CHN_ATTR_S
*pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstAttr	Pointer to the attributes of a VENC channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_venc.h, hi_comm_rc.h, mpi_venc.h
- Library file: libmpi.a

[Difference]

- The Hi3531/Hi3532 supports H.264 high profile (HP) encoding and MPEG-4 simple profile (SP) encoding. The Hi3521/Hi3520A/Hi3518/Hi3516C does not support H.264 HP encoding and MPEG-4 encoding.
- The Hi3521/Hi3520A does not support H.264 high profile (HP) encoding and MPEG-4 encoding.
- The Hi3518/Hi3516C supports H.264 high profile (HP) encoding. However, the Hi3518/Hi3516C does not support 8 x 8 conversion and quantization table configuration during the HP encoding.
- The Hi3518/Hi3516C supports SVC-T profile encoding. However, the Hi3518/Hi3516C does not support frame skipping reference configuration during the SVC-T profile encoding.
- The Hi3520D/Hi3515A/Hi3515C supports H.264 HP encoding but does not support MPEG-4 encoding.
- The Hi3520D supports a maximum of 64 H.264 VENC channels, whereas the Hi3515A/Hi3515C supports a maximum of eight H.264 VENC channels.
- The Hi3520D/Hi3515A/Hi3515C supports at most four channels whose resolution is greater than or equal to D1, and the resolution of other channels is less than or equal to CIF.
- For details about RC parameters, see the *Description of Bit Rate Control*.
- The widths and heights of VENC channels vary depending on chip type. See Table 6-4.

Table 6-4 Widths and heights of VENC channels

CHIP	Width or Height (Pixel)	H.264	JPEG	MJPEG	MPEG-4
Hi3531/Hi3532	MIN_WIDTH	160	32	32	160
	MAX_WIDTH	7040	8192	8192	1920
	MIN_HEIGHT	64	32	32	64
	MAX_HEIGHT	8176	8192	8192	1088
	MIN_ALIGN	2	4	4	2
Hi3521/Hi3520A/Hi3518/Hi3516C	MIN_WIDTH	160	64	64	Not Supported
	MAX_WIDTH	1920	8192	8192	Not Supported
	MIN_HEIGHT	64	64	64	Not Supported
	MAX_HEIGHT	2048	8192	8192	Not

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

CHIP	Width or Height (Pixel)	H.264	JPEG	MJPEG	MPEG-4
					Supported
	MIN_ALIGN	2	4	4	Not Supported
Hi3520D/Hi3515A/Hi3515C	MIN_WIDTH	160	64	64	Not Supported
	MAX_WIDTH	1920	8192	8192	Not Supported
	MIN_HEIGHT	64	64	64	Not Supported
	MAX_HEIGHT	2048	8192	8192	Not Supported
	MIN_ALIGN	2	4	4	Not Supported

[Note]

- The attributes of a VENC channel include the attributes of the encoder and RC attributes.
- To set the attributes of the encoder, select encoding protocols and set attributes of each protocol.
- The encoder does not support small streams. Therefore, you must set the stream type to large stream.
- The encoder does not support field encoding. Therefore, you must set the channel encoding type to frame encoding mode.
- The maximum width and height defined in the encoder attribute and the channel width and height must meet the following conditions:
  - MaxPicWidth ϵ [MIN_WIDTH, MAX_WIDTH]
  - MaxPicHeight ϵ [MIN_HEIGHT, MAX_HEIGHT]
  - PicWidth ϵ [MIN_WIDTH, MaxPicwidth]
  - PicHeight ϵ [MIN_HEIGHT, MaxPicHeight]
  - The maximum width, maximum height, channel width, or channel height must be an integral multiple of MIN_ALIGN.

  MIN_WIDTH, MAX_WIDTH, MIN_HEIGHT, and MAX_HEIGHT indicate the minimum width, maximum width, minimum height, and maximum height of a VENC channel respectively. MIN_ALIGN indicates the minimum alignment unit (in pixel). **u32MaxPicWidth** and **u32MaxPicHeight** in the encoder attributes are used to allocate the memory. The actual encoding resolution is specified by **u32PicWidth** and **u32PicHeight**.
- The encoder attributes including the VENC buffer depth and mode of obtaining streams must be set. Table 6-5 describes the limitations on encoder attributes based on protocols.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Table 6-5 Limitations on encoder attributes

Encoding Protocol	Large/ Small Stream	Encoding Mode	Depth of the Stream Buffer	Stream Acquisition Mode	Encoding Profile
H.264	Large stream	Frame	Buffer ≥ MaxPicwidth x MaxPicheight x 3/4	Frame/Slice	Baseline Main profile High profile SVC profile
MJPEG	Large stream	Frame	Buffer ≥ MaxPicwidth x MaxPicheight x 2	Frame/entropy coded segment (ECS)	None
JPEG	Large stream	Frame	Buffer ≥ MaxPicwidth x MaxPicheight x 2	Frame/ECS	Baseline
MPEG-4	Large stream	Frame	Buffer ≥ MaxPicwidth x MaxPicheight x 3/4	Frame/Packet	Simple profile

- A VENC channel can be started only when the width and height of the input picture are less than or equal to the maximum width and height of the channel.
- Recommended width and height of a VECN channel: 2592x1944 (5 megapixels), 1920x1080 (1080p), 1280x720 (720p), 960x540, 640x360, 704x576, 704x480, 352x288, and 352x240.
- All encoder attributes are static attributes except **u32PicWidth** and **u32PicHeight**. After a VENC channel is created successfully, the static attributes cannot be modified unless the channel is destroyed and then recreated. For details about the precautions to be taken during configuration, see the description of HI_MPI_VENC_SetChnAttr.
- The RC mode (one of RC attributes) must be set first. The JPEG capture channel does not need to be configured with RC attributes. Other protocol channels (H.264, MJPEG, and MPEG-4 are supported) must be configured with RC attributes. The RC mode of the RC must match the protocol type of the encoder.
- During H.264 encoding, MJPEG encoding, or MPEG-4 encoding, the bit control modes CBR, VBR, and FIXQP are supported. For different protocols, the attribute variables in the same RC mode are the same. Figure 6-2 describes the public attributes of three RC modes.

Table 6-6 Public attributes of three RC modes

RC Mode	GOP	StatTime(s)	FrmRate(ViFrmRate/TargetFrmRate)
CBR	≥ 1	≥ 1	ViFrmRate ≥ TargetFrmRate
VBR	≥ 1	≥ 1	ViFrmRate ≥ TargetFrmRate
FixQp	≥ 1	≥ 1	ViFrmRate ≥ TargetFrmRate

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- ViFrmRate must be set to the actual input frame rate of the encoder. The RC calculates the actual frame rate and controls the bit rate based on the value of ViFrmRate. If the output frame rate of the VI channel is 30 and GROUP does not control the frame rate, ViFrmRate is set to 30. If the output frame rate of the VI channel is 30 and GROUP controls the frame rate, the source frame rate and ViFrmRate must be set to 30. When you set TargetFrmRate, its data type is defined as the fractional type HI_FR32 that is the same as HI_U32. That is, the higher 16 bits indicate the denominator and the lower 16 bits indicate the numerator. To set TargetFrmRate to an integer, set the upper 16 bits to zeros. For example, if you set ViFrmRate to 25 and TargetFrmRate to 12, 12 frames will be selected from 25 frames for encoding, the other 13 frames are discarded. If you set ViFrmRate to 25 and TargetFrmRate to 15/2, the encoder will encode 15 frames in two seconds.
- TargetFrmRate can be set as follows:
  - If you want the integral frame rate 25, set TargetFrmRate to 25.
  - If you want the fractional frame rate 15/2, set TargetFrmRate to [15 + (2 << 16)].
- In addition to the attributes described in Figure 6-2, the average bit rate (in kbit/s) and bit rate fluctuation level must be set for the CBR. The average bit rate relates to the width and height of a VENC channel and picture frame rate. Table 6-7 describes the typical average bit rate. Assume that the full encoding frame rate is 25 fps. When the frame rate is not full (for example, 10), you can divide the following bit rate by 2.5 (the product of 25 divided by 10) to obtain the actual bit rate.

Table 6-7 Typical configuration of the average bit rate

Width and Height of a Picture/Bit Rate Level	D1 (720 x 576)	720p (1280 x 720)	1080p (1920 x 1080)
Low bit rate	< 400 kbit/s	< 800 kbit/s	< 2000 kbit/s
Medium bit rate	400–1000 kbit/s	800–4000 kbit/s	2000–8000 kbit/s
High bit rate	> 1000 kbit/s	> 4000 kbit/s	> 8000 kbit/s

- There are six fluctuation levels for the bit rate. A higher level indicates a wider range of bit rate fluctuation. If the fluctuation level is high, the picture quality is more stable when the picture contents of adjacent frames vary significantly. The high level applies to the scenario in which the bandwidth margin is large. If the fluctuation level is low, the encoding bit rate is more stable. When the picture contents of adjacent frames vary significantly, the picture quality at a low level may be worse than that at a high level. The low level applies to the scenario in which the bandwidth margin is small.
- In VBR mode, besides the attributes described in Figure 6-2, you also need to set MaxBitRate, MaxQp, and MinQp. MaxBitRate is the maximum bit rate of a VENC channel within the bit rate statistics period; MaxQp is the maximum QP value; MinQp is the minimum QP value.
- In FixQp mode, besides the attributes described in Figure 6-2, you also need to set IQp and PQp. IQp is the fixed QP value for pictures in the I frame. PQp is the fixed QP value for pictures in the P frame. When setting the IQp and PQp, you can increase or decrease them based on the current bandwidth. To reduce the respiratory effect, ensure that the IQp is greater than the PQp by 2 or 3.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

See the example of HI_MPI_VENC_CreateGroup.

[See Also]

None

HI_MPI_VENC_DestroyChn

[Description]

Destroys a VENC channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_DestroyChn(VENC_CHN VeChn);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- When you destroy a nonexistent channel, an error code indicating failure is returned.
- Make sure that the VENC channel to be destroyed deregisters from the channel group. Otherwise, an error code indicating failure is returned.

[Example]

None

[See Also]

None

HI_MPI_VENC_RegisterChn

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Registers a VENC channel to a VENC channel group.

[Syntax]

```
HI_S32 HI_MPI_VENC_RegisterChn(VENC_GRP VeGroup,VENC_CHN VeChn);
```

[Parameter]

Parameter	Description	Input/Output
VeGroup	ID of a VENC channel group Value range: [0, AENC_MAX_ GRP_NUM)	Input
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- When you register a nonexistent channel, an error code indicating failure is returned.
- When you register a VENC channel to a nonexistent VENC channel group, an error code indicating failure is returned.
- A VENC channel can register with only one VENC channel group. If a registered channel attempts to register with another group, an error code indicating failure is returned.
- If a VENC channel group is registered, the group cannot be registered by other channels unless the registration is deleted.

[Example]

See the example of HI_MPI_VENC_CreateGroup.

[See Also]

None

HI_MPI_VENC_UnRegisterChn

[Description]

Deregisters a VENC channel from a VENC channel group.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_VENC_UnRegisterChn(VENC_CHN VeChn);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- When you deregister a nonexistent VENC channel, an error code indicating failure is returned.
- When you deregister an unregistered VENC channel, an error code indicating failure is returned.
- When you deregister a VENC channel that is receiving pictures (picture receiving can be stopped by calling the HI_MPI_VENC_StopRecvPic MPI), an error code indicating failure is returned.
- After VENC channels are deregistered, they are reset, and the corresponding stream buffers are cleared. In a stream buffer that is not released in a timely manner is still used, the data in this buffer may be incorrect. You can call HI_MPI_VENC_Query the query the status of a buffer corresponding to a VENC channel. Ensure that the data in a stream buffer is fetched before deregistering the corresponding VENC channel.

[Example]

See the example of HI_MPI_VENC_DestroyGroup.

[See Also]

None

HI_MPI_VENC_StartRecvPic

[Description]

Enables a VENC channel to start receiving pictures.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_VENC_StartRecvPic(VENC_CHN VeChn);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- If a VENC channel is not created, error HI_ERR_VENC_UNEXIST is returned.
- If the channel does not register with a VENC channel group, an error code indicating failure is returned.
- This MPI does not check whether the function of receiving picture is enabled. That is, the function can be repeatedly enabled, and no error is returned.
- Starting receiving pictures is specific to the channel. The encoder starts receiving pictures only after this function is enabled.

[Example]

See the example of HI_MPI_VENC_CreateGroup.

[See Also]

None

HI_MPI_VENC_StartRecvPicEx

[Description]

Enables a VENC channel to start receiving input pictures and automatically stop the VENC channel from receiving input pictures after the specified number of frames is exceeded.

[Syntax]

```
HI_S32 HI_MPI_VENC_StartRecvPicEx(VENC_CHN VeChn, VENC_RECV_PIC_PARAM_S
*pstRecvParam);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstRecvParam	Pointer to the received picture parameter structure. This pointer indicates the number of frames to be received.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- If the VENC channel is not created, HI_ERR_VENC_UNEXIST is returned.
- A failure message is returned if the channel is not registered with a channel group.
- If the VENC channel is enabled to start receiving pictures without stop by calling HI_MPI_VENC_StartRecvPic or has not received sufficient pictures by calling HI_MPI_VENC_StartRecvPicEx last time, "This operation is prohibited" is returned if this MPI is called again.

[Example]

```
HI_S32 JpegSnapProcess(HI_VOID)
{
    HI_S32 s32Ret;
    VENC_GRP VeGroup = 0;
    VENC_CHN VeChn = 0;
    VENC_CHN_ATTR_S stAttr;
    VENC_RECV_PIC_PARAM_S stRecvParam;
    /* set jpeg channel video encoding attribute */
    stAttr.stVeAttr.enType   = PT_JPEG;
    stAttr.stVeAttr.stAttrJpeg.u32PicWidth = u32PicWidth;
    stAttr.stVeAttr.stAttrJpeg.u32PicHeight = u32PicHeigh;
    stAttr.stVeAttr.stAttrJpeg.u32MaxPicWidth = u32MaxPicWidth;
    stAttr.stVeAttr.stAttrJpeg.u32MaxPicHeight = u32MaxPicHeigh;
    …… // omit other video encode assignments here.
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
// create jpeg channel
s32Ret = HI_MPI_VENC_CreateGroup(VeGroup);
if (HI_SUCCESS != s32Ret)
{
    printf("HI_MPI_VENC_CreateGroup err 0x%x\n",s32Ret);
    return HI_FAILURE;
}
s32Ret = HI_MPI_VENC_CreateChn(VeChn, &stAttr);
if (HI_SUCCESS != s32Ret)
{
    printf("HI_MPI_VENC_CreateChn err 0x%x\n",s32Ret);
    return HI_FAILURE;
}

s32Ret = HI_MPI_VENC_RegisterChn(VeGroup, VeChn);
if (HI_SUCCESS != s32Ret)
{
    printf("HI_MPI_VENC_RegisterChn err 0x%x\n",s32Ret);
    return HI_FAILURE;
}
// start snapping
stRecvParam.s32RecvPicNum = 2;
s32Ret = HI_MPI_VENC_StartRecvPicEx(VeChn, &stRecvParam);
if (s32Ret != HI_SUCCESS)
{
    printf("HI_MPI_VENC_StartRecvPicEx err 0x%x\n",s32Ret);
    return HI_FAILURE;
}
…… // wait until all pictures have been encoded.

s32Ret = HI_MPI_VENC_StopRecvPic(VeChn);
if (s32Ret != HI_SUCCESS)
{
    printf("HI_MPI_VENC_StopRecvPic err 0x%x\n",s32Ret);
    return HI_FAILURE;
}
// destroy jpeg channel
s32Ret = HI_MPI_VENC_UnRegisterChn(VeChn);
if (s32Ret != HI_SUCCESS)
{
    printf("HI_MPI_VENC_UnRegisterChn err 0x%x\n",s32Ret);
    return HI_FAILURE;
}
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    s32Ret = HI_MPI_VENC_DestroyChn(VeChn);
    if (s32Ret != HI_SUCCESS)
    {
        printf("HI_MPI_VENC_DestroyChn err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }
    s32Ret = HI_MPI_VENC_DestroyGroup(VeGroup);
    if (s32Ret != HI_SUCCESS)
    {
    printf("HI_MPI_VENC_DestroyGroup err 0x%x\n",s32Ret);
    return HI_FAILURE;
    }

   return HI_SUCCESS;
}
```

[See Also]

None

HI_MPI_VENC_StopRecvPic

[Description]

Stops a VENC channel from receiving input pictures.

[Syntax]

```
HI_S32 HI_MPI_VENC_StopRecvPic(VENC_CHN VeChn);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

- If a VENC channel is not created, an error code indicating failure is returned.
- If the channel does not register with a VENC channel group, an error code indicating failure is returned.
- This MPI does not check whether picture receiving is stopped. That is, picture receiving can be repeatedly stopped, and no error is returned.
- This MPI is called to stop the VENC channel from receiving pictures. Before deregistered, the channel must stop receiving pictures.
- When this MPI is called, only receiving raw data is stopped but the stream buffer is not cleared.

[Example]

See the example of HI_MPI_VENC_DestroyGroup.

[See Also]

None

HI_MPI_VENC_Query

[Description]

Queries the status of a VENC channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_Query(VENC_CHN VeChn, VENC_CHN_STAT_S *pstStat);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstStat	Pointer to the status of a VENC channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- If a VENC channel is not created, an error code indicating failure is returned.
- This MPI is called to query the status of the encoder. The pstStat parameter contains the following information:
  - bRegistered indicates the registration state of the current VENC channel.
  - u32LeftPics indicates the number of frames to be encoded.

    Before deregistering a VENC channel, you can check whether there are pictures to be encoded, to prevent the frames from being cleared during deregistration.
  - u32LeftStreamBytes indicates the number of remaining bytes in the stream buffer.

    Before deregistering a VENC channel, you can check whether there are streams to be processed, to prevent the streams from being cleared during deregistration.
  - u32LeftStreamFrames indicates the number of remaining frames in the stream buffer.

    Before deregistering a VENC channel, you can check whether there are streams that are not obtained, to prevent the streams from being cleared during deregistration.
  - u32CurPacks indicates the number of stream packets in the current frames. Before calling HI_MPI_VENC_GetStream, ensure that u32CurPacks is greater than 0.

    When streams are obtained by packet, the current frame may not be a complete frame (part of the frame is obtained). When streams are obtained by frame, this parameter indicates the number of packets in the current frame (if it is not a complete frame, this parameter is set to 0). When obtaining streams by frame, you need to check the number of stream packets of a complete frame. Typically, you can select a frame and query the number of stream packets in the selected frame, which is specified by the u32CurPacks parameter.
  - u32LeftRecvPics indicates the number of frames to be received after HI_MPI_VENC_StartRecvPicEx is called.
  - u32LeftEncPics indicates the number of frames to be encoded after HI_MPI_VENC_StartRecvPicEx is called.

    The number of frames to be received or encoded is 0 if HI_MPI_VENC_StartRecvPicEx is not called.

[Example]

See HI_MPI_VENC_GetStream.

[See Also]

None

HI_MPI_VENC_SetChnAttr

[Description]

Sets attributes of a VENC channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetChnAttr(VENC_CHN VeChn, const VENC_CHN_ATTR_S*
pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
	Value range: [0, VENC_MAX_CHN_NUM)	
pstChnAttr	Pointer to the attributes of a VENC channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI must be called after a VENC channel is created.
- The attributes of a VENC channel consist of the encoder attributes and RC attributes.
- Only the values of **u32PicWidth** and **u32PicHeight** for the VENC channel can be changed when the encoder attributes are configured.
- Call HI_MPI_VENC_StopRecvPic to stop receiving pictures before calling HI_MPI_VENC_SetChnAttr to change the value of **u32PicWidth** or **u32PicHeight**.
- After the values of **u32PicWidth** and **u32PicHeight** are changed by calling this MPI, all parameters of the encoding channel group are restored to default values except the frame rate control attributes of the channel group, and the stream buffer as well as the queue for storing pictures are cleared.
- If pstAttr is empty, an error code indicating failure is returned.
- A VENC channel has dynamic attributes and static attributes. The dynamic attributes are set when the channel is created and can be modified before the channel is destroyed. The static attributes are set when the channel is created and cannot be modified after the channel is created.
- This MPI is called to configure only the dynamic attributes of a VENC channel. If you attempt to configure the static attributes by calling this MPI, an error code indicating failure is returned. The static attributes include the maximum width and height of the VENC channel, large/small streams, encoding protocol, frame/field mode for encoding, frame/field mode for input pictures, mode in which streams are obtained (by frame or by packet). The static attributes of each encoding protocol are specified by each protocol module. For details, see VENC_CHN_ATTR_S.
- For details about RC parameters, see the *Description of Bit Rate Control*.
- When the bit rate control mode is changed, all the advanced parameters of the RC are automatically restored to default values. You need to reset the advanced parameters of the RC.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[See Also]

None

HI_MPI_VENC_GetChnAttr

[Description]

Obtains attributes of a VENC channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetChnAttr(VENC_CHN VeChn, VENC_CHN_ATTR_S*pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstChnAttr	Pointer to the attributes of a VENC channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- When you attempt to obtain attributes of a nonexistent VENC channel, error HI_ERR_VENC_UNEXIST is returned.
- If pstAttr is empty, an error code indicating failure is returned.

[Example]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VENC_GetStream

[Description]

Obtains encoded streams.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetStream(VENC_CHN VeChn, VENC_STREAM_S *pstStream,
HI_BOOL bBlockFlag);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstStream	Pointer to the stream structure	Output
bBlockFlag	Block mode Value range: • HI_TRUE: block • HI_FALSE: non-block	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- If a VENC channel is not created, an error code indicating failure is returned.
- If pstStream is empty, an error code indicating failure is returned.
- Streams can be obtained in block or non-block mode, and select or poll system calling is supported.
  - In non-block mode, if there is no data available in the buffer, the error code HI_ERR_VENC_BUF_EMPTY is returned.
  - In block mode, if there is data in the buffer, a message indicating success is returned.
- Streams can be obtained by packet or frame. When streams are obtained by packet:
  - For the H.264 protocol, an NAL unit is obtained each time this MPI is called.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

  - For the JPEG protocol (including JPEG capture and MJPEG), an ECS or a picture parameter stream packet is obtained each time this MPI is called.
  - For the MPEG-4 protocol, a packet is a frame. Therefore, streams obtained by frame or by packet are the same.
- The stream structure VENC_STREAM_S contains the following information:
  - Pointer to a stream packet pstPack

    This parameter specifies the memory space of the VENC_PACK_S. The space is allocated by the caller. When streams are obtained by packet, the space is not smaller than the size specified by the VENC_PACK_S; when streams are obtained by frame, the space is not smaller than n x size specified by the VENC_PACK_S. Here, n indicates the number of stream packets in the current frame and can be obtained by calling the query MPI after the Select invocation.
  - Number of stream packets u32PackCount

    This parameter specifies the value of VENC_PACK_S in pstPack. When streams are obtained by packet, u32PackCount should not be smaller than 1; when streams are obtained by frame, u32PackCount should not be less than the number of packets in the current frame. After this MPI is successfully called, the value of u32PackCount is the number of packets carried in pstPack.
  - Sequence number u32Seq

    When streams are obtained by frame, it is the sequence number of a frame; when streams are obtained by packet, it is the sequence number of a packet.
  - Stream features stH624Info/stJpegInfo/stMpeg4Info. The data type is consortium, including stream features corresponding to various protocols. The output of this information supports the upper-layer application on the user side.
- If streams are not fetched for a long period of time, the stream buffer becomes full. If the stream buffer corresponding to a VENC channel is full, the newly received pictures are discarded until streams are fetched. Stream encoding starts only when the buffer for encoding is sufficient.
- You are advised to call the HI_MPI_VENC_ReleaseStream MPI immediately after calling this MPI and release streams immediately after use to prevent the stream buffer from being full (streams are being obtained in user mode).

[Example]

```
HI_S32 VencGetH264Stream(HI_VOID)
{
    HI_S32 i;
    HI_S32 s32Ret;
    HI_S32 s32VencFd;
    HI_U32 u32FrameIdx = 0;
    VENC_CHN VeChn = 0;
    VENC_CHN_STAT_S stStat;
    VENC_STREAM_S stStream;
    fd_set read_fds;
    FILE *pFile  = NULL;
    pFile = fopen("stream.h264","wb");
    if(pFile == NULL)
    {
        return HI_FAILURE;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    }

    s32VencFd = HI_MPI_VENC_GetFd(VeChn);
    do{
        FD_ZERO(&read_fds);
        FD_SET(s32VencFd,&read_fds);

        s32Ret = select(s32VencFd+1, &read_fds, NULL, NULL, NULL);
        if (s32Ret < 0)
        {
            printf("select err\n");
            return HI_FAILURE;
        }
        else if (0 == s32Ret)
        {
            printf("time out\n");
            return HI_FAILURE;
        }
        else
        {
            if (FD_ISSET(s32VencFd, &read_fds))
            {
                s32Ret = HI_MPI_VENC_Query(VeChn, &stStat);
                if (s32Ret != HI_SUCCESS)
                {
                    return HI_FAILURE;
                }

                stStream.pstPack = (VENC_PACK_S*)
malloc(sizeof(VENC_PACK_S)*stStat.u32CurPacks);
                if (NULL == stStream.pstPack)
                {
                    return HI_FAILURE;
                }

                stStream.u32PackCount = stStat.u32CurPacks;
                s32Ret = HI_MPI_VENC_GetStream(VeChn, &stStream, HI_TRUE);
                if (HI_SUCCESS != s32Ret)
                {
                    free(stStream.pstPack);
                    stStream.pstPack = NULL;
                    return HI_FAILURE;
                }
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
                    for (i=0; i< stStream.u32PackCount; i++)
                    {
                        fwrite(stStream.pstPack[i].pu8Addr[0],
                                stStream.pstPack[i].u32Len[0], 1, pFile);
                        if (stStream.pstPack[i].u32Len[1] > 0)
                        {
                            fwrite(stStream.pstPack[i].pu8Addr[1],
                                    stStream.pstPack[i].u32Len[1], 1, pFile);
                        }
                    }

                    s32Ret = HI_MPI_VENC_ReleaseStream(VeChn,&stStream);
                    if (HI_SUCCESS != s32Ret)
                    {
                        free(stStream.pstPack);
                        stStream.pstPack = NULL;
                        return HI_FAILURE;
                    }

                    free(stStream.pstPack);
                    stStream.pstPack = NULL;
                }
            }
            u32FrameIdx++;
        }while (u32FrameIdx < 0xff);

        fclose(pFile);
        return HI_SUCCESS;
    }
```

For details, see the related sample codes.

[See Also]

None

HI_MPI_VENC_ReleaseStream

[Description]

Releases a stream buffer.

[Syntax]

```
HI_S32 HI_MPI_VENC_ReleaseStream(VENC_CHN VeChn, VENC_STREAM_S
*pstStream);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstStream	Pointer to the stream structure	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- If a VENC channel is not created, error HI_ERR_VENC_UNEXIST is returned.
- If pstStream is empty, error HI_ERR_VENC_NULL_PTR is returned.
- This MPI must work with the HI_MPI_VENC_GetStream MPI. You are advised to release the obtained stream buffer. Otherwise, the stream buffer may be full, affecting encoding. In addition, you must comply with the principle of "first obtained, first released".
- When a VENC channel is deregistered, all stream packets that are not released are invalid and the stream buffer cannot be used or released any more.
- When you release invalid streams, error HI_ERR_VENC_ILLEGAL_PARAM is returned.

[Example]

See HI_MPI_VENC_GetStream.

[See Also]

None

HI_MPI_VENC_InsertUserData

[Description]

Inserts user data.

[Syntax]

```
HI_S32 HI_MPI_VENC_InsertUserData(VENC_CHN VeChn, HI_U8 *pu8Data, HI_U32
u32Len);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pu8Data	Pointer to user data	Input
u32Len	Length of user data, in byte Value range: (0, 1024]	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- If a VENC channel is not created, an error code indicating failure is returned.
- If pu8Data is empty, an error code indicating failure is returned.
- User data can be inserted by following the H.264 and MJEPG/JPEG protocols rather than the MPEG-4 protocol.
- You are advised to start the encoder before inserting user data into an H.264 channel. If you insert user data before starting the encoder, several SEI packets are generated before the first SPS packet in the encoded streams. Most decoders such as the Hi3531 decoder discard the SEI packets before the first SPS packets. As a result, the user data in the SEI packets is lost.
- For the H.264 channel, at most four memory spaces are provided for buffering user data, and the size of each data segment is not greater than 1 KB. If more than four data segments are inserted or a user data segment is greater than 1 KB, an error is returned. Each user data segment is inserted as an SEI packet before the latest picture stream packet. After a user data segment is encoded and sent, the memory space in the the H.264 channel for buffering the user data is cleared for storing new user data.
- For the JPEG/MJPEG channel, at most one memory space is provided for buffering 1 KB user data. The user data is inserted into picture streams as APP segment (0xFFEF). After user data is encoded and sent, the memory space in the JPEG/MJPEG channel for buffering the user data is cleared for storing new user data.

[Example]

```
HI_U8 au8UserData[] = "hisilicon2011";
    s32Ret = HI_MPI_VENC_InsertUserData(VeChn, au8UserData,
```


Copyright © HiSilicon Technologies Co., Ltd

```
            sizeof(au8UserData));
    if(HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_InsertUserData err 0x%xn",s32Ret);
    }
```

[See Also]

None

HI_MPI_VENC_SendFrame

[Description]

Sends raw pictures for encoding.

[Syntax]

```
HI_S32 HI_MPI_VENC_SendFrame(VENC_GRP VeGroup, VIDEO_FRAME_INFO_S
*pstFrame)
```

[Parameter]

Parameter	Description	Input/Output
VeGroup	ID of a VENC channel group Value range: [0, AENC_MAX_ GRP_NUM)	Input
pstFrame	Pointer to a raw picture structure	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI allows you to send pictures to a VENC channel group.
- The source pictures must be semi-planar YUV 4:2:0 or semi-planar YUV 4:2:2 pictures. The H.264 or MPEG-4 VENC channel receives semi-planar420 pictures, and the JPEG or MJPEG VENC channel receives semi-planar420 or semi-planar422 pictures.
- The source picture size must be greater than or equal to the VENC channel size.
- This MPI does not allow you to send a picture histogram to a VENC channel group.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Example]

For details, see the related sample codes.

[See Also]

None

HI_MPI_VENC_SetMaxStreamCnt

[Description]

Sets the maximum number of frames in a stream buffer.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetMaxStreamCnt(VENC_CHN VeChn,HI_U32 u32MaxStrmCnt);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
u32MaxStrmCnt	Maximum number of frames in a stream buffer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This API is used to set the maximum number of frames in a stream buffer.
- If the buffered stream frames reach the max, the picture to be encoded is discarded.
- If the number of stream frames in the buffer reaches the maximum number of stream frames, the picture to be encoded is discarded.
- By default, the maximum number of stream frames is set to 200 by the system when a VENC channel is created.
- This MPI must be called after a VENC channel is created and before the channel is destroyed. This MPI takes effect before a frame is encoded and can be called repeatedly. You are advised to set the maximum number of stream frames after a VENC channel is created and before frames are encoded. Dynamic settings are not recommended.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Example]

None

[See Also]

None

HI_MPI_VENC_GetMaxStreamCnt

[Description]

Obtains the maximum number of frames in a stream buffer.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetMaxStreamCnt(VENC_CHN VeChn,HI_U32 *pu32MaxStrmCnt);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pu32MaxStrmCnt	Pointer to the maximum number of frames in a stream buffer	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VENC_RequestIDR

[Description]

Requests I frame.

[Syntax]

```
HI_S32 HI_MPI_VENC_RequestIDR( VENC_CHN VeChn );
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- If no VENC channel is created, an error code indicating failure is returned.
- When a message requesting an IDR frame or I frame is received, such a frame is encoded in the next frame (the frame rate is controlled).
- This MPI supports only the H.264 and MPEG-4 protocols.

[Example]

```
HI_S32 s32Ret;
VENC_CHN VeChn = 0;
    s32Ret = HI_MPI_VENC_RequestIDR(VeChn);
    if(HI_SUCCESS != s32Ret)
    {
    printf("HI_MPI_VENC_RequestIDR err 0x%xn",s32Ret);
    }
```

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VENC_RequestIDRInst

[Description]

Requests an I frame and encodes an I frame immediately.

[Syntax]

```
HI_S32 HI_MPI_VENC_RequestIDRInst( VENC_CHN VeChn );
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- If no VENC channel is created, an error code indicating failure is returned.
- After a message requesting an IDR frame or I frame is received, such a frame is encoded immediately and frame rate control exerts no restriction on the request.
- This MPI supports only the H.264 and MPEG-4 encoding protocols.
- This MPI is not affected by frame rate control, and an IDR frame or I frame is encoded each time this MPI is called. The frame rate and bit rate are unstable if this MPI is frequently called.

[Example]

```
HI_S32 s32Ret;
VENC_CHN VeChn = 0;
    s32Ret = HI_MPI_VENC_RequestIDRInst(VeChn);
    if(HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_RequestIDRInst err 0x%xn",s32Ret);
    }
```

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

HI_MPI_VENC_GetFd

[Description]

Obtains the device file handle of a VENC channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetFd(VENC_CHN VeChn);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
A positive value	Valid
A non-positive value	Invalid

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

None

HI_MPI_VENC_SetRoiCfg

[Description]

Sets the ROI attributes of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetRoiCfg(VENC_CHN VeChn, VENC_ROI_CFG_S
*pstVencRoiCfg);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstVencRoiCfg	ROI attributes of the H.264 streams	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the ROI attributes of an H.264 encoding channel.
- The ROI parameters include:
  - u32Index: The system can be configured with eight ROIs for each H.264 channel, internally numbered 0 to 7 for management. The value of this parameter indicates the index number of an ROI set by users. ROIs can be overlaid. When ROIs are overlaid, the priority of the ROIs increases with the index number.
  - bEnable: Whether the current ROI is enabled.
  - bAbsQp: Whether the current ROI uses an absolute QP or a relative QP.
  - s32Qp: When bAbsQp is set to true, this parameter indicates the QP value of all macroblocks in an ROI; when bAbsQp is set to false, this parameter indicates the relative QP value of all macroblocks in an ROI.
  - stRect: Specifies the coordinate position and size of the current ROI. The initial position along the horizontal and vertical axes must be within a picture and a multiple of 16. The height and width of the ROI must be a multiple of 16. The ROI must be within a picture.
- This MPI is an enhanced MPI. By default, the ROI function is not enabled. You must call this MPI to enable the ROI function.
- This MPI can be set after a VENC channel is created and before the channel is destroyed. This MPI takes effect after a next frame is encoded.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.
- You are advised to call the HI_MPI_VENC_GetRoiCfg MPI to obtain the ROI configuration of the current channel before calling this MPI.

[Example]

```
HI_S32 SetRoi(HI_VOID)
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
{
    HI_S32 s32Ret = HI_FAILURE;
    VENC_ROI_CFG_S    stRoiCfg;
    VENC_CHN_ATTR_S   stChnAttr;
    HI_S32 index = 0;
    VENC_CHN VeChnId = 0;

    //...omit other thing
    s32Ret = HI_MPI_VENC_GetChnAttr(VeChnId,&stChnAttr);
if (HI_SUCCESS != s32Ret)
{
        printf("HI_MPI_VENC_GetChnAttr err 0x%x\n",s32Ret);
    return HI_FAILURE;
}

    s32Ret = HI_MPI_VENC_GetRoiCfg(VeChnId, index, &stRoiCfg);
    if (HI_SUCCESS != s32Ret)
{
        printf("HI_MPI_VENC_GetRoiCfg err 0x%x\n",s32Ret);
    return HI_FAILURE;
}

    stSrcVencRoiCfg.bEnable         = HI_TRUE;
    stSrcVencRoiCfg.bAbsQp          = HI_TRUE;
    stSrcVencRoiCfg.s32Qp           = 10;
    stSrcVencRoiCfg.stRect.s32X     = 16;
    stSrcVencRoiCfg.stRect.s32Y     = 16;
    stSrcVencRoiCfg.stRect.u32Width  = 16;
    stSrcVencRoiCfg.stRect.u32Height = 16;
    stSrcVencRoiCfg.u32Index        = 0;
    s32Ret = HI_MPI_VENC_SetRoiCfg(VeChnId,  &stSrcVencRoiCfg);
    if (HI_SUCCESS != s32Ret)
    {
            printf("HI_MPI_VENC_SetRoiCfg err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }

    return HI_SUCCESS;
}
```

[See Also]

None

Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VENC_GetRoiCfg

[Description]

Obtains the ROI attributes of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetRoiCfg(VENC_CHN VeChn, HI_U32 u32Index,
VENC_ROI_CFG_S *pstVencRoiCfg);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
u32Index	Index of an ROI on an H.264 channel	Input
pstVencRoiCfg	Configuration of the ROI	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the index ROI attributes of an H.264 encoding channel.
- This MPI must be called after a VENC channel is created and before the channel is destroyed.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.

[Example]

See HI_MPI_VENC_GetRoiCfg.

[See Also]

None

HI_MPI_VENC_SetH264SliceSplit

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Sets the slice attributes of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetH264SliceSplit(VENC_CHN VeChn, const
VENC_PARAM_H264_SLICE_SPLIT_S * pstSliceSplit);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstSliceSplit	Slice mode of an H.264 stream	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to set the slice attributes of an H.264 channel.
- The slice parameters include:
  - bSplitEnable: Whether the current frame is sliced.
  - u32SplitMode: indicates a slice mode. When this parameter is set to 0, the frame is sliced by byte; when this parameter is set to 1, the frame is sliced by macroblock row. This parameter cannot be set to be greater than 1.

    This parameter is effective only when bSplitEnable is set to true.
  - u32SliceSize: indicates the size of a slice. The meaning of this parameter varies with the value of u32SplitMode. When u32SplitMode is set to 0, this parameter indicates the average number of byte in each slice. The encoder encodes a picture from the top down and from the left right in the raster order by macroblock. Each slice is segmented based on the value of this parameter. The actual size of a slice, however, may not be strictly consistent with the value set by users. An error may exist and the offset is always positive. This is because when the last slice is encoded, the remaining byte of the macroblocks are regarded as a complete slice though the number of bits is less than the value of this parameter. When u32SplitMode is set to 1, this parameter indicates the number of macroblock rows occupied by a slice. The unit of this parameter is row of macroblocks. The remaining rows of macroblocks are regarded as a slice though the number of rows of macroblocks is less than the value of this parameter.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- This MPI is an enhanced interface. You can call it as required. However, this MPI is not recommended. By default, bSplitEnable is set to false.
- This MPI can be set after a VENC channel is created and before the channel is destroyed. This MPI takes effect after a next frame is encoded.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.
- You are advised to call the HI_MPI_VENC_GetH264SliceSplit MPI to obtain the slice configuration of the current channel before calling this MPI.

[Example]

```
HI_S32 SetSliceSplit(HI_VOID)
{
    HI_S32 s32Ret = HI_FAILURE;
VENC_PARAM_H264_SLICE_SPLIT_S stSlice;
    VENC_CHN VeChnId = 0;
    //...omit other thing
    s32Ret = HI_MPI_VENC_GetH264SliceSplit(VeChnId,  &stSlice);
    if (HI_SUCCESS != s32Ret)
{
        printf("HI_MPI_VENC_ GetH264SliceSplit err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }

    stSlice.bSplitEnable = HI_TRUE;
    stSlice.u32SplitMode = 0;
    stSlice.u32SliceSize = 2048;
    s32Ret = HI_MPI_VENC_SetH264SliceSplit(VeChnId,  &stSlice);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_ SetH264SliceSplit err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }

    return HI_SUCCESS;
}
```

[See Also]

None

HI_MPI_VENC_GetH264SliceSplit

[Description]

Obtains the slice attributes of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetH264SliceSplit(VENC_CHN VeChn,
```


Copyright © HiSilicon Technologies Co., Ltd

```
VENC_PARAM_H264_SLICE_SPLIT_S *pstSliceSplit);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstSliceSplit	Slice mode of an H.264 stream	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the slice attributes of an H.264 encoding channel.
- This MPI must be called after a VENC channel is created and before the channel is destroyed.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.

[Example]

See HI_MPI_VENC_SetH264SliceSplit.

[See Also]

None

HI_MPI_VENC_SetH264InterPred

[Description]

Sets the inter-prediction attributes of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetH264InterPred(VENC_CHN VeChn, const
VENC_PARAM_H264_INTER_PRED_S *pstH264InterPred);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstH264InterPred	Inter-frame prediction attributes of an H.264 channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Difference]

- The search widow size of the H.264 encoder varies depending on chip type. See Table 6-8.

Table 6-8 Search widow size

Chip	MIN_HW	MAX_HW	MIN_VW	MAX_VW
Hi3531/Hi3532	0	11	0	4
Hi3521/Hi3520A/Hi3518/Hi3516C	0	5	0	2
Hi3520D/Hi3515A/Hi3515C	0	5	0	2

MIN_HW, MAX_HW, MIN_VW, and MAX_VW indicate the minimum horizontal width, maximum horizontal width, minimum vertical width, and maximum vertical width respectively.

- The search window size of the H.264 encoder also varies depending on resolution. See Table 6-9, Table 6-10 and Table 6-11.

Table 6-9 Sizes of the search window of the Hi3531/Hi3532 H.264 encoder under various resolutions

PicWidth (Pixel)	u32HWSize	u32VWSize
≤ 1000	3	2
(1000, 1600]	7	4
(1600, 1920]	9	4
(1920, 2133]	9	3

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

PicWidth (Pixel)	u32HWSize	u32VWSize
(2133, 2742]	9	2
(2742, 3840]	9	1
≥ 3840	9	0

Table 6-10 Sizes of the search window of the Hi3521/Hi3520A H.264 encoder under various resolutions

PicWidth (Pixel)	u32HWSize	u32VWSize
≤ 960	5	2
(960, 1280]	5	1
≥ 1280	5	0

Table 6-11 Sizes of the search window of the Hi3518/Hi3516C H.264 encoder under various resolutions

PicWidth (Pixel)	u32HWSize	u32VWSize
≤ 1280	5	2
> 1280	5	1

Table 6-12 Default sizes of the search windows of the Hi3520D/Hi3515A/Hi3515C H.264 encoder under various resolutions

PicWidth(Pixel)	u32HWSize	u32VWSize
≤ 720	5	2
(720, 1280]	5	1
(1280, 1920]	5	0

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the inter-prediction configuration of an H.264 encoding channel.
- The inter-prediction parameters include:
  - u32HWSize: size of a horizontal search window during inter-prediction.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- u32VWSize: size of a vertical search window during inter-prediction.
- bInter16x16PredEn: a flag used to enable 16x16 inter-prediction.
- bInter16x8PredEn: a flag used to enable 16x8 inter-prediction.
- bInter8x16PredEn: a flag used to enable 8x16 inter-prediction.
- bInter8x8PredEn: a flag used to enable 8x8 inter-prediction.
- bExtedgeEn: enables edge extension of inter-prediction. When inter-prediction is performed on the macroblocks on the boundary of a picture. The search window may exceed the picture boundary. In this case, you can enable edge extension to extend the picture for inter-prediction. If this function is disabled, inter-prediction is not performed on the part of the search window beyond the picture.

- This MPI is an advanced interface. You can call it as required. However, this MPI is not recommended. The inter-prediction attributes of an H.264 channel have default values. By default, the size of the search window varies with the picture resolution and the other five flags are enabled.
- One of the following flags must be enabled: bInter16x16PredEn, bInter16x8PredEn, bInter8x16PredEn, or bInter8x8PredEn. The system does not disable the four flags at the same time.
- This MPI can be set after a VENC channel is created and before the channel is destroyed. This MPI takes effect after a next I frame is encoded.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.
- You are advised to call the HI_MPI_VENC_GetH264InterPred MPI to obtain the InterPred configuration of the current channel before calling this MPI.
- The limitations on the size of a horizontal search window are as follows:

  The horizontal window width must meet the following condition: MIN_HW ≤ horizontal window width ≤ MAX_HW

  The picture width (in macroblock) must be two macroblocks greater than the actual width of a search window. The actual width (in macroblock) of a search window is calculated as follows: Width of a search window = (1 + u32HWSize) x 2 + 1. If the width of the search window is set to 10 pixels, the actual width of a picture must be greater than 384. For example, the actual width can be 386 pixels. The width is calculated as follows: [(1 + 10) x 2 + 1 + 1] x 16 = 384.
- The limitations on the size of a vertical search window are as follows:

  The vertical window width must meet the following condition: MIN_VW ≤ vertical window width ≤ MAX_VW

  The picture height (in macroblock) must be one macroblock greater than the actual height of a search window. Actual height of a search window (in macroblock) is calculated as follows: Height of a search window = (1 + u32VWSize) x 2 + 1. If the height of the search window is set to 4 pixels, the actual picture height must be greater than 176 pixels. For example, the height can be 178 pixels. The height is calculated as follows: [(1 + 4) x 2 + 1] x 16 = 176.
- For the Hi3521, if the picture width is less than or equal to 960 pixels, the width of the vertical search window must be less than or equal to 2 pixels. If the picture width is greater than 960 pixels but is less than or equal to 1280 pixels, the width of the vertical search window must be less than or equal to 1 pixel. If the picture width is greater than 1280 pixels but is less than or equal to 1920 pixels, the width of the vertical search window must be 0.

  For the Hi3518/Hi3516C, if the picture width is less than or equal to 1280 pixels, the width of the vertical search window must be less than or equal to 2 pixels. If the picture

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

width is greater than 1280 pixels but is less than or equal to 1920 pixels, the width of the vertical search window must be less than or equal to 1 pixel.

For the Hi3520D/Hi3515A/Hi3515C, if the picture width is less than or equal to 720 pixels, the width of the vertical search window must be less than or equal to 2 pixels. If the picture width is greater than 720 pixels but is less than and equal to 1280 pixels, the width of the vertical search window must be less than or equal to 1 pixel. If the picture width is greater than 1280 pixels but is less than or equal to 1920 pixels, the width of the vertical search window must be 0.

[Example]

```
HI_S32 SetInterPred(HI_VOID)
{
    HI_S32 s32Ret = HI_FAILURE;
VENC_PARAM_H264_INTER_PRED_S  stInterPred;
    VENC_CHN VeChnId = 0;

    //...omit other thing

    s32Ret = HI_MPI_VENC_GetH264InterPred(VeChnId,  &stInterPred);
    if (HI_SUCCESS != s32Ret)
{
        printf("HI_MPI_VENC_GetH264InterPred err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }

    stInterPred.bExtedgeEn         = HI_TRUE;
    stInterPred.bInter16x16PredEn = HI_TRUE;
    stInterPred.bInter16x8PredEn  = HI_FALSE;
    stInterPred.bInter8x16PredEn  = HI_FALSE;
    stInterPred.bInter8x8PredEn   = HI_FALSE;
    stInterPred.u32HWSize        = 4;
    stInterPred.u32VWSize        = 2;
    s32Ret = HI_MPI_VENC_SetH264InterPred(VeChnId,  &stInterPred);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_ SetH264InterPred err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }
    return HI_SUCCESS;
}
```

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VENC_GetH264InterPred

[Description]

Obtains the inter-prediction attributes of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetH264InterPred(VENC_CHN VeChn,
VENC_PARAM_H264_INTER_PRED_S *pstH264InterPred);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstH264InterPred	Inter-frame prediction attributes of an H.264 channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the inter-prediction mode of an H.264 encoding channel.
- This MPI must be called after a VENC channel is created and before the channel is destroyed.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.
- The Hi3518/Hi3516C does not support 8 x 8 conversion and quantization table configuration. The 8 x 8 conversion and quantization table cannot be configured during the HP encoding and SVC profile encoding.

[Example]

See HI_MPI_VENC_SetH264InterPred.

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VENC_SetH264IntraPred

[Description]

Sets the intra-prediction attributes of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetH264IntraPred(VENC_CHN VeChn, const
VENC_PARAM_H264_INTRA_PRED_S *pstH264IntraPred);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstH264IntraPred	Intra-frame prediction attributes of an H.264 channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Difference]

- As the Hi3531/Hi3532 does not support IP camera prediction, bIpcmEn must be set to HI_FALSE.
- The Hi3521/Hi3520A supports IP camera prediction. You can enable or disable this function. This function is enabled by default.
- The Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C supports IP camera prediction. You can enable or disable this function. This function is enabled by default.

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the intra-prediction configuration of an H.264 encoding channel.
- The intra-prediction parameters include:
  - bIntra16x16PredEn: a flag used to enable 16x16 intra-prediction.
  - bIntraNxNPredEn: a flag used to enable NxN intra-prediction. Here, NxN indicates 4x4 or 8x8.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

  - bIpcmEn: a flag used to enable IPCM intra-prediction.
  - constrained_intra_pred_flag: For details, see the H.264 protocol.
- This MPI is an advanced interface. You can call it as required. However, this MPI is not recommended. The intra-prediction attributes of an H.264 channel have default values. By default, bIntra16x16PredEn, bIntraNxNPredEn, and bIPCMPredEn are enabled and constrained_intra_pred_flag is set to 0. The default values vary depending on chip type.
- Either bIntra16x16PredEn or bIntraNxNPredEn must be enabled. The system does not disable the two flags at the same time.
- This MPI can be set after a VENC channel is created and before the channel is destroyed. This MPI takes effect after a second I frame is encoded.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.
- You are advised to call the HI_MPI_VENC_GetH264IntraPred MPI to obtain the intra-prediction configuration of the current channel before calling this MPI.

[Example]

```
HI_S32 SetIntraPred(HI_VOID)
{
    HI_S32 s32Ret = HI_FAILURE;
    VENC_PARAM_H264_INTRA_PRED_S  stIntraPred;
    VENC_CHN VeChnId = 0;

    //...omit other thing

    s32Ret = HI_MPI_VENC_GetH264IntraPred(VeChnId,  &stIntraPred);
    if (HI_SUCCESS != s32Ret)
{
        printf("HI_MPI_VENC_GetH264IntraPred err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }

    stIntraPred.bIntra16x16PredEn = HI_TRUE;
    stIntraPred.bIntraNxNPredEn    = HI_FALSE;
    stIntraPred.bIpcmEn        = HI_FALSE;
    stIntraPred.constrained_intra_pred_flag = 0;
    s32Ret = HI_MPI_VENC_SetH264IntraPred(VeChnId,  &stIntraPred);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_SetH264IntraPred err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }

    return HI_SUCCESS;
}
```

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

HI_MPI_VENC_GetH264IntraPred

[Description]

Obtains the intra-prediction attributes of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetH264IntraPred(VENC_CHN VeChn,
VENC_PARAM_H264_INTRA_PRED_S *pstH264IntraPred);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstH264IntraPred	Intra-frame prediction attributes of an H.264 channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the intra-prediction mode of an H.264 encoding channel.
- This MPI must be called after a VENC channel is created and before the channel is destroyed.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.

[Example]

See HI_MPI_VENC_SetH264IntraPred.

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VENC_SetH264Trans

[Description]

Sets the transformation and quantization attributes of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetH264Trans(VENC_CHN VeChn, const
VENC_PARAM_H264_TRANS_S *pstH264Trans);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstH264Trans	Conversion and quantization attributes of an H.264 channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the conversion and quantization configurations of a H.264 encoding.
- The transformation and quantization parameters include:
  - u32IntraTransMode: transformation attributes of an intra-macroblock. When this parameter is set to 0, 4x4 and 8x8 transformations are supported for the intra-macroblock; when this parameter is set to 1, only 4x4 transformation is supported for the intra-macroblock; when this parameter is set to 2, only 8x8 transformation is supported for the intra-macroblock. 4x4 transformation (this parameter is set to 1) is available to the H.264 baseline profile and main profile. 8x8 transformation is available only to the H.264 high profile.
  - u32InterTransMode: transformation attributes of an inter-macroblock. When this parameter is set to 0, 4x4 and 8x8 transformations are supported for the intra-macroblock; when this parameter is set to 1, only 4x4 transformation is supported for the intra-macroblock; when this parameter is set to 2, only 8x8 transformation is supported for the intra-macroblock. 4x4 transformation (this parameter is set to 1) is

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

available to the H.264 baseline profile and main profile. 8x8 transformation is available only to the H.264 high profile.

- bScalingListValid: Whether InterScalingList8x8 and IntraScalingList8x8 are valid. In the baseline and main profiles, the H.264 protocol does not support 8x8 transformation. Therefore, this parameter does not take effect. In the high profile, H.264 channels allow you to deliver a quantization table. If this parameter is set to false, the system uses the quantization table specified by the H.264 protocol. Otherwise, the quantization table delivered by users is used.
- InterScalingList8x8: You provide a quantization table by using this array when 8x8 inter-prediction is performed on an inter-macroblock.
- IntraScalingList8x8: You provide a quantization table by using this array when 8x8 transformation is performed on an intra-macroblock.
- chroma_qp_index_offset: For details, see the H.264 protocol.

- This MPI is an advanced interface. You can call it as required. However, this MPI is not recommended. The transformation and quantization attributes of an H.264 channel have default values. By default, the system sets the parameters based on the profile type.

Table 6-13 Default trans parameter values for different profile types

Profile	u32IntraTransMode	u32InterTransMode	bScalingListValid
Baseline/main profile	1	1	false
High profile	0	0	false

- This MPI can be set after a VENC channel is created and before the channel is destroyed. This MPI takes effect after a next I frame is encoded.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.
- You are advised to call the HI_MPI_VENC_GetH264Trans MPI to obtain the trans configuration of the current channel before calling this MPI.

[Example]

```
HI_S32 SetTrans(HI_VOID)
{
    HI_S32 s32Ret = HI_FAILURE;
    VENC_PARAM_H264_TRANS_S stTrans;
    VENC_CHN VeChnId = 0;

    //...omit other thing

    s32Ret = HI_MPI_VENC_GetH264Trans(VeChnId,  &stTrans);
    if (HI_SUCCESS != s32Ret)
    {
        printf("stTrans err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    stTrans.u32IntraTransMode     = 2;
    stTrans.u32InterTransMode     = 2;
    stTrans.bScalingListValid     = HI_FALSE;
    stTrans.chroma_qp_index_offset = 2;
    s32Ret = HI_MPI_VENC_SetH264Trans(VeChnId,  &stTrans);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_SetH264Trans err 0x%x\n",s32Ret);
    return HI_FAILURE;
    }

    return HI_SUCCESS;
}
```

[See Also]

None

HI_MPI_VENC_GetH264Trans

[Description]

Obtains the transformation and quantization attributes of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetH264Trans(VENC_CHN VeChn, VENC_PARAM_H264_TRANS_S
*pstH264Trans);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstH264Trans	Conversion and quantization attributes of an H.264 channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the transformation and quantization configurations of an H.264 encoding channel.
- This MPI must be called after a VENC channel is created and before the channel is destroyed.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.

[Example]

See HI_MPI_VENC_SetH264Trans.

[See Also]

None

HI_MPI_VENC_SetH264Entropy

[Description]

Sets the entropy encoding mode for an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetH264Entropy(VENC_CHN VeChn, const
VENC_PARAM_H264_ENTROPY_S *pstH264EntropyEnc);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstH264EntropyEnc	Entropy encoding mode for an H.264 channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

- This MPI is used to obtain the entropy encoding configuration of an H.264 encoding channel.
- The entropy encoding parameters include:
  - u32EntropyEncModeI: the entropy mode of I frame. When this parameter is set to 0, the I frame is encoded in context adaptive variable length coding (CAVLC) mode; when this parameter is set to 1, the I frame is encoded in context-based adaptive binary arithmetic coding (CABAC) mode.
  - u32EntropyEncModeP: the entropy mode of P frame. When this parameter is set to 0, the P frame is encoded in CAVLC mode; when this parameter is set to 1, the P frame is encoded in CABAC mode.
  - cabac_stuff_en: enables CABAC encoding stuff. By default, it is set to 0. For details, see the H.264 protocol.
  - Cabac_init_idc: indexes CABAC initialization. By default, it is set to 0. For details, see the H.264 protocol.
- The entropy modes of I frame and P frame can be set independently.
- The baseline profile does not support CABAC encoding but supports CAVLC encoding. The main profile and high profile support both the CABAC and CAVLC encoding modes.
- Compared with the CAVLC encoding, CABAC encoding requires greater calculation amount but generates fewer streams. If system performance is low, you are advised to use CAVLC encoding for I frame and CABAC encoding for P frame.
- This MPI is an advanced interface. You can call it as required. However, this MPI is not recommended. The entropy encoding mode for an H.264 channel has a default value. By default, the system sets the parameters based on the profile type.

Table 6-14 Default entropy encoding parameter values for different profile types

Profile	u32EntropyEncModeI	u32EntropyEncModeP
Baseline	0	0
Main profile/High profile	1	1

- This MPI can be set after a VENC channel is created and before the channel is destroyed. This MPI takes effect after a next I frame is encoded.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.
- You are advised to call the HI_MPI_VENC_GetH264Entropy MPI to obtain the entropy configuration of the current channel before calling this MPI.

[Example]

```
HI_S32 SetEntropy(HI_VOID)
{
    HI_S32 s32Ret = HI_FAILURE;
    VENC_PARAM_H264_ENTROPY_S stEntropy;
    VENC_CHN VeChnId = 0;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    //...omit other thing
    s32Ret = HI_MPI_VENC_GetH264Entropy(VeChnId,  &stTrans);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_GetH264Entropy err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }

    stEntropy.u32EntropyEncModeI  = 0;
    stEntropy.u32EntropyEncModeP  = 0;
    stEntropy.cabac_stuff_en     = 0;
    stEntropy.Cabac_init_idc     = 0;
    s32Ret = HI_MPI_VENC_SetH264Entropy(VeChnId,  &stEntropy);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_SetH264Entropy err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }
    return HI_SUCCESS;
}
```

[See Also]

None

HI_MPI_VENC_GetH264Entropy

[Description]

Obtains the entropy encoding attributes of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetH264Entropy(VENC_CHN VeChn,
VENC_PARAM_H264_ENTROPY_S *pstH264EntropyEnc);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstH264EntropyEnc	Entropy encoding mode of an H.264 channel	Output

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the entropy encoding configuration of an H.264 encoding channel.
- This MPI must be called after a VENC channel is created and before the channel is destroyed.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.

[Example]

See HI_MPI_VENC_SetH264Entropy.

[See Also]

None

HI_MPI_VENC_SetH264Poc

[Description]

Sets the picture order count (POC) type of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetH264Poc(VENC_CHN VeChn, const VENC_PARAM_H264_POC_S
*pstH264Poc);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstH264Poc	POC type of an H.264 channel	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the POC configuration of an H.264 encoding channel.
- The POC attribute mainly indicates the POC type of H.264 streams. For details, see the H.264 protocol.
- There are three POC types, which are specified by pic_order_cnt_type (the value can be set to 0, 1, or 2). By default, it is set to 2.
- This MPI can be set after a VENC channel is created and before the channel is destroyed. This MPI takes effect after a next I frame is encoded.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.
- You are advised to call the HI_MPI_VENC_GetH264Poc MPI to obtain the POC configuration of the current channel before calling this MPI.

[Example]

```
HI_S32 SetPoc(HI_VOID)
{
    HI_S32 s32Ret = HI_FAILURE;
    VENC_PARAM_H264_POC_S        stPoc;
    VENC_CHN VeChnId = 0;
    //...omit other thing
    s32Ret = HI_MPI_VENC_GetH264Poc(VeChnId,  &stPoc);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_GetH264Poc err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }

    stPoc.pic_order_cnt_type     = 0;
    s32Ret = HI_MPI_VENC_SetH264Poc(VeChnId,  &stPoc);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_SetH264Poc err 0x%x\n",s32Ret);
        return HI_FAILURE;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    }
    return HI_SUCCESS;
}
```

[See Also]

None

HI_MPI_VENC_GetH264Poc

[Description]

Obtains the POC type of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetH264Poc(VENC_CHN VeChn, VENC_PARAM_H264_POC_S
*pstH264Poc);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstH264Poc	POC type of an H.264 channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the POC configuration of an H.264 encoding channel.
- This MPI must be called after a VENC channel is created and before the channel is destroyed.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.

[Example]

See HI_MPI_VENC_SetH264Poc.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

None

HI_MPI_VENC_SetH264Dblk

[Description]

Sets the de-blocking type of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetH264Dblk(VENC_CHN VeChn, const
VENC_PARAM_H264_DBLK_S *pstH264Dblk);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstH264Dblk	De-blocking type of an H.264 parameter	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the de-blocking configuration of an H.264 encoding channel.
- The de-blocking parameters include:
  - disable_deblocking_filter_idc: For details, see the H.264 protocol.
  - slice_alpha_c0_offset_div2: For details, see the H.264 protocol.
  - slice_beta_offset_div2: For details, see the H.264 protocol.
- By default, the de-blocking function is enabled. Therefore, disable_deblocking_filter_idc = 0, slice_alpha_c0_offset_div2 = 5, and slice_beta_offset_div2 = 5.
- To disable the de-blocking function, set disable_deblocking_filter_idc to 1.
- This MPI can be set after a VENC channel is created and before the channel is destroyed. This MPI takes effect after a next I frame is encoded.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.
- You are advised to call the HI_MPI_VENC_GetH264Dblk MPI to obtain the de-blocking configuration of the current channel before calling this MPI.

[Example]

```
HI_S32 SetDblk(HI_VOID)
{
    HI_S32 s32Ret = HI_FAILURE;
    VENC_PARAM_H264_DBLK_S       stDblk;
    VENC_CHN VeChnId = 0;
    //...omit other thing
    s32Ret = HI_MPI_VENC_GetH264Dblk(VeChnId,  &stDblk);
    if (HI_SUCCESS != s32Ret)
{
        printf("HI_MPI_VENC_GetH264Dblk err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }
    stDblk.disable_deblocking_filter_idc = 0;
    stDblk.slice_alpha_c0_offset_div2    = 6;
    stDblk.slice_beta_offset_div2       = 5;
    s32Ret = HI_MPI_VENC_SetH264Dblk(VeChnId,  &stDblk);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_SetH264Dblk err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }
    return HI_SUCCESS;
}
```

[See Also]

None

HI_MPI_VENC_GetH264Dblk

[Description]

Obtains the de-blocking type of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetH264Dblk(VENC_CHN VeChn, VENC_PARAM_H264_DBLK_S
*pstH264Dblk);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstH264Dblk	De-blocking type of an H.264 channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the de-blocking configuration of an H.264 encoding channel.
- This MPI must be called after a VENC channel is created and before the channel is destroyed.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.

[Example]

See HI_MPI_VENC_SetH264Dblk.

[See Also]

None

HI_MPI_VENC_SetH264Vui

[Description]

Sets the VUI parameters of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetH264Vui(VENC_CHN VeChn, const VENC_PARAM_H264_VUI_S
*pstH264Vui);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstH264Vui	VUI of an H.264 channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the VUI configuration of an H.264 encoding channel.
- The VUI parameters include:
  - chroma_qp_index_offset: For details, see the H.264 protocol.
  - num_units_in_tick: For details, see the H.264 protocol.
  - time_scale: For details, see the H.264 protocol.
  - fixed_frame_rate_flag: For details, see the H.264 protocol.
- This MPI can be set after a VENC channel is created and before the channel is destroyed. This MPI takes effect after a next I frame is encoded.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.
- You are advised to call HI_MPI_VENC_GetH264Vui to obtain the VUI configuration of the current channel before calling this MPI.

[Example]

```
HI_S32 SetVui(HI_VOID)
{
    HI_S32 s32Ret = HI_FAILURE;
    VENC_PARAM_H264_VUI_S       stVui;
    VENC_CHN VeChnId = 0;
    //...omit other thing
    s32Ret = HI_MPI_VENC_GetH264Vui(VeChnId,  &stVui);
    if (HI_SUCCESS != s32Ret)
    {
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
        printf("HI_MPI_VENC_GetH264Vui err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }
    stVui.timing_info_present_flag = 1;
    s32Ret = HI_MPI_VENC_SetH264Vui(VeChnId,  &stVui);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_SetH264Vui err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }
    return HI_SUCCESS;
}
```

[See Also]

None

HI_MPI_VENC_GetH264Vui

[Description]

Obtains the VUI parameter settings of an H.264 channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetH264Vui(VENC_CHN VeChn, VENC_PARAM_H264_VUI_S
*pstH264Vui);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstH264Vui	VUI of an H.264 channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

- This MPI is used to obtain the VUI configuration of an H.264 encoding channel.
- This MPI must be called after a VENC channel is created and before the channel is destroyed.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.

[Example]

See HI_MPI_VENC_SetH264Vui.

[See Also]

None

HI_MPI_VENC_SetJpegParam

[Description]

Sets the advanced parameters of a JPEG channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetJpegParam(VENC_CHN VeChn, const VENC_PARAM_JPEG_S
*pstJpegParam);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstJpegParam	Advanced parameters set of a JPEG channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to set the advanced parameters of a JPEG encoding channel.
- The advanced parameters include:

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

  - u32Qfactor: The value of a quantization factor ranges from 1 to 99. When this parameter is set to a larger value, the quantization coefficient in the quantization table becomes smaller, the obtained picture quality becomes better, and the encoding compression rate becomes lower; when this parameter is set to a smaller value, the quantization coefficient in the quantization table becomes larger, the obtained picture quality becomes poorer, and the encoding compression rate becomes higher. For details about the relationship between this parameter and the quantization table, see the related RFC2435 standard.
  - u8YQt[64], u8CbQt[64], and u8CrQt[64]: These parameters correspond to three quantization tables. You can set a quantization table by using these three parameters.
  - u32MCUPerECS: indicates the number of minimum coded units (MCUs) of each ECS. When this parameter is set to 0, all MCUs in the current frame are encoded into an ECS. The value range is (picwidth + 15) >> 4 x (picheight + 15) >> 4 x 2.
- To use a quantization table, set Qfactor to 50 when setting the quantization table.
- This MPI can be set after a VENC channel is created and before the channel is destroyed. This MPI takes effect after a next I frame is encoded.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.
- You are advised to call the HI_MPI_VENC_GetJpegParam MPI to obtain the JPEG configuration of the current channel before calling this MPI.

[Example]

```
HI_S32 SetJpegParam(HI_VOID)
{
    HI_S32 s32Ret = HI_FAILURE;
    VENC_PARAM_JPEG_S stParamJpeg;
    VENC_CHN VeChnId = 0;
    //...omit other thing
    s32Ret = HI_MPI_VENC_GetJpegParam(VeChnId,  &stParamJpeg);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_GetJpegParam err 0x%x\n",s32Ret);
        return HI_FAILURE;
    }

    stParamJpeg.u32MCUPerECS = 100;
    for (i = 0; i < 64; i++)
    {
       stParamJpeg.u8YQt[i] = 16;
       stParamJpeg.u8CbQt[i] = 17;
       stParamJpeg.u8CrQt[i] = 18;
    }
    s32Ret = HI_MPI_VENC_SetJpegParam(VeChnId,  &stParamJpeg);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_SetJpegParam err 0x%x\n",s32Ret);
        return HI_FAILURE;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
}
return HI_SUCCESS;
}
```

[See Also]

None

HI_MPI_VENC_GetJpegParam

[Description]

Obtains the advanced parameters of a JPEG channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetJpegParam(VENC_CHN VeChn, VENC_PARAM_JPEG_S
*pstJpegParam);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstJpegParam	Advanced parameters of a JPEG channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the configurations of advanced parameters of a JPEG encoding channel.
- This MPI must be called after a VENC channel is created and before the channel is destroyed.
- You are advised to call this MPI after a VENC channel is created and before frames are encoded, to minimize times of calling this MPI during frame encoding.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

See HI_MPI_VENC_SetJpegParam.

[See Also]

None

HI_MPI_VENC_SetMpeg4Param

[Description]

Sets the advanced parameters of an MPEG-4 encoding channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetMpeg4Param(VENC_CHN VeChn, const VENC_PARAM_MPEG4_S
*pstMpeg4Param);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstMpeg4Param	Advanced parameter set of an MPEG-4 encoding channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to set the advanced parameters of an MPEG-4 encoding channel.
- The following describes five advanced parameters:
  - bSplitEnable: Indicates whether the packets of the current frame are split.
  - u32SplitMode: Indicates the mode of splitting packets. The value of u32SplitMode can be set to 0 or 1. If u32SplitMode is set to 0, packets are split by bits; if u32SplitMode is set to 1, packets are split by macroblocks.

    u32SplitMode is valid only when bSplitEnable is set to HI_TRUE.
  - u32PacketSize: Indicates the packet size. The definition of u32PacketSize varies according to the values of u32SplitMode. When u32SplitMode is 0, u32PacketSize

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

indicates the average number of bits in each packet. The encoder starts to encode the picture from the upper left corner of the picture in raster sequence (that is, from left to right and from top to bottom) by macroblocks. The packet size is determined based on the value of u32PacketSize. The size of the packet to be encoded may be inconsistent with the value of u32PacketSize. The difference is allowed and the actual offset is a positive number. When the last packet is to be encoded and the number of remaining macroblocks (in bits) is smaller than the value of u32PacketSize, all remaining macroblocks are encoded as a packet. When u32SplitMode is 1, u32PacketSize indicate the number of macroblock lines occupied by each packet. When the last lines of the picture are to be encoded and the number of remaining macroblock lines is smaller than the value of u32PacketSize, the remaining macroblock lines are considered as a packet.

- u32HWSize: Indicates the size of the horizontal search window during inter-prediction. This parameter must be set to 0 or 1.
- u32VWSize: Indicates the size of the vertical search window during inter-prediction. This parameter must be set to 0.

- The principle and mode of splitting the packets in the MPEG-4 encoding channel are similar to that of splitting the packets in the H.264 encoding channel.
- The sizes of the search windows of the MPEG-4 encoding channel and H.264 encoding channel are similar. However, the ranges of parameter values are different.
- This MPI is applicable to the MPEG-4 encoding channel. The mpeg4param parameters have default values. You can determine whether to call this MPI as required. By default, bSplitEnable is set to HI_FALSE, u32HWSiz is set to 1, and u32VWSize is set to 0.
- This MPI must be called after an MPEG-4 encoding channel is created and before the channel is destroyed. If the MPI is called during encoding, the operation takes effect only after the next frame is encoded.
- You are advised to call this MPI after creating an MPEG-4 encoding channel and before starting encoding. This reduces the number of times of calling this MPI during encoding.
- Before calling this MPI, you are advised to call HI_MPI_VENC_GetMpeg4Param to obtain the Mpeg4Param configurations of the current encoding channel.

[Example]

```
HI_S32 SetMpeg4Param(HI_VOID)
{
    HI_S32 s32Ret = HI_FAILURE;
VENC_PARAM_MPEG4_S stParamMpeg4;
    VENC_CHN VeChnId = 0;
    //...omit other thing
    s32Ret = HI_MPI_VENC_GetMpeg4Param(VeChnId,  &stParamMpeg4);
    if (HI_SUCCESS != s32Ret)
{
        printf("HI_MPI_VENC_GetMpeg4Param err 0x%x\n",s32Ret);
    return HI_FAILURE;
}
stParamMpeg4. bSplitEnable = HI_TRUE;
stParamMpeg4. u32SplitMode = 1;
stParamMpeg4. u32PacketSize = 10;
s32Ret = HI_MPI_VENC_SetMpeg4Param(VeChnId,  &stParamMpeg4);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
if (HI_SUCCESS != s32Ret)
{
    printf("HI_MPI_VENC_SetMpeg4Param err 0x%x\n",s32Ret);
    return HI_FAILURE;
}
return HI_SUCCESS;
}
```

[See Also]

None

HI_MPI_VENC_GetMpeg4Param

[Description]

Obtains the configurations of the advanced parameters of an MPEG-4 encoding channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetMpeg4Param(VENC_CHN VeChn, VENC_PARAM_MPEG4_S
*pstMpeg4Param);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstMpeg4Param	Configurations of the advanced parameters of an MPEG-4 encoding channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the configurations of advanced parameters of an MPEG-4 encoding channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- This MPI must be called after an MPEG-4 encoding channel is created and before the channel is destroyed.
- You are advised to call this MPI after creating an MPEG-4 encoding channel and before starting encoding. This reduces the number of times of calling this MPI during encoding.

[Example]

See the example of HI_MPI_VENC_SetJpegParam.

[See Also]

None

HI_MPI_VENC_SetMjpegParam

[Description]

Sets the advanced parameters of a JPEG encoding channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetMjpegParam(VENC_CHN VeChn, const VENC_PARAM_MJPEG_S
*pstMjpegParam);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstMjpegParam	Advanced parameter set of a JPEG encoding channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to set the advanced parameters of a JPEG encoding channel.
- The following describes four advanced parameters:
  - u8YQt[64], u8CbQt[64], and u8CrQt[64]: The three parameters correspond to three quantization tables. You can set quantization tables by setting the parameters.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

  - u32MCUPerECS: It indicates the number of MCUs in each ECS. When u32MCUPerECS is set to 0, all MCUs of the current frame are encoded as an ECS. The minimum value of u32MCUPerECS is 0, and the maximum value of u32MCUPerECS cannot be greater than {[(picwidth + 15) >> 4] x [(picheight + 15) >> 4]} x 2.

- To use the user-defined quantization tables, set Qfactor to 50 when setting quantization tables.
- This MPI must be called after a JPEG encoding channel is created and before the channel is destroyed. If the MPI is called during encoding, the operation takes effect only after the next frame is encoded.
- You are advised to call this MPI after creating a channel and before starting encoding. This reduces the number of times of calling this MPI during encoding.
- Before calling this MPI, you are advised to call HI_MPI_VENC_GetMjpegParam to obtain the MjpegParam configurations of the current encoding channel.

[Example]

```
HI_S32 SetMjpegParam(HI_VOID)
{
    HI_S32 s32Ret = HI_FAILURE;
VENC_PARAM_MJPEG_S stParamMjpeg;
    VENC_CHN VeChnId = 0;
    //...omit other thing
    s32Ret = HI_MPI_VENC_GetMjpegParam(VeChnId,  &stParamMjpeg);
    if (HI_SUCCESS != s32Ret)
{
        printf("HI_MPI_VENC_GetMjpegParam err 0x%x\n",s32Ret);
    return HI_FAILURE;
}
stParamMjpeg.u32MCUPerECS = 100;
for (i = 0; i < 64; i++)
{
   stParamMjpeg.u8YQt[i] = 16;
   stParamMjpeg.u8CbQt[i] = 17;
   stParamMjpeg.u8CrQt[i] = 18;
}
s32Ret = HI_MPI_VENC_SetMjpegParam(VeChnId,  &stParamMjpeg);
if (HI_SUCCESS != s32Ret)
{
        printf("HI_MPI_VENC_SetMjpegParam err 0x%x\n",s32Ret);
    return HI_FAILURE;
}
return HI_SUCCESS;
}
```

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VENC_GetMjpegParam

[Description]

Obtains the configurations of the advanced parameters of a JPEG encoding channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetMjpegParam(VENC_CHN VeChn, VENC_PARAM_MJPEG_S
*pstMjpegParam);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstMjpegParam	Configurations of the advanced parameters of a JPEG encoding channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the configurations of advanced parameters of a JPEG encoding channel.
- This MPI must be called after a JPEG encoding channel is created and before the channel is destroyed.
- You are advised to call this MPI after creating a channel and before starting encoding. This reduces the number of times of calling this MPI during encoding.

[Example]

See the example of HI_MPI_VENC_SetMjpegParam.

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VENC_SetGrpFrmRate

[Description]

Sets the frame rate control attributes of a channel group.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetGrpFrmRate(VENC_GRP VeGroup, const
GROUP_FRAME_RATE_S *pstGrpFrmRate);
```

[Parameter]

Parameter	Description	Input/Output
VeGroup	ID of a VENC channel group Value range: [0, VENC_MAX_GRP_NUM)	Input
pstGrpFrmRate	Pointer to frame rate control attributes of a channel group	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Request]

- Header files: hi_comm_venc.h, mpi_venc.h.
- Library file: libmpi.a

[Note]

- The frame rate control attributes of a channel group include the input frame rate ViFrmRate and output frame rate VpssFrmRate.
- If you set the attributes of a channel group that does not exist, an error code indicating failure is returned.
- If pstGrpFrmRate is empty, an error code indicating failure is returned.
- ViFrmRate and VpssFrmRate must be greater than 0 or equal to –1 when you set the frame rate control attributes of the channel group.
- If ViFrmRate and VpssFrmRate are greater than 0, ViFrmRate must be greater than or equal to VpssFrmRate. For example, if ViFrmRate is 30 and VpssFrmRate is 15, only 15 frames of the 30 input pictures are encoded.
- When ViFrmRate and VpssFrmRate are –1, the frame rate is not controlled.

[Example]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

None

HI_MPI_VENC_GetGrpFrmRate

[Description]

Obtains the frame rate control attributes of a channel group.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetGrpFrmRate(VENC_GRP VeGroup, GROUP_FRAME_RATE_S
*pstGrpFrmRate);
```

[Parameter]

Parameter	Description	Input/Output
VeGroup	ID of encoding VENC channel group Value range: [0, VENC_MAX_GRP_NUM)	Input
pstGrpFrmRate	Pointer to frame rate control attributes of a channel group	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Request]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- If you obtain the attributes of channel that does not exist, the error code HI_ERR_VENC_UNEXIST is returned.
- If pstGrpFrmRate is empty, an error code indicating failure is returned.

[Example]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VENC_SetRcPara

[Description]

Sets the advanced parameters for the RC of a VENC channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetRcPara(VENC_CHN VeChn, VENC_RC_PARAM_S *pstRcPara);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstRcPara	Pointer to the advanced parameters for the RC of a VENC channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- The advanced parameters have default values. You do not need to call this MPI to enable a VENC channel.
- You are advised to call HI_MPI_VENC_GetRcPara to obtain the values of the advanced parameters for the RC, modify parameters values, and call HI_MPI_VENC_SetRcPara to set advanced parameters.
- The advanced parameters for the RC are available only for the H.264 CBR and H.264 VBR modes.
- When the bit rate control mode is changed, all the advanced parameters of the RC are automatically restored to default values. You need to reset the advanced parameters of the RC.
- The advanced parameters for the RC are as follows:
  - u32ThrdI[12] and u32ThrdP[12]

    The bit rate is controlled by macroblock and based on texture. These two parameters indicate two groups of thresholds for evaluating macroblock complexity of I frames and P frames respectively. Each group has 12 thresholds sorted in ascending order. The value of each threshold ranges from 0 to 255. When the bit rate is controlled

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

based on the macroblock, the thresholds can be used to adjust the QP values of macroblocks based on picture complexity. If the picture complexity value of the current macroblock is between two thresholds, the QP value of the current macroblock is the start QP value of the macroblock row plus the index value of the smaller threshold. For example, u32Thrd[2] < C < u32Thr[3], where the letter C is the picture complexity value. The QP value of the current macroblock is the start QP value of the macroblock row plus 2. If the bit rate of the current channel is high, you can set the thresholds to 255 to save picture details. If the bit rate of the current channel is low, you can decrease the thresholds to ignore some picture details. You need to ensure the picture quality and requirements of the low bit rate. When all the values are set to 255, the macroblock-level bit rate control based on texture is disabled. The default value of u32ThrdI[12] is [5, 5, 5, 10, 10, 10, 255, 255, 255, 255, 255, 255], and the default value of u32ThrdP[12] is [5, 5, 5, 255, 255, 255, 255, 255, 255, 255, 255, 255].

- u32QpDelta

  The bit rate is controlled by macroblock and based on row. This parameter indicates the fluctuation amplitude of the start QP value of each macroblock row relative to the start QP value of the frame when the bit rate is controlled based on the macroblock. If the system requires that the bit rate fluctuates slightly, you can increase the value of u32QpDelta to control the bit rate accurately. However, the quality of the macroblocks of some pictures may be different. In high bit rate mode, the recommended value of u32QpDelta is 0; in medium bit rate mode, the recommended value is 0 or 1; in low bit rate mode, the recommended value is 0, 1, or 2. The default value is 2.

- u32MinIprop and u32MaxIprop

  Indicate the minimum IP proportion and maximum IP proportion respectively. The two parameters are advanced CBR parameters. The IP proportion is the ratio of the bits of I frames to the bits of P frames. The two parameters control the range of each IP proportion. When the value of u32MinIprop increases, the I frame becomes distinct and the P frame becomes blurred. When the value of u32MaxIprop decreases, the I frame becomes blurred and the P frame becomes distinct. You are advised not to limit the IP proportion. This avoids the respiratory effect and bit rate fluctuation. The default value of u32MinIprop is 1, and the default value of u32MaxIprop is 20. If the size of the I frame is limited, you can set the values of u32MinIprop and u32MaxIprop based on requirement on the size change of the I frame.

- u32MaxQp&u32MinQp&u32MaxStartQp

  u32MaxQp and u32MinQp are advanced CBR parameters. u32MaxQp indicates the maximum QP value of the current frame, and u32MinQp indicates the minimum QP value of the current frame. The two parameters control the QP values of all pictures effectively. For example, the two parameters can be used to control the bit rate for macroblocks. The QP value of each picture always ranges from u32MinQp to u32MaxQp. u32MaxStartQp indicates the maximum output QP for controlling the bit rate of frames. Its value range is [u32MinQp, u32MaxQp]. If the bit rate is controlled by macroblock, the QPs of some macroblocks may be greater than or less than u32MaxStartQp. However, the value range is still [u32MinQp, u32MaxQp]. The default value of u32MinQp is 10, and the default value of u32MaxQp or u32MaxStartQp is 51. You are advised to retain the parameter values if there are no special quality requirements.

- u32MaxPPDeltaQp and u32MaxIPDeltaQp

  Indicate the maximum difference between the QP values of two consecutive P frames and the maximum difference between the QP values of consecutive I frame and P frame respectively. The two parameters are advanced CBR parameters. When a large

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

area moves fast or the application scenario changes, the difference between the QP values of P frames may be too large if you want to retain stable bit rate. As a result, the mosaic effect occurs. To avoid the change picture quality, you can change the values of the two parameters to control QP values. The default value of u32MaxPPDelta is 3, and the default value of u32MaxIPdeltaQP is 5. You can change the default values based on the required picture quality and bit rate.

- bLostFrmOpen

  Indicates whether to discard frames to ensure stable bit rate when the instant bit rate is exceeded. This parameter is an advanced CBR parameter. By default, the frame discard function is enabled.

- u32LostFrmBpsThr

  Indicates the instant bit rate threshold for determining whether to discard frames. This parameter is an advanced CBR parameter and is in the unit of bit/s. The default bit rate threshold is 80 x 1024 x 1024 bit/s. You need to determine whether to discard frames and the bit rate when frames are discarded as required.

- enSuperFrmMode

  Indicates the mode of processing the jumbo frame. This parameter is an advanced CBR or VBR parameter. In CBR bit, the jumbo frame can be ignored, discarded, or re-encoded. The ignore mode indicates that no special measure is taken for the jumbo frame. The discard mode indicates that the current jumbo frame is discarded. The re-encode mode indicates that the current jumbo frame is re-encoded. By default, the ignore mode is used.

- u32SuperIFrmBitsThr and u32SuperPFrmBitsThr

  Indicate the threshold of the jumbo I frame for enabling the jumbo frame processing mode and the threshold of the jumbo P frame for enabling the jumbo frame processing mode respectively. The two parameters are advanced CBR or VBR parameters. The thresholds are in the unit of bit. If some frames can be discarded, you are advised not to select the re-encoded mode. This saves hardware resources and avoids the change of picture quality. Adequate thresholds are recommended. If the thresholds are inadequate, consecutive frames are discarded or frames are re-encoded frequently.

- s32IPQPDelta

  Indicates the difference between the average QP value and the QP value of the current I frame. This parameter is an advanced CBR parameter and is used to adjust oversized I frame and avoids the respiratory effect. The parameter value can be a negative value and the default value is 2. When the value increases, the I frame becomes distinct.

- u32RQRatio[8]

  Indicates the weight of stable bit rate and stable quality. This parameter is an advanced CBR parameter. u32RQRatio[i]/100 indicates the weight of stable quality, and (1 – u32RQRatio[i])/100 indicates the weight of stable bit rate. For example, u32RQRatio[i] = 75. This indicates that the weight of stable quality is 75%, and the weight of stable bit rate is 25%. In CBR mode, eight scenarios are supported: Normal, Move (motion scenario), Still, StillToMove (still-to-motion scenario), MoveToStill (motion-to-still scenario), SceneSwitch (scenario switching), SharpMove (strenuous motion scenario), and Init (program initialization scenario). The eight scenarios correspond to the index IDs 0–7 of u32RQRatio. The default value of u32RQRatio [] is {75, 75, 75, 50, 50, 20, 30, 0}. You can set the weights during bit rate control based on the application scenario. Assume that a large area moves fast, the weight of stable quality is 30%, and the weight of stable bit rate is 70%. You can decrease the weight of stable quality to ensure more stable bit rate.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- s32DeltaQP

  Indicates the maximum change of QP values of frames when the picture quality changes. This parameter is an advanced VBR parameter. It is used to avoid mosaic and its default value is 2.

- s32ChangePos

  Indicates the ratio of the current bit rate to the maximum bit rate when the QP value starts to be adjusted. This parameter is an advanced VBR parameter, and its default value is 90. If the macroblocks of consecutive frames on the similar positions have significant differences but the maximum bit rate cannot be exceeded, you are advised to decrease the value of s32ChangePos and increase the value of s32DeltaQP. However, when the bit rate becomes stable, the bit rate is low and the picture quality is poor.

- PRcParam

  Indicates an advanced user-defined parameter of the RC. It is reserved.

- You are advised to call this MPI after creating a channel and before starting encoding. This reduces the number of times of calling this MPI during encoding.
- For details about the advanced parameters of the RC, see the *Description of Bit Rate Control*.

[Example]

```
HI_S32 SetRcParam(HI_VOID)
{
    HI_S32 s32Ret = HI_FAILURE;
    VENC_RC_PARAM_S stVencRcPara;
    VENC_CHN VeChnId = 0;
    //...omit other thing

    s32Ret = HI_MPI_VENC_GetRcPara(VeChnId, &stVencRcPara);
    if (HI_SUCCESS != s32Ret)
    {
    printf("HI_MPI_VENC_GetRcPara err 0x%x\n", s32Ret);
       return HI_FAILURE;
    }
    stVencRcPara.stParamH264Cbr.enSuperFrmMode = SUPERFRM_DISCARD;

    s32Ret = HI_MPI_VENC_SetRcPara(VeChnId, &stVencRcPara);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_VENC_SetRcPara err 0x%x\n", s32Ret);
    return HI_FAILURE;
    }
    //...omit other thing
    return HI_SUCCESS;
}
```

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

HI_MPI_VENC_GetRcPara

[Description]

Obtains the configurations of the advanced parameters for the RC of a VENC channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetRcPara(VENC_CHN VeChn, VENC_RC_PARAM_S *pstRcPara);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstRcPara	Pointer to the advanced parameters for the RC of a VENC channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the configurations of the advanced parameters for the RC of an H.264 encoding channel. For details about parameters, see the description of VENC_RC_PARAM_S.
- If pstRcPara is empty, an error code indicating failure is returned.

[Example]

None

[See Also]

None

HI_MPI_VENC_SetH264eRefMode

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Sets the frame skipping reference mode of an H.264 VENC channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetH264eRefMode(VENC_CHN VeChn,
VENC_ATTR_H264_REF_MODE_E enRefMode);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
enRefMode	Frame skipping reference mode of an H.264 VENC channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- When an H.264 VENC channel is created, the 1x frame skipping reference mode is used by default. That is, frame skipping reference is disabled by default. If you want to change the reference mode, you are advised to call this MPI to set the reference mode after creating the VENC channel but before starting encoding. This reduces the times of calling the MPI during encoding.
- The frame skipping reference configuration is not supported during the SVC profile encoding.

[Note]

None

[See Also]

None

HI_MPI_VENC_GetH264eRefMode

[Description]

Obtains the frame skipping reference mode of an H.264 VENC channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_VENC_GetH264eRefMode(VENC_CHN VeChn,
VENC_ATTR_H264_REF_MODE_E * penRefMode);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
penRefMode	Frame skipping reference mode of an H.264 VENC channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

If penRefMode is not set, an error code indicating failure is returned.

[Note]

None

[See Also]

None

HI_MPI_VENC_SetH264eRefParam

[Description]

Sets the advanced frame skipping reference parameters for an H.264 VENC channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetH264eRefParam(VENC_CHN VeChn,
VENC_ATTR_H264_REF_PARAM_S * pstRefParam);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstRefParam	Advanced frame skipping reference parameters for an H.264 VENC channel	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- If pstRefParam is left blank, an error code indicating failure is returned.
- When an H.264 VENC channel is created, the 1x frame skipping reference mode is used by default. If you want to change the reference mode, you are advised to call this MPI to set the reference mode after creating the VENC channel but before starting encoding. This reduces the times of calling the MPI during encoding.
- If the MPI is called during encoding, the operation takes effect only when the next I frame is encoded.
- The frame skipping reference configuration is not supported during the SVC profile encoding.

[Example]

None

[See Also]

None

HI_MPI_VENC_GetH264eRefParam

[Description]

Obtains the advanced frame skipping reference parameters for an H.264 VENC channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetH264eRefParam(VENC_CHN VeChn,
VENC_ATTR_H264_REF_PARAM_S * pstRefParam);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstRefParam	Advanced frame skipping reference parameters for an H.264 VENC channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

If pstRefParam is left blank, an error code indicating failure is returned.

[Example]

None

[See Also]

None

HI_MPI_VENC_SetColor2GreyConf

[Description]

Configures the color-to-gray function globally. That is, all groups are affected after this MPI is called.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetColor2GreyConf (const GROUP_COLOR2GREY_CONF_S*
pstGrpColor2GreyConf );
```

[Parameter]

Parameter	Description	Input/Output
pstGrpColor2GreyConf	Global setting parameter for the color-to-gray function	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- GROUP_COLOR2GREY_CONF_S has the following parameters:
  - bEnable: It is used to enable or disable the color-to-gray functions of all groups.
  - u32MaxWidth: It is used to set the maximum width of a group whose color-to-gray function will be enabled. The group width is the width of a channel that is registered with the group.
  - u32MaxHeight: It is used to set the maximum height of a group whose color-to-gray function will be enabled. The group height is the height of a channel that is registered with the group.
- Before modifying any of the preceding parameters, ensure that the color-to-gray functions of all groups are disabled.
- All groups are affected after this MPI is called. Before calling HI_MPI_VENC_SetGrpColor2Grey to enable the color-to-gray function of a group, you must call HI_MPI_VENC_SetColor2GreyConf, and set bEnable to HI_TRUE.

[Example]

For details, see VENC samples.

[See Also]

None

HI_MPI_VENC_GetColor2GreyConf

[Description]

Obtains the global settings of the color-to-gray function.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetColor2GreyConf (const GROUP_COLOR2GREY_CONF_S*
pstGrpColor2GreyConf )
```

[Parameter]

Parameter	Description	Input/Output
pstGrpColor2GreyConf	Global setting parameter for the color-to-gray	Output

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
	function	

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

If pstGrpColor2GreyConf is not set, an error code indicating failure is returned.

[Example]

None

[See Also]

None

HI_MPI_VENC_SetGrpColor2Grey

[Description]

Enables or disables the color-to-gray function of a group.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetGrpColor2Grey(VENC_GRP VeGroup, const
GROUP_COLOR2GREY_S* pstGrpColor2Grey)
```

[Parameter]

Parameter	Description	Input/Output
VeGroup	ID of a channel group	Input
pstGrpColor2Grey	Parameter for enabling or disabling the color-to-gray function	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

Before you call this MPI to enable the color-to-gray function of a group, ensure that:

- The global color-to-gray function is set by calling HI_MPI_VENC_SetColor2GreyConf, and bEnable is set to HI_TRUE.
- A channel is registered with the group.
- The width and height of the registered channel are less than the maximum width and height configured during the setting of the global color-to-gray function.

[Example]

For details, see VENC samples.

[See Also]

None

HI_MPI_VENC_GetGrpColor2Grey

[Description]

Obtains the enable status of the color-to-gray function of a group.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetGrpColor2Grey(VENC_GRP VeGroup, GROUP_COLOR2GREY_S*
pstGrpColor2Grey)
```

[Parameter]

Parameter	Description	Input/Output
VeGroup	ID of a channel group	Input
pstGrpColor2Grey	Parameter for enabling or disabling the color-to-gray function	Output

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

If pstGrpColor2Grey is not set, an error code indicating failure is returned.

[Example]

None

[See Also]

None

HI_MPI_VENC_SetGrpCrop

[Description]

Sets the crop attribute of a channel group. This MPI applies only to the Hi3518 or Hi3516C.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetGrpCrop(VENC_GRP VeGroup, const GROUP_CROP_CFG_S
*pstGrpCropCfg);
```

[Parameter]

Parameter	Description	Input/Output
VeGroup	ID of a channel group	Input
pstGrpCropCfg	Crop attribute of a channel group	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to set the crop attribute of a channel group.
- The channel group crops pictures, and then determines whether to zoom out on pictures after comparing the cropped picture size with the VENC channel size.
- This MPI must be called after channels are registered with the channel group and before channels are deregistered from the channel group.
- Before channels in the channel group are deregistered, ensure that the crop function of the channel group is disabled.
- Crop attribute parameters consist of:
  - bEnable: This parameter is used to enable or disable the crop function of the channel group.
  - stRect: The parameter is used to set the crop region attributes including the start point coordinates of the crop region and the crop region size.

[Example]

None

[See Also]

None

HI_MPI_VENC_GetGrpCrop

[Description]

Obtains the crop attribute of a channel group. This MPI applies only to the Hi3518 or Hi3516C.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetGrpCrop(VENC_GRP VeGroup, GROUP_CROP_CFG_S
*pstGrpCropCfg);
```

[Parameter]

Parameter	Description	Input/Output
VeGroup	ID of a channel group	Input
pstGrpCropCfg	Crop attribute of a channel group	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
	For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the crop attribute of a channel group.
- This MPI must be called after a channel group is created and before the group is destroyed.

[Example]

None

[See Also]

None

HI_MPI_VENC_SetJpegSnapMode

[Description]

Sets the snapshot mode of a JPEG snapshot channel. This MPI applies only to the Hi3518 or Hi3516C.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetJpegSnapMode(VENC_CHN VeChn, VENC_JPEG_SNAP_MODE_E
enJpegSnapMode);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID	Input
enJpegSnapMode	Channel snapshot mode	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to set the snapshot mode of a JPEG snapshot channel.
- This MPI must be called after a JPEG VENC channel is created and before the channel is destroyed.
- This MPI applies only to the JPEG VENC channel.
- A JPEG channel can work in either of the following modes:
  - JPEG_SNAP_ALL mode. After the channel starts to receive pictures, all received pictures are encoded.
  - JPEG_SNAP_FLASH mode. After the channel starts to receive pictures, only the pictures that are captured when the camera flash is on are encoded.
- After a JPEG channel is created, it is in JPEG_SNAP_ALL mode by default. To capture pictures only when the camera flash is on, call this MPI to set the JPEG channel mode to JPEG_SNAP_FLASH.
- The JPEG_SNAP_FLASH mode is available only when the front-end camera flash works.
- When the JPEG channel is set to the flash snapshot mode, the channel group receives pictures only from the VIU (pictures are not transmitted to the VPSS). Otherwise, the JPEG channel cannot capture pictures.

[Example]

None

[See Also]

None

HI_MPI_VENC_GetJpegSnapMode

[Description]

Obtains the snapshot mode of a JPEG snapshot channel. This MPI applies only to the Hi3518 or Hi3516C.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetJpegSnapMode(VENC_CHN VeChn, VENC_JPEG_SNAP_MODE_E
*penJpegSnapMode);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID	Input
penJpegSnapMode	Snapshot mode	Output

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to obtain the snapshot mode of a JPEG snapshot channel.
- This MPI applies only to the JPEG VENC channel.
- This MPI must be called after a channel is created and before the channel is destroyed.

[Example]

None

[See Also]

None

HI_MPI_VENC_GetStreamBufInfo

[Description]

Obtains the physical address and size of a stream buffer.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetStreamBufInfo(VENC_CHN VeChn,
VENC_STREAM_BUF_INFO_S *pstStreamBufInfo)
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID Value range: [0, VENC_MAX_CHN_NUM)	Input
pstStreamBufInfo	Pointer to stream buffer information	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- If no VENC channel is created, HI_ERR_VENC_UNEXIST is returned.
- If **pstStream** is null, HI_ERR_VENC_NULL_PTR is returned.
- If you want to use the physical address for a frame stream, you need to obtain the start physical address and size of the stream buffer by calling this MPI for checking whether the stream is wrapped. If the following condition is met, the frame stream is wrapped and two physical addresses are obtained: **pstPack[0]. u32PhyAddr** + **u32Len** > **pstStreamBufInfo-> u32PhyAddr** + **pstStreamBufInfo-> u32BufSize**. This ensures that the correct physical address is used. **pstPack[0]** is the variable of VENC_STREAM_S, and **u32Len** indicates the length of a frame stream. See Figure 6-8.

[Example]

```
HI_S32 VencGetH264StreamBufInfo(HI_VOID)
{
    HI_S32 i;
    HI_S32 s32Ret;
    HI_S32 s32VencFd;
    HI_U32 u32FrameIdx = 0;
    VENC_CHN VeChn = 0;
    VENC_CHN_STAT_S stStat;
    VENC_STREAM_S stStream;
    fd_set read_fds;
    VENC_STREAM_BUF_INFO_S stStreamBufInfo;
    HI_U32 u32Left;
    HI_U32 u32SrcPhyAddr, u32DestPhyAddr;

    s32Ret = HI_MPI_VENC_GetStreamBufInfo (VeChn,&stStreamBufInfo);
    if (HI_SUCCESS != s32Ret)
    {
            return HI_FAILURE;
    }
    do{
        s32Ret = HI_MPI_VENC_Query(VeChn, &stStat);
        if (s32Ret != HI_SUCCESS)
        {
            return HI_FAILURE;
        }
        stStream.pstPack = (VENC_PACK_S*)
                        malloc(sizeof(VENC_PACK_S)*stStat.u32CurPacks);
        if (NULL == stStream.pstPack)
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

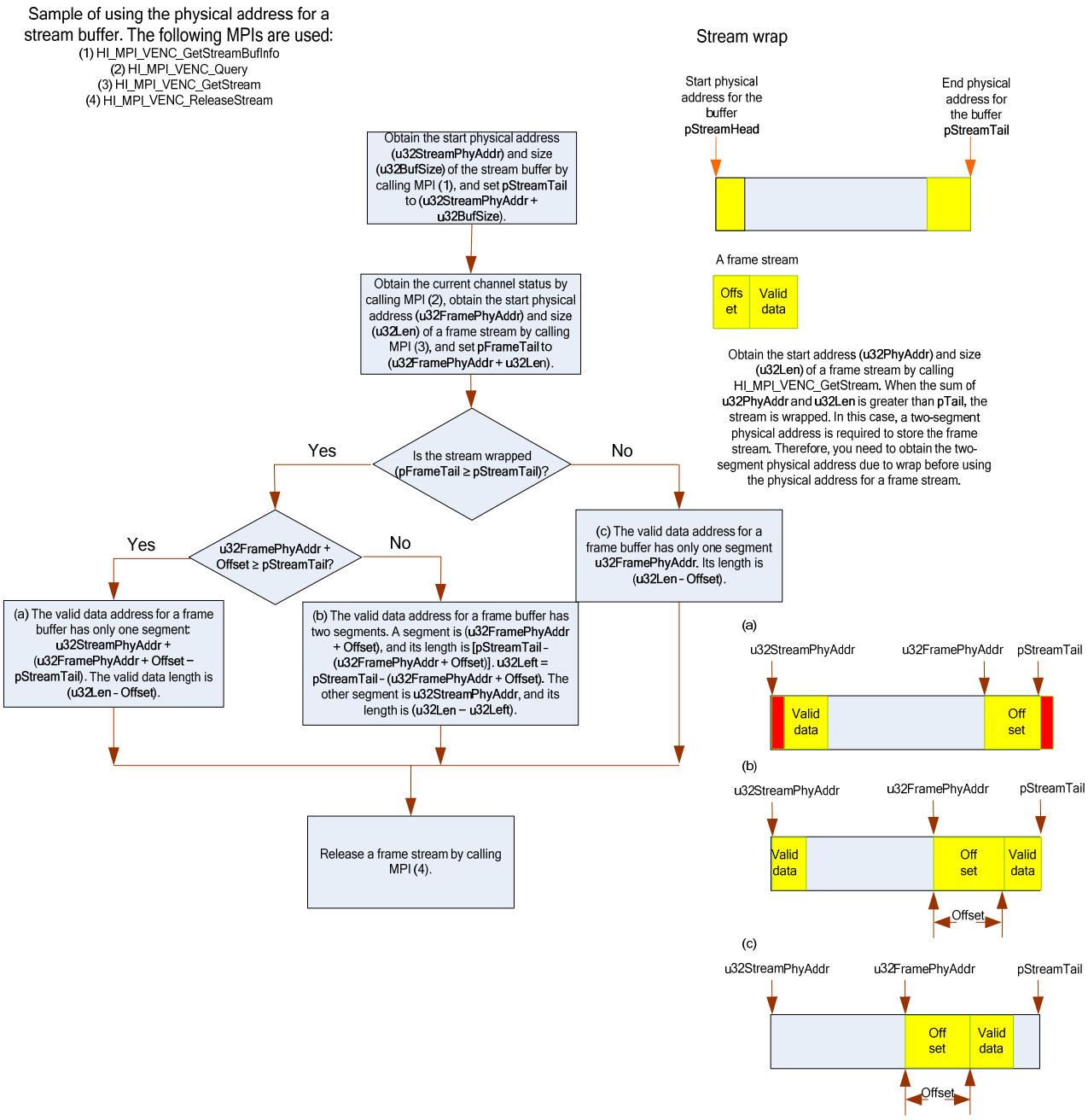
```
{
    return HI_FAILURE;
}
stStream.u32PackCount = stStat.u32CurPacks;
s32Ret = HI_MPI_VENC_GetStream(VeChn, &stStream, -1);
if (HI_SUCCESS != s32Ret)
{
    free(stStream.pstPack);
    stStream.pstPack = NULL;
    return HI_FAILURE;
}
//DMA Transfer. Only the physical address is used.
for (i=0; i< stStream.u32PackCount; i++)
{
    if(stStream.pstPack[i].u32PhyAddr+
    stStream.pstPack[i].u32Len >=
    stStreamBufInfo.u32PhyAddr +
    stStreamBufInfo.u32BufSize)
    {
        if(stStream.pstPack[i].u32PhyAddr+
        stStream.pstPack[i].u32Offset >=
        stStreamBufInfo.u32PhyAddr +
        stStreamBufInfo.u32BufSize)
        {
            //(a) branch in picture
            u32SrcPhyAddr = stStreamBufInfo.u32PhyAddr+
            ((stStream.pstPack[i].u32PhyAddr+
            stStream.pstPack[i].u32Offset) -
            (stStreamBufInfo.u32PhyAddr +
            stStreamBufInfo. u32BufSize));

            DMA_TransFer(u32SrcPhyAddr, u32DestPhyAddr,
            stStream.pstPack[i].u32Len-
            stStream.pstPack[i].u32Offset);
        }
        else
        {
                // (b) branch in picture
                u32Left = (stStreamBufInfo.u32PhyAddr +
                        stStreamBufInfo.u32BufSize) -
                        stStream.pstPack[i].u32PhyAddr;

                DMA_TransFer(stStream.pstPack[i].u32PhyAddr +
                stStream.pstPack[i].u32Offset,
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
                            u32DestPhyAddr,u32Left -
                            stStream.pstPack[i].u32Offset);

                            DMA_TransFer(stStreamBufInfo.u32PhyAddr,
                            u32DestPhyAddr, stStream.pstPack[i].u32Len-
                            u32Left);
                        }
                    }
                    else
                    {
                        //(c) branch in picture
                        DMA_TransFer (stStream.pstPack[i].u32PhyAddr +
                        stStream.pstPack[i].u32Offset,
                        stStream.pstPack[i].u32Len-
                        stStream.pstPack[i].u32Offset);
                    }
                }

                s32Ret = HI_MPI_VENC_ReleaseStream(VeChn,&stStream);
                if (HI_SUCCESS != s32Ret)
                {
                    free(stStream.pstPack);
                    stStream.pstPack = NULL;
                    return HI_FAILURE;
                }

                free(stStream.pstPack);
                stStream.pstPack = NULL;

                u32FrameIdx++;
            }while (u32FrameIdx < 0xff);
            return HI_SUCCESS;
        }
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 6-8 Checking whether the stream buffer is wrapped



[See Also]

None

HI_MPI_VENC_SetRcPriority

[Description]

Sets the RC priority.

Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_VENC_SetRcPriority(VENC_CHN VeChn, VENC_RC_PRIORITY_E
enRcPriority);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID	Input
enRcPriority	Priority enumeration	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to set priority to the target bit rate or the jumbo frame threshold in bit rate control. If the target bit rate takes priority over the jumbo frame threshold, when the bit rate is too low, jumbo frames may be encoded to compensate the rate and they are not re-encoded. If the jumbo frame threshold takes priority over the target bit rate, the jumbo frames are re-encoded to decrease the number of bits. As a result, the bit rate may be too low.
- This MPI must be called after the encoding channel is created and before the channel is destroyed.
- This MPI is valid only for the CBRv2 and VBRv2.

[Example]

None

[See Also]

None

HI_MPI_VENC_GetRcPriority

[Description]

Obtains the RC priority.

[Syntax]

Copyright © HiSilicon Technologies Co., Ltd

```
HI_S32 HI_MPI_VENC_GetRcPriority(VENC_CHN VeChn, VENC_RC_PRIORITY_E
*penRcPriority);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID	Input
penRcPriority	Pointer to the RC priority	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI must be called after the encoding channel is created and before the channel is destroyed.
- This MPI is valid only for the CBRv2 and VBRv2.

[Example]

None

[See Also]

None

HI_MPI_VENC_SetLostFrameStrategy

[Description]

Sets the frame discarding policy when the bit rate is too high.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetLostFrameStrategy(VENC_CHN VeChn, const
VENC_RC_PARAM_LOSTFRM_S *pstLostFrmParam);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VeChn	VENC channel ID	Input
pstLostFrmParam	Pointer to the frame discarding policy	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI must be called after a channel is created and before the channel is destroyed.
- This MPI is valid only for the CBRV2 and VBRV2 modes during the H.264 encoding.
- When the bit rate is too high, there are two frame discarding modes: the normal frame discarding mode and the Pskip frame encoding mode. In normal frame discarding mode, consecutive *n* (not greater than **u32EncFrmGaps**) frames are discarded until the bit rate is lower than the threshold. In Pskip frame encoding mode, consecutive *n* (not greater than **u32EncFrmGaps**) frames are encoded as Pskip frames until the bit rate is lower than the threshold.
- In normal frame discarding mode, if **u32EncFrmGaps** is not **0**, the number of frames discarded is limited and at most **u32EncFrmGaps** frames are discarded. If **u32EncFrmGaps** is **0**, the number of frames discarded is not limited and frames are discarded until the bit rate is lower than the threshold.
- In Pskip frame encoding mode, if **u32EncFrmGaps** is not **0**, the number of frames encoded as Pskip frames is limited and at most **u32EncFrmGaps** frames are encoded as Pskip frames. If **u32EncFrmGaps** is **0**, the number of frames encoded as Pskip frames is not limited and frames are encoded as Pskip frames until the bit rate is lower than the threshold. However, if the current frame is at the base layer and referenced for the advanced frame skipping, it cannot be encoded as the Pskip frame. In this case, the current frame is output normally.
- The frame discarding mode takes effect after **bLostFrmOpen** is enabled and **u32LostFrmBpsThr** is set.
- **bLostFrmOpen** and **u32LostFrmBpsThr** can be set by configuring the advanced RC parameters (HI_MPI_VENC_SetRcPara) and calling this MPI, and the latest settings prevail.
- When the bit rate control mode is changed, all the advanced parameters of the RC are automatically restored to default values. You need to reset the advanced parameters of the RC.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[See Also]

None

HI_MPI_VENC_GetLostFrameStrategy

[Description]

Obtains the frame discarding policy when the bit rate is too high.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetLostFrameStrategy(VENC_CHN VeChn, const
VENC_RC_PARAM_LOSTFRM_S *pstLostFrmParam);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID	Input
pstLostFrmParam	Pointer to the frame discarding policy structure	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

This MPI must be called after a channel is created and before the channel is destroyed.

[Example]

None

[See Also]

None

HI_MPI_VENC_AttachVbPool

[Description]

Binds a VENC channel to a VB pool.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_VENC_AttachVbPool(VENC_CHN VeChn, VB_POOL hVbPool);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID	Input
hVbPool	VB pool ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h, hi_comm_vb.h
- Library file: libmpi.a

[Note]

- Ensure that a VENC channel is created before calling this MPI. Otherwise, the error code indicating that no channel is created is returned.
- You must create a VB pool by calling HI_MPI_VB_CreatePool, and bind the current VENC channel to the VB pool with a fixed pool ID by calling HI_MPI_VENC_AttachVbPool. Multiple VENC channels can be bound to the same VB pool, but a VENC channel cannot be bound to multiple VB pools.
- If you want to bind the VENC channel to another VB pool, you only need to configure the VB pool to be bound by calling HI_MPI_VENC_AttachVbPool.
- The UserVB pool mode is supported only during H.264 encoding. If the current mode of allocating the buffers for storing encoded frames is not the encoding UserVB pool mode, the error code indicating that the mode is not supported by the VENC channel is returned.
- **hVbPool** must be a valid pool ID of the created VB pool.

HI_MPI_VENC_DetachVbPool

[Description]

Unbinds a VENC channel from a VB pool.

[Syntax]

```
HI_S32 HI_MPI_VENC_DetachVbPool(VENC_CHN VeChn);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VeChn	VENC channel ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h, hi_comm_vb.h
- Library file: libmpi.a

[Note]

- Ensure that a VENC channel is created before calling this MPI. Otherwise, the error code indicating that no channel is created is returned.
- If the current mode of allocating the buffers for storing encoded frames is not the encoding UserVB pool mode, the error code indicating that the mode is not supported by the VENC channel is returned.

HI_MPI_VENC_SetH264VideoSignal

[Description]

Sets the VideoSignal parameter of an H.264 encoding channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_SetH264VideoSignal(VENC_CHN VeChn, const
VENC_PARAM_H264_VIDEO_SIGNAL_S *pstH264VideoSignal);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID	Input
pstH264VideoSignal	VideoSignal parameter of an H.264 encoding channel	Input

[Return Value]

Return Value	Description
0	Success

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

- This MPI is used to set the VideoSignal parameter of an H.264 encoding channel. For details, see the H.264 protocol.
- This MPI can be called after a VENC channel is created and before the channel is destroyed. If this MPI is called during encoding, the operation takes effect only when the next I frame is encoded.
- You are advised to call this MPI after creating a channel but before starting encoding. This reduces the number of times of calling this MPI during encoding.
- Before calling this MPI to set the VideoSignal parameter, you are advised to call HI_MPI_VENC_GetH264VideoSignal to obtain the VideoSignal configurations of the current VENC channel.

[Example]

None

[See Also]

None

HI_MPI_VENC_GetH264VideoSignal

[Description]

Obtains the VideoSignal parameter of an H.264 encoding channel.

[Syntax]

```
HI_S32 HI_MPI_VENC_GetH264VideoSignal(VENC_CHN VeChn,
VENC_PARAM_H264_VIDEO_SIGNAL_S *pstH264VideoSignal);
```

[Parameter]

Parameter	Description	Input/Output
VeChn	VENC channel ID	Input
pstH264VideoSignal	VideoSignal parameter of an H.264 encoding channel	Output

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 6.5 "Error Codes."

[Requirement]

- Header files: hi_comm_venc.h, mpi_venc.h
- Library file: libmpi.a

[Note]

This MPI can be called after a VENC channel is created and before the channel is destroyed.

[Example]

None

[See Also]

None

6.4 Data Types

VENC data types and data structures are defined as follows:

- H264E_NALU_TYPE_E: Defines the NALU type of an H.264 stream.
- H264E_REFSLICE_TYPE_E: Defines the reference frame type in a specific frame skipping reference mode based on the obtained H.264 streams.
- H264E_REF_TYPE_E: Defines the frame type and reference attributes of the H.264 frame skipping reference streams.
- JPEGE_PACK_TYPE_E: Defines the Pack type of a JPEG stream.
- MPEG4E_PACK_TYPE_E: Defines the Pack type of an MPEG-4 stream.
- VENC_DATA_TYPE_U: Defines the stream result type.
- VENC_PACK_S: Defines the structure of a stream packet.
- VENC_STREAM_INFO_H264_S: Defines the features of an H.264 stream.
- VENC_STREAM_INFO_JPEG_S: Defines the features of a JPEG stream.
- VENC_STREAM_INFO_MPEG4_S: Defines the features of an MPEG-4 stream
- VENC_STREAM_S: Defines the stream frame type.
- VENC_ATTR_H264_S: Defines the attributes of the H.264 encoder.
- VENC_ATTR_MJPEG_S: Defines the attributes of the MJPEG encoder.
- VENC_ATTR_JPEG_S: Defines the attributes of the JPEG snapshot encoder.
- VENC_ATTR_MPEG4_S: Defines the attributes of the MPEG-4 encoder.
- VENC_ATTR_S: Defines the encoder attributes.
- VENC_CHN_ATTR_S: Defines the attributes of a VENC channel.
- VENC_CHN_STAT_S: Defines the status of a VENC channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- VENC_PARAM_H264_SLICE_SPLIT_S: Defines the slice attributes of an H.264 channel.
- VENC_PARAM_H264_INTER_PRED_S: Defines the inter-prediction attributes of an H.264 channel.
- VENC_PARAM_H264_INTRA_PRED_S: Defines the intra-prediction attributes of an H.264 channel.
- VENC_PARAM_H264_TRANS_S: Defines the transformation and quantization attributes of an H.264 channel.
- VENC_PARAM_H264_ENTROPY_S: Defines the entropy attributes of an H.264 channel.
- VENC_PARAM_H264_POC_S: Defines the POC attributes of an H.264 channel.
- VENC_PARAM_H264_DBLK_S: Defines the de-blocking attributes of an H.264 channel.
- VENC_PARAM_H264_VUI_S: Defines the VUI attributes of an H.264 channel.
- VENC_PARAM_JPEG_S: Defines the parameters of the JPEG encoder.
- VENC_PARAM_MJPEG_S: Defines the advanced parameters of an MJPEG encoding channel.
- VENC_PARAM_MPEG4_S: Defines the parameters of the MPEG-4 encoder.
- VENC_ROI_CFG_S: Defines an ROI for a VENC channel.
- VENC_ATTR_H264_REF_MODE_E: Defines the H.264 frame skipping reference mode.
- VENC_ATTR_H264_REF_PARAM_S: Defines the advanced frame skipping reference parameters for H.264 encoding.
- VENC_RC_ATTR_S: Defines RC attributes for controlling the bit rate of a VENC channel.
- VENC_RC_MODE_E: Defines the mode of the RC for controlling the bit rate of a VENC channel.
- VENC_ATTR_H264_CBR_S: Defines the structure of the CBR attribute of an H.264 encoding channel.
- VENC_ATTR_H264_VBR_S: Defines the structure of the VBR attribute of an H.264 encoding channel.
- VENC_ATTR_H264_FIXQP_S: Defines the structure of the FixQP attribute of an H.264 encoding channel.
- VENC_ATTR_H264_ABR_S: Defines the structure of the ABR attribute of an H.264 encoding channel.
- VENC_ATTR_MPEG4_CBR_S: Defines the structure of the CBR attribute of an MPEG-4 encoding channel.
- VENC_ATTR_MPEG4_VBR_S: Defines the structure of the VBR attribute of an MPEG-4 encoding channel.
- VENC_ATTR_MPEG4_FIXQP_S: Defines the structure of the FixQp attribute of an MPEG-4 encoding channel.
- VENC_ATTR_MJPEG_FIXQP_S: Defines the structure of the FixQp attribute of an MJPEG encoding channel.
- VENC_ATTR_MJPEG_CBR_S: Defines the structure of the CBR attribute of an MJPEG encoding channel.
- VENC_ATTR_MJPEG_VBR_S: Defines the structure of the VBR attribute of an MJPEG encoding channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- VENC_SUPERFRM_MODE_EN: Defines the mode of processing jumbo frames when the bit rate is being controlled.
- VENC_PARAM_H264_CBR_S: Defines the configurations of the advanced parameters related to the CBR control mode of an H.264 encoding channel.
- VENC_PARAM_H264_VBR_S: Defines the configurations of the advanced parameters related to the VBR control mode of an H.264 encoding channel.
- VENC_PARAM_MJPEG_CBR_S: Defines the configurations of the advanced parameters related to the CBR control mode of an MJPEG encoding channel.
- VENC_PARAM_MJPEG_VBR_S: Defines the configurations of the advanced parameters related to the VBR control mode of an MJPEG channel.
- VENC_PARAM_MPEG4_CBR_S: Defines the configurations of the advanced parameters related to the CBR control mode of an MPEG-4 encoding channel.
- VENC_PARAM_MPEG4_VBR_S: Defines the configurations of the advanced parameters related to the VBR control mode of an MPEG-4 channel.
- VENC_PARAM_H264_CBRV2_S: Defines the configurations of the advanced parameters related to the H.264 CBR mode of the new bit rate control algorithm.
- VENC_RC_PARAM_S: Defines the advanced parameters for controlling the bit rate of a VENC channel.
- GROUP_CROP_CFG_S: Defines the clip parameters of a channel group.
- GROUP_FRAME_RATE_S Defines the parameters for controlling the frame rate of a channel group.
- GROUP_COLOR2GREY_S: Defines the color-to-gray function of a group.
- GROUP_COLOR2GREY_CONF_S: Defines the global parameters for the color-to-gray function.
- VENC_JPEG_SNAP_MODE_E: Defines the snapshot mode of a JPEG VENC channel.
- VENC_RECV_PIC_PARAM_S: Defines the number of received pictures.
- VENC_RC_PRIORITY_E: Defines the RC priority.
- VENC_RC_LOSTFRM_MODE_E: Defines the frame discarding modes.
- VENC_RC_PARAM_LOSTFRM_S: Defines the frame discarding policy.
- VENC_PARAM_H264_VIDEO_SIGNAL_S: Defines the VideoSignal attributes of an H.264 encoding channel.

H264E_NALU_TYPE_E

[Description]

Defines the NALU type of an H.264 stream.

[Syntax]

```
typedef enum hiH264E_NALU_TYPE_E
{
    H264E_NALU_PSLICE   = 1,
    H264E_NALU_ISLICE   = 5,
    H264E_NALU_SEI      = 6,
    H264E_NALU_SPS      = 7,
    H264E_NALU_PPS      = 8,
    H264E_NALU_BUTT
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
} H264E_NALU_TYPE_E;
```

[Member]

Member	Description
H264E_NALU_PSLICE	PSLICE
H264E_NALU_ISLICE	PSLICE
H264E_NALU_SEI	SEI
H264E_NALU_SPS	SPS
H264E_NALU_PPS	PPS

[Note]

None

[See Also]

None

H264E_REFSLICE_TYPE_E

[Description]

Defines the reference frame type in a specific frame skipping reference mode based on the obtained H.264 streams.

[Syntax]

```
typedef enum hiH264E_REFSLICE_TYPE_E
{
    H264E_REFSLICE_FOR_1X       = 1,
    H264E_REFSLICE_FOR_2X       = 2,
    H264E_REFSLICE_FOR_4X        = 5,
    H264E_REFSLICE_FOR_BUTT
} H264E_REFSLICE_TYPE_E;
```

[Member]

Member	Description
H264E_REFSLICE_FOR_1X	Reference frame in 1x frame skipping reference mode
H264E_REFSLICE_FOR_2X	Reference fame in 2x frame skipping reference mode or reference frame in 4x frame skipping reference mode for 2x frame skipping reference
H264E_REFSLICE_FOR_4X	Reference frame in 4x frame skipping reference mode
H264E_REFSLICE_BUTT	Non-reference mode

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

None

[See Also]

None

H264E_REF_TYPE_E

[Description]

Defines the frame type and reference attributes of the H.264 frame skipping reference streams.

[Syntax]

```
typedef enum hiH264E_REF_TYPE_E
{
    BASE_IDRSLICE = 0,
    BASE_PSLICE_REFBYBASE,
    BASE_PSLICE_REFBYENHANCE,
    ENHANCE_PSLICE_REFBYENHANCE,
    ENHANCE_PSLICE_NOTFORREF,
    ENHANCE_PSLICE_BUTT
} H264E_REF_TYPE_E;
```

[Member]

Member	Description
BASE_IDRSLICE	IDR frame at the base layer
BASE_PSLICE_REFBYBASE	P frame at the base layer, referenced by other frames at the base layer
BASE_PSLICE_REFBYENHANCE	P frame at the base layer, referenced by frames at the enhance layer
ENHANCE_PSLICE_REFBYENHANCE	P frame at the enhance layer, referenced by other frames at the enhance layer
ENHANCE_PSLICE_NOTFORREF	P frame at the enhance layer, not referenced

[Note]

None

[See Also]

None

JPEGE_PACK_TYPE_E

[Description]

Defines the PACK type of a JPEG stream.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
typedef enum hiJPEGE_PACK_TYPE_E
{
    JPEGE_PACK_ECS = 5,
    JPEGE_PACK_APP = 6,
    JPEGE_PACK_VDO = 7,
    JPEGE_PACK_PIC = 8,
    JPEGE_PACK_BUTT
} JPEGE_PACK_TYPE_E;
```

[Member]

Member	Description
JPEGE_PACK_ECS	ECS
JPEGE_PACK_APP	APP
JPEGE_PACK_VDO	VDO
JPEGE_PACK_PIC	PIC

[Note]

None

[See Also]

None

MPEG4E_PACK_TYPE_E

[Description]

Defines the Pack type of an MPEG-4 stream.

[Syntax]

```
typedef enum hiMPEG4E_PACK_TYPE_E
{
 MPEG4E_PACK_VOP_P = 1,
    MPEG4E_PACK_VOP_I = 5,
    MPEG4E_PACK_VOS   = 6,
    MPEG4E_PACK_VO    = 7,
    MPEG4E_PACK_VOL   = 8,
    MPEG4E_PACK_GVOP  = 9

} MPEG4E_PACK_TYPE_E;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
MPEG4E_PACK_VOS	Video object sequence type
MPEG4E_PACK_VO	Video object type
MPEG4E_PACK_VOL	Video object layer type
MPEG4E_PACK_GVOP	Group of video object plane types
MPEG4E_PACK_VOP_P	Video object plane type for the P frame
MPEG4E_PACK_VOP_I	Video packet type for the I frame

[Note]

None

[See Also]

None

VENC_DATA_TYPE_U

[Description]

Defines the stream result type.

[Syntax]

```
typedef union hiVENC_DATA_TYPE_U
{
    H264E_NALU_TYPE_E   enH264EType;
    JPEGE_PACK_TYPE_E   enJPEGEType;
    MPEG4E_PACK_TYPE_E  enMPEG4EType;
}VENC_DATA_TYPE_U;
```

[Member]

Member	Description
enH264EType	Type of an H.264 stream packet
enJPEGEType	Type of a JPEG stream packet
enMPEG4EType	Type of an MPEG-4 stream packet

[Note]

None

[See Also]

- H264E_NALU_TYPE_E
- JPEGE_PACK_TYPE_E

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- MPEG4E_PACK_TYPE_E

VENC_PACK_S

[Description]

Defines the structure of a stream packet.

[Syntax]

```
typedef struct hiVENC_PACK_S
{
    HI_U32              u32PhyAddr[2];
    HI_U8               *pu8Addr[2];
    HI_U32              u32Len[2];
    VENC_DATA_TYPE_U    DataType;
    HI_U64              u64PTS;
    HI_BOOL             bFieldEnd;
    HI_BOOL             bFrameEnd;
}VENC_PACK_S;
```

[Member]

Member	Description
pu8Addr[2]	Initial address of a stream packet
u32PhyAddr[2]	Physical address of a stream packet
u32Len[2]	Length of a stream packet
DataType	Type of a stream. H.264, JPEG, and MPEG-4 packets are supported.
u64PTS	Time stamp, in μs
bFieldEnd	Flag of ending a field The Hi3531 does not support field encoding. This member is invalid.
bFrameEnd	Flag of ending a frame Value range: HI_TRUE: indicates the last stream packet of a frame. HI_FALSE: indicates that the stream packet is not the last one of a frame.

[Note]

None

[See Also]

VENC_DATA_TYPE_U

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VENC_STREAM_INFO_H264_S

[Description]

Defines the features of an H.264 stream.

[Syntax]

```
typedef struct hiVENC_STREAM_INFO_H264_S
{
    HI_U32 u32PicBytesNum;
    HI_U32 u32PSkipMbNum;
    HI_U32 u32IpcmMbNum;
    HI_U32 u32Inter16x8MbNum;
    HI_U32 u32Inter16x16MbNum;
    HI_U32 u32Inter8x16MbNum;
    HI_U32 u32Inter8x8MbNum;
    HI_U32 u32Intra16MbNum;
    HI_U32 u32Intra8MbNum;
    HI_U32 u32Intra4MbNum;
    H264E_REFSLICE_TYPE_E enRefSliceType;
    H264E_REF_TYPE_E      enRefType;
    HI_U32 u32UpdateAttrCnt;
}VENC_STREAM_INFO_H264_S;
```

[Member]

Member	Description
u32PicBytesNum	Number of bytes to be encoded in the current frame
u32PSkipMbNum	Number of macroblocks in skip encoding mode to be encoded in the current frame
u32IpcmMbNum	Number of macroblocks in IPCM encoding mode to be encoded in the current frame
u32Inter16x8MbNum	Number of macroblocks in 16x8 inter-prediction encoding mode to be encoded in the current frame
u32Inter16x16MbNum	Number of macroblocks in 16x16 inter-prediction encoding mode to be encoded in the current frame
u32Inter8x16MbNum	Number of macroblocks in 8x16 inter-prediction encoding mode to be encoded in the current frame
u32Inter8x8MbNum	Number of macroblocks in 8x8 inter-prediction encoding mode to be encoded in the current frame
u32Intra16MbNum	Number of macroblocks in intra16 prediction encoding mode to be encoded in the current frame
u32Intra8MbNum	Number of macroblocks in intra8 prediction encoding mode to be encoded in the current frame

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u32Intra4MbNum	Number of macroblocks in intra4 prediction encoding mode to be encoded in the current frame
enRefSliceType	Encode the frame whose reference frame is in a specific frame skipping reference mode
enRefType	Type of encoded frame in advanced frame skipping reference mode
u32UpdateAttrCnt	Number of times for which channel attributes or parameters (including RC parameters) are set

[Note]

You can store only the H.264 streams whose reference frames are in a specific frame skipping reference mode. The details are as follows:

- In 1x frame skipping reference mode, you can store only the streams whose enRefSliceType value is H264E_REFSLICE_FOR_1X.
- In 2x frame skipping reference mode, you can store only the streams whose enRefSliceType value is H264E_REFSLICE_FOR_2X.
- In 4x frame skipping reference mode, you can store only the streams whose enRefSliceType value is H264E_REFSLICE_FOR_4X or store the streams whose enRefSliceType value is H264E_REFSLICE_FOR_2X or H264E_REFSLICE_FOR_4X.

[See Also]

- VENC_DATA_TYPE_U
- H264E_REFSLICE_TYPE_E

VENC_STREAM_INFO_JPEG_S

[Description]

Defines the features of a JPEG stream.

[Syntax]

```
typedef struct hiVENC_STREAM_INFO_JPEG_S
{
    HI_U32 u32PicBytesNum;
    HI_U32 u32UpdateAttrCnt;
    HI_U32 u32Qfactor;
}VENC_STREAM_INFO_JPEG_S
```

[Member]

Member	Description
u32PicBytesNum	Size of a JPEG stream, in byte
u32UpdateAttrCnt	Number of times for which channel attributes or parameters

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	(including RC parameters) are set
u32Qfactor	Encode Qfactor of the current frame

[Note]

None

[See Also]

VENC_DATA_TYPE_U

VENC_STREAM_INFO_MPEG4_S

[Description]

Defines the features of an MPEG-4 stream.

[Syntax]

```
typedef struct hiVENC_STREAM_INFO_MPEG4_S
{
   HI_U32 u32PicBytesNum;
   HI_U32 u32UpdateAttrCnt;
}VENC_STREAM_INFO_MPEG4_S;
```

[Member]

Member	Description
u32PicBytesNum	Size of an MPEG-4 stream, in byte
u32UpdateAttrCnt	Number of times for which channel attributes or parameters (including RC parameters) are set

[Note]

None

[See Also]

VENC_DATA_TYPE_U

VENC_STREAM_S

[Description]

Defines the stream frame type.

[Syntax]

```
typedef struct hiVENC_STREAM_S
{
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    VENC_PACK_S  *pstPack;
    HI_U32       u32PackCount;
    HI_U32       u32Seq;
    union
    {
        VENC_STREAM_INFO_H264_S stH264Info;
        VENC_STREAM_INFO_JPEG_S stJpegInfo;
        VENC_STREAM_INFO_MPEG4_S stMpeg4Info;
    };
}VENC_STREAM_S
```

[Member]

Member	Description
pstPack	Structure of a stream frame
u32PackCount	Number of stream packets per frame
u32Seq	Sequence number of a stream The sequence number of a frame is obtained by frame and the sequence number of a packet is obtained by packet.
stH624Info/ stJpegInfo/stMpeg4Info	Features of a stream

[Note]

None

[See Also]

- VENC_PACK_S
- HI_MPI_VENC_GetStream

VENC_ATTR_H264_S

[Description]

Defines the attribute structure of H.264 encoding.

[Syntax]

```
typedef struct hiVENC_ATTR_H264_S
{
    HI_U32  u32MaxPicWidth;
    HI_U32  u32MaxPicHeight;
    HI_U32  u32BufSize;
    HI_U32  u32Profile;
    HI_BOOL bByFrame;
    HI_S32  bField;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_S32  bMainStream;
    HI_U32  u32Priority;
    HI_BOOL bVIField;
    HI_U32  u32PicWidth;
    HI_U32  u32PicHeight;
}VENC_ATTR_H264_S;
```

[Member]

Member	Description
u32MaxPicWidth	Maximum width of a picture to be encoded, in pixel Value range: [MIN_WIDTH, MAX_WIDTH] The value must be an integral multiple of MIN_ALIGN. Static attribute
u32PicWidth	Width of a picture to be encoded, in pixel Value range: [MIN_WIDTH, u32MaxPicWidth] The value must be an integral multiple of MIN_ALIGN. Dynamic attribute
u32MaxPicHeight	Maximum height of a picture to be encoded, in pixel Value range: [MIN_WIDTH, MAX_HEIGHT] The value must be an integral multiple of MIN_ALIGN. Static attribute
u32PicHeight	Height of a picture to be encoded, in pixel Value range: [MIN_HEIGHT, u32MaxPicHeight] The value must be an integral multiple of MIN_ALIGN. Dynamic attribute
u32BufSize	Stream buffer size, in byte Value range: [Min, Max] The value must be an integral multiple of 64. You are advised to set the buffer size to the maximum size of the picture to be encoded. That is, the recommended value is as follows: (u32MaxPicWidth x u32MaxPicHeight x 1.5) bytes Minimum value: 1/2 of the maximum size of the picture to be encoded in normal mode; (32 x 1024) bytes in memory reduction mode Maximum value: There is no limitation but much memory is consumed if the buffer size is large. Static attribute
bByFrame	Obtaining streams by frame/packet Value range: [HI_TRUE, HI_FALSE] • HI_TRUE: obtains streams by frame. • HI_ FALSE: obtains streams by packet.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	Static attribute
bField	Frame/field encoding mode Value range: [HI_TRUE, HI_FALSE] • HI_TRUE: field encoding. • HI_FALSE: frame encoding. Only frame encoding is supported. Static attribute
bMainStream	Flag of large and small streams Value range: [HI_TRUE, HI_FALSE] • HI_TRUE: large stream • HI_FALSE: small stream Only the large stream is supported. Static attribute
u32Priority	Channel priority This member is not used currently, and its value is not defined.
bVIField	Whether a picture is input by frame/field Value range: [HI_TRUE, HI_FALSE] • HI_TRUE: field • HI_FALSE: frame This member is not used currently, and its value is not defined. The recommended value is HI_FALSE. Static attribute
u32Profile	Encoding level Value range: [0, 3] 0: baseline 1: main profile 2: high profile 3: SVC-T Profile

[Note]

None

[See Also]

None

VENC_ATTR_MJPEG_S

[Description]

Defines the attributes of the MJPEG encoding.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
typedef struct hiVENC_ATTR_MJPEG_S
{
   HI_U32  u32MaxPicWidth;
   HI_U32  u32MaxPicHeight;
   HI_U32   u32BufSize;
   HI_BOOL  bByFrame;
   HI_BOOL  bMainStream;
   HI_BOOL  bVIField;
   HI_U32   u32Priority;
   HI_U32   u32PicWidth;
   HI_U32   u32PicHeight;
}VENC_ATTR_MJPEG_S;
```

[Member]

Member	Description
u32MaxPicWidth	Maximum width of a picture to be encoded, in pixel Value range: [MIN_WIDTH, MAX_WIDTH]. The value must be an integral multiple of MIN_ALIGN. Static attribute
u32PicWidth	Width of a picture to be encoded, in pixel Value range: [MIN_WIDTH, u32MaxPicWidth] The value must be an integral multiple of MIN_ALIGN. Dynamic attribute
u32MaxPicHeight	Maximum height of a picture to be encoded, in pixel Value range: [MIN_HEIGHT, MAX_HEIGHT] The value must be an integral multiple of MIN_ALIGN. Static attribute
u32PicHeight	Height of a picture to be encoded, in pixel Value range: [MIN_HEIGHT, u32MaxPicHeight] The value must be an integral multiple of MIN_ALIGN. Dynamic attribute
u32BufSize	Size of a buffer Value range: [Min, Max] (in byte) The value of this member must be an integral multiple of 64. Min: (u32MaxPicWidth x u32MaxPicHeight) in normal mode; (32 x 1024) bytes in memory reduction mode Max: There is no limitation but much memory is consumed if the buffer size is large. Static attribute

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
bByFrame	Stream acquisition mode Value range: [HI_TRUE, HI_FALSE] • HI_TRUE: obtains streams by frame. • HI_ FALSE: obtains streams by packet. Static attribute
bMainStream	Flag of large and small streams Value range: [HI_TRUE, HI_FALSE] • HI_TRUE: large stream • HI_FALSE: small stream Only the large stream is supported. Static attribute
bVIField	Flag of frame/field for an input picture Value range: [HI_TRUE, HI_FALSE] • HI_TRUE: field • HI_FALSE: frame This member is not used currently, and its value is not defined. The recommended value is HI_FALSE. Static attribute
u32Priority	Channel priority This member is not used currently, and its value is not defined.

[Note]

None

[See Also]

None

VENC_ATTR_JPEG_S

[Description]

Defines the attributes of the JPEG snapshot.

[Syntax]

```
typedef struct hiVENC_ATTR_JPEG_S
{
   HI_U32  u32MaxPicWidth;
   HI_U32  u32MaxPicHeight;
   HI_U32   u32BufSize;
   HI_BOOL  bByFrame;
   HI_BOOL  bVIField;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_U32   u32Priority;
    HI_U32   u32PicWidth;
    HI_U32   u32PicHeight;
}VENC_ATTR_JPEG_S;
```

[Member]

Member	Description
u32MaxPicWidth	Maximum width of a picture to be encoded, in pixel Value range: [MIN_WIDTH, MAX_WIDTH] The value must be an integral multiple of MIN_ALIGN. Static attribute
u32PicWidth	Width of a picture to be encoded Value range: [MIN_WIDTH, u32MaxPicWidth] The value must be an integral multiple of MIN_ALIGN. Dynamic attribute
u32MaxPicHeight	Maximum height of a picture to be encoded, in pixel Value range: [MIN_HEIGHT, MAX_HEIGHT] The value must be an integral multiple of MIN_ALIGN. Static attribute
u32PicHeight	Height of a picture to be encoded Value range: [MIN_HEIGHT, u32MaxPicHeight] The value must be an integral multiple of MIN_ALIGN. Dynamic attribute
u32BufSize	Buffer size Value range: [Min, Max] (in byte) Min: (u32MaxPicWidth x u32MaxPicHeight) in normal mode; (32 x 1024) bytes in memory reduction mode Max: There is no limitation but much memory is consumed if the buffer size is large. The member value must be an integral multiple of 64. Static attribute
bByFrame	Obtaining streams by frame/packet Value range: [HI_TRUE, HI_FALSE] • HI_TRUE: obtains streams by frame. • HI_ FALSE: obtains streams by packet. Static attribute
u32Priority	Channel priority This member is not used currently, and its value is not defined.
bVIField	Flag of frame/field for an input picture

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	Value range: [HI_TRUE, HI_FALSE] • HI_TRUE: field • HI_FALSE: frame This member is not used currently, and its value is not defined. The recommended value is HI_FALSE. Static attribute

[Note]

None

[See Also]

None

VENC_ATTR_MPEG4_S

[Description]

Defines the attributes of the MPEG-4 channel.

[Syntax]

```
typedef struct  hiVENC_ATTR_MPEG4_S
{
   HI_U32  u32MaxPicWidth;
   HI_U32  u32MaxPicHeight;
   HI_U32  u32BufSize;
   HI_BOOL bByFrame;
   HI_S32  bField;
   HI_S32  bMainStream;
   HI_U32  u32Priority;
   HI_BOOL bVIField;
   HI_U32  u32PicWidth;
   HI_U32  u32PicHeight;
}VENC_ATTR_MPEG4_S;
```

[Member]

Member	Description
u32MaxPicWidth	Maximum width of a picture to be encoded, in pixel Value range: [MIN_WIDTH, MAX_WIDTH] The value must be an integral multiple of MIN_ALIGN. Static attribute
u32PicWidth	Width of a picture to be encoded

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	Value range: [MIN_WIDTH, u32MaxPicWidth] The value must be an integral multiple of MIN_ALIGN. Dynamic attribute
u32MaxPicHeight	Maximum height of a picture to be encoded, in pixel Value range: [MIN_HEIGHT, MAX_HEIGHT] The value must be an integral multiple of MIN_ALIGN. Static attribute
u32PicHeight	Height of a picture to be encoded Value range: [MIN_HEIGHT, u32MaxPicHeight] The value must be an integral multiple of MIN_ALIGN. Dynamic attribute
u32BufSize	Stream buffer size, in byte Value range: [Min, Max] The value must be an integral multiple of 64. You are advised to set the buffer size to the maximum size of a picture to be encoded. That is, the recommended value is as follows: u32MaxPicWidth x u32MaxPicHeight x1.5 bytes Minimum value: 1/2 of the maximum size of the picture to be encoded in normal mode; (32 x 1024) bytes in memory reduction mode Maximum value: There is no limitation but much memory is consumed if the buffer size is large. Static attribute
bByFrame	Mode for obtaining streams Values: {HI_TRUE, HI_FALSE} • HI_TRUE: obtain streams by frame • HI_FALSE: obtain streams by packet Static attribute
bField	Perform encoding by frame or fields Values: {HI_TRUE, HI_FALSE} • HI_TRUE: perform encoding by field • HI_FALSE: perform encoding by frame Only encoding by frame is supported. Static attribute
bMainStream	Large or small stream flag Values: {HI_TRUE, HI_FALSE} • HI_TRUE: large stream • HI_FALSE: small stream Only the large stream is supported. Static attribute

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u32Priority	Channel priority This member is not used currently, and its value is not defined.
bVIField	Frame/Field flag of an input picture Values: {HI_TRUE, HI_FALSE} • HI_TRUE: field • HI_FALSE: frame Static attribute

[Note]

None

[See Also]

None

VENC_ATTR_S

[Description]

Defines the encoder attributes.

[Syntax]

```
typedef struct hiVENC_ATTR_S
{
    PAYLOAD_TYPE_E  enType;
    union
    {
        VENC_ATTR_H264_S  stAttrH264e;
        VENC_ATTR_MJPEG_S stAttrMjpeg;
        VENC_ATTR_JPEG_S  stAttrJpeg;
        VENC_ATTR_MPEG4_S stAttrMpeg4;
    };
}VENC_ATTR_S;
```

[Member]

Member	Description
enType	Protocol type
stAttrH264e/stAttrMjpeg/ stAttrJpeg/ stAttrMpeg4	Attributes of a protocol-compliant encoder

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[See Also]

HI_MPI_VENC_CreateChn

VENC_CHN_ATTR_S

[Description]

Defines the attributes of a VENC channel.

[Syntax]

```
typedef struct hiVENC_CHN_ATTR_S
{
    VENC_ATTR_S     stVeAttr;
    VENC_RC_ATTR_S stRcAttr;
}VENC_CHN_ATTR_S;
```

[Member]

Member	Description
stVeAttr	Encoder attributes
stRcAttr	RC attributes

[Note]

None

[See Also]

HI_MPI_VENC_CreateChn

VENC_CHN_STAT_S

[Description]

Defines the status of a VENC channel.

[Syntax]

```
typedef struct hiVENC_CHN_STAT_S
{
    HI_BOOL bRegistered;
    HI_U32  u32LeftPics;
    HI_U32  u32LeftStreamBytes;
    HI_U32 u32LeftStreamFrames;
    HI_U32  u32CurPacks;
    HI_U32 u32LeftRecvPics;
    HI_U32 u32LeftEncPics;
}VENC_CHN_STAT_S;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Member]

Member	Description
bRegistered	Flag of channel group registration Value range: [HI_TRUE, HI_FALSE] • HI_TRUE: registered • HI_FALSE: unregistered
u32LeftPics	Number of pictures to be encoded
u32LeftStreamBytes	Number of remaining bytes in a stream buffer
u32LeftStreamFrames	Number of remaining frames in a stream buffer
u32CurPacks	Number of stream packets in the current frame
u32LeftRecvPics	Number of frames to be received. This member is valid after HI_MPI_VENC_StartRecvPicEx is called.
u32LeftEncPics	Number of frames to be encoded. This member is valid after HI_MPI_VENC_StartRecvPicEx is called.

[Note]

None

[See Also]

HI_MPI_VENC_Query

VENC_PARAM_H264_SLICE_SPLIT_S

[Description]

Defines the slice structure of an H.264 channel.

[Syntax]

```
typedef struct hiVENC_PARAM_H264_SLICE_SPLIT_S
{
    HI_BOOL bSplitEnable;
    HI_U32  u32SplitMode;
    HI_U32  u32SliceSize;
} VENC_PARAM_H264_SLICE_SPLIT_S;
```

[Member]

Member	Description
bSplitEnable	Whether the slice function is enabled
u32SplitMode	Slice mode • 0: slice a frame by byte.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	• 1: slice a frame by rows of macroblocks. This member must be set to be less than 2.
u32SliceSize	When u32SplitMode is set to 0, it indicates the number of byte per slice. Minimum value: 128 Maximum value: min((0xFFFF), u32PicSize/2) where, u32PicSize = picwidth x picheight x 3/2. When u32SplitMode is set to 1, it indicates the number of rows of macroblocks occupied by each slice. Minimum value: 1 Maximum value: (Picture height + 15)/16

[Note]

None

[See Also]

- HI_MPI_VENC_SetH264SliceSplit
- HI_MPI_VENC_GetH264SliceSplit

VENC_PARAM_H264_INTER_PRED_S

[Description]

Defines the inter-prediction structure of an H.264 channel.

[Syntax]

```
typedef struct hiVENC_PARAM_H264_INTER_PRED_S
{
    /* search window */
    HI_U32 u32HWSize;
    HI_U32 u32VWSize;
    HI_BOOL bInter16x16PredEn;
    HI_BOOL bInter16x8PredEn;
    HI_BOOL bInter8x16PredEn;
    HI_BOOL bInter8x8PredEn;
    HI_BOOL bExtedgeEn;
} VENC_PARAM_H264_INTER_PRED_S;
```

[Member]

Member	Description
u32HWSize	Size of a horizontal search window

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u32VWSize	Size of a vertical search window
bInter16x16PredEn	16x16 inter-prediction. By default, this function is enabled.
bInter16x8PredEn	16x8 inter-prediction enable. By default, this function is enabled.
bInter8x16PredEn	8x16 inter-prediction enable. By default, this function is enabled.
bInter8x8PredEn	8x8 inter-prediction. By default, this function is enabled.
bExtedgeEn	When a search window is larger than a picture, this member indicates whether to enable edge extension of the picture. By default, edge extension is enabled.

[Note]

None

[See Also]

- HI_MPI_VENC_SetH264InterPred
- HI_MPI_VENC_GetH264InterPred

VENC_PARAM_H264_INTRA_PRED_S

[Description]

Defines the intra-prediction structure of an H.264 channel.

[Syntax]

```
typedef struct hiVENC_PARAM_H264_INTRA_PRED_S
{
   HI_BOOL bIntra16x16PredEn;
   HI_BOOL bIntraNxNPredEn;
   HI_U32 constrained_intra_pred_flag;
   HI_BOOL bIpcmEn;
}VENC_PARAM_H264_INTRA_PRED_S;
```

[Member]

Member	Description
bIntra16x16PredEn	16x16 intra-prediction enable. By default, this function is enabled. Value: • 0: disabled • 1: enabled
bIntraNxNPredEn	NxN intra-prediction enable. By default, this function is

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	enabled. Value: • 0: disabled • 1: enabled
constrained_intra_pred_flag;	Default value: 0 Value: 0 or 1
bIpcmEn	IP camera prediction enable. The default value varies depending on chip type. Value: 0 or 1

[Note]

For the specific meanings of the members, see the H.264 protocol.

[See Also]

- HI_MPI_VENC_SetH264IntraPred
- HI_MPI_VENC_GetH264IntraPred

VENC_PARAM_H264_TRANS_S

[Description]

Defines the transformation and quantization structure of an H.264 channel.

[Syntax]

```
typedef struct hiVENC_PARAM_H264E_TRANS_S
{
    HI_U32 u32IntraTransMode;
    HI_U32 u32InterTransMode;
    HI_BOOL bScalingListValid;
    HI_U8  InterScalingList8x8[64];
    HI_U8  IntraScalingList8x8[64];
    HI_S32 chroma_qp_index_offset;
}VENC_PARAM_H264_TRANS_S;
```

[Member]

Member	Description
u32IntraTransMode	Conversion mode for intra-prediction • 0: supports 4x4 and 8x8 transformation and high profile. • 1: supports 4x4 transformation and baseline, main, and high profiles. • 2: supports 8x8 transformation and high profile. The system sets this member based on the channel protocol

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	type.
u32InterTransMode	Transformation mode for inter-prediction • 0: supports 4x4 and 8x8 transformation and high profile. • 1: supports 4x4 transformation and baseline, main, and high profiles. • 2: supports 8x8 transformation and high profile. The system sets this member based on the channel protocol type.
bScalingListValid	InterScalingList8x8 and IntraScalingList8x8 are valid only in the H.264 high profile. Value: • 0: invalid • 1: valid
InterScalingList8x8	A quantization table for 8x8 inter-prediction. You can use your own quantization table in the high profile. Value range: [1, 255]
IntraScalingList8x8	A quantization table for 8x8 intra-prediction. You can use your own quantization table in the high profile. Value range: [1, 255]
chroma_qp_index_offset	For details, see the H.264 protocol. Default value: 0 Value range: [–12, +12]

[Note]

For the specific meanings of the members, see the H.264 protocol.

[See Also]

- HI_MPI_VENC_SetH264Trans
- HI_MPI_VENC_GetH264Trans

VENC_PARAM_H264_ENTROPY_S

[Description]

Defines the entropy structure of an H.264 channel.

[Syntax]

```
typedef struct hiVENC_PARAM_H264E_ENTROPY_S
{
    HI_U32 u32EntropyEncModeI;
    HI_U32 u32EntropyEncModeP;
    HI_U32 cabac_stuff_en;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_U32 Cabac_init_idc;
}VENC_PARAM_H264_ENTROPY_S;
```

[Member]

Member	Description
u32EntropyEncModeI	Entropy mode for I frame • 0: CAVLC • 1: CABAC Other values: undefined The baseline profile does not support CABAC.
u32EntropyEncModeP	Entropy mode for P frame • 0: CAVLC • 1: CABAC Other values: null The baseline profile does not support CABAC.
cabac_stuff_en	For details, see the H.264 protocol. By default, this member is set to 0. Value: 0 or 1
Cabac_init_idc	For details, see the H.264 protocol. The value range is 0–2. The default value is 0.

[Note]

None

[See Also]

- HI_MPI_VENC_SetH264Entropy
- HI_MPI_VENC_GetH264Entropy

VENC_PARAM_H264_POC_S

[Description]

Defines the POC structure of an H.264 channel.

[Syntax]

```
typedef struct hiVENC_PARAM_H264_POC_S
{
    HI_U32 pic_order_cnt_type;
}VENC_PARAM_H264_POC_S;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
pic_order_cnt_type	The value range is 0–2, and the default value is 2. For details, see the H.264 protocol.

[Note]

None

[See Also]

- HI_MPI_VENC_SetH264Poc
- HI_MPI_VENC_GetH264Poc

VENC_PARAM_H264_DBLK_S

[Description]

Defines the de-blocking structure of an H.264 channel.

[Syntax]

```
typedef struct hiVENC_PARAM_H264E_DBLK_S
{
    HI_U32 disable_deblocking_filter_idc;         /*Default value: 0. {0,
1, 2} */
    HI_S32 slice_alpha_c0_offset_div2;           /*Default value: 5. [-6,
+6]? */
    HI_S32 slice_beta_offset_div2;              /*Default value: 5. [-6,
+6]? */
}VENC_PARAM_H264_DBLK_S;
```

[Member]

Member	Description
disable_deblocking_filter_idc	The value range is 0–2, and the default value is 0. For details, see the H.264 protocol.
slice_alpha_c0_offset_div2	The value range is [–6, +6], and the default value is 5. For details, see the H.264 protocol.
slice_beta_offset_div2	The value range is [–6, +6], and the default value is 5. For details, see the H.264 protocol.

[Note]

None

[See Also]

- HI_MPI_VENC_SetH264Dblk

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_VENC_GetH264Dblk

VENC_PARAM_H264_VUI_S

[Description]

Defines the VUI structure of an H.264 channel.

[Syntax]

```
typedef struct hiVENC_PARAM_H264E_VUI_S
{
    HI_S32 timing_info_present_flag;
    HI_S32 num_units_in_tick;
    HI_S32 time_scale;
    HI_S32 fixed_frame_rate_flag;
}VENC_PARAM_H264_VUI_S;
```

[Member]

Member	Description
timing_info_present_flag	The parameter value is 0 or 1, and its default value is 0. For details, see the H.264 protocol.
num_units_in_tick	The parameter value is greater than 0, and its default value is 1. For details, see the H.264 protocol.
time_scale	The parameter value is greater than 0, and its default value is 60. For details, see the H.264 protocol.
fixed_frame_rate_flag;	The parameter value is 0 or 1, and its default value is 1. For details, see the H.264 protocol.

[Note]

None

[See Also]

- HI_MPI_VENC_SetH264Vui
- HI_MPI_VENC_GetH264Vui

VENC_PARAM_JPEG_S

[Description]

Defines the parameters of the JPEG encoder.

[Syntax]

```
typedef struct hiVENC_PARAM_JPEG_S
{
    HI_U32 u32Qfactor;
    HI_U8  u8YQt[64];
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_U8  u8CbQt[64];
    HI_U8  u8CrQt[64];
    HI_U32 u32MCUPerECS;
} VENC_PARAM_JPEG_S;
```

[Member]

Member	Description
u32Qfactor	The value range is [1, 99], and the default value is 90. For details, see the RFC2435 protocol.
u8YQt	Y quantization table Value range: [1, 255]
u8CbQt	Cb quantization table Value range: [1, 255]
u8CrQt	Cr quantization table Value range: [1, 255]
u32MCUPerECS	This member indicates the number of MCUs per ECS. By default, the value is set to 0, indicating no ECS division. u32MCUPerECS: [0, (picwidth + 15) >> 4 x (picheight + 15) >> 4 x 2]

[Note]

None

[See Also]

- HI_MPI_VENC_SetJpegParam
- HI_MPI_VENC_GetJpegParam

VENC_PARAM_MJPEG_S

[Description]

Defines the advanced parameters of an MJPEG encoding channel.

[Syntax]

```
typedef struct hiVENC_PARAM_MJPEG_S
{
    HI_U8  u8YQt[64];
    HI_U8  u8CbQt[64];
    HI_U8  u8CrQt[64];
    HI_U32 u32MCUPerECS;
} VENC_PARAM_MJPEG_S;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u8YQt	Y quantization table Value range: [1, 255]
u8CbQt	Cb quantization table Value range: [1, 255]
u8CrQt	Cr quantization table Value range: [1, 255]
u32MCUPerECS	Number of MCUs in each ECS. The default value 0 indicates that there are no ECSs. u32MCUPerECS: [0, (picwidth + 15) >> 4 x (picheight + 15) >> 4 x 2]

[Note]

None

[See Also]

- HI_MPI_VENC_SetMjpegParam
- HI_MPI_VENC_GetMjpegParam

VENC_PARAM_MPEG4_S

[Description]

Defines the parameters of the MPEG-4 encoder.

[Syntax]

```
typedef struct hiVENC_PARAM_MPEG4_S
{
    HI_BOOL bSplitEnable;
    HI_U32 u32SplitMode;
    HI_U32 u32PacketSize;
    HI_U32 u32HWSize;
    HI_U32 u32VWSize;
} VENC_PARAM_MPEG4_S;
```

[Member]

Member	Description
bSplitEnable	Packet split enable. Packets are not split by default. Value: • 0: disabled • 1: enabled
u32SplitMode	Mode of splitting packets

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	• 0: split packets by bits • 1: split packets by macroblock lines The value of 32SplitMode can be set to 0 or 1.
u32PacketSize	When u32SplitMode is 0, u32PacketSize indicates the number of bits in each packet. The minimum value of u32PacketSize is 128, and the maximum value of u32PacketSize is min ((0xFFFF), u32PicSize/2). The value of u32PicSize is calculated as follows: u32PicSize = picwidth x picheight x 3/2 When u32SplitMode is 1, u32PacketSize indicates the number of macroblock lines occupied by each slice. The minimum value of u32PacketSize is 1, and the maximum value of u32PacketSize is (Picture height + 15)/16.
u32HWSize	Size of a horizontal search window The default value is 1. Each value maps to the actual range of the search window. The details are as follows: 0000: [–16, +15] 0001: [–32, +31] Other values: undefined
u32VWSize	Size of a vertical search window The default value is 0. Each value maps to the actual range of the search window. The details are as follows: 000: [–16, +15] Other values: undefined

[Note]

None

[See Also]

- HI_MPI_VENC_SetMpeg4Param
- HI_MPI_VENC_GetMpeg4Param

VENC_ROI_CFG_S

[Description]

Defines an ROI.

[Syntax]

```
typedef struct hiVENC_ROI_CFG_S
{
    HI_U32  u32Index;
    HI_BOOL bEnable;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_BOOL bAbsQp;
    HI_S32  s32Qp;
    RECT_S  stRect;
}VENC_ROI_CFG_S;
```

[Member]

Member	Description
u32Index	Index of an ROI. The system supports indexes ranging from 0 to 7.
bEnable	Whether to enable this ROI
bAbsQp	QP mode of an ROI • HI_FALSE: relative QP • HI_TURE: absolute QP
s32Qp	QP value. When the QP mode is set to HI_FALSE, this member indicates the QP offset, and the value range is [–51, +51]. When the QP mode is set to HI_TRUE, this member indicates the macroblock QP value, and the value range is [0, 51].
stRect	Region of an ROI The value of s32X, s32Y, u32Width, and u32Height must be a multiple of 16.

[Note]

None

[See Also]

- HI_MPI_VENC_SetRoiCfg
- HI_MPI_VENC_GetRoiCfg

VENC_ATTR_H264_REF_MODE_E

[Description]

Defines the H.264 frame skipping reference mode.

[Syntax]

```
typedef enum hiVENC_ATTR_H264_REF_MODE_E
{
    H264E_REF_MODE_1X = 1,
    H264E_REF_MODE_2X = 2,
    H264E_REF_MODE_4X = 5,
    H264E_REF_MODE_BUTT,
}VENC_ATTR_H264_REF_MODE_E;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
H264E_REF_MODE_1X	Normal reference mode
H264E_REF_MODE_2X	Two frames are separated by two skipping frames.
H264E_REF_MODE_4X	Two frames are separated by four skipping frames.

[Note]

None

[See Also]

None

VENC_ATTR_H264_REF_PARAM_S

[Description]

Defines the advanced frame skipping reference parameters for H.264 encoding.

[Syntax]

```
typedef enum hiVENC_ATTR_H264_REF_PARAM_S
{
    HI_U32   u32Base;
    HI_U32   u32Enhance;
    HI_BOOL  bEnablePred;
}VENC_ATTR_H264_REF_PARAM_S;
```

[Member]

Member	Description
u32Base	Base layer period
u32Enhance	Enhance layer period
bEnablePred	Whether some frames at the base layer are referenced by other frames at the base layer. When **bEnablePred** is **HI_FALSE**, all frames at the base layer reference IDR frames.

[Note]

None

[See Also]

None

VENC_RC_ATTR_S

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Defines RC attributes for controlling the bit rate of a VENC channel.

[Syntax]

```
typedef struct hiVENC_RC_ATTR_S
{
    VENC_RC_MODE_E      enRcMode;
    union
    {
        VENC_ATTR_H264_CBR_S    stAttrH264Cbr;
        VENC_ATTR_H264_VBR_S    stAttrH264Vbr;
        VENC_ATTR_H264_FIXQP_S  stAttrH264FixQp;
        VENC_ATTR_H264_ABR_S    stAttrH264Abr;
        VENC_ATTR_MPEG4_VBR_S   stAttrMpeg4Vbr;
        VENC_ATTR_MPEG4_FIXQP_S stAttrMpeg4FixQp;
        VENC_ATTR_MPEG4_CBR_S   stAttrMpeg4Cbr;
        VENC_ATTR_MJPEG_FIXQP_S stAttrMjpegeFixQp;
        VENC_ATTR_MJPEG_CBR_S   stAttrMjpegeCbr;
        VENC_ATTR_MJPEG_VBR_S   stAttrMjpegeVbr;
    };
    HI_VOID*      pRcAttr ;

}VENC_RC_ATTR_S;
```

[Member]

Member	Description
enRcMode	RC mode
stAttrH264Cbr	CBR mode attribute of an H.264 encoding channel
stAttrH264Vbr	VBR mode attribute of an H.264 encoding channel
stAttrH264FixQp	FixQp mode attribute of an H.264 encoding channel
stAttrH264Abr	Average bit rate (ABR) mode attribute of an H.264 encoding channel (not supported)
stAttrMpeg4Vbr	VBR mode attribute of an MPEG-4 encoding channel
stAttrMpeg4FixQp	FixQp mode attribute of an MPEG-4 encoding channel
stAttrMpeg4Cbr	CBR mode attribute of an MPEG-4 encoding channel
stAttrMjpegeFixQp	FixQp mode attribute of an MJPEG encoding channel
stAttrMjpegeCbr	CBR mode attribute of an MJPEG encoding channel
stAttrMjpegeVbr	VBR mode attribute of an MJPEG encoding channel

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[See Also]

HI_MPI_VENC_CreateChn

VENC_RC_MODE_E

[Description]

Defines the mode of the RC for controlling the bit rate of a VENC channel.

[Syntax]

```
typedef enum hiVENC_RC_MODE_E
{
    VENC_RC_MODE_H264CBR = 1,
    VENC_RC_MODE_H264VBR,
    VENC_RC_MODE_H264ABR,
    VENC_RC_MODE_H264FIXQP,
    VENC_RC_MODE_MJPEGCBR,
    VENC_RC_MODE_MJPEGVBR,
    VENC_RC_MODE_MJPEGABR,
    VENC_RC_MODE_MJPEGFIXQP,
    VENC_RC_MODE_MPEG4CBR,
    VENC_RC_MODE_MPEG4VBR,
    VENC_RC_MODE_MPEG4ABR,
    VENC_RC_MODE_MPEG4FIXQP,
    VENC_RC_MODE_H264CBRv2,
    VENC_RC_MODE_H264VBRv2,
    VENC_RC_MODE_MJPEGCBRv2,
    VENC_RC_MODE_MJPEGVBRv2,
    VENC_RC_MODE_BUTT,
}VENC_RC_MODE_E;
```

[Member]

Member	Description
VENC_RC_MODE_H264CBR	H.264 CBR mode
VENC_RC_MODE_H264VBR	H.264 VBR mode
VENC_RC_MODE_H264ABR	H.264 ABR mode (not supported)
VENC_RC_MODE_H264FIXQP	H.264 FixQp mode
VENC_RC_MODE_MJPEGCBR	MJPEG CBR mode
VENC_RC_MODE_MJPEGVBR	MJPEG VBR mode
VENC_RC_MODE_MJPEGABR	MJPEG ABR mode (not supported)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
VENC_RC_MODE_MJPEGFIXQP	MJPEG FixQp mode
VENC_RC_MODE_MPEG4CBR	MPEG-4 CBR mode (not supported)
VENC_RC_MODE_MPEG4VBR	MPEG-4 VBR mode (not supported)
VENC_RC_MODE_MPEG4ABR	MPEG-4 ABR mode (not supported)
VENC_RC_MODE_MPEG4FIXQP	MPEG-4 FixQp mode
VENC_RC_MODE_H264CBRv2	H.264 CBR mode of the new bit rate control algorithm
VENC_RC_MODE_H264VBRv2	H.264 VBR mode of the new bit rate control algorithm
VENC_RC_MODE_MJPEGCBRv2	MJPEG CBR mode of the new bit rate control algorithm
VENC_RC_MODE_MJPEGVBRv2	MJPEG VBR mode of the new bit rate control algorithm

[Note]

None

[See Also]

HI_MPI_VENC_CreateChn

VENC_ATTR_H264_CBR_S

[Description]

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

[Syntax]

```
typedef struct hiVENC_ATTR_H264_CBR_S
{
    HI_U32      u32Gop;
    HI_U32      u32StatTime;
    HI_U32      u32ViFrmRate;
    HI_FR32     fr32TargetFrmRate;
    HI_U32      u32BitRate;
    HI_U32      u32FluctuateLevel;
} VENC_ATTR_H264_CBR_S;
```

[Member]

Member	Description
u32Gop	H.264 group of picture (GOP) value

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	Value range: [1, 65536]
u32StatTime	CBR statistics time, in second Value range: [1, 60]
u32ViFrmRate	VI frame rate, in frame/s Value range: [1, 60]
fr32TargetFrmRate	Output frame rate of the encoder, in frame/s Value range: (0, u32ViFrmRate]
u32BitRate	Average bit rate, in kbit/s Value range: [2, 40960]
u32FluctuateLevel	Fluctuation level of the maximum bit rate relative to the average bit rate Value range: [0, 5] The fluctuation level 0 is recommended.

[Note]

- ViFrmRate must be set to the actual input frame rate of the encoder. The RC calculates the actual frame rate and controls the bit rate based on the value of ViFrmRate. If the output frame rate of the VI channel is 30 and GROUP does not control the frame rate, ViFrmRate is set to 30. If the output frame rate of the VI channel is 30 and GROUP controls the frame rate, ViFrmRate and the input frame rate controlled by GROUP must be set to 30. If GROUP and RC control the frame rate, the final output frame rate is the smaller one. For example, if you set the input frame rate controlled by GROUP to 30 and the output frame rate to 6; the input frame rate controlled by the RC to 30 and the target frame rate to 10, the final output frame rate is 6. When you set TargetFrmRate, its data type is defined as the fractional type HI_FR32 that is the same as HI_U32. That is, the higher 16 bits indicate the denominator and the lower 16 bits indicate the numerator. To set TargetFrmRate to an integer, set the upper 16 bits to zeros. For example, if you set ViFrmRate to 25 and TargetFrmRate to 12, 12 frames will be selected from 25 frames for encoding, the other 13 frames are discarded. If you set ViFrmRate to 25 and TargetFrmRate to 15/2, the encoder will encode 15 frames in two seconds.
- TargetFrmRate can be set as follows:
  - If you want the integral frame rate 25, set TargetFrmRate to 25.
  - If you want the fractional frame rate 15/2, set TargetFrmRate to [15 + (2 << 16)].

[See Also]

HI_MPI_VENC_CreateChn

VENC_ATTR_H264_VBR_S

[Description]

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

[Syntax]

Copyright © HiSilicon Technologies Co., Ltd

```
typedef struct hiVENC_ATTR_H264_VBR_S
{
    HI_U32      u32Gop;
    HI_U32      u32StatTime;
    HI_U32      u32ViFrmRate;
    HI_FR32     fr32TargetFrmRate;
    HI_U32      u32MaxBitRate;
    HI_U32      u32MaxQp;
    HI_U32      u32MinQp;
}VENC_ATTR_H264_VBR_S;
```

[Member]

Member	Description
u32Gop	H.264 GOP value Value range: [1, 65536]
u32StatTime	VBR statistics time, in second Value range: [1, 60]
u32ViFrmRate	VI frame rate, in frame/s Value range: [1, 60]
fr32TargetFrmRate	Output frame rate of the encoder, in frame/s Value range: (0, u32ViFrmRate]
u32MaxBitRate	Maximum output bit rate of the encoder, in frame/s Value range: [2, 40960]
u32MaxQp	Maximum QP supported by the encoder Value range: (u32MinQp, 51]
u32MinQp	Minimum QP supported by the encoder Value range: [0, 51]

[Note]

For details, see the descriptions of u32ViFrmRate and fr32TargetFrmRate of VENC_ATTR_H264_CBR_S.

[See Also]

HI_MPI_VENC_CreateChn

VENC_ATTR_H264_FIXQP_S

[Description]

Defines the structure of the FixQP attribute of an H.264 encoding channel.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
typedef struct hiVENC_ATTR_H264_FIXQP_S
{
    HI_U32      u32Gop;
    HI_U32      u32ViFrmRate;
    HI_FR32     fr32TargetFrmRate ;
    HI_U32      u32IQp;
    HI_U32      u32PQp;
} VENC_ATTR_H264_FIXQP_S;
```

[Member]

Member	Description
u32Gop	H.264 GOP value Value range: [1, 65536]
u32ViFrmRate	VI frame rate, in frame/s Value range: [1, 60]
fr32TargetFrmRate	Output frame rate of the encoder, in frame/s Value range: (0, u32ViFrmRate]
u32IQp	QP values of all the macroblocks of I frames Value range: [0, 51]
u32PQp	QP values of all the macroblocks of P frames Value range: [0, 51]

[Note]

For details, see the descriptions of u32ViFrmRate and fr32TargetFrmRate of VENC_ATTR_H264_CBR_S.

[See Also]

HI_MPI_VENC_CreateChn

VENC_ATTR_H264_ABR_S

[Description]

Defines the structure of the ABR attribute of an H.264 encoding channel.

[Syntax]

```
typedef struct hiVENC_ATTR_H264_ABR_S
{
    HI_U32   u32Gop;              /*the interval of ISLICE. */
    HI_U32   u32StatTime;          /* the rate statistic time, the unit is
second(s) */
    HI_U32   u32ViFrmRate;         /* the input frame rate of the venc
channel */
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_FR32  fr32TargetFrmRate ;    /* the target frame rate of the venc
channel */
    HI_U32   u32AvgBitRate;         /* average bitrate */
    HI_U32   u32MaxBitRate;         /* the max bitrate */
}VENC_ATTR_H264_ABR_S;
```

[Member]

Member	Description
u32Gop	H.264 GOP value Value range: [1, 65536]
u32StatTime	ABR statistical time, in second Value range: [1, 60]
u32ViFrmRate	VI frame rate, in frame/s Value range: [1, 60]
fr32TargetFrmRate	Output frame rate of the encoder, in frame/s Value range: (0, u32ViFrmRate]
u32AvgBitRate	Average bit rate Value range: [2000, 40960000]
u32MaxBitRate	Maximum bit rate Value range: [2000, 40960000]

[Note]

For details, see the descriptions of u32ViFrmRate and fr32TargetFrmRate of VENC_ATTR_H264_CBR_S.

[See Also]

HI_MPI_VENC_CreateChn

VENC_ATTR_MPEG4_CBR_S

[Description]

Defines the structure of the CBR attribute of an MPEG-4 encoding channel.

[Syntax]

```
typedef struct hiVENC_ATTR_MPEG4_CBR_S
{
    HI_U32     u32Gop;
    HI_U32     u32StatTime;
    HI_U32     u32ViFrmRate;
    HI_FR32    fr32TargetFrmRate;
    HI_U32     u32BitRate;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_U32      u32FluctuateLevel;
}VENC_ATTR_MPEG4_CBR_S;
```

[Member]

Member	Description
u32Gop	MPEG-4 GOP value Value range: [1, 65536]
u32ViFrmRate	VI frame rate, in frame/s Value range: [1, 60]
fr32TargetFrmRate	Output frame rate of the encoder, in frame/s Value range: (0, u32ViFrmRate]
u32StatTime	CBR statistics time, in second Value range: [1, 60]
u32BitRate	Average bit rate, in kbit/s Value range: [2, 40960]
u32FluctuateLevel	Fluctuation level of the maximum bit rate relative to the average bit rate Value range: [0, 5] The fluctuation level 0 is recommended.

[Note]

For details, see the descriptions of u32ViFrmRate and fr32TargetFrmRate of VENC_ATTR_H264_CBR_S.

[See Also]

HI_MPI_VENC_CreateChn

VENC_ATTR_MPEG4_VBR_S

[Description]

Defines the structure of the VBR attribute of an MPEG-4 encoding channel.

[Syntax]

```
typedef struct hiVENC_ATTR_MPEG4_VBR_S
{
    HI_U32      u32Gop;
    HI_U32      u32StatTime;
    HI_U32      u32ViFrmRate;
    HI_FR32     fr32TargetFrmRate;
    HI_U32      u32MaxBitRate;
    HI_U32      u32MaxQp;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_U32      u32MinQp;
}VENC_ATTR_MPEG4_VBR_S;
```

[Member]

Member	Description
u32Gop	MPEG-4 GOP value Value range: [1, 65536]
u32StatTime	VBR statistical time, in second Value range: [1, 60]
u32ViFrmRate	VI frame rate, in frame/s Value range: [1, 60]
fr32TargetFrmRate	Output frame rate of the encoder, in frame/s Value range: (0, u32ViFrmRate]
u32MaxBitRate	Maximum output bit rate of the encoder, in kbit/s Value range: [2, 40960]
u32MaxQp	Maximum QP supported by the encoder Value range: (u32MinQp, 31]
u32MinQp	Minimum QP supported by the encoder Value range: [1, 31]

[Note]

For details, see the descriptions of u32ViFrmRate and fr32TargetFrmRate of VENC_ATTR_H264_CBR_S.

[See Also]

HI_MPI_VENC_CreateChn

VENC_ATTR_MPEG4_FIXQP_S

[Description]

Defines the structure of the FixQp attribute of an MPEG-4 encoding channel.

[Syntax]

```
typedef struct hiVENC_ATTR_MPEG4_FIXQP_S
{
    HI_U32      u32Gop;
    HI_U32      u32ViFrmRate;
    HI_FR32     fr32TargetFrmRate;
    HI_U32      u32IQp;
    HI_U32      u32PQp;
}VENC_ATTR_MPEG4_FIXQP_S;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Member]

Member	Description
u32Gop	MPEG-4 GOP value Value range: [1, 65536]
u32ViFrmRate	VI frame rate, in frame/s Value range: [1, 60]
fr32TargetFrmRate	Output frame rate of the encoder, in frame/s Value range: (0, u32ViFrmRate]
u32IQp	QP values of all the macroblocks of I frames Value range: [1, 31]
u32PQp	QP values of all the macroblocks of P frames Value range: [1, 31]

[Note]

For details, see the descriptions of u32ViFrmRate and fr32TargetFrmRate of VENC_ATTR_H264_CBR_S.

[See Also]

HI_MPI_VENC_CreateChn

VENC_ATTR_MJPEG_FIXQP_S

[Description]

Defines the structure of the FixQp attribute of an MJPEG encoding channel.

[Syntax]

```
typedef struct hiVENC_ATTR_MJPEG_FIXQP_S
{
    HI_U32      u32ViFrmRate;
    HI_FR32     fr32TargetFrmRate;
    HI_U32      u32Qfactor;
}VENC_ATTR_MJPEG_FIXQP_S;
```

[Member]

Member	Description
u32ViFrmRate	VI frame rate, in frame/s Value range: [1, 60]
fr32TargetFrmRate	Output frame rate of the encoder, in frame/s Value range: (0, u32ViFrmRate]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u32Qfactor	Qfactor for MJPEG encoding Value range: [1, 99]

[Note]

For details, see the descriptions of u32ViFrmRate and fr32TargetFrmRate of VENC_ATTR_H264_CBR_S.

[See Also]

HI_MPI_VENC_CreateChn

VENC_ATTR_MJPEG_CBR_S

[Description]

Defines the structure of the CBR attribute of an MJPEG encoding channel.

[Syntax]

```
typedef struct hiVENC_ATTR_MJPEG_CBR_S
{
    HI_U32      u32StatTime;
    HI_U32      u32ViFrmRate;
    HI_FR32     fr32TargetFrmRate;
    HI_U32      u32BitRate;
    HI_U32      u32FluctuateLevel;
} VENC_ATTR_MJPEG_CBR_S;
```

[Member]

Member	Description
u32StatTime	CBR statistics time, in second Value range: [1, 16]
u32ViFrmRate	Input frame rate of the encoder, in frame/s Value range: (0, 60]
fr32TargetFrmRate	Output frame rate of the encoder, in frame/s Value range: (0, u32ViFrmRate]
u32BitRate	Average bit rate, in kbit/s Value range: [2, 40960]
u32FluctuateLevel	Fluctuation level of the maximum bit rate relative to the average bit rate Value range: [0, 5] The fluctuation level 0 is recommended.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

For details, see the descriptions of u32ViFrmRate and fr32TargetFrmRate of VENC_ATTR_H264_CBR_S.

[See Also]

HI_MPI_VENC_CreateChn

VENC_ATTR_MJPEG_VBR_S

[Description]

Defines the structure of the VBR attribute of an MJPEG encoding channel.

[Syntax]

```
typedef struct hiVENC_ATTR_MJPEG_VBR_S
{
    HI_U32      u32StatTime;
    HI_U32      u32ViFrmRate;
    HI_FR32     fr32TargetFrmRate;
    HI_U32      u32MaxBitRate;
    HI_U32     u32MaxQfactor;
    HI_U32     u32MinQfactor;
}VENC_ATTR_MJPEG_VBR_S;
```

[Member]

Member	Description
u32StatTime	VBR statistics time, in second Value range: [1, 60]
u32ViFrmRate	Input frame rate of the encoder, in frame/s Value range: (0, 60]
fr32TargetFrmRate	Output frame rate of the encoder, in frame/s Value range: (0, u32ViFrmRate]
u32MaxBitRate	Maximum bit rate, in kbit/s Value range: [2, 40960]
u32MaxQfactor	Maximum quantization factor Value range: [1, 99]
u32MinQfactor	Minimum quantization factor Value range: [1, u32MaxQfactor)

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

For details, see the descriptions of u32ViFrmRate and fr32TargetFrmRate of VENC_ATTR_H264_CBR_S.

[See Also]

HI_MPI_VENC_CreateChn

VENC_SUPERFRM_MODE_EN

[Description]

Defines the mode of processing jumbo frames when the bit rate is being controlled.

[Syntax]

```
typedef enum hiRC_SUPERFRM_MODE_EN
{
    SUPERFRM_NONE,
    SUPERFRM_DISCARD,
    SUPERFRM_REENCODE,
    SUPERFRM_BUTT
}VENC_SUPERFRM_MODE_EN;
```

[Member]

Member	Description
SUPERFRM_NONE	No special measure is taken.
SUPERFRM_DISCARD	Jumbo frames are discarded.
SUPERFRM_REENCODE	Jumbo frames are re-encoded.

[Note]

None

[See Also]

HI_MPI_VENC_GetRcPara

VENC_PARAM_H264_CBR_S

[Description]

Defines the configurations of the advanced parameters related to the CBR control mode of an H.264 encoding channel.

[Syntax]

```
typedef struct hiVENC_PARAM_H264_CBR_S
{
    HI_U32  u32MinIprop;
    HI_U32  u32MaxIprop;
    HI_U32  u32MaxQp;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_U32   u32MaxStartQp;
    HI_U32  u32MinQp;
    HI_U32  u32MaxPPDeltaQp;
    HI_U32  u32MaxIPDeltaQp;
    HI_BOOL bLostFrmOpen;
    HI_U32  u32LostFrmBpsThr;
    VENC_SUPERFRM_MODE_EN  enSuperFrmMode;
    HI_U32  u32SuperIFrmBitsThr;
    HI_U32  u32SuperPFrmBitsThr;
    HI_S32 s32IPQPDelta;
    HI_U32  u32RQRatio[8];
    HI_U32  u32MinIQp;
}VENC_PARAM_H264_CBR_S;
```

[Member]

Member	Description
u32MinIprop	Minimum ratio of I frames to P frames Value range: (0, 100]
u32MaxIprop	Maximum ratio of I frames to P frames Value range: [u32MinIprop, 100]
u32MaxQp	Maximum frame QP value for controlling the picture quality Value range: (u32MinQp, 51]
u32MaxStartQp	Maximum output QP for controlling the bit rate of frames Value range: [u32MinQp, u32MaxQp] If the bit rate is controlled by macroblock, the QPs of some macroblocks may be greater than or less than u32MaxStartQp. However, the value range is still [u32MinQp, u32MaxQp].
u32MinQp	Minimum frame QP value for controlling bit rate fluctuation Value range: [0, 51]
u32MaxPPDeltaQp	Maximum QP value between P frames Value range: [0, 10]
u32MaxIPDeltaQp	Maximum QP value between I frames and P frames Value range: [0, 10]
bLostFrmOpen	Whether to discard frames when the instant bit rate is above the threshold Value: HI_TRUE: discard HI_FALSE: do not discard
u32LostFrmBpsThr	Instant bit rate threshold, in bit/s. If the current bit rate is above this threshold, frames can be discarded.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	Value range: [64 x 1024, 80 x 1024 x 1024]
enSuperFrmMode	Jumbo frame processing mode Value: SUPERFRM_NONE: No special measure is taken. SUPERFRM_DISCARD: Jumbo frames are discarded. SUPERFRM_REENCODE: Jumbo frames are re-encoded.
u32SuperIFrmBitsThr	Threshold for determining whether an I frame is a jumbo frame, in bit The threshold must be greater than or equal to 0.
u32SuperPFrmBitsThr	Threshold for determining whether a P frame is a jumbo frame, in bit The threshold must be greater than or equal to 0.
s32IPQPDelta	IP QP variance Value range: [–10, +10]
u32RQRatio	R-Q ratio Value range: [0, 100]
u32MinIQp	Minimum QP of an I frame. It is used to control the maximum number of bits if an I frame. Value range: [0, 51]

[Note]

None

[See Also]

- HI_MPI_VENC_SetRcPara
- HI_MPI_VENC_GetRcPara

VENC_PARAM_H264_VBR_S

[Description]

Defines the configurations of the advanced parameters related to the VBR control mode of an H.264 encoding channel.

[Syntax]

```
typedef struct hiVENC_PARAM_H264_VBR_S
{
    HI_S32  s32DeltaQP;
    HI_S32  s32ChangePos;
   HI_U32  u32MinIprop;
   HI_U32  u32MaxIprop;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_BOOL bLostFrmOpen;
    HI_U32  u32LostFrmBpsThr;
    VENC_SUPERFRM_MODE_EN  enSuperFrmMode;
    HI_U32  u32SuperIFrmBitsThr;
    HI_U32  u32SuperPFrmBitsThr;
    HI_U32  u32MinIQP;
}VENC_PARAM_H264_VBR_S;
```

[Member]

Member	Description
s32DeltaQP	Maximum QP variance value between frames when the QP value is adjusted in VBR mode Value range: [0, 10]
s32ChangePos	Percentage of the bit rate when the QP starts to be adjusted in VBR mode relative to the maximum bit rate Value range: [50, 100]
u32MinIprop	Minimum ratio of the I/P frame bit rates. The minimum ratio is 1.
u32MaxIprop	Maximum ratio of the I/P frame bit rate. The maximum ratio is 100.
bLostFrmOpen	Whether to enable the lost frame function
u32LostFrmBpsThr	Bit rate threshold for discarding frames. The threshold is valid only when the lost frame function is enabled.
enSuperFrmMode	Jumbo frame processing mode Value: SUPERFRM_NONE: No special measure is taken. SUPERFRM_DISCARD: Jumbo frames are discarded. SUPERFRM_REENCODE: Jumbo frames are re-encoded.
u32SuperIFrmBitsThr	Threshold for determining whether an I frame is a jumbo frame, in bit The threshold must be greater than or equal to 0.
u32SuperPFrmBitsThr	Threshold for determining whether a P frame is a jumbo frame, in bit The threshold must be greater than or equal to 0.
u32MinIQP	Minimum QP of an I frame. It is used to control the maximum number of bits if an I frame. Value range: [0, 51]

[Note]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

- HI_MPI_VENC_SetRcPara
- HI_MPI_VENC_GetRcPara

VENC_PARAM_MJPEG_CBR_S

[Description]

Defines the configurations of the advanced parameters related to the CBR control mode of an MJPEG encoding channel.

[Syntax]

```
typedef struct hiVENC_PARAM_MJPEG_CBR_S
{
    HI_U32  u32MaxQfactor;;
    HI_U32  u32MinQfactor;
    HI_BOOL bLostFrmOpen;
    HI_U32  u32LostFrmBpsThr;
    VENC_SUPERFRM_MODE_EN  enSuperFrmMode;
    HI_U32  u32SuperFrmBitsThr;
    HI_U32  u32RQRatio[8];
}VENC_PARAM_MJPEG_CBR_S;
```

[Member]

Member	Description
u32MaxQfactor	Maximum frame Qfactor for controlling bit rate fluctuation Value range: [u32MinQp, 99]
u32MinQfactor	Minimum frame Qfactor for controlling the picture quality Value range: [1, 99]
bLostFrmOpen	Whether to enable the lost frame function
u32LostFrmBpsThr	Bit rate threshold for discarding frames. The threshold is valid only when the lost frame function is enabled.
enSuperFrmMode	Jumbo frame processing mode Value: SUPERFRM_NONE: No special measure is taken. SUPERFRM_DISCARD: Jumbo frames are discarded. SUPERFRM_REENCODE: Jumbo frames are re-encoded.
u32SuperFrmBitsThr	Jumbo frame threshold, in bit The threshold must be greater than or equal to 0.
u32RQRatio	R-Q ratio Value range: [0, 100]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

None

[See Also]

- HI_MPI_VENC_SetRcPara
- HI_MPI_VENC_GetRcPara

VENC_PARAM_MJPEG_VBR_S

[Description]

Defines the configurations of the advanced parameters related to the VBR control mode of an MJPEG channel.

[Syntax]

```
typedef struct hiVENC_PARAM_MJPEG_VBR_S
{
    HI_S32  s32DeltaQfactor;
    HI_S32  s32ChangePos;
   HI_BOOL bLostFrmOpen;
   HI_U32  u32LostFrmBpsThr;
   VENC_SUPERFRM_MODE_EN  enSuperFrmMode;
   HI_U32  u32SuperFrmBitsThr;
 }VENC_PARAM_MJPEG_VBR_S;
```

[Member]

Member	Description
s32DeltaQfactor	Maximum Qfactor variance value between frames when the Qfactor value is adjusted in VBR mode Value range: [0, 10]
s32ChangePos	Percentage of the bit rate when the Qfactor starts to be adjusted in VBR mode relative to the maximum bit rate Value range: [50, 100]
bLostFrmOpen	Whether to enable the lost frame function
u32LostFrmBpsThr	Bit rate threshold for discarding frames. The threshold is valid only when the lost frame function is enabled.
enSuperFrmMode	Jumbo frame processing mode Value: SUPERFRM_NONE: No special measure is taken. SUPERFRM_DISCARD: Jumbo frames are discarded. SUPERFRM_REENCODE: Jumbo frames are re-encoded.
u32SuperFrmBitsThr	Jumbo frame threshold, in bit The threshold must be greater than or equal to 0.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

None

[See Also]

- HI_MPI_VENC_SetRcPara
- HI_MPI_VENC_GetRcPara

VENC_PARAM_MPEG4_CBR_S

[Description]

Defines the configurations of the advanced parameters related to the CBR control mode of an MPEG-4 encoding channel.

[Syntax]

```
typedef struct hiVENC_PARAM_MPEG4_CBR_S
{

    HI_U32  u32MaxQp;
    HI_U32  u32MaxStartQp;
    HI_U32  u32MinQp;
    HI_U32  u32MaxPPDeltaQp;
    HI_U32  u32MaxIPDeltaQp;
    HI_U32  u32MinIprop;
    HI_U32  u32MaxIprop;
    HI_BOOL bLostFrmOpen;
    HI_U32  u32LostFrmBpsThr;
    VENC_SUPERFRM_MODE_EN  enSuperFrmMode;
    HI_U32  u32SuperIFrmBitsThr;
    HI_U32  u32SuperPFrmBitsThr;
    HI_S32 s32IPQPDelta;
    HI_U32  u32RQRatio[8];
}VENC_PARAM_MPEG4_CBR_S;
```

[Member]

Member	Description
u32MaxQp	Maximum frame QP value for controlling the picture quality Value range: (u32MinQp, 31]
u32MaxStartQp	Maximum output QP for controlling the bit rate of frames Value range: [u32MinQp, u32MaxQp] If the bit rate is controlled by macroblock, the QPs of some macroblocks may be greater than or less than u32MaxStartQp. However, the value range is still

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	[u32MinQp, u32MaxQp].
u32MinQp	Minimum frame QP value for controlling bit rate fluctuation Value range: [1, 31]
u32MaxPPDeltaQp	Maximum QP value between P frames Value range: [0, 10]
u32MaxIPDeltaQp	Maximum QP value between I frames and P frames Value range: [0, 10]
u32MinIprop	Minimum ratio of the I/P frame bit rate. The minimum ratio is 1.
u32MaxIprop	Maximum ratio of the I/P bit rate. The maximum ratio is 100.
bLostFrmOpen	Whether to enable the lost frame function
u32LostFrmBpsThr	Bit rate threshold for discarding frames. The threshold is valid only when the lost frame function is enabled.
enSuperFrmMode	Jumbo frame processing mode Value: SUPERFRM_NONE: No special measure is taken. SUPERFRM_DISCARD: Jumbo frames are discarded. SUPERFRM_REENCODE: Jumbo frames are re-encoded.
u32SuperIFrmBitsThr	Threshold for determining whether an I frame is a jumbo frame, in bit The threshold must be greater than or equal to 0.
u32SuperPFrmBitsThr	Threshold for determining whether a P frame is a jumbo frame, in bit The threshold must be greater than or equal to 0.
s32IPQPDelta	IP QP variance Value range: [–10, +10]
u32RQRatio	R-Q ratio Value range: [0, 100]

[Note]

None

[See Also]

- HI_MPI_VENC_SetRcPara
- HI_MPI_VENC_GetRcPara

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VENC_PARAM_MPEG4_VBR_S

[Description]

Defines the configurations of the advanced parameters related to the VBR control mode of an MPEG-4 encoding channel.

[Syntax]

```
typedef struct hiVENC_PARAM_MPEG4_VBR_S
{
    HI_S32  s32DeltaQP;
    HI_S32  s32ChangePos;
    HI_U32  u32MinIprop;
    HI_U32  u32MaxIprop;
    HI_BOOL bLostFrmOpen;
    HI_U32  u32LostFrmBpsTh;
    VENC_SUPERFRM_MODE_EN  enSuperFrmMode;
    HI_U32  u32SuperIFrmBitsThr;
    HI_U32  u32SuperPFrmBitsThr;
}VENC_PARAM_MPEG4_VBR_S;;
```

[Member]

Member	Description
s32DeltaQP	Maximum QP variance value between frames when the QP value is adjusted in VBR mode Value range: [0, +10]
s32ChangePos	Percentage of the bit rate when the QP starts to be adjusted in VBR mode relative to the maximum bit rate Value range: [50, 100]
u32MinIprop	Minimum ratio of I frames to P frames Value range: (0, 100]
u32MaxIprop	Maximum ratio of I frames to P frames Value range: (u32MinIprop, 100]
bLostFrmOpen	Whether to enable the lost frame function
u32LostFrmBpsThr	Bit rate threshold for discarding frames. The threshold is valid only when the lost frame function is enabled.
enSuperFrmMode	Jumbo frame processing mode Value: SUPERFRM_NONE: No special measure is taken. SUPERFRM_DISCARD: Jumbo frames are discarded. SUPERFRM_REENCODE: Jumbo frames are re-encoded.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u32SuperIFrmBitsThr	Threshold for determining whether an I frame is a jumbo frame, in bit The threshold must be greater than or equal to 0.
u32SuperPFrmBitsThr	Threshold for determining whether a P frame is a jumbo frame, in bit The threshold must be greater than or equal to 0.

[Note]

None

[See Also]

- HI_MPI_VENC_SetRcPara
- HI_MPI_VENC_GetRcPara

VENC_PARAM_H264_CBRV2_S

[Description]

Defines the configurations of the advanced parameters related to the H.264 CBR mode of the new bit rate control algorithm.

[Syntax]

```
typedef struct hiVENC_PARAM_H264_CBRV2_S
{
    HI_U32  u32MinIprop;
    HI_U32  u32MaxIprop;
    HI_U32  u32MaxQp;
    HI_U32  u32MaxStartQp;
    HI_U32  u32MinQp;
    HI_BOOL bLostFrmOpen;
    HI_U32  u32LostFrmBpsThr;
    VENC_SUPERFRM_MODE_EN  enSuperFrmMode;
    HI_U32  u32SuperIFrmBitsThr;
    HI_U32  u32SuperPFrmBitsThr;
    HI_S32  s32IPQPDelta;
    HI_S32  s32QualityLevel;
    HI_S32  s32MaxReEncodeTimes;
    HI_U32  u32MinIQp;
}VENC_PARAM_H264_CBRV2_S;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u32MinIprop	Minimum ratio of I frames to P frames Value range: (0, 100]
u32MaxIprop	Maximum ratio of I frames to P frames Value range: (u32MinIprop, 100]
u32MaxQp	Maximum frame QP value for controlling the picture quality Value range: (u32MinQp, 51]
u32MaxStartQp	Maximum output QP for controlling the bit rate of frames Value range: [u32MinQp, u32MaxQp] If the bit rate is controlled by macroblock, the QPs of some macroblocks may be greater than or less than u32MaxStartQp. However, the value range is still [u32MinQp, u32MaxQp].
u32MinQp	Minimum frame QP value for controlling bit rate fluctuation Value range: [0, 51]
bLostFrmOpen	Whether to discard frames when the instant bit rate is above the threshold Value: HI_TRUE: discard HI_FALSE: do not discard
u32LostFrmBpsThr	Instant bit rate threshold, in bit/s. If the current bit rate is above this threshold, frames can be discarded. Value range: [64 x 1024, 80 x 1024 x 1024]
enSuperFrmMode	Jumbo frame processing mode Value: SUPERFRM_NONE: No special measure is taken. SUPERFRM_DISCARD: Jumbo frames are discarded. SUPERFRM_REENCODE: Jumbo frames are re-encoded.
u32SuperIFrmBitsThr	Threshold for determining whether an I frame is a jumbo frame, in bit The threshold must be greater than or equal to 0.
u32SuperPFrmBitsThr	Threshold for determining whether a P frame is a jumbo frame, in bit. The threshold must be greater than or equal to 0.
s32IPQPDelta	IP QP variance

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	Value range: [–10, +10]
s32QualityLevel	Quality level. A smaller value indicates a higher quality level. Value range: [1, 5]
s32MaxReEncodeTimes	Number of times each frame is re-encoded. The value 0 indicates that re-encoding is not performed. Value range: [0, 3]
u32MinIQp	Minimum QP of an I frame. It is used to control the maximum number of bits if an I frame. Value range: [0, 51]

[Note]

None

[See Also]

- HI_MPI_VENC_GetRcPara
- HI_MPI_VENC_SetRcPara

VENC_RC_PARAM_S

[Description]

Defines the advanced parameters for controlling the bit rate of a VENC channel.

[Syntax]

```
typedef struct hiVENC_RC_PARAM_S
{
    HI_U32 u32ThrdI[12];
    HI_U32 u32ThrdP[12];
    HI_U32 u32QpDelta;
    union
    {
        VENC_PARAM_H264_CBR_S stParamH264Cbr;
        VENC_PARAM_H264_VBR_S stParamH264VBR;
        VENC_ATTR_H264_ABR_S stParamH264ABR;
        VENC_PARAM_MJPEG_CBR_S stParamMjpegCbr;
        VENC_PARAM_MJPEG_VBR_S stParamMjpegVbr;
        VENC_PARAM_MPEG4_CBR_S stParamMpeg4Cbr;
        VENC_PARAM_MPEG4_VBR_S stParamMpeg4Vbr;
        VENC_PARAM_H264_CBRV2_S stParamH264CbrV2;
    };
    HI_VOID* pRcParam;
}VENC_RC_PARAM_S;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Member]

Member	Description
u32ThrdI	Mad threshold for controlling the macroblock-level bit rate of I frames Value range: [0, 255]
u32ThrdP	Mad threshold for controlling the macroblock-level bit rate of P frames Value range: [0, 255]
u32QpDelta	Bit rate controlled by row Value range: [0, 10]
stParamH264Cbr	Advanced parameter of the CBR control mode for an H.264 channel
stParamH264Vbr	Advanced parameter of the VBR control mode for an H.264 channel Advanced parameter of the H.264 VBR mode for the new bit rate control algorithm
stParamH264Abr	Advanced parameter of the ABR control mode for an H.264 channel
stParamMjpegCbr	Advanced parameter of the CBR control mode for an MJPEG channel
stParamMjpegVbr	Advanced parameter of the VBR control mode of an MJPEG channel
stParamMpeg4Cbr	Advanced parameter of the CBR control mode for an MPEG-4 channel
stParamMpeg4Vbr	Advanced parameter of the VBR control mode of an MPEG-4 channel
stParamH264CbrV2	Advanced parameter of the H.264 CBR mode for the new bit rate control algorithm
pRcParam	Reserved

[Note]

None

[See Also]

- HI_MPI_VENC_SetRcPara
- HI_MPI_VENC_GetRcPara

GROUP_CROP_CFG_S

[Description]

Defines the clip parameters of a channel group.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
typedef struct hiGROUP_CROP_CFG_S
{
    HI_BOOL bEnable;    /* Crop region enable */
    RECT_S  stRect;     /* Crop region. Note that s32X must be a multiple
of 16.*/
}GROUP_CROP_CFG_S;
```

[Member]

Member	Description
bEnable	Crop enable Value range: [HI_FALSE, HI_TRUE] HI_TRUE: enabled HI_FALSE: disabled
stRect	Crop region stRect.s32X: The parameter value must be 16-pixel-aligned. stRect.s32Y: There is no limitation. stRect.u32Width and s32Rect.u32Height meet the width and height requirements of a VENC channel.

[Note]

None

[See Also]

- HI_MPI_VENC_SetGrpCrop
- HI_MPI_VENC_GetGrpCrop

GROUP_FRAME_RATE_S

[Description]

Defines the parameters for controlling the frame rate of a channel group.

[Syntax]

```
typedef struct hiGROUP_FRAME_RATE_S
{
    HI_S32  s32ViFrmRate;
    HI_S32  s32VpssFrmRate;
} GROUP_FRAME_RATE_S;
```

[Member]

Member	Description
s32ViFrmRate	Input frame rate of a channel group

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
s32VpssFrmRate	Output frame rate of a channel group

[Note]

None

[See Also]

- HI_MPI_VENC_SetGrpFrmRate
- HI_MPI_VENC_GetGrpFrmRate

GROUP_COLOR2GREY_S

[Description]

Defines the color-to-gray function of a group.

[Syntax]

```
typedef struct hiGROUP_COLOR2GREY_S
{
   HI_BOOL  bColor2Grey;       /* Whether to enable Color2Grey.*/
}GROUP_COLOR2GREY_S;
```

[Member]

Member	Description
bColor2Grey	Enable or disable the color-to-gray function of a group

[Note]

None

[See Also]

- HI_MPI_VENC_SetGrpColor2Grey
- HI_MPI_VENC_GetGrpColor2Grey

GROUP_COLOR2GREY_CONF_S

[Description]

Defines the global parameters for the color-to-gray function.

[Syntax]

```
typedef struct hiGROUP_COLOR2GREY_CONF_S
{
   HI_BOOL     bEnable;
   HI_U32      u32MaxWidth;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_U32      u32MaxHeight;
}GROUP_COLOR2GREY_CONF_S;
```

[Member]

Member	Description
bEnable	Enable or disable the color-to-gray functions of all groups
u32MaxWidth	Maximum width of a group whose color-to-gray function will be enabled. The group width is the width of a channel that is registered with the group.
u32MaxHeight	Maximum height of a group whose color-to-gray function will be enabled. The group height is the height of a channel that is registered with the group.

[Note]

None

[See Also]

- HI_MPI_VENC_SetColor2GreyConf
- HI_MPI_VENC_GetColor2GreyConf

VENC_JPEG_SNAP_MODE_E

[Description]

Defines the snapshot mode of a JPEG VENC channel.

[Syntax]

```
typedef enum hiVENC_JPEG_SNAP_MODE_E
{
    JPEG_SNAP_ALL    = 0,
    JPEG_SNAP_FLASH  = 1,
    JPEG_SNAP_BUTT,
}VENC_JPEG_SNAP_MODE_E;
```

[Member]

Member	Description
JPEG_SNAP_ALL	Snapshot mode in which all captured pictures are encoded. This mode is the default snapshot mode of a JPEG channel.
JPEG_SNAP_FLASH	Snapshot mode in which only the pictures are captured when the camera flash is on are encoded. This mode is available only when the front-end camera flash works.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

None

[See Also]

- HI_MPI_VENC_SetJpegSnapMode
- HI_MPI_VENC_GetJpegSnapMode

VENC_RECV_PIC_PARAM_S

[Description]

Defines the number of received pictures.

[Syntax]

```
typedef enum hiVENC_RECV_PIC_PARAM_S
{
    HI_S32 s32RecvPicNum;
} VENC_RECV_PIC_PARAM_S;
```

[Member]

Member	Description
s32RecvPicNum	Number of frames received and encoded by the encoding channel

[Note]

None

[See Also]

HI_MPI_VENC_SetJpegSnapMode

VENC_RC_PRIORITY_E

[Description]

Defines the RC priority.

[Syntax]

```
typedef enum hiVENC_RC_PRIORITY_E
{
    VENC_RC_PRIORITY_BITRATE_FIRST = 1,
    VENC_RC_PRIORITY_FRAMEBITS_FIRST,
    VENC_RC_PRIORITY_BUTT,
} VENC_RC_PRIORITY_E;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
VENC_RC_PRIORITY_BITRATE_FIRST	The target bit rate takes priority.
VENC_RC_PRIORITY_FRAMEBITS_FIRST	The jumbo frame threshold takes priority.

[Note]

None

[See Also]

None

VENC_RC_LOSTFRM_MODE_E

[Description]

Defines the frame discarding modes.

[Syntax]

```
typedef enum hiVENC_RC_LOSTFRM_MODE_E
{
    LOSTFRM_NORMAL,                         /*normal mode*/
    LOSTFRM_PSKIP,                          /*pskip*/
    LOSTFRM_BUTT,
}VENC_RC_LOSTFRM_MODE_E;
```

[Member]

Member	Description
LOSTFRM_NORMAL	Normal frame discarding mode
LOSTFRM_PSKIP	Pskip frame encoding mode

[Note]

None

[See Also]

VENC_RC_PARAM_LOSTFRM_S

VENC_RC_PARAM_LOSTFRM_S

[Description]

Defines the frame discarding policy.

[Syntax]

```
typedef struct hiVENC_RC_PARAM_LOSTFRM_S
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
{
    HI_BOOL bLostFrmOpen;             /* Indicates whether to discard
frames to ensure stable bit rate when the instant bit rate is exceeded */
    HI_U32  u32LostFrmBpsThr;         /* the instant bit rate threshold */
    VENC_LOSTFRM_MODE_E enLostFrmMode;  /* lost frame strategy */
    HI_U32  u32LostFrmGaps; }VENC_RC_PARAM_LOSTFRM_S;
```

[Member]

Member	Description
bLostFrmOpen	Whether to discard frames when the instantaneous bit rate exceeds the threshold
u32LostFrmBpsThr	Instantaneous bit rate threshold Value range: [64x1024, 80x1024x1024]
enLostFrmMode	Enumeration of the frame discarding policy
u32EncFrmGaps	Maximum number of consecutive discarded frames Value range: [0, 65535]

[Note]

None

[See Also]

- HI_MPI_VENC_SetLostFrameStrategy
- HI_MPI_VENC_GetLostFrameStrategy

VENC_PARAM_H264_VIDEO_SIGNAL_S

[Description]

Defines the VideoSignal attributes of an H.264 encoding channel.

[Syntax]

```
typedef struct hiVENC_PARAM_H264E_VIDEO_SIGNAL_S
{
    HI_U8  video_signal_type_present_flag;
    HI_U8  video_format;
    HI_U8  video_full_range_flag;
    HI_U8  colour_description_present_flag;
    HI_U8  colour_primaries;
    HI_U8  transfer_characteristics;
    HI_U8  matrix_coefficients;
}VENC_PARAM_H264_VIDEO_SIGNAL_S;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
video_signal_type_present_flag	For details, see the H.264 protocol. The default value is 0. Value range: [0, 1]
video_format	For details, see the H.264 protocol. The default value is 0. Value range: [0, 5]
video_full_range_flag	For details, see the H.264 protocol. The default value is 0. Value range: [0, 1]
colour_description_present_flag	For details, see the H.264 protocol. The default value is 0. Value range: [0, 1]
colour_primaries	For details, see the H.264 protocol. The default value is 0. Value range: [0, 9]
transfer_characteristics	For details, see the H.264 protocol. The default value is 0. Value range: [1, 15]
matrix_coefficients	For details, see the H.264 protocol. The default value is 0. Value range: [0, 10]

[Note]

None

[See Also]

- HI_MPI_VENC_SetH264VideoSignal
- HI_MPI_VENC_GetH264VideoSignal

6.5 Error Codes

Table 6-15 describes the error codes for the VENC APIs.

Table 6-15 Error codes for the VENC APIs

Error Code	Macro Definition	Description
0xA0078001	HI_ERR_VENC_INVALID_DEVID	The device ID is invalid.
0xA0078002	HI_ERR_VENC_INVALID_CHNID	The channel ID is invalid.
0xA0078003	HI_ERR_VENC_ILLEGAL_PARAM	The parameter is invalid.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Macro Definition	Description
0xA0078004	HI_ERR_VENC_EXIST	The device, channel or resource to be created or applied for exists.
0xA0078005	HI_ERR_VENC_UNEXIST	The device, channel or resource to be used or destroyed does not exist.
0xA0078006	HI_ERR_VENC_NULL_PTR	The parameter pointer is null.
0xA0078007	HI_ERR_VENC_NOT_CONFIG	No parameter is set before use.
0xA0078008	HI_ERR_VENC_NOT_SUPPORT	The parameter or function is not supported.
0xA0078009	HI_ERR_VENC_NOT_PERM	The operation, for example, modifying static parameters, is forbidden.
0xA007800C	HI_ERR_VENC_NOMEM	The memory fails to be allocated due to some causes such as insufficient system memory.
0xA007800D	HI_ERR_VENC_NOBUF	The buffer fails to be allocated due to some causes such as oversize of the data buffer applied for.
0xA007800E	HI_ERR_VENC_BUF_EMPTY	The buffer is empty.
0xA007800F	HI_ERR_VENC_BUF_FULL	The buffer is full.
0xA0078010	HI_ERR_VENC_SYS_NOTREADY	The system is not initialized or the corresponding module is not loaded.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Contents

7 VDA 7-1
7.1 Overview 7-1
7.2 Functions 7-1
7.2.1 Concepts 7-1
7.2.2 Creating VDA Channels 7-3
7.2.3 Input Sources 7-5
7.2.4 Processing VDA Results 7-6
7.3 API Reference 7-7
7.4 Data Types 7-21
7.5 Error Codes 7-37

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figures

Figure 7-1 Memory allocation for the macroblock SAD.....7-6

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Tables

Table 7-1 Error codes for VDA APIs 7-37

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

7 VDA

7.1 Overview

The VDA module obtains video detection results by detecting the video luminance variance. The VDA module supports the motion detection (MD) mode and occlusion detection (OD) mode. Accordingly, the video detection results are classified into MD results and OD results. The details are as follows:

- In MD mode, the VDA module checks the video motion status received by the VDA channel, and outputs MD results. The results include the sum of absolute difference (SAD) for each macroblock of each frame, object (OBJ) region information, and number of alarm pixels for the entire frame.
- In OD mode, the VDA module detects whether the video received by the VDA channel is occluded, and outputs OD results.

You can create a VDA channel, bind the VI source, and start the VDA function. Each channel can be set to an operating mode and channel attributes are set based on the operating mode. For details, see the description of HI_MPI_VDA_CreateChn.

7.2 Functions

7.2.1 Concepts

- VDA interval

  Pictures are received from the input source at intervals to control the frame rate. The interval is adjusted based on the actual frame rate of the input source. For example, if the VI frame rate is 25 frames per second, it is recommended that five frames are received each second, and the interval is set to 4 frames. Unless otherwise specified, the frame described in this chapter is the picture that is received from the input source, and transmitted to the VDA channel.
- Macroblock

  A picture is divided into blocks (in pixels), for example, 8x8 or 16x16 blocks. Each block is called a macroblock.
- Macroblock size

  Only 8x8 and 16x16 macroblocks are supported currently.
- Macroblock SAD

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

The macroblock SAD is the sum of the absolute luminance differences between the macroblocks of two frames. The greater the macroblock SAD, the greater the luminance difference.

- Bits of the macroblock SAD

  The Hi35xx supports 8-bit or 16-bit macroblock SAD. That is, the maximum bits of the detected macroblock SAD are 16 bits. In 16-bit mode, the actual macroblock SAD is output, and high bandwidth is required; in 8-bit mode, upper eight bits of the actual macroblock SAD are output, and low bandwidth is required. You need to select the 8-bit or 16-bit mode as required.

- SAD threshold

  The Hi3531, Hi3532 or Hi3521 processes macroblocks by 4x4 macroblocks. The SAD threshold is used for internal processing of The Hi3531, Hi3532 or Hi3521. That is, the SAD threshold is used to check whether a macroblock is a motion one. If the macroblock SAD is greater than or equal to the SAD threshold, the macroblock is considered as a motion one. Based on the motion macroblocks, the MD OBJ region information, and number of alarm pixels for the entire frame, or OD information is obtained. The preceding macroblocks are 4x4 macroblocks processed in The Hi3531, Hi3532 or Hi3521. The SAD threshold is based on 4x4 macroblocks, which is independent of the configured macroblock size.

- Reference frame mode

  The reference frame mode includes static reference frame mode, dynamic reference frame mode, and user reference frame mode. The reference frame mode determines the way in which reference frames are generated during VDA processing.

  The user reference frame mode is not supported currently, and the dynamic reference frame mode is recommended.

- VDA algorithm

  The VDA algorithm includes the frame reference algorithm (VDA_ALG_REF) and background algorithm (VDA_ALG_BG). The VDA algorithm is specified when a channel is created, and the background algorithm is recommended.

  - Frame reference algorithm

    This algorithm indicates that a picture is used as the reference frame to obtain VDA results.

    The reference frame is obtained in either of the following ways:

    - Uses a fixed picture as the reference frame in static reference frame mode.
    - Uses the previous frame as the reference frame in dynamic reference frame mode

  - Background algorithm

    This algorithm indicates that the background picture of the current video is generated during VDA processing, and the background picture is used as the reference frame to obtain VDA results.

    The background picture is obtained in either of the following ways:

    - Uses a fixed frame as the background picture in static reference frame mode.
    - Checks the still parts of the video, and obtains the background picture through internal processing in dynamic reference frame mode.

- Background refresh weight

  The configured background update weight takes effect only when the background algorithm and dynamic reference frame mode are used. During VDA processing, still pictures are abstracted. The pixels of these pictures and background are blended. The formula is as follows:

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Pixels of the new background = (Blending weight of still pictures x Pixels of still pictures + (256 – Blending weight of still pictures) x Pixels of the old background)/256

The blending weight of still pictures is the background refresh weight. The greater the weight, the fast the background is refreshed. The recommended weight is 128.

- VDA result

  The VDA results are classified into MD results and OD results based on the operating mode of the channel. For details, see the description of VDA_DATA_S.

7.2.2 Creating VDA Channels

You can start the VDA function by creating a VDA channel, binding the channel and the input source, and calling related MPI to receive pictures. Only one operating mode can be set for a channel. The width, height, and attributes of a channel are configured based on its operating mode. The channel in MD mode is an MD channel, and the channel in OD mode is an OD channel. This section describes the configuration items of the two types of channels, and configuration methods.

Same Configuration Items

- VDA algorithm (enVdaAlg)

  The frame reference algorithm and background algorithm are supported.

- Reference frame mode (enRefMode)

  The static reference frame mode and dynamic reference frame mode are supported.

- VDA interval (u32VdaIntvl)

  You need to adjust the VDA interval to ensure optimum VDA performance. The maximum VDA performance is D1@ 15 fps or CIF@ 30 fps.

- Macroblock size (enMbSize)

  The macroblock sizes of 8x8 and 16x16 are supported.

- SAD output bits (enMbSadBits)

  The 8-bit and 16-bit modes are supported.

- Background refresh weight (u32BgUpSrcWgt)

  This item takes effect only when the background algorithm is used.

Configuration Items for the MD Channel

- Number of MD result buffers (u32MdBufNum)

  The MD channel outputs MD results such as the macroblock SADs of each frame. The results of each frame occupy a large memory, and a result queue is generated. You can specify the depth of the result queue, that is, the number of MD results buffers.

- SAD threshold (u32SadTh)

  The MD channel processes a frame, and output the MD results of the frame. All motion-related information is determined based on the SAD threshold.

- Maximum number of output OBJ regions (u32ObjNumMax)

  The Hi3531 or Hi3532 checks whether a macroblock is moved based on its macroblock SAD and the user-defined SAD threshold. By using the edge detecting algorithm, OBJ regions are detected, and the coordinates of the upper left and lower right corners of each OBJ region are output. The number of detected OBJ regions varies according to frames. You can set u32ObjNumMax to control the maximum number of output OBJ regions. The recommended value is 128.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Configuration Items for the OD Channel

- Number of OD regions (u32RgnNum)

  The OD allows you to divide a frame into multiple regions for occlusion detection. A maximum of four regions is supported for each frame. You can set the number of OD regions and set the attributes (astOdRgnAttr) of each region. The region array index ranges from 0 to (u32RgnNum – 1). The OD channel processes only one region of a frame each time.

- Region range (stRect)

  The region range specifies the position and size of a region in a frame. Regions can be occluded by each other. For details about restrictions, see the description of VDA_OD_RGN_ATTR_S.

- Region SAD threshold (u32SadTh)

  The functions of the region SAD threshold and MD SAD threshold are the same. The difference is that the region SAD threshold is valid only for the current region. You must set same or different SAD thresholds for each region.

- Region area alarm threshold (u32AreaTh)

  Based on the region SAD threshold, the VDA module detects the maximum OBJ region after processing a frame. When the percentage of the OBJ region in the current region area is greater than or equal to the region area alarm threshold, the current region is occluded. The region area alarm threshold is a percentage relative to the region area, and ranges from 0 to 100. The recommended value is 70%.

- Region occlusion count alarm threshold (u32OccCntTh)

  No alarm information is output when the VDA module detects that a region is occluded once. That is, the VDA module outputs the occlusion alarm information only after detecting that this region is occluded for several times. The number of occlusion times is the region occlusion count alarm threshold. When the number of occlusion times is greater than or equal to the threshold, alarm information is output, and the VDA module stops detecting the region.

- Uncover count (u32UnOccCntTh)

  The uncover count is called the error value. The region occlusion count stops until the uncover count reaches the configured error value, regardless of whether the VDA module detects that the region is not occluded. If the occlusion count is much greater than the uncover count, the VDA module considers that the region is occluded, and notifies you of the alarm information.

Configuration Methods

- Configuring the SAD threshold

  When the picture noise is large, the SAD threshold is large accordingly.

  If you configure the SAD threshold for the first time, do as follows:

  - Create an MD channel, set the macroblock size to 8x8, set the output bit to 16 bits, and output the macroblock SAD.
  - Perform occlusion operations, choose an intermediate value from the output SADs, and divide the intermediate value by 4 to obtain an approximate SAD threshold.
  - Create an OD channel, and adjust the approximate SAD threshold to obtain the optimum SAD threshold. You are advised to respectively increase and decrease the approximate SAD threshold by 16 for five times.

- Configuring the VDA interval

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

The following is an example using the application scenario of VI D1@ 30 fps, one MD channel, and one OD channel (one region).

In this case, it is recommended that you set both the MD interval and OD interval to 5.

The performance of the MD channel or OD channel is calculated as follows:

30/(5 + 1)=5 fps

Therefore, the total performance is 10 fps, which is allowed by the VDA performance.

- Configuring the number of MD result buffers (u32MdBufNum)

  When the number of MD result buffers is large, the occupied memory is high. This indicates that the VDA threshold is also high.

  When the number of MD result buffers is small, the occupied memory is low, and the frequency of obtaining results is high. Otherwise, VDA processing is blocked.

  The recommended value is 8.

- Configuring the macroblock size (enMbSize) and output bits (enMbSadBits)

  If the macroblock size set to 8x8, the SAD precision and required bandwidth is high; if the macroblock size is set to 16x16, the SAD precision and required bandwidth are low.

  In 8-bit output mode, the SAD precision and required bandwidth are low; in 16-bit output mode, the SAD precision and bandwidth are high.

  You need to balance the SAD precision against the bandwidth as required.

- Configuring u32OccCntTh and u32UnOccCntTh

  The u32OccCntTh, input source frame rate, and VDA interval determine the region occlusion duration for outputting the occlusion alarm information.

  In general, VI frame rate is set to 30 fps, VDA interval is set to 5, and u32OccCntTh is set to 20 (u32UnOccCntTh is ignored).

  In this case, the region occlusion duration is calculated as follows:

  20/(30/(5 + 1) = 4s

  This indicates that when the lens is occluded by an object, the occlusion alarm information is output about four seconds later. You can adjust the value of u32OccCntTh based on the VI frame rate, and VDA interval to obtain the expected region occlusion duration.

  u32UnOccCntTh is an auxiliary parameter. You are advised to set the value of u32UnOccCntTh to 1/10 of the value of u32OccCntTh or smaller. In the preceding example, u32OccCntTh is set to 20. You can set the value of u32UnOccCntTh to 2, 1, or 0.

7.2.3 Input Sources

You can select the following bound input sources:

- VI output
- Decoding output (not supported by the Hi3532 and Hi3518/Hi3516C)
- Bypass channel output of the VPSS
- Other VPSS channel output in user mode

  You can send pictures to a VDA channel by calling HI_MPI_VDA_UserSendPic without binding.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

7.2.4 Processing VDA Results

After you create a VDA channel, bind the channel and input source, and enable the channel to receive pictures, the related hardware is started for VDA processing. This section describes the results generated in the MD channel and OD channels.

MD Channel

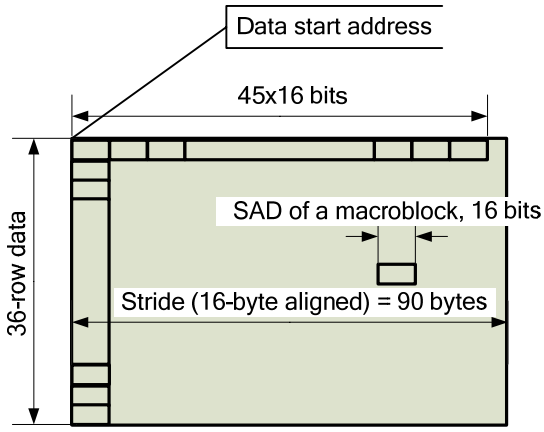
- MD results

  For details, see the description of VDA_MD_DATA_S.

- Macroblock SAD

  The SAD thresholds of all macroblocks in the entire frame are output. For details, see the description of VDA_MB_SAD_DATA_S. The SAD threshold consists of three variables: data start address, stride, and output bits. Assume that the macroblock size is 16x16, output bits are 16 bits, and the MD channel size is 720x576. The width and height of the MD channel is 45x36 in macroblocks. That is, there are 45x36 macroblocks. Figure 7-1 shows the memory allocation for the macroblock SAD. For details about related read and write operations, see related samples.

Figure 7-1 Memory allocation for the macroblock SAD



- OBJ region information

  The OBJ region is a rectangular motion part of the current frame that is detected by the hardware based on the SAD threshold. For details, see the description of VDA_OBJ_DATA_S. For details about related read and write operations, see related samples.

- Number of the alarm pixels in an entire frame

  It is the number of motion pixels in an entire frame. During the spray test, the Hi3531 or Hi3532 detects many but small OBJ regions. In this case, the number of the alarm pixels in an entire frame serves as an auxiliary parameter.

OD Channel

Only one alarm is provided for the OD channel. This alarm is used to check whether the specified region is occluded.

- Region alarm information

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

For details, see the description of VDA_OD_DATA_S. The region alarm information includes the number of regions, and Boolean array for identifying whether the regions are occluded. The value HI_TRUE indicates that a region is occluded and the value HI_FALSE indicates that a region is not occluded. For example, if you set three regions A, B, and C, and region B is occluded, the following message is displayed:

u32RgnNum = 3;abRgnAlarm[0] = 0, abRgnAlarm[1] = 1, abRgnAlarm[2] = 0, abRgnAlarm[3] = invalid

7.3 API Reference

The VDA provides the following MPIs:

- HI_MPI_VDA_CreateChn: Creates a VDA channel.
- HI_MPI_VDA_DestroyChn: Destroys a VDA channel.
- HI_MPI_VDA_SetChnAttr: Sets the attributes of a VDA channel.
- HI_MPI_VDA_GetChnAttr: Obtains the attributes of a VDA channel.
- HI_MPI_VDA_StartRecvPic: Starts to receive pictures.
- HI_MPI_VDA_StopRecvPic: Stops receiving pictures.
- HI_MPI_VDA_GetData: Obtains the VDA results.
- HI_MPI_VDA_ReleaseData: Release the buffer for storing the VDA results.
- HI_MPI_VDA_ResetOdRegion: Performs the region occlusion detection again.
- HI_MPI_VDA_Query: Queries the status of a VDA channel.
- HI_MPI_VDA_UpdateRef: Updates the reference picture.
- HI_MPI_VDA_GetFd: Obtains the device file descriptor (FD) corresponding to a VDA channel.
- HI_MPI_VDA_UserSendPic: Sends pictures to a VDA channel.

HI_MPI_VDA_CreateChn

[Description]

Creates a VDA channel.

[Syntax]

```
HI_S32 HI_MPI_VDA_CreateChn(VDA_CHN VdaChn, const VDA_CHN_ATTR_S
*pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
VdaChn	VDA channel ID. Value range: [0, VDA_CHN_NUM_MAX)	Input
pstAttr	Attributes of a VDA channel.	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success.
Other values	For details, see section 7.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vda.h, mpi_vda.h
- Library file: libmpi.a

[Note]

- Only one operating mode can be set for a VDA channel.
- A maximum of VDA_CHN_NUM_MAX channels can be created.
- If the VDA algorithm is set to frame reference algorithm, the background weight must also be set for checking parameters.
- The VDA function is enabled only after the input source is bound and HI_MPI_VDA_StartRecvPic is called.

[Example]

```
HI_S32 s32Ret = HI_SUCCESS;
    VDA_CHN VdaChn;
    VDA_CHN_ATTR_S stVdaChnAttr;
    MPP_CHN_S stSrcChn;
    MPP_CHN_S stDestChn;
    VdaChn = 0;
    stVdaChnAttr.enWorkMode = VDA_WORK_MODE_MD;
    stVdaChnAttr.u32Width   = 720;
    stVdaChnAttr.u32Height  = 576;
    stVdaChnAttr.unAttr.stMdAttr.enVdaAlg     = VDA_ALG_BG;
    stVdaChnAttr.unAttr.stMdAttr.enMbSize     = VDA_MB_16PIXEL;
    stVdaChnAttr.unAttr.stMdAttr.enMbSadBits   = VDA_MB_SAD_8BIT;
    stVdaChnAttr.unAttr.stMdAttr.enRefMode    = VDA_REF_MODE_DYNAMIC;
    stVdaChnAttr.unAttr.stMdAttr.u32VdaIntvl   = 12;
    stVdaChnAttr.unAttr.stMdAttr.u32BgUpSrcWgt = 128;
    stVdaChnAttr.unAttr.stMdAttr.u32MdBufNum   = 8;
    stVdaChnAttr.unAttr.stMdAttr.u32ObjNumMax  = 128;
    stVdaChnAttr.unAttr.stMdAttr.u32SadTh     = 40;

    /*Create a VDA channel.*/
    s32Ret = HI_MPI_VDA_CreateChn(VdaChn, &stVdaChnAttr);
    if(s32Ret != HI_SUCCESS)
    {
        return s32Ret;
    }
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    /*Bind the VDA channel and VI source.*/
    stSrcChn.enModId = HI_ID_VIU;
    stSrcChn.s32ChnId = 0;
    stDestChn.enModId = HI_ID_VDA;
    stDestChn.s32ChnId = VdaChn;

    s32Ret = HI_MPI_SYS_Bind(&stSrcChn, &stDestChn);
    if(s32Ret != HI_SUCCESS)
    {
        return s32Ret;
    }
    /*Enable the channel to receive pictures.*/
    s32Ret = HI_MPI_VDA_StartRecvPic(VdaChn);
    if(s32Ret != HI_SUCCESS)
    {
        return s32Ret;
    }

    /*Stop receiving pictures.*/
    s32Ret = HI_MPI_VDA_StopRecvPic(VdaChn);
    if(s32Ret != HI_SUCCESS)
    {
        return s32Ret;
    }
    /*Unbind the VDA channel and VI source.*/
    s32Ret = HI_MPI_SYS_UnBind(&stSrcChn, &stDestChn);
    if(s32Ret != HI_SUCCESS)
    {
        return s32Ret;
}
s32Ret = HI_MPI_VDA_DestroyChn(VdaChn);
if(s32Ret != HI_SUCCESS)
{
    return s32Ret;
}
```

[See Also]

HI_MPI_VDA_DestroyChn

HI_MPI_VDA_DestroyChn

[Description]

Destroys a VDA channel.

[Syntax]

Copyright © HiSilicon Technologies Co., Ltd

```
HI_S32 HI_MPI_VDA_DestroyChn(VDA_CHN VdaChn);
```

[Parameter]

Parameter	Description	Input/Output
VdaChn	VDA channel ID. Value range: [0, VDA_CHN_NUM_MAX)	Input

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see section 7.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vda.h, mpi_vda.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, you must create a VDA channel.
- Before calling this MPI, you must call HI_MPI_VDA_StopRecvPic to stop receiving pictures.

[Example]

For details, see the sample of HI_MPI_VDA_CreateChn.

[See Also]

None

HI_MPI_VDA_SetChnAttr

[Description]

Sets the attributes of a VDA channel.

[Syntax]

```
HI_S32 HI_MPI_VDA_SetChnAttr(VDA_CHN VdaChn, const VDA_CHN_ATTR_S
*pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
VdaChn	VDA channel ID. Value range: [0, VDA_CHN_NUM_MAX)	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
pstAttr	Attributes of a VDA channel.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see section 7.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vda.h, mpi_vda.h
- Library file: libmpi.a

[Note]

- Only dynamic attributes can be modified. For details about restrictions, see the description of VDA_CHN_ATTR_S.
- Before calling this MPI, you must create a VDA channel.
- You are advised to obtain the attributes of a VDA channel before setting attributes.
- The pointer to input channel attributes cannot be null.

[Example]

```
HI_S32 s32Ret = HI_SUCCESS;
    VDA_CHN VdaChn;
    VDA_CHN_ATTR_S stVdaChnAttrIn;
    VdaChn = 0;
    s32Ret = HI_MPI_VDA_GetChnAttr(VdaChn, &stVdaChnAttrIn);
    if(s32Ret != HI_SUCCESS)
    {
        return s32Ret;
    }
    /*Change the VDA interval.*/
    stVdaChnAttrIn.unAttr.stMdAttr.u32VdaIntvl   = 7;
    s32Ret = HI_MPI_VDA_SetChnAttr(VdaChn, &stVdaChnAttrIn);
    if(s32Ret != HI_SUCCESS)
    {
        return s32Ret;
    }
```

[See Also]

HI_MPI_VDA_GetChnAttr

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VDA_GetChnAttr

[Description]

Obtains the attributes of a VDA channel.

[Syntax]

```
HI_S32 HI_MPI_VDA_GetChnAttr(VDA_CHN VdaChn, VDA_CHN_ATTR_S *pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
VdaChn	VDA channel ID. Value range: [0, VDA_CHN_NUM_MAX)	Input
pstAttr	Attributes of a VDA channel.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see section 7.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vda.h, mpi_vda.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, you must create a VDA channel.
- The pointer to output channel attributes cannot be null.

[Example]

For details, see the sample of HI_MPI_VDA_SetChnAttr.

[See Also]

None

HI_MPI_VDA_StartRecvPic

[Description]

Starts to receive pictures.

[Syntax]

```
HI_S32 HI_MPI_VDA_StartRecvPic(VDA_CHN VdaChn);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VdaChn	VDA channel ID. Value range: [0, VDA_CHN_NUM_MAX)	Input

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see section 7.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vda.h, mpi_vda.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, you must create a VDA channel.
- You can call this MPI regardless of whether the input source is bound.
- This MPI can be called repeatedly.

[Example]

For details, see the sample of HI_MPI_VDA_CreateChn.

[See Also]

None

HI_MPI_VDA_StopRecvPic

[Description]

Stops receiving pictures.

[Syntax]

```
HI_S32 HI_MPI_VDA_StopRecvPic(VDA_CHN VdaChn);
```

[Parameter]

Parameter	Description	Input/Output
VdaChn	VDA channel ID. Value range: [0, VDA_CHN_NUM_MAX)	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success.
Other values	For details, see section 7.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vda.h, mpi_vda.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, you must create a VDA channel.
- You can call this MPI regardless of whether the input source is unbound.
- This MPI can be called repeatedly.

[Example]

For details, see the sample of HI_MPI_VDA_CreateChn.

[See Also]

None

HI_MPI_VDA_GetData

[Description]

Obtains the VDA results. If the VDA channel works in MD mode, MD results are obtained; of the VDA channel works in OD mode, OD results are obtained.

[Syntax]

```
HI_S32 HI_MPI_VDA_GetData(VDA_CHN VdaChn, VDA_DATA_S *pstVdaData, HI_BOOL bBlock);
```

[Parameter]

Parameter	Description	Input/Output
VdaChn	VDA channel ID. Value range: [0, VDA_CHN_NUM_MAX)	Input
pstVdaData	VDA results.	Output
bBlock	Block flag. Value: • HI_TRUE: block • HI_FALSE: non-block	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success.
Other values	For details, see section 7.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vda.h, mpi_vda.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, you must create a VDA channel.
- The VDA results can be obtained in block mode.
- The pointer to output results cannot be null.
- When the MD channel works in block mode and the MD result buffer is empty, you must wait until the MD processing is complete.
- When the MD channel works in non-block mode and the MD result buffer is empty, an error is returned.
- When VDA results are obtained from the MD channel, the time stamp in the results cannot be changed.
- VDA results can be obtained repeatedly from the OD channel in block mode or non-block mode.

[Example]

For details, see the related sample.

[See Also]

None

HI_MPI_VDA_ReleaseData

[Description]

Release the buffer for storing the VDA results.

[Syntax]

```
HI_S32 HI_MPI_VDA_ReleaseData(VDA_CHN VdaChn, const VDA_DATA_S*
pstVdaData);
```

[Parameter]

Parameter	Description	Input/Output
VdaChn	VDA channel ID. Value range: [0, VDA_CHN_NUM_MAX)	Input
pstVdaData	VDA results.	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see section 7.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vda.h, mpi_vda.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, you must create a VDA channel.
- The pointer to input results cannot be null.
- For the MD channel, you must release the VDA result buffer after VDA results are used. If the buffer is not released in a timely manner, the buffer is fully occupied, and the MD channel is blocked.
- After VDA results are obtained from the MD channel, the VDA result buffer cannot be released for multiple times.
- For the MD channel, if there is no result in the buffer, the release operation fails.
- After VDA results are obtained from the OD channel, you can determine the release times and whether to perform a release operation.
- It is recommended that you do not destroy the channel when no code is returned after this MPI is called. Otherwise, the channel is deleted successfully, but an error code is returned, indicating that the channel does not exist.

[Example]

For details, see the related sample.

[See Also]

HI_MPI_VDA_GetData

HI_MPI_VDA_ResetOdRegion

[Description]

Performs the region occlusion detection again. This MPI is valid only for the OD channel. Assume that four regions are set for an OD channel: A, B, C, and D. If region A is occluded, the VDA module stops checking region A, and outputs the region occlusion alarm information. The regions B, C, and D are still being checked.

After receiving the region occlusion alarm information, you can enable region A to work by calling the MPI and entering the channel ID and the array index 0 of region A. Then region A continues to be checked, which has no effect on other regions.

[Syntax]

```
HI_S32 HI_MPI_VDA_ResetOdRegion(VDA_CHN VdaChn, HI_S32 s32RgnIndex);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VdaChn	VDA channel ID. Value range: [0, VDA_CHN_NUM_MAX)	Input
s32RgnIndex	Region array index. Value range: [0, VDA_OD_RGN_NUM_MAX)	Input

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see section 7.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vda.h, mpi_vda.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, you must create a VDA channel.
- This MPI can be called repeatedly.
- After receiving the region occlusion alarm information, you need to call this MPI to enable the region to work.

[Example]

For details, see the related sample.

[See Also]

None

HI_MPI_VDA_Query

[Description]

Queries the status of a VDA channel. The status information includes the number of pictures in the channel picture queue, picture receiving status, and number of results in the result buffer.

[Syntax]

```
HI_S32 HI_MPI_VDA_Query(VDA_CHN VdaChn, VDA_CHN_STAT_S *pstChnStat);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VdaChn	VDA channel ID. Value range: [0, VDA_CHN_NUM_MAX)	Input
pstChnStat	Channel status.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see section 7.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vda.h, mpi_vda.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, you must create a VDA channel.
- The pointer to output channel status cannot be null.

[Example]

For details, see the related sample.

[See Also]

None

HI_MPI_VDA_UpdateRef

[Description]

Updates the reference frame. This MPI is valid only when the user reference frame mode is selected.

[Syntax]

```
HI_S32 HI_MPI_VDA_UpdateRef(VDA_CHN VdaChn, const VIDEO_FRAME_INFO_S
*pstRefFrame);
```

[Parameter]

Parameter	Description	Input/Output
VdaChn	VDA channel ID. Value range: [0, VDA_CHN_NUM_MAX)	Input
pstRefFrame	Reference picture information.	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see section 7.5 "Error Codes."

[Requirement]

- Header files: hi_comm_vda.h, mpi_vda.h
- Library file: libmpi.a

[Note]

In user reference frame mode, you must call this MPI to set reference frames first. Otherwise, an error occurs when HI_MPI_VDA_StartRecvPic() is called , and the VDA cannot receive pictures properly.

[Example]

None

[See Also]

None

HI_MPI_VDA_GetFd

[Description]

Obtains the device FD corresponding to a VDA channel.

[Syntax]

```
HI_S32 HI_MPI_VDA_GetFd(VDA_CHN VdaChn);
```

[Parameter]

Parameter	Description	Input/Output
VdaChn	VDA channel ID. Value range: [0, VDA_CHN_NUM_MAX)	Input

[Return Value]

Return Value	Description
Positive value	Valid return value
Non-positive value	Invalid return value

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_vda.h, mpi_vda.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

None

HI_MPI_VDA_UserSendPic

[Description]

Sends pictures to a VDA channel.

[Syntax]

```
HI_S32 HI_MPI_VDA_UserSendPic(VDA_CHN VdaChn, const VIDEO_FRAME_INFO_S
*pstUserFrame, HI_BOOL bBlock, HI_U32 u32MilliSec)
```

[Parameter]

Parameter	Description	Input/Output
VdaChn	VDA channel ID. Value range: [0, VDA_CHN_NUM_MAX)	Input
pstUserFrame	Pointer to the information about the pictures to be sent.	Input
bBlock	Block flag. Value: • HI_TRUE: block • HI_FALSE: non-block	Input
u32MilliSec	Wait timeout period in block mode, in ms.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Timeout or an error occurs. For details, see section 7.5 "Error Codes."

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_vda.h, mpi_vda.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, you must create a VDA channel.
- The channel must be receiving data.
- The pstUserFrame pointer cannot be empty.
- In block mode, if u32MilliSec is 0, the MPI is blocked until the VDA can receive pictures. If u32MilliSec is not 0, a code indicating success is returned when pictures are received in u32MilliSec ms. If the wait time is longer than u32MilliSec ms, timeout occurs and an error is returned.
- As the timeout period is in the unit of 10 ms, the maximum wait time must be a multiple of 10 ms.

[Example]

None

[See Also]

None

7.4 Data Types

The VDA data types are as follows:

- VDA_CHN: Defines a VDA channel.
- VDA_CHN_NUM_MAX: Defines the maximum number of VDA channels.
- VDA_MAX_WIDTH: Defines the maximum width of a VDA channel.
- VDA_MAX_HEIGHT: Defines the maximum height of a VDA channel.
- VDA_OD_RGN_NUM_MAX: Defines the maximum number of OD regions.
- VDA_OBJ_S: Defines the OBJ region information.
- VDA_REF_MODE_E: Defines the reference frame mode.
- VDA_ALG_E: Defines the VDA algorithm.
- VDA_MB_SIZE_E: Defines the macroblock size.
- VDA_MB_SADBITS_E: Defines the precision of the MD SAD.
- VDA_OD_RGN_ATTR_S: Defines the attributes of an occlusion region.
- VDA_MD_ATTR_S: Defines the MD attributes.
- VDA_OD_ATTR_S: Defines the OD attributes.
- VDA_WORK_MODE_E: Defines the VDA operating mode.
- VDA_WORK_MODE_ATTR_U: Defines the operating mode attributes.
- VDA_CHN_ATTR_S: Defines the channel attributes.
- VDA_MB_SAD_DATA_S: Defines the SAD information about a macroblock.
- VDA_OBJ_DATA_S: Defines the OBJ region results.
- VDA_MD_DATA_S: Defines the MD results.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- VDA_OD_DATA_S: Defines the OD results.
- VDA_DATA_U: Defines the VDA result union.
- VDA_DATA_S: Defines the VDA results.
- VDA_CHN_STAT_S: Defines the status of a VDA channel.

VDA_CHN

[Description]

Defines a VDA channel.

[Syntax]

```
typedef HI_S32 VDA_CHN;
```

[Member]

None

[Note]

None

[See Also]

None

VDA_CHN_NUM_MAX

[Description]

Defines the maximum number of VDA channels.

[Syntax]

For Hi3521/Hi3520A/Hi3518/Hi3516C/Hi3520D/Hi3515A/Hi3515C

```
#define VDA_CHN_NUM_MAX   32
```

For Hi3531/Hi3532

```
#define VDA_CHN_NUM_MAX   48
```

[Member]

None

[Note]

None

[See Also]

None

VDA_MAX_WIDTH

[Description]

Defines the maximum width of a VDA channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
#define VDA_MAX_WIDTH   960
```

[Member]

None

[Note]

None

[See Also]

None

VDA_MAX_HEIGHT

[Description]

Defines the maximum height of a VDA channel.

[Syntax]

```
#define VDA_MAX_HEIGHT   576
```

[Member]

None

[Note]

None

[See Also]

None

VDA_OD_RGN_NUM_MAX

[Description]

Defines the maximum number of OD regions.

[Syntax]

```
#define VDA_OD_RGN_NUM_MAX 4
```

[Member]

None

[Note]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

VDA_OBJ_S

[Description]

Defines the OBJ region information.

[Syntax]

```
typedef struct hiVDA_OBJ_S
{
    HI_U16 u16Left;
    HI_U16 u16Top;
    HI_U16 u16Right;
    HI_U16 u16Bottom;
}VDA_OBJ_S;
```

[Member]

Member	Description
u16Left	Horizontal coordinate of the upper left corner of an OBJ region.
u16Top	Vertical coordinate of the upper left corner of an OBJ region.
u16Right	Horizontal coordinate of the lower right corner of an OBJ region.
u16Bottom	Vertical coordinate of the lower right corner of an OBJ region.

[Note]

None

[See Also]

None

VDA_REF_MODE_E

[Description]

Defines the reference frame mode.

[Syntax]

```
typedef enum hiVDA_REF_MODE_E
{
    VDA_REF_MODE_DYNAMIC = 0,
    VDA_REF_MODE_STATIC,
    VDA_REF_MODE_USER,
    VDA_REF_MODE_BUTT
}VDA_REF_MODE_E;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
VDA_REF_MODE_DYNAMIC	Dynamic reference frame mode.
VDA_REF_MODE_STATIC	Static reference frame mode.
VDA_REF_MODE_USER	User reference frame mode (not supported).

[Note]

None

[See Also]

None

VDA_ALG_E

[Description]

Defines the VDA algorithm.

[Syntax]

```
typedef enum hiVDA_ALG_E
{
    VDA_ALG_BG = 0,
    VDA_ALG_REF,
    VDA_ALG_BUTT
}VDA_ALG_E;
```

[Member]

Member	Description
VDA_ALG_BG	Background algorithm.
VDA_ALG_REF	Frame reference algorithm.

[Note]

None

[See Also]

None

VDA_MB_SIZE_E

[Description]

Defines the macroblock size

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
typedef enum hiVDA_MB_SIZE_E
{
    VDA_MB_8PIXEL,      /* 8*8 */
    VDA_MB_16PIXEL,     /* 16*16 */
    VDA_MB_BUTT
}VDA_MB_SIZE_E;
```

[Member]

Member	Description
VDA_MB_8PIXEL	Macroblock size of 8x8.
VDA_MB_16PIXEL	Macroblock size of 16x16.

[Note]

None

[See Also]

None

VDA_MB_SADBITS_E

[Description]

Defines the precision of the MD SAD.

[Syntax]

```
typedef enum hiVDA_MB_SADBITS_E
{
   VDA_MB_SAD_8BIT = 0,  /*SAD precision of 8 bits*/
   VDA_MB_SAD_16BIT,     /*SAD precision of 16 bits*/
   VDA_MB_SAD_BUTT      /*reserved*/
} VDA_MB_SADBITS_E;
```

[Member]

Member	Description
VDA_MB_SAD_8BIT	8 bits
VDA_MB_SAD_16BIT	16 bits

[Note]

- The output value is based on the configured macroblock size.
- In 8-bit output mode, the upper eight bits among 16 bits are output.

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

VDA_OD_RGN_ATTR_S

[Description]

Defines the attributes of an occlusion region.

[Syntax]

```
typedef struct hiVDA_OD_RGN_ATTR_S
{
    RECT_S stRect;
    HI_U32 u32SadTh;
    HI_U32 u32AreaTh;
    HI_U32 u32OccCntTh;
    HI_U32 u32UnOccCntTh;
}VDA_OD_RGN_ATTR_S;
```

[Member]

Member	Description
stRect	Region range. X: value range of [0, VDA_MAX_WIDTH), 16-pixel alignment Y: value range of [0, VDA_MAX_HEIGHT) W: value range of [16, VDA_MAX_WIDTH), 16-pixel alignment H: value range of [16, VDA_MAX_HEIGHT), 16-pixel alignment X + W ≤ Channel width Y + H ≤ Channel height Static attribute.
u32SadTh	Region macroblock alarm threshold. Value range: [0, 4080]. Dynamic attribute.
u32AreaTh	Region area alarm threshold. Value range: [0, 100]. Dynamic attribute.
u32OccCntTh	Region occlusion count alarm threshold. Value range: [1, 126]. Dynamic attribute.
u32UnOccCntTh	Uncover count. Value range: [0, 256]. Dynamic attribute.

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[See Also]

None

VDA_MD_ATTR_S

[Description]

Defines the MD attributes.

[Syntax]

```
typedef struct hiVDA_MD_ATTR_S
{
    VDA_ALG_E        enVdaAlg;
    VDA_MB_SIZE_E    enMbSize;
    VDA_MB_SADBITS_E enMbSadBits;
    VDA_REF_MODE_E   enRefMode;
    HI_U32           u32MdBufNum;
    HI_U32           u32VdaIntvl;
    HI_U32           u32BgUpSrcWgt;
    HI_U32           u32SadTh;
    HI_U32           u32ObjNumMax;
}VDA_MD_ATTR_S;
```

[Member]

Member	Description
enVdaAlg	VDA algorithm. Static attribute.
enMbSize	Macroblock size. Static attribute.
enMbSadBits	SAD output precision. Static attribute.
enRefMode	Reference picture mode. Static attribute.
u32MdBufNum	Number of MD result buffers. Value range: [1, 16] Static attribute.
u32VdaIntvl	VDA interval. Value range: [0, 256], in frames. Dynamic attribute.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u32BgUpSrcWgt	Weight of source pictures when the background is updated. The total weight is 256. Value range: [1, 255]. The value 128 is recommended. Dynamic attribute.
u32SadTh	SAD alarm threshold. Dynamic attribute.
u32ObjNumMax	Number of output OBJ regions. Value range: [1, 128] Dynamic attribute.

[Note]

None

[See Also]

None

VDA_OD_ATTR_S

[Description]

Defines the OD attributes.

[Syntax]

```
typedef struct hiVDA_OD_ATTR_S
{
    HI_U32          u32RgnNum;
    VDA_OD_RGN_ATTR_S astOdRgnAttr[VDA_OD_RGN_NUM_MAX];
    VDA_ALG_E        enVdaAlg;
    VDA_MB_SIZE_E    enMbSize;
    VDA_MB_SADBITS_E enMbSadBits;
    VDA_REF_MODE_E  enRefMode;
    HI_U32          u32VdaIntvl;
    HI_U32          u32BgUpSrcWgt;
}VDA_OD_ATTR_S;
```

[Member]

Member	Description
u32RgnNum	Number of regions. Value range: [1, VDA_OD_RGN_NUM_MAX] Static attribute.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
astOdRgnAttr[VDA_OD_RGN_NUM_MAX]	Region attribute. For details, see the internal structure.
enVdaAlg	VDA algorithm. Static attribute.
enMbSize	Macroblock size. Static attribute.
enMbSadBits	SAD output precision. Static attribute.
enRefMode	Reference picture mode. Static attribute.
u32VdaIntvl	VDA interval. Value range: [0, 256], in frames. Dynamic attribute.
u32BgUpSrcWgt	Weight of source pictures when the background is updated. The total weight is 256. Value range: [1, 255]. The value 128 is recommended. Dynamic attribute.

[Note]

None

[See Also]

None

VDA_WORK_MODE_E

[Description]

Defines the VDA operating mode.

[Syntax]

```
typedef enum hiVDA_WORK_MODE_E
{
    VDA_WORK_MODE_MD = 0,
    VDA_WORK_MODE_OD,
    VDA_WORK_MODE_BUTT
}VDA_WORK_MODE_E;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
VDA_WORK_MODE_MD	Motion detection.
VDA_WORK_MODE_OD	Occlusion detection.

[Note]

None

[See Also]

None

VDA_WORK_MODE_ATTR_U

[Description]

Defines the operating mode attributes.

[Syntax]

```
typedef union hiVDA_WORK_MODE_ATTR_U
{
    VDA_MD_ATTR_S stMdAttr;
    VDA_OD_ATTR_S stOdAttr;
}VDA_WORK_MODE_ATTR_U;
```

[Member]

Member	Description
stMdAttr	Motion detection attribute.
stOdAttr	Occlusion detection attribute.

[Note]

None

[See Also]

None

VDA_CHN_ATTR_S

[Description]

Defines the channel attributes.

[Syntax]

```
typedef struct hiVDA_CHN_ATTR_S
{
    VDA_WORK_MODE_E enWorkMode;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    VDA_WORK_MODE_ATTR_U unAttr;
    HI_U32 u32Width;
    HI_U32 u32Height;
}VDA_CHN_ATTR_S;
```

[Member]

Member	Description
enWorkMode	Operating Mode. Static attribute.
unAttr	Operating mode attribute.
u32Width	Width of a channel. Value range: [16, VDA_MAX_WIDTH], 16-byte alignment. Static attribute.
u32Height	Channel height. Value range: [16, VDA_MAX_HEIGHT], 16-byte alignment. Static attribute.

[Note]

The channel size can be different from the input picture size. However, the width and height of an input picture must be greater than or equal to the channel width and height.

[See Also]

None

VDA_MB_SAD_DATA_S

[Description]

Defines the SAD information about a macroblock.

[Syntax]

```
typedef struct hiVDA_MB_SAD_DATA_S
{
    HI_VOID    *pAddr;           /*address*/
    HI_U32      u32Stride;        /*stride*/
    VDA_MB_SADBITS_E enMbSadBits;  /*MB SAD size*/
}VDA_MB_SAD_DATA_S;
```

[Member]

Member	Description
pAddr	Start address of the memory for storing macroblock SADs.
u32Stride	Stride of the memory for storing the macroblock data, in bytes.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
enMbSadBits	SAD output bits.

[Note]

None

[See Also]

None

VDA_OBJ_DATA_S

[Description]

Defines the OBJ region information.

[Syntax]

```
typedef struct hiVDA_OBJ_DATA_S
{
    HI_U32 u32ObjNum;
    VDA_OBJ_S    *pstAddr;
    HI_U32 u32IndexOfMaxObj;
    HI_U32 u32SizeOfMaxObj;
    HI_U32 u32SizeOfTotalObj;
}VDA_OBJ_DATA_S;
```

[Member]

Member	Description
u32ObjNum	Number of OBJ regions.
pstAddr	Start address of the memory for storing the OBJ region information.
u32IndexOfMaxObj	Index of the maximum OBJ region.
u32SizeOfMaxObj	Size of the maximum OBJ region.
u32SizeOfTotalObj	Total size of OBJ regions.

[Note]

- The index of the maximum OBJ region is numbered from 0.
- If the reported coordinates of an OBJ region are (0, 4), (16, 36), the size of the OBJ area is calculated as follows:

  (16 – 0 + 4) x (36 – 4 + 4) = 720

  The width of the OBJ region is calculated as follows:

  Actual width of the OBJ region = Width obtained from the reported region coordinates + 4.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

The height of the OBJ region is calculated in the same way.

- The detected VDA_OBJ_S information is stored in the memory starting from pstAddr in sequence. A maximum of u32ObjNum data segments are allowed.

[See Also]

None

VDA_MD_DATA_S

[Description]

Defines the MD results.

[Syntax]

```
typedef struct hiVDA_MD_DATA_S
{
    HI_BOOL          bMbSadValid;
    VDA_MB_SAD_DATA_S stMbSadData;
    HI_BOOL          bObjValid;
    VDA_OBJ_DATA_S    stObjData;
    HI_BOOL          bPelsNumValid;
    HI_U32            u32AlarmPixCnt;
}VDA_MD_DATA_S;
```

[Member]

Member	Description
bMbSadValid	Validity of the macroblock SAD. HI_TRUE: valid HI_FALSE: invalid
stMbSadData	Macroblock SAD result.
bObjValid	Validity of the OBJ region. HI_TRUE: valid HI_FALSE: invalid
stObjData	OBJ region result.
bPelsNumValid	Validity of the number of alarm pixels. HI_TRUE: valid HI_FALSE: invalid
u32AlarmPixCnt	Total number of alarm pixels.

[Note]

By default, bMbSadValid, bObjValid, and bPelsNumValid are valid. No MPIs are provided for setting these members.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

None

VDA_OD_DATA_S

[Description]

Defines the OD results.

[Syntax]

```
typedef struct hiVDA_OD_DATA_S
{
    HI_U32  u32RgnNum;
    HI_BOOL abRgnAlarm[VDA_OD_RGN_NUM_MAX];
}VDA_OD_DATA_S;
```

[Member]

Member	Description
u32RgnNum	Number of regions.
abRgnAlarm[VDA_OD_RGN_NUM_MAX]	Region alarm flag. HI_TRUE: alarm HI_FALSE: no alarm

[Note]

None

[See Also]

None

VDA_DATA_U

[Description]

Defines the VDA result union.

[Syntax]

```
typedef union hiVDA_DATA_U
{
    VDA_MD_DATA_S stMdData;
    VDA_OD_DATA_S stOdData;
}VDA_DATA_U;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
stMdData	Motion detection result.
stOdData	Occlusion detection result.

[Note]

None

[See Also]

None

VDA_DATA_S

[Description]

Defines the VDA results.

[Syntax]

```
typedef struct hiVDA_DATA_S
{
    VDA_WORK_MODE_E enWorkMode;
    VDA_DATA_U      unData;
    VDA_MB_SIZE_E   enMbSize;
    HI_U32          u32MbWidth;
    HI_U32          u32MbHeight;
    HI_U64          u64Pts;
}VDA_DATA_S;
```

[Member]

Member	Description
enWorkMode	Operating mode.
unData	Result union.
enMbSize	Macroblock size.
u32MbWidth	Channel width, in macroblock.
u32MbHeight	Channel height, in macroblock.
u64Pts	PTS.

[Note]

None

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

VDA_CHN_STAT_S

[Description]

Defines the status of a VDA channel.

[Syntax]

```
typedef struct hiVDA_CHN_STAT_S
{
   HI_BOOL bStartRecvPic;
   HI_U32 u32LeftPic;
   HI_U32 u32LeftRst;
}VDA_CHN_STAT_S;
```

[Member]

Member	Description
bStartRecvPic	Whether to receive pictures.
u32LeftPic	Number of remaining pictures in the channel picture queue.
u32LeftRst	Number of remaining results in the buffer.

[Note]

None

[See Also]

None

7.5 Error Codes

Table 7-1 describes the error codes for VDA APIs.

Table 7-1 Error codes for VDA APIs

Error Code	Error Code	Description
0xA0098001	HI_ERR_VDA_INVALID_DEVID	The device ID exceeds the valid range.
0xA0098002	HI_ERR_VDA_INVALID_CHNID	The channel ID exceeds the valid range.
0xA0098003	HI_ERR_VDA_ILLEGAL_PARAM	The parameter value exceeds its valid range.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Error Code	Description
0xA0098004	HI_ERR_VDA_EXIST	The device, channel, or resource to be created or applied for already exists.
0xA0098005	HI_ERR_VDA_UNEXIST	The device, channel, or resource to be used or destroyed does not exist.
0xA0098006	HI_ERR_VDA_NULL_PTR	The pointer is null.
0xA0098007	HI_ERR_VDA_NOT_CONFIG	The system or VDA channel is not configured.
0xA0098008	HI_ERR_VDA_NOT_SUPPORT	The parameter or function is not supported.
0xA0098009	HI_ERR_VDA_NOT_PERM	The operation, for example, attempting to modify the value of a static parameter, is forbidden.
0xA009800C	HI_ERR_VDA_NOMEM	The memory fails to be allocated due to some causes such as insufficient system memory.
0xA009800D	HI_ERR_VDA_NOBUF	The buffer fails to be allocated due to some causes such as oversize of the data buffer applied for.
0xA009800E	HI_ERR_VDA_BUF_EMPTY	The buffer is empty.
0xA009800F	HI_ERR_VDA_BUF_FULL	The buffer is full.
0xA0098010	HI_ERR_VDA_SYS_NOTREADY	The system is not initialized or the corresponding module is not loaded.
0xA0098012	HI_ERR_VDA_BUSY	The system is busy.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Contents

8 Region Management 8-1
8.1 Overview 8-1
8.2 Function Description 8-1
8.2.1 Concepts 8-1
8.2.2 Usage Guidelines 8-4
8.3 API Reference 8-4
8.4 Data Types 8-17
8.5 Error Codes 8-34

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figures

Figure 8-1 Region overlay 8-2

Figure 8-2 Non-16-pixel aligned overlay region 8-27

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Tables

Table 8-1 Modules supported by regions....8-3

Table 8-2 Error codes for region management APIs....8-34

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

8 Region Management

8.1 Overview

To display specific information such as the channel ID and the PTS, you need to overlay OSDs on the video. You may also need to fill color blocks. The OSDs and color blocks for covering videos are called regions. The REGION module manages these regions.

Region management involves creating regions and overlaying regions on the video or covering video. After creating a region, you can call HI_MPI_RGN_AttachToChn to overlay this region to channels such as multiple GROUP channels, multiple VI channels, or multiple GROUP channels and VI channels. The display attributes, including the position, layer, and transparency of each channel, can be different. When the channel is scheduled, OSDs are overlaid on the video. The Hi3531 provides only video overlay regions, and each channel supports a maximum of eight overlay regions. The regions are overlaid based on the region level. The region of low level is overlaid with the region of high level. Regions can be overlaid only in the GROUP channel.

Currently, the Hi35xx allows you to overlay regions on a GROUP channel, fill Cover regions and CoverEx regions, and overlay Overlay regions on a VI channel. During the overlay operation, the sequence of displaying regions are determined based on the region level. The regions at low layers are overlaid. Due to the restrictions of the Hi3531 and the Hi3532, the types and number of regions to be created vary depending on modules. For example, a maximum of eight overlay regions can be created for each GROUP channel; a maximum of four Cover regions, 16 CoverEx regions, and 16 OverlayEx regions can be created for each VI channel.

8.2 Function Description

8.2.1 Concepts

- Overlay region

  The video overlay region is used to load bitmaps and update the background color in the GROUP channels.
- Cover

  The cover region enables VI hardware to fill VI channels with pure color blocks.
- CoverEx

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

It is the extended video cover region for short and it is used to fill bound channels with pure color blocks. When cover regions are insufficient, the CoverEx regions can be used.

- OverlayEx

  It is the extended video overlay region for short. It is used to load bitmaps and refresh the background color in bound channels.

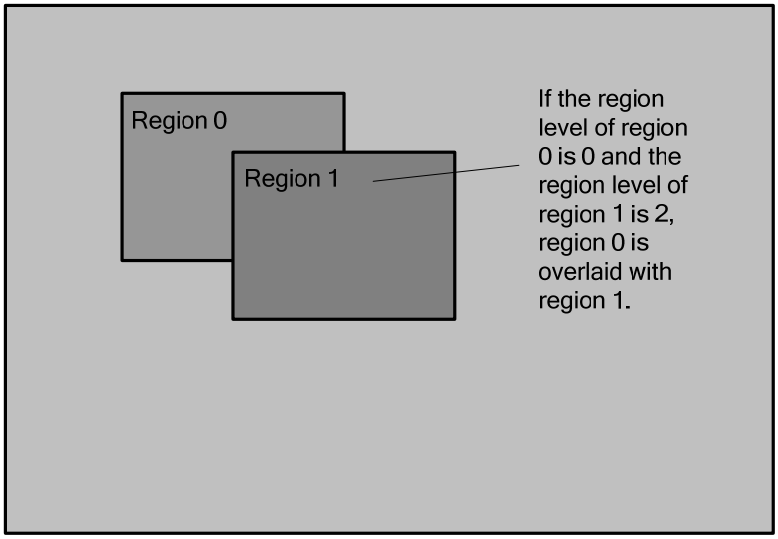
CAUTION

For the Hi3520D/Hi3515A/Hi3515C, note the settings of the g_u32TdeTmpBufW and g_u32TdeTmpBufH parameters of the TDE module if the overlayEx is used. For details, see the *TDE API Reference*.

- Region level

  The region level indicates the overlay level. The higher the region level is, the higher the region display level is. When regions are overlaid, the region with low level is overlaid with the region with high level. If the regions with the same level are overlaid, the existing region is overlaid with the new region.

Figure 8-1 Region overlay



- Bitmap filling (valid for overlay and OverlayEx)

  Bitmap filling indicates that the memory value of a bitmap is filled in the region memory from the upper left corner of the region. When the bitmap size is smaller than the region size, a part of memory is filled, and the remaining part is retained. When the sizes of the bitmap and region are the same, the region memory is filled completely. When the bitmap size is greater than the region size, the region memory is filled with the information of the bitmap size.

- Region attribute

  After creating a region, you must set its attributes, including public resource information. For example, the overlay attribute includes the pixel format, region size, and background color.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Channel display attribute (RGN_CHN_ATTR_S)

  The channel display attributes indicate the region display features in a channel. For example, the overlay channel display attributes include the display position, level, foreground alpha, background alpha, and quantizer parameter (QP) information required for encoding. When the region display attribute parameter bShow is set to TRUE, the region is displayed in the channel; otherwise, the region is hidden in the channel.

- OSD color inversion (valid only for overlay)

  The OSD overlaid on the video may become blurry when the video background changes. The OSD color inversion function can adapt to the change of the video background and automatically adjust the OSD color for ensuring clear OSD. The principles are as follows: When the luminance of the video background is greater than or less than the configured luminance threshold, the OSD color is inverted. The luminance is related to the invert color trigger mode. When the luminance of the video background is above the luminance threshold, the color inversion trigger mode is MORETHAN_LUM_THRESH. When the luminance of the video background is below the luminance threshold, the color inversion trigger mode is LESSTHAN_LUM_THRESH.

- Supported modules

  Table 8-1 describes the modules supported by regions. You need to bind regions to channels by following Table 8-1.

Table 8-1 Modules supported by regions

Module	Region	Region Supported	Value Range of the Device ID	Value Range of the Channel ID
HI_ID_VIU	OVERLAY_RGN	No	None	None
	COVER_RGN	Yes	0	[0, VIU_MAX_CHN_NUM – 1]
	COVEREX_RGN	Yes	0	[0, VIU_MAX_CHN_NUM – 1]
	OVERLAYEX_RGN	Yes	0	[0, VIU_MAX_CHN_NUM – 1]
HI_ID_GROUP	OVERLAY_RGN	Yes	[0, VENC_MAX_GRP_NUM – 1]	0
	COVER_RGN	No	None	None
	COVEREX_RGN	No	None	None
	OVERLAYEX_RGN	No	None	None
HI_ID_PCIV	OVERLAY_RGN	No	None	None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Module	Region	Region Supported	Value Range of the Device ID	Value Range of the Channel ID
	COVER_RGN	No	None	None
	COVEREX_RGN	No	None	None
	OVERLAYEX_RGN	Yes	0	[0, PCIV_MAX_CHN_NUM – 1]

NOTE

The Hi3518/Hi3516C does not support OVERLAYEX_RGN.

8.2.2 Usage Guidelines

To create and use a region, perform the following operations:

- Set region attributes and create a region.
- Set the display attributes of a channel such as a GROUP channel, and attach the region to the channel.

To control region attributes and channel display attributes, perform the following operations:

- Obtain region attributes by calling HI_MPI_RGN_GetAttr, and set region attributes by calling HI_MPI_RGN_SetAttr.
- Set the bitmap information about a region by calling HI_MPI_RGN_SetBitMap. This MPI is applicable only to the overlay operation.
- Obtain the channel display attributes of a channel such as a GROUP channel by calling HI_MPI_RGN_GetDisplayAttr, and set channel display attributes by calling HI_MPI_RGN_SetDisplayAttr.
- (Optional) Detach the region from the channel, and destroy the region.

For details, see related samples.

8.3 API Reference

The REGION module manages regions, including creating and destroying regions, obtaining and setting region attributes, and obtaining and setting channel display attributes of regions.

The module provides the following MPIs:

- HI_MPI_RGN_Create: Creates a region.
- HI_MPI_RGN_Destroy: Destroys a region.
- HI_MPI_RGN_GetAttr: Obtains region attributes.
- HI_MPI_RGN_SetAttr: Sets region attributes.
- HI_MPI_RGN_SetBitMap: Sets a region bitmap.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_RGN_SetAttachField: Sets the identifier of a frame/field to be overlaid with a region.
- HI_MPI_RGN_GetAttachField: Obtains the identifier of a frame/field to be overlaid with a region.
- HI_MPI_RGN_AttachToChn: Overlays a channel with a region.
- HI_MPI_RGN_DetachFrmChn: Detaches a region from a channel.
- HI_MPI_RGN_SetDisplayAttr: Sets the channel display attributes of a region.
- HI_MPI_RGN_GetDisplayAttr: Obtains the channel display attributes of a region.

HI_MPI_RGN_Create

[Description]

Creates a region.

[Syntax]

```
HI_S32 HI_MPI_RGN_Create(RGN_HANDLE Handle,const RGN_ATTR_S *pstRegion);
```

[Parameter]

Parameter	Description	Input/Output
Handle	Region handle ID. The handle ID must be unique. Value range: [0, RGN_HANDLE_MAX)	Input
pstRegion	Pointer to region attributes.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 8.5 "Error Codes."

[Requirement]

- Header files: hi_comm_region.h, mpi_region.h
- Library file: libmpi.a

[Note]

- The handle is specified by users and is equivalent to the ID.
- A region handle ID can be used once only.
- Region attributes must be valid. For details about restrictions, see the description of RGN_ATTR_S.
- The region attribute pointer cannot be null.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Before creating a Cover or CoverEx region, you need only to specify the region type. Other attributes such as the region position and layer are specified when you call HI_MPI_RGN_AttachToChn.

[Example]

```
RGN_HANDLE Handle;
RGN_ATTR_S stRgnAttr;
    HI_S32 s32Ret = HI_SUCCESS;

    Handle = 0;

    stRgnAttr.enType = OVERLAY_RGN;
    stRgnAttr.unAttr.stOverlay.enPixelFmt = PIXEL_FORMAT_RGB_1555;
    stRgnAttr.unAttr.stOverlay.stSize.u32Width  = 16;
     stRgnAttr.unAttr.stOverlay.stSize.u32Height = 16;
    stRgnAttr.unAttr.stOverlay.u32BgColor = 0x0000ffff;

s32Ret = HI_MPI_RGN_Create(Handle,&stRgnAttr);
if(s32Ret != HI_SUCCESS)
{
       return s32Ret;
}
    s32Ret = HI_MPI_RGN_GetAttr(Handle,&stRgnAttr);
    if(s32Ret != HI_SUCCESS)
    {
       return s32Ret;
    }

    stRgnAttr.unAttr.stOverlay.u32BgColor = 0x0000cccc;
    s32Ret = HI_MPI_RGN_SetAttr(Handle,&stRgnAttr);
    if(s32Ret != HI_SUCCESS)
    {
       return s32Ret;
    }

    s32Ret = HI_MPI_RGN_Destroy(Handle);
    if(s32Ret != HI_SUCCESS)
    {
       return s32Ret;
    }
```

[See Also]

- HI_MPI_RGN_Destroy
- HI_MPI_RGN_GetAttr

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_RGN_SetAttr

HI_MPI_RGN_Destroy

[Description]

Destroys a region.

[Syntax]

```
HI_S32 HI_MPI_RGN_Destroy(RGN_HANDLE Handle);
```

[Parameter]

Parameter	Description	Input/Output
Handle	Region handle ID. Value range: [0, RGN_HANDLE_MAX)	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 8.5 "Error Codes."

[Requirement]

- Header files: hi_comm_region.h, mpi_region.h
- Library file: libmpi.a

[Note]

Create a region before calling this MPI.

[Example]

For details, see the sample of HI_MPI_RGN_Create.

[See Also]

None

HI_MPI_RGN_GetAttr

[Description]

Obtains region attributes.

[Syntax]

```
HI_S32 HI_MPI_RGN_GetAttr(RGN_HANDLE Handle,RGN_ATTR_S *pstRegion);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
Handle	Region handle ID. Value range: [0, RGN_HANDLE_MAX)	Input
pstRegion	Pointer to region attributes.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 8.5 "Error Codes."

[Requirement]

- Header files: hi_comm_region.h, mpi_region.h
- Library file: libmpi.a

[Note]

- Create a region before calling this MPI.
- The region attribute pointer cannot be null.

[Example]

For details, see the sample of HI_MPI_RGN_Create.

[See Also]

None

HI_MPI_RGN_SetAttr

[Description]

Sets region attributes.

[Syntax]

```
HI_S32 HI_MPI_RGN_SetAttr(RGN_HANDLE Handle,const RGN_ATTR_S *pstRegion);
```

[Parameter]

Parameter	Description	Input/Output
Handle	Region handle ID. Value range: [0, RGN_HANDLE_MAX)	Input
pstRegion	Pointer to region attributes.	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 8.5 "Error Codes."

[Requirement]

- Header files: hi_comm_region.h, mpi_region.h
- Library file: libmpi.a

[Note]

- Create a region before calling this MPI.
- Only the dynamic region attributes can be changed. For details about dynamic attributes of all types of regions, see the related data types.
- The region attribute pointer cannot be null.

[Example]

For details, see the sample of HI_MPI_RGN_Create.

[See Also]

None

HI_MPI_RGN_SetBitMap

[Description]

Set a region bitmap, that is, fills in the region bitmap.

[Syntax]

```
HI_S32 HI_MPI_RGN_SetBitMap(RGN_HANDLE Handle,const BITMAP_S *pstBitmap);
```

[Parameter]

Parameter	Description	Input/Output
Handle	Region handle ID. Value range: [0, RGN_HANDLE_MAX)	Input
pstBitmap	Pointer to bitmap attributes.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 8.5 "Error Codes."

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_region.h, mpi_region.h
- Library file: libmpi.a

[Note]

- Create a region before calling this MPI.
- The sizes of the bitmap and region can be different.
- The bitmap is loaded from the (0, 0) of the region. If the bitmap size is greater than the region size, the bitmap is automatically cropped to the region size.
- The pixel formats of the bitmap and region must be the same.
- The bitmap attribute pointer cannot be null.
- This MPI can be called repeatedly.
- This MPI is valid only for the overlay region.

[Example]

```
RGN_HANDLE Handle;
BITMAP_S stBitmap;
HI_S32 s32Ret = HI_SUCCESS;

stBitmap.enPixelFormat = PIXEL_FORMAT_RGB_1555;
stBitmap.u32Width  = 4;
stBitmap.u32Height = 4;

stBitmap.pData = malloc(2 * stBitmap.u32Width * stBitmap.u32Height);
if(HI_NULL == stBitmap.pData)
{
    return HI_FAILURE;
}

memset(stBitmap.pData,0xcc,u32Size);

s32Ret = HI_MPI_RGN_SetBitMap(Handle,&stBitmap);
if(s32Ret != HI_SUCCESS)
{
    free(stBitmap.pData);
    return s32Ret;
}

free(stBitmap.pData);
```

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_RGN_SetAttachField

[Description]

Sets the identifier of a frame/field to be overlaid with a region.

[Syntax]

```
HI_S32 HI_MPI_RGN_SetAttachField(RGN_HANDLE Handle, RGN_ATTACH_FIELD_E
enAttachField);
```

[Parameter]

Parameter	Description	Input/Output
Handle	Region handle ID. Value range: [0, RGN_HANDLE_MAX)	Input
enAttachField	Identifier of a frame/field to be overlaid with a region.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 8.5 "Error Codes."

[Requirement]

- Header files: hi_comm_region.h, mpi_region.h
- Library file: libmpi.a

[Note]

- Call HI_MPI_RGN_SetAttachField before calling HI_MPI_RGN_AttachToChn.
- This MPI supports only the COVEREX_RGN or OVERLAYEX_RGN region.
- When a region is overlaid by field, the coordinates, width, and height of the region are configured based on the entire frame.
- The function of overlaying a region by field is valid only for interlaced 2-field pictures from the VIU. If you overlay a frame picture with a region or overlay a region of other modules, calling this MPI has no effect.
- When a top or bottom field is overlaid with a region, the vertical coordinate and height of the region must be 4-pixel-aligned. When a frame is overlaid with a region, the vertical coordinate and height of the region must be 2-pixel-aligned.

[Example]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_RGN_GetAttachField

[Description]

Obtains the identifier of a frame/field to be overlaid with a region.

[Syntax]

```
HI_S32 HI_MPI_RGN_GetAttachField(RGN_HANDLE Handle, RGN_ATTACH_FIELD_E
*penAttachField);
```

[Parameter]

Parameter	Description	Input/Output
Handle	Region handle ID. Value range: [0, RGN_HANDLE_MAX)	Input
penAttachField	Pointer to the identifier of a frame/field to be overlaid with a region.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 8.5 "Error Codes."

[Requirement]

- Header files: hi_comm_region.h, mpi_region.h
- Library file: libmpi.a

[Note]

- Create a region before calling this MPI.
- This MPI supports only the COVEREX_RGN or OVERLAYEX_RGN region.

[Example]

None

[See Also]

None

HI_MPI_RGN_AttachToChn

[Description]

Overlays a channel with a region.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
HI_MPI_RGN_AttachToChn(RGN_HANDLE Handle,const MPP_CHN_S *pstChn,const
RGN_CHN_ATTR_S *pstChnAttr);
```

[Parameter]

Parameter	Description	Input/Output
Handle	Region handle ID. Value range: [0, RGN_HANDLE_MAX)	Input
pstChn	Pointer to the channel structure.	Input
pstChnAttr	Pointer to the channel display attributes of a region.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 8.5 "Error Codes."

[Requirement]

- Header files: hi_comm_region.h, mpi_region.h
- Library file: libmpi.a

[Note]

- Create a region before calling this MPI.
- The pointer that points to the channel structure cannot be null.
- The pointer that points to channel display attributes cannot be null.
- This MPI can be called repeatedly, but cannot be used to modify attributes.
- For the overlay area, pay attention to QP information. For details, see the description of OVERLAY_QP_INFO_S.

[Example]

None

[See Also]

None

HI_MPI_RGN_DetachFrmChn

[Description]

Detaches a region from a channel.

[Syntax]

```
HI_MPI_RGN_DetachFromChn(RGN_HANDLE Handle,const MPP_CHN_S *pstChn);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
Handle	Region handle ID. Value range: [0, RGN_HANDLE_MAX)	Input
pstChn	Pointer to the channel structure.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 8.5 "Error Codes."

[Requirement]

- Header files: hi_comm_region.h, mpi_region.h
- Library file: libmpi.a

[Note]

- Create a region before calling this MPI.
- The pointer that points to the channel structure cannot be null.
- This MPI can be called repeatedly.

[Example]

None

[See Also]

None

HI_MPI_RGN_SetDisplayAttr

[Description]

Sets the channel display attributes of a region.

[Syntax]

```
HI_S32 HI_MPI_RGN_SetDisplayAttr(RGN_HANDLE Handle,const MPP_CHN_S
*pstChn,const RGN_CHN_ATTR_S *pstChnAttr);
```

[Parameter]

Parameter	Description	Input/Output
Handle	Region handle ID. Value range: [0, RGN_HANDLE_MAX)	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
pstChn	Pointer to the channel structure.	Input
pstChnAttr	Pointer to the channel display attributes of a region.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 8.5 "Error Codes."

[Requirement]

- Header files: hi_comm_region.h, mpi_region.h
- Library file: libmpi.a

[Note]

- Create a region before calling this MPI.
- You are advised to obtain attributes before setting attributes.
- The pointer that points to the channel structure cannot be null.
- The pointer that points to channel display attributes cannot be null.
- Overlay the channel with a region before calling this MPI.
- Only dynamic attributes can be modified. For details, see the description of RGN_CHN_ATTR_S.

[Example]

```
HI_S32 s32Ret = HI_SUCCESS;
    RGN_HANDLE Handle;
    MPP_CHN_S stChn;
     RGN_ATTR_S stRgnAttr;
    RGN_CHN_ATTR_S stChnAttr;

    Handle = 0;

    stChn.enModId = HI_ID_GROUP;
    stChn.s32DevId = 0;
    stChn.s32ChnId = 0;

    s32Ret =  HI_MPI_RGN_GetDisplayAttr(Handle,&stChn,&stChnAttr);
    if(s32Ret != HI_SUCCESS)
    {
        NOT_PASS(s32Ret);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    goto END;
}

stChnAttr.bShow = HI_TRUE;
stChnAttr.enType = OVERLAY_RGN;
stChnAttr.unChnAttr.stOverlayChn.stPoint.s32X = 4;
stChnAttr.unChnAttr.stOverlayChn.stPoint.s32Y = 4;
stChnAttr.unChnAttr.stOverlayChn.u32BgAlpha = 128;
stChnAttr.unChnAttr.stOverlayChn.u32FgAlpha = 128;
stChnAttr.unChnAttr.stOverlayChn.u32Layer = 0;

stChnAttr.unChnAttr.stOverlayChn.stQpInfo.bAbsQp = HI_FALSE;
stChnAttr.unChnAttr.stOverlayChn.stQpInfo.s32Qp  = 0;

s32Ret = HI_MPI_RGN_SetDisplayAttr(Handle,&stChn,&stChnAttr);
if(s32Ret != HI_SUCCESS)
{
    return s32Ret;
}
```

[See Also]

None

HI_MPI_RGN_GetDisplayAttr

[Description]

Obtains the channel display attributes of a region.

[Syntax]

```
HI_S32 HI_MPI_RGN_GetDisplayAttr(RGN_HANDLE Handle,const MPP_CHN_S
*pstChn,RGN_CHN_ATTR_S *pstChnAttr);
```

[Parameter]

Parameter	Description	Input/Output
Handle	Region handle ID. Value range: [0, RGN_HANDLE_MAX)	Input
pstChn	Pointer to the channel structure.	Input
pstChnAttr	Pointer to the channel display attributes of a region.	Output

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success.
Other values	Failure. For details, see section 8.5 "Error Codes."

[Requirement]

- Header files: hi_comm_region.h, mpi_region.h
- Library file: libmpi.a

[Note]

- Create a region before calling this MPI.
- The pointer that points to the channel structure cannot be null.
- The pointer that points to channel display attributes cannot be null.

[Example]

For details, see the sample of HI_MPI_RGN_SetDisplayAttr.

[See Also]

None

8.4 Data Types

The data types related to region management are as follows:

- RGN_HANDLE_MAX: Defines the maximum number of region handles.
- OVERLAY_MAX_NUM: Defines the maximum number of overlay regions of a GROUP channel
- COVER_MAX_NUM: Defines the maximum number of Cover regions of a channel.
- COVEREX_MAX_NUM: Defines the maximum number of extended Cover regions of a channel.
- OVERLAYEX_MAX_NUM: Defines the maximum number of extended overlay regions of a channel.
- RGN_HANDLE: Defines the region handle.
- RGN_TYPE_E: Defines the region type.
- INVERT_COLOR_MODE_E: Defines the OSD color inversion trigger mode.
- OVERLAY_QP_INFO_S: Defines the QP attributes of an overlay region.
- OVERLAY_INVERT_COLOR_S: Defines the attributes related to OSD color inversion.
- RGN_ATTACH_FIELD_E: Defines the type of a frame/field to be overlaid with a region.
- OVERLAY_ATTR_S: Defines the overlay region attribute structure of a GROUP channel.
- OVERLAY_CHN_ATTR_S: Defines the channel display attributes of an extended overlay region of a GROUP channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- COVER_CHN_ATTR_S: Defines the channel display attributes of a Cover region of a VI channel.
- COVEREX_CHN_ATTR_S: Defines the channel display attributes of an extended Cover region of a VI channel.
- OVERLAYEX_ATTR_S: Defines the attribute structure of an extended overlay region of a VI channel.
- OVERLAYEX_CHN_ATTR_S: Defines the channel display attribute of an overlay region of a VI channel.
- RGN_ATTR_U: Defines the region attribute union.
- RGN_CHN_ATTR_U: Defines the display attributes of a region channel.
- RGN_ATTR_S: Defines the region attribute structure.
- RGN_CHN_ATTR_S: Defines the structure of the channel display attributes of a region.

RGN_HANDLE_MAX

[Description]

Defines the maximum number of region handles.

[Syntax]

```
#define RGN_HANDLE_MAX  1024
```

[Member]

None

[Note]

None

[See Also]

None

OVERLAY_MAX_NUM

[Description]

Defines the maximum number of overlay regions of a GROUP channel.

[Syntax]

```
#define OVERLAY_MAX_NUM        8
```

[Member]

None

[Note]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

COVER_MAX_NUM

[Description]

Defines the maximum number of Cover regions of a channel.

[Syntax]

```
#define COVER_MAX_NUM        4
```

[Member]

None

[Note]

None

[See Also]

None

COVEREX_MAX_NUM

[Description]

Defines the maximum number of extended Cover regions of a channel.

[Syntax]

```
#define COVEREX_MAX_NUM       16
```

[Member]

None.

[Note]

None.

[See Also]

None.

OVERLAYEX_MAX_NUM

[Description]

Defines the maximum number of extended overlay regions of a channel.

[Syntax]

For Hi3531/Hi3532/Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C:

```
#define OVERLAYEX_MAX_NUM        16
```

For Hi3518/Hi3516C:

```
#define OVERLAYEX_MAX_NUM        0
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None.

[Note]

None.

[See Also]

None.

RGN_HANDLE

[Description]

Defines the region handle.

[Syntax]

```
typedef HI_U32 RGN_HANDLE;
```

[Member]

Member	Description
RGN_HANDLE	Region handle.

[Note]

None

[See Also]

None

RGN_TYPE_E

[Description]

Defines the region type.

[Syntax]

```
typedef enum hiRGN_TYPE_E
{
    OVERLAY_RGN = 0,
    COVER_RGN,
    COVEREX_RGN,
    OVERLAYEX_RGN,
    RGN_BUTT
} RGN_TYPE_E;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
OVERLAY_RGN	Video overlay region of the GROUP channel.
COVER_RGN	Video Cover region of the VI channel.
COVEREX_RGN	Extended video Cover region.
OVERLAYEX_RGN	Extended video overlay region.

[Note]

None

[See Also]

None

INVERT_COLOR_MODE_E

[Description]

Defines the OSD color inversion trigger mode.

[Syntax]

```
typedef enum hiINVERT_COLOR_MODE_E
{
    WHITE_TO_BLACK = 0,
    BLACK_TO_WHITE,
    INVERT_COLOR_BUTT
}INVERT_COLOR_MODE_E;
```

[Member]

Member	Description
LESSTHAN_LUM_THRESH	Color inversion is triggered when the average video background luminance is less than the luminance threshold.
MORETHAN_LUM_THRESH	Color inversion is triggered when the average video background luminance is greater than the luminance threshold.

[Note]

This configuration is valid when color inversion is enabled.

[See Also]

None.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

OVERLAY_QP_INFO_S

[Description]

Defines the QP attributes of an overlay region.

[Syntax]

```
typedef struct hiOVERLAY_QP_INFO_S
{
    HI_BOOL bAbsQp;
    HI_S32 s32Qp;
}OVERLAY_QP_INFO_S;
```

[Member]

Member	Description
bAbsQp	Absolute QP or not.
s32Qp	QP value. When bAbsQp is HI_TRUE, the QP value range from 0 to 51. When bAbsQp is HI_FALSE, the QP value range from –51 to +51.

[Note]

This data type is used for controlling the bit rate during encoding. If you are not familiar with the data type, you are advised to set all parameters to 0.

[See Also]

None

OVERLAY_INVERT_COLOR_S

[Description]

Defines the attributes related to OSD color inversion.

[Syntax]

```
typedef struct hiOVERLAY_INVERT_COLOR_S
{
    SIZE_S stInvColArea;
    HI_U32 u32LumThresh;
    INVERT_COLOR_MODE_E enChgMod;
    HI_BOOL bInvColEn;
}OVERLAY_INVERT_COLOR_S;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
stInvColArea	Basic color inversion unit. Value range: Height: [16, 64], 16-pixel alignment Width: [16, 64], 16-pixel alignment
u32LumThresh	Luminance threshold. Value range: [0, 255]
enChgMod	OSD color inversion trigger mode.
bInvColEn	OSD color inversion enable. HI_TRUE: enabled HI_TRUE: disabled

[Note]

- **stInvColArea**, **u32LumThresh**, and **enChgMod** are public attributes. They are valid for the OSD areas whose color inversion function is enabled. If the public attributes are set to different values each time, the attributes are subject to the latest configurations.
- The **bInvColEn** parameter controls whether to enable color inversion of an OSD area. The color inversion function of each OSD area can be controlled separately.
- When the color inversion function of an OSD area is enabled, the start position and width and height of the OSD area must be 16-pixel-aligned. In addition, you are advised to set the start position, width, and height of the OSD area to multiples of the basic color inversion unit. Otherwise, only the color of some parts of a basic unit is inverted.

[See Also]

None.

RGN_ATTACH_FIELD_E

[Description]

Defines the type of a frame/field to be overlaid with a region.

[Syntax]

```
typedef enum hiRGN_ATTACH_FIELD_E
{
    RGN_ATTACH_FIELD_FRAME = 0,           /*Entire frame*/
    RGN_ATTACH_FIELD_TOP,                 /*Top field*/
    RGN_ATTACH_FIELD_BOTTOM,              /*Bottom field*/
    RGN_ATTACH_FIELD_BUTT
} RGN_ATTACH_FIELD_E;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
RGN_ATTACH_FIELD_FRAME	The entire frame is overlaid with a region.
RGN_ATTACH_FIELD_TOP	The top field is overlaid with a region.
RGN_ATTACH_FIELD_BOTTOM	The bottom field is overlaid with a region.

[Note]

RGN_ATTACH_FIELD_TOP or **RGN_ATTACH_FIELD_BOTTOM** is valid only for interlaced 2-field pictures.

[See Also]

None

OVERLAY_ATTR_S

[Description]

Defines the overlay region attribute structure of a GROUP channel.

[Syntax]

```
typedef struct hiOVERLAY_COMM_ATTR_S
{
    PIXEL_FORMAT_E enPixelFmt;
   HI_U32 u32BgColor;
   SIZE_S stSize;
}OVERLAY_ATTR_S;
```

[Member]

Member	Description
enPixelFmt	Pixel format. Value: PIXEL_FORMAT_RGB_1555 or PIXEL_FORMAT_RGB_4444 Static attribute.
u32BgColor	Region background color. Value range: none Dynamic attribute.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
stSize	Region height and region width. Value range: When the OSD invert color is disabled: Width: [4, 4096], 2-pixel-aligned Height: [4, 4096], 2-pixel-aligned When the OSD invert color is enabled: Width: [4, 4096], 16-pixel-aligned Height: [4, 4096], 16-pixel-aligned Static attribute.

[Note]

- When setting the width and height of the area, note whether the OSD invert color function is enabled.
- When the color inversion function of an OSD area is enabled, the start position and width and height of the OSD area must be 16-pixel-aligned. In addition, you are advised to set the start position, width, and height of the OSD area to multiples of the basic color inversion unit. Otherwise, only the color of some parts of a basic unit is inverted.

[See Also]

None

OVERLAY_CHN_ATTR_S

[Description]

Defines the channel display attributes of an overlay region of a GROUP channel.

[Syntax]

```
typedef struct hiOVERLAY_CHN_ATTR_S
{
    POINT_S stPoint;
    HI_U32 u32FgAlpha;
    HI_U32 u32BgAlpha;
    HI_U32 u32Layer;
    RGN_HANDLE stQpInfo;
    OVERLAY_INVERT_COLOR_S stInvertColor;
}OVERLAY_CHN_ATTR_S;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
stPoint	Region position. The origin is the upper left point of the region. Value range: When OSD color inversion is disabled: Horizontal coordinate: [0, 4096], 4-pixel alignment Vertical coordinate: [0, 4096], 4-pixel alignment When OSD color inversion is enabled: Horizontal coordinate: [0, 4096], 16-pixel alignment Vertical coordinate: [0, 4096], 16-pixel alignment Dynamic attribute.
u32FgAlpha	Transparency of the pixel whose alpha bit is 1. u32FgAlpha is called foreground alpha. Value range: [0, 128] The smaller the value is, the more transparent the pixel is. Dynamic attribute.
u32BgAlpha	Transparency of the pixel whose alpha bit is 0. u32BgAlpha is called background alpha. Value range: [0, 128]. The smaller the value is, the more transparent the pixel is. Dynamic attribute.
u32Layer	Region level. Value range: [0, 7] The larger the value is, the higher the region level is. Dynamic attribute.
stQpInfo	QP value when the region is encoded. For details about the value range, see the description of OVERLAY_QP_INFO_S. Dynamic attribute.
stInvertColor	Region invert color configuration information. For details about the value range, see the description of OVERLAY_INVERT_COLOR_S. Dynamic attribute.

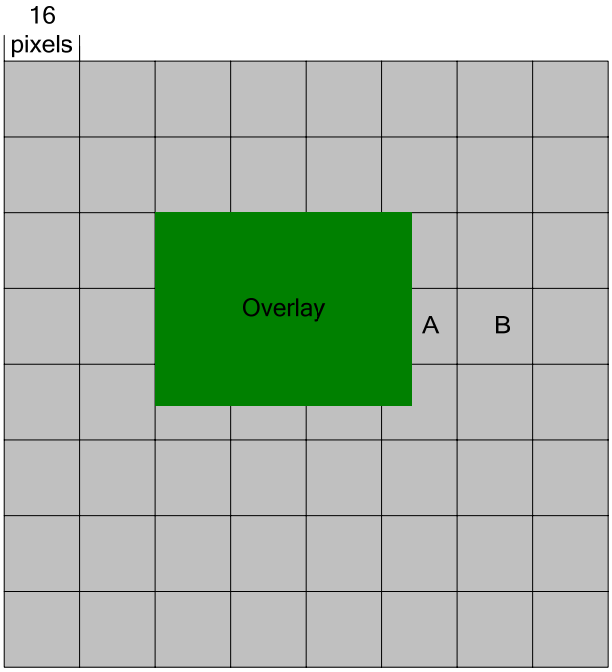
[Note]

- You are advised to set the start position, width, and height of an overlay region to 16-pixel aligned. As shown in Figure 8-2, the width and height of the overlay region are not 16-pixel aligned. As a result, a macroblock may have two QPs. For example, macroblock A has the overlay region QP and the calculated QP. However, only one QP is allowed for each macroblock. In this case, the encoder selects the region QP for macroblock A. If the

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

difference between the QPs of macroblock A and macroblock B is great, the picture quality of the macroblocks around the overlay region is inconsistent.

Figure 8-2 Non-16-pixel aligned overlay region



- When the region memory information is in PIXEL_FORMAT_RGB_1555 format, the Hi3531 expands the alpha value. When the alpha bit is 1, the Hi3531 performs transparency overlay by setting u32FgAlpha; when the alpha bit is 0, the Hi3531 performs transparency overlay by setting u32BgAlpha.
- The value 0 indicates 100% transparency, and the value 128 indicates opacity.
- Check whether OSD color inversion is enabled when setting the start position of an OSD area.
- When the color inversion function of an OSD area is enabled, the start position and width and height of the OSD area must be 16-pixel-aligned. In addition, you are advised to set the start position, width, and height of the OSD area to multiples of the basic color inversion unit. Otherwise, only the color of some parts of a basic unit is inverted.

[See Also]

None

COVER_CHN_ATTR_S

[Description]

Defines the channel display attributes of a Cover region of a VI channel.

[Syntax]

```
typedef struct hiCOVER_CHN_ATTR_S
{
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    RECT_S stRect;
    HI_U32 u32Color;
    HI_U32 u32Layer;
}COVER_CHN_ATTR_S;
```

[Member]

Member	Description
stRect	Region position, width, and height. Value range of the position: Horizontal coordinate: [0, 4094), 2-pixel alignment Vertical coordinate: [0, 4636), 4-pixel alignment Value range of the width and height: Width: [2, 4094), 2-pixel-aligned Height: [4, 4094), 4-pixel-aligned Dynamic attribute.
u32Color	Region color. Dynamic attribute.
u32Layer	Region level. Value range: [0, 3] The larger the value, the higher the region level. Dynamic attribute.

[Note]

The region color data format is PIXEL_FORMAT_RGB_888.

[See Also]

None

COVEREX_CHN_ATTR_S

[Description]

Defines the channel display attributes of an extended Cover region.

[Syntax]

```
typedef struct hiCOVEREX_CHN_ATTR_S
{
    RECT_S stRect;
    HI_U32 u32Color;
    HI_U32 u32Layer;
}COVEREX_CHN_ATTR_S;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
stRect	Region position, width and height. Value range of the position: Horizontal coordinate: [0, 4096], 2-pixel-aligned Vertical coordinate: [0, 4096], 2-pixel-aligned Value range of the width and height: Width: [2, 4096], 2-pixel-aligned Height: [2, 4096], 2-pixel-aligned Dynamic attribute.
u32Color	Region color. Value range: none Dynamic attribute.
u32Layer	Region level. Value range: [0, 15] The larger the value is, the higher the region level is. Dynamic attribute.

[Note]

- The region color data format is PIXEL_FORMAT_RGB_888.
- When the VI channel captures pictures with different resolutions, stRect is configured based on the Crop value of the VI channel. In the SDK, the cover region is adapted based on the current picture size to enable the final cover region to be in the field of vision. In this case, you are advised to set stRect based on the 4-pixel-alignment rule, ensuring that stRect is 2-pixel-aligned after adaptation. When the VI channel does not capture pictures with different resolutions, stRect is configured based on the current picture size.
- When videos are previewed on the CVBS device in single-picture mode, you are advised to stop capturing pictures with different resolutions for the current VI channel but capture D1 streams in real time if you want to overlay a CoverEx region. Otherwise, the CoverEx region flickers during previewing due to scaling. However, you cannot find that the CoverEx region flickers in multi-picture mode.
- Typically, pictures are transmitted to the VENC module for CIF encoding after being processed by the VPSS. If you overlay a CoverEx region when VI channel captures pictures with different resolutions and display pictures based on the original stream size, you cannot find that the CoverEx region flickers. However, if you enlarge the pictures, you may find that the CoverEx region flickers. Therefore, overlay a CoverEx region with cautions when the VI channel captures pictures with different resolutions.

[See Also]

None.

OVERLAYEX_ATTR_S

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Defines the attribute structure of an extended overlay region of a VI channel.

[Syntax]

```
typedef struct hiOVERLAYEX_COMM_ATTR_S
{
    PIXEL_FORMAT_E enPixelFmt;
   HI_U32 u32BgColor;
   SIZE_S stSize;
}OVERLAYEX_ATTR_S;
```

[Member]

Member	Description
enPixelFmt	Pixel format. Value: PIXEL_FORMAT_RGB_1555 Static attribute.
u32BgColor	Region background color. Value range: none Dynamic attribute.
stSize	Width and height of the region. Value: Width: [16, 1920], 2-pixel-aligned Height: [16, 1080], 2-pixel-aligned width x height ≤ 480000 Static attribute.

[Note]

None.

[See Also]

None.

OVERLAYEX_CHN_ATTR_S

[Description]

Defines the channel display attribute of an overlay region of a GROUP channel.

[Syntax]

```
typedef struct hiOVERLAYEX_CHN_ATTR_S
{
   POINT_S stPoint;
   HI_U32 u32FgAlpha;
   HI_U32 u32BgAlpha;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_U32 u32Layer;
}OVERLAYEX_CHN_ATTR_S;
```

[Member]

Member	Description
stPoint	Region position Value range: Horizontal coordinate: [0, 4096], 2-pixel-aligned Vertical coordinate: [0, 4636], 2-pixel-aligned
u32FgAlpha	Transparency of the pixel whose alpha bit is 1. u32FgAlpha is called foreground alpha. Value range: [0, 255] The smaller the value is, the more transparent the pixel is. Dynamic attribute.
u32BgAlpha	Transparency of the pixel whose alpha bit is 0. u32BgAlpha is called background alpha. Value range: [0, 255] The smaller the value is, the more transparent the pixel is. Dynamic attribute.
u32Layer	Region level. Value range: [0, 15] The larger the value is, the higher the region level is. Dynamic attribute.

[Note]

- When the region memory information is in PIXEL_FORMAT_RGB_1555 format, the Hi3531 and the Hi3532 expand the alpha value. When the alpha bit is 1, the Hi3531 and the Hi3532 overlay the transparency by setting u32FgAlpha; when the alpha bit is 0, the Hi3531 and the Hi3532 overlay the transparency by setting u32BgAlpha.
- The value 0 indicates 100% transparency, and the value 255 indicates opacity.

[See Also]

None.

RGN_ATTR_U

[Description]

Defines the region attribute union.

[Syntax]

```
typedef union hiRGN_ATTR_U
{
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    OVERLAY_ATTR_S      stOverlay;
    OVERLAYEX_ATTR_S    stOverlayEx;
} RGN_ATTR_U;
```

[Member]

Member	Description
stOverlay	Overlay region attributes of the GROUP channel.
stOverlayEx	Extended overlay region attributes.

[Note]

As the Cover region and CoverEx region do not have region attributes, this union does not contain related members.

[See Also]

None

RGN_CHN_ATTR_U

[Description]

Defines the display attributes of a region channel.

[Syntax]

```
typedef union hiRGN_CHN_ATTR_U
{
   OVERLAY_CHN_ATTR_S     stOverlayChn;
   COVER_CHN_ATTR_S       stCoverChn;
   COVEREX_CHN_ATTR_S     stCoverExChn;
   OVERLAYEX_CHN_ATTR_S   stOverlayEXChn;
} RGN_CHN_ATTR_U;
```

[Member]

Member	Description
stOverlayChn	Display attributes of an overlay channel of the GROUP channel.
stCoverChn	Display attributes of a Cover region.
stCoverExChn	Channel display attributes of the extended Cover region
stOverlayExChn	Channel display attributes of the extended overlay region.

[Note]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

None

RGN_ATTR_S

[Description]

Defines the region attribute structure.

[Syntax]

```
typedef struct hiRGN_ATTR_S
{
    RGN_TYPE_E enType;
    RGN_ATTR_U unAttr;
} RGN_ATTR_S;
```

[Member]

Member	Description
enType	Region type.
unAttr	Region attribute.

[Note]

None

[See Also]

None

RGN_CHN_ATTR_S

[Description]

Defines the structure of the channel display attributes of a region.

[Syntax]

```
typedef struct hiRGN_CHN_ATTR_S
{
    HI_BOOL        bShow;
    RGN_TYPE_E     enType;
    RGN_CHN_ATTR_U unChnAttr;
} RGN_CHN_ATTR_S;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
bShow	Whether to display a region. Value: HI_TRUE or HI_FALSE Dynamic attribute.
enType	Region type. Static attribute.
unChnAttr	Channel display attributes of a region.

[Note]

None

[See Also]

None

8.5 Error Codes

Table 8-2 describes the error codes for region management APIs.

Table 8-2 Error codes for region management APIs

Error Code	Macro Definition	Description
0xA0128001	HI_ERR_RGN_INVALID_DEVID	The device ID exceeds the valid range.
0xA0128002	HI_ERR_RGN_INVALID_CHNID	The channel ID is incorrect or the region handle is invalid.
0xA0128003	HI_ERR_RGN_ILLEGAL_PARAM	The parameter value exceeds its valid range.
0xA0128004	HI_ERR_RGN_EXIST	The device, channel, or resource to be created already exists.
0xA0128005	HI_ERR_RGN_UNEXIST	The device, channel, or resource to be used or destroyed does not exist.
0xA0128006	HI_ERR_RGN_NULL_PTR	The pointer is null.
0xA0128007	HI_ERR_RGN_NOT_CONFIG	The module is not configured.
0xA0128008	HI_ERR_RGN_NOT_SUPPORT	The parameter or function is not supported.
0xA0128009	HI_ERR_RGN_NOT_PERM	The operation, for example, attempting to modify the value of a static parameter, is forbidden.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Macro Definition	Description
0xA012800C	HI_ERR_RGN_NOMEM	The memory fails to be allocated due to some causes such as insufficient system memory.
0xA012800D	HI_ERR_RGN_NOBUF	The buffer fails to be allocated due to some causes such as oversize of the data buffer applied for.
0xA012800E	HI_ERR_RGN_BUF_EMPTY	The buffer is empty.
0xA012800F	HI_ERR_RGN_BUF_FULL	The buffer is full.
0xA0128010	HI_ERR_RGN_NOTREADY	The system is not initialized or the corresponding module is not loaded.
0xA0128011	HI_ERR_RGN_BADADDR	The address is invalid.
0xA0128012	HI_ERR_RGN_BUSY	The system is busy.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Contents

9 Audio 9-1
9.1 Overview 9-1
9.2 Functions 9-1
9.2.1 AI and AO 9-1
9.2.2 Audio Encoding and Decoding 9-9
9.2.3 Differences Among Chips 9-11
9.2.4 Built-In Audio CODEC 9-15
9.3 API Reference 9-19
9.3.1 AI 9-19
9.3.2 AO 9-48
9.3.3 AENC 9-67
9.3.4 ADEC 9-77
9.3.5 Built-In Audio CODEC 9-86
9.4 Data Types 9-137
9.4.1 AI and AO 9-137
9.4.2 AENC 9-157
9.4.3 ADEC 9-161
9.4.4 Built-In Audio CODEC 9-166
9.5 Error Codes 9-169

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figures

Figure 9-1 Schematic diagram of audio recording and playing 9-2
Figure 9-2 2/4/8/16-channel multiplexing of I^2S 9-3
Figure 9-3 PCM TX timing 9-5
Figure 9-4 Echo cancellation 9-6
Figure 9-5 Mapping between SIO interfaces and AI/AO devices for the Hi3531 or Hi3532 9-13
Figure 9-6 Mapping between SIO interfaces and AI/AO devices for the Hi3521 or Hi3520A 9-14
Figure 9-7 Mapping between SIO interfaces and AI/AO devices for the Hi3518/Hi3516C 9-14
Figure 9-8 Mapping between AIPs/AOPs and AI/AO devices for the Hi3520D/Hi3515A/Hi3515C 9-15

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Tables

Table 9-1 Maximum number of AI and AO channels supported by SIO interfaces....9-4
Table 9-2 Maximum number of AI and AO channels supported by AIO interfaces....9-4
Table 9-3 Sampling rates corresponding to resampling multiples....9-7
Table 9-4 Audio encoding and decoding protocols....9-9
Table 9-5 Structure of the HiSilicon voice frame....9-11
Table 9-6 Audio interface type for each chip....9-12
Table 9-7 Ranges of AI and AO device IDs....9-15
Table 9-8 Parameters of the ioctl function....9-16
Table 9-9 Error codes for the AI MPIs....9-169
Table 9-10 Error codes for the AO MPIs....9-169
Table 9-11 Error codes for the AENC MPIs....9-170
Table 9-12 Error codes for the ADEC MPIs....9-171

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

9 Audio

9.1 Overview

The audio module consists of the audio input unit (AIU), audio output unit (AOU), audio encoding (AENC) module, and audio decoding (ADEC) module. The AIU and AOU control the audio interfaces to implement the audio input (AI) and audio output (AO) functions. The AENC module and ADEC module encode and decode G711, G726, or ADPCM streams, and record and play LPCM audio files.

9.2 Functions

9.2.1 AI and AO

9.2.1.1 Audio Interfaces, AI Devices, and AO Devices

Audio interfaces are classified into sonic input/output (SIO) interfaces and audio input/output (AIO) interfaces. All audio interfaces are used to connect the chip to the audio CODEC to implement audio recording and playing. SIO interfaces support both inputs and outputs. AIO interfaces include audio input ports (AIPs) and audio output ports (AOPs).

From the aspect of software, the units for implementing abstract input functions of audio interfaces are called AI devices, and the units for implementing abstract output functions of audio interfaces are called AO devices.

Software maps each input or output interface to the AI or AO device based on interface functions. For example, SIO0 supports both audio inputs and outputs, and it maps to AiDev0 as an input unit and AoDev0 as an output unit. AIP1 supports only audio inputs, and maps to AiDev1. AOP2 supports only audio outputs, and maps to AoDev2.

9.2.1.2 Recording and Playing Principles

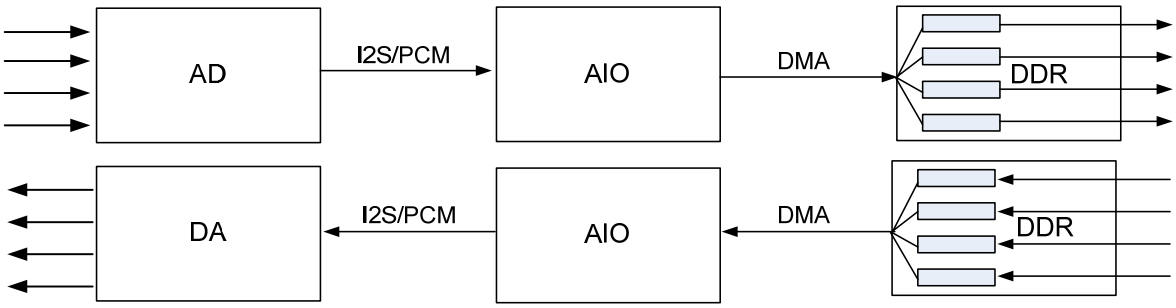
The raw audio signals are analog signals. After processed by the audio CODEC based on the sampling rate and sampling precision, the analog signals are converted into digital signals. The audio CODEC transmits the digital signals to the AI device using the I^2S or PCM timing. The chip transmits the audio data from the AI device to the memory in DMA mode to implement recording.

The principle of playing audio data is the same as that of recording audio data. The chip transmits the audio data from the memory to the AO device in DMA mode. The AO device

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

transmits data to the audio CODEC using the I²S or PCM timing. The audio CODEC converts digital signals into analog signals and outputs the analog signals.

Figure 9-1 Schematic diagram of audio recording and playing



9.2.1.3 Audio Interface Timing and AI and AO Channel Arrangement

Audio Interface Timing

Audio interfaces support the standard I²S interface timing and PCM interface timing, and provide various configurations to interconnect with audio CODECs complying with multiple specifications. For details about timings, see chapter 13 "Audio Interfaces" in the related data sheet.

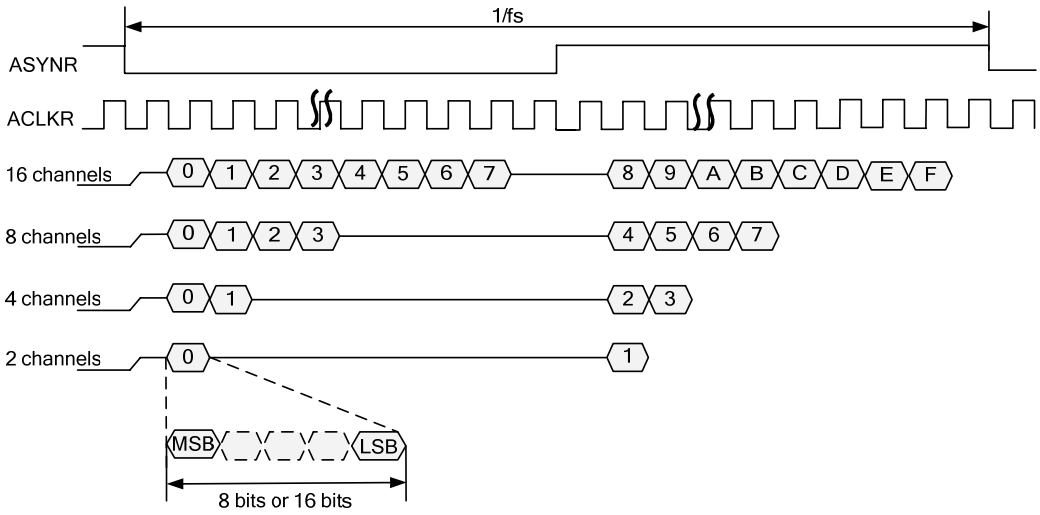
To ensure component connection, understand the I²S or PCM protocol and timing capability of the interconnected audio CODECs. The following describes features of the Hi35xx I²S and PCM:

- Based on the standard I²S or PCM protocol, the most significant bit (MSB) is always prior to the least significant bit (LSB) in transmission. That is, serial data is transmitted from the MSB to the LSB.
- The AI device supports multi-channel I²S and PCM RX timings. The timing mode, sync clock, and sampling bit width of the audio CODEC must be consistent with those configured in the AI device. Otherwise, the collected data may be incorrect.
- The AI and AO devices support master and slave modes. In mast mode, the AI or AO device provides clocks. For the chip using SIO interfaces, the input and output clocks are shared. For the chip using AIO interfaces, there is no such a limitation if the AI and AO device do not simultaneously connect to a CODEC. In slave mode, the external audio CODEC provides independent input and output clocks.
- When the AI or AO device works in slave mode, you are advised to configure the interconnected CODEC and then the AI or AO device. When the AI or AO device works in master mode, you are advised to configure the AI or AO device and then the interconnected CODEC. The recommendations are provided based on timings.
- In master mode, some AI and AO devices provide only the frame sync clock and bit stream clock for timing synchronization but not the main clock (MCLK). In this case, if the audio CODEC uses the external crystal oscillator clock as the working clock, the sound may be distorted. Therefore, you are advised to select the slave mode or use the clock generated by the bit stream clock as the working clock of the audio CODEC.
- If the AI or AO device works in master mode and provides the MCLK, the MCLK is configured as follows:

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

  - When the sampling rate is 48 kHz, 24 kHz, or 12 kHz, a 12.288 MHz MCLK is provided.
  - When the sampling rate is 32 kHz, 16 kHz, or 8 kHz, the bit width for the 32 kHz sampling rate is not 256 bits, and the bit width for the 8 kHz sampling rate is not 16 bits, a 12.288 MHz MCLK is provided.
  - When the sampling rate is 32 kHz, 16 kHz, or 8 kHz, the bit width for the 32 kHz sampling rate is 256 bits, or the bit width for the 8 kHz sampling rate is 16 bits, an 8.192 MHz MCLK is provided.
  - When the sampling rate is 44.1 kHz, 22.05 kHz, or 11.025 kHz, an 11.2896 MHz MCLK is provided.
- The AI or AO device supports standard and customized PCM modes.

Figure 9-2 2/4/8/16-channel multiplexing of I^2S



AI and AO Channels

AI and AO channels are concepts from the aspect of software. The following describes the meanings of AI and AO channels. For example, when an AI device uses the I^2S timing supporting multi-channel multiplexing to receive data, the standard I^2S protocol supports only an audio-left channel and an audio-right channel. In this case, the AI device receives at most 128-bit audio data on the audio-left channel and the audio-right channel respectively. Assume that the audio CODEC has the multiplexing function. The 16 channels of 16-bit audio data are multiplexed into 2 channels of 128-bit data, one transmitted to the audio-left channel and the other transmitted to the audio-right channel. Then the AI device parses the 16-channel audio data. The 16 channels are called 16 audio channels.

The multiplexing modes supported by SIO or AIO interfaces vary according to the protocol. See Table 9-1 and Table 9-2.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Table 9-1 Maximum number of AI and AO channels supported by SIO interfaces

SIO Multiplexing	Maximum Audio Data	Channel Instance
I^2S receive (RX) timing	128 bits on the audio-left channel and audio-right channel respectively	Sixteen 16-bit channels or sixteen 8-bit channels
PCM RX timing	256 bits on a mono channel	Sixteen 16-bit channels or sixteen 8-bit channels
I^2S transmit (TX) timing	32 bits on the audio-left channel and audio-right channel respectively	Four 16-bit channels or eight 8-bit channels
PCM TX timing	8 bits to 128 bits on a mono channel	Eight 16-bit channels

Table 9-2 Maximum number of AI and AO channels supported by AIO interfaces

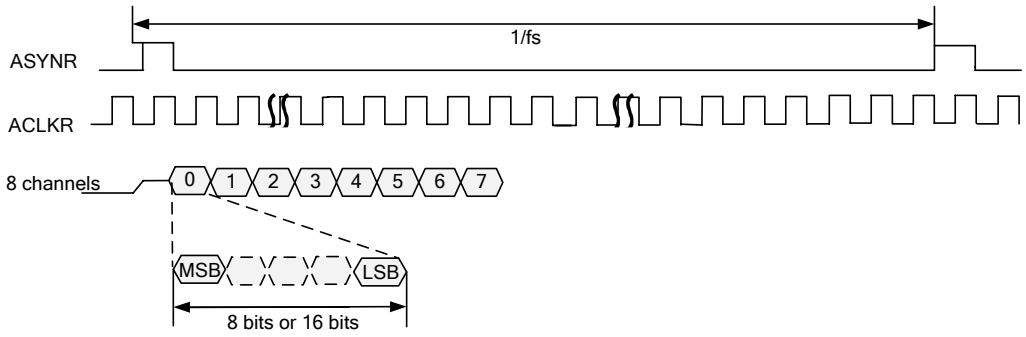
AIO Multiplexing	Maximum Audio Data	Channel Instance
I^2S RX timing	128 bits on the audio-left channel and audio-right channel respectively	Sixteen 16-bit channels or sixteen 8-bit channels
PCM RX timing	256 bits on a mono channel	Sixteen 16-bit channels or sixteen 8-bit channels
I^2S TX timing	128 bits on the audio-left channel and audio-right channel respectively	Sixteen 16-bit channels or sixteen 8-bit channels
PCM TX timing	256 bits on a mono channel	Sixteen 16-bit channels or sixteen 8-bit channels

Based on the sampling precision and timing, the AI channel and the AO channel can be split into AiChn0, AiChn1, …, and AiChnN or AoChn0, AoChn1, …, AoChnN respectively when the data amount does not exceed the threshold of the AI or AO device. Audio data can be correctly received or sent when the I^2S or PCM timing configured on the AI or AO device is consistent with that configured in the audio CODEC. The process of unifying the timings of the SIO interface and the audio CODEC is called interconnection.

Assume that the standard PCM slave mode, 8 kHz sampling rate, 16-bit sampling precision, and four channels are configured for the AOU, and 8 kHz frame sync clock, 8 kHz x 64 bit stream, and 64-bit (the audio CODEC can multiplex four 16-bit sampling points as one sampling point) sampling point are configured for the external CODEC. In this case, the channel arrangement for the PCM timing is shown in Figure 9-3.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 9-3 PCM TX timing



The Hi3531, Hi3532, Hi3521, or Hi3520A supports a maximum of 16 AI channels and 16 AO channels. The Hi3518/Hi3516C supports two AI channels and two AO channels due to the limitations exerted by the built-in audio CODEC. When you configure the AI device and AO device, ensure that the number of channels is an even number. In stereo mode, the channels whose channel IDs are smaller than half of the number of channels are audio-left channels, and the other channels are audio-right channels. Two channels whose channel ID difference is half of the number of channels are a stereo channel pair. You need to operate only audio-left channels, because the corresponding audio-right channels are automatically operated in the SDK. As shown in Figure 9-2, channel 0 and channel 4, channel 1 and channel 5, channel 2 and channel 6, and channel 3 and channel 7 are four stereo channel pairs, and four stereo outputs are supported. You need to operate only audio-left channels 0–3.

For the chip that uses SIO interface, the AI and AO devices connected to the same SIO interface must work in the same mode. For example, the mode of the AI and AO devices is configured to the I^2S timing slave mode. When the SIO interface works in master mode, the product of the number of AI channels multiplied by the sampling precision must be equal to the product of the number of AO channels multiplied by the sampling precision. This ensures that the clock transmitted by the SIO interface to the audio CODEC is consistent for the AI device and AO device. For example, if the AI device uses four 16-bit channels, the AO device can use eight 8-bit channels. When the SIO interface works in slave mode, the multiplication product for the AI channels can be inconsistent with that for the AO channels. In this case, the audio CODEC transmits different clocks to the AI device and AO device.

For the chip using AIO interfaces, the AI and AO devices for AIO interfaces are independent. The preceding limitation does not apply when the AI and AO devices do not connect to the same CODEC. The Hi35xx SDK can bind interfaces by using the SYS module to establish the binding relationship between an AI channel and an AO channel, playing audio in real time.

9.2.1.4 Echo Cancellation, Resampling, and Noise Reduction

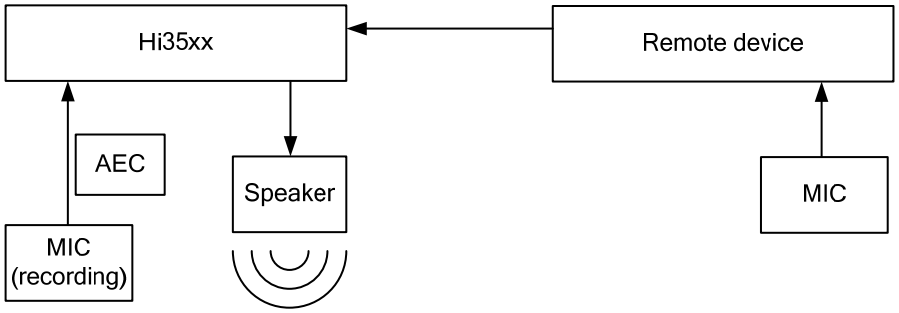
NOTE
The functions described in this section are not supported currently.

Echo Cancellation

The Hi35xx cancels the echoes generated in audio data sampling. When remote audio data is played on the Hi35xx AO device, the data is sampled locally by using a MIC. The Hi35xx can cancel the echoes generated in recording audio data played on the AO device.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 9-4 Echo cancellation



To enable the echo cancellation function, ensure that the following conditions are met:

- A mono channel is used.
- The sampling rate is 8 kHz, 11 kHz, or 16 kHz.
- The sampling precision is 8 bits or 16 bits.
- When the sampling rate is 8 kHz, the frame length with 80 or 160 sampling points is supported.
- When the sampling rate is 11 kHz or 16 kHz, the frame length with 160 sampling points is supported.
- The frame length of the voice data captured by the AI channel must be the same as that of the remote voice played on the AO device.

Both the AI and AO devices must meet the preceding requirements.

CAUTION

- If an AI channel is bound to an AENC channel by calling the system binding interface, perform echo cancellation in the AENC channel and then encoding after enabling echo cancellation in the AI channel.
- If an AI channel is bound to an AENC channel and an AO channel by other means but not calling the system binding interface, you can obtain AI frames by calling the related MPI, perform echo cancellation in the AI channel, and then transmit the frames to the AENC and AO channels after enabling echo cancellation in the AI channel. In this case, echo cancellation takes effect in the AI channel.

Resampling

The Hi35xx AIU and AOU can resample audio data. If HI_MPI_AI_GetFrame is called when AI resampling sampling is enabled, resampling is internally performed and then the processed data is obtained. If AO resampling is enabled, audio data is resampled before it is transmitted to the AO channel for playing.

Audio resampling is classified into 1x2, 1x4, 1x6, 2x1, 4x1, and 6x1 resampling. The supported sampling rates vary according to resampling multiples. See Table 9-3.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Table 9-3 Sampling rates corresponding to resampling multiples

Resampling Multiple	Sampling Rate
1x2	8 kHz, 11.025 kHz, 16 kHz, 22.05 kHz, 24 kHz
1x4	8 kHz, 11.025 kHz, 12 kHz
1x6	8 kHz
2x1	48 kHz, 44.1 kHz, 32 kHz, 16 kHz
4x1	48 kHz, 44.1 kHz, 32 kHz
6x1	48 kHz

Resampling is available only for the mono channel. Echo cancellation is not supported during resampling.

CAUTION

- When an AI channel and an AENC channel or an AI channel and an AO channel are bound by the system for data transfer, AI or AO resampling is invalid. You can call related MPIs to obtain AI frames and transmit frames to the AENC channel and AO channel. In this way, data is transferred between the AI channel and AENC channel or AI channel and AO channel, and AI or AO resampling is valid.
- The preceding limits are not applicable to the data transfer between an ADEC channel and an AO channel. That is, when an ADEC channel and an AO channel are bound by the system, AO resampling is also valid.

Resampling attributes need to be configured for this function:

- u32InPointNum: number of sampling points per frame before resampling.
- enInSampleRate: sampling rate before resampling.
- enReSampleType: resampling type.

The sampling rate and sampling points of the audio frames before and after resampling are different. For example, if the AI sampling rate is 32 kHz and the number of sampling points is 1280, the AI resampling parameter enInSampleRate must be set to 32 kHz and u32InPointNum must be set to 1280. If enReSampleType is set to 4x1, the sample rate is changed to 8 kHz and the number of sampling points is changed to 320 after resampling.

Noise Reduction

The Hi35xx AIU supports noise reduction for audio data. If AI noise reduction is enabled, it is performed in HI_MPI_AI_GetFrame, which is similar to resampling.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

CAUTION

- If data is transferred between the AIU and the AOU or between the AIU and the AENC module in SYS_BIND mode, noise reduction does not work. You can transfer data by using user threads.
- Echo cancellation involves noise reduction. Therefore, noise reduction is not required if you enable echo cancellation.
- Noise reduction applies only to the audio data that is sampled at the sampling rate of 8 kHz (80 sampling points per frame), 11 kHz (160 sampling points per frame), or 16 kHz (320 sampling points per frame). The stereo data is not supported.

9.2.1.5 Advanced Resampling and VQE

Advanced Resampling

The audio advanced resampling of the Hi35xx supports the resampling between any two different sampling rates. The input and output sampling rates supported by the resampling include: 8 kHz, 11.025 kHz, 12 kHz, 16 kHz, 22.05 kHz, 24 kHz, 32 kHz, 44.1 kHz, and 48 kHz.

Voice Quality Enhancement

The Hi35xx AIU supports voice quality enhancement (VQE). The VQE includes echo cancellation, noise reduction, automatic gain and high-pass filtering. The preceding four modules of the VQE can be used at the same time or separately. If the AI VQE is enabled, it is performed in HI_MPI_AI_GetFrame in non-binding mode and performed in HI_MPI_AENC_SendFrame in binding mode.

Echo cancellation: The echo cancellation function here is the same as that described in 9.2.1.4. To enable the echo cancellation function, ensure that the following conditions are met:

- A mono channel is used.
- The sampling rate is 8 kHz, 11.025 kHz, 12 kHz, or 16 kHz.
- The sampling precision is 16 bits.
- The frame length of the voice data captured by the AI channel must be the same as that of the remote voice played on the AO device.

Both the AI and AO devices must meet the preceding requirements.

Noise reduction: The noise reduction function here is the same as that described in 9.2.1.4. To enable the noise reduction function, ensure that the following conditions are met:

- A mono channel is used.
- The sampling rate is 8 kHz, 11.025 kHz, 12 kHz, or 16 kHz.

Automatic gain: The Hi35xx AIU supports automatic gain (the automatic gain module in the VQE). When the input volume changes, the automatic gain module limits the output volume to a range. The sampling rate is 8 kHz, 11.025 kHz, 12 kHz, or 16 kHz.

High-pass filtering: The high-pass filtering function is used to filter the low-frequency noises in the audio frame. The sampling rate is 8 kHz, 11.025 kHz, 12 kHz, or 16 kHz. You are advised to always enable high-pass filtering when VQE is used.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

CAUTION

- If an AI channel is bound to an AO or AENC channel by calling a system binding interface, only the echo cancellation module of the VQE takes effect. If an AI channel is bound to an AO or AENC channel by other means but not calling the system binding interface, you can obtain AI audio frames by calling HI_MPI_AI_GetFrame, and then transmit the frames to the AENC or AO channel. In this case, the three modules of the AI VQE take effect.
- The VQE includes noise reduction, automatic gain control, and echo cancellation. You can configure some or all functions of VQE by calling HI_MPI_AI_SetVqeAttr.
- The VQE does not support stereo. For details about the requirements on the parameters related to sampling rate, frame length, automatic level range, noise reduction, and echo cancellation, see the members of AI_ALC_CONFIG_S, AI_AEC_CONFIG_S, and AI_VQE_CONFIG_S in section 9.4.

9.2.2 Audio Encoding and Decoding

Audio Encoding and Decoding Processes

For the Hi35xx SDK, G711, G726, ADPCM_DVI4, or ADPCM_ORG_DVI4 audio decoding is hardware encoding, and ADPCM_IMA audio decoding is CPU software decoding. As the Hi3518/Hi3516C does not provide a hardware encoding module, all Hi3518/Hi3516C encoding modes are software encoding. All decoding functions are implemented based on the HiSilicon audio encoding and decoding library that is independently encapsulated. The core decoder works in user mode and performs decoding by using the CPU. The SDK can bind an AI channel to an AENC channel by using an SYS module to implement the encoding function; the SDK can also bind an ADEC channel to an AO channel to implement the decoding function.

Audio Encoding and Decoding Protocols

Table 9-4 describes the audio encoding and decoding protocols supported by the Hi35xx.

Table 9-4 Audio encoding and decoding protocols

Protocol	Sampling Rate	Frame Duration (Sampling Point)	Bit Rate (kbit/s)	Compression Rate	CPU Usage	Description
G711	8 kHz	80/160/ 240/320/480	64	2	1 MHz	Advantages: best voice quality; low CPU consumption; wide application; and free of charge. Disadvantage: low compression efficiency. The G.711 provides A-law and μ-law compression codes, which are applicable to the ISTN and most digital telephone lines. North America and Japan usually adopt μ-law code. Europe and other

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Protocol	Sampling Rate	Frame Duration (Sampling Point)	Bit Rate (kbit/s)	Compression Rate	CPU Usage	Description
						regions usually adopt A-law code.
G726	8 kHz	80/160/ 240/320/480	16, 24, 32, 40 (Note: G726 encoding is a lossy compression . When the bit rate is low, the compression rate and quantization error are large, which may affect the voice quality.)	8–3.2	5 MHz	Advantages: simple algorithm; good voice quality; ensured voice quality after conversions for several times; network-level voice quality at a low bit rate. Disadvantage: low compression efficiency. The G726_16KBPS and the MEDIA_G726_16KBPS encoders differ in the format of packages output. The G726_16KBPS is applicable to network transmission and the MEDIA_G726_16KBPS is applicable to ASF storage. For details, see the *RFC3551*.
ADPCM	8 kHz	80/160/ 240/320/480 or 81/161/241/ 321/481	32	4	2 MHz	Advantages: simple algorithm; good voice quality; ensured voice quality after conversions for several times; network-level voice quality at a low bit rate. Disadvantage: low compression efficiency. The ADPCM_IMA, ADPCM_ORG_DVI4, and ADPCM_DVI formats are packet encapsulation formats of the ADPCM. The IMA packet encapsulation uses the first sampling point as the predicted value, that is, one more sampling point needs to be input per frame. The DVI packet encapsulation uses the preceding frame as the predicted value. Therefore, the input sampling points for the IMA encoding are 81/161/241/321/481 and the input sampling points for the DVI encoding are 80/160/240/320/480.

NOTE

- The CPU usage is calculated based on the 288 MHz ARM9. For example, the value 2 MHz indicates that 2 MHz CPU is used for decoding, and the CPU usage is 0.69 (2/288).

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- G726 encoding is a lossy compression. When the bit rate is low, the compression rate and quantization error are large, which may affect the voice quality.
- Exceptions might occur when packets are discarded during transmission because the predicted transmission value is not provided for ADPCM_ORG_DVI4. Therefore, you are advised not to use this protocol in audio stream transmission applications.
- When the LPCM protocol is used, the AIU and AOU can record and play audio files in LPCM format.

Structure of the HiSilicon Voice Frame

When the HiSilicon voice encoding/decoding library is used to encode G711, G726, and ADPCM audio data, the encoded stream complies with the structure described in Table 9-5. That is, a 4-byte frame header is added before the payload of each frame stream. When the voice encoding/decoding library is used to decode the audio data, the header information needs to be read.

The data payload length in the frame header is calculated as follows:

- Data payload length = Bit rate x Number of sampling points/1280000 (for G726 encoding)
- Data payload length = Number of sampling points/2 (for G711a encoding)
- Data payload length = Number of sampling points/4 + 2 (for ADPCM-DVI4 encoding)

Table 9-5 Structure of the HiSilicon voice frame

Parameter Position (Unit: HI_S16)	Parameter Bit	Meanings
0	[15:8]	Flag of the frame type. 01: voice frame. Other values: reserved.
	[7:0]	Reserved.
1	[15:8]	Frame circulation counter: 0 ≥ 255.
	[7:0]	Length of the payload (unit: HI_S16).
2	[15:0]	Payload data.
3	[15:0]	Payload data.
…	[15:0]	Payload data.
2 + n – 1	[15:0]	Payload data.
2 + n	[15:0]	Payload data.

9.2.3 Differences Among Chips

Audio Interfaces

Table 9-6 describes the audio interface type for each chip.

Copyright © HiSilicon Technologies Co., Ltd

Table 9-6 Audio interface type for each chip

Chip	Audio Interface Type
Hi3531/Hi3532 Hi3521/Hi3520A Hi3518/Hi3516C	SIO interface
Hi3520D/Hi3515A/Hi3515C	AIO interface

Chips Using SIO Interfaces

The integrated SIO interfaces are classified into:

- SIO interface supporting only audio inputs
- SIO interface supporting intercom
- SIO interface connected to the HDMI module of the chip

For details about SIO specifications, see the related data sheet.

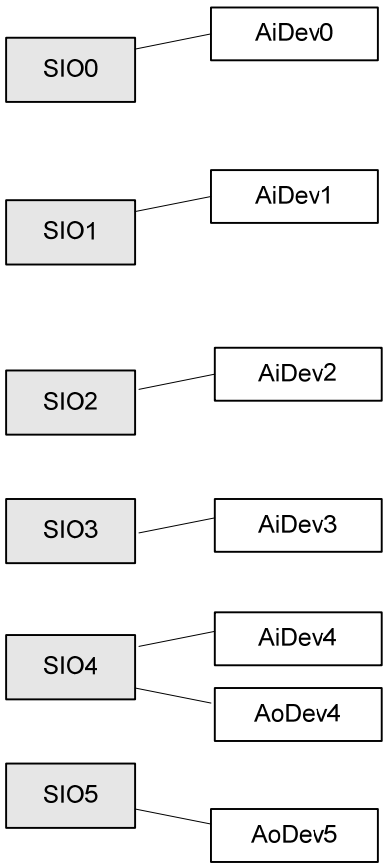
When an SIO interface that supports only audio inputs or intercom works in master mode, only the frame sync clock and bit stream clock for timing synchronization are provided. That is, no MCLK is provided. If the audio CODEC uses the external crystal oscillator clock as the working clock, the sound may be distorted. Therefore, you are advised to select the slave mode or use the clock generated by the bit stream clock as the working clock of the audio CODEC. Note that the SIO interface connected to the HDMI interface must work in master I^2S mode.

When two or more SIO interfaces that support only audio inputs work in master mode, they share an MCLK. The MCLK value is set after the attributes of the AI or AO device corresponding to the first SIO interface in master mode are set successfully. When the attributes of the AI or AO devices corresponding to other SIO interfaces are set, the required clocks must be the same as the MCLK. Otherwise, an error occurs.

Figure 9-5 shows the mapping between SIO interfaces and AI/AO devices for the Hi3531 or Hi3532.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 9-5 Mapping between SIO interfaces and AI/AO devices for the Hi3531 or Hi3532



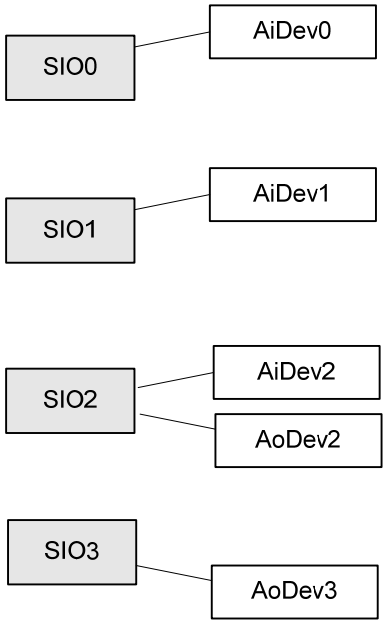
NOTE

- In the software development kit (SDK), an AI device and an AO device are reserved for the SIO interface that supports only input or output. However, if these AI devices or AO devices such as AoDev0–AoDev3 and AoDev5 are used, an error occurs.
- SIO4 supports intercom.
- Only the Hi3531 provides SIO5. SIO5 connects to the HDMI and supports only the master I^2S mode.

Figure 9-6 shows the mapping between SIO interfaces and AI/AO devices for the Hi3521 or Hi3520A.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 9-6 Mapping between SIO interfaces and AI/AO devices for the Hi3521 or Hi3520A

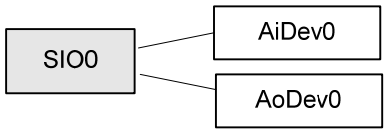


NOTE

- In the software development kit (SDK), an AI device and an AO device are reserved for the SIO interface that supports only input or output. However, if these AI devices or AO devices such as AoDev0 and AoDev1 and AoDev3 are used, an error occurs.
- SIO2 supports intercom.
- SIO3 connects to the HDMI and supports only the master I^2S mode.

Figure 9-7 shows the mapping between SIO interface and AI/AO devices for the Hi3518/Hi3516C.

Figure 9-7 Mapping between SIO interfaces and AI/AO devices for the Hi3518/Hi3516C



NOTE

- The Hi3518/Hi3516C uses the internal audio CODEC. The Hi3518A/Hi3516C supports stereo channels. That is, audio-left channel and audio-right channel inputs and outputs are supported. The Hi3518C/Hi3518E supports only the mono channel. That is, audio-left channel input and audio-left channel output are supported.
- SIO0 supports only the master I^2S mode and can be used for intercom.

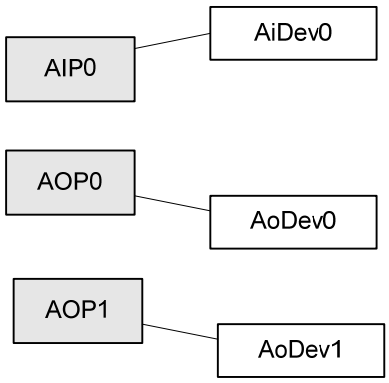
Chips Using AIO Interfaces

- The integrated AIO interfaces include AIPs and AOPs. For details about AIO interface specifications, see the *Hi35xx H.264 CODEC Processor Data Sheet*. Figure 9-8 shows the mapping between AIPs/AOPs and AI/AO devices for the Hi3520D/Hi3515A/Hi3515C.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- When AOP0 works in master mode, you can determine whether to share the configurations of the frame sync clock and bit stream clock of AOP0. When AOP0 works in slave mode, its configurations of the frame sync clock and bit stream clock cannot be shared.

Figure 9-8 Mapping between AIPs/AOPs and AI/AO devices for the Hi3520D/Hi3515A/Hi3515C



NOTE

- AIP0 and AOP0 simultaneously connect to a CODEC and can be used for intercom.
- AOP1 connects to the HDMI and supports only the I^2S master mode.

Ranges of AI and AO Device IDs

Table 9-7 lists the ranges of AI and AO device IDs.

Table 9-7 Ranges of AI and AO device IDs

Chip	Range of the AI Device ID	Range of the AO Device ID
Hi3531	[0, 4]	[4, 5]
Hi3532	[0, 4]	[4]
Hi3521/Hi3520A	[0, 2]	[2, 3]
Hi3518/Hi3516C	[0]	[0]
Hi3520D/Hi3515A/Hi3515C	[0]	[0, 1]

9.2.4 Built-In Audio CODEC

9.2.4.1 Overview

The Hi3518A, Hi3518C/Hi3518E, or Hi3516C provides a built-in audio CODEC that connects to the SIO0 interface in the chip. That is, the SIO0 interface can play and record audio only by connecting to the built-in audio CODEC. Because the audio CODEC cannot transmit the sync clock to the SIO0 interface, the SIO0 interface works only in master I^2S

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

timing mode. The interconnection timing for the SIO0 interface and the audio CODEC must be properly configured so that data can be transferred between the audio CODEC and the SIO0 interface.

9.2.4.2 Important Concepts

The audio CODEC consists of an analog part and a digital part. The analog part selects the microphone input or LINEIN input by using the analog mixing (MICPGA). The analog mixing supports gain adjustment. The digital part includes the ADC and DAC. The ADC and DAC are used to convert analog signals and digital signals, and adjust the volume. You can adjust both the analog volume and digital volume. It is recommended that you adjust the analog volume first. If the analog gain does not meet requirements, enable **Boost**. The digital gain is adjusted to the optimal value by default.

The DAC supports soft mute and soft unmute. Soft mute is used to control the digital audio value until it decreases to 0, and soft unmute is used to control the digital audio value until it increases to the configured gain value. The rate for changing the value is controlled by mute_rate. When soft mute is enabled, the digital audio value gradually decreases to 0; when soft mute is disabled, the digital audio value restores to the configured gain value.

The working clock of the audio CODEC is provided by the SIO0 interface. The clocks of the analog part and digital part are originated from the working clock of the audio CODEC, and the clocks of the ADC and DAC are originated from the clock of the digital part. If the sound is abnormal when default configurations are used, you can reverse the clocks of the digital part and analog part or the clocks of the DAC and ADC. The default sampling precision of the audio CODEC is 16 bits.

The audio CODEC supports de-emphasis filtering, pop sound suppression, and high-pass filtering. These functions are enabled by default.

9.2.4.3 ioctl Function

The user-mode interface of the audio CODEC is an ioctl function. Its syntax is as follows:

```
int ioctl (int fd,
           unsigned long cmd,
          ……
          );
```

This function is a standard Linux interface and has variable parameters. Only three parameters are required for the audio CODEC. The syntax is as follows:

```
int ioctl (int fd,
           unsigned long cmd,
           CMD_DATA_TYPE *cmddata);
```

The value of **CMD_DATA_TYPE** varies according to the values of **cmd**. Table 9-8 describes the three parameters of the ioctl function.

Table 9-8 Parameters of the ioctl function

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file. It is returned after the audio CODEC file is opened by calling the open function.	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
cmd	The main command control words are as follows: • ACODEC_SOFT_RESET_CTRL: Restores the audio CODEC to default settings. • ACODEC_SET_I2S1_FS: Sets the sampling rate of the I²S1 interface. • ACODEC_SET_MIXER_MIC: Selects the input audio channel MICIN or LINEIN. • ACODEC_SET_GAIN_MICL: Sets the analog gain of the audio-left input channel. • ACODEC_SET_GAIN_MICR: Sets the analog gain of the audio-right input channel. • ACODEC_SET_DACL_VOL: Sets the volume of the audio-left output channel. • ACODEC_SET_DACR_VOL: Sets the volume of the audio-right output channel. • ACODEC_SET_ADCL_VOL: Sets the volume of the audio-left input channel. • ACODEC_SET_ADCR_VOL: Sets the volume of the audio-right input channel. • ACODEC_SET_MICL_MUTE: Sets the mute of the audio-left input channel. • ACODEC_SET_MICR_MUTE: Sets the mute of the audio-right input channel. • ACODEC_SET_DACL_MUTE: Sets the mute of the audio-left output channel. • ACODEC_SET_DACR_MUTE: Sets the mute of the audio-right output channel. • ACODEC_DAC_SOFT_MUTE: Controls the soft mute function of the DAC. • ACODEC_DAC_SOFT_UNMUTE: Controls the soft unmute function of the DAC. • ACODEC_GET_GAIN_MICL: Obtains the analog gain of the audio-left input channel. • ACODEC_GET_GAIN_MICR: Obtains the analog gain of the audio-right input channel. • ACODEC_GET_DACL_VOL: Obtains the volume of the audio-left output channel. • ACODEC_GET_DACR_VOL: Obtains the volume of the audio-right output channel. • ACODEC_GET_ADCL_VOL: Obtains the volume of the audio-left input channel. • ACODEC_GET_ADCR_VOL: Obtains the volume of the audio-right input channel. • ACODEC_SET_PD_DACL: Powers off the audio-left output channel.	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
	• ACODEC_SET_PD_DACR: Powers off the audio-right output channel. • ACODEC_SET_PD_ADCL: Powers off the audio-left input channel. • ACODEC_SET_PD_ADCR: Powers off the audio-right channel. • ACODEC_SEL_DAC_CLK: Sets the clock edges of the DAC and ADC to same or opposite. • ACODEC_SEL_ANA_MCLK: Sets the clock edges of the analog part and digital part to same or opposite. • ACODEC_DACL_SEL_TRACK: Sets the DACL for selecting the audio channel. • ACODEC_DACR_SEL_TRACK: Sets the DACR for selecting the audio channel. • ACODEC_ADCL_SEL_TRACK: Sets the ADCL for selecting the audio channel. • ACODEC_ADCR_SEL_TRACK: Sets the ADCR for selecting the audio channel. • ACODEC_SET_DAC_DE_EMPHASIS: Controls the de-emphasis filtering function of the DAC. • ACODEC_SET_ADC_HP_FILTER: Controls the high-pass filtering function of the ADC. • ACODEC_DAC_POP_FREE: Deletes the POP noise of the DAC. • ACODEC_DAC_SOFT_MUTE_RATE: Controls the rate for soft mute of the DAC. • ACODEC_DAC_SEL_I2S: Selects the I^2S interface for the DAC. • ACODEC_ADC_SEL_I2S: Selects the I^2S interface for the ADC. • ACODEC_SET_I2S1_DATAWIDTH: Sets the data width of the I^2S1 interface. • ACODEC_SET_I2S2_DATAWIDTH: Sets the data width of the I^2S2 interface. • ACODEC_SET_I2S2_FS: Sets the sampling rate of the I^2S2 interface. • ACODEC_SET_DACR2DACL_VOL: Mixes the volume of the DACR with that of the DACL. • ACODEC_SET_DACL2DACR_VOL: Mixes the volume of the DACL with that of the DACR. • ACODEC_SET_ADCL2DACL_VOL: Mixes the volume of the ADCL with that of the DACL. • ACODEC_SET_ADCR2DACL_VOL: Mixes the volume of the ADCR with that of the DACL. • ACODEC_SET_ADCL2DACR_VOL: Mixes the volume of the DACR with that of the DACL.	

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
	• ACODEC_SET_ADCR2DACR_VOL: Mixes the volume of the ADCR with that of the DACR.	
cmddata	Data pointers corresponding to command control words	Input/output

9.3 API Reference

9.3.1 AI

The AIU is used to configure and enable AI devices and obtain audio frames.

The AIU provides the following MPIs:

- HI_MPI_AI_SetPubAttr: Sets attributes of an AI device.
- HI_MPI_AI_GetPubAttr: Obtains attributes of an AI device.
- HI_MPI_AI_Enable: Enables an AI device.
- HI_MPI_AI_Disable: Disables an AI device.
- HI_MPI_AI_EnableChn: Enables an AI channel.
- HI_MPI_AI_DisableChn: Disables an AI channel.
- HI_MPI_AI_GetFrame: Obtains audio frames.
- HI_MPI_AI_ReleaseFrame: Releases the buffer for storing audio frames.
- HI_MPI_AI_SetChnParam: Sets the parameters of an AI channel.
- HI_MPI_AI_GetChnParam: Obtains the parameters of an AI channel.
- HI_MPI_AI_EnableAec: Enables the echo cancellation function.
- HI_MPI_AI_DisableAec: Disables the echo cancellation function.
- HI_MPI_AI_EnableReSmp: Enables AI resampling.
- HI_MPI_AI_DisableReSmp: Disables AI resampling.
- HI_MPI_AI_EnableAnr: Enables AI noise reduction.
- HI_MPI_AI_DisableAnr: Disables AI noise reduction.
- HI_MPI_AI_GetFd: Obtains the device file descriptor (FD) of an AI channel.
- HI_MPI_AI_SetHpfAttr: Sets the AI high-pass filtering attribute.
- HI_MPI_AI_GetHpfAttr: Obtains the AI high-pass filtering attribute.
- HI_MPI_AI_SetVqeAttr: Sets the VQE attributes.
- HI_MPI_AI_GetVqeAttr: Obtains the VQE attributes.
- HI_MPI_AI_EnableVqe: Enables the VQE functions.
- HI_MPI_AI_DisableVqe: Disables the VQE functions.
- HI_MPI_AI_EnableReSmpEx: Enables the AI advanced resampling function.
- HI_MPI_AI_DisableReSmpEx: Disables the AI advanced resampling function.
- HI_MPI_AI_ClrPubAttr: Clears public attributes.
- HI_MPI_AI_Save_File: Enables the function of saving AI files.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_AI_SetPubAttr

[Description]

Sets attributes of an AI device.

[Syntax]

```
HI_S32 HI_MPI_AI_SetPubAttr(AUDIO_DEV AudioDevId, const AIO_ATTR_S
*pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
pstAttr	Pointer to the AI attributes	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

The attributes of the AI device determine the format of the input data. The attributes include SIO working mode, sampling rate, sampling precision, buffer size, number of sampling points per frame, extension flag, clock selection, and number of channels. An AI device can properly connect to an external CODEC only when the preceding attributes are consistent with these of the timings of the connected CODEC.

- Working mode

  Currently, the AI and AO devices support master and slave I^2S modes, master and slave standard PCM modes, and master and slave customized PCM modes.

  When an AI device works in slave mode, you are advised to configure the interconnected CODEC and then the AI device. When an AI device works in master mode, you are advised to configure the AI device and then the interconnected CODEC. The recommendations are provided based on timings.

- Sampling rate

  Sampling rate indicates the number of sampling points per second. A high sampling rate means that the distortion is low. The data that needs to be processed increases. Typically, 8 kHz sampling rate is used for voice and 32 kHz or higher sampling rate is used for

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

audio. Ensure that the sampling rate to be set is supported by the interconnected audio CODEC.

- Sampling precision

  Sampling precision indicates the width of sampling point data on a channel. This attribute determines the allocation of channels throughout the device. The sampling precision can be set to 8 bits or 16 bits. The actual sampling precision is affected by the audio CODEC.

- Buffer size

  u32FrmNum in the AIO_ATTR_S is used to configure the number of audio frames in the buffer storing the audio data received by the AIU. The recommended value for this parameter is 5 or greater. Otherwise, exceptions such as frame loss may occur during sampling.

- Number of sampling points per frame

  When the sampling rate is high, you are advised to increase the number of sampling points per frame. To transmit the captured audio data for encoding, ensure that the duration for each frame is longer than or equal to 10 ms. Otherwise, the sound is abnormal after decoding. For example, if the sampling rate is 16 kHz, the number of sampling points for each frame must be at least 160.

- Extension flag

  This flag is valid only when the sampling precision is 8 bits. This flag determines whether to expand the 8-bit data to 16–bit data by signing it. The extended data is 16 bits, meeting requirements of the encoder.

- Number of channels

  This attribute indicates the number of channels supporting the AI function of the current input device. The number of channels must be the same as that of the audio CODEC. The number of channels can be 2, 4, 8, or 16.

- Clock selection
  - An AI device can properly work only when it works with the ADC. To obtain data from the correct channel, you must be familiar with the relationship between the channel IDs and the distribution of data collected by the ADC. When the AI device works in master mode, its output clock depends on the sampling rate, sampling precision, and number of channels. The sampling rate multiplied by the number of channels is the bit width for a sampling using the AI device timing.
  - If the chip uses an SIO interface and the SIO interface supports both RX and TX functions, the interface provides RX and TX sync clock lines to interconnect with the audio CODEC. This attribute can be used to determine whether the TX sync clock is multiplexed with the RX sync clock. In master mode, the AI and AO clocks are provided by the SIO interface. Therefore, the clocks are always consistent and u32ClkSel is invalid. In slave mode, if u32ClkSel is set to 1, the AI and AO clocks are multiplexed, and the AI and AO clocks must be consistent. If u32ClkSel is set to 0, the AI and AO clocks are provided by different sources. Ensure that the AI device is disabled before setting this attribute.
  - The master and slave timings of the AI device and AO device under the same SIO interface must be consistent. If the timings are not consistent, an error is returned. For example, when the AI device is configured with the slave I^2S mode, if the AO device is configured with the master PCM mode, an error is returned.
  - When the SIO interface works in master mode, the configuration of the SIO output clock depends on the sampling rate, sampling precision, and number of channels. The sampling precision multiplied by number of channels is the bit width of each sampling in accordance with the SIO timing.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- When the SIO interface works in master mode or u32ClkSel is 1, the clocks configured for the AI device and AO device under the same SIO interface must be consistent. That is, for the AI device and AO device, the products of the sampling precision multiplied by the number of channels must be the same, and the sampling rates must also be the same.
- For the chip using AIO interfaces, the AI and AO devices are independent. When AoDev0 works in master mode, you can configure u32ClkSel to determine whether to share the configurations of the frame sync clocks and bit stream clocks of AiDev0 and AoDev0. If u32ClkSel is set to 1, the preceding configurations of AiDev0 and AoDev0 are shared. In this case, ensure that the clock configurations of AiDev0 are the same as those of AoDev0. If u32ClkSel is set to 0, the frame sync clocks and bit stream clocks of AiDev0 and AoDev0 are provided by difference sources. In slave mode, the configurations of the frame sync clocks and bit stream clocks of AiDev0 and AoDev0 are not shared, and setting u32ClkSel has no effect. When AiDev0 and AoDev0 work in master mode and u32ClkSel is set to 1, the configurations of the frame sync clock and bit stream clock of AiDev0 must be the same as those of AoDev0. That is, the product of the sampling precision multiplied by the number of channels for AiDev0 must be the same as that for AoDev0, and the sampling rates for AiDev0 and AoDev0 must also be the same.

[Example]

To set attributes of an AI device and enable the AI device, see the following codes:

```
HI_S32 s32ret;
AIO_ATTR_S stAttr;
AUDIO_DEV AiDevId = 1;

stAttr.enBitwidth = AUDIO_BIT_WIDTH_16;
stAttr.enSamplerate = AUDIO_SAMPLE_RATE_8000;
stAttr.enSoundmode = AUDIO_SOUND_MODE_MONO;
stAttr.enWorkmode = AIO_MODE_I2S_SLAVE;
stAttr.u32EXFlag = 0;
stAttr.u32FrmNum = 5;
stAttr.u32PtNumPerFrm = 160;
stAttr.u32ChnCnt = 16;
stAttr.u32ClkSel = 1;

/* set public attribute of AI device*/
s32ret = HI_MPI_AI_SetPubAttr(AiDevId, &stAttr);
if(HI_SUCCESS != s32ret)
{
printf("set ai %d attr err:0x%x\n", AiDevId,s32ret);
return s32ret;
}
/* enable AI device */
s32ret = HI_MPI_AI_Enable(AiDevId);
if(HI_SUCCESS != s32ret)
{
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
printf("enable ai dev %d err:0x%x\n", AiDevId, s32ret);
return s32ret;
}
```

[See Also]

None

HI_MPI_AI_GetPubAttr

[Description]

Obtains attributes of an AI device.

[Syntax]

```
HI_S32 HI_MPI_AI_GetPubAttr(AUDIO_DEV AudioDevId, AIO_ATTR_S*pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
pstAttr	Pointer to the general AI attributes	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

- The obtained attributes are the attributes configured last time.
- If attributes are never configured, a code indicating failure is returned.

[Example]

```
HI_S32 s32ret;
AUDIO_DEV AiDevId = 1;
AIO_ATTR_S stAttr;

s32ret = HI_MPI_AI_GetPubAttr(AiDevId, &stAttr);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
if(HI_SUCCESS != s32ret)
{
    printf("get ai %d attr err:0x%x\n", AiDevId,s32ret);
    return s32ret;
}
```

[See Also]

None

HI_MPI_AI_Enable

[Description]

Enables an AI device.

[Syntax]

```
HI_S32 HI_MPI_AI_Enable(AUDIO_DEV AudioDevId);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Table 9-7.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

- The attributes of the AI device must be configured before the AI device is enabled. Otherwise, an error indicating that attributes are not configured is returned.
- A code indicating success is returned if the AI device is enabled.

[Example]

See HI_MPI_AI_ClrPubAttr.

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_AI_Disable

[Description]

Disables an AI device.

[Syntax]

```
HI_S32 HI_MPI_AI_Disable(AUDIO_DEV AudioDevId);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Table 9-7.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

- A code indicating success is returned if the AI device is disabled.
- All AI channels of the AI device must be disabled before the AI device is disabled.
- Before calling this MPI to disable an AI device, you must disable the AENC channel and AO device related to the AI device. Otherwise, calling this MPI fails.

[Example]

```
HI_S32 s32ret;
AUDIO_DEV AiDevId = 1;

s32ret = HI_MPI_AI_Disable(AiDevId);
if(HI_SUCCESS != s32ret)
{
    printf("disable ai %d err:0x%x\n", AiDevId);
    return s32ret;
}
```

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

HI_MPI_AI_EnableChn

[Description]

Enables an AI channel.

[Syntax]

```
HI_S32 HI_MPI_AI_EnableChn(AUDIO_DEV AudioDevId, AI_CHN AiChn);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
AiChn	ID of an AI channel The ID range depends on the AI device attribute parameters **u32ChnCnt** (maximum number of channels) and **enSoundmode** (audio channel mode). For details, see AUDIO_SOUND_MODE_E.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, i_ai.h, _comm_aio.h
- Library file: libmpi.a

[Note]

Before the AI channel is enabled, the AI device on which the AI channel exists must be enabled. Otherwise, an error indicating that the device is not enabled is returned.

[Example]

None

[See Also]

None

HI_MPI_AI_DisableChn

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Disables an AI channel.

[Syntax]

```
HI_S32 HI_MPI_AI_DisableChn(AUDIO_DEV AudioDevId, AI_CHN AiChn);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
AiChn	ID of an AI channel Value range: [0, AIO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

None

[Example]

None

HI_MPI_AI_GetFrame

[Description]

Obtains audio frames.

[Syntax]

```
HI_S32 HI_MPI_AI_GetFrame(AUDIO_DEV AudioDevId, AI_CHN AiChn,
AUDIO_FRAME_S *pstFrm, AEC_FRAME_S *pstAecFrm, HI_BOOL bBlock);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
AiChn	ID of an AI channel The ID range depends on the AI device attribute parameters **u32ChnCnt** (maximum number of channels) and **enSoundmode** (audio channel mode).	Input
pstFrm	Pointer to the structure of the audio frame	Output
pstAecFrm	Pointer to the structure of the reference frame for echo cancellation	Output
bBlock	Block/Non-block flag Value: HI_TRUE or HI_FALSE	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, ensure that the AI channel is enabled or paused. Otherwise, the error code is returned, indicating that no AI channel is enabled.
- If AI echo cancellation is enabled, **pstAecFrm** cannot be **null**; if AI echo cancellation is disabled, **pstAecFrm** can be **null**.
- The AIU buffers the audio frame data that can be fetched by users. You can set the buffer depth by calling HI_MPI_AI_SetChnParam. The default buffer depth is 0.

[Example]

None

[See Also]

None

HI_MPI_AI_ReleaseFrame

[Description]

Releases the buffer for storing audio frames.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_AI_ReleaseFrame(AUDIO_DEV AudioDevId, AI_CHN AiChn,
AUDIO_FRAME_S *pstFrm, AEC_FRAME_S *pstAecFrm);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
AiChn	ID of an AI channel The ID range depends on the AI device attribute parameters **u32ChnCnt** (maximum number of channels) and **enSoundmode** (audio channel mode).	Input
pstFrm	Pointer to the structure of the audio frame	Input
pstAecFrm	Pointer to the structure of the reference frame for echo cancellation	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

The value of **pstAecFrm** in HI_MPI_AI_ReleaseFrame must be the same as the value of **pstAecFrm** in HI_MPI_AI_GetFrame.

[Example]

None

[See Also]

None

HI_MPI_AI_SetChnParam

[Description]

Sets the parameters of an AI channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_AI_SetChnParam(AUDIO_DEV AudioDevId, AI_CHN AiChn,
AI_CHN_PARAM_S *pstChnParam);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
AiChn	ID of an AI channel The ID range depends on the AI device attribute parameters **u32ChnCnt** (maximum number of channels) and **enSoundmode** (audio channel mode).	Input
pstChnParam	Audio channel parameter	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, ensure that the AI device is enabled. Otherwise, the error code is returned, indicating that no AI device is enabled.
- The channel parameter has only one variable. This variable is used to set the depth of the buffer for storing the audio frames obtained by users. The default buffer depth is 0. The maximum value of the variable is 30.
- You are advised to call HI_MPI_AI_GetChnParam to obtain default configurations before calling HI_MPI_AI_SetChnParam to modify configurations. This facilitates the future expansion.

[Example]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_AI_GetChnParam

[Description]

Obtains the parameters of an AI channel.

[Syntax]

```
HI_S32 HI_MPI_AI_GetChnParam(AUDIO_DEV AudioDevId, AI_CHN AiChn,
AI_CHN_PARAM_S *pstChnParam);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
AiChn	ID of an AI channel The ID range depends on the AI device attribute parameters **u32ChnCnt** (maximum number of channels) and **enSoundmode** (audio channel mode).	Input
pstChnParam	Audio channel parameter	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

Before calling this MPI, ensure that the AI device is enabled. Otherwise, the error code is returned, indicating that no AI device is enabled.

[Example]

None

[See Also]

None

Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_AI_EnableAec

[Description]

Enables the echo cancellation function of the specified AIU/AI channel and AOU/AO channel.

[Syntax]

```
HI_S32 HI_MPI_AI_EnableAec(AUDIO_DEV AiDevId,
AI_CHN AiChn, AUDIO_DEV AoDevId, AO_CHN AoChn);
```

[Parameter]

Parameter	Description	Input/Output
AiDevId	ID of the AI device on which the echo needs to be canceled For details about the value range, see Table 9-7.	Input
AiChn	ID of the AI channel on which the echo needs to be canceled The ID range depends on the maximum number of channels specified by the AI device attribute parameter **u32ChnCnt**.	Input
AoDevId	ID of the AO device on which the echo cancellation function is enabled For details about the value range, see Table 9-7.	Input
AoChn	ID of the AO channel on which the echo cancellation function is enabled Value range: [0, AIO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library files: libmpi.a, libresampler.a, libaec.a, libanr.a

[Note]

- The AI channel must be enabled before the echo cancellation function is enabled.
- This MPI can be called repeatedly. If this MPI is called repeatedly, a code indicating success is returned.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Call this MPI to enable echo cancellation if you want to use echo cancellation after enabling an AI channel again.

[Example]

```
HI_S32 s32ret;
AUDIO_DEV AiDevId = 1;
AI_CHN AiChn = 0;
AUDIO_DEV AoDevId = 0;
AO_CHN AoChn = 0;

s32ret = HI_MPI_AI_EnableAec(AiDevId, AiChn,AUDIO_DEV AoDevId,AO_CHN
AoChn);
if(HI_SUCCESS != s32ret)
{
    printf("enable aec err:0x%x\n", s32ret);
    return s32ret;
}

s32ret = HI_MPI_AI_DisableAec(AiDevId, AiChn)
if(HI_SUCCESS != s32ret)
{
    printf("disable aec err:0x%x\n", s32ret);
    return s32ret;
}
```

[See Also]

None

HI_MPI_AI_DisableAec

[Description]

Disables the echo cancellation function.

[Syntax]

```
HI_S32 HI_MPI_AI_DisableAec(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

[Parameter]

Parameter	Description	Input/Output
AiDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
AiChn	ID of an AI channel Value range: [0, AIO_MAX_CHN_NUM)	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h
- Library files: libmpi.a, libresampler.a, libaec.a, libanr.a

[Note]

This MPI can be called repeatedly. If this MPI is called repeatedly, a code indicating success is returned.

[Example]

See HI_MPI_AI_EnableAec.

[See Also]

None

HI_MPI_AI_EnableReSmp

[Description]

Enables AI resampling.

[Syntax]

```
HI_S32 HI_MPI_AI_EnableReSmp(AUDIO_DEV AudioDevId, AI_CHN AiChn,
AUDIO_RESAMPLE_ATTR_S *pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
AiChn	ID of an AI channel The ID range depends on the maximum number of channels specified by the AI device attribute parameter **u32ChnCnt**.	Input
pstAttr	Pointer to the AI resampling attribute structure	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library files: libmpi.a, libresampler.a, libaec.a, libanr.a

[Note]

- Call this MPI to enable resampling after enabling an AI channel.
- Resampling can be repeatedly enabled. Ensure that the attributes configured every time are the same.
- If you want to enable resampling after disabling an AI channel, you must enable this channel and call this MPI again.

[Example]

To set the AI resampling rate from 32 kHz to 8 kHz, do as follows:

```
/* dev attr of ai */
stAioAttr.u32ChnCnt = 2;
stAioAttr.enBitwidth = AUDIO_BIT_WIDTH_16;
stAioAttr.enSamplerate = AUDIO_SAMPLE_RATE_32000;
stAioAttr.enSoundmode = AUDIO_SOUND_MODE_MONO;
stAioAttr.u32EXFlag = 1;
stAioAttr.u32FrmNum = 30;
stAioAttr.u32PtNumPerFrm = 320*4;

/* attr of resample */
stReSampleAttr.u32InPointNum = 320 * 4;
stReSampleAttr.enInSampleRate = AUDIO_SAMPLE_RATE_32000;
stReSampleAttr.enReSampleType = AUDIO_RESAMPLE_4X1;
```

HI_MPI_AI_DisableReSmp

[Description]

Disables AI resampling.

[Syntax]

```
HI_S32 HI_MPI_AI_DisableReSmp(AUDIO_DEV AudioDevId, AI_CHN AiChn);
```

[Parameter]

Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
AiChn	ID of an AI channel Value range: [0, AIO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library files: libmpi.a, libresampler.a, libaec.a, libanr.a

[Note]

- This MPI is called to disable AI resampling when resampling is not used any more.
- Before calling this MPI, you must disable the AENC channel and AO channel corresponding to the AI device. Otherwise, calling this MPI fails.

HI_MPI_AI_EnableReSmpEx

[Description]

Enables the AI advanced resampling function.

[Syntax]

```
HI_S32 HI_MPI_AI_EnableReSmpEx(AUDIO_DEV AudioDevId, AI_CHN AiChn,
AUDIO_RESAMPLE_ATTR_EX_S *pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Figure 9-7.	Input
AiChn	ID of an AI channel Value range: [0, AIO_MAX_CHN_NUM)	Input
pstAttr	Pointer to the AI advanced resampling attribute structure	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library files: libmpi.a, libresampler.a, libaec.a, libanr.a

[Note]

- Call this MPI to enable resampling after enabling an AI channel.
- Resampling can be repeatedly enabled. Ensure that the attributes configured every time are the same.
- After an AI channel is disabled, if you want to enable it again and use the resampling function, you need to call this MPI to enable resampling.
- The advanced resampling supports the resampling between any two different sampling rates. The input and output sampling rates supported by the resampling include: 8 kHz, 11.025 kHz, 12 kHz, 16 kHz, 22.05 kHz, 24 kHz, 32 kHz, 44.1 kHz, and 48 kHz.
- The AI resampling function and the AI advanced resampling function cannot be enabled at the same time.

[Example]

None

HI_MPI_AI_DisableReSmpEx

[Description]

Disables the AI advanced resampling function.

[Syntax]

```
HI_S32 HI_MPI_AI_DisableReSmpEx(AUDIO_DEV AudioDevId, AI_CHN AiChn);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Figure 9-7.	Input
AiChn	ID of an AI channel Value range: [0, AIO_MAX_CHN_NUM)	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library files: libmpi.a, libresampler.a, libaec.a, libanr.a

[Note]

- If AI advanced resampling is not used any more, call this MPI to disable it.
- Before calling this MPI, you must disable the AENC channel and AO channel corresponding to the AI device. Otherwise, calling this MPI fails.

HI_MPI_AI_EnableAnr

[Description]

Enables AI noise reduction.

[Syntax]

```
HI_S32 HI_MPI_AI_EnableAnr(AUDIO_DEV AudioDevId, AI_CHN AiChn);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
AiChn	ID of an AI channel The ID range depends on the maximum number of channels specified by the AI device attribute parameter **u32ChnCnt**.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h

Copyright © HiSilicon Technologies Co., Ltd

- Library files: libmpi.a, libresampler.a, libaec.a, libanr.a

[Note]

- Call this MPI to enable AI noise reduction after enabling an AI channel.
- A code indicating success is returned if you call this MPI repeatedly.
- Call this MPI to enable noise reduction if you want to use noise reduction after enabling an AI channel again.

[Example]

None

HI_MPI_AI_DisableAnr

[Description]

Disables AI noise reduction.

[Syntax]

```
HI_S32 HI_MPI_AI_DisableAnr (AUDIO_DEV AudioDevId, AI_CHN AiChn);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
AiChn	ID of an AI channel Value range: [0, AIO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h
- Library files: libmpi.a, libresampler.a, libaec.a, libanr.a

[Note]

If noise reduction is not used any more, call this MPI to disable it.

HI_MPI_AI_GetFd

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Obtains the device FD of an AI channel.

[Syntax]

```
HI_S32 HI_MPI_AI_GetFd(AUDIO_DEV AudioDevId, AI_CHN AiChn)
```

[Parameter]

Parameter	Description	Input/Output
AiDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
AiChn	ID of an AI channel Value range: [0, AIO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
Positive value	Valid
Non-positive value	Invalid

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h
- Library file: libmpi.a

[Note]

None

[Example]

```
HI_S32 s32AiFd;
HI_S32 s32ret;
AUDIO_DEV AiDevId = 1;
AI_CHN AiChn = 0;

s32AiFd = HI_MPI_AI_GetFd(AiDevId, AiChn);
if(s32AiFd <= 0)
{
    return HI_FAILURE;
}
```

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_AI_SetHpfAttr

[Description]

Sets the AI high-pass filtering attribute.

[Syntax]

```
HI_S32 HI_MPI_AI_SetHpfAttr(AUDIO_DEV AiDevId, AI_CHN AiChn,
AI_HPF_ATTR_S *pstHpfAttr);
```

[Parameter]

Parameter	Description	Input/Output
AiDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
AiChn	ID of an AI channel Value range: [0, AIO_MAX_CHN_NUM)	Input
pstHpfAttr	Pointer to the AI high-pass filtering attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

- This MPI must be called before VQE is enabled; otherwise, an error occurs. This MPI cannot be dynamically called after VQE is enabled.
- If this MPI is not called before VQE is enabled, the high-pass filtering function is disabled by default.

[Example]

None

[See Also]

None

HI_MPI_AI_GetHpfAttr

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Obtains the AI high-pass filtering attribute.

[Syntax]

```
HI_S32 HI_MPI_AI_GetHpfAttr(AUDIO_DEV AiDevId, AI_CHN AiChn,
AI_HPF_ATTR_S *pstHpfAttr);
```

[Parameter]

Parameter	Description	Input/Output
AiDevId	ID of an AI device For details about the value range, see Table 9-7.	Input
AiChn	ID of an AI channel Value range: [0, AIO_MAX_CHN_NUM)	Input
pstHpfAttr	Pointer to the AI high-pass filtering attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

None

HI_MPI_AI_SetVqeAttr

[Description]

Sets the AI VQE attributes.

[Syntax]

```
HI_S32 HI_MPI_AI_SetVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_DEV
AoDevId, AO_CHN AoChn, AI_VQE_CONFIG_S *pstVqeConfig);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
AiDevId	ID of an AI device For details about the value range, see Figure 9-7.	Input
AiChn	ID of an AI channel Value range: [0, AIO_MAX_CHN_NUM)	Input
AoDevId	ID of the AO device for which echo cancellation is performed For details about the value range, see Figure 9-7.	Input
AoChn	ID of the AO channel for which echo cancellation is performed Value range: [0, AIO_MAX_CHN_NUM)	Input
pstVqeConfig	Pointer to the AI VQE attribute structure	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

- Set the VQE attributes of the corresponding AI channel before enabling the VQE functions.
- Enable the corresponding AI channel before configuring AI VQE attributes.
- The VQE attributes of an AI channel cannot be dynamically set. Therefore, before setting the VQE attributes of an AI channel again, you need to disable its VQE functions.
- The VQE functions include echo cancellation, noise reduction, and automatic gain control. You can enable some or all functions of VQE by setting corresponding VQE attributes.

[Example]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_AI_GetVqeAttr

[Description]

Obtains the AI VQE attributes.

[Syntax]

```
HI_S32 HI_MPI_AI_GetVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn,
AI_VQE_CONFIG_S *pstVqeConfig);
```

[Parameter]

Parameter	Description	Input/Output
AiDevId	ID of an AI device For details about the value range, see Figure 9-7.	Input
AiChn	ID of an AI channel Value range: [0, AIO_MAX_CHN_NUM)	Input
pstVqeConfig	Pointer to the AI VQE attribute structure	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

Before calling this MPI, ensure that the VQE attributes are configured.

[Example]

None

[See Also]

None

HI_MPI_AI_EnableVqe

[Description]

Enables the AI VQE functions.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
HI_S32 HI_MPI_AI_EnableVqe(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

[Parameter]

Parameter	Description	Input/Output
AiDevId	ID of an AI device For details about the value range, see Figure 9-7.	Input
AiChn	ID of an AI channel Value range: [0, AIO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library files: libmpi.a, libresampler.a, libaec.a, libanr.a

[Note]

- Enable the corresponding AI channel before enabling VQE.
- If VQE of an AI channel is enabled for multiple times, a code indicating success is returned.
- After an AI channel is disabled, if you want to enable it again and use VQE, you need to call this MPI to enable VQE.

[See Also]

None

HI_MPI_AI_DisableVqe

[Description]

Disables the AI VQE functions.

[Syntax]

```
HI_S32 HI_MPI_AI_DisableVqe(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
AiDevId	ID of an AI device For details about the value range, see Figure 9-7.	Input
AiChn	ID of an AI channel Value range: [0, AIO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h
- Library files: libmpi.a, libresampler.a, libaec.a, libanr.a

[Note]

This interface can be called repeatedly without failure.

[Example]

None

[See Also]

None

HI_MPI_AI_ClrPubAttr

[Description]

Clears public attributes.

[Syntax]

```
HI_S32 HI_MPI_AI_ClrPubAttr(AUDIO_DEV AudioDevId);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Figure 9-7.	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Positive value	Valid
Non-positive value	Invalid

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h
- Library file: libmpi.a

[Note]

None

[See Also]

None

HI_MPI_AI_Save_File

[Description]

Enables the function of saving AI files.

[Syntax]

```
HI_S32 HI_MPI_AI_Save_File(AUDIO_DEV AudioDevId, AI_CHN AiChn,
AUDIO_SAVE_FILE_INFO_S *pstSaveFileInfo);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AI device For details about the value range, see Figure 9-7.	Input
AiChn	ID of an AI channel Value range: [0, AIO_MAX_CHN_NUM)	Input
pstSaveFileInfo	Pointer to the attribute structure of the function of saving AI files	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Header files: hi_comm_aio.h, mpi_ao.h
- Library file: libmpi.a

[Note]

This MPI is only used to dump AI files in non-binding mode. After this MPI is called, three files in specified size are generated in the specified folder. Sin.pcm is the input frame, Rin.pcm is the reference frame, and Sou.pcm is the output frame.

[See Also]

None

9.3.2 AO

An AOU mainly implements the following functions: enabling an AOU and transmitting audio frames to AO channels.

The AOU provides the following MPIs:

- HI_MPI_AO_SetPubAttr: Sets attributes of an AO device.
- HI_MPI_AO_GetPubAttr: Obtains attributes of an AO device.
- HI_MPI_AO_Enable: Enables an AO device.
- HI_MPI_AO_Disable: Disables an AO device.
- HI_MPI_AO_EnableChn: Enables an AO channel.
- HI_MPI_AO_DisableChn: Disables an AO channel.
- HI_MPI_AO_SendFrame: Transmits AO frames.
- HI_MPI_AO_EnableReSmp: Enables AO resampling.
- HI_MPI_AO_DisableReSmp: Disables AO resampling.
- HI_MPI_AO_EnableReSmpEx: Enables the AO advanced resampling function.
- HI_MPI_AO_DisableReSmpEx: Disables the AO advanced resampling function.
- HI_MPI_AO_PauseChn: Pauses an AO channel.
- HI_MPI_AO_ResumeChn: Resumes an AO channel.
- HI_MPI_AO_ClearChnBuf: Clears the current audio data buffer on the AO channel.
- HI_MPI_AO_QueryChnStat: Queries the status of the current audio data buffer on the AO channel.
- HI_MPI_AO_SetVolume: Sets the volume of an AO device.
- HI_MPI_AO_GetVolume: Obtains the volume of an AO device.
- HI_MPI_AO_GetFd: Obtains the device FD of an AO channel.
- HI_MPI_AO_ClrPubAttr: Clears public attributes.

HI_MPI_AO_SetPubAttr

[Description]

Sets attributes of an AO device.

[Syntax]

```
HI_S32 HI_MPI_AO_SetPubAttr(AUDIO_DEV AudioDevId, const AIO_ATTR_S
*pstAttr);
```


Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Table 9-7.	Input
pstAttr	Attributes of the AO device	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h
- Library file: libmpi.a

[Note]

- The AO device must be disabled before attributes are set. If the AO device is in Enable state, disable the AO device.
- The AI device and AO device under the same SIO interface must be consistent in master/slave mode and clocks. Otherwise, an error is returned when attributes are set.
- The extension flag is invalid to the AO device.
- When the SIO interface works in master mode, the configuration of the SIO output clock depends on the sampling rate, sampling precision, and number of channels. The sampling precision multiplied by number of channels is the bit width of each sampling in accordance with the SIO timing.
- When the SIO interface works in master mode, the clocks for the AI device and AO device under the same SIO interface must be consistent, namely, the value of **u32ClkSel** in AIO_ATTR_S is the same.
- The AO device must work with the DA. To transmit data to the right channel, users need to know the relationship between data (sent by the DA) allocation and channels.
- The extended flag is invalid for the AO device.
- For other attributes of the AO device, see the related description about the AIU in section 9.3.1 "AI."

[Example]

```
HI_S32 s32ret;
AIO_ATTR_S stAttr;
AUDIO_DEV AoDevId = 0;

stAttr.enBitwidth = AUDIO_BIT_WIDTH_16;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
stAttr.enSamplerate = AUDIO_SAMPLE_RATE_8000;
stAttr.enSoundmode = AUDIO_SOUND_MODE_MONO;
stAttr.enWorkmode = AIO_MODE_I2S_SLAVE;
stAttr.u32EXFlag = 0;
stAttr.u32FrmNum = 5;
stAttr.u32PtNumPerFrm = 160;
stAttr.u32ChnCnt = 2;
stAttr.u32ClkSel = 0;

/* set ao public attr*/
s32ret = HI_MPI_AO_SetPubAttr(AoDevId, &stAttr);
if(HI_SUCCESS != s32ret)
{
    printf("set ao %d attr err:0x%x\n", AoDevId,s32ret);
    return s32ret;
}
/* enable ao device*/
s32ret = HI_MPI_AO_Enable(AoDevId);
if(HI_SUCCESS != s32ret)
{
    printf("enable ao dev %d err:0x%x\n", AoDevId, s32ret);
    return s32ret;
}
```

[See Also]

None

HI_MPI_AO_GetPubAttr

[Description]

Obtains attributes of an AO device.

[Syntax]

```
HI_S32 HI_MPI_AO_GetPubAttr(AUDIO_DEV AudioDevId, AIO_ATTR_S *pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Table 9-7.	Input
pstAttr	Pointer to the AO attributes	Output

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h
- Library file: libmpi.a

[Note]

- The obtained attributes are the attributes configured last time.
- If attributes are never configured, a code indicating failure is returned.

[Example]

```
HI_S32 s32ret;
AUDIO_DEV AoDevId = 0;
AIO_ATTR_S stAttr;

/* first enable ao device*/

s32ret = HI_MPI_AO_GetPubAttr(AoDevId, &stAttr);
if(HI_SUCCESS != s32ret)
{
    printf("get ao %d attr err:0x%x\n", AoDevId,s32ret);
    return s32ret;
}
```

[See Also]

None

HI_MPI_AO_Enable

[Description]

Enables an AO device.

[Syntax]

```
HI_S32 HI_MPI_AO_Enable(AUDIO_DEV AudioDevId);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Table 9-7.	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h
- Library file: libmpi.a

[Note]

- The attributes of the AO device must be configured before the AO device is enabled. Otherwise, an error indicating that attributes are not configured is returned.
- A code indicating success is returned if the AO device is enabled.

[Example]

See HI_MPI_AO_SetPubAttr.

[See Also]

None

HI_MPI_AO_Disable

[Description]

Disables an AO device.

[Syntax]

```
HI_S32 HI_MPI_AO_Disable(AUDIO_DEV AudioDevId);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Table 9-7.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h
- Library file: libmpi.a

[Note]

- A code indicating success is returned if the AO device is disabled.
- All AO channels of the AO device must be disabled before the AO device is disabled.

[Example]

None

[See Also]

None

HI_MPI_AO_EnableChn

[Description]

Enables an AO channel.

[Syntax]

```
HI_S32 HI_MPI_AO_EnableChn(AUDIO_DEV AudioDevId, AI_CHN AoChn);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Table 9-7.	Input
AoChn	ID of an AO channel The ID range depends on the AO device attribute parameters **u32ChnCnt** (maximum number of channels) and **enSoundmode** (audio channel mode).	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h, hi_comm_aio.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

Before the AO channel is enabled, the AO device on which the AI channel exists must be enabled. Otherwise, an error indicating that the device is not enabled is returned.

[Example]

See HI_MPI_AI_SetPubAttr.

[See Also]

None

HI_MPI_AO_DisableChn

[Description]

Disables an AO channel.

[Syntax]

```
HI_S32 HI_MPI_AO_DisableChn(AUDIO_DEV AudioDevId, AI_CHN AoChn);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Table 9-7.	Input
AoChn	ID of an AO channel Value range: [0, AIO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

None

[Example]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_AO_SendFrame

[Description]

Transmits AO frames.

[Syntax]

```
HI_S32 HI_MPI_AO_SendFrame(AUDIO_DEV AudioDevId, AO_CHN AoChn, const
AUDIO_FRAME_S *pstData, HI_BOOL bBlock);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Table 9-7.	Input
AoChn	ID of the AO channel The ID range depends on the AO device attribute parameters **u32ChnCnt** (maximum number of channels) and **enSoundmode** (audio channel mode).	Input
pstData	Pointer to the structure of the audio frame	Input
bBlock	Block/non-block flag Value: HI_TRUE or HI_FALSE	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, ensure that the AO channel is enabled or paused. Otherwise, the error code is returned, indicating that no AO channel is enabled.
- HI_MPI_AO_SendFrame is used to transmit audio frames to the AO channel. If the AO channel is bound to the AI or ADEC channel by calling HI_MPI_SYS_Bind, do not call HI_MPI_AO_SendFrame.

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_AO_EnableReSmp

[Description]

Enables AO resampling.

[Syntax]

```
HI_S32 HI_MPI_AO_EnableReSmp(AUDIO_DEV AudioDevId, AO_CHN AoChn,
AUDIO_RESAMPLE_ATTR_S *pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Table 9-7.	Input
AoChn	ID of an AO channel The ID range depends on the maximum number of channels specified by the AO device attribute parameter **u32ChnCnt**.	Input
pstAttr	Pointer to the AO resampling attribute structure	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h, hi_comm_aio.h
- Library files: libmpi.a, libresampler.a, libaec.a, libanr.a

[Note]

- This MPI is called before an AO channel is bound and after an AO channel is enabled.
- Resampling can be repeatedly enabled. Ensure that the attributes configured every time are the same.
- If you want to enable resampling after disabling an AO channel, you must enable this channel and call this MPI again.

[Example]

To set the resampling rate for the ADEC module to decode data to the AOU from 8 kHz to 32 kHz, do as follows:

```
/* dev attr of ao */
stAioAttr.u32ChnCnt = 2;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
stAioAttr.enBitwidth = AUDIO_BIT_WIDTH_16;
stAioAttr.enSamplerate = AUDIO_SAMPLE_RATE_32000;
stAioAttr.enSoundmode = AUDIO_SOUND_MODE_MONO;
stAioAttr.u32EXFlag = 1;
stAioAttr.u32FrmNum = 30;
stAioAttr.u32PtNumPerFrm = 320*4;

/* attr of resample */
stReSampleAttr.u32InPointNum = 320;
stReSampleAttr.enInSampleRate = AUDIO_SAMPLE_RATE_8000;
stReSampleAttr.enReSampleType = AUDIO_RESAMPLE_1X4;
```

HI_MPI_AO_DisableReSmp

[Description]

Disables AO resampling.

[Syntax]

```
HI_S32 HI_MPI_AO_DisableReSmp(AUDIO_DEV AudioDevId, AO_CHN AoChn);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Table 9-7.	Input
AoChn	ID of an AO channel Value range: [0, AIO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h, hi_comm_aio.h
- Library files: libmpi.a, libresampler.a, libaec.a, libanr.a

[Note]

This MPI is called to disable AO resampling when resampling is not used any more.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_AO_EnableReSmpEx

[Description]

Enables the AO advanced resampling function.

[Syntax]

```
HI_S32 HI_MPI_AO_EnableReSmpEx(AUDIO_DEV AudioDevId, AO_CHN AoChn,
AUDIO_RESAMPLE_ATTR_EX_S *pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Figure 9-7.	Input
AoChn	ID of an AO channel Value range: [0, AIO_MAX_CHN_NUM)	Input
pstAttr	Pointer to the AO advanced resampling attribute structure	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h, hi_comm_aio.h
- Library files: libmpi.a, libresampler.a, libaec.a, libanr.a

[Note]

- Call this MPI to enable the resampling function after the AO channel is enabled and before the AO channel is bound.
- Resampling can be repeatedly enabled. Ensure that the attributes configured every time are the same.
- After an AO channel is disabled, if you want to enable it again and use the resampling function, you need to call this MPI to enable resampling.

[Example]

None

HI_MPI_AO_DisableReSmpEx

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Disables the AO advanced resampling function.

[Syntax]

```
HI_S32 HI_MPI_AO_DisableReSmpEx(AUDIO_DEV AudioDevId, AO_CHN AoChn);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Figure 9-7.	Input
AoChn	ID of an AO channel Value range: [0, AIO_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h, hi_comm_aio.h
- Library files: libmpi.a, libresampler.a, libaec.a, libanr.a

[Note]

If AO advanced resampling is not used any more, call this MPI to disable it.

[See Also]

None

HI_MPI_AO_PauseChn

[Description]

Pauses an AO channel.

[Syntax]

```
HI_S32 HI_MPI_AO_PauseChn(AUDIO_DEV AudioDevId, AI_CHN AoChn);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Table 9-7.	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
AoChn	ID of an AO channel The ID range depends on the AO device attribute parameters **u32ChnCnt** (maximum number of channels) and **enSoundmode** (audio channel mode).	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

- After an AO channel is paused, if the ADEC channel bound to the AO channel transmits audio frames to this channel, the frames are blocked; if the AI channel bound to the AO channel transmits audio frames to this channel, the frames are stored in the channel buffer when the channel buffer is not full or the frames are discarded when the channel buffer is full.
- When the AO channel is disabled, you are not allowed to call this MPI to pause the AO channel.

[Example]

None

[See Also]

None

HI_MPI_AO_ResumeChn

[Description]

Resumes an AO channel.

[Syntax]

```
HI_S32 HI_MPI_AO_ResumeChn(AUDIO_DEV AudioDevId, AI_CHN AoChn);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Table 9-7.	Input
AoChn	ID of an AO channel The ID range depends on the AO device attribute parameters **u32ChnCnt** (maximum number of channels) and **enSoundmode** (audio channel mode).	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

- This MPI is called to resume a paused AO channel.
- If an AO channel is paused or enabled, calling this MPI returns a code indicating success. If an AO channel is in other states, calling this MPI returns an error.

[Example]

None

HI_MPI_AO_ClearChnBuf

[Description]

Clears the current audio data buffer on an AO channel.

[Syntax]

```
HI_S32 HI_MPI_AO_ClearChnBuf(AUDIO_DEV AudioDevId, AO_CHN AoChn);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Table 9-7.	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
AoChn	ID of an AO channel The ID range depends on the AO device attribute parameters **u32ChnCnt** (maximum number of channels) and **enSoundmode** (audio channel mode).	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

- This MPI is called after the AO channel is enabled.
- This MPI must work with HI_MPI_ADEC_ClearChnBuf to completely clear the buffer data on the decoding channel.

[Example]

None

HI_MPI_AO_QueryChnStat

[Description]

Queries the status of the current audio data buffer on the AO channel.

[Syntax]

```
HI_S32 HI_MPI_AO_QueryChnStat(AUDIO_DEV AudioDevId, AO_CHN AoChn,
AO_CHN_STATE_S *pstStatus);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an audio device For details about the value range, see Table 9-7.	Input
AoChn	ID of an AO channel The ID range depends on the AO device attribute parameters **u32ChnCnt** (maximum number of channels) and **enSoundmode** (audio channel mode).	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
pstStatus	Pointer to the buffer status data structure	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h, and hi_comm_aio.h
- Library file: libmpi.a

[Note]

Call this MPI only after enabling the AO channel successfully.

[Example]

None

HI_MPI_AO_SetVolume

[Description]

Sets the volume of an AO device.

[Syntax]

```
HI_S32 HI_MPI_AO_SetVolume(AUDIO_DEV AudioDevId, AO_CHN AoChn, HI_S32
s32VolumeDb);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an audio device For details about the value range, see Table 9-7.	Input
AoChn	ID of the AO channel The ID range depends on the AO device attribute parameters **u32ChnCnt** (maximum number of channels) and **enSoundmode** (audio channel mode).	Input
s32VolumeDb	Volume (in dB) Its value range is [−81, +6].	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

- Call this MPI only after enabling an AO device successfully.
- Only the volume of 16-bit data can be controlled.
- This MPI implements volume adjustment by using a software algorithm. The default output volume (source data volume) is 0 dB. When the value is adjusted from 0 dB to −81 dB, the volume is decreased. When the value is adjusted from 0 dB to 6 dB, the volume is increased. The minimum volume is −81 dB, which indicates the mute status. When the volume of some high-frequency audio data is increased, the waveform amplitude may be exceeded. As a result, crackles occur.
- If an AO device is disabled and then enabled, the volume is restored to the default value 0 dB.
- Volume control is performed based on the AO device. The channel ID is reserved and invalid currently. You only need to set the channel ID to a valid value.
- The Hi3518 and Hi3516C do not support this MPI.

[Example]

None

HI_MPI_AO_GetVolume

[Description]

Obtains the volume of an AO device.

[Syntax]

```
HI_S32 HI_MPI_AO_GetVolume(AUDIO_DEV AudioDevId, AO_CHN AoChn, HI_S32
*ps32VolumeDb);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an audio device For details about the value range, see Table 9-7.	Input
AoChn	ID of the AO channel The ID range depends on the AO device attribute	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
	parameters **u32ChnCnt** (maximum number of channels) and **enSoundmode** (audio channel mode).	
ps32VolumeDb	Pointer to the volume	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

- Call this MPI only after enabling an AO device successfully.
- Volume control is performed based on the AO device. The channel ID is reserved and invalid currently. You only need to set the channel ID to a valid value.
- The Hi3518 and Hi3516C do not support this MPI.

[Example]

None

HI_MPI_AO_GetFd

[Description]

Obtains the device FD of an AO channel.

[Syntax]

```
HI_S32 HI_MPI_AO_GetFd(AUDIO_DEV AudioDevId, AO_CHN AoChn)
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Table 9-7.	Input
AoChn	ID of an AO channel Value range: [0, AIO_MAX_CHN_NUM)	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
Positive value	Valid
Non-positive value	Invalid

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h
- Library file: libmpi.a

[Note]

None

[Example]

```
HI_S32 s32AoFd;
HI_S32 s32ret;
AUDIO_DEV AoDevId = 0;
AO_CHN AoChn = 0;

/* first enable ao device */

s32AoFd = HI_MPI_AO_GetFd(AoDevId, AoChn);
if(s32AoFd <= 0)
{
return HI_FAILURE;
}
```

[See Also]

None

HI_MPI_AO_ClrPubAttr

[Description]

Clears public attributes.

[Syntax]

```
HI_S32 HI_MPI_AO_ClrPubAttr(AUDIO_DEV AudioDevId);
```

[Parameter]

Parameter	Description	Input/Output
AudioDevId	ID of an AO device For details about the value range, see Figure 9-7.	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
Positive value	Valid
Non-positive value	Invalid

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h
- Library file: libmpi.a

[Note]

None

[See Also]

None

9.3.3 AENC

An AENC module mainly provides the following functions: creating AENC channels, transmitting audio frames, and obtaining encoded streams.

The AENC module provides the following MPIs:

- HI_MPI_AENC_CreateChn: Creates an AENC channel.
- HI_MPI_ADEC_DestroyChn: Destroys an AENC channel.
- HI_MPI_AENC_SendFrame: Transmits audio frames for encoding.
- HI_MPI_AENC_GetStream: Obtains AENC streams.
- HI_MPI_AENC_ReleaseStream: Releases AENC streams.
- HI_MPI_AENC_GetFd: Obtains the device FD corresponding to an AENC channel ID.
- HI_MPI_AENC_RegeisterEncoder: Registers an encoder.
- HI_MPI_AENC_UnRegisterEncoder: Deregisters an encoder.
- HI_MPI_AENC_Save_File: Enables the function of saving AENC files.

HI_MPI_AENC_CreateChn

[Description]

Creates an AENC channel.

[Syntax]

```
HI_S32 HI_MPI_AENC_CreateChn(AENC_CHN AeChn, const AENC_CHN_ATTR_S
*pstAttr);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
AeChn	ID of an AENC channel Value range: [0, AENC_MAX_CHN_NUM)	Input
pstAttr	Pointer to the AENC channel attributes	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, hi_comm_aenc.h, mpi_aenc.h
- Library files: libmpi.a, lib_VoiceEngine.a, lib_aec.a

[Note]

- **enType** specifies an encoding protocol for the AENC channel. Currently, G711, G726, and ADPCM are supported. For details, see Table 9-4.
- The protocols listed in Table 9-4 support only 16-bit linear PCM audio data. If the input data is 8-bit sampling precision data, the AENC module extends the data to 16 bits. You are advised to set the extension flag to 1 when configuring the general attributes of the AI device. In this way, AI data is automatically extended from 8 bits to 16 bits.
- Table 9-5 describes the structure of HiSilicon voice frames.
- Some attributes of the AENC channel must match attributes of input audio data, for example, sampling rate and duration of a frame (number of sampling points per frame).
- The buffer size is in unit of frame and its value range is [2, MAX_AUDIO_FRAME_NUM]. You are advised to set the size to 10 or larger. A small buffer may cause exceptions such as frame loss.
- This MPI can be called when the channel ID is not allocated. Otherwise, an error indicating that the channel is created is returned.

[Example]

```
HI_S32 s32ret;
AENC_CHN_ATTR_S stAencAttr;
ADEC_ATTR_ADPCM_S stAdpcmAenc;
AENC_CHN AencChn = 0;
AUDIO_FRAME_S stAudioFrm;
AUDIO_STREAM_S stAudioStream;

stAencAttr.enType = PT_ADPCMA;/* ADPCM */
stAencAttr.u32BufSize = 8;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
stAencAttr.pValue = &stAdpcmAenc;
stAdpcmAenc.enADPCMType = ADPCM_TYPE_DVI4;

/* create aenc chn*/
s32ret = HI_MPI_AENC_CreateChn(AencChn, &stAencAttr);
if (HI_SUCCESS != s32ret)
{
printf("create aenc chn %d err:0x%x\n", AencChn,s32ret);
return s32ret;
}

/* bind AENC to AI channel  */
s32Ret = HI_MPI_AENC_BindAi(AencChn, AiDev, AiChn);
if (s32Ret != HI_SUCCESS)
{
    return s32Ret;
}

/* get stream from aenc chn */
s32ret = HI_MPI_AENC_GetStream(AencChn, &stAudioStream);
if (HI_SUCCESS != s32ret )
{
    printf("get stream from aenc chn %d fail \n", AencChn);
    return s32ret;
}

/* deal with audio stream */

/* release audio stream */
s32ret = HI_MPI_AENC_ReleaseStream(AencChn, &stAudioStream);
if (HI_SUCCESS != s32ret )
{
    return s32ret;
}

/* destroy aenc chn */
s32ret = HI_MPI_AENC_DestroyChn(AencChn);
if (HI_SUCCESS != s32ret )
{
    return s32ret;
}
```

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_AENC_DestroyChn

[Description]

Destroys an AENC channel.

[Syntax]

```
HI_S32 HI_MPI_AENC_DestroyChn(AENC_CHN AeChn);
```

[Parameter]

Parameter	Description	Input/Output
AeChn	ID of an AENC channel Value range: [0, AENC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, hi_comm_aenc.h, mpi_aenc.h
- Library files: libmpi.a, lib_VoiceEngine.a, lib_aec.a

[Note]

- If no channel is created, calling this MPI returns a code indicating success.
- A code indicating failure is returned if users attempt to destroy an AENC channel on which streams are being obtained/released or frames are being sent. Users cannot destroy such an AENC channel.

[Example]

See HI_MPI_AENC_CreateChn.

[See Also]

None

HI_MPI_AENC_SendFrame

[Description]

Transmits audio frames for encoding.

[Syntax]

```
HI_S32 HI_MPI_AENC_SendFrame(AENC_CHN AeChn, const AUDIO_FRAME_S *pstFrm,
const AEC_FRAME_S *pstAecFrm);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
AeChn	ID of an AENC channel Value range: [0, AENC_MAX_CHN_NUM)	Input
pstFrm	Pointer to the structure of the audio frame	Input
pstAecFrm	Pointer to the structure of the reference frame for echo cancellation	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ai.h, hi_comm_aio.h
- Library file: libmpi.a

[Note]

- Before calling this MPI, ensure that an AENC channel is created.
- To disable echo cancellation, set pstAecFrm to null.
- This MPI is a non-block interface. If the audio stream buffer is full, an error code indicating failure is returned.
- HI_MPI_AENC_SendFrame is used to transmit audio frames for encoding. If the AENC channel is bound to the AI channel by calling HI_MPI_SYS_Bind, do not call HI_MPI_AENC_SendFrame.

[Example]

None

[See Also]

None

HI_MPI_AENC_GetStream

[Description]

Obtains encoded streams.

[Syntax]

```
HI_S32 HI_MPI_AENC_GetStream(AENC_CHN AeChn, AUDIO_STREAM_S*pstStream,
HI_BOOL bBlock);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
AeChn	ID of an AENC channel Value range: [0, AENC_MAX_CHN_NUM)	Input
pstStream	Obtained audio streams	Output
bBlock	Block flag Value range: HI_TRUE: block HI_FALSE: non-block	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, hi_comm_aenc.h, mpi_aenc.h
- Library files: libmpi.a, lib_VoiceEngine.a, lib_aec.a

[Note]

- Streams can be obtained after an AENC channel is created. Otherwise, a code indicating failure is returned. If users attempt to destroy an AENC channel on which streams are being obtained, a code indicating failure is returned.
- Streams can be obtained in block or non-block mode. The standard Select system invocation is supported.
- When streams are obtained in block mode, this MPI fails to be called if the audio buffer is empty. This MPI can be called only when new data is received in the buffer, the AENC channel is destroyed, or the buffer is still empty after the MPI is block for 1s.
- To directly obtain raw AI data, create an AENC channel, set enType to PT_LPCM, and bind the AENC channel. Audio data along this AENC channel is the raw AII data.

[Example]

See HI_MPI_AENC_CreateChn.

[See Also]

None

HI_MPI_AENC_ReleaseStream

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Releases streams obtained from an AENC channel.

[Syntax]

```
HI_S32 HI_MPI_AENC_ReleaseStream(AENC_CHN AeChn, const AUDIO_STREAM_S
*pstStream);
```

[Parameter]

Parameter	Description	Input/Output
AeChn	ID of an AENC channel Value range: [0, AENC_MAX_CHN_NUM)	Input
pstStream	Pointer to the obtained streams	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, hi_comm_aenc.h, mpi_aenc.h
- Library files: libmpi.a, lib_VoiceEngine.a, lib_aec.a

[Note]

- You are advised to release streams after use. If the streams are not released in time, encoding may be blocked.
- Streams must be released from the channel on which the streams are obtained. Modification of stream structure is not allowed. Otherwise, streams cannot be released, and the stream buffer is lost and even a program exception occurs.
- Streams can be released after an AENC channel is created. Otherwise, a code indicating failure is returned. If users attempt to destroy an AENC channel on which streams are being released, a code indicating failure is returned.

[Example]

See HI_MPI_AENC_CreateChn.

[See Also]

None

HI_MPI_AENC_GetFd

[Description]

Obtains the device FD corresponding to an AENC channel ID.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_AENC_GetFd(AENC_CHN AeChn)
```

[Parameter]

Parameter	Description	Input/Output
AeChn	ID of an AENC channel Value range: [0, AENC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
Positive value	Valid
Non-positive value	Invalid

[Requirement]

- Header files: hi_comm_aio.h, hi_comm_aenc.h, mpi_aenc.h
- Library file: libmpi.a

[Note]

None

HI_MPI_AENC_RegeisterEncoder

[Description]

Registers an encoder.

[Syntax]

```
HI_S32 HI_MPI_AENC_RegeisterEncoder(HI_S32 *ps32Handle, AENC_ENCODER_S
*pstEncoder)
```

[Parameter]

Parameter	Description	Input/Output
ps32Handle	Registration handle	Output
pstEncoder	Structure defining the encoder attributes	Input

[Return Value]

Return Value	Description
0	Success

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aenc.h, mpi_aenc.h
- Library file: libmpi.a

[Note]

- You can register an encoder with the AENC module by sending the encoder attribute structure to this module, and the registration handle is returned. You can deregister this encoder by using the registration handle.
- You can register a maximum of 20 encoders (including the five encoders LPCM, G711a, G711μ, G726, and ADPCM registered with the AENC module) with this module.
- For example, you are not allowed to register another G726 encoder if you have registered one.

[Example]

None

[See Also]

None

HI_MPI_AENC_UnRegisterEncoder

[Description]

Deregisters an encoder.

[Syntax]

```
HI_S32 HI_MPI_AENC_UnRegisterEncoder(HI_S32 s32Handle)
```

[Parameter]

Parameter	Description	Input/Output
s32Handle	Registration handle returned when an encoder is registered	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_aenc.h, mpi_aenc.h
- Library file: libmpi.a

[Note]

Typically, encoders do not need to be deregistered.

[Example]

None

[See Also]

None

HI_MPI_AENC_Save_File

[Description]

Enables the function of saving AENC files.

[Syntax]

```
HI_S32 HI_MPI_AENC_Save_File(AENC_CHN AeChn, AUDIO_SAVE_FILE_INFO_S
*pstSaveFileInfo)
```

[Parameter]

Parameter	Description	Input/Output
AeChn	ID of an AENC channel Value range: [0, AIO_MAX_CHN_NUM)	Input
pstSaveFileInfo	Pointer to the attribute structure of the function of saving AENC files	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, mpi_ao.h
- Library file: libmpi.a

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

This MPI is only used to dump AENC files in binding mode. After this MPI is called, three files in specified size are generated in the specified folder. Sin.pcm is the input frame, Rin.pcm is the reference frame, and Sou.pcm is the output frame.

[See Also]

None

9.3.4 ADEC

An ADEC module mainly provides the following functions: creating ADEC channels, transmitting audio frames, and obtaining decoded audio frame streams.

The ADEC module provides the following MPIs:

- HI_MPI_ADEC_CreateChn: Creates an ADEC channel.
- HI_MPI_ADEC_DestroyChn: Destroys an ADEC channel.
- HI_MPI_ADEC_SendStream: Sends audio streams to an ADEC channel.
- HI_MPI_ADEC_ClearChnBuf: Clears the current audio data buffer on an ADEC channel.
- HI_MPI_ADEC_RegeisterDecoder: Registers a decoder.
- HI_MPI_ADEC_UnRegisterDecoder: Deregisters a decoder.
- HI_MPI_ADEC_GetFrame: Obtains the decoded audio frame data.
- HI_MPI_ADEC_ReleaseFrame: Releases the buffer for releasing the decoded audio frame data.

HI_MPI_ADEC_CreateChn

[Description]

Creates an ADEC channel.

[Syntax]

```
HI_S32 HI_MPI_ADEC_CreateChn(ADEC_CHN AdChn, ADEC_CHN_ATTR_S *pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
AdChn	ID of an ADEC channel Value range: [0, ADEC_MAX_CHN_NUM)	Input
pstAttr	Pointer to the ADEC channel attributes	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: hi_comm_aio.h, hi_comm_adec.h, mpi_adec.h
- Library files: libmpi.a, lib_VoiceEngine.a, lib_aec.a

[Note]

- **enType** specifies a decoding protocol for the ADEC channel. Currently, G711, G726, and ADPCM are supported.
- Some attributes of the ADEC channel must be consistent with attributes of the AO device, for example, sampling rate and length of a frame (number of sampling points per frame).
- The buffer size is in unit of frame and its value range is [2, MAX_AUDIO_FRAME_NUM]. You are advised to set the size to 10 or greater. A small buffer may cause exceptions such as frame loss.
- This MPI can be called when the channel is not created or it is destroyed. If an ADEC channel is created, an error indicating that the channel is created is returned.

[Example]

```
HI_S32 s32ret;
ADEC_CHN_ATTR_S stAdecAttr;
ADEC_ATTR_ADPCM_S stAdpcm;
ADEC_CHN AdChn = 0;
AUDIO_STREAM_S stAudioStream;
AUDIO_DEV AoDev = 0;
AO_CHN AoChn = 0;
MPP_CHN_S stSrcChn;
MPP_CHN_S stDestChn;

stAdecAttr.enType = PT_ADPCMA;
stAdecAttr.u32BufSize = 8;
stAdecAttr.enMode = ADEC_MODE_STREAM;
stAdecAttr.pValue = &stAdpcm;
stAdpcm.enADPCMType = ADPCM_TYPE_DVI4;

/* create adec chn*/
s32ret = HI_MPI_ADEC_CreateChn(AdChn, &stAdecAttr);
if (s32ret)
{
    printf("create adec chn %d err:0x%x\n", AdChn,s32ret);
    return s32ret;
}


/* bind ADEC to AO channel*/
stSrcChn.enModId = HI_ID_ADEC;
stSrcChn.s32DevId = 0;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
stSrcChn.s32ChnId = AdChn;
stDestChn.enModId = HI_ID_AO;
stDestChn.s32DevId = AoDev;
stDestChn.s32ChnId = AoChn;

s32ret = HI_MPI_SYS_Bind(&stSrcChn, &stDestChn);
if (s32ret)
{
    printf("bind adec chn %d to ao(%d, %d) error:0x%x\n", AdChn, AoDev,
AoChn, s32ret);
    return s32ret;
}

/* get audio stream from network or file*/

/* send audio stream to adec chn */
s32ret = HI_MPI_ADEC_SendStream(AdChn, &stAudioStream);
if (s32ret)
{
    printf("send stream to adec fail\n");
    return s32ret;
}

/* destroy adec chn */
s32ret = HI_MPI_ADEC_DestroyChn(AdChn);
if (HI_SUCCESS != s32ret )
{
    return s32ret;
}
```

[See Also]

None

HI_MPI_ADEC_DestroyChn

[Description]

Destroys an ADEC channel.

[Syntax]

```
HI_S32 HI_MPI_ADEC_DestroyChn(ADEC_CHN AdChn);
```


Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
AdChn	ID of an ADEC channel Value range: [0, ADEC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, hi_comm_adec, mpi_adec.h
- Library files: libmpi.a, lib_VoiceEngine.a, lib_aec.a

[Note]

- If no channel is created, calling this MPI returns a code indicating success.
- A code indicating failure is returned if users attempt to destroy an ADEC channel on which streams are being obtained/released or frames are being sent.

[Example]

See HI_MPI_ADEC_CreateChn

[See Also]

None

HI_MPI_ADEC_SendStream

[Description]

Sends streams to an ADEC channel.

[Syntax]

```
HI_S32 HI_MPI_ADEC_SendStream(ADEC_CHN AdChn, const AUDIO_STREAM_S
*pstStream, HI_BOOL bBlock);
```

[Parameter]

Parameter	Description	Input/Output
AdChn	ID of an ADEC channel Value range: [0, ADEC_MAX_CHN_NUM)	Input
pstStream	Audio streams	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
bBlockFlag	Block flag Value range: HI_TRUE: block HI_FALSE: non-block	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, hi_comm_adec.h, mpi_adec.h
- Library files: libmpi.a, lib_VoiceEngine.a, lib_aec.a

[Note]

- Specify the decoding mode to Pack or Stream when creating an ADEC channel.
  - The Pack mode is used when users determine that a stream is exactly one frame, for example, a stream directly obtained from the AENC channel or the stream of a frame (the frame boundary is easily determined because the length of the voice encoding stream is fixed) in a file. This mode provides high decoding efficiency.
  - The Stream mode is used when users cannot determine whether a stream is one frame. In this mode, efficiency is low and delay may occur.
- The LPCM supports only the Pack mode. Other audio formats support the two decoding modes.
- Streams can be sent after an ADEC channel is created. Otherwise, a code indicating failure is returned. If users attempt to destroy an ADEC channel on which streams are being sent, a code indicating failure is returned.
- Streams can be sent in block or non-block mode.
- When streams are sent in block mode, if the buffer caching decoded audio frames is full, this MPI fails to be called. This MPI can be successfully called only after the decoded audio frames are released or the ADEC channel is destroyed.
- Make sure that the stream data on the ADEC channel is correct. Otherwise, the decoder may exit exceptionally.

[Example]

See HI_MPI_ADEC_CreateChn.

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_ADEC_ClearChnBuf

[Description]

Clears the current audio data buffer on an ADEC channel.

[Syntax]

```
HI_S32 HI_MPI_ADEC_ClearChnBuf(ADEC_CHN AdChn);
```

[Parameter]

Parameter	Description	Input/Output
AdChn	ID of an ADEC channel Value range: [0, ADEC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_aio.h, hi_comm_adec.h, mpi_adec.h
- Library files: libmpi.a, lib_VoiceEngine.a, lib_aec.a

[Note]

- This MPI can be called after an ADEC channel is created. Otherwise, a code indicating that the channel does not exist is returned.
- When this MPI is called, the Stream code is not recommended for clearing buffer data. In Stream mode, ensure that data transmitted to the decoder comprises a stream containing a complete frame after clearing the buffer. Otherwise, the decoder may operate exceptionally.
- No matter whether the data is decoded in Stream mode, users must ensure that transmitting data to be decoded must be synchronous with clearing the buffer.

[Example]

None

[See Also]

None

HI_MPI_ADEC_RegeisterDecoder

[Description]

Registers a decoder.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_ADEC_RegeisterDecoder(HI_S32 *ps32Handle, ADEC_DECODER_S
*pstDecoder);
```

[Parameter]

Parameter	Description	Input/Output
ps32Handle	Registration handle	Output
pstDecoder	Structure defining the encoder attributes	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_adec.h, mpi_adec.h
- Library file: libmpi.a

[Note]

- You can register a decoder with the ADEC module by sending the decoder attribute structure to this module, and the registration handle is returned. You can deregister this decoder by using the registration handle.
- You can register a maximum of 20 decoders (including the five decoders LPCM, G711a, G711μ, G726, and ADPCM registered with the ADEC module) with this module.
- For example, you are not allowed to register another G726 decoder if you have registered one.

[Example]

None

[See Also]

None

HI_MPI_ADEC_UnRegisterDecoder

[Description]

Deregisters a decoder.

[Syntax]

```
HI_S32 HI_MPI_ADEC_UnRegisterDecoder(HI_S32 s32Handle);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
s32Handle	Registration handle returned when a decoder is registered	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_adec.h, mpi_adec.h
- Library file: libmpi.a

[Note]

Typically, decoders do not need to be deregistered.

[Example]

None

[See Also]

None

HI_MPI_ADEC_GetFrame

[Description]

Obtains the decoded audio frame data.

[Syntax]

```
HI_S32 HI_MPI_ADEC_GetFrame(ADEC_CHN AdChn, AUDIO_FRAME_INFO_S
*pstFrmInfo, HI_BOOL bBlock);
```

[Parameter]

Parameter	Description	Input/Output
AdChn	ID of an ADEC channel	Input
pstFrmInfo	Pointer to the audio frame data structure	Output
bBlock	Whether the decoded audio frame data is obtained in block mode	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_adec.h, mpi_adec.h
- Library file: libmpi.a

[Note]

- This MPI must be called after an ADEC channel is created.
- When the decoded frame data is obtained by calling this MPI, it is recommended that streams be sent by frame
- If streams are sent by stream when the decoded frame data is obtained by calling this MPI, ensure that the decoded frame data is obtained in a timely manner. Otherwise, exceptions may occur.
- When the decoded frame data is obtained by calling this MPI, unbind the ADEC channels from the AO channels. Otherwise, the obtained frames are discontinuous.

[Example]

None

[See Also]

None

HI_MPI_ADEC_ReleaseFrame

[Description]

Releases the buffer for releasing the decoded audio frame data.

[Syntax]

```
HI_S32 HI_MPI_ADEC_ReleaseFrame(ADEC_CHN AdChn, AUDIO_FRAME_INFO_S
*pstFrmInfo);
```

[Parameter]

Parameter	Description	Input/Output
AdChn	ID of an ADEC channel	Input
pstFrmInfo	Pointer to the audio frame data structure	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. The return value is an error code. For details, see section 9.5 "Error Codes."

[Requirement]

- Header files: hi_comm_adec.h, mpi_adec.h
- Library file: libmpi.a

[Note]

- This MPI must be called after an ADEC channel is created.
- This MPI must work with HI_MPI_ADEC_GetFrame.

[Example]

None

[See Also]

None

9.3.5 Built-In Audio CODEC

The built-in audio CODEC supports the operations of the hardware device by using the ioctl function. You do not need to call cmds of the ioctl function. For the cmds that are not called, use the default configurations when loading modules. When the ioctl function is called, the registers of the audio CODEC are read or written.

9.3.5.1 Common Commands

The following describes common commands:

- ACODEC_SOFT_RESET_CTRL: Restores the audio CODEC to default settings.
- ACODEC_SET_I2S1_FS: Sets the sampling rate of the I^2S1 interface.
- ACODEC_SET_MIXER_MIC: Selects the input audio channel MICIN or LINEIN.
- ACODEC_SET_GAIN_MICL: Sets the analog gain of the audio-left input channel.
- ACODEC_SET_GAIN_MICR: Sets the analog gain of the audio-right input channel.
- ACODEC_SET_DACL_VOL: Sets the volume of the audio-left output channel.
- ACODEC_SET_DACR_VOL: Sets the volume of the audio-right output channel.
- ACODEC_SET_ADCL_VOL: Sets the volume of the audio-left input channel.
- ACODEC_SET_ADCR_VOL: Sets the volume of the audio-right input channel.
- ACODEC_SET_MICL_MUTE: Sets the mute of the audio-left input channel.
- ACODEC_SET_MICR_MUTE: Sets the mute of the audio-right input channel.
- ACODEC_SET_DACL_MUTE: Set the mute of the audio-left output channel.
- ACODEC_SET_DACR_MUTE: Sets the mute of the audio-right output channel.
- ACODEC_DAC_SOFT_MUTE: Controls the soft mute function of the DAC.
- ACODEC_DAC_SOFT_UNMUTE: Controls the soft unmute function of the DAC.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- ACODEC_GET_GAIN_MICL: Obtains the analog gain of the audio-left input channel.
- ACODEC_GET_GAIN_MICR: Obtains the analog gain of the audio-right input channel.
- ACODEC_GET_DACL_VOL: Obtains the volume of the audio-left output channel.
- ACODEC_GET_DACR_VOL: Obtains the volume of the audio-right output channel.
- ACODEC_GET_ADCL_VOL: Obtains the volume of the audio-left input channel.
- ACODEC_GET_ADCR_VOL: Obtains the volume of the audio-right input channel.
- ACODEC_SET_PD_DACL: Powers off the audio-left output channel.
- ACODEC_SET_PD_DACR: Powers off the audio-right output channel.
- ACODEC_SET_PD_ADCL: Powers off the audio-left input channel.
- ACODEC_SET_PD_ADCR: Powers off the audio-right input channel.
- ACODEC_ENABLE_BOOSTL: Enables the analog boost gain of the audio-left channel.
- ACODEC_ENABLE_BOOSTR: Enables the analog boost gain of the audio-right channel.
- ACODEC_SET_INPUT_VOL: Sets the input volume of the audio CODEC.
- ACODEC_GET_INPUT_VOL: Obtains the input volume of the audio CODEC.
- ACODEC_SET_OUTPUT_VOL: Sets the output volume of the audio CODEC.
- ACODEC_GET_OUTPUT_VOL: Obtains the output volume of the audio CODEC.

ACODEC_SOFT_RESET_CTRL

[Description]

Restores the audio CODEC to default settings.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SOFT_RESET_CTRL);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SOFT_RESET_CTRL	ioctl number	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

Call this MPI only after AI and AO devices are disabled.

[Example]

```
if (ioctl(s32Acodec_Fd, ACODEC_SOFT_RESET_CTRL))
{
    printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_I2S1_FS

[Description]

Sets the sampling rate of the I^2S1 interface.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_I2S1_FS,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_I2S1_FS	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_FS_E i2s_fs_sel;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
i2s_fs_sel = ACODEC_FS_48000;
if (ioctl(s32Acodec_Fd, ACODEC_SET_I2S1_FS, &i2s_fs_sel))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_MIXER_MIC

[Description]

Selects the input audio channel MICIN or LINEIN.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_MIXER_MIC,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_MIXER_MIC	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

You can only select LINEIN. If MICIN is selected, there is no input signal. If the inputs are passive audio signals, the analog gain needs to be increased.

[Example]

```
ACODEC_MIXER_E mixer_mic_ctrl;
mixer_mic_ctrl = ACODEC_MIXER_LINEIN;
if (ioctl(s32Acodec_Fd, ACODEC_SET_MIXER_MIC, &mixer_mic_ctrl))
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
{
    printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_GAIN_MICL

[Description]

Controls the analog gain of the audio-left input channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_GAIN_MICL,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_GAIN_MICL	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

If the inputs are passive audio signals, the analog gain needs to be increased.

[Example]

```
unsigned int gain_mac;
gain_mic = 0x18;
if (ioctl(s32Acodec_Fd,  ACODEC_SET_GAIN_MICL, &gain_mic))
{
    printf("ioctl err!\n");
}
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

None

ACODEC_SET_GAIN_MICR

[Description]

Sets the analog gain of the audio-right input channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_GAIN_MICR,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_GAIN_MICR	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

If the inputs are passive audio signals, the analog gain needs to be increased.

[Example]

```
unsigned int gain_mic;
gain_mic = 0x18;
if (ioctl(s32Acodec_Fd,  ACODEC_SET_GAIN_MICR, &gain_mic))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

ACODEC_SET_DACL_VOL

[Description]

Set the volume of the audio-left output channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_DACL_VOL,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_DACL_VOL	ioctl number	Input
arg	Pointer to ACODEC_VOL_CTRL	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_DACL_VOL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

ACODEC_SET_DACR_VOL

[Description]

Sets the output volume of the audio-right channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_DACR_VOL,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_DACL_VOL	ioctl number	Input
arg	Pointer to ACODEC_VOL_CTRL	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_DACR_VOL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

ACODEC_SET_ADCL_VOL

[Description]

Sets the volume of the audio-left input channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_ADCL_VOL,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_ADCL_VOL	ioctl number	Input
arg	Pointer to ACODEC_VOL_CTRL	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_ADCL_VOL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

ACODEC_SET_ADCR_VOL

[Description]

Sets the volume of the audio-right input channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_ADCR_VOL,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_ADCR_VOL	ioctl number	Input
arg	Pointer to ACODEC_VOL_CTRL	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_ADCR_VOL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

ACODEC_SET_MICL_MUTE

[Description]

Sets the mute of the audio-left input channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_MICL_MUTE,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_MICL_MUTE	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int mute_ctrl;
mute_ctrl = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SET_MICL_MUTE, &mute_ctrl))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_MICR_MUTE

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Sets the mute of the audio-right input channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_MICR_MUTE,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_MICR_MUTE	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int mute_ctrl;
ute_ctrl = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SET_MICR_MUTE, &mute_ctrl))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_DACL_MUTE

[Description]

Sets the mute of the audio-left output channel.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
int ioctl (int fd,
           ACODEC_SET_DACL_MUTE,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_DACL_MUTE	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int mute_ctrl;
mute_ctrl = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SET_DACL_MUTE, &mute_ctrl))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_DACR_MUTE

[Description]

Sets the mute of the audio-right output channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_DACR_MUTE,
           unsigned int *arg);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_DACR_MUTE	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int  mute_ctrl;
mute_ctrl = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SET_DACR_MUTE, &mute_ctrl))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_DAC_SOFT_MUTE

[Description]

Controls the soft mute function of the DAC.

[Syntax]

```
int ioctl (int fd,
           ACODEC_DAC_SOFT_MUTE,
           unsigned int  *arg);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_DAC_SOFT_MUTE	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int  soft_mute_ctrl;
soft_mute_ctrl = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_DAC_SOFT_MUTE, &soft_mute_ctrl))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_DAC_SOFT_UNMUTE

[Description]

Controls the soft unmute function of the DAC.

[Syntax]

```
int ioctl (int fd,
           ACODEC_DAC_SOFT_UNMUTE,
           unsigned int *arg);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_DAC_SOFT_UNMUTE	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int  soft_mute_ctrl;
soft_nute_ctrl = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_DAC_SOFT_UNMUTE, &soft_mute_ctrl))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_GET_GAIN_MICL

[Description]

Obtains the analog gain of the audio-left input channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_GET_GAIN_MICL,
           unsigned int  *arg);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
Fd	Descriptor of the audio CODEC file	Input
ACODEC_GET_GAIN_MICL	ioctl number	Input
arg	Unsigned integer pointer	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int gain_mic;
if (ioctl(s32Acodec_Fd,  ACODEC_GET_GAIN_MICL, &gain_mic))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_GET_GAIN_MICR

[Description]

Obtains the analog gain of the audio-right input channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_GET_GAIN_MICR,
           unsigned int  *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
ACODEC_GET_GAIN_MICR	ioctl number	Input
arg	Unsigned integer pointer	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int gain_mic;
if (ioctl(s32Acodec_Fd,  ACODEC_GET_GAIN_MICR, &gain_mic))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_GET_DACL_VOL

[Description]

Obtains the volume of the audio-left output channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_GET_DACL_VOL,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_GET_DACL_VOL	ioctl number	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
arg	Pointer to ACODEC_VOL_CTRL	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_VOL_CTRL vol_ctrl;
if (ioctl(s32Acodec_Fd, ACODEC_GET_DACL_VOL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_GET_DACR_VOL

[Description]

Obtains the volume of the audio-right output channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_GET_DACR_VOL,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_GET_DACL_VOL	ioctl number	Input
arg	Pointer to ACODEC_VOL_CTRL	Output

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_VOD_CTRL vol_ctrl;
if (ioctl(s32Acodec_Fd, ACODEC_GET_DACR_VOL, &vol_ctrl))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_GET_ADCL_VOL

[Description]

Obtains the volume of the audio-left input channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_GET_ADCL_VOL,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_GET_ADCL_VOL	ioctl number	Input
arg	Pointer to ACODEC_VOL_CTRL	Output

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_VOL_CTRL vol_ctrl;
if (ioctl(s32Acodec_Fd, ACODEC_GET_ADCL_VOL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_GET_ADCR_VOL

[Description]

Obtains the volume of the audio-right input channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_GET ADCR_VOL,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_GET_ADCR_VOL	ioctl number	Input
arg	Pointer to ACODEC_CTRL	Output

[Return Value]

Return Value	Description
0	Success

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_VOL_CTRL vol_ctrl;
if (ioctl(s32Acodec_Fd, ACODEC_GET_ADCR_VOL, &vol_ctrl))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_PD_DACL

[Description]

Powers off the audio-left output channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_PD_DACL,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_PD_DACL	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int pd_ctrl;
pd_ctrl = 0x01;
if (ioctl(s32Acodec_Fd, ACODEC_SET_PD_DACL, &pd_ctrl))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_PD_DACR

[Description]

Powers off the audio-right output channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_PD_DACR,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_PD_DACR	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

acodec.h

[Note]

None

[Example]

```
unsigned int pd_ctrl;
pd_ctrl = 0x01
if (ioctl(s32Acodec_Fd, ACODEC_SET_PD_DACR, &pd_ctrl))
{
printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_PD_ADCL

[Description]

Powers off the audio-left input channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_PD_ADCL,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_PD_ADCL	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[Example]

```
unsigned int pd_ctrl;
pd_ctrl = 0x01;
if (ioctl(s32Acodec_Fd, ACODEC_SET_PD_ADCL, &pd_ctrl))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_PD_ADCR

[Description]

Powers off the audio-right input channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_PD_ADCR,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_PD_ADCR	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
unsigned int pd_ctrl;
pd_ctl = 0x01;
if (ioctl(s32Acodec_Fd, ACODEC_SET_PD_ADCR, &pd_ctrl))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_ENABLE_BOOSTL

[Description]

Enables the analog boost gain of the audio-left channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_ENABLE_BOOST,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	File descriptor of the audio CODEC	Input
ACODEC_ENABLE_BOOST	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

After the boost is enabled, the analog gain increases by 26 dB. ACODEC_SET_GAIN_MICL and ACODEC_SET_GAIN_MICR can be used for fine adjustment.

[Example]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

None

ACODEC_ENABLE_BOOSTR

[Description]

Enables the analog boost gain of the audio-right channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_ENABLE_BOOST,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	File descriptor of the audio CODEC	Input
ACODEC_ENABLE_BOOST	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

After the boost is enabled, the analog gain increases by 26 dB. ACODEC_SET_GAIN_MICL and ACODEC_SET_GAIN_MICR can be used for fine adjustment.

[Example]

None

[See Also]

None

ACODEC_SET_INPUT_VOL

[Description]

Sets the input volume of the audio CODEC.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
int ioctl (int fd, ACODEC_SET_INPUT_VOL,
int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	File descriptor of the audio CODEC	Input
ACODEC_SET_INPUT_VOL	ioctl number	Input
arg	Pointer to the volume (in dB)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
int db;
db = 0x30;
if (ioctl(s32Acodec_Fd, ACODEC_SET_INPUT_VOL, &db))
{
   printf("ioctl err!\n");
}
```

[See Also]

ACODEC_GET_INPUT_VOL

ACODEC_GET_INPUT_VOL

[Description]

Obtains the input volume of the audio CODEC.

[Syntax]

```
int ioctl (int fd,
           ACODEC_GET_INPUT_VOL,
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
            int *db);
```

[Parameter]

Parameter	Description	Input/Output
fd	File descriptor of the audio CODEC	Input
ACODEC_GET_INPUT_VOL	ioctl number	Input
arg	Pointer to the volume (in dB)	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
int *db;
if (ioctl(s32Acodec_Fd, ACODEC_GET_INPUT_VOL, db))
{
   printf("ioctl err!\n");
}
```

[See Also]

ACODEC_SET_INPUT_VOL

ACODEC_SET_OUTPUT_VOL

[Description]

Sets the output volume of the audio CODEC.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_OUTPUT_VOL,
           int *arg);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
fd	File descriptor of the audio CODEC	Input
ACODEC_SET_OUTPUT_VOL	ioctl number	Input
arg	Pointer to the volume (in dB)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
int db;
db = 0x30;
if (ioctl(s32Acodec_Fd, ACODEC_SET_OUTPUT_VOL, &db))
{
   printf("ioctl err!\n");
}
```

[See Also]

ACODEC_GET_OUTPUT_VOL

ACODEC_GET_OUTPUT_VOL

[Description]

Obtains the output volume of the audio CODEC.

[Syntax]

```
int ioctl (int fd,
           ACODEC_GET_OUTPUT_VOL,
           int *db);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
fd	File descriptor of the audio CODEC	Input
ACODEC_GET_OUTPUT_VOL	ioctl number	Input
arg	Pointer to the volume (in dB)	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
int *db;
if (ioctl(s32Acodec_Fd, ACODEC_GET_OUTPUT_VOL, db))
{
   printf("ioctl err!\n");
}
```

[See Also]

ACODEC_SET_OUTPUT_VOL

9.3.5.2 Extended Commands

The following are extended commands:

- ACODEC_SEL_DAC_CLK: Sets the clock edges of the DAC and ADC to same or opposite.
- ACODEC_SEL_ANA_MCLK: Sets the clock edges of the analog part and digital part to same or opposite.
- ACODEC_DACL_SEL_TRACK: Sets the DACL for selecting the audio channel.
- ACODEC_DACR_SEL_TRACK: Sets the DACR for selecting the audio channel.
- ACODEC_ADCL_SEL_TRACK: Sets the ADCL for selecting the audio channel.
- ACODEC_ADCR_SEL_TRACK: Sets the ADCR for selecting the audio channel.
- ACODEC_SET_DAC_DE_EMPHASIS: Controls the de-emphasis filtering function of the DAC.
- ACODEC_SET_ADC_HP_FILTER: Controls the high-pass filtering function of the ADC.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- ACODEC_DAC_POP_FREE: Deletes the POP noise of the DAC.
- ACODEC_DAC_SOFT_MUTE_RATE: Controls the rate for soft mute of the DAC.
- ACODEC_DAC_SEL_I2S: Selects the I^2S interface for the DAC.
- ACODEC_ADC_SEL_I2S: Selects the I^2S interface for the ADC.
- ACODEC_SET_I2S1_DATAWIDTH: Sets the data width of the I^2S1 interface
- ACODEC_SET_I2S2_DATAWIDTH: Sets the data width of the I^2S2 interface
- ACODEC_SET_I2S2_FS: Sets the sampling rate of the I^2S2 interface.
- ACODEC_SET_DACR2DACL_VOL: Mixes the volume of the DACR with that of the DACL.
- ACODEC_SET_DACL2DACR_VOL: Mixes the volume of the DACL with that of the DACR.
- ACODEC_SET_ADCL2DACL_VOL: Mixes the volume of the ADCL with that of the DACL.
- ACODEC_SET_ADCR2DACL_VOL: Mixes the volume of the ADCR with that of the DACL.
- ACODEC_SET_ADCL2DACR_VOL: Mixes the volume of the DACR with that of the DACL.
- ACODEC_SET_ADCR2DACR_VOL: Mixes the volume of the ADCR with that of the DACR.

ACODEC_SEL_DAC_CLK

[Description]

Sets the clock edges of the DAC and ADC to same or opposite.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SEL_DAC_CLK,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SEL_DAC_CLK	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int clk_sel;
clk_sel = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SEL_DAC_CLK, &clk_sel))
{
printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SEL_ANA_MCLK

[Description]

Sets the clock edges of the analog part and digital part to same or opposite.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SEL_ANA_MCLK,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SEL_ANA_MCLK	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

None

[Example]

```
unsigned int clk_sel;
clk_sel = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SEL_ANA_MCLK, &clk_sel))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_DACL_SEL_TRACK

[Description]

Sets the DACL for selecting the audio channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_DACL_SEL_TRACK,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_DACL_SEL_TRACK	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Example]

```
unsigned int track_select;
track_select = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_DACL_SEL_TRACK, &track_select))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_DACR_SEL_TRACK

[Description]

Sets the DACR for selecting the audio channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_DACR_SEL_TRACK,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_DACR_SEL_TRACK	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int track_select;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
track_select = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_DACR_SEL_TRACK, &track_select)
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_ADCL_SEL_TRACK

[Description]

Sets the ADCL for selecting the audio channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_ADCL_SEL_TRACK,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_ADCL_SEL_TRACK	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int track_select;
track_select = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_ADCL_SEL_TRACK, &track_select))
{
```


Copyright © HiSilicon Technologies Co., Ltd

```
    printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_ADCR_SEL_TRACK

[Description]

Sets the ADCR for selecting the audio channel.

[Syntax]

```
int ioctl (int fd,
           ACODEC_ADCR_SEL_TRACK,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_ADCR_SEL_TRACK	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int track_select;
track_select = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_ADCR_SEL_TRACK, &track_select))
{
    printf("ioctl err!\n");
}
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

None

ACODEC_SET_DAC_DE_EMPHASIS

[Description]

Control the de-emphasis filtering function of the DAC.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_DAC_DE_EMPHASIS,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_DAC_DE_EMPHASIS	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int dac_deemphasis;
dac_deemphasis = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SET_DAC_DE_EMPHASIS, &dac_deemphasis))
{
    printf("ioctl err!\n");
}
```

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

ACODEC_SET_ADC_HP_FILTER

[Description]

Controls the high-pass filtering function of the ADC.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_ADC_HP_FILTER,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_ADC_HP_FILTER	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int adc_hpf;
adc_hpf = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SET_ADC_HP_FILTER, &adc_hpf))
{
    printf("ioctl err!\n");
}
```

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

ACODEC_DAC_POP_FREE

[Description]

Deletes the POP noise of the DAC.

[Syntax]

```
int ioctl (int fd,
           ACODEC_DAC_POP_FREE,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_DAC_POP_FREE	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int dac_pop_free;
dac_pop_free = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_DAC_POP_FREE, &dac_pop_free))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_DAC_SOFT_MUTE_RATE

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Controls the rate for soft mute of the DAC.

[Syntax]

```
int ioctl (int fd,
           ACODEC_DAC_POP_FREE,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_DAC_SOFT_MUTE_RATE	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int soft_mute_rate;
soft_mute_rate = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_DAC_SOFT_MUTE_RATE, &soft_mute_rate))
{
printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_DAC_SEL_I2S

[Description]

Selects the I^2S interface for the DAC.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
int ioctl (int fd,
           ACODEC_DAC_SEL_I2S,
           unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_DAC_SEL_I2S	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int i2s_sel;
i2s_sel = 0x0;
if (ioctl(s32Acodec_Fd, ACODEC_DAC_SOFT_MUTE_RATE, &i2s_sel))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_ADC_SEL_I2S

[Description]

Selects the I^2S interface for the ADC.

[Syntax]

```
int ioctl (int fd,
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
            ACODEC_ADC_SEL_I2S,
            unsigned int *arg);
```

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_ADC_SEL_I2S	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int i2s_sel;
i2s_sel = 0x0;
if (ioctl(s32Acodec_Fd, ACODEC_ADC_SEL_I2S, &i2s_sel))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_I2S1_DATAWIDTH

[Description]

Sets the data width of the I^2S1 interface.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_I2S1_DATAWIDTH,
           unsigned int *arg);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_I2S1_DATAWIDTH	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int i2s_datawidth;
i2s_datawidth = 0x0;
if (ioctl(s32Acodec_Fd, ACODEC_SET_I2S1_DATAWIDTH, &i2s_datawidth))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_I2S2_DATAWIDTH

[Description]

Sets the data width of the I^2S2 interface.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_I2S2_DATAWIDTH,
           unsigned int *arg);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_I2S2_DATAWIDTH	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
unsigned int i2s_datawidth;
i2s_datawidth = 0x0;
if (ioctl(s32Acodec_Fd, ACODEC_SET_I2S2_DATAWIDTH, &i2s_datawidth))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_I2S2_FS

[Description]

Sets the sampling rate of the I^2S2 interface.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_I2S2_FS,
           unsigned int *arg);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_I2S2_FS	ioctl number	Input
arg	Unsigned integer pointer	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_FS_E i2s_fs_sel;
i2s_fs_sel = ACODEC_FS_48000;
if (ioctl(s32Acodec_Fd, ACODEC_SET_I2S2_FS, &i2s_fs_sel))
{
   printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_DACR2DACL_VOL

[Description]

Mixes the volume of the DACR with that of the DACL.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_DACR2DACL_VOL,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_DACR2DACL_VOL	ioctl number	Input
arg	Pointer to ACODEC_VOL_CTRL	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_DACR2DACL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_DACL2DACR_VOL

[Description]

Mixes the volume of the DACL with that of the DACR.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_DACL2DACR_VOL,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_DACL2DACR_VOL	ioctl number	Input
arg	Pointer to ACODEC_VOL_CTRL	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_DACL2DACR, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_ADCL2DACL_VOL

[Description]

Mixes the volume of the ADCL with that of the DACL.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_ADCL2DACL_VOL,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_ADCL2DACL_VOL	ioctl number	Input
arg	Pointer to ACODEC_VOL_CTRL	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_ADCL2DACL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_ADCR2DACL_VOL

[Description]

Mixes the volume of the ADCR with that of the DACL.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_ADCR2DACL_VOL,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_ADCR2DACL_VOL	ioctl number	Input
arg	Pointer to ACODEC_VOL_CTRL	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_ADCR2DACL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_ADCL2DACR_VOL

[Description]

Mixes the volume of the DACR with that of the DACL.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_ADCL2DACR_VOL,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_ADCL2DACR_VOL	ioctl number	Input
arg	Pointer to ACODEC_VOL_CTRL	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_ADCL2DACR, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

[See Also]

None

ACODEC_SET_ADCR2DACR_VOL

[Description]

Mixes the volume of the ADCR with that of the DACR.

[Syntax]

```
int ioctl (int fd,
           ACODEC_SET_ADCR2DACR_VOL,
           ACODEC_VOL_CTRL *arg);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
fd	Descriptor of the audio CODEC file	Input
ACODEC_SET_ADCR2DACR_VOL	ioctl number	Input
arg	Pointer to ACODEC_VOL_CTRL	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure

[Requirement]

acodec.h

[Note]

None

[Example]

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_ADCR2DACR, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

[See Also]

None

9.4 Data Types

9.4.1 AI and AO

The AI and AO data types are defined as follows:

- SIO_MAX_NUM: Defines the maximum number of AI and AO devices.
- AIO_MAX_CHN_NUM: Defines the maximum number of channels for AI and AO devices.
- AUDIO_SAMPLE_RATE_E: Defines an audio sampling rate.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- AUDIO_BIT_WIDTH_E: Defines the audio sampling precision.
- AIO_MODE_E: Defines the AI/AO working mode.
- AUDIO_SOUND_MODE_E: Defines the audio sound-channel mode.
- AIO_ATTR_S: Defines the attribute structure of an AI/AO device.
- AUDIO_FRAME_S: Defines the structure of an audio frame.
- AEC_FRAME_S: Defines the structure of a reference echo cancellation frame.
- AI_CHN_PARAM_S: Defines the channel parameters.
- AUDIO_STREAM_S: Defines the structure of an audio stream.
- AUDIO_RESAMPLE_TYPE_E: Defines the type of audio resampling.
- AUDIO_RESAMPLE_ATTR_S: Defines the structure of the audio resampling attribute.
- AUDIO_RESAMPLE_ATTR_EX_S: Defines the structure of the audio advanced resampling attribute.
- AO_CHN_STATE_S: Defines the status of the data buffer on the AO channel.
- G726_BPS_E: Defines the bit rate for the G.726 encoding/decoding protocol.
- ADPCM_TYPE_E: Defines the type of the ADPCM encoding/decoding protocol.
- AUDIO_AEC_MODE_E: Defines the audio echo cancellation mode.
- AI_AEC_CONFIG_S: Defines the configuration information about audio echo cancellation.
- AI_ANR_CONFIG_S: Defines the configuration information about audio noise reduction.
- AI_ALC_CONFIG_S: Defines the configuration information about audio automatic level control.
- AI_VQE_CONFIG_S: Defines the structure of VQE attributes.
- AUDIO_HPF_FREQ_E: Defines the cut-off frequency of AI high-pass filtering.
- AI_HPF_CONFIG_S: Defines the high-pass filtering parameter.
- AI_HPF_ATTR_S: Defines the configuration information about AI high-pass filtering.
- AUDIO_SAVE_FILE_INFO_S: Defines the configuration information about the audio saving file function.

SIO_MAX_NUM

[Description]

Defines the maximum number of AI and AO devices.

[Syntax]

For the Hi3531:

```
#define SIO_MAX_NUM        6
```

For the Hi3532:

```
#define SIO_MAX_NUM        5
```

For the Hi3521/Hi3520A:

```
#define SIO_MAX_NUM        4
```

For the Hi3518/Hi3516C:

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
#define SIO_MAX_NUM       1
```

For the Hi3520D/Hi3515A/Hi3515C:

```
#define SIO_MAX_NUM       2
```

[Note]

None

[See Also]

None

AIO_MAX_CHN_NUM

[Description]

Defines the maximum number of channels for AI and AO devices.

[Syntax]

For the Hi3531/Hi3532:

```
#define AIO_MAX_CHN_NUM     16
```

For the Hi3521/Hi3520A:

```
#define AIO_MAX_CHN_NUM     16
```

For the Hi3518/Hi3516C:

```
#define AIO_MAX_CHN_NUM       2
```

For the Hi3520D/Hi3515A/Hi3515C:

```
#define AIO_MAX_CHN_NUM     16
```

[Note]

None

[See Also]

None

AUDIO_SAMPLE_RATE_E

[Description]

Defines an audio sampling rate.

[Syntax]

```
typedef enum hiAUDIO_SAMPLE_RATE_E
{
  AUDIO_SAMPLE_RATE_8000    =8000,      /* 8kHz sampling rate */
  AUDIO_SAMPLE_RATE_12000   =12000,     /* 12kHz sampling rate */
  AUDIO_SAMPLE_RATE_11025   =11025,     /* 11.025kHz sampling rate */
  AUDIO_SAMPLE_RATE_16000   =16000,     /* 16kHz sampling rate */
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
  AUDIO_SAMPLE_RATE_22050   =22050,     /* 22.050kHz sampling rate */
  AUDIO_SAMPLE_RATE_24000   =24000,     /* 24kHz sampling rate */
  AUDIO_SAMPLE_RATE_32000   =32000,     /* 32kHz sampling rate */
  AUDIO_SAMPLE_RATE_44100   =44100,     /* 44.1kHz sampling rate */
  AUDIO_SAMPLE_RATE_48000   =48000,     /* 48kHz sampling rate */
}AUDIO_SAMPLE_RATE_E;
```

[Member]

Member	Description
AUDIO_SAMPLE_RATE_8000	Sampling rate of 8 kHz
AUDIO_SAMPLE_RATE_12000	Sampling rate of 12 kHz
AUDIO_SAMPLE_RATE_11025	Sampling rate of 11.025 kHz
AUDIO_SAMPLE_RATE_16000	Sampling rate of 16 kHz
AUDIO_SAMPLE_RATE_22050	Sampling rate of 22.050 kHz
AUDIO_SAMPLE_RATE_24000	Sampling rate of 24 kHz
AUDIO_SAMPLE_RATE_32000	Sampling rate of 32 kHz
AUDIO_SAMPLE_RATE_44100	Sampling rate of 44.1 kHz
AUDIO_SAMPLE_RATE_48000	Sampling rate of 48 kHz

[Note]

The value is not enumerated from 0. Instead, the value is consistent with the actual sampling rate.

[See Also]

AIO_ATTR_S

AUDIO_BIT_WIDTH_E

[Description]

Defines the audio sampling precision.

[Syntax]

```
typedef enum hiAUDIO_BIT_WIDTH_E
{
    AUDIO_BIT_WIDTH_8   =0,     /* 8bit/sample */
    AUDIO_BIT_WIDTH_16  =1,     /* 16bit/sample */
    AUDIO_BIT_WIDTH_32  =2,     /* 32bit/sample */
    AUDIO_BIT_WIDTH_BUTT,
}AUDIO_BIT_WIDTH_E;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Member]

Member	Description
AUDIO_BIT_WIDTH_8	Sampling precision of 8 bits
AUDIO_BIT_WIDTH_16	Sampling precision of 16 bits
AUDIO_BIT_WIDTH_32	Sampling precision of 32 bits

[Note]

- Currently, 32-bit sampling precision is not supported.
- The Hi3518/Hi3516C supports only 16-bit sampling precision.

[See Also]

AIO_ATTR_S

AIO_MODE_E

[Description]

Defines the AI/AO working mode.

[Syntax]

```
typedef enum hiAIO_MODE_E
{
    AIO_MODE_I2S_MASTER = 0,        /* I2S master mode */
    AIO_MODE_I2S_SLAVE  = 1,        /* I2S slave mode */
    AIO_MODE_PCM_SLAVE_STD,         /* SIO PCM slave standard mode */
    AIO_MODE_PCM_SLAVE_NSTD,        /* SIO PCM slave non-standard mode */
    AIO_MODE_PCM_MASTER_STD,       /* SIO PCM master standard mode */
    AIO_MODE_PCM_MASTER_NSTD,      /* SIO PCM master non-standard mode */
    AIO_MODE_BUTT
}AIO_MODE_E;
```

[Member]

Member	Description
AIO_MODE_I2S_MASTER	Master I^2S mode
AIO_MODE_I2S_SLAVE	Slave I^2S mode
AIO_MODE_PCM_SLAVE_STD	Slave PCM mode (standard protocol)
AIO_MODE_PCM_SLAVE_NSTD	Slave PCM mode (customized protocol)
AIO_MODE_PCM_MASTER_STD	Master PCM mode (standard protocol)
AIO_MODE_PCM_MASTER_NSTD	Master PCM mode (customized protocol)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

- For the Hi3531 or Hi3532

  When SIO0 to SIO4 work in master mode, only the frame sync clock and bit stream clock for timing synchronization are provided. That is, no MCLK is provided. In this case, the audio CODEC uses the external crystal oscillator clock as the working clock. As a result, the sound may be distorted. Therefore, the slave mode is recommended for SIO0 to SIO4. SIO5 must work in master I^2S mode.

- For the Hi3521 or Hi3520A

  When SIO0 to SIO2 work in master mode, only the frame sync clock and bit stream clock for timing synchronization are provided. That is, no MCLK is provided. In this case, the audio CODEC uses the external crystal oscillator clock as the working clock. As a result, the sound may be distorted. Therefore, the slave mode is recommended for SIO0 to SIO2. SIO3 must work in master I^2S mode.

- For the Hi3518/Hi3516C

  The Hi3518/Hi3516C uses the internal audio CODEC and master I^2S mode.

- For the Hi3520D/Hi3515A/Hi3515C

  The Hi3520D/Hi3515A/Hi3515C uses the AIO interface as the audio input/output interface. When AiDev0 or AoDev0 is in slave mode, AiDev0 does not share the configurations of the frame sync clocks and bit stream clocks of AoDev0. AoDev1 is used to connect the HDMI and must work in master I^2S mode.

[See Also]

AIO_ATTR_S

AUDIO_SOUND_MODE_E

[Description]

Defines the audio sound-channel mode.

[Syntax]

```
typedef enum hiAIO_SOUND_MODE_E
{
    AUDIO_SOUND_MODE_MONO   =0,     /*Mono*/
    AUDIO_SOUND_MODE_STEREO =1,     /*Stereo*/
    AUDIO_SOUND_MODE_BUTT
}AUDIO_SOUND_MODE_E;
```

[Member]

Member	Description
AUDIO_SOUND_MODE_MONO	Mono
AUDIO_SOUND_MODE_STEREO	Stereo

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

In stereo mode, the channels whose channel IDs are smaller than half of the number of channels are audio-left channels, and the other channels are audio-right channels. You need to operate only audio-left channels, because the corresponding audio-right channels are automatically operated in the SDK.

[See Also]

AIO_ATTR_S

AIO_ATTR_S

[Description]

Defines the attribute structure of an AI/AO device.

[Syntax]

```
typedef struct hiAIO_ATTR_S
{
    AUDIO_SAMPLE_RATE_E enSamplerate;       /*Sampling rate*/
    AUDIO_BIT_WIDTH_E   enBitwidth;         /*Bit width*/
    AIO_MODE_E          enWorkmode;         /*Master or slave mode*/
    AUDIO_SOUND_MODE_E  enSoundmode;        /*Mono or stereo*/
    HI_U32              u32EXFlag;          /*Expand 8 bits to 16bits */
    HI_U32              u32FrmNum;          /*Frame num in buffer*/
    HI_U32              u32PtNumPerFrm;     /*Number of samples*/
    HI_U32              u32ChnCnt;
    HI_U32              u32ClkSel;
}AIO_ATTR_S;
```

[Member]

Member	Description
enSamplerate	Audio sampling rate (in slave mode, this parameter is invalid) Static attribute
enBitwidth	Audio sampling precision (in slave mode, this parameter must match the sampling precision of the audio AD/DA) Static attribute
enWorkmode	Working mode of an AI/AO device Static attribute
enSoundmode	Audio sound-channel mode Static attribute
u32EXFlag	8-bit to 16-bit extension flag. This parameter is valid only when the AI sampling precision is 8 bits. The values are as follows: • 0: not extend. • 1: extend.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	Static attribute
u32FrmNum	Number of buffered frames Value range: [2, MAX_AUDIO_FRAME_NUM] Static attribute
u32PtNumPerFrm	Number of sampling points per frame Value range: In G711, G726, and ADPCM_DVI4, the value is 80, 160, 240, 320, or 480; in ADPCM_IMA, the value is 81, 161, 241, 321, or 481. Static attribute
u32ChnCnt	Number of channels Value: 2, 4, 8, or 16
u32ClkSel	Whether to multiplex the TX clock and the RX clock of SIO4. Other SIO interfaces can be ignored. The values are as follows: • 0: not multiplex • 1: multiplex

[Note]

None

[See Also]

HI_MPI_AI_SetPubAttr

AUDIO_FRAME_S

[Description]

Defines the structure of an audio frame.

[Syntax]

```
typedef struct hiAUDIO_FRAME_S
{
    AUDIO_BIT_WIDTH_E   enBitwidth;  /*Audio frame bit width*/
    AUDIO_SOUND_MODE_E  enSoundmode; /*Audio frame mono or stereo mode*/
    HI_VOID             *pVirAddr[2];
    HI_U32              u32PhyAddr[2];
    HI_U64              u64TimeStamp;/*audio frame timestamp*/
    HI_U32              u32Seq;      /*audio frame seq*/
    HI_U32              u32Len;      /*data length per channel in frame*/
    HI_U32              u32PoolId[2];
}AUDIO_FRAME_S;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Member]

Member	Description
enBitwidth	Audio sampling precision
enSoundmode	Audio sound-channel mode
pVirAddr[2]	Virtual address for an audio frame
u32PhyAddr[2]	Physical address for an audio frame
u64TimeStamp	Time stamp of an audio frame, in μs
u32Seq	Sequence number of an audio frame
u32Len	Length of an audio frame, in byte
u32PoolId[2]	ID of the audio frame buffer

[Note]

- u32Len (length of an audio frame) indicates the length of data on a mono channel.
- Mono data is directly stored, for which the sampling points are specified by ptnum and the length of bytes is specified by len. Stereo data is stored in the audio-left channel and audio-right channel in sequence, for which the sampling points are specified by ptnum and the length of bytes is specified by len.

[See Also]

None

AEC_FRAME_S

[Description]

Defines the structure of a reference echo cancellation frame.

[Syntax]

```
typedef struct hiAEC_FRAME_S
{
    AUDIO_FRAME_S   stRefFrame;     /* aec reference audio frame */
    HI_BOOL         bValid;         /* whether frame is valid */
}AEC_FRAME_S;
```

[Member]

Member	Description
stRefFrame	Structure of a reference echo cancellation frame
bValid	Flag indicating an invalid reference frame Value range: HI_TRUE: a valid reference frame

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	HI_FALSE: an invalid reference frame. An invalid reference frame cannot be used to perform echo cancellation.

[Note]

None

[See Also]

None

AI_CHN_PARAM_S

[Description]

Defines the channel parameters.

[Syntax]

```
typedef struct hiAI_CHN_PARAM_S
{
   HI_U32 u32UsrFrmDepth;
   HI_S32 s32Rev;
} AI_CHN_PARAM_S;
```

[Member]

Member	Description
u32UsrFrmDepth	Depth of the buffer for storing audio frames
s32Rev	Reserved

[Note]

None

[See Also]

None

AUDIO_STREAM_S

[Description]

Defines the structure of an audio stream.

[Syntax]

```
typedef struct hiAUDIO_STREAM_S
{
   HI_U8 *pStream;       /*Stream virtual address*/
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_U32 u32PhyAddr;     /*Stream physical address*/
    HI_U32 u32Len;         /*Stream length (in byte)*/
    HI_U64 u64TimeStamp;    /*Frame timestamp*/
    HI_U32 u32Seq;         /*Frame sequence. If the stream is not a valid
frame, u32Seq is 0.*/
} AUDIO_STREAM_S;
```

[Member]

Member	Description
pStream	Pointer to an audio stream
u32PhyAddr	Physical address for an audio stream
u32Len	Length (in byte) of an audio stream
u64TimeStamp	Timestamp of an audio stream
u32Seq	Sequence number of an audio stream

[Note]

None

[See Also]

HI_MPI_AENC_GetStream

AUDIO_RESAMPLE_TYPE_E

[Description]

Defines the type of audio resampling.

[Syntax]

```
typedef enum hiAUDIO_RESAMPLE_TYPE_E
{
   AUDIO_RESAMPLE_1X2 = 0x1,
   AUDIO_RESAMPLE_2X1 = 0x2,
   AUDIO_RESAMPLE_1X4 = 0x3,
   AUDIO_RESAMPLE_4X1 = 0x4,
   AUDIO_RESAMPLE_1X6 = 0x5,
   AUDIO_RESAMPLE_6X1 = 0x6,
   AUDIO_RESAMPLE_BUTT
} AUDIO_RESAMPLE_TYPE_E;
```

[Member]

Member	Description
AUDIO_RESAMPLE_1X2	1X2 resampling

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
AUDIO_RESAMPLE_2X1	2X1 resampling
AUDIO_RESAMPLE_1X4	1X4 resampling
AUDIO_RESAMPLE_4X1	4X1 resampling
AUDIO_RESAMPLE_1X6	1X6 resampling
AUDIO_RESAMPLE_6X1	6X1 resampling

[Note]

None

[See Also]

- AUDIO_RESAMPLE_ATTR_S
- HI_MPI_AO_EnableReSmp

AUDIO_RESAMPLE_ATTR_S

[Description]

Defines the structure of the audio resampling attribute.

[Syntax]

```
typedef struct hiAUDIO_RESAMPLE_ATTR_S
{
    HI_U32                u32InPointNum;
    AUDIO_SAMPLE_RATE_E     enInSampleRate;
    AUDIO_RESAMPLE_TYPE_E  enReSampleType;
} AUDIO_RESAMPLE_ATTR_S;
```

[Member]

Member	Description
u32InPointNum	Number of sampling points per frame
enInSampleRate	Sampling rate
enReSampleType	Resampling type

[Note]

None

[See Also]

- AUDIO_SAMPLE_RATE_E
- AUDIO_RESAMPLE_TYPE_E

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

AUDIO_RESAMPLE_ATTR_EX_S

[Description]

Defines the structure of the audio advanced resampling attribute.

[Syntax]

```
typedef struct hiAUDIO_RESAMPLE_ATTR_EX_S
{
    HI_U32                u32InPointNum;
    AUDIO_SAMPLE_RATE_E     enInSampleRate;
    AUDIO_SAMPLE_RATE_E     enOutSampleRate;
} AUDIO_RESAMPLE_ATTR_EX_S;
```

[Member]

Member	Description
u32InPointNum	Number of sampling points in an input frame
enInSampleRate	Input sampling rate
enOutSampleRate	Output sampling rate

[Note]

None

[See Also]

AUDIO_SAMPLE_RATE_E

AO_CHN_STATE_S

[Description]

Defines the status of the data buffer on the AO channel.

[Syntax]

```
typedef struct hiAO_CHN_STATE_S
{
    HI_U32                u32ChnTotalNum;
    HI_U32                u32ChnFreeNum;
    HI_U32                u32ChnBusyNum;
} AO_CHN_STATE_S;
```

[Member]

Member	Description
u32ChnTotalNum	Number of buffers on an AO channel

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u32ChnFreeNum	Number of available buffers
u32ChnBusyNum	Number of used buffers

[Note]

None

[See Also]

HI_MPI_AO_QueryChnStat

G726_BPS_E

[Description]

Defines the bit rate for the G.726 encoding/decoding protocol.

[Syntax]

```
typedef enum hiG726_BPS_E
{
    G726_16K = 0,
    G726_24K,
    G726_32K,
    G726_40K,
    MEDIA_G726_16K,     /* G726 16kbit/s for ASF */
    MEDIA_G726_24K,     /* G726 24kbit/s for ASF */
    MEDIA_G726_32K,     /* G726 32kbit/s for ASF */
    MEDIA_G726_40K,     /* G726 40kbit/s for ASF */
    G726_BUTT,
}G726_BPS_E;
```

[Member]

Member	Description
G726_16K	16 kbit/s G.726. See 4.5.4 G72616 in the *RFC3551*.
G726_24K	24 kbit/s G.726. See 4.5.4 G72624 in the *RFC3551*.
G726_32K	32 kbit/s G.726. See 4.5.4 G72632 in the *RFC3551*.
G726_40K	40 kbit/s G.726. See 4.5.4 G72640 in the *RFC3551*.
MEDIA_G726_16K	G.726 16 kbit/s for ASF
MEDIA_G726_24K	G.726 24 kbit/s for ASF
MEDIA_G726_32K	G.726 32 kbit/s for ASF
MEDIA_G726_40K	G.726 40 kbit/s for ASF

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

None

[See Also]

None

ADPCM_TYPE_E

[Description]

Defines the type of the ADPCM encoding/decoding protocol.

[Syntax]

```
typedef enum hiADPCM_TYPE_E
{
    ADPCM_TYPE_DVI4 = 0,
    ADPCM_TYPE_IMA,
    ADPCM_TYPE_BUTT,
}ADPCM_TYPE_E;
```

[Member]

Member	Description
ADPCM_TYPE_DVI4	32 kbit/s ADPCM (DVI4)
ADPCM_TYPE_IMA	32 kbit/s ADPCM (IMA)

[Note]

None

[See Also]

None

AUDIO_AEC_MODE_E

[Description]

Defines the audio echo cancellation mode.

[Syntax]

```
typedef enum hiAUDIO_AEC_MODE_E
{
    AUDIO_AEC_MODE_RECEIVER  = 0,
    AUDIO_AEC_MODE_SPEAKER   = 1,
    AUDIO_AEC_MODE_HEADPHONE = 2,
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    AUDIO_AEC_MODE_BUTT
} AUDIO_AEC_MODE_E;
```

[Member]

Member	Description
AUDIO_AEC_MODE_RECEIVER	Receiver mode
AUDIO_AEC_MODE_SPEAKER	Speaker mode
AUDIO_AEC_MODE_HEADPHONE	Headphone mode

[Note]

The default mode is the speaker mode.

[See Also]

None

AI_AEC_CONFIG_S

[Description]

Defines the configuration information about audio echo cancellation.

[Syntax]

```
typedef struct hiAI_AEC_CONFIG_S
{
    AUDIO_AEC_MODE_E    enAecMode;      /* AEC mode, default is speaker,
0:receiver  1:speaker  2:headphone */
    HI_S32              s32Reserved;
} AI_AEC_CONFIG_S;
```

[Member]

Member	Description
enAecMode	The modes of echo cancellation are as follows: 0: receiver mode 1: speaker mode 2: headphone mode
s32Reserved	Reserved

[Note]

None

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

AI_ALC_CONFIG_S

[Description]

Defines the configuration information about audio automatic level control.

[Syntax]

```
typedef struct hiAI_ALC_CONFIG_S
{
    HI_S32 s32MaxLev;        /*s32MaxLev: [-23dBm0, -4dBm0]. Default: -
    4dBm0*/
    HI_S32 s32MinLev;        /*s32MinLev: [-23dBm0, -4dBm0]. Default: -
    16dBm0*/
    HI_U32 u32MaxGain;       /*u32MaxGain: [3 dB, 12 dB]. Default: 12 dB*/
} AI_ALC_CONFIG_S;
```

[Member]

Member	Description
s32MaxLev	Maximum target level (in dBm0) Its value range is [−23 dBm0, −4 dBm0], and the default value is **−4** dBm0.
s32MinLev	Minimum target level (in dBm0) Its value range is [−23 dBm0, −4 dBm0], and the default value is **−16** dBm0.
u32MaxGain	Maximum gain Its value range is [3 dB, 12 dB], and the default value is 12 dB.

[Note]

The minimum target level must be less than or equal to the maximum target level.

[See Also]

None

AUDIO_HPF_FREQ_E

[Description]

Defines the cut-off frequency of AI high-pass filtering.

[Syntax]

```
typedef enum hiAUDIO_HPF_FREQ_E
{
    AUDIO_HPF_FREQ_80   = 80,    /*80 Hz*/
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    AUDIO_HPF_FREQ_120  = 120,   /*120 Hz*/
    AUDIO_HPF_FREQ_150  = 150,   /*150 Hz*/
    AUDIO_HPF_FREQ_BUTT,
} AUDIO_HPF_FREQ_E;
```

[Member]

Member	Description
AUDIO_HPF_FREQ_80	The cut-off frequency is 80 Hz.
AUDIO_HPF_FREQ_120	The cut-off frequency is 120 Hz.
AUDIO_HPF_FREQ_150	The cut-off frequency is 150 Hz.

[Note]

You are advised to use the **AUDIO_HPF_FREQ_150** parameter. If the requirements on the low frequency is high, you can set the cut-off frequency to 80 Hz or 120 Hz.

[See Also]

None

AI_HPF_CONFIG_S

[Description]

Defines the high-pass filtering parameter.

[Syntax]

```
typedef struct hiAI_HPF_CONFIG_S
{
    AUDIO_HPF_FREQ_E emHpfFreq; /*freq to be processed*/
} AI_HPF_CONFIG_S;
```

[Member]

Member	Description
AUDIO_HPF_FREQ_E	Cut-off frequency of high-pass filtering

[Note]

None

[See Also]

AUDIO_HPF_FREQ_E

AI_HPF_ATTR_S

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Defines the configuration information about AI high-pass filtering.

[Syntax]

```
typedef struct hiAI_HPF_ATTR_S
{
    AI_HPF_CONFIG_S stHpfCfg;
    HI_S32          bHpfOpen;
} AI_HPF_ATTR_S;
```

[Member]

Member	Description
stHpfCfg	High-pass filtering parameter
bHpfOpen	High-pass filtering enable

[Note]

None

[See Also]

AI_HPF_CONFIG_S

AI_ANR_CONFIG_S

[Description]

Defines the configuration information about audio noise reduction.

[Syntax]

```
typedef struct hiAI_ANR_CONFIG_S
{
    HI_S32 s32Reserved;
} AI_ANR_CONFIG_S;
```

[Member]

Member	Description
s32Reserved	Reserved

[Note]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

AI_VQE_CONFIG_S

[Description]

Defines the configuration information about AI VQE.

[Syntax]

```
typedef struct hiAI_VQE_CONFIG_S
{
    HI_S32            bAecOpen;
    HI_S32            bAnrOpen;
    HI_S32            bAlcOpen;
    HI_S32            s32SampleRate;
    HI_S32            s32FrameSample;
    AI_AEC_CONFIG_S    stAecCfg;
    AI_ANR_CONFIG_S    stAnrCfg;
    AI_ALC_CONFIG_S    stAlcCfg;
} AI_VQE_CONFIG_S;
```

[Member]

Member	Description
bAecOpen	Whether echo cancellation is enabled
bAnrOpen	Whether noise reduction is enabled
bAlcOpen	Whether automatic level control is enabled
s32SampleRate	Sampling rate It can be 8 kHz, 11 kHz or 16 kHz, and the default value is 8 kHz.
s32FrameSample	VQE frame length (number of sampling points) When the sampling rate is 8 kHz, the VQE frame length is 80, 160, 240, 320, 400, 480, or 960 and the default frame length is 160. When the sampling rate is 16 kHz, the VQE frame length is 160, 320, 480 or 960, and the default frame length is 160.
stAecCfg	Configuration information about echo cancellation
stAnrCfg	Configuration information about noise reduction
stAlcCfg	Configuration information about automatic level control

[Note]

None

[See Also]

- AI_AEC_CONFIG_S
- AI_ANR_CONFIG_S

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- AI_ALC_CONFIG_S

AUDIO_SAVE_FILE_INFO_S

[Description]

Defines the configuration information about the function of saving audio files.

[Syntax]

```
typedef struct hiAUDIO_SAVE_FILE_INFO_S
{
    HI_BOOL  bCfg;
    HI_CHAR  aFilePath[256];
    HI_U32 u32FileSize;  /*in KB*/
} AUDIO_SAVE_FILE_INFO_S;
```

[Member]

Member	Description
bCfg	Configure the enable
aFilePath[256]	Path for saving the audio file
u32FileSize	File size, in KB Value range is [1, 8192]

[Note]

None

[See Also]

None

9.4.2 AENC

The AENC data types are defined as follows:

- AENC_MAX_CHN_NUM: Defines the maximum number of AENC channels.
- AENC_ATTR_G711_S: Defines the attribute structure of the G.711 encoding protocol.
- AENC_ATTR_G726_S: Defines the attribute structure of the G.726 encoding protocol.
- AENC_ATTR_ADPCM_S: Defines the attribute structure of the ADPCM encoding protocol.
- AENC_CHN_ATTR_S: Defines the attribute structure of an AENC channel.
- AENC_ENCODER_S: Defines encoder attributes.

AENC_MAX_CHN_NUM

[Description]

Defines the maximum number of AENC channels.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
#define AENC_MAX_CHN_NUM        32
```

[Note]

None

[See Also]

None

AENC_ATTR_G711_S

[Description]

Defines the attribute structure of the G.711 encoding protocol.

[Syntax]

```
typedef struct hiAENC_ATTR_G711_S
{
    HI_U32 resv;
}AENC_ATTR_G711_S;
```

[Member]

Member	Description
resv	To be extended (it is not used currently)

[Note]

None

[See Also]

None

AENC_ATTR_G726_S

[Description]

Defines the attribute structure of the G.726 encoding protocol.

[Syntax]

```
typedef struct hiAENC_ATTR_G726_S
{
    G726_BPS_E  enG726bps;
}AENC_ATTR_G726_S;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
enG726bps	Bit rate of the G.726 protocol

[Note]

None

[See Also]

G726_BPS_E

AENC_ATTR_ADPCM_S

[Description]

Defines the attribute structure of the ADPCM encoding protocol.

[Syntax]

```
typedef struct hiAENC_ATTR_ADPCM_S
{
    ADPCM_TYPE_E    enADPCMType;
}AENC_ATTR_ADPCM_S;
```

[Member]

Member	Description
enADPCMType	ADPCM type

[Note]

None

[See Also]

ADPCM_TYPE_E

AENC_CHN_ATTR_S

[Description]

Defines the attribute structure of an AENC channel.

[Syntax]

```
typedef struct hiAENC_CHN_ATTR_S
{
    PAYLOAD_TYPE_E  enType;
    HI_U32  u32BufSize; /*buffer size. Unit: frame [2-
    MAX_AUDIO_FRAME_NUM]*/
    HI_VOID *pValue;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
}AENC_CHN_ATTR_S;
```

[Member]

Member	Description
enType	Type of an AENC protocol Static attribute
u32BufSize	Size of an AENC buffer, in frame Value range: [2, MAX_AUDIO_FRAME_NUM] Static attribute
pValue	Pointer to the attribute of a specific protocol Static attribute

[Note]

None

[See Also]

None

AENC_ENCODER_S

[Description]

Defines encoder attributes.

[Syntax]

```
typedef struct hiAENC_ENCODER_S
{
    PAYLOAD_TYPE_E  enType;
    HI_U32     u32MaxFrmLen;
    HI_CHAR    aszName[16];
    HI_S32     (*pfnOpenEncoder)(HI_VOID *pEncoderAttr, HI_VOID
**ppEncoder);
    HI_S32     (*pfnEncodeFrm)(HI_VOID *pEncoder, const AUDIO_FRAME_S
*pstData, HI_U8 *pu8Outbuf,HI_U32 *pu32OutLen);
    HI_S32     (*pfnCloseEncoder)(HI_VOID *pEncoder);
} AENC_ENCODER_S;
```

[Member]

Member	Description
enType	Type of the encoding protocol
u32MaxFrmLen	Maximum stream length

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
aszName	Encoder name
pfnOpenEncoder	Pointer to the function for starting an encoder
pfnEncodeFrm	Pointer to encoding functions
pfnCloseEncoder	Pointer to the function for closing an encoder

[Note]

See the *Audio Components API Reference*.

[See Also]

HI_MPI_AENC_RegeisterEncoder

9.4.3 ADEC

The ADEC data types are defined as follows:

- MAX_AUDIO_FRAME_NUM: Defines the maximum number of buffered ADEC frames.
- ADEC_MAX_CHN_NUM: Defines the maximum number of ADEC channels.
- ADEC_ATTR_G711_S: Defines the attribute structure of the G.711 decoding protocol.
- ADEC_ATTR_G726_S: Defines the attribute structure of the G.726 decoding protocol.
- ADEC_ATTR_ADPCM_S: Defines the attribute structure of the ADPCM encoding protocol.
- ADEC_MODE_E: Defines a decoding mode.
- ADEC_CHN_ATTR_S: Defines the attribute structure of an ADEC channel.
- ADEC_DECODER_S: Defines decoder attributes.
- AUDIO_FRAME_INFO_S: Defines the pointer to the structure of the decoded frame information.

MAX_AUDIO_FRAME_NUM

[Description]

Defines the maximum number of buffered ADEC frames.

[Syntax]

```
#define MAX_AUDIO_FRAME_NUM       50
```

[Note]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

ADEC_MAX_CHN_NUM

[Description]

Defines the maximum number of ADEC channels.

[Syntax]

```
#define ADEC_MAX_CHN_NUM        32
```

[Note]

None

[See Also]

None

ADEC_ATTR_G711_S

[Description]

Defines the attribute structure of the G.711 decoding protocol.

[Syntax]

```
typedef struct hiADEC_ATTR_G711_S
{
    HI_U32  resv;
}ADEC_ATTR_G711_S;
```

[Member]

Member	Description
resv	To be extended (it is not used currently)

[Note]

None

[See Also]

None

ADEC_ATTR_G726_S

[Description]

Defines the attribute structure of the G.726 decoding protocol.

[Syntax]

```
typedef struct hiADEC_ATTR_G726_S
{
    G726_BPS_E  enG726bps;
}ADEC_ATTR_G726_S;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Member]

Member	Description
enG726bps	Bit rate of the G.726 protocol

[Note]

None

[See Also]

G726_BPS_E

ADEC_ATTR_ADPCM_S

[Description]

Defines the attribute structure of the ADPCM encoding protocol.

[Syntax]

```
typedef struct hiADEC_ATTR_ADPCM_S
{
    ADPCM_TYPE_E    enADPCMType;
}ADEC_ATTR_ADPCM_S;
```

[Member]

Member	Description
enADPCMType	ADPCM type

[Note]

None

[See Also]

ADPCM_TYPE_E

ADEC_MODE_E

[Description]

Defines a decoding mode.

[Syntax]

```
typedef enum hiADEC_MODE_E
{
ADEC_MODE_PACK = 0,/*require input is valid dec pack(a
                        complete frame encode result),
                        e.g.the stream get from AENC is a
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
                        valid dec pack, the stream know actually
                        pack len from file is also a dec pack.
                        this mode is high-performative*/
ADEC_MODE_STREAM ,/*input is stream,low-performative,
                        if you cannot find out whether a stream is
                        valid dec pack, you could use
                        this mode*/
    ADEC_MODE_BUTT
}ADEC_MODE_E;
```

[Member]

Member	Description
ADEC_MODE_PACK	Pack mode
ADEC_MODE_STREAM	Stream mode

[Note]

- The Pack mode is used when users determine that the current stream is exactly one frame. The decoder directly decodes the stream. If the stream is not one frame, an error occurs in the decoder. This mode enables high efficiency. If a stream encoded by an AENC module is not corrupted, this mode can be used to decode the stream.
- The Stream mode is used when users cannot determine whether the current stream is one frame. The decoder must identify and buffer the stream. Due to low efficiency, this mode is used to read streams in a file to be decoded or streams whose boundaries are uncertain. The pack mode is recommended because the length of the voice encoding stream is fixed; therefore, the boundary of a frame is easily determined.

[See Also]

None

ADEC_CHN_ATTR_S

[Description]

Defines the attribute structure of an ADEC channel.

[Syntax]

```
typedef struct hiADEC_CH_ATTR_S
{
    PAYLOAD_TYPE_E  enType;
    HI_U32          u32BufSize;
    ADEC_MODE_E     enMode;
    HI_VOID         *pValue;
}ADEC_CHN_ATTR_S;
```

[Member]

Copyright © HiSilicon Technologies Co., Ltd

Member	Description
enType	Type of an ADEC protocol Static attribute
u32BufSize	Size of an ADEC buffer, in frame Value range: [2, MAX_AUDIO_FRAME_NUM] Static attribute
enMode	Decoding mode Static attribute
pValue	Pointer to the attribute of a specific protocol

[Note]

None

[See Also]

None

ADEC_DECODER_S

[Description]

Defines decoder attributes.

[Syntax]

```
typedef struct hiADEC_DECODER_S
{
    PAYLOAD_TYPE_E  enType;
    HI_CHAR         aszName[16];
    HI_S32          (*pfnOpenDecoder)(HI_VOID *pDecoderAttr, HI_VOID
**ppDecoder);
    HI_S32          (*pfnDecodeFrm)(HI_VOID *pDecoder, HI_U8
**pu8Inbuf,HI_S32 *ps32LeftByte, HI_U16 *pu16Outbuf,HI_U32
*pu32OutLen,HI_U32 *pu32Chns);
    HI_S32          (*pfnGetFrmInfo)(HI_VOID *pDecoder, HI_VOID *pInfo);
    HI_S32          (*pfnCloseDecoder)(HI_VOID *pDecoder);
} ADEC_DECODER_S;
```

[Member]

Member	Description
enType	Type of the decoding protocol
aszName	Decoder name
pfnOpenDecoder	Pointer to the function for starting a decoder

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
pfnDecodeFrm	Pointer to decoding functions
pfnGetFrmInfo	Pointer to the function for obtaining audio frame information
pfnCloseDecoder	Pointer to the function for closing a decoder

[Note]

See the *Audio Components API Reference*.

[See Also]

HI_MPI_ADEC_RegeisterDecoder

AUDIO_FRAME_INFO_S

[Description]

Defines the pointer to the structure of the decoded frame information.

[Syntax]

```
typedef struct hiAUDIO_FRAME_INFO_S
{
    AUDIO_FRAME_S *pstFrame;/*frame ptr*/
    HI_U32        u32Id;   /*frame id*/
} AUDIO_FRAME_INFO_S;
```

[Member]

Member	Description
pstFrame	Pointer to the decoded frame data
u32Id	Reserved

[Note]

None

[See Also]

- HI_MPI_ADEC_GetFrame
- HI_MPI_ADEC_ReleaseFrame

9.4.4 Built-In Audio CODEC

ACODEC_FS_E

[Description]

Defines the sampling rate of the I^2S interface.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
typedef enum hiACODEC_FS_E {
    ACODEC_FS_48000 =   0x1a,
    ACODEC_FS_24000 =   0x19,
    ACODEC_FS_12000 =   0x18,
    ACODEC_FS_44100 =   0x1a,
    ACODEC_FS_22050 =   0x19,
    ACODEC_FS_11025 =   0x18,
    ACODEC_FS_32000 =   0x1a,
    ACODEC_FS_16000 =   0x19,
    ACODEC_FS_8000  =   0x18,
    ACODEC_FS_BUTT = 0x1b,
} ACODEC_FS_E;
```

[Member]

Member	Description
ACODEC_FS_48000	48 kHz
ACODEC_FS_24000	24 kHz
ACODEC_FS_12000	12 kHz
ACODEC_FS_44100	44.1 kHz
ACODEC_FS_22050	22.05 kHz
ACODEC_FS_11025	11.025 kHz
ACODEC_FS_32000	32 kHz
ACODEC_FS_16000	16 kHz
ACODEC_FS_8000	8 kHz

[Note]

None

[See Also]

None

ACODEC_MIXER_E

[Description]

Defines the input mode of the audio CODEC.

[Syntax]

```
typedef enum hiACODEC_MIXER_E {
    ACODEC_MIXER_LINEIN = 0x0,
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    ACODEC_MIXER_MICIN  = 0x1,
    ACODEC_MIXER_BUTT,
} ACODEC_MIXER_E
```

[Member]

Member	Description
ACODEC_MIXER_LINEIN	LINEIN input for the MICPGA
ACODEC_MIXER_MICIN	MICIN input for the MICPGA

[Note]

None

[See Also]

None

ACODEC_VOL_CTRL

[Description]

Defines the volume control structure of the audio CODEC.

[Syntax]

```
typedef struct {
    /*volume control, 0x00~0x7e, 0x7F:mute*/
    unsigned int vol_ctrl;
    /*adc/dac mute control, 1:mute, 0:unmute*/
    unsigned int vol_ctrl_mute;
} ACODEC_VOL_CTRL;
```

[Member]

Member	Description
vol_ctrl	Volume
vol_ctrl_mute	Mute control

[Note]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

9.5 Error Codes

AI Error Codes

Table 9-9 describes the error codes for the AI MPIs.

Table 9-9 Error codes for the AI MPIs

Error Code	Macro Definition	Description
0xA0158001	HI_ERR_AI_INVALID_DEVID	The AI device ID is invalid.
0xA0158002	HI_ERR_AI_INVALID_CHNID	The AI channel ID is invalid.
0xA0158003	HI_ERR_AI_ILLEGAL_PARAM	The settings of the AI parameters are invalid.
0xA0158005	HI_ERR_AI_NOT_ENABLED	The AI device or AI channel is not enabled.
0xA0158006	HI_ERR_AI_NULL_PTR	The input parameter pointer is null.
0xA0158007	HI_ERR_AI_NOT_CONFIG	The attributes of an AI device are not set.
0xA0158008	HI_ERR_AI_NOT_SUPPORT	The operation is not supported.
0xA0158009	HI_ERR_AI_NOT_PERM	The operation is forbidden.
0xA015800C	HI_ERR_AI_NOMEM	The memory fails to be allocated.
0xA015800D	HI_ERR_AI_NOBUF	The AI buffer is insufficient.
0xA015800E	HI_ERR_AI_BUF_EMPTY	The AI buffer is empty.
0xA015800F	HI_ERR_AI_BUF_FULL	The AI buffer is full.
0xA0158010	HI_ERR_AI_SYS_NOTREADY	The AI system is not initialized.
0xA0158012	HI_ERR_AI_BUSY	The AI system is busy.

AO Error Codes

Table 9-10 describes the error codes for the AO MPIs.

Table 9-10 Error codes for the AO MPIs

Error Code	Macro Definition	Description
0xA0168001	HI_ERR_AO_INVALID_DEVID	The AO device ID is invalid.
0xA0168002	HI_ERR_AO_INVALID_CHNID	The AO channel ID is

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Macro Definition	Description
		invalid.
0xA0168003	HI_ERR_AO_ILLEGAL_PARAM	The settings of the AO parameters are invalid.
0xA0168005	HI_ERR_AO_NOT_ENABLED	The AO device or AO channel is not enabled.
0xA0168006	HI_ERR_AO_NULL_PTR	The output parameter pointer is null.
0xA0168007	HI_ERR_AO_NOT_CONFIG	The attributes of an AO device are not set.
0xA0168008	HI_ERR_AO_NOT_SUPPORT	The operation is not supported.
0xA0168009	HI_ERR_AO_NOT_PERM	The operation is forbidden.
0xA016800C	HI_ERR_AO_NOMEM	The system memory is insufficient.
0xA016800D	HI_ERR_AO_NOBUF	The AO buffer is insufficient.
0xA016800E	HI_ERR_AO_BUF_EMPTY	The AO buffer is empty.
0xA016800F	HI_ERR_AO_BUF_FULL	The AO buffer is full.
0xA0168010	HI_ERR_AO_SYS_NOTREADY	The AO system is not initialized.
0xA0168012	HI_ERR_AO_BUSY	The AO system is busy.

AENC Error Codes

Table 9-11 describes the error codes for the AENC MPIs.

Table 9-11 Error codes for the AENC MPIs

Error Code	Macro Definition	Description
0xA0178001	HI_ERR_AENC_INVALID_DEVID	The AENC device ID is invalid.
0xA0178002	HI_ERR_AENC_INVALID_CHNID	The AENC channel ID is invalid.
0xA0178003	HI_ERR_AENC_ILLEGAL_PARAM	The settings of the AENC parameters are invalid.
0xA0178004	HI_ERR_AENC_EXIST	An AENC channel is created.
0xA0178005	HI_ERR_AENC_UNEXIST	An AENC channel is not created.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Macro Definition	Description
0xA0178006	HI_ERR_AENC_NULL_PTR	The input parameter pointer is null.
0xA0178007	HI_ERR_AENC_NOT_CONFIG	The AENC channel is not configured.
0xA0178008	HI_ERR_AENC_NOT_SUPPORT	The operation is not supported.
0xA0178009	HI_ERR_AENC_NOT_PERM	The operation is forbidden.
0xA017800C	HI_ERR_AENC_NOMEM	The system memory is insufficient.
0xA017800D	HI_ERR_AENC_NOBUF	The buffer for AENC channels fails to be allocated.
0xA017800E	HI_ERR_AENC_BUF_EMPTY	The AENC channel buffer is empty.
0xA017800F	HI_ERR_AENC_BUF_FULL	The AENC channel buffer is full.
0xA0178010	HI_ERR_AENC_SYS_NOTREADY	The system is not initialized.
0xA0178040	HI_ERR_AENC_ENCODER_ERR	An AENC data error occurs.

ADEC Error Codes

Table 9-12 describes the error codes for the ADEC MPIs.

Table 9-12 Error codes for the ADEC MPIs

Error Code	Macro Definition	Description
0xA0188001	HI_ERR_ADEC_INVALID_DEVID	The ADEC device is invalid.
0xA0188002	HI_ERR_ADEC_INVALID_CHNID	The ADEC channel ID is invalid.
0xA0188003	HI_ERR_ADEC_ILLEGAL_PARAM	The settings of the ADEC parameters are invalid.
0xA0188004	HI_ERR_ADEC_EXIST	An ADEC channel is created.
0xA0188005	HI_ERR_ADEC_UNEXIST	An ADEC channel is not created.
0xA0188006	HI_ERR_ADEC_NULL_PTR	The input parameter pointer is null.
0xA0188007	HI_ERR_ADEC_NOT_CONFIG	The attributes of an ADEC channel are not set.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Macro Definition	Description
0xA0188008	HI_ERR_ADEC_NOT_SUPPORT	The operation is not supported.
0xA0188009	HI_ERR_ADEC_NOT_PERM	The operation is forbidden.
0xA018800C	HI_ERR_ADEC_NOMEM	The system memory is insufficient.
0xA018800D	HI_ERR_ADEC_NOBUF	The buffer for ADEC channels fails to be allocated.
0xA018800E	HI_ERR_ADEC_BUF_EMPTY	The ADEC channel buffer is empty.
0xA018800F	HI_ERR_ADEC_BUF_FULL	The ADEC channel buffer is full.
0xA0188010	HI_ERR_ADEC_SYS_NOTREADY	The system is not initialized.
0xA0188040	HI_ERR_ADEC_DECODER_ERR	An ADEC data error occurs.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Contents

10 VDEC 10-1
10.1 Overview 10-1
10.2 Concepts 10-3
10.3 API Reference 10-5
10.4 Data Types 10-39
10.5 Error Codes 10-50

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Tables

Table 10-1 VDEC features of each chip 10-1
Table 10-2 Parameter configurations in different decoding output modes 10-3
Table 10-3 Error codes of VDEC APIs 10-50

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

10 VDEC

10.1 Overview

The VDEC module provides MPIs for driving sub video decoding modules. Table 10-1 describes the VDEC features of each chip.

Table 10-1 VDEC features of each chip

Item	Hi3531	Hi3521	Hi3520A	Hi3520D/Hi3515A/Hi3515C
Sub video decoding modules	Video decoder for high-definition (VDH) JPEG decoder (JPEGD)	VDH Video encoding/decoding unit (VEDU) JPEGD	VDH VEDU JPEGD	VEDU JPEGD
Maximum number of VDEC channels (VDEC_MAX_CHN_NUM)	64	64 (channels 0–31 for the VHD, and channel 32–63 for the VEDU)	64 (channels 0–31 for the VHD, and channel 32–63 for the VEDU)	64
Supported protocol	**VDH:** H.264/VC1/MPEG4/ MPEG2/AVS **JPEGD:** JPEG/MJPEG	**VDH:** H.264 **JPEGD:** JPEG/MJPEG **VEDU:** H.264 (only the streams encoded or decoded by the Hi3521 are supported)	**VDH:** H.264 (B frames and decoding by field are not supported) **JPEGD:** JPEG/MJPEG **VEDU:** H.264 (only the streams encoded or decoded by the Hi3520A are	**VEDU:** H.264 (B frames and decoding by field are not supported) **JPEGD:** JPEG/MJPEG

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Item	Hi3531	Hi3521	Hi3520A	Hi3520D/Hi3515A/Hi3515C
			supported)	
Maximum resolution	**VDH:** H.264: max 5632x4224, min 64x64 MPEG4/MPEG2/VC1/AVS: max1920x1080, min 64x64 **JPEGD:** JPEG/MJPEG: max 4096x4080, min 64x64	**VDH:** H.264: max 5632x4224, min 64x64 MPEG4: max 1920x1080, min 64x64 **VEDU:** H.264: max 1920x2048, min 160x64 **JPEGD:** JPEG/MJPEG: max 4096x4080, min 64x64	**VDH:** H.264: max 1920x2048, min 64x64 **VEDU:** H.264: max 1920x2048, min 160x64 **JPEGD:** JPEG/MJPEG: max 4096x4080, min 64x64	**VEDU:** H.264: max 1920x2048, min 80x64 **JPEGD:** JPEG/MJPEG: max 4096x4080, min 64x64
Maximum number of decoded pixels per second This parameter value is estimated based on the decoder performance and is only for reference.	3 megapixels x 60 fps x2 = 360 megapixels	2 megapixels x 60 fps = 120 megapixels	2 megapixels x 30 fps = 60 megapixels	2 megapixels x 30 fps = 60 megapixels
Required drivers	hi3531_vdec.ko hi3531_vfmw.ko jpeg.ko	hi3521_vdec.ko hi3521_vfmw.ko hi3521_chnl.ko (for the VEDU) jpeg.ko	hi3520A_vdec.ko hi3520A_vfmw.ko hi3520A_chnl.ko	hi3520D_vdec.ko hi3520D_vfmw.ko hi3520D_chnl.ko (decoded by the VEDU) jpeg.ko

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

10.2 Concepts

Note the following:

- Mode of sending streams

  The VEDU supports the following modes of sending streams:

  - Sending streams by stream: You can send the streams in any length to the decoder. The decoder automatically identifies a frame of streams. Note that the decoder can identify the end of current frame only when it receives the next frame. Therefore, streams cannot be immediately decoded when a frame of H.264 streams is received.
  - Sending streams by frame: The decoder identifies the end of the current frame when you send a complete frame to the decoder by calling the stream sending MPI. Ensure that the stream sent by calling the stream sending MPI is a complete frame. Otherwise, a decoding error occurs. This mode ensures that streams are rapidly decoded.

  The mode of sending streams is set when the channel attributes are set. For details, see section 10.3 "API Reference." In the two modes, the size of the stream sent by calling the stream sending MPI cannot be greater than the buffer size configured in the channel attributes. You can select different picture output modes by setting the mode of sending streams and the decoder output sequence.

- Picture output mode

  According to the H.264 protocol, the decoded pictures may not be output immediately after decoding. You can enable the VEDU decoder to output decoded pictures rapidly by setting the picture output mode. There are three picture output modes:

  - Common output: The pictures are output according to the H.264 protocol.
  - Direct output: The picture of the current frame is output after the stream of the next frame is received.
  - Output by frame: The picture of the current frame is output after the stream of the current frame is received.

  In the preceding output modes, the output speed of the first mode is the lowest and the output speed of the third mode is the highest. The modes of fast output and output by frame do not comply with the H.264 protocol. To output pictures properly, you need to configure and use the decoder properly. For details, see Table 10-2.

- Channel decoding capability

  The channel decoding capability indicates the maximum resolution allowed of the picture to be decoded by the VDEC channel. For example, if a VDEC channel with 720-pixel width and 576-pixel height is created, its decoding capability is 720 x 576. When the memory is sufficient, the VDEC channel can decode the streams with the resolution of 720 x 576 (W x H) or lower.

  Note that the required memory size varies according to the stream width and height even though the total resolution (W x H) is the same. For details about the calculation method of frame buffer size, see the description of HI_MPI_VDEC_CreateChn.

Table 10-2 Parameter configurations in different decoding output modes

Output Mode	Channel Attribute	Channel Parameter	Remarks
	enMode	s32DecOrderOutput	
Common output	VIDEO_MODE_STREAM (sending streams by stream)	0 (output in the display sequence)	None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Output Mode	Channel Attribute	Channel Parameter	Remarks
	enMode	s32DecOrderOutput	
	VIDEO_MODE_FRAME (sending streams by frame)	0 (output in the display sequence)	Ensure that the streams are sent by frame.
Direct output	VIDEO_MODE_STREAM (sending streams by stream)	1 (output in the decoding sequence)	Ensure that the decoding sequence is the same as the display sequence.
Output by frame	VIDEO_MODE_FRAME (sending streams by frame)	1 (output in the decoding sequence)	Ensure that the streams are sent by frame and the decoding sequence is the same as the display sequence.

According to the H.264 protocol, the video decoding sequence may not be the video output sequence (display sequence). For example, a B frame is decoded by referencing its previous and next P frames. Therefore, the previous P frame is decoded before the B frame is decoded, but the B frame is output before its previous frame. Output in decoding sequence is necessary for fast output. If output in decoding sequence is selected, the sequence of decoding streams must be the same as the display sequence.

The stream transmission mode enMode can be set by calling HI_MPI_VDEC_CreateChn, and the image output mode s32DecOrderOutput can be set by calling HI_MPI_VDEC_SetChnParam. For details, see section 10.3 "API Reference." PTS processing

If streams are sent by calling HI_MPI_VDEC_SendStream in VIDEO_MODE_FRAME mode, the presentation timestamps (PTSs) of the decoded pictures are the PTSs of the streams. The decoder does not change the PTSs. If streams are sent in VIDEO_MODE_STREAM mode, the PTSs of decoded pictures are 0.

- Configuration mode of the stream buffer

  The stream buffer for decoding supports the common mode and memory reduction mode.

  - Common mode: The total size of the stream buffer is calculated as follows: Total size of the stream buffer = **u32BufSize** + Channel width x Channel height/2. The minimum value of **u32BufSize** is calculated as follows: Minimum value of **u32BufSize** = Channel width x Channel height x 3/4. In this mode, the stream buffer can store all stream frames including the jumbo frames.
  - Memory reduction mode: The total size of the stream buffer is the value of **u32BufSize** configured when a VDEC channel is created. The minimum value of **u32BufSize** is 32 KB. Ensure that the size of each frame sent to the stream buffer must be less than or equal to half of **u32BufSize**. The memory reduction mode is valid only when streams are decoded by frame.

  You can configure the stream buffer mode by setting the module parameter **MiniBufMode** when loading the .ko drivers. If **MiniBufMode** is **1**, the memory reduction mode is selected; if **MiniBufMode** is **0**, the common mode is selected.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

10.3 API Reference

The VDEC module supports the functions including creating a VDEC channel, sending video streams, and obtaining the decoded pictures.

The VDEC module provides the following MPIs:

- HI_MPI_VDEC_CreateChn: Creates a VDEC channel.
- HI_MPI_VDEC_DestroyChn: Destroys a VDEC channel.
- HI_MPI_VDEC_StartRecvStream: Starts to receive the streams sent by the user.
- HI_MPI_VDEC_StopRecvStream: Stops receiving the streams sent by the user.
- HI_MPI_VDEC_Query: Queries the status of a VDEC channel.
- HI_MPI_VDEC_QueryData: Query whether there are output pictures or user data in a VDEC channel.
- HI_MPI_VDEC_GetChnAttr: Obtains the attributes of a VDEC channel.
- HI_MPI_VDEC_SendStream: Sends streams to a VDEC channel.
- HI_MPI_VDEC_GetUserData: Obtains user data in a VDEC channel.
- HI_MPI_VDEC_ReleaseUserData: Releases the buffer for storing the user data of a VDEC channel.
- HI_MPI_VDEC_GetImage: Obtains decoded pictures from a VDEC channel.
- HI_MPI_VDEC_GetImage_TimeOut: Obtains decoded pictures from a VDEC channel in timeout block mode.
- HI_MPI_VDEC_ReleaseImage: Releases the buffer for storing the decoded pictures of a VDEC channel.
- HI_MPI_VDEC_GetFd: Obtains the device file descriptor (FD) corresponding to a VDEC channel.
- HI_MPI_VDEC_ResetChn: Resets a VDEC channel.
- HI_MPI_VDEC_SetChnParam: Sets the parameters of a VDEC channel.
- HI_MPI_VDEC_GetChnParam: Obtains the parameters of a VDEC channel.
- HI_MPI_VDEC_SetPrtclParam: Sets the memory allocation parameters related to protocols in the decoding channel.
- HI_MPI_VDEC_GetPrtclParam: Obtains the memory allocation parameters related to protocols in the decoding channel.
- HI_MPI_VDEC_SetDisplayFrameNum: Sets the number of display frame buffers.
- HI_MPI_VDEC_GetDisplayFrameNum: Obtains the number of display frame buffers.
- HI_MPI_VDEC_SetChnPrevMode: Sets the channel preview mode.
- HI_MPI_VDEC_GetChnPrevMode: Obtains the channel preview mode.

HI_MPI_VDEC_CreateChn

[Description]

Creates a VDEC channel.

[Syntax]

```
HI_S32 HI_MPI_VDEC_CreateChn(VDEC_CHN VdChn, const VDEC_CHN_ATTR_S *pstAttr)
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
pstAttr	Pointer to the decoding attributes	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_NULL_PTR	The pointer is null.
HI_ERR_VDEC_ILLEGAL_PARAM	The parameter is incorrect or is not within the valid range.
HI_ERR_VDEC_NOT_SUPPORT	The decoding capability set is exceeded. For details about the capability set, see Table 10-1.
HI_ERR_VDEC_EXIST	The device, channel, or resource to be applied for or created already exists.
HI_ERR_VDEC_NOMEM	The memory fails to be allocated due to insufficient memory.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

- The channel ID cannot be greater than the maximum channel ID.
- The .ko driver corresponding to related protocol must be loaded before decoding. For details, see Table 10-1.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- If pstAttr is invalid, HI_ERR_VDEC_ILLEGAL_PARAM is returned; if pstAttr is null, HI_ERR_VDEC_NULL_PTR is returned. For details about the description and value range of pstAtt, see the description of VDEC_CHN_ATTR_S.
- Before creating a VDEC channel, ensure that no channel is created or the channel is destroyed. Otherwise, the error code indicating that the channel already exists is returned.
- When the system memory is insufficient, the error code HI_ERR_VDEC_NOMEM is returned. In this case, you can extend the media memory zone (MMZ) or OS memory.
- When the values of channel attributes exceed the decoding capability set, the error code HI_ERR_VDEC_NOT_SURPPORT is returned. For details about the decoding capability set, see Table 10-1.
- The stream width and height cannot be greater than the maximum channel width and height. Otherwise, the decoder does not output pictures. You can query whether the value of SizeSetErr increases by viewing the proc information or log. The decoder cannot decode the picture whose width exceeds the maximum channel width or whose height exceeds the maximum channel height. If the maximum channel width or maximum channel height is exceeded, the decoder has no output. You can check whether such error occurs by viewing PicSizeErr or logs.
- The channel attributes are static attributes. The attributes can be set only when a channel is created.
- The Hi3520D supports a maximum of 64 H.264 VDEC channels, whereas the Hi3515A/Hi3515C supports a maximum of four H.264 VDEC channels.
- If you decode the streams with the resolutions of 720 x 576 and 576 x 720 in the same decoding channel, you can calculate the memory size required by the two streams. The size of the frame buffer required by the 720 x 576 stream is 663,552 bytes and the size of the frame buffer required by the 576 x 720 stream is 622,080 bytes. Then create a decoding channel based on the larger memory size.

CAUTION

- Only the mode of sending streams by frame is supported during JPEG or MJPEG decoding.
- If you want to decode JPEG pictures in YUV444 format, you must set the width or height of the decoding channel to two times of the width or height of the JPEG picture. Otherwise, pictures fail to be decoded due to decoding buffer insufficiency.

[Example]

None

[See Also]

The decoding frame buffer size is calculated as follows:

The following example assumes that the width of the stream is Width, the height of the stream is Height, and the frame buffer size required for decoding is FrameSize.

SlotWidth = (Width + 63) & (~63);

SlotHeight = (Height+ 16) & (~16);

For H.264 and MPEG4, FrameSize = (SlotWidth x SlotHeight x 3)/2;

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

For JPEG, FrameSize = SlotWidth x SlotHeight x 2.

HI_MPI_VDEC_DestroyChn

[Description]

Destroys a VDEC channel.

[Syntax]

```
HI_S32 HI_MPI_VDEC_DestroyChn(VDEC_CHN VdChn);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_NOT_PERM	The operation is not permitted. Before this operation, stream receiving must be stopped.
HI_ERR_VDEC_UNEXIST	No channel is created or the channel has been destroyed.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Before destroying a VDEC channel, ensure that the channel exists. Otherwise, the error code indicating that no channel is created is returned.
- Before destroying a VDEC channel, ensure that the channel stops receiving streams or the channel does not start to receive streams. Otherwise, the error code HI_ERR_VDEC_NOT_PERM is returned.

[Example]

None

[See Also]

None

HI_MPI_VDEC_StartRecvStream

[Description]

Starts to receive the streams sent by the user.

[Syntax]

```
HI_S32 HI_MPI_VDEC_StartRecvStream(VDEC_CHN VdChn);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_UNEXIST	No channel is created or the channel has been destroyed.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

- Before starting to receive streams, ensure that a channel is created. Otherwise, the error code indicating that no channel is created is returned.
- You can successfully send streams by calling HI_MPI_VDEC_SendStream only after streams are received.
- If you call the MPI repeatedly, HI_SUCCESS is returned.

[Example]

None

[See Also]

None

HI_MPI_VDEC_StopRecvStream

[Description]

Stops receiving the streams sent by the user.

[Syntax]

```
HI_S32 HI_MPI_VDEC_StopRecvStream(VDEC_CHN VdChn);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Definition
HI_ERR_VDEC_UNEXIST	No channel is created or the channel has been destroyed.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

- Before stop receiving streams, ensure that a channel is created. Otherwise, the error code indicating that no channel is created is returned.
- If you call HI_MPI_VDEC_SendStream after calling HI_MPI_VDEC_StopRecvStream, an error code indicating failure is returned.
- If you call the MPI repeatedly, HI_SUCCESS is returned.

[Example]

None

[See Also]

None

HI_MPI_VDEC_Query

[Description]

Queries the status of a VDEC channel.

[Syntax]

```
HI_S32 HI_MPI_VDEC_Query(VDEC_CHN VdChn,VDEC_CHN_STAT_S *pstStat);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
pstStat	Pointer to the status structure of a VDEC channel	Output

[Return Value]

Return Value	Description
0	Success

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_UNEXIST	No channel is created or exists.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_NULL_PTR	The pointer to the status structure of a VDEC channel is null.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

Before querying the status of a channel, ensure that the channel is created. Otherwise, the error code indicating that no channel is created is returned.

[Example]

None

[See Also]

None

HI_MPI_VDEC_QueryData

[Description]

Query whether there are output pictures or user data in a VDEC channel.

[Syntax]

```
HI_S32 HI_MPI_VDEC_QueryData(VDEC_CHN VdChn,HI_BOOL *pbIsData);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
pbIsData	Whether there are output pictures or user data in a VDEC channel	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_UNEXIST	No channel is created or exists.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_NULL_PTR	The pointer to the status structure of a VDEC channel is null.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

Before querying whether there are data in a VDEC channel, ensure that the channel is created. Otherwise, the error code indicating that no channel is created is returned.

[Example]

None

[See Also]

None

HI_MPI_VDEC_GetChnAttr

[Description]

Obtains the attributes of a VDEC channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_VDEC_GetChnAttr(VDEC_CHN VdChn, VDEC_CHN_ATTR_S*pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
pstAttr	Pointer to the decoding attributes	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_NULL_PTR	The pointer is null.
HI_ERR_VDEC_UNEXIST	The channel does not exist.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

Before obtaining the attributes of a channel, ensure that the channel exists. Otherwise, the error code indicating that no channel is created is returned.

[Example]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VDEC_SendStream

[Description]

Sends streams to a VDEC channel.

[Syntax]

```
HI_S32 HI_MPI_VDEC_SendStream(VDEC_CHN VdChn, const VDEC_STREAM_S *pstStream,
HI_U32 u32BlockFlag);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
pstStream	Pointer to the decoding stream data.	Input
u32BlockFlag	Block flag Value: • HI_IO_BLOCK: block • HI_IO_NOBLOCK: non-block Dynamic attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_NULL_PTR	The pointer is null.
HI_ERR_VDEC_ILLEGAL_PARAM	The parameter is invalid.
HI_ERR_VDEC_UNEXIST	The device, channel, or resource being used or destroyed does not exist.
HI_ERR_VDEC_NOT_PERM	The operation is not permitted. The streams are

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Definition
	sent to a channel before the channel is ready.
HI_ERR_VDEC_BUF_FULL	The buffer is full. This error code is returned if streams fail to be sent by calling HI_MPI_VDEC_SendStream in non-block mode.
HI_ERR_VDEC_BADADDR	The stream address is incorrect.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

- This MPI supports both the block mode and non-block mode, which is switched by setting the u32BlockFlag value.
- Before sending streams, ensure that the channel starts to receive streams by calling HI_MPI_VDEC_StartRecvStream. Otherwise, HI_ERR_VDEC_NOT_PERM is returned. If stream reception stops when streams are being sent, HI_ERR_VDEC_NOT_PERM is returned immediately.
- Before sending streams, ensure that a channel is created. Otherwise, HI_ERR_VDEC_UNEXIST is returned. If the channel is destroyed when streams are being sent, HI_ERR_VDEC_UNEXIST is returned immediately.
- You must send streams by using the mode (by frame or by stream) that is set when a VDEC channel is created. When streams are sent by frame by calling this MPI, ensure that each time a complete frame is sent. Otherwise, a decoding error occurs. There is such limitation when streams are sent by stream.
- The stream length cannot be 0. Otherwise, HI_ERR_VDEC_ILLEGAL_PARAM is returned.
- When streams are sent by frame, the PTSs of decoded pictures are the PTSs transferred to pstStream. When streams are sent by stream, the PTSs of decoded pictures are 0.

[Example]

None

[See Also]

None

HI_MPI_VDEC_SendStream_TimeOut

[Description]

Sends streams to a VDEC channel in timeout mode.

[Syntax]

```
HI_S32 HI_MPI_VDEC_SendStream_TimeOut(VDEC_CHN VdChn, const VDEC_STREAM_S
*pstStream, HI_U32 u32MilliSec);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
pstStream	Pointer to the decoding stream data	Input
u32MilliSec	Timeout period, in the unit of ms Value range: [0, 10000]	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_NULL_PTR	The pointer is null.
HI_ERR_VDEC_ILLEGAL_PARAM	The parameter is invalid.
HI_ERR_VDEC_UNEXIST	The device, channel, or resource to be used or destroyed does not exist.
HI_ERR_VDEC_NOT_PERM	The operation is not permitted. The streams are sent to a channel before the channel is ready.
HI_ERR_VDEC_BUSY	A timeout occurs because the buffer cannot receive streams during the specified period.
HI_ERR_VDEC_BADADDR	The stream address is incorrect.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- When this MPI works in block mode and no streams are received during the specified period, a timeout occurs.
- Before sending streams, ensure that the channel starts to receive streams by calling HI_MPI_VDEC_StartRecvStream. Otherwise, HI_ERR_VDEC_NOT_PERM is returned. If stream reception stops when streams are being sent, HI_ERR_VDEC_NOT_PERM is returned immediately.
- Before sending streams, ensure that a channel is created. Otherwise, HI_ERR_VDEC_UNEXIST is returned. If the channel is destroyed when streams are being sent, HI_ERR_VDEC_UNEXIST is returned immediately.
- You must send streams by using the mode (by frame or by stream) that is set when a VDEC channel is created. When streams are sent by frame by calling this MPI, ensure that each time a complete frame is sent. Otherwise, a decoding error occurs. There is such limitation when streams are sent by stream.
- The stream length cannot be 0. Otherwise, HI_ERR_VDEC_ILLEGAL_PARAM is returned.
- When streams are sent by frame, the presentation time stamp (PTS) of the decoded picture is the same as that of the stream packet. When streams are sent by stream, the PTS of the decoded picture is 0.
- As the timeout period is in the unit of 10 ms, the maximum wait time must be a multiple of 10 ms.
- When the timeout period is 0, an HI_IO_BLOCK operation is performed. The MPI is returned only when streams are stored in the buffer or the VDEC channel stops receiving streams.

[Example]

None

[See Also]

None

HI_MPI_VDEC_GetUserData

[Description]

Obtains user data in a VDEC channel.

[Syntax]

```
HI_S32 HI_MPI_VDEC_GetUserData(VDEC_CHN VdChn, VDEC_USERDATA_S * pstUserData,
HI_U32 u32BlockFlag);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
u32BlockFlag	Block flag Value: • HI_IO_BLOCK: block • HI_IO_NOBLOCK: non-block	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
	Dynamic attribute	
pstUserData	Pointer to the decoded user data that is obtained	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_NULL_PTR	The pointer is null.
HI_ERR_VDEC_UNEXIST	The device, channel, or resource being used or destroyed does not exist.
HI_ERR_VDEC_BUF_EMPTY	The data in the buffer is insufficient in non-block mode.
HI_ERR_VDEC_NOT_PERM	This error code is returned if a reset operation is performed when user data is being obtained.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

- User data is the data that is inserted into the SEI segment during H.264 encoding. User data is not supported during JPEG or MJPEG encoding.
- Before obtaining the user data, ensure that a channel is created. Otherwise, an error code is returned directly. In addition, if the channel is destroyed when data is being obtained, HI_ERR_VDEC_UNEXIST is returned immediately.
- This MPI supports both block mode and non-block mode. If the user data is obtained in non-block mode and the decoded user data in the decoder is less than 1 byte, HI_ERR_VDEC_BUF_EMPTY is returned.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- After the decoded user data is obtained by calling HI_MPI_VDEC_GetUserData, the buffer storing the decoded data must be released by calling HI_MPI_VDEC_ReleaseUserData and the user data information during the obtain and release operations must be the same.
- The error code HI_ERR_VDEC_NOT_PERM may be returned if a reset operation is performed when user data is being obtained.
- If the user data is not obtained in time, the user data buffer may be full, which results in data loss.

[Example]

None

[See Also]

None

HI_MPI_VDEC_GetUserData_TimeOut

[Description]

Obtains user data from a VDEC channel in timeout mode.

[Syntax]

```
HI_S32 HI_MPI_VDEC_GetUserData_TimeOut(VDEC_CHN VdChn, VDEC_USERDATA_S *
pstUserData, HI_U32 u32MilliSec);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
u32MilliSec	Timeout period, in the unit of ms Value range: [0, 10000]	Input
pstUserData	Pointer to the decoded user data that is obtained	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_NULL_PTR	The pointer is null.
HI_ERR_VDEC_UNEXIST	The device, channel, or resource to be used or destroyed does not exist.
HI_ERR_VDEC_NOT_PERM	This error code is returned if a reset operation is performed when user data is being obtained.
HI_ERR_VDEC_BUSY	A timeout occurs because no user data is obtained during the specified period.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

- User data is the data that is inserted into the SEI segment during H.264 encoding. User data is not supported during JPEG or MJPEG encoding.
- Before obtaining the user data, ensure that a channel is created. Otherwise, an error code is returned directly. In addition, if the channel is destroyed when data is being obtained, HI_ERR_VDEC_UNEXIST is returned immediately.
- This MPI supports both block mode and non-block mode. If the user data is obtained in non-block mode and the decoded user data in the decoder is less than 1 byte, HI_ERR_VDEC_BUF_EMPTY is returned.
- After the decoded user data is obtained by calling HI_MPI_VDEC_GetUserData, the buffer storing the decoded data must be released by calling HI_MPI_VDEC_ReleaseUserData and the user data information during the obtain and release operations must be the same.
- The error code HI_ERR_VDEC_NOT_PERM may be returned if a reset operation is performed when user data is being obtained.
- If the user data is not obtained in time, the user data buffer may be full, which results in data loss.
- As the timeout period is in the unit of 10 ms, the maximum wait time must be a multiple of 10 ms.
- When the timeout period is 0, an HI_IO_BLOCK operation is performed. The MPI is returned only when pictures are received or the VDEC channel is destroyed.

[Example]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_VDEC_ReleaseUserData

[Description]

Releases the buffer for storing the user data of a VDEC channel.

[Syntax]

```
HI_S32 HI_MPI_VDEC_ReleaseUserData(VDEC_CHN VdChn, VDEC_USERDATA_S *
pstUserData);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
pstUserData	Pointer to the decoded user data. The information is obtained by calling HI_MPI_VDEC_GetUserData.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_NULL_PTR	The pointer is null.
HI_ERR_VDEC_UNEXIST	The device, channel, or resource being used or destroyed does not exist.
HI_ERR_VDEC_ILLEGAL_PARAM	The parameter is invalid.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

- This MPI must be called if HI_MPI_VDEC_GetUserData is called. In addition, the buffer must be released immediately after data is obtained.
- The buffer to be released must be the buffer that stores the data obtained by calling HI_MPI_VDEC_GetUserData. Note that the information pointed by the pstData pointer cannot be modified. Otherwise, HI_ERR_VDEC_ILLEGAL_PARAM is returned.
- Before releasing a buffer, ensure that a channel is created. Otherwise, HI_ERR_VDEC_UNEXIST is returned. If the channel is destroyed when the buffer is being released, HI_ERR_VDEC_UNEXIST is returned immediately.
- If a reset operation is performed when the buffer is being released, HI_ERR_VDEC_NOT_PERM is returned.

[Example]

None

[See Also]

None

HI_MPI_VDEC_GetImage

[Description]

Obtains decoded pictures from a VDEC channel.

[Syntax]

```
HI_S32 HI_MPI_VDEC_GetImage(VDEC_CHN VdChn, VIDEO_FRAME_INFO_S
*pstFrameInfo, HI_U32 u32BlockFlag);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
u32BlockFlag	Block flag Value: • HI_IO_BLOCK: block • HI_IO_NOBLOCK: non-block Dynamic attribute	Input
pstFrameInfo	Information about the decoded pictures that are obtained	Output

[Return Value]

Return Value	Description
0	Success

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_NULL_PTR	The pointer is null.
HI_ERR_VDEC_UNEXIST	The device, channel, or resource being used or destroyed does not exist.
HI_ERR_VDEC_BUF_EMPTY	In non-block mode, the buffer is empty.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

- This interface supports both the block mode and non-block mode.
- After obtaining decoded pictures by calling HI_MPI_VDEC_GetImage, you need to release the buffer for storing the decoded pictures by calling HI_MPI_VDEC_ReleaseImage.
- Before obtaining decoded pictures, ensure that a channel is created. Otherwise, an error code is returned. In addition, if the channel is destroyed when decoded pictures are being obtained, an error coded indicating failure is returned immediately.
- If the channel is reset when decoded pictures are being obtained, the error code HI_ERR_VDEC_NOT_PERM is returned immediately.

[Example]

None

[See Also]

None

HI_MPI_VDEC_GetImage_TimeOut

[Description]

Obtains decoded pictures from a VDEC channel in timeout block mode.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
HI_MPI_VDEC_GetImage_TimeOut(VDEC_CHN VdChn, VIDEO_FRAME_INFO_S
*pstFrameInfo,HI_U32 u32MilliSec)
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
u32MilliSec	Timeout period, in ms Value range: [0, 10000]	Input
pstFrameInfo	Information about the decoded pictures that are obtained	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_NULL_PTR	The pointer is null.
HI_ERR_VDEC_UNEXIST	The device, channel, or resource being used or destroyed does not exist.
HI_ERR_VDEC_BUSY	Timeout occurs because no pictures are obtained during the specified period.
HI_ERR_VDEC_NOT_PERM	The channel being used is destroyed or reset.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- When you call this MPI in block mode and no pictures are obtained during the specified period, an error code indicating timeout is returned.
- After obtaining decoded pictures by calling HI_MPI_VDEC_GetImage_TimeOut, you need to release the buffer for storing the decoded pictures by calling HI_MPI_VDEC_ReleaseImage.
- Before obtaining decoded pictures, ensure that a channel is created. Otherwise, an error code indicating failure is returned. In addition, if the channel is destroyed when decoded pictures are being obtained, an error coded indicating failure is returned immediately.
- If the channel is reset when decoded pictures are being obtained, the error code HI_ERR_VDEC_NOT_PERM is returned immediately.
- As the timeout period is a multiple of 10 ms, the maximum wait time must be a multiple of 10 ms.
- When the timeout period is 0, an HI_IO_BLOCK operation is performed. The MPI is returned only when pictures are received or the VDEC channel is destroyed.

[Example]

None

[See Also]

None

HI_MPI_VDEC_ReleaseImage

[Description]

Releases the buffer for storing the decoded pictures of a VDEC channel.

[Syntax]

```
HI_S32 HI_MPI_VDEC_ReleaseImage(VDEC_CHN VdChn, VIDEO_FRAME_INFO_S
*pstFrameInfo);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
pstFrameInfo	Pointer to the information about the decoded picture. The information is obtained by calling HI_MPI_VDEC_GetImage.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_NULL_PTR	The pointer is null.
HI_ERR_VDEC_UNEXIST	The device, channel, or resource being used or destroyed does not exist.
HI_ERR_VDEC_ILLEGAL_PARAM	The parameter is invalid.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

- This MPI must be called if HI_MPI_VDEC_GetImage is called. The buffer must be released immediately after data is obtained. Otherwise, the decoding process is blocked for waiting for resources.
- The buffer to be released must be the buffer that stores the data obtained by calling HI_MPI_VDEC_GetImage. Note that the information pointed by the pstFrameInfo pointer cannot be modified and the information cannot be used for other channels. Otherwise, the target buffer fails to be released, data in the buffer is lost, and the program is even abnormal.
- Before releasing a buffer, ensure that a channel is created. Otherwise, HI_ERR_VDEC_UNEXIST is returned. If the channel is destroyed when the buffer is being released, HI_ERR_VDEC_UNEXIST is returned immediately.
- Buffers can be released not in the sequence of obtaining pictures.
- If the channel is reset when pictures are being obtained, an error code is returned immediately.

[Example]

None

[See Also]

None

HI_MPI_VDEC_GetFd

[Description]

Obtains the device FD corresponding to a VDEC channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_VDEC_GetFd(VDEC_CHN VdChn);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input

[Return Value]

Return Value	Description
Positive number	Valid value
Zero or negative number	Invalid value

[Error Code]

None

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

None

[Example]

None

[See Also]

None

HI_MPI_VDEC_ResetChn

[Description]

Resets a VDEC channel.

[Syntax]

```
HI_S32 HI_MPI_VDEC_ResetChn(VDEC_CHN VdChn);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
	Value range: [0, VDEC_MAX_CHN_NUM)	

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_UNEXIST	No channel exists.
HI_ERR_VDEC_NOT_PERM	This operation is not permitted. You must stop receiving stream before this operation.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

- Before resetting a VDEC channel, ensure that the channel is created. Otherwise, HI_ERR_VDEC_UNEXIST is created.
- You must stop receiving streams before resetting a VDEC channel. Otherwise, HI_ERR_VDEC_NOT_PERM is returned.

[Example]

None

[See Also]

None

HI_MPI_VDEC_SetChnParam

[Description]

Sets the parameters of a VDEC channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_VDEC_SetChnParam(VDEC_CHN VdChn, VDEC_CHN_PARAM_S* pstParam);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
pstParam	Channel parameters	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_UNEXIST	No channel exists.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_ILLEGAL_PARAM	The parameter value is invalid.
HI_ERR_VDEC_NOT_SURPPORT	The JPEG channel or MJPEG channel is not supported.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

- Before setting channel parameters, ensure that the channel is created. Otherwise, an error code is returned, indicating that no channel is created.
- This MPI can be used to set some advanced attributes of a VDEC channel. The parameters of a channel are set to default values after the channel is created. For details about default values, see the description of VDEC_CHN_PARAM_S.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- If the parameter values are not within the value range, the error code HI_ERR_VDEC_ILLEGAL_PARAM is returned. For details about the value range, see the description of VDEC_CHN_PARAM_S.
- This MPI is not supported during JPEG or MJPEG decoding.

[Example]

None

[See Also]

None

HI_MPI_VDEC_GetChnParam

[Description]

Obtains the parameters of a VDEC channel.

[Syntax]

```
HI_S32 HI_MPI_VDEC_GetChnParam(VDEC_CHN VdChn, VDEC_CHN_PARAM_S* pstParam);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
pstParam	Channel parameters	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_UNEXIST	No channel exists.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_NOT_SURPPORT	The JPEG channel or MJPEG channel is not

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Definition
	supported.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

- This MPI is not supported during JPEG or MJPEG decoding.
- Before obtaining channel parameters, ensure that the channel is created. Otherwise, an error code is returned, indicating that no channel is created.

[Example]

None

[See Also]

None

HI_MPI_VDEC_SetPrtclParam

[Description]

Sets the memory allocation parameters related to protocols in the decoding channel.

[Syntax]

```
HI_S32 HI_MPI_VDEC SetPrtclParam (VDEC_CHN VdChn, VDEC_PRTCL_PARAM_S*
pstParam);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
pstParam	Channel protocol parameter	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_UNEXIST	No channel exists.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.
HI_ERR_VDEC_ILLEGAL_PARAM	The parameter is incorrect or is not within the valid range.
HI_ERR_VDEC_NOT_PERM	Stream receiving is not stopped or the channel is created or destroyed when being operated.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

- Before setting protocol parameters, ensure that the channel is created. Otherwise, an error code is returned, indicating that no channel is created.
- You cannot re-create or destroy the channel when calling this MPI.
- You must stop receiving streams before setting protocol parameters. Otherwise, HI_ERR_VDEC_NOT_PERM is returned.
- You are advised to call HI_MPI_VDEC_GetPrtclParam to obtain the default protocol parameters before setting the protocol parameters. The default value of **s32SCDBufSize** is recommended, which is the minimum value.

[Example]

None

[See Also]

None

HI_MPI_VDEC_GetPrtclParam

[Description]

Obtains the memory allocation parameters related to protocols in the decoding channel.

[Syntax]

```
HI_S32 HI_MPI_VDEC_GetPrtclParam(VDEC_CHN VdChn, VDEC_PRTCL_PARAM_S *
pstParam);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
pstParam	Channel parameters	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_UNEXIST	No channel exists.
HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized or the required .ko driver is not loaded.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

Before obtaining protocol parameters, ensure that the channel is created. Otherwise, an error code is returned, indicating that no channel is created.

[Example]

None

[See Also]

None

HI_MPI_VDEC_SetDisplayFrameNum

[Description]

Sets the number of display frame buffers.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
HI_S32 HI_MPI_VDEC_SetDisplayFrameNum(VDEC_CHN VdChn, HI_S32
s32DisplayFrameNum);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
s32DisplayFrameNum	Number of displayed frames Value range: [0, 16]	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_NOT_PERM	The operation is forbidden.
HI_ERR_VDEC_ILLEGAL_PARAM	The parameter is incorrect or is not within the valid range.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

- This MPI must be called before a VDEC channel is created; otherwise, the error code HI_ERR_VDEC_NOT_PERM is returned.
- The number of displayed frames ranges from 0 and 16, and the default value is 2.
- **s32DisplayFrameNum** can be set to **0** only when decoded pictures are obtained in user mode or the back-end modules bound to the VDEC do not buffer any decoded picture. Otherwise, inversion pressure occurs and decoding stops due to insufficiency of the frame buffer.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- The number of frame buffers for H.264 decoding channel or MPEG4 decoding channel is (Number of reference frames + Number of display frames + 1), and the number of frame buffers for JPEG decoding channel is (Number of display frames + 1).

[Example]

None

[See Also]

None

HI_MPI_VDEC_GetDisplayFrameNum

[Description]

Obtains the number of display frame buffers.

[Syntax]

```
HI_S32 HI_MPI_VDEC_GetDisplayFrameNum(VDEC_CHN VdChn, HI_S32
*ps32DisplayFrameNum);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
ps32DisplayFrameNum	Pointer to the number of displayed frames	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

This MPI can be called before or after a VDEC channel is created.

[Example]

None

[See Also]

None

HI_MPI_VDEC_SetChnPrevMode

[Description]

Sets the channel preview mode.

[Syntax]

```
HI_S32 HI_MPI_VDEC_SetChnPrevMode(VDEC_CHN VdChn, const HI_BOOL bPrevMode);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
bPrevMode	Whether the mode is the preview mode 0: no 1: yes	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_UNEXIST	No channel exists.

[Requirement]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

This MPI must be called after a VDEC channel is created; otherwise, the error code HI_ERR_VDEC_UNEXIST is returned, indicating that no channel is created.

[Example]

None

[See Also]

None

HI_MPI_VDEC_GetChnPrevMode

[Description]

Obtains the channel preview mode.

[Syntax]

```
HI_S32 HI_MPI_VDEC_GetChnPrevMode(VDEC_CHN VdChn, HI_BOOL *pbPrevMode);
```

[Parameter]

Parameter	Description	Input/Output
VdChn	VDEC channel ID Value range: [0, VDEC_MAX_CHN_NUM)	Input
pbPrevMode	Pointer to the preview mode	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 10.5 "Error Codes."

[Error Code]

Error Code	Definition
HI_SUCCESS	Success
HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
HI_ERR_VDEC_UNEXIST	No channel exists.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Requirement]

- Header files: mpi_vdec.h, hi_comm_vdec.h
- Library file: libmpi.a

[Note]

This MPI must be called after a VDEC channel is created; otherwise, the error code HI_ERR_VDEC_UNEXIST is returned, indicating that no channel is created.

[Example]

None

[See Also]

None

10.4 Data Types

The VDEC data types are as follows:

- VDEC_CHN_ATTR_S: Defines the attributes of a VDEC channel.
- VDEC_ATTR_JPEG_S: Defines the attributes of a VDEC channel complying with the JPEG decoding protocol.
- VDEC_ATTR_VIDEO_S: Defines the attributes of a VDEC channel complying with the protocols rather than the JPEG decoding protocol.
- VDEC_STREAM_S: Defines the decoded stream.
- VDEC_USERDATA_S: Defines the user private data.
- VDEC_CHN_STAT_S: Defines the status of a VDEC channel.
- VDEC_DECODE_ERROR_S: Defines the structure of the decoding error information.
- VIDEO_MODE_E: Defines the mode of sending streams.
- VDEC_CHN_PARAM_S: Defines the parameters of a VDEC channel.
- VDEC_PRTCL_PARAM_S: Defines the memory parameters related to protocols.
- VDEC_STD_EXTENSION_U: Defines whether the VC1 stream belongs to the advanced property (AP) and contains the version information.

VDEC_CHN_ATTR_S

[Description]

Defines the attributes of a VDEC channel.

[Syntax]

```
typedef struct hiVDEC_CHN_ATTR_S
{
    PAYLOAD_TYPE_E enType;
ER
   HI_U32      u32Priority;       /*Priority*/
    HI_U32      u32PicWidth;       /*Maximum picture width*/
    HI_U32      u32PicHeight;      /*Maximum picture height*/
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    union
    {
        VDEC_ATTR_JPEG_S stVdecJpegAttr;
        VDEC_ATTR_VIDEO_S stVdecVideoAttr;
    };
}VDEC_CHN_ATTR_S;
```

[Member]

Member	Description
enType	Decoding protocol type Static attribute
u32BufSize	Size of a stream buffer The stream buffer size is greater than or equal to 32 KB in memory reduction mode when streams are decoded by frame. In other cases, the stream buffer size is greater than or equal to 3/4 of the VDEC channel size (channel width x channel height). That is, the buffer size is greater than or equal to half of a YUV420 picture (channel width x channel height x 3/2 x 1/2). The buffer size is in the unit of byte. You are advised to set the buffer size to the size of a decoded YUV420 picture, that is, channel width x channel height x 1.5. Static attribute
u32Priority	Channel priority, ranging from 1 to 255. A larger value indicates a higher channel priority. Note: The value 0 indicates that the channel is not scheduled. The priority of the Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C VEDU is invalid, but decoding is available only when the member is set to 0.
u32PicWidth	Maximum width of the picture to be decoded (in pixel) Static attribute
u32PicHeight	Maximum height of the picture to be decoded (in pixel) Static attribute
stVdecJpegAttr	Attributes of a JPEG channel
stVdecVideoAttr	Attributes of other channels rather than the JPEG channel

[Note]

None

[See Also]

None

VDEC_ATTR_JPEG_S

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Defines the attributes of a VDEC channel complying with the JPEG decoding protocol.

[Syntax]

```
typedef struct hiVDEC_ATTR_JPEG_S
{
    VIDEO_MODE_E     enMode;
}VDEC_ATTR_JPEG_S,*PTR_VDEC_ATTR_JPEG_S;
```

[Member]

Member	Description
enMode	Mode of sending streams The JPEG streams can be sent by frame. Static attribute

[Note]

None

[See Also]

None

VDEC_ATTR_VIDEO_S

[Description]

Defines the attributes of a VDEC channel complying with the protocols rather than the JPEG decoding protocol.

[Syntax]

```
typedef struct hiVDEC_ATTR_VIDEO_S
{
     HI_U32           u32RefFrameNum  ;
     VIDEO_MODE_E     enMode;
     HI_S32           s32SupportBFrame;
}VDEC_ATTR_VIDEO_S;
```

[Member]

Member	Description
u32RefFrameNum	Number of reference frames, in frame Value range: [0, 16] The value determines the number of reference frames required during decoding. As the reference frames may occupy much memory, you can set the number of reference frames to a proper value as required. • Streams encoded by HiSilicon: The value 2 is recommended. • Other monitor streams: The value 5 is recommended.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	• Test streams: The value 16 is recommended. Static attribute
enMode	Mode of sending streams The streams can be sent by frame or by stream. Static attribute
s32SupportBFrame	Whether to support decoding of the B frame Value range: [0, 1]

[Note]

- When only I frames are decoded during the frame stream decoding, you can set **u32RefFrameNum** to **0** to reduce the required frame buffer.
- When only I fields are decoded during field stream decoding, decoding is successful only when **u32RefFrameNum** is set to **1**.

[See Also]

None

VDEC_STREAM_S

[Description]

Defines the decoded stream.

[Syntax]

```
typedef struct hiVDEC_STREAM_S
{
    HI_U8*  pu8Addr;
    HI_U32  u32Len;
    HI_U64  u64PTS;
}VDEC_STREAM_S;
```

[Member]

Member	Description
pu8Addr	Address of a stream packet
u32Len	Length of a stream packet, in byte
u64PTS	PTS of a stream packet, in μs

[Note]

- When streams are sent by frame, the PTS of the decoded picture is the same as that of the stream packet.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- When streams are sent by stream, the PTS of the decoded picture is 0.

[See Also]

None

VDEC_USERDATA_S

[Description]

Defines the user private data.

[Syntax]

```
typedef struct hiVDEC_USERDATA_S
{
   HI_U8*      pu8Addr;        /*Private data address*/
   HI_U32      u32PhyAddr;
   HI_U32      u32Len;         /*Private data len*/
HI_BOOL     bValid;        /*Whether it is valid*/}VDEC_USERDATA_S;
}VDEC_PRIDATA_S ;
```

[Member]

Member	Description
pu8Addr	Virtual address of the private data
u32PhyAddr	Physical address of the private data
u32Len	Length of the private data, in byte
bValid	Validity flag of the current private data Value: • HI_TRUE: valid • HI_FALSE: invalid

[Note]

- When user data fails to be obtained by calling the corresponding MPI in non-block mode, bvalid is HI_FALSE.
- When data is successfully obtained, bvalid is HI_TRUE.

[See Also]

None

VDEC_CHN_STAT_S

[Description]

Defines the status of a VDEC channel.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
typedef struct hiVDEC_CHN_STAT_S
{
    PAYLOAD_TYPE_E enType;      /* Decoding protocol */
    HI_U32  u32LeftStreamBytes; /* Number of streams to be decoded (in byte)*/
    HI_U32 u32LeftStreamFrames; /*Number of frames to be decoded*/
    HI_U32  u32LeftPics;        /*Number of pictures to be obtained */
    HI_BOOL bStartRecvStream;   /*Whether to start to receive streams*/
     HI_U32 u32RecvStreamFrames;  /*Number of received frames*/
    HI_U32 u32DecodeStreamFrames; /*Number of decoded frames*/
VDEC_DECODE_ERROE_S stVdecDecErr;  /* Decoding error information */
}VDEC_CHN_STAT_S;
```

[Member]

Member	Description
enType	Decoding protocol type
u32LeftStreamBytes	Number of streams to be decoded in the stream buffer, in byte
u32LeftStreamFrames	Number of frames to be decoded in the stream buffer The value –1 indicates invalid. This parameter takes effect only when streams are sent by frame.
u32LeftPics	Number of remaining pictures in the picture buffer
bStartRecvStream	Whether the decoder starts to receive streams
u32RecvStreamFrames	Number of received frames in the stream buffer The value –1 indicates invalid. This parameter takes effect only when streams are sent by frame.
u32DecodeStreamFrames	Number of decoded frames in the stream buffer The value –1 indicates invalid. This parameter takes effect only when streams are sent by frame. u32RecvStreamFrames = u32DecodeStreamFrames + u32LeftStreamFrames
stVdecDecErr	Decoding error information

[Note]

None

[See Also]

None

Copyright © HiSilicon Technologies Co., Ltd

VIDEO_MODE_E

[Description]

Defines the mode of sending streams.

[Syntax]

```
typedef enum hiVIDEO_MODE_E
{
    VIDEO_MODE_STREAM = 0,
    VIDEO_MODE_FRAME,
    VIDEO_MODE_BUTT
}VIDEO_MODE_E;
```

[Member]

Member	Description
VIDEO_MODE_STREAM	Streams are sent by stream
VIDEO_MODE_FRAME	Streams are sent by frame, in frame

[Note]

None

[See Also]

None

VDEC_DECODE_ERROR_S

[Description]

Defines the structure of the decoding error information.

[Syntax]

```
typedef struct hi_VDEC_DECODE_ERROR_S
{
   HI_S32 s32FormatErr;          /* Format not supported */
   HI_S32 s32PicSizeErrSet;      /* Picture width (or height) is greater than that of the channel
   */
   HI_S32 s32StreamUnsprt;       /* Specification not supported */
   HI_S32 s32PackErr;            /* Stream error */
   HI_S32 s32PrtclNumErrSet;     /* Number of protocol parameters set is insufficient. */
   HI_S32 s32RefErrSet;          /* Number of reference frames set is insufficient. */
   HI_S32 s32PicBufSizeErrSet;   /* Picture buffer size is insufficient. */
}VDEC_DECODE_ERROR_S;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
s32FormatErr	Format not supported. For example, H.264 field stream and Progressive stream of JPEG.
s32PicSizeErrSet	The width (or height) of the picture is greater than that of the channel.
s32StreamUnsprt	Specification not supported. (The stream specification is inconsistent with the nominal specification of the chip.)
s32PackErr	Stream error exists.
s32PrtclNumErrSet	The number of protocol parameters set is insufficient. For example, the number of Slice, PPS, and SPS parameters. (This parameter only takes effect for H.264 decoding.)
s32RefErrSet	The number of reference frames set is insufficient. (This parameter does not take effect for JPEG decoding.)
s32PicBufSizeErrSet	The picture buffer size is insufficient during decoding of JPEG pictures in YUV444 format. The picture buffer size can be increased by enlarging the width or height of the decoding channel.

[Note]

- All error information is presented in accumulative counting mode. For example, the s32PackErr value increases by 1 every time when a stream error is found during decoding.
- The count of all error information is set to 0 after the channel is reset.

[See Also]

None

VDEC_CHN_PARAM_S

[Description]

Defines the parameters of a VDEC channel.

[Syntax]

```
typedef struct hiVDEC_CHN_PARAM_S
{
    VDEC_STD_EXTENSION_U    StdExt;
    HI_S32  s32ChanErrThr;
    HI_S32  s32ChanStrmOFThr;
    HI_S32  s32DecMode;
    HI_S32  s32DecOrderOutput;
    HI_S32  s32DnrTfEnable;
    HI_S32  s32DnrDispOutEnable;
    HI_U32  u32MaxFramesInDec;
}VDEC_CHN_PARAM_S;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Member]

Member	Description
StdExt	Extended information. It defines whether the VC1-related streams contain the AP and version information.
s32ChanErrThr	Error threshold, ranging from 0 to 100. The value 0 indicates the frames are discarded when an error occurs, and the value 100 indicates that decoding continues regardless of the number of errors. The decoder calculates the error ratio of each frame as follows: Number of error macroblocks/Total number of macroblocks x 100%. If the error ratio of a frame is greater than s32ChanErrThr, this frame is discarded. If the frame is a reference frame, its subsequent frames are all discarded until the next I frame occurs. Default value: 30 This parameter is valid only for the channels rather than the JPEG VDEC channel.
s32ChanStrmOFThr	Threshold for discarding frames before decoding. The threshold must be greater than 0. When the decoder works, it detects whether the front-end stream buffer nearly overflows. If the buffer nearly overflows, the decoder stops decoding and consumes streams rapidly until the number of streams is not greater than s32ChanStrmOFThr. When the number of streams is greater than s32ChanStrmOFThr, the decoder considers that the buffer nearly overflows. You can set s32ChanStrmOFThr to 0 to prevent streams being discarded by the decoder. Default value: 0 This parameter is valid only for decoding channels except JPEG and is valid only in IPB decoding mode.
s32DecMode	Decoding mode 0: normal mode (the I frame, P frame, or B frame can be decoded) 1: IP mode (the B frame that is not the reference frame is discarded without being decoded) 2: I mode (the I frame is decoded and the P frame and B frame are discarded) Default value: 1 This parameter is valid only for the channels rather than the JPEG VDEC channel.
s32DecOrderOutput	Output sequence of the decoded pictures 0: output in the display sequence 1: output in the decoding sequence Default value: 1

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	This parameter is valid only for the channels rather than the JPEG VDEC channel.
s32DnrTfEnable	Reserved
s32DnrDispOutEnable	Reserved

[Note]

When pictures are decoded by using the Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C VEDU, only the operation of checking pictures correctness is supported. If the value range of s32ChanErrThr is [0, 10), incorrect decoded pictures are discarded. If the value range of s32ChanErrThr is [10, 100], no decoded pictures are discarded.

[See Also]

None

VDEC_PRTCL_PARAM_S

[Description]

Defines the memory parameters related to protocols.

[Syntax]

```
typedef struct hiVDEC_PRTCL_PARAM_S
{
    HI_S32  s32MaxSliceNum;
    HI_S32  s32MaxSpsNum;
    HI_S32  s32MaxPpsNum;
    HI_S32  s32SCDBufSize;
    HI_S32  s32DisplayFrameNum;
}VDEC_PRTCL_PARAM_S;
```

[Member]

Member	Description
s32MaxSliceNum	Maximum number of slices supported by a channel. The default value is 16, and this parameter is valid only for the H.264 channel. Value Range: [1, 136]
s32MaxSpsNum	Maximum number of sequence parameter sets (SPSs) during decoding. The default value is 2, and this parameter is valid only for the H.264 channel. Value Range: [1, 32]
s32MaxPpsNum	Maximum number of picture parameter sets (PPSs) during decoding. The default value is 2, and this parameter is valid only for the H.264 channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	Value Range: [1, 256]
s32SCDBufSize	Buffer size of the stream splitting module. This parameter is invalid for JPEG or MJPEG. Value Range: • In common mode, the buffer size is greater than or equal to the larger value between 256 x 1024 and (channel width x channel height x 3/2). • When streams are decoded by frame in memory reduction mode, the buffer size is greater than or equal to the larger value between 32 x 1024 and **u32BufSize**. The value of **u32BufSize** is the configured stream buffer size.
s32DisplayFrameNum	Minimum number of buffered frames to be displayed and decoded. The default value is 2. Value range: [0, 16]

[Note]

s32DisplayFrameNum can be set to **0** only when decoded pictures are obtained in user mode or the back-end modules bound to the VDEC do not buffer any decoded picture. Otherwise, inversion pressure occurs and decoding stops due to insufficiency of the frame buffer.

[See Also]

None

VDEC_STD_EXTENSION_U

[Description]

Defines whether the VC1 stream belongs to the AP and contains the version information.

[Syntax]

```
typedef union
{
    struct
    {
        HI_S32 IsAdvProfile;
        HI_S32 CodecVersion;
    } Vc1Ext;
    struct
    {
       HI_S32 bReversed;
    } Vp6Ext;
} VDEC_STD_EXTENSION_U;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
IsAdvProfile	Whether the VC1-related streams belong to AP (VC1 extended information) Default value: 0
CodecVersion	Version information (VC1 extended information) Default value: 0
bReversed	This parameter is set to 1 when a picture needs to be reversed (Vp6Ext extended information). Default value: 0

[Note]

None

[See Also]

None

10.5 Error Codes

Table 10-3 describes the error codes of VDEC APIs.

Table 10-3 Error codes of VDEC APIs

Error Code	Macro Definition	Description
0xA0058001	HI_ERR_VDEC_INVALID_DEVID	The device ID is invalid.
0xA0058002	HI_ERR_VDEC_INVALID_CHNID	The channel ID is invalid.
0xA0058003	HI_ERR_VDEC_ILLEGAL_PARAM	The parameter value is invalid.
0xA0058004	HI_ERR_VDEC_EXIST	The device, channel, or resource to be applied for or created already exists.
0xA0058005	HI_ERR_VDEC_UNEXIST	The device, channel, or resource being used or destroyed does not exist.
0xA0058006	HI_ERR_VDEC_NULL_PTR	The pointer is null.
0xA0058007	HI_ERR_VDEC_NOT_CONFIG	The system or VDA channel is not configured.
0xA0058008	HI_ERR_VDEC_NOT_SUPPORT	The parameter or function is not supported.
0xA0058009	HI_ERR_VDEC_NOT_PERM	The operation, for example, attempting to modify the value

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Macro Definition	Description
		of a static parameter, is forbidden.
0xA005800C	HI_ERR_VDEC_NOMEM	The memory fails to be allocated due to the causes such as insufficient system memory.
0xA005800D	HI_ERR_VDEC_NOBUF	The buffer fails to be allocated due to some causes. For example, the data buffer applied for is too large.
0xA005800E	HI_ERR_VDEC_BUF_EMPTY	The buffer is empty.
0xA005800F	HI_ERR_VDEC_BUF_FULL	The buffer is full.
0xA0058010	HI_ERR_VDEC_SYS_NOTREADY	The system is not initialized.
0xA0058011	HI_ERR_VDEC_BADADDR	The stream address is incorrect.
0xA0058012	HI_ERR_VDEC_BUSY	The system is busy.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Contents

11 IVE 11-1
11.1 Overview 11-1
11.2 Function Description 11-1
11.2.1 Concepts 11-1
11.2.2 Usage Guidelines 11-6
11.3 API Reference 11-6
11.4 Data Types 11-29
11.5 Error Codes 11-40

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figures

Figure 11-1 IVE_SRC_FMT_SINGLE format (single-component format) 11-2
Figure 11-2 IVE_SRC_FMT_SP420 format 11-3
Figure 11-3 IVE_SRC_FMT_SP422 format 11-3
Figure 11-4 IVE_CSC_OUT_FMT_PLANAR format 11-4
Figure 11-5 IVE_CSC_OUT_FMT_PACKAGE format 11-4
Figure 11-6 Sobel H/V output format 11-5
Figure 11-7 Canny output format (MAG and ANG) 11-5
Figure 11-8 Integral output format 11-6
Figure 11-9 Histogram output format 11-6

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Tables

Table 11-5 Error codes for the IVE 11-40

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

11 IVE

11.1 Overview

The intelligent video engine (IVE) is a hardware acceleration module in the intelligent analysis system. By using the IVE, the ARM can perform intelligent analysis rapidly. The IVE provides 14 basic operators, including direct memory access (DMA), template filter, color space conversion (CSC), template filter+CSC, sobel edge extraction, canny edge extraction, dilate, erode, thresh, AND, subtract, OR, thresh, integral statistics, and histogram statistics.

11.2 Function Description

11.2.1 Concepts

Note the following:

- Stride

  Number of units for measuring a picture row by pixel. The pixel bit width can be 8 bits or 16 bits.

- Stride calculation based on the 8-byte-aligned address of the data source

  The input and output addresses must be 8-byte-aligned for the operators template filter, CSC, template filter+CSC, sobel edge extraction, canny edge extraction, dilate, erode, integral statistics, and histogram statistics. When you configure the strides of these operators, ensure that each stride is 8-pixel-aligned and each stride is greater than or equal to the source data width. The related equations are as follows:

$$\begin{cases} stride \geq width \\ stride\%8 = 0 \end{cases}$$

- Stride calculation based on the byte-aligned address of the data source

  When configuring the strides of the operators DMA, thresh, AND, subtract, and OR, you must consider the address of the source data, data width, and 8-byte alignment. The related equations are as follows:

$$\begin{cases} stride \geq \{(8-((width+(src\%8))\%8))\%8+(src\%8)+width\} \\ stride\%8 = 0 \end{cases}$$

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- bInstant

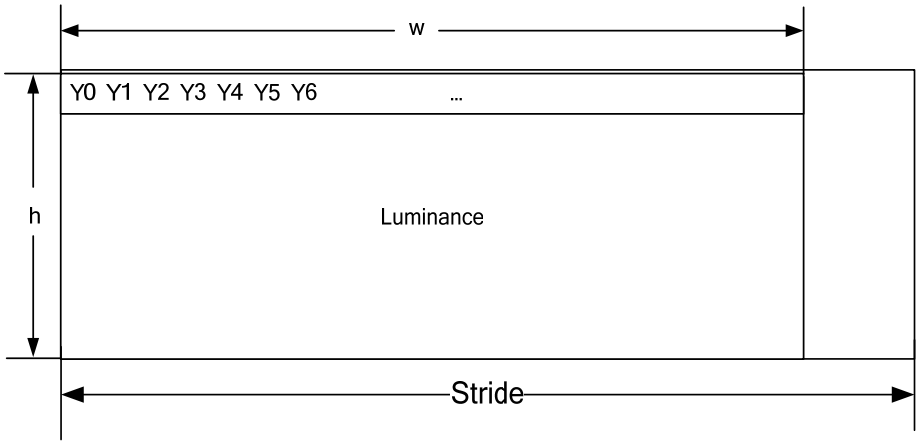
  bInstant is a flag indicating whether to return results in time. If you want to obtain the completion information about a task, you must set bInstant to HI_TRUE when creating this task. If you do not concern about the completion information, you are advised to set bInstant to HI_FALSE.

- Flush cache

  The IVE can obtain data only from the DDR. If the CPU has accessed a cacheable space when users call an IVE task, users need to call HI_MPI_SYS_MmzFlushCache to flush data from the cache to the DDR. This ensures that the input and output data of the IVE is not interfered by the CPU cache.

- Input data formats
  - IVE_SRC_FMT_SINGLE: It indicates a single-component format, and the data of this format is stored as an unsigned number in byte. It can be any value such as Y, Cb, Cr, R, G, or B. The value range is [0, 255]. See Figure 11-1.
  - IVE_SRC_FMT_SP420: It indicates the YCbCr420_SemiPlanar format. See Figure 11-2.
  - IVE_SRC_FMT_SP422: It indicates the YCbCr422_SemiPlanar format. See Figure 11-3.

Figure 11-1 IVE_SRC_FMT_SINGLE format (single-component format)



HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 11-2 IVE_SRC_FMT_SP420 format

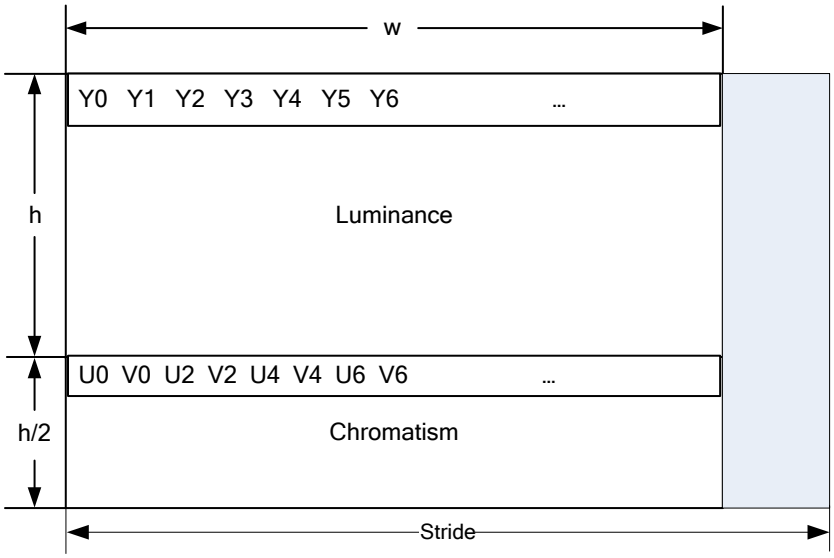
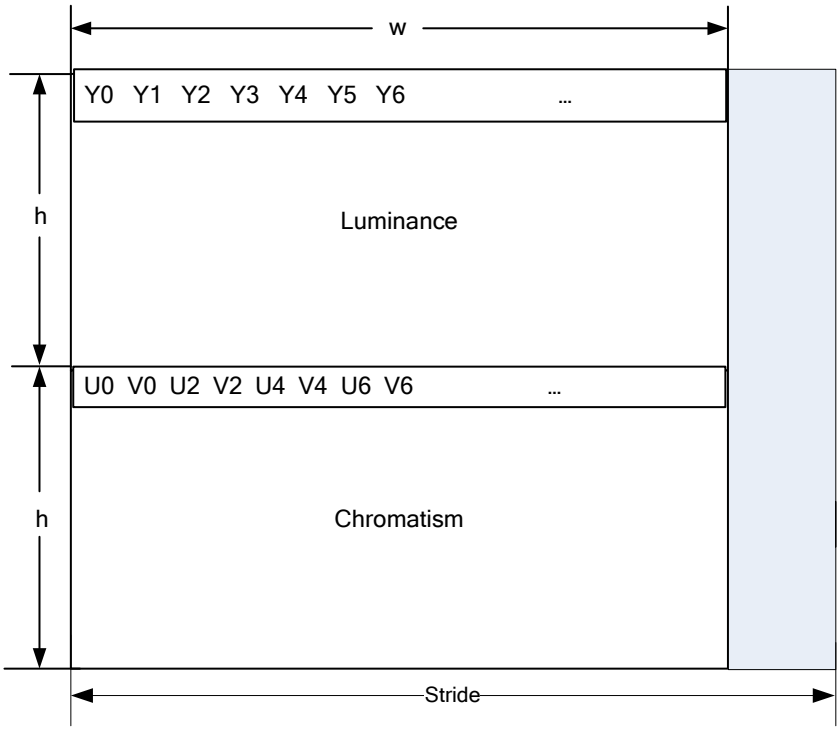


Figure 11-3 IVE_SRC_FMT_SP422 format



- Output data formats
  - The single-component output format is the same as the single-component input format.
  - IVE_CSC_OUT_FMT_PLANAR. See Figure 11-4.
  - IVE_CSC_OUT_FMT_PACKAGE. See Figure 11-5.
  - Sobel output H/V format. See Figure 11-6.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Canny output format (MAG and ANG). See Figure 11-7.
- Integral output format. In dual words, S (picture sum) occupies lower 28 bits, and SQ (sum of squares) occupies upper 36 bits. See Figure Figure 11-8.
- Histogram output format. See Figure Figure 11-9.

Figure 11-4 IVE_CSC_OUT_FMT_PLANAR format

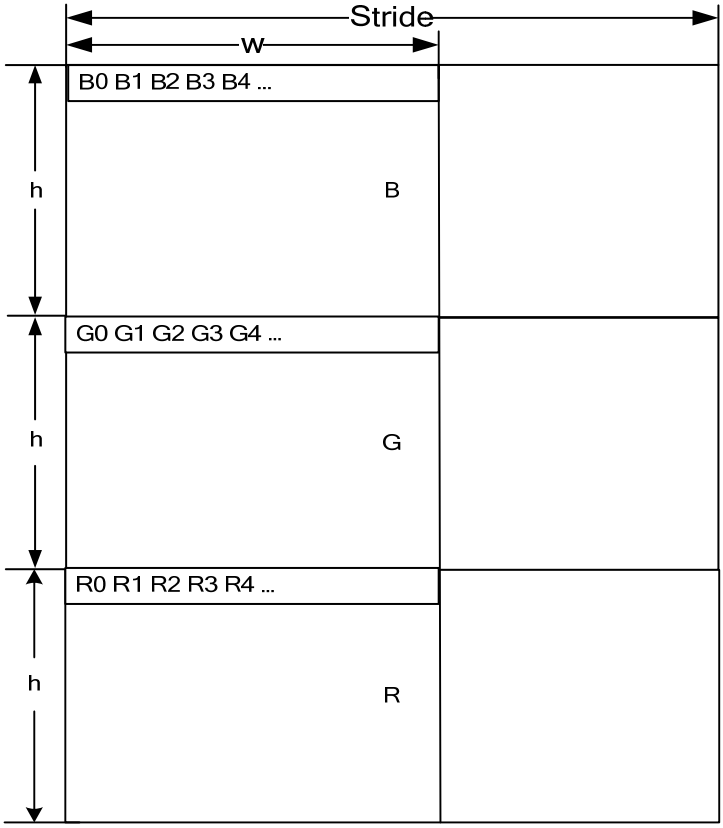
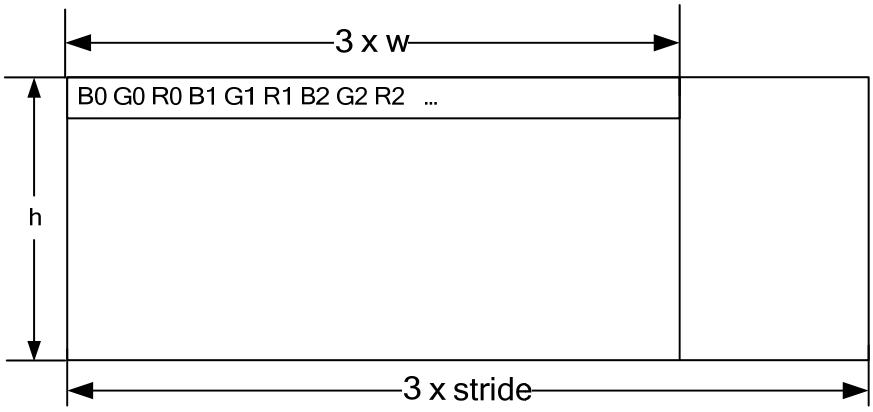


Figure 11-5 IVE_CSC_OUT_FMT_PACKAGE format



HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

NOTE

Data is stored in the memory in little endian mode. The storage unit of word or double word is used in the following figures.

Figure 11-6 Sobel H/V output format

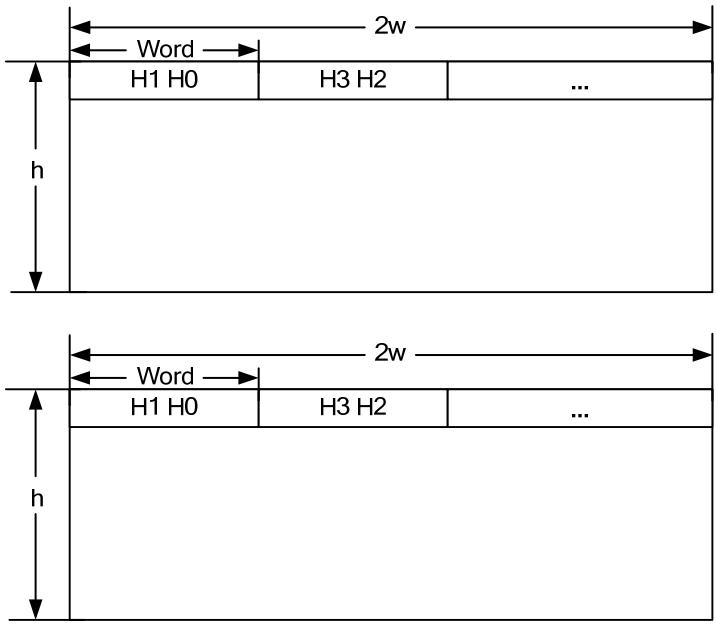
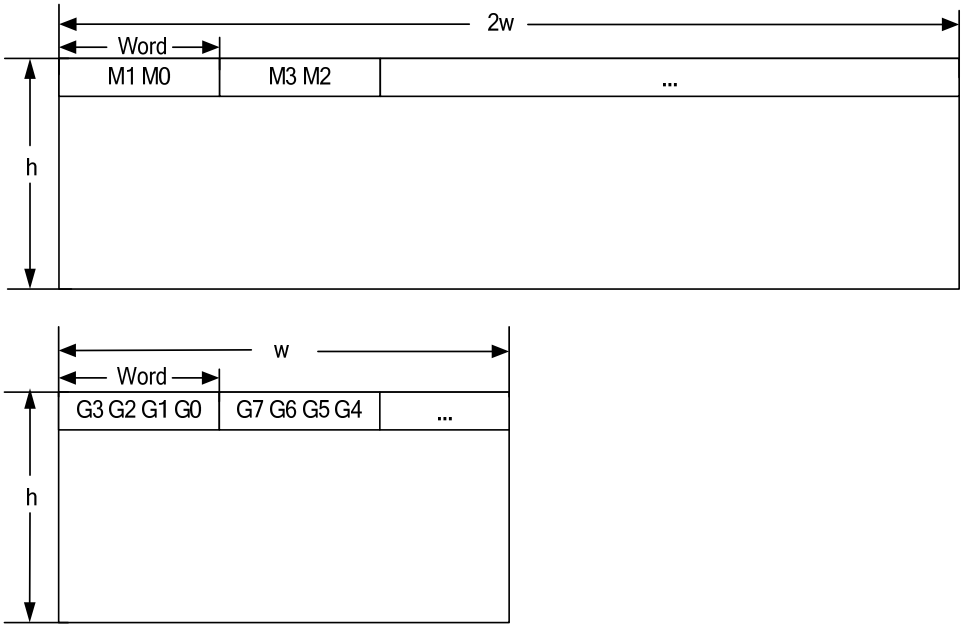


Figure 11-7 Canny output format (MAG and ANG)



HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Figure 11-8 Integral output format

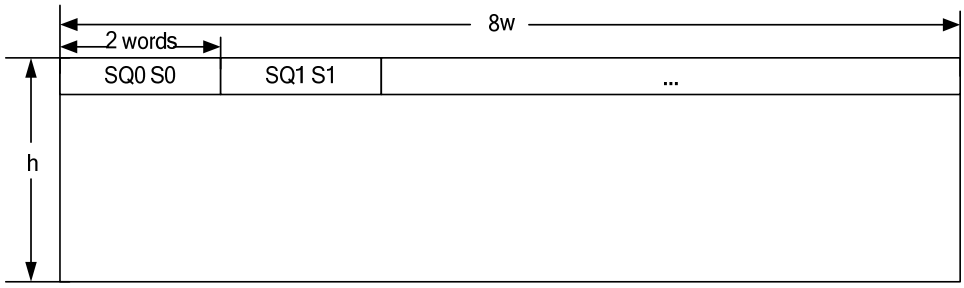
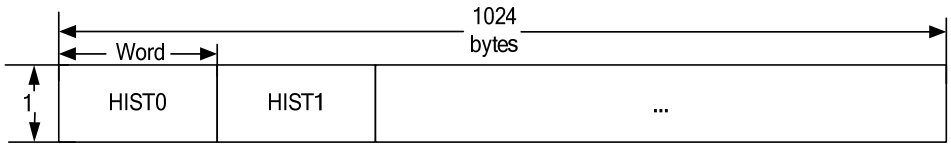


Figure 11-9 Histogram output format



11.2.2 Usage Guidelines

Follow the following guidelines:

- Call corresponding operator MPIs to specify the value of bInstant, create a task, and record the returned handle ID of the task as required.
- Specify the block mode based on the returned handle ID to query the completion status of a task.

11.3 API Reference

The IVE provides the following MPIs:

- HI_MPI_IVE_DMA: Creates a DMA task.
- HI_MPI_IVE_FILTER: Creates a template filter task.
- HI_MPI_IVE_CSC: Creates a CSC task.
- HI_MPI_IVE_FILTER_AND_CSC: Creates a task combined with template filter and CSC.
- HI_MPI_IVE_SOBEL: Creates a sobel edge extraction task.
- HI_MPI_IVE_CANNY: Creates a canny edge extraction task.
- HI_MPI_IVE_DILATE: Creates a dilate task.
- HI_MPI_IVE_ERODE: Creates an erode task.
- HI_MPI_IVE_THRESH: Creates a thresh task.
- HI_MPI_IVE_AND: Creates an AND task for two pictures.
- HI_MPI_IVE_SUB: Creates a subtract task for two pictures.
- HI_MPI_IVE_OR: Creates an OR task for two pictures.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_IVE_INTEG: Creates an integral statistics task.
- HI_MPI_IVE_HIST: Creates a histogram statistics task.
- HI_MPI_IVE_Query: Queries the completion status of an existing task.

HI_MPI_IVE_DMA

[Description]

Creates a DMA task for transferring data from a double-data rate (DDR) to another DDR.

[Syntax]

```
HI_S32 HI_MPI_IVE_DMA(IVE_HANDLE *pIveHandle, IVE_SRC_INFO_S *pstSrc,
IVE_MEM_INFO_S *pstDst, HI_BOOL bInstant);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle ID.	Output
pstSrc	Pointer to the source data.	Input
pstDst	Pointer to the output data.	Input
bInstant	Flag indicating whether results need to be returned in time.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 11.5 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

[Note]

- The handle ID is returned by the system.
- The pointers to the source data and output data cannot be null.
- The source data format must be single component.
- It is only required that the addresses of the source data and output data are aligned by bytes.
- The source data size (width x height) ranges from 32x1 pixels to 1920x1080 pixels.
- The output data stride must be configured based on stride requirements in section 11.2.1 "Concepts."

[Formula]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

$I_{out}(x,y) = I(x,y) \qquad (0 \le x \le width, 0 \le y \le height)$

$I(x,y)$ is defined by pstSrc, and $I_{out}(x,y)$ is defined by pstDst.

[Example]

```
HI_S32 s32Ret = HI_SUCCESS;
IVE_HANDLE IveHandle;
IVE_SRC_INFO_S stSrc;
IVE_MEM_INFO_SstDst;
HI_VOID *pVirtSrc;
HI_VOID *pVirtDst;

stSrc.stSrcMem.u32Stride  = 352;
stSrc.u32Height           = 288;
stSrc.u32Width            = 352;
stSrc.enSrcFmt            = IVE_SRC_FMT_SINGLE;
stDst.u32Stride           = 352;
bInstant                  = HI_TRUE;

s32Ret = HI_MPI_SYS_MmzAlloc_Cached (&stSrc.stSrcMem.u32PhyAddr,
&pVirtSrc,
    "User", HI_NULL, stSrc.u32Height * stSrc.stSrcMem.u32Stride);
memset(pVirtSrc,1,stSrc.u32Height * stSrc.stSrcMem.u32Stride);

s32Ret = HI_MPI_SYS_MmzAlloc_Cached (&stDst.u32PhyAddr, &pVirtDst,
    "User", HI_NULL, stSrc.u32Height * stDst.u32Stride);

s32Ret = HI_MPI_IVE_DMA(&IveHandle, &stSrc, &stDst, bInstant);

HI_MPI_SYS_MmzFree(stSrc.stSrcMem.u32PhyAddr, pVirtSrc);
HI_MPI_SYS_MmzFree(stDst.u32PhyAddr, pVirtDst);

return s32Ret;
```

[See Also]

None

HI_MPI_IVE_FILTER

[Description]

Creates a template filter task. You can set different template coefficients to implement various filtering effects.

[Syntax]

```
HI_S32 HI_MPI_IVE_FILTER(IVE_HANDLE *pIveHandle, IVE_MEM_INFO_S *pstSrc,
```


Copyright © HiSilicon Technologies Co., Ltd

```
IVE_MEM_INFO_S *pstDst, IVE_FILTER_CTRL_S *pstFilterCtrl,HI_BOOL
bInstant);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle ID.	Output
pstSrc	Pointer to the source data.	Input
pstDst	Pointer to the output data.	Input
pstFilterCtrl	Pointer to the control information.	Input
bInstant	Flag indicating whether results need to be returned in time.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 11.5 "Error Codes."

[Requirement]

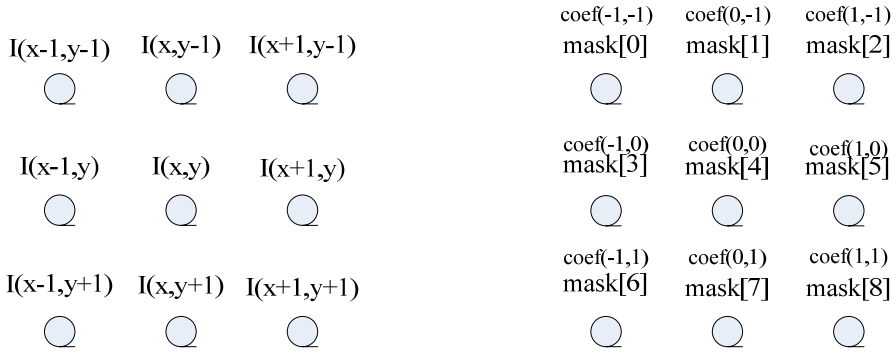
- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

[Note]

- The handle ID is returned by the system.
- The pointers to the source data, output data, and control information cannot be null.
- The source data format can be single component, SP420, or SP422.
- The addresses of the source data and output data must be 8-byte-aligned.
- The source data size (width x height) ranges from 64x64 pixels to 1920x1024 pixels.
- When the source data format is SP420 or SP422, the width must be an even number.
- The output data stride must be configured based on stride requirements in section 11.2.1 "Concepts."

[Formula]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd



$$Iout(x,y)=\left\{\sum_{-2<j<2}\sum_{-2<i<2} I(x+i,y+j)\bullet coef(i,j)\right\} >> \text{norm}$$

Note: $I(x,y)$ is defined by pstSrc, $I_{out}(x,y)$ is defined by pstDst, $coef$ (mask) is defined by as8Mask[9] of pstCscCtrl, and norm is defined by u8Norm of pstCscCtrl.

[Example]

None

[See Also]

None

HI_MPI_IVE_CSC

[Description]

Creates a CSC task for converting a YUV space into an RGB space.

[Syntax]

```
HI_S32 HI_MPI_IVE_CSC(IVE_HANDLE *pIveHandle,IVE_SRC_INFO_S *pstSrc,
IVE_MEM_INFO_S *pstDst, IVE_CSC_CTRL_S *pstCscCtrl,HI_BOOL bInstant);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle ID.	Output
pstSrc	Pointer to the source data.	Input
pstDst	Pointer to the output data.	Input
pstCscCtrl	Pointer to the control information.	Input
bInstant	Flag indicating whether results need to be returned in time.	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success.
Other values	Failure. For details, see section 11.5 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

[Note]

- The handle ID is returned by the system.
- The pointers to the source data, output data, and control information cannot be null.
- The source data format can be SP420 or SP422.
- The addresses of the source data and output data must be 8-byte-aligned.
- The source data size (width x height) ranges from 64x64 pixels to 1920x1080 pixels.
- The source data with must be an even number.
- The output data stride must be configured based on stride requirements in section 11.2.1 "Concepts." The size of the output data buffer must be calculated as follows: Source picture height x Output data stride x 3
- Two output formats are supported. For details, see IVE_CSC_OUT_FMT_E
- Four operating modes are supported. For details, see IVE_CSC_MODE_E.

[Formula]

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = A_{CSC} \begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} + B_{CSC}$$

Note: Input picture $\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix}$ is defined by pstSrc, output picture $\begin{bmatrix} R \\ G \\ B \end{bmatrix}$ is defined by enOutFmt of pstDst and pstCscCtrl, and A_{CSC} and B_{CSC} depend on enCscMode of pstCscCtrl.

[Example]

None

[See Also]

None

HI_MPI_IVE_FILTER_AND_CSC

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Creates a task combined with template filter and CSC. In this way, two functions are achieved at one time.

[Syntax]

```
HI_S32 HI_MPI_IVE_FILTER_AND_CSC(IVE_HANDLE *pIveHandle, IVE_SRC_INFO_S
*pstSrc, IVE_MEM_INFO_S *pstDst, IVE_FILTER_AND_CSC_CTRL_S
*pstFltCscCtrl,HI_BOOL bInstant);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle ID.	Output
pstSrc	Pointer to the source data.	Input
pstDst	Pointer to the output data.	Input
pstFltCscCtrl	Pointer to the control information.	Input
bInstant	Flag indicating whether results need to be returned in time.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 11.5 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

[Note]

- The handle ID is returned by the system.
- The pointers to the source data, output data, and control information cannot be null.
- The source data format can be SP420 or SP422.
- The addresses of the source data and output data must be 8-byte-aligned.
- The source data size (width x height) ranges from 64x64 pixels to 1920x1024 pixels.
- The source data with must be an even number.
- The output data stride must be configured based on stride requirements in section 11.2.1 "Concepts." The size of the output data buffer must be calculated as follows: Source picture height x Output data stride x 3
- Two output formats are supported. For details, see IVE_CSC_OUT_FMT_E.
- Four operating modes are supported. For details, see IVE_CSC_MODE_E

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[See Also]

None

HI_MPI_IVE_SOBEL

[Description]

Creates a sobel edge extraction task.

[Syntax]

```
HI_S32 HI_MPI_IVE_SOBEL(IVE_HANDLE *pIveHandle, IVE_SRC_INFO_S *pstSrc,
IVE_MEM_INFO_S *pstDstH, IVE_MEM_INFO_S *pstDstV,IVE_SOBEL_CTRL_S
*pstSobelCtrl,HI_BOOL bInstant);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle ID.	Output
pstSrc	Pointer to the source data.	Input
pstDstH	Pointer to the horizontal output data.	Input
pstDstV	Pointer to the vertical output data.	Input
pstSobelCtrl	Pointer to the control information.	Input
bInstant	Flag indicating whether results need to be returned in time.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 11.5 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

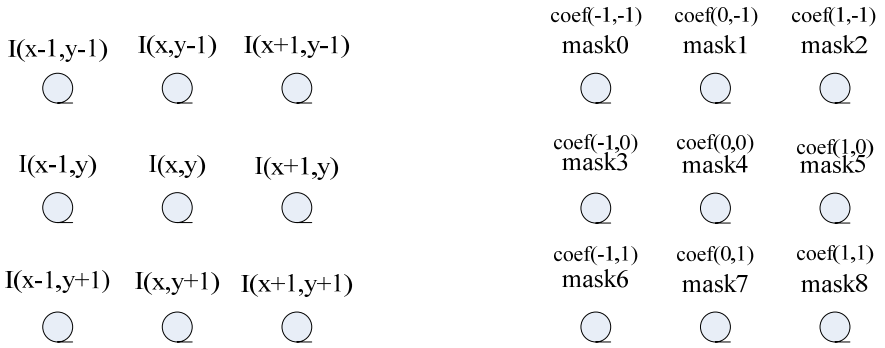
[Note]

- The handle ID is returned by the system.
- The pointer to the source data, horizontal output data, vertical output data, and control information cannot be null.
- The source data format must be single component.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- The addresses of the source data and output data must be 8-byte-aligned.
- The source data size (width x height) ranges from 64x64 pixels to 1920x1024 pixels.
- The output data stride must be configured based on stride requirements in section 11.2.1 "Concepts." The size of the buffer for storing the data output horizontally and vertically must be calculated as follows: Source picture height x Output data stride x 2

[Formula]



$$Hout(x,y)=\sum_{-2<j<2}\sum_{-2<i<2}I(x+i,y+j)\bullet coef(i,j)$$

$$Vout(x,y)=\sum_{-2<j<2}\sum_{-2<i<2}I(x+i,y+j)\bullet coef(j,i)$$

Note: $I(x,y)$ is defined by pstSrc, $Hout(x,y)$ is defined by pstDstH, $Vout(x,y)$ is defined by pstDstV, and $coef$ (mask) is defined by as8Mask[9] of pstSobelCtrl.

[Example]

None

[See Also]

None

HI_MPI_IVE_CANNY

[Description]

Creates a canny edge extraction task.

[Syntax]

```
HI_S32 HI_MPI_IVE_CANNY(IVE_HANDLE *pIveHandle, IVE_SRC_INFO_S *pstSrc,
IVE_MEM_INFO_S *pstDstMag, IVE_MEM_INFO_S *pstDstAng,IVE_CANNY_CTRL_S
*pstCannyCtrl,HI_BOOL bInstant);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle ID.	Output

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
pstSrc	Pointer to the source data.	Input
pstDstMag	Pointer to the output magnitude.	Input
pstDstAng	Pointer to the output angle.	Input
pstCannyCtrl	Pointer to the control information.	Input
bInstant	Flag indicating whether results need to be returned in time.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 11.5 "Error Codes."

[Requirement]

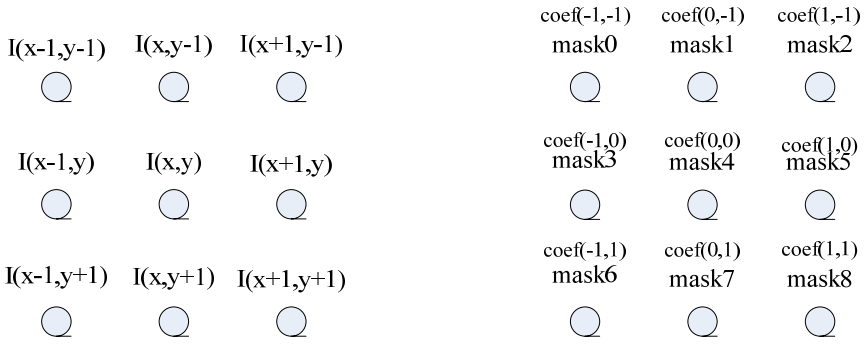
- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

[Note]

- The handle ID is returned by the system.
- Two output formats are supported. For details, see IVE_CANNY_OUT_FMT_E.
- When only magnitude is output, the pointer to the source data, output magnitude, and control information cannot be null.
- When both magnitude and angle are output, the pointer to the source data, output magnitude, output angle, and control information cannot be null.
- The source data format must be single component.
- The addresses of the source data and output data must be 8-byte-aligned.
- The source data size (width x height) ranges from 64x64 pixels to 1920x1024 pixels.
- The output data stride must be configured based on stride requirements in section 11.2.1 "Concepts." The size of the output magnitude data buffer must be calculated as follows: Source picture height x Output data stride x 2
- The size of the output angle data buffer must be calculated as follows: Source picture height x Output data stride

[Formula]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd



$$Hout(x,y)=\sum_{-2<j<2}\sum_{-2<i<2}I(x+i,y+j)\bullet coef(i,j)$$

$$Vout(x,y)=\sum_{-2<j<2}\sum_{-2<i<2}I(x+i,y+j)\bullet coef(j,i)$$

$$Mag(x,y)=abs(Hout(x,y))+abs(Vout(x,y))$$

$$\theta=\left[\frac{\arctan(\frac{V}{H})*12}{\pi}\right]$$

Note: $I(x,y)$ is defined by pstSrc, $Mag(x,y)$ is defined by pstDstMag, $\theta(x,y)$ is defined by pstDstAng, and $coef$ (mask) is defined by as8Mask[9] of pstCannyCtrl.

[Example]

None

[See Also]

None

HI_MPI_IVE_DILATE

[Description]

Creates a dilate task.

[Syntax]

```
HI_S32 HI_MPI_IVE_DILATE(IVE_HANDLE *pIveHandle, IVE_SRC_INFO_S *pstSrc,
IVE_MEM_INFO_S *pstDst, IVE_DILATE_CTRL_S *pstDilateCtrl,HI_BOOL
bInstant);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle ID.	Output

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
pstSrc	Pointer to the source data.	Input
pstDst	Pointer to the output data.	Input
pstDilateCtrl	Pointer to the control information.	Input
bInstant	Flag indicating whether results need to be returned in time.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 11.5 "Error Codes."

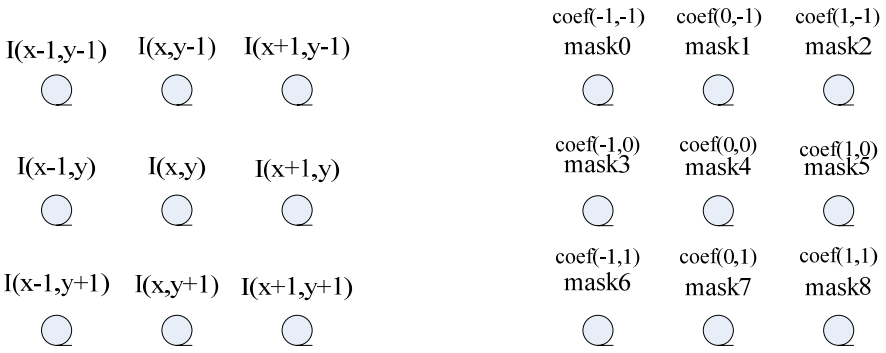
[Requirement]

- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

[Note]

- The handle ID is returned by the system.
- The pointers to the source data, output data, and control information cannot be null.
- The source data format must be single component.
- The addresses of the source data and output data must be 8-byte-aligned.
- The source data size (width x height) ranges from 64x64 pixels to 1920x1024 pixels.
- The output data stride must be configured based on stride requirements in section 11.2.1 "Concepts." The template coefficient must be 0 or 255.

[Formula]



$$Iout(x,y) = f(I(x+(k\,\&\,3)-1, y+(k\%3)-1)\,\&\,coef((k\,\&\,3)-1,(k\%3)-1),|,0,8)$$

$$f(A_k, \Theta, c_{min}, c_{max}) = A_{c_{min}} \Theta A_{c_{min+1}} \cdots \Theta A_{c_{max}}$$

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Note: In the preceding formula, | indicates bitwise OR operation, & indicates bitwise AND operation, and % indicates REM operation.

$I(x,y)$ is defined by pstSrc, $I_{out}(x,y)$ is defined by pstDst, and $coef$ (mask) is defined by au8Mask[9] of pstDilateCtrl.

[Example]

None

[See Also]

None

HI_MPI_IVE_ERODE

[Description]

Creates an erode task.

[Syntax]

```
HI_S32 HI_MPI_IVE_ERODE(IVE_HANDLE *pIveHandle, IVE_SRC_INFO_S *pstSrc,
IVE_MEM_INFO_S *pstDst, IVE_ERODE_CTRL_S *pstErodeCtrl,HI_BOOL bInstant);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle ID.	Output
pstSrc	Pointer to the source data.	Input
pstDst	Pointer to the output data.	Input
pstErodeCtrl	Pointer to the control information.	Input
bInstant	Flag indicating whether results need to be returned in time.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 11.5 "Error Codes."

[Requirement]

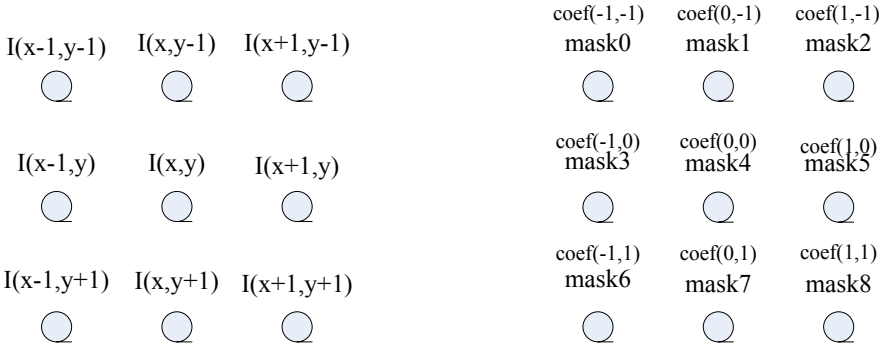
- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

[Note]

Copyright © HiSilicon Technologies Co., Ltd

- The handle ID is returned by the system.
- The pointers to the source data, output data, and control information cannot be null.
- The source data format must be single component.
- It is only required that the addresses of the source data and output data are 8-byte-aligned.
- The source data size (width x height) ranges from 64x64 pixels to 1920x1024 pixels.
- The output data stride must be configured based on stride requirements in section 11.2.1 "Concepts." The template coefficient must be 0 or 255.

[Formula]



$$Iout(x,y) = f(I(x+(k\&3)-1, y+(k\%3)-1) \mid coef((k\&3)-1,(k\%3)-1),\&,0,8)$$

$$f(A_k, \Theta, c_{min}, c_{max}) = A_{c_{min}} \Theta A_{c_{min+1}} \cdots \Theta A_{c_{max}}$$

Note: In the preceding formula, | indicates bitwise OR operation, & indicates bitwise AND operation, and % indicates REM operation.

$I(x,y)$ is defined by pstSrc, and $I_{out}(x,y)$ is defined by pstDst, $coef$ (mask) is defined by au8Mask[9] of pstErodeCtrl.

[Example]

None

[See Also]

None

HI_MPI_IVE_THRESH

[Description]

Creates a thresh task.

[Syntax]

```
HI_S32 HI_MPI_IVE_THRESH(IVE_HANDLE *pIveHandle, IVE_SRC_INFO_S *pstSrc,
IVE_MEM_INFO_S *pstDst, IVE_THRESH_CTRL_S *pstThreshCtrl,HI_BOOL
bInstant);
```

[Parameter]

Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle ID.	Output
pstSrc	Pointer to the source data.	Input
pstDst	Pointer to the output data.	Input
pstThreshCtrl	Pointer to the control information.	Input
bInstant	Flag indicating whether results need to be returned in time.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 11.5 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

[Note]

- The handle ID is returned by the system.
- The pointers to the source data, output data, and control information cannot be null.
- The source data format must be single component.
- It is only required that the addresses of the source data and output data are aligned by bytes.
- The source data size (width x height) ranges from 64x64 pixels to 1920x1080 pixels.
- The output data stride must be configured based on stride requirements in section 11.2.1 "Concepts." Three output formats are supported. For details, see IVE_THRESH_OUT_FMT_E

[Formula]

$$\text{Mode} = 2:\quad I_{out}(x,y) = \begin{cases} \min Value & (I(x,y) \le threshold) \\ I(x,y) & (I(x,y) > threshold) \end{cases}$$

$$\text{Mode} = 1:\quad I_{out}(x,y) = \begin{cases} I(x,y) & (I(x,y) \le threshold) \\ \max Value & (I(x,y) > threshold) \end{cases}$$

$$\text{Mode} = 0:\quad I_{out}(x,y) = \begin{cases} \min Value & (I(x,y) \le threshold) \\ \max Value & (I(x,y) > threshold) \end{cases}$$

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Note: $I(x, y)$ is defined by pstSrc, $I_{out}(x, y)$ is defined by pstDst, mode is defined by enOutFmt of pstThreshCtrl, and threshold, minValue, and maxValue correspond to u32Thresh, u32MinVal, and u32MaxVal of pstThreshCtrl respectively.

[Example]

None

[See Also]

None

HI_MPI_IVE_AND

[Description]

Creates an AND task for two pictures.

[Syntax]

```
HI_S32 HI_MPI_IVE_AND(IVE_HANDLE *pIveHandle, IVE_SRC_INFO_S *pstSrc1,
IVE_SRC_INFO_S *pstSrc2,IVE_MEM_INFO_S *pstDst, HI_BOOL bInstant);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle ID.	Output
pstSrc1	Pointer to source data 1.	Input
pstSrc2	Pointer to source data 2.	Input
pstDst	Pointer to the output data.	Input
bInstant	Flag indicating whether results need to be returned in time.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 11.5 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

[Note]

- The handle ID is returned by the system.
- The pointers to source data 1, source data 2, and output data cannot be null.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- The source data format must be single component.
- It is only required that the addresses of the source data and output data are aligned by bytes.
- The source data size (width x height) ranges from 64x64 pixels to 1920x1080 pixels.
- The output data stride must be configured based on stride requirements in section 11.2.1 "Concepts."

[Formula]

$$I_{out}(x,y) = I_{src1}(x,y) \& I_{src2}(x,y)$$

Note: $I_{src1}(x,y)$ is defined by pstSrc1, $I_{src2}(x,y)$ is defined by pstSrc2, and $I_{out}(x,y)$ is defined by pstDst.

[Example]

None

[See Also]

None

HI_MPI_IVE_SUB

[Description]

Creates a subtract task for two pictures.

[Syntax]

```
HI_S32 HI_MPI_IVE_SUB(IVE_HANDLE *pIveHandle, IVE_SRC_INFO_S *pstSrc1,
IVE_SRC_INFO_S *pstSrc2,IVE_MEM_INFO_S *pstDst, IVE_SUB_OUT_FMT_E
enOutFmt,HI_BOOL bInstant);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle ID.	Output
pstSrc1	Pointer to source data 1.	Input
pstSrc2	Pointer to source data 2.	Input
pstDst	Pointer to the output data.	Input
enOutFmt	Format of the output data.	Input
bInstant	Flag indicating whether results need to be returned in time.	Input

[Return Value]

Return Value	Description
0	Success.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Other values	Failure. For details, see section 11.5 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

[Note]

- The handle ID is returned by the system.
- The pointers to source data 1, source data 2, and output data cannot be null.
- The source data format must be single component.
- It is only required that the addresses of the source data and output data are aligned by bytes.
- The source data size (width x height) ranges from 64x64 pixels to 1920x1080 pixels.
- The output data stride must be configured based on stride requirements in section 11.2.1 "Concepts." Two output formats are supported. For details, see IVE_SUB_OUT_FMT_E

[Formula]

$$I_{out}(x,y) = abs(I_{src1}(x,y) - I_{src2}(x,y)) \qquad \mathrm{mod}\, e = 0$$
$$I_{out}(x,y) = (I_{src1}(x,y) - I_{src2}(x,y)) >> 1 \qquad \mathrm{mod}\, e = 1$$

Note: $I_{src1}(x,y)$ is defined by pstSrc1, $I_{src2}(x,y)$ is defined by pstSrc2, $I_{out}(x,y)$ is defined by pstDst, and $\mathrm{mod}\, e$ is defined by enOutFmt.

[Example]

None

[See Also]

None

HI_MPI_IVE_OR

[Description]

Creates an OR task for two pictures.

[Syntax]

```
HI_S32 HI_MPI_IVE_OR(IVE_HANDLE *pIveHandle, IVE_SRC_INFO_S *pstSrc1,
IVE_SRC_INFO_S *pstSrc2,IVE_MEM_INFO_S *pstDst, HI_BOOL bInstant);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle ID.	Output
pstSrc1	Pointer to source data 1.	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
pstSrc2	Pointer to source data 2.	Input
pstDst	Pointer to the output data.	Input
bInstant	Flag indicating whether results need to be returned in time.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 11.5 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

[Note]

- The handle ID is returned by the system.
- The pointers to source data 1, source data 2, and output data cannot be null.
- The source data format must be single component.
- It is only required that the addresses of the source data and output data are aligned by bytes.
- The source data size (width x height) ranges from 64x64 pixels to 1920x1080 pixels.
- The output data stride must be configured based on stride requirements in section 11.2.1 "Concepts." The OR operator is available for data.

[Formula]

$$I_{out}(x,y) = I_{src1}(x,y) \mid I_{src2}(x,y)$$

Note: $I_{src1}(x,y)$ is defined by pstSrc1, $I_{src2}(x,y)$ is defined by pstSrc2, and $I_{out}(x,y)$ is defined by pstDst.

[Example]

None

[See Also]

None

HI_MPI_IVE_INTEG

[Description]

Creates an integral task.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
HI_S32 HI_MPI_IVE_INTEG(IVE_HANDLE *pIveHandle, IVE_SRC_INFO_S *pstSrc,
IVE_MEM_INFO_S *pstDst, HI_BOOL bInstant);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle ID	Output
pstSrc	Pointer to the source data	Input
pstDst	Pointer to the output data	Input
bInstant	Flag indicating whether results need to be returned in time	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 11.5 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

[Note]

- The handle ID is returned by the system.
- The pointers to the source data and output data cannot be null.
- The source data format must be single component.
- The addresses of the source data and output data must be 8-byte-aligned.
- The source data size (width x height) ranges from 64x64 pixels to 1920x1080 pixels.
- The output data stride must be configured based on stride requirements in section 11.2.1 "Concepts." The size of the output data buffer must be calculated as follows: Source picture height x Output data stride x 8

[Formula]

$$I_{sum}(x,y)=\sum_{i\geq 0}^{i\leq x}\sum_{j\geq 0}^{j\leq y}I(i,j)$$

$$I_{sq}(x,y)=\sum_{i\geq 0}^{i\leq x}\sum_{j\geq 0}^{j\leq y}(I(i,j)\bullet I(i,j))$$

$$I_{out}(x,y)=(I_{sum}(x,y)\,\&\,0\text{x}FFFFFFF)\,|\,(I_{sq}(x,y)<<28)$$

Note: $I(x,y)$ is defined by pstSrc, and $I_{out}(x,y)$ is defined by pstDst.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Example]

None

[See Also]

None

HI_MPI_IVE_HIST

[Description]

Creates a histogram statistics task.

[Syntax]

```
HI_S32 HI_MPI_IVE_HIST(IVE_HANDLE *pIveHandle, IVE_SRC_INFO_S *pstSrc,
IVE_MEM_INFO_S *pstDst, HI_BOOL bInstant);
```

[Parameter]

Parameter	Description	Input/Output
pIveHandle	Pointer to the handle ID.	Output
pstSrc	Pointer to the source data.	Input
pstDst	Pointer to the output data.	Input
bInstant	Flag indicating whether results need to be returned in time.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 11.5 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

[Note]

- The handle ID is returned by the system.
- The pointers to the source data and output data cannot be null.
- The source data format must be single component.
- The addresses of the source data and output data must be 8-byte-aligned.
- The source data size (width x height) ranges from 64x64 pixels to 1920x1080 pixels.
- The output stride is set to the stride of the source data, and the buffer for storing the output histogram data is set to 1024 bytes.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Formula]

$$I_{out}(x) = \sum_i \sum_j ((I(i,j) == x)?1:0) \qquad x = 0 \ldots 255$$

Note: $I(x,y)$ is defined by pstSrc, and $I_{out}(x,y)$ is defined by pstDst.

[Example]

None

[See Also]

None

HI_MPI_IVE_Query

[Description]

Queries the completion status of an existing task.

[Syntax]

```
HI_S32 HI_MPI_IVE_Query(IVE_HANDLE IveHandle, HI_BOOL *pbFinish, HI_BOOL bBlock);
```

[Parameter]

Parameter	Description	Input/Output
IveHandle	Handle ID of a task.	Input
pbFinish	Pointer to the task completion status.	Output
bBlock	Flag indicating whether a task is blocked.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. For details, see section 11.5 "Error Codes."

[Requirement]

- Header files: hi_comm_ive.h, mpi_ive.h
- Library file: libmpi.a

[Note]

- The pbFinish pointer cannot be null.
- Before users use the results of the IVE task, to ensure the IVE tasks are completed, users can call the interface query by using block mode. In addition, the IVE implements tasks

Copyright © HiSilicon Technologies Co., Ltd

based on the tasks creation sequence, therefore, it is not necessary to call the query interface every time. If task A and B are created in sequence and the task B is completed, task A is also completed. In this case, you are not necessary to call the query interface when you use the result of task A.

[Example]

```
    HI_S32 s32Ret = HI_SUCCESS;
    IVE_HANDLE IveHandle;
    IVE_SRC_INFO_S stSrc;
    IVE_MEM_INFO_S stDst;
    HI_BOOL bInstant;
    HI_VOID *pVirtSrc;
    HI_VOID *pVirtDst;
    HI_BOOL bFinish, bBlock;

    stSrc.stSrcMem.u32Stride  = 352;
    stSrc.u32Height           = 288;
    stSrc.u32Width            = 352;
    stSrc.enSrcFmt            = IVE_SRC_FMT_SINGLE;
    stDst.u32Stride           = 352;
    bInstant                  = HI_TRUE;

    s32Ret = HI_MPI_SYS_MmzAlloc_Cached (&stSrc.stSrcMem.u32PhyAddr,
&pVirtSrc,
        "User", HI_NULL, stSrc.u32Height * stSrc.stSrcMem.u32Stride);

memset(pVirtSrc,1,stSrc.u32Height * stSrc.stSrcMem.u32Stride);

    s32Ret = HI_MPI_SYS_MmzAlloc_Cached (&stDst.u32PhyAddr, &pVirtDst,
        "User", HI_NULL, stSrc.u32Height * stDst.u32Stride);

memset(pVirtDst,0,stSrc.u32Height * stDst.u32Stride);

s32Ret = HI_MPI_IVE_DMA(&IveHandle, &stSrc, &stDst, bInstant);

    bBlock = HI_FALSE;
s32Ret = HI_MPI_IVE_Query(IveHandle, &bFinish, bBlock);
if (SUCCESS == s32Ret)
{
printf("bFinish=%d\n", bFinish);
}

    HI_MPI_SYS_MmzFree(stSrc.stSrcMem.u32PhyAddr, pVirtSrc);
HI_MPI_SYS_MmzFree(stDst.u32PhyAddr, pVirtDst);
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
return s32Ret;
```

[See Also]

None

11.4 Data Types

The IVE provides the following data types:

- IVE_MAX_HANDLE_NR: Defines the maximum number of task handles.
- IVE_MEM_INFO_S: Defines the information about the memory for storing input data or output data.
- IVE_SRC_FMT_E: Defines the source data format.
- IVE_SRC_INFO_S: Defines the source data information.
- IVE_CSC_OUT_FMT_E: Defines the CSC output format.
- IVE_CSC_MODE_E: Defines the CSC mode.
- IVE_CSC_CTRL_S: Defines the CSC control information.
- IVE_FILTER_CTRL_S: Defines the control information about template filter.
- IVE_FILTER_AND_CSC_CTRL_S: Defines the control information about template filter and CSC.
- IVE_SOBEL_CTRL_S: Defines the control information about sobel edge extraction.
- IVE_CANNY_OUT_FMT_E: Defines the canny output format.
- IVE_CANNY_CTRL_S: Defines the control information about canny edge extraction.
- IVE_DILATE_CTRL_S: Defines the dilate control information.
- IVE_ERODE_CTRL_S: Defines the erode control information.
- IVE_THRESH_OUT_FMT_E: Defines the thresh output format.
- IVE_THRESH_CTRL_S: Defines the thresh control information.
- IVE_SUB_OUT_FMT_E: Defines the output format after the subtraction operation between two pictures.

IVE_MAX_HANDLE_NR

[Description]

Defines the maximum number of task handles.

[Definition]

```
#define IVE_MAX_HANDLE_NR 0x0FFFFFFF
```

[Member]

None

[Note]

None

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

IVE_MEM_INFO_S

[Description]

Defines the information about the memory for storing input data or output data.

[Definition]

```
typedef struct hiIVE_DATA_ATTR_S
{
    HI_U32 u32PhyAddr;
    HI_U32 u32Stride;
}IVE_MEM_INFO_S;
```

[Member]

Member	Description
u32PhyAddr	Data physical address.
u32Stride	Data stride.

[Note]

The alignment requirement on the addresses of the input data and output data vary according to operators. Therefore, the strides are also different.

[See Also]

None

IVE_SRC_FMT_E

[Description]

Defines the source data format.

[Definition]

```
typedef enum hiIVE_SRC_FMT_E
{
    IVE_SRC_FMT_SINGLE = 0, /*Single component*/
    IVE_SRC_FMT_SP420,      /*YUV SP420*/
    IVE_SRC_FMT_SP422,      /*YUV SP422*/
    IVE_SRC_FMT_BUTT
}IVE_SRC_FMT_E;
```

[Member]

Member	Description
IVE_SRC_FMT_SINGLE	Simple component.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
IVE_SRC_FMT_SP420	SP420 format.
IVE_SRC_FMT_SP422	SP422 format.

[Note]

- The operators template filter, CSC, and template filter+CSC support the SP420 or SP422 input data.
- Other operators only support simple-component input data.

[See Also]

None

IVE_SRC_INFO_S

[Description]

Defines the source data information.

[Definition]

```
typedef struct hiIVE_SRC_INFO_S
{
    IVE_SRC_FMT_E  enSrcFmt;
    IVE_MEM_INFO_S stSrcMem;
    HI_U32 u32Height;
    HI_U32 u32Width;
}IVE_SRC_INFO_S;
```

[Member]

Member	Description
enSrcFmt	Source data format.
stSrcMem	Memory for storing the source data.
u32Height	Source data height.
u32Width	Source data width.

[Note]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

IVE_CSC_OUT_FMT_E

[Description]

Defines the CSC output format.

[Definition]

```
typedef enum hiIVE_CSC_OUT_FMT_E
{
    IVE_CSC_OUT_FMT_PACKAGE = 0,
    IVE_CSC_OUT_FMT_PLANAR,
    IVE_CSC_OUT_FMT_BUTT
}IVE_CSC_OUT_FMT_E;
```

[Member]

Member	Description
IVE_CSC_OUT_FMT_PACKAGE	Package format.
IVE_CSC_OUT_FMT_PLANAR	Planar format.

[Note]

None

[See Also]

None

IVE_CSC_MODE_E

[Description]

Defines the CSC mode.

[Definition]

```
typedef enum hiIVE_CSC_MODE_E
{
    IVE_CSC_MODE_VIDEO_BT601_AND_BT656 = 0,
    IVE_CSC_MODE_VIDEO_BT709,
    IVE_CSC_MODE_PIC_BT601_AND_BT656,
    IVE_CSC_MODE_PIC_BT709,
    IVE_CSC_MODE_BUTT
}IVE_CSC_MODE_E;
```

[Member]

Member	Description
IVE_CSC_MODE_VIDEO_BT601_AND_BT656	BT601&BT656 video conversion.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
IVE_CSC_MODE_VIDEO_BT709	BT709 video conversion.
IVE_CSC_MODE_PIC_BT601_AND_BT656	BT601&BT656 picture conversion.
IVE_CSC_MODE_PIC_BT709	BT709 picture conversion.

[Note]

- In IVE_CSC_MODE_VIDEO_BT601_AND_BT656 or IVE_CSC_MODE_VIDEO_BT709 mode, the output format must meet the following condition: 16 pixels ≤ R, G, B ≤ 235 pixels
- In IVE_CSC_MODE_PIC_BT601_AND_BT656 or IVE_CSC_MODE_PIC_BT709 mode, the output format must meet the following condition: 0 ≤ R, G, B ≤ 255 pixels

[See Also]

None

IVE_CSC_CTRL_S

[Description]

Defines the CSC control information.

[Definition]

```
typedef struct hiIVE_CSC_CTRL_S
{
    IVE_CSC_OUT_FMT_E enOutFmt;
    IVE_CSC_MODE_E    enCscMode;
}IVE_CSC_CTRL_S;
```

[Member]

Member	Description
enOutFmt	Output format.
enCscMode	Operating mode.

[Note]

None

[See Also]

None

IVE_FILTER_CTRL_S

[Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Defines the control information about template filter.

[Definition]

```
typedef struct hiIVE_FILTER_CTRL_S
{
   HI_S8 as8Mask[9];
   HI_S8 u8Norm;
}IVE_FILTER_CTRL_S;
```

[Member]

Member	Description
as8Mask[9]	3x3 template coefficient.
u8Norm	Norm parameter. Value range: [0, 11]

[Note]

You can set different template coefficients to implement various filtering effects.

[See Also]

None

IVE_FILTER_AND_CSC_CTRL_S

[Description]

Defines the control information about template filter and CSC.

[Definition]

```
typedef struct hiIVE_FILTER_AND_CSC_CTRL_S
{
   IVE_CSC_OUT_FMT_E enOutFmt;
   IVE_CSC_MODE_E enCscMode;
   HI_S8 as8Mask[9];
   HI_S8 u8Norm;
}IVE_FILTER_AND_CSC_CTRL_S;
```

[Member]

Member	Description
enOutFmt	Output format.
enCscMode	Operating mode.
as8Mask[9]	3x3 template coefficient.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u8Norm	Norm parameter. Value range: [0, 11]

[Note]

None

[See Also]

None

IVE_SOBEL_CTRL_S

[Description]

Defines the control information about sobel edge extraction.

[Definition]

```
typedef struct hiIVE_SOBEL_CTRL_S
{
   HI_S8 as8Mask[9];
}IVE_SOBEL_CTRL_S;
```

[Member]

Member	Description
as8Mask[9]	3x3 template coefficient.

[Note]

None

[See Also]

None

IVE_CANNY_OUT_FMT_E

[Description]

Defines the canny output format.

[Definition]

```
typedef enum hiIVE_CANNY_OUT_FMT_E
{
    IVE_CANNY_OUT_FMT_ONLY_MAG = 0,
    IVE_CANNY_OUT_FMT_MAG_AND_ANG,
   IVE_CANNY_OUT_FMT_BUTT
```


Copyright © HiSilicon Technologies Co., Ltd

```
}IVE_CANNY_OUT_FMT_E;
```

[Member]

Member	Description
IVE_CANNY_OUT_FMT_ONLY_MAG	Magnitude only.
IVE_CANNY_OUT_FMT_MAG_AND_ANG	Magnitude and angle.

[Note]

None

[See Also]

None

IVE_CANNY_CTRL_S

[Description]

Defines the control information about canny edge extraction.

[Definition]

```
typedef struct hiIVE_CANNY_CTRL_S
{
    IVE_CANNY_OUT_FMT_E enOutFmt;
    HI_S8 as8Mask[9];
}IVE_CANNY_CTRL_S;
```

[Member]

Member	Description
enOutFmt	Output format.
as8Mask[9]	3x3 template coefficient.

[Note]

None

[See Also]

None

IVE_DILATE_CTRL_S

[Description]

Defines the dilate control information.

[Definition]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
typedef struct hiIVE_DILATE_CTRL_S
{
   HI_U8 au8Mask[9];
}IVE_DILATE_CTRL_S;
```

[Member]

Member	Description
au8Mask[9]	3x3 template coefficient. Value: 0 or 255

[Note]

None

[See Also]

None

IVE_ERODE_CTRL_S

[Description]

Defines the erode control information.

[Definition]

```
typedef struct hiIVE_ERODE_CTRL_S
{
   HI_U8 au8Mask[9];
}IVE_ERODE_CTRL_S;
```

[Member]

Member	Description
au8Mask[9]	3x3 template coefficient. Value: 0 or 255

[Note]

None

[See Also]

None

IVE_THRESH_OUT_FMT_E

[Description]

Defines the thresh output format.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Definition]

```
typedef enum hiIVE_THRESH_OUT_FMT_E
{
    IVE_THRESH_OUT_FMT_BINARY = 0,
    IVE_THRESH_OUT_FMT_TRUNC,
    IVE_THRESH_OUT_FMT_TOZERO,
    IVE_THRESH_OUT_FMT_BUTT
}IVE_THRESH_OUT_FMT_E;
```

[Member]

Member	Description
IVE_THRESH_OUT_FMT_BINARY	If the source value is greater than the thresh, the maximum value is used; if the source value is smaller than or equal to the thresh, the minimum value is used.
IVE_THRESH_OUT_FMT_TRUNC	If the source value is greater than the thresh, the maximum value is used; if the source value is smaller than or equal to the thresh, the source value is retained.
IVE_THRESH_OUT_FMT_TOZERO	If the source value is greater than the thresh, the source value is retained; if the source value is smaller than or equal to the thresh, the minimum value is used.

[Note]

None

[See Also]

None

IVE_THRESH_CTRL_S

[Description]

Defines the thresh control information.

[Definition]

```
typedef struct hiIVE_THRESH_CTRL_S
{
    IVE_THRESH_OUT_FMT_E enOutFmt;
    HI_U32 u32Thresh;
    HI_U32 u32MinVal;
    HI_U32 u32MaxVal;
}IVE_THRESH_CTRL_S;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Member]

Member	Description
enOutFmt	Output format.
u32Thresh	Threshold. Value range: [0, 255]
u32MinVal	Minimum value. Value range: [0, 255]
u32MaxVal	Maximum value. Value range: [0, 255]

[Note]

None

[See Also]

None

IVE_SUB_OUT_FMT_E

[Description]

Defines the output format after the subtraction operation between two pictures.

[Definition]

```
typedef enum hiIVE_SUB_OUT_FMT_E
{
    IVE_SUB_OUT_FMT_ABS = 0,
    IVE_SUB_OUT_FMT_SFR,
    IVE_SUB_OUT_FMT_BUTT
}IVE_SUB_OUT_FMT_E;
```

[Member]

Member	Description
IVE_SUB_OUT_FMT_ABS	Absolute value of the difference.
IVE_SUB_OUT_FMT_SFR	The output result is obtained by shifting the result one digit right to reserve the signed bit.

[Note]

None

[See Also]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

11.5 Error Codes

Table 11-5 describes the error codes for the IVE.

Table 11-5 Error codes for the IVE

Error Code	Macro Definition	Description
0xA01D8001	HI_ERR_IVE_INVALID_DEVID	The device ID exceeds the valid range.
0xA01D8002	HI_ERR_IVE_INVALID_CHNID	The channel ID is incorrect or the region handle is invalid.
0xA01D8003	HI_ERR_IVE_ILLEGAL_PARAM	The parameter value exceeds the valid range.
0xA01D8004	HI_ERR_IVE_EXIST	The device, channel, or resource to be created exists.
0xA01D8005	HI_ERR_IVE_UNEXIST	The device, channel, or resource to be used or destroyed does not exist.
0xA01D8006	HI_ERR_IVE_NULL_PTR	The pointer is null.
0xA01D8007	HI_ERR_IVE_NOT_CONFIG	The module is not configured.
0xA01D8008	HI_ERR_IVE_NOT_SUPPORT	The parameter or function is not supported.
0xA01D8009	HI_ERR_IVE_NOT_PERM	The operation, for example, attempting to modify the value of a static parameter, is forbidden.
0xA01D800C	HI_ERR_IVE_NOMEM	The memory fails to be allocated due to the causes such as insufficient system memory.
0xA01D800D	HI_ERR_IVE_NOBUF	The buffer fails to be allocated due to the causes such as the data buffer applied for is too large.
0xA01D800E	HI_ERR_IVE_BUF_EMPTY	The buffer is empty.
0xA01D800F	HI_ERR_IVE_BUF_FULL	The buffer is full.
0xA01D8010	HI_ERR_IVE_NOTREADY	The system is not initialized or the corresponding module is not loaded.
0xA01D8011	HI_ERR_IVE_BADADDR	The address is invalid.
0xA01D8012	HI_ERR_IVE_BUSY	The system is busy.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Contents

12 HDMI 12-1
12.1 Overview 12-1
12.2 Concepts 12-1
12.3 API Reference 12-1
12.4 Data Types 12-18
12.5 Error Codes 12-40

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Tables

Table 12-1 Error codes for HDMI APIs.....12-40

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

12 HDMI

12.1 Overview

The embedded high-definition multimedia interface (HDMI) supports the video output.

12.2 Concepts

The audio of the HDMI cannot be output independently, which depends on the video output. The HDMI clock is derived from the VO clock. Therefore, the interface invocation sequence must be as follows: enable the VO, call the HDMI, and configure the audio and video (AV) output. The embedded HDMI does not support the high-bandwidth digital content protection (HDCP) function.

12.3 API Reference

The HDMI provides the following MPIs:

- HI_MPI_HDMI_Init: Initializes the HDMI.
- HI_MPI_HDMI_DeInit: Deinitializes the HDMI.
- HI_MPI_HDMI_Open: Opens the HDMI.
- HI_MPI_HDMI_Close: Closes the HDMI.
- HI_MPI_HDMI_SetAttr: Sets the HDMI attributes.
- HI_MPI_HDMI_GetAttr: Obtains the HDMI attributes.
- HI_MPI_HDMI_Start: Enables the HDMI.
- HI_MPI_HDMI_Stop: Disables the HDMI.
- HI_MPI_HDMI_ GetSinkCapability: Obtains the HDMI Sink capability.
- HI_MPI_HDMI_ SetAVMute: Sets the AV mute function of the HDMI.
- HI_MPI_HDMI_Force_GetEDID: Obtains the extended display identification data (EDID) information about the HDMI.
- HI_MPI_HDMI_SetDeepColor: Sets the deep color mode.
- HI_MPI_HDMI_GetDeepColor: Obtains the deep color mode.
- HI_MPI_HDMI_SetCsc: Sets the HDMI output effect of a device.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_HDMI_GetCsc: Obtains the HDMI output effect of a device.
- HI_MPI_HDMI_SetInfoFrame: Sets HDMI information frames.
- HI_MPI_HDMI_GetInfoFrame: Obtains HDMI information frames.

HI_MPI_HDMI_Init

[Description]

Initializes the HDMI.

[Syntax]

```
HI_S32 HI_MPI_HDMI_Init(HI_HDMI_INIT_PARA_S *pstHdmiPara);
```

[Parameter]

Parameter	Description	Input/Output
pstHdmiPara	Pointer to the initialization parameter structure	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

By default, the HDMI provides the event handling function in the kernel state. If users do not need to specially respond to the events, the function pointer to the initialization parameter structure can be set to NULL.

[Example]

```
HI_HDMI_INIT_PARA_S stHdmiPara;
HI_HDMI_ATTR_S     stAttr;
HI_HDMI_VIDEO_FMT_E enVideoFmt;
/* enable the VO */

enVideoFmt = HI_HDMI_VIDEO_FMT_1080P_60;

/* enable the HDMI */
stHdmiPara.enForceMode = HI_HDMI_FORCE_HDMI;
stHdmiPara.pCallBackArgs = NULL;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
stHdmiPara.pfnHdmiEventCallback = NULL;
HI_MPI_HDMI_Init(&stHdmiPara);

HI_MPI_HDMI_Open(0);

HI_MPI_HDMI_GetAttr(0, &stAttr);
stAttr.bEnableHdmi = HI_TRUE;
stAttr.bEnableVideo = HI_TRUE;
stAttr.enVideoFmt = enVideoFmt;

stAttr.enVidOutMode = HI_HDMI_VIDEO_MODE_YCBCR444;
stAttr.enDeepColorMode = HI_HDMI_DEEP_COLOR_24BIT;
stAttr.bxvYCCMode = HI_FALSE;

stAttr.bEnableAudio = HI_FALSE;
stAttr.enSoundIntf = HI_HDMI_SND_INTERFACE_I2S;
stAttr.bIsMultiChannel = HI_FALSE;

stAttr.enBitDepth = HI_HDMI_BIT_DEPTH_16;

stAttr.bEnableAviInfoFrame = HI_TRUE;
stAttr.bEnableAudInfoFrame = HI_TRUE;
stAttr.bEnableSpdInfoFrame = HI_FALSE;
stAttr.bEnableMpegInfoFrame = HI_FALSE;

stAttr.bDebugFlag = HI_FALSE;
stAttr.bHDCPEnable = HI_FALSE;

stAttr.b3DEnable = HI_FALSE;
HI_MPI_HDMI_SetAttr(0, &stAttr);

HI_MPI_HDMI_Start(0);

/* stop the HDMI */
HI_MPI_HDMI_Stop(0);

HI_MPI_HDMI_Close(0);

HI_MPI_HDMI_DeInit();
```

[See Also]

HI_MPI_HDMI_DeInit

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_HDMI_DeInit

[Description]

Deinitializes the HDMI.

[Syntax]

```
HI_S32 HI_MPI_HDMI_DeInit(HI_VOID);
```

[Parameter]

Parameter	Description	Input/Output
None	None	None

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

None

[Example]

See the example of HI_MPI_HDMI_Init

[See Also]

HI_MPI_HDMI_Init

HI_MPI_HDMI_Open

[Description]

Opens the HDMI.

[Syntax]

```
HI_S32 HI_MPI_HDMI_Open(HI_HDMI_ID_E enHdmi);
```

[Parameter]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

- Call HI_MPI_HDMI_Init before opening the HDMI.
- The error code HI_SUCCESS is returned if this HDMI is open repeatedly.

[Example]

See the example of HI_MPI_HDMI_Init

[See Also]

HI_MPI_HDMI_Close

HI_MPI_HDMI_Close

[Description]

Disables the HDMI.

[Syntax]

```
HI_S32 HI_MPI_HDMI_Close(HI_HDMI_ID_E enHdmi);
```

[Parameter]

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input

[Return Value]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

The error code HI_SUCCESS is returned if this HDMI can be closed repeatedly.

[Example]

See the example of HI_MPI_HDMI_Init.

[See Also]

HI_MPI_HDMI_Open

HI_MPI_HDMI_SetAttr

[Description]

Sets the HDMI attributes.

[Syntax]

```
HI_S32 HI_MPI_HDMI_SetAttr(HI_HDMI_ID_E enHdmi, HI_HDMI_ATTR_S *pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input
pstAttr	Pointer to the HDMI attribute structure	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

- Enable the HDMI before setting the HDMI attributes.
- Set the HDMI attributes before calling HI_MPI_HDMI_Start.
- If you want to set the HDMI attributes after calling HI_MPI_HDMI_Start, you are advised to call HI_MPI_HDMI_ SetAVMute to set AV mute function, set the new HDMI attributes, and then disable the AV mute function.

[Example]

```
HI_HDMI_INIT_PARA_S stHdmiPara;
HI_HDMI_ATTR_S     stAttr;
HI_HDMI_VIDEO_FMT_E enVideoFmt;

/* enable the VO */

/* enable the HDMI */

/* set the HDMI attributes after starting the HDMI */
HI_MPI_HDMI_SetAVMute(0, HI_TRUE);
HI_MPI_HDMI_GetAttr(0, &stAttr);

/* Users change the HDMI attributes */
stAttr.enVideoFmt = HI_HDMI_VIDEO_FMT_720P_60;

HI_MPI_HDMI_SetAttr(0, &stAttr);
HI_MPI_HDMI_SetAVMute(0, HI_FALSE);
```

[See Also]

HI_MPI_HDMI_SetAttr

HI_MPI_HDMI_GetAttr

[Description]

Obtains the HDMI attributes.

[Syntax]

```
HI_S32 HI_MPI_HDMI_GetAttr(HI_HDMI_ID_E enHdmi, HI_HDMI_ATTR_S *pstAttr);
```

[Parameter]

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter	Description	Input/Output
pstAttr	Pointer to the HDMI attribute structure	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

None

[Example]

See the example of HI_MPI_HDMI_Init.

[See Also]

HI_MPI_HDMI_SetAttr

HI_MPI_HDMI_Start

[Description]

Enables the HDMI.

[Syntax]

```
HI_S32 HI_MPI_HDMI_Start(HI_HDMI_ID_E enHdmi);
```

[Parameter]

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input

[Return Value]

Return Value	Description
0	Success

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Return Value	Description
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

Call HI_MPI_HDMI_Open first before enabling the HDMI.

[Example]

See the example of HI_MPI_HDMI_Init.

[See Also]

HI_MPI_HDMI_Stop

HI_MPI_HDMI_Stop

[Description]

Disables the HDMI.

[Syntax]

```
HI_S32 HI_MPI_HDMI_Stop(HI_HDMI_ID_E enHdmi);
```

[Parameter]

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

None

[Example]

See the example of HI_MPI_HDMI_Init.

[See Also]

HI_MPI_HDMI_Start

HI_MPI_HDMI_ GetSinkCapability

[Description]

Obtains the HDMI Sink capability.

[Syntax]

```
HI_S32 HI_MPI_HDMI_GetSinkCapability(HI_HDMI_ID_E enHdmi,
HI_HDMI_SINK_CAPABILITY_S *pstSinkCap);
```

[Parameter]

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input
pstSinkCap	Pointer to the HDMI sink capability structure	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

Enable the HDMI before calling this MPI.

[Example]

```
HI_HDMI_INIT_PARA_S stHdmiPara;
HI_HDMI_ATTR_S     stAttr;
HI_HDMI_VIDEO_FMT_E enVideoFmt;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
HI_HDMI_SINK_CAPABILITY_S stSinkCap;

/* enable the VO */

/* enable the HDMI */

/* Obtain the HDMI Sink capability */
HI_MPI_HDMI_GetSinkCapability(0, &stSinkCap);
```

[See Also]

None

HI_MPI_HDMI_ SetAVMute

[Description]

Sets the AV mute function of the HDMI.

[Syntax]

```
HI_S32 HI_MPI_HDMI_SetAVMute(HI_HDMI_ID_E enHdmi, HI_BOOL bAvMute);
```

[Parameter]

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input
bAvMute	Whether to set the AV mute function of the HDMI Value range: • HI_TRUE: Set the AV mute function. • HI_FALSE: Disable the AV mute function.	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

None

[Example]

See the example of HI_MPI_HDMI_SetAttr.

[See Also]

None

HI_MPI_HDMI_Force_GetEDID

[Description]

Obtains the EDID information about the HDMI.

[Syntax]

```
HI_S32 HI_MPI_HDMI_Force_GetEDID(HI_HDMI_ID_E enHdmi, HI_HDMI_EDID_S
*pstEdidData);
```

[Parameter]

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input
pstEdidData	EDID information about the HDMI	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

Enable the HDMI before calling this MPI.

[Example]

```
HI_HDMI_INIT_PARA_S stHdmiPara;
HI_HDMI_ATTR_S     stAttr;
HI_HDMI_EDID_S stEdidData;

/*Enable VO.*/
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
/*Enable the HDMI.*/

/*Obtain the EDID information about the HDMI.*/
HI_MPI_HDMI_Force_GetEDID(0, &stEdidData);
```

[See Also]

None

HI_MPI_HDMI_SetDeepColor

[Description]

Sets the deep color mode.

[Syntax]

```
HI_S32 HI_MPI_HDMI_SetDeepColor(HI_HDMI_ID_E enHdmi, HI_HDMI_DEEP_COLOR_E
enDeepColor);
```

[Parameter]

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input
enDeepColor	HDMI deep color mode	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

- HI_MPI_HDMI_SetDeepColor is an advanced interface. It is not used typically.
- Enable the HDMI before calling this MPI.
- HI_MPI_HDMI_SetDeepColor is used to set the deep color modes that are not supported by HI_MPI_HDMI_SetAttr.

[Example]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
HI_HDMI_INIT_PARA_S stHdmiPara;
HI_HDMI_ATTR_S     stAttr;

/*Enable the VO.*/

/*Enable the HDMI.*/

/*Set the deep color mode.*/
HI_MPI_HDMI_SetDeepColor (0, HI_HDMI_DEEP_COLOR_24BIT);
```

[See Also]

None

HI_MPI_HDMI_GetDeepColor

[Description]

Obtains the deep color mode.

[Syntax]

```
HI_S32 HI_MPI_HDMI_GetDeepColor(HI_HDMI_ID_E enHdmi, HI_HDMI_DEEP_COLOR_E
*penDeepColor);
```

[Parameter]

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input
penDeepColor	Pointer to the deep color mode of the HDMI	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

- HI_MPI_HDMI_GetDeepColor is an advanced interface. It is not used typically.
- Enable the HDMI before calling this MPI.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Example]

None

[See Also]

HI_MPI_HDMI_SetDeepColor

HI_MPI_HDMI_SetCsc

[Description]

Sets the HDMI output effect of a device.

[Syntax]

```
HI_S32 HI_MPI_HDMI_SetCsc(HI_HDMI_ID_E enHdmi, HI_HDMI_CSC_S *pstCsc);
```

[Parameter]

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input
pstCsc	Pointer to the HDMI picture output effect	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

- This MPI applies only to the Hi3520D.
- The HDMI must be enabled before you call this MPI.
- This MPI is used to adjust the effect of output pictures, including the brightness, contrast, hue, and saturation. The effect value ranges from 0 to 100.
- The CSC matrix must be HI_HDMI_CSC_MATRIX_RGB_TO_BT601_PC or HI_HDMI_CSC_MATRIX_RGB_TO_BT709_PC.

[Example]

None

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_MPI_HDMI_GetCsc

HI_MPI_HDMI_GetCsc

[Description]

Obtains the HDMI output effect of a device.

[Syntax]

```
HI_S32 HI_MPI_HDMI_GetCsc(HI_HDMI_ID_E enHdmi, HI_HDMI_CSC_S *pstCsc)
```

[Parameter]

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input
pstCsc	Pointer to the HDMI picture output effect	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

- This MPI applies only to the Hi3520D.
- The HDMI must be enabled before you call this MPI.

[Example]

None

[See Also]

HI_MPI_HDMI_SetCsc

HI_MPI_HDMI_SetInfoFrame

[Description]

Sets HDMI information frames.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
HI_S32 HI_MPI_HDMI_SetInfoFrame(HI_HDMI_ID_E enHdmi, HI_HDMI_INFOFRAME_S
*pstInfoFrame);
```

[Parameter]

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input
pstInfoFrame	Pointer to the structure of a picture information frame	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

- Generally, this interface is used to configure information frame attributes of the callback function in user mode. It is usually not required in other scenarios.
- Enable the HDMI before calling this interface.
- This interface supports setting only AVI and AUDIO information frames. When it is used to set information frames of other types, the settings are invalid. In addition, AVI information does not support the HI_HDMI_VIDEO_MODE_YCBCR422 output type.
- It is recommended that you use this interface before the setting.

[Example]

None

[See Also]

HI_MPI_HDMI_GetInfoFrame

HI_MPI_HDMI_GetInfoFrame

[Description]

Obtains HDMI information frames.

[Syntax]

```
HI_S32 HI_MPI_HDMI_GetInfoFrame(HI_HDMI_ID_E enHdmi,
HI_HDMI_INFOFRAME_TYPE_E enInfoFrameType, HI_HDMI_INFOFRAME_S
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
*pstInfoFrame);
```

[Parameter]

Parameter	Description	Input/Output
enHdmi	HDMI ID The value is 0.	Input
enInfoFrameType	Information frame type	Input
pstInfoFrame	Pointer to the structure of a picture information frame	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see section 12.5 "Error Codes."

[Requirement]

- Header files: mpi_hdmi.h, hi_comm_hdmi.h
- Library file: libhdmi.a

[Note]

- Generally, this interface is used to configure information frame attributes of the callback function in user mode. It is usually not required in other scenarios.
- Enable the HDMI before calling this interface.
- This interface supports setting only AVI and AUDIO information frames. When it is used to set information frames of other types, the settings are invalid. In addition, AVI information does not support the HI_HDMI_VIDEO_MODE_YCBCR422 output type.
- It is recommended that you use this interface before the setting.

[Example]

None

[See Also]

HI_MPI_HDMI_SetInfoFrame

12.4 Data Types

The data types related to the HDMI are defined as follows:

- HI_HDMI_ID_E: Defines the HDMI ID.
- HI_HDMI_INIT_PARA_S: Defines the HDMI initialization attribute structure.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_HDMI_EVENT_TYPE_E: Defines the HDMI event notice enumeration.
- HI_HDMI_FORCE_ACTION_E: Defines the HDMI forcible output mode enumeration.
- HI_HDMI_ATTR_S: Defines the HDMI attribute structure.
- HI_HDMI_VIDEO_FMT_E: Defines the HDMI video norm enumeration.
- HI_HDMI_VIDEO_MODE_E: Defines the HDMI color space type enumeration.
- HI_HDMI_DEEP_COLOR_E: Defines the HDMI deep color mode enumeration.
- HI_HDMI_SND_INTERFACE_E: Defines the HDMI audio output (AO) interface type enumeration.
- HI_HDMI_SAMPLE_RATE_E: Defines the HDMI AO sampling enumeration.
- HI_HDMI_BIT_DEPTH_E: Defines the HDMI AO sampling bit width enumeration.
- HI_HDMI_SINK_CAPABILITY_S: Defines the HDMI sink capability structure.
- HI_HDMI_CSC_S: Defines the HDMI picture output effect.
- HI_HDMI_INFOFRAME_TYPE_E: Defines the HDMI information frame type enumeration.
- HI_HDMI_AVI_INFOFRAME_VER2_S: Defines the AVI information frame.
- HI_HDMI_AUD_INFOFRAME_VER1_S: Defines the audio information frame.
- HI_HDMI_SPD_INFOFRAME_S: Defines the SPD information frame.
- HI_HDMI_MPEGSOURCE_INFOFRAME_S: Defines the MPEG source information frame.
- HI_HDMI_VENDORSPEC_INFOFRAME_S: Defines the vendor specific information frame.
- HI_HMDI_INFORFRAME_UNIT_U: Defines the HDMI information frame union.
- HI_HDMI_INFOFRAME_S: Defines the HDMI information frame.

HI_HDMI_ID_E

[Description]

Defines the HDMI ID.

[Syntax]

```
typedef enum hiHDMI_ID_E
{
    HI_HDMI_ID_0        = 0,
    HI_HDMI_ID_BUTT
} HI_HDMI_ID_E;
```

[Member]

Member	Description
HI_HDMI_ID_0	HDMI 0

[Note]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

None

HI_HDMI_INIT_PARA_S

[Description]

Defines the HDMI initialization attribute structure.

[Syntax]

```
typedef struct hiHDMI_INIT_PARA_S
{
    HI_HDMI_CallBack pfnHdmiEventCallback;
    HI_VOID          *pCallBackArgs;
    HI_HDMI_FORCE_ACTION_E enForceMode;
}HI_HDMI_INIT_PARA_S;
```

[Member]

Member	Description
pfnHdmiEventCallback	Event handling callback function
pCallBackArgs	Private data of the callback functions and parameters
enForceMode	HDMI forcible output mode

[Note]

- By default, the HDMI provides the event handling function in the kernel state. If users do not need to specially respond to the events, the function pointer to the initialization parameter structure can be set to NULL.
- The HDMI of the Hi3531 does not support the HDCP. The events related to the HDCP are invalid events.
- If the display device does not support the HDMI, enForceMode can be set to HI_HDMI_FORCE_DVI and the digital visual interface (DVI) is set to forcibly output the video. Otherwise, enForceMode is set to HI_HDMI_FORCE_DVI and the HDMI is set to forcibly output the video.

[See Also]

HI_MPI_HDMI_Init

HI_HDMI_EVENT_TYPE_E

[Description]

Defines the HDMI event notice enumeration.

[Syntax]

```
typedef enum hiHDMI_EVENT_TYPE_E
{
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_HDMI_EVENT_HOTPLUG = 0x10,
    HI_HDMI_EVENT_NO_PLUG,
    HI_HDMI_EVENT_EDID_FAIL,
    HI_HDMI_EVENT_HDCP_FAIL,
    HI_HDMI_EVENT_HDCP_SUCCESS,
    HI_HDMI_EVENT_HDCP_USERSETTING,
    HI_HDMI_EVENT_BUTT
}HI_HDMI_EVENT_TYPE_E;
```

[Member]

Member	Description
HI_HDMI_EVENT_HOTPLUG	HDMI hot-plug event
HI_HDMI_EVENT_NO_PLUG	HDMI cable disconnection event
HI_HDMI_EVENT_EDID_FAIL	HDMI EDID read failure event
HI_HDMI_EVENT_HDCP_FAIL	HDCP verification failure event
HI_HDMI_EVENT_HDCP_SUCCESS	HDCP verification success event
HI_HDMI_EVENT_HDCP_USERSETTING	HDMI reset event

[Note]

The HDMI of the Hi35xx does not support the HDCP. The events related to the HDCP are invalid.

[See Also]

HI_MPI_HDMI_Init

HI_HDMI_FORCE_ACTION_E

[Description]

Defines the HDMI forcible output mode enumeration.

[Syntax]

```
typedef enum hiHDMI_FORCE_ACTION_E
{
    HI_HDMI_FORCE_NULL,
    HI_HDMI_FORCE_HDMI,
    HI_HDMI_FORCE_DVI,
    HI_HDMI_INIT_BOOT_CONFIG
}HI_HDMI_FORCE_ACTION_E;
```

[Member]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
HI_HDMI_FORCE_NULL	Standard mode
HI_HDMI_FORCE_HDMI	Forcibly output video in HDMI mode
HI_HDMI_FORCE_DVI	Forcibly output video in DVI mode
HI_HDMI_INIT_BOOT_CONFIG	Test only

[Note]

None

[See Also]

HI_MPI_HDMI_Init

HI_HDMI_ATTR_S

[Description]

Defines the HDMI attribute structure.

[Syntax]

```
typedef struct hiHDMI_ATTR_S
{
    HI_BOOL               bEnableHdmi;
    HI_BOOL               bEnableVideo;
    HI_HDMI_VIDEO_FMT_E   enVideoFmt;
    HI_HDMI_VIDEO_MODE_E  enVidOutMode;
    HI_HDMI_DEEP_COLOR_E  enDeepColorMode;
    HI_BOOL               bxvYCCMode;
    HI_BOOL               bEnableAudio;
    HI_HDMI_SND_INTERFACE_E enSoundIntf;
    HI_BOOL               bIsMultiChannel;
    HI_HDMI_SAMPLE_RATE_E enSampleRate;
    HI_U8                 u8DownSampleParm;
    HI_HDMI_BIT_DEPTH_E   enBitDepth;
    HI_U8                 u8I2SCtlVbit;
    HI_BOOL               bEnableAviInfoFrame;
    HI_BOOL               bEnableAudInfoFrame;
    HI_BOOL               bEnableSpdInfoFrame;
    HI_BOOL               bEnableMpegInfoFrame;
    HI_U32                bDebugFlag;
    HI_BOOL               bHDCPEnable;
    HI_BOOL               b3DEnable;
    HI_U32                u83DParam;
} HI_HDMI_ATTR_S;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Member]

Member	Description
bEnableHdmi	Whether to forcibly output the video over the HDMI HI_TRUE: Forcibly output the video over the HDMI. HI_FALSE: Output the video over the DVI. This value must be set before HI_HDMI_Start is called or after the HI_HDMI_Stop is called.
bEnableVideo	Whether to output video The value must be HI_TRUE.
enVideoFmt	Video norm. This value of the video norm must be consistent with the norm of the video output.
enVidOutMode	HDMI video output mode VIDEO_MODE_YCBCR444, VIDEO_MODE_YCBCR422, and VIDEO_MODE_RGB444 are supported, but VIDEO_MODE_YCBCR422 is not supported.
enDeepColorMode	DeepColor output mode It is HI_HDMI_DEEP_COLOR_24BIT by default. HI_HDMI_DEEP_COLOR_30BIT and HI_HDMI_DEEP_COLOR_36BIT are not supported.
bxvYCCMode	Whether to enable the xvYCC output mode It is HI_FALSE by default.
bEnableAudio	Whether to enable the audio
enSoundIntf	HDMI audio source It is set to HI_HDMI_SND_INTERFACE_I2S, which needs to be consistent with the VO interface.
bIsMultiChannel	Multi-channel or stereo 0: stereo 1: eight-channel fixed for multi-channel
enSampleRate	Audio sampling rate. This parameter needs to be consistent with that of the VO.
u8DownSampleParm	Audio down sampling rate parameter. Its default value is 0. The parameter configuration is invalid.
enBitDepth	Audio bit width. Its default value is 16. This parameter needs to be consistent with that of the VO.
u8I2SCtlVbit	Reserved. It is set to 0. It has impacts on the content on the I^2S control register of the HDMI. The parameter configuration is invalid.
bEnableAviInfoFrame	Whether to enable AVI InfoFrame You are advised to enable this function.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
bEnableAudInfoFrame	Whether to enable audio InfoFrame You are advised to enable this function. Audio InfoFrame cannot be enabled in DVI mode.
bEnableSpdInfoFrame	Whether to enable SPD InfoFrame You are advised to disable this function. This function is disabled by default, and the parameter configuration is invalid.
bEnableMpegInfoFrame	Whether to enable MPEG InfoFrame You are advised to disable this function. This function is disabled by default, and the parameter configuration is invalid.
bDebugFlag	Whether to enable the debug information in the HDMI You are advised to disable this function. The default value is 0, and the parameter configuration is invalid.
bHDCPEnable	Whether to enable the HDCP 0: disabled 1: enabled The default value is 0, and the parameter configuration is invalid.
b3DEnable	Whether to enable 3D mode 0: disabled 1: enabled The default value is 0, and the parameter configuration is invalid.
u83DParam	3D parameter The default value is 0, and the parameter configuration is invalid.

[Note]

- Users can set the HDMI attributes based on the advised values.
- The values of the following HDMI attributes are fixed: bDebugFlag, bHDCPEnable, b3Denable, u83DParam, u8DownSampleParm, u8I2SCtlVbit, bEnableMpegInfoFrame, and bEnableSpdInfoFrame. The HDMI of the Hi35xx does not support HDCP and 3D.

[See Also]

HI_MPI_HDMI_SetAttr

HI_HDMI_VIDEO_FMT_E

[Description]

Defines the HDMI video norm enumeration.

[Syntax]

Copyright © HiSilicon Technologies Co., Ltd

```
typedef enum hiHDMI_VIDEO_FMT_E
{
    HI_HDMI_VIDEO_FMT_1080P_60 = 0,
    HI_HDMI_VIDEO_FMT_1080P_50,
    HI_HDMI_VIDEO_FMT_1080P_30,
    HI_HDMI_VIDEO_FMT_1080P_25,
    HI_HDMI_VIDEO_FMT_1080P_24,

    HI_HDMI_VIDEO_FMT_1080i_60,
    HI_HDMI_VIDEO_FMT_1080i_50,

    HI_HDMI_VIDEO_FMT_720P_60,
    HI_HDMI_VIDEO_FMT_720P_50,

    HI_HDMI_VIDEO_FMT_576P_50,
    HI_HDMI_VIDEO_FMT_480P_60,

    HI_HDMI_VIDEO_FMT_PAL,
    HI_HDMI_VIDEO_FMT_PAL_N,
    HI_HDMI_VIDEO_FMT_PAL_Nc,

    HI_HDMI_VIDEO_FMT_NTSC,
    HI_HDMI_VIDEO_FMT_NTSC_J,
    HI_HDMI_VIDEO_FMT_NTSC_PAL_M,

    HI_HDMI_VIDEO_FMT_SECAM_SIN,
    HI_HDMI_VIDEO_FMT_SECAM_COS,

    HI_HDMI_VIDEO_FMT_861D_640X480_60,
    HI_HDMI_VIDEO_FMT_VESA_800X600_60,
    HI_HDMI_VIDEO_FMT_VESA_1024X768_60,
    HI_HDMI_VIDEO_FMT_VESA_1280X720_60,
    HI_HDMI_VIDEO_FMT_VESA_1280X800_60,
    HI_HDMI_VIDEO_FMT_VESA_1280X1024_60,
    HI_HDMI_VIDEO_FMT_VESA_1366X768_60,
    HI_HDMI_VIDEO_FMT_VESA_1440X900_60,
    HI_HDMI_VIDEO_FMT_VESA_1440X900_60_RB,
    HI_HDMI_VIDEO_FMT_VESA_1600X900_60_RB,
    HI_HDMI_VIDEO_FMT_VESA_1600X1200_60,
    HI_HDMI_VIDEO_FMT_VESA_1920X1080_60,
    HI_HDMI_VIDEO_FMT_VESA_1920X1200_60,
    HI_HDMI_VIDEO_FMT_VESA_2560X1440_30,
    HI_HDMI_VIDEO_FMT_VESA_2048X1152_60,
    HI_HDMI_VIDEO_FMT_VESA_CUSTOMER_DEFINE,
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_HDMI_VIDEO_FMT_BUTT
}HI_HDMI_VIDEO_FMT_E;
```

[Member]

None

[Note]

You can set the HDMI norms based on the video output norms.

[See Also]

HI_MPI_HDMI_SetAttr

HI_HDMI_VIDEO_MODE_E

[Description]

Defines the HDMI color space type enumeration.

[Syntax]

```
typedef enum hiHDMI_VIDEO_MODE
{
    HI_HDMI_VIDEO_MODE_RGB444,
    HI_HDMI_VIDEO_MODE_YCBCR422,
    HI_HDMI_VIDEO_MODE_YCBCR444,
    HI_HDMI_VIDEO_MODE_BUTT
}HI_HDMI_VIDEO_MODE_E;
```

[Member]

Member	Description
HI_HDMI_VIDEO_MODE_RGB444	RGB444 output mode
HI_HDMI_VIDEO_MODE_YCBCR422	YCBCR422 output mode
HI_HDMI_VIDEO_MODE_YCBCR444	YCBCR444 output mode

[Note]

None

[See Also]

HI_MPI_HDMI_SetAttr

HI_HDMI_DEEP_COLOR_E

[Description]

Defines the HDMI deep color mode enumeration.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Syntax]

```
typedef enum hiHDMI_DEEP_COLOR_E
{
    HI_HDMI_DEEP_COLOR_24BIT = 0x00,
    HI_HDMI_DEEP_COLOR_30BIT,
    HI_HDMI_DEEP_COLOR_36BIT,
    HI_HDMI_DEEP_COLOR_OFF   = 0xff,
    HI_HDMI_DEEP_COLOR_BUTT
}HI_HDMI_DEEP_COLOR_E;
```

[Member]

Member	Description
HI_HDMI_DEEP_COLOR_24BIT	HDMI deep-color 24-bit mode
HI_HDMI_DEEP_COLOR_30BIT	HDMI deep-color 30-bit mode
HI_HDMI_DEEP_COLOR_36BIT	HDMI deep-color 36-bit mode
HI_HDMI_DEEP_COLOR_OFF	HDMI deep-color disable mode

[Note]

None

[See Also]

HI_MPI_HDMI_SetAttr

HI_HDMI_SND_INTERFACE_E

[Description]

Defines the HDMI AO interface type enumeration.

[Syntax]

```
typedef enum hiHDMI_SND_INTERFACE_E
{
    HI_HDMI_SND_INTERFACE_I2S,
    HI_HDMI_SND_INTERFACE_SPDIF,
    HI_HDMI_SND_INTERFACE_HBR,
    HI_HDMI_SND_INTERFACE_BUTT
}HI_HDMI_SND_INTERFACE_E;
```

[Member]

Member	Description
HI_HDMI_SND_INTERFACE_I2S	I^2S interface type

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
HI_HDMI_SND_INTERFACE_SPDIF	SPDIF interface type
HI_HDMI_SND_INTERFACE_HBR	HBR interface type

[Note]

The HDMI audio output of the Hi35xx is I^2S interface type.

[See Also]

HI_MPI_HDMI_SetAttr

HI_HDMI_SAMPLE_RATE_E

[Description]

Defines the HDMI AO sampling enumeration.

[Syntax]

```
typedef enum hiHDMI_SAMPLE_RATE_E
{
    HI_HDMI_SAMPLE_RATE_UNKNOWN=0,
    HI_HDMI_SAMPLE_RATE_8K    = 8000,
    HI_HDMI_SAMPLE_RATE_11K   = 11025,
    HI_HDMI_SAMPLE_RATE_12K   = 12000,
    HI_HDMI_SAMPLE_RATE_16K   = 16000,
    HI_HDMI_SAMPLE_RATE_22K   = 22050,
    HI_HDMI_SAMPLE_RATE_24K   = 24000,
    HI_HDMI_SAMPLE_RATE_32K   = 32000,
    HI_HDMI_SAMPLE_RATE_44K   = 44100,
    HI_HDMI_SAMPLE_RATE_48K   = 48000,
    HI_HDMI_SAMPLE_RATE_88K   = 88200,
    HI_HDMI_SAMPLE_RATE_96K   = 96000,
     HI_HDMI_SAMPLE_RATE_176K  = 176400,
    HI_HDMI_SAMPLE_RATE_192K  = 192000,
    HI_HDMI_SAMPLE_RATE_BUTT
}HI_HDMI_SAMPLE_RATE_E;
```

[Member]

None

[Note]

None

[See Also]

HI_MPI_HDMI_SetAttr

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

HI_HDMI_BIT_DEPTH_E

[Description]

Defines the HDMI AO sampling bit width enumeration.

[Syntax]

```
typedef enum hiHDMI_BIT_DEPTH_E
{
    HI_HDMI_BIT_DEPTH_UNKNOWN =0,
    HI_HDMI_BIT_DEPTH_8  = 8,
    HI_HDMI_BIT_DEPTH_16 = 16,
    HI_HDMI_BIT_DEPTH_18 = 18,
    HI_HDMI_BIT_DEPTH_20 = 20,
    HI_HDMI_BIT_DEPTH_24 = 24,
    HI_HDMI_BIT_DEPTH_32 = 32,
    HI_HDMI_BIT_DEPTH_BUTT
}HI_HDMI_BIT_DEPTH_E;
```

[Member]

None

[Note]

None

[See Also]

HI_MPI_HDMI_SetAttr

HI_HDMI_SINK_CAPABILITY_S

[Description]

Defines the HDMI sink capability structure.

[Syntax]

```
typedef struct hiHDMI_SINK_CAPABILITY_S
{
    HI_BOOL           bConnected;
    HI_BOOL           bSupportHdmi;
    HI_BOOL           bIsSinkPowerOn;
    HI_BOOL           bIsRealEDID;
    HI_HDMI_VIDEO_FMT_E enNativeVideoFormat;
    HI_BOOL           bVideoFmtSupported[HI_HDMI_VIDEO_FMT_BUTT];
    HI_BOOL           bSupportYCbCr;
    HI_BOOL           bSupportxvYCC601;
    HI_BOOL           bSupportxvYCC709;
    HI_U8             u8MDBit;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_BOOL           bAudioFmtSupported[HI_HDMI_MAX_AUDIO_CAP_COUNT];
    HI_U32
u32AudioSampleRateSupported[HI_HDMI_MAX_AUDIO_SMPRATE_COUNT];
    HI_U32            u32MaxPcmChannels;
    HI_U8             u8Speaker;
    HI_U8             u8IDManufactureName[4];
    HI_U32            u32IDProductCode;
    HI_U32            u32IDSerialNumber;
    HI_U32            u32WeekOfManufacture;
    HI_U32            u32YearOfManufacture;
    HI_U8             u8Version;
    HI_U8             u8Revision;
    HI_U8             u8EDIDExternBlockNum;
    HI_U8             u8IEERegId[3];
    HI_BOOL           bIsPhyAddrValid;
    HI_U8             u8PhyAddr_A;
    HI_U8             u8PhyAddr_B;
    HI_U8             u8PhyAddr_C;
    HI_U8             u8PhyAddr_D;
    HI_BOOL           bSupportDVIDual;
    HI_BOOL           bSupportDeepColorYCBCR444
    HI_BOOL           bSupportDeepColor30Bit;
    HI_BOOL           bSupportDeepColor36Bit;
    HI_BOOL           bSupportDeepColor48Bit;
    HI_BOOL           bSupportAI;
    HI_U32            u8MaxTMDSClock;
    HI_BOOL           bI_Latency_Fields_Present
    HI_BOOL           bLatency_Fields_Present;
    HI_BOOL           bHDMI_Video_Present;
    HI_U8             u8Video_Latency;
    HI_U8             u8Audio_Latency;
    HI_U8             u8Interlaced_Video_Latency;
    HI_U8             u8Interlaced_Audio_Latency;
} HI_HDMI_SINK_CAPABILITY_S;
```

[Member]

Member	Description
bConnected	Whether the devices are connected
bSupportHdmi	Whether the HDMI is supported by the device. If the HDMI is not supported by the device, the device is DVI.
bIsSinkPowerOn	Whether the sink device is powered on
bIsRealEDID	Whether the EDID obtains the flag from the sink device

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	HI_TRUE: The EDID information is correctly read. HI_FASE: default settings
enNativeVideoFormat	Physical resolution of the display device
bVideoFmtSupported	Video capability set HI_TRUE: This display format is supported. HI_FALSE: This display format is not supported.
bSupportYCbCr	Whether the YCBCR display is supported HI_TRUE: The YCBCR display is supported. HI_FALSE: Only red-green-blue (RGB) is supported.
bSupportxvYCC601	Whether the xvYCC601 color format is supported
bSupportxvYCC709	Whether the xvYCC709 color format is supported
u8MDBit	Transfer profile supported by xvYCC601. 1: P0; 2: P1; 4: P2
bAudioFmtSupported	Audio capability set. For details, see Table 37 in EIA-CEA-861-D. HI_TRUE: This display format is supported. HI_FALSE: This display format is not supported.
u32AudioSampleRateSupported	Audio sampling rate capability set. The value 0 is invalid. Other values are the supported audio sampling rates.
u32MaxPcmChannels	Maximum pulse code modulation (PCM) channel number of the audio
u8Speaker	Speaker position. For details, see the definition of SpeakerDATABlock in EIA-CEA-861-D.
u8IDManufactureName	Device vendor flag
u32IDProductCode	Device ID
u32IDSerialNumber	Device sequence number
u32WeekOfManufacture	Device production data (week)
u32YearOfManufacture	Set the production data (year)
u8Version	Device version number
u8Revision	Device sub version number
u8EDIDExternBlockNum	EDID extended block number
u8IEERegId	24-bit IEEE registration identifier (0x000C03)
bIsPhyAddrValid	Valid flag of the consumer electronics control (CEC) physical address
u8PhyAddr_A	CEC physical address A

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u8PhyAddr_B	CEC physical address B
u8PhyAddr_C	CEC physical address C
u8PhyAddr_D	CEC physical address D
bSupportDVIDual	Whether to support the DVI dual-link operation
bSupportDeepColorYCBCR444	Whether to support the YCBCR 4:4:4 deep-color mode
bSupportDeepColor30Bit	Whether to support the deep-color 30-bit mode
bSupportDeepColor36Bit	Whether to support the deep-color 36-bit mode
bSupportDeepColor48Bit	Whether to support the deep-color 48-bit mode
bSupportAI	Whether to support the Supports_AI mode
u8MaxTMDSClock	Maximum TMDS clock
bI_Latency_Fields_Present	Delay flag bit
bLatency_Fields_Present	Whether Video_Latency and Audio_Latency fields exist
bHDMI_Video_Present	Special video format
u8Video_Latency	Video delay
u8Audio_Latency	Audio delay
u8Interlaced_Video_Latency	Video delay in interlaced video mode
u8Interlaced_Audio_Latency	Audio delay in interlaced video mode

[Note]

None

[See Also]

HI_MPI_HDMI_ GetSinkCapability

HI_HDMI_EDID_S

[Description]

Defines the structure of the HDMI EDID information.

[Syntax]

```
typedef struct hiHI_HDMI_EDID_S
{
    HI_BOOL              bEdidValid;
    HI_U32               u32Edidlength;
    HI_U8                u8Edid[512];
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
}HI_HDMI_EDID_S;
```

[Member]

Member	Description
bEdidValid	EDID information validity
u32Edidlength	EDID information length
u8Edid	EDID information

[Note]

None

[See Also]

HI_MPI_HDMI_Force_GetEDID

HI_HDMI_CSC_S

[Description]

Defines the HDMI picture output effect.

[Syntax]

```
typedef struct hiHI_HDMI_CSC_S
{
    VO_CSC_MATRIX_E enCscMatrix;
    HI_U32 u32Luma;                 /*Luminance: 0-100 default: 50 */
    HI_U32 u32Contrast;             /*Contrast: 0-100 default: 50 */
    HI_U32 u32Hue;                  /*Hue: 0-100 default: 50 */
    HI_U32 u32Satuature;            /*Saturation: 0-100 default: 50 */
} HI_HDMI_CSC_S;
```

[Member]

Member	Description
enCscMatrix	CSC matrix select
u32Luma	Luminance
u32Contrast	Contrast
u32Hue	Hue
u32Satuature	Saturation

[Note]

None

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[See Also]

HI_MPI_HDMI_SetCsc

HI_HDMI_INFOFRAME_TYPE_E

[Description]

Defines the HDMI information frame type enumeration.

[Syntax]

```
typedef enum tagHI_HDMI_INFOFRAME_TYPE_E
{
    HI_INFOFRAME_TYPE_AVI,          /* HDMI AVI InfoFrame */
    HI_INFOFRAME_TYPE_SPD,          /* HDMI SPD InfoFrame */
    HI_INFOFRAME_TYPE_AUDIO,        /* HDMI AUDIO InfoFrame */
    HI_INFOFRAME_TYPE_MPEG,         /* HDMI MPEG InfoFrame */
    HI_INFOFRAME_TYPE_VENDORSPEC,   /* HDMI Vendor Specific InfoFrame */
    HI_INFOFRAME_TYPE_BUTT
}HI_HDMI_INFOFRAME_TYPE_E;
```

[Member]

Member	Description
HI_INFOFRAME_TYPE_AVI	AVI information frame type
HI_INFOFRAME_TYPE_SPD	SPD information frame type
HI_INFOFRAME_TYPE_MPEG	MPEG information frame type
HI_INFOFRAME_TYPE_VENDORSPEC	Vendor specific information frame type

[Note]

None

[See Also]

HI_MPI_HDMI_GetInfoFrame

HI_HDMI_AVI_INFOFRAME_VER2_S

[Description]

Defines the AVI information frame.

[Syntax]

```
typedef struct hiUNF_HDMI_AVI_INFOFRAME_VER2_S
{
    HI_HDMI_VIDEO_FMT_E       enTimingMode;
    HI_HDMI_VIDEO_MODE_E      enOutputType;
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
    HI_BOOL                 bActive_Infor_Present;
    HI_HDMI_BARINFO_E         enBarInfo;
    HI_HDMI_SCANINFO_E        enScanInfo;
    HI_HDMI_COLORSPACE_E      enColorimetry;
    HI_HDMI_ASPECT_RATIO_E    enAspectRatio;
    HI_HDMI_ASPECT_RATIO_E    enActiveAspectRatio;
    HI_HDMI_PICTURE_SCALING_E enPictureScaling;
    HI_HDMI_RGB_QUAN_RAGE_E   enRGBQuantization;
    HI_BOOL                 bIsITContent;
    HI_U32                  u32PixelRepetition;

    HI_HDMI_CONTENT_TYPE_E    enContentType;
    HI_HDMI_YCC_QUAN_RAGE_E   enYCCQuantization;

    HI_U32                  u32LineNEndofTopBar;
    HI_U32                  u32LineNStartofBotBar;
    HI_U32                  u32PixelNEndofLeftBar;
    HI_U32                  u32PixelNStartofRightBar;
}HI_HDMI_AVI_INFOFRAME_VER2_S;
```

[Member]

Member	Description
enTimingMode	Time sequence mode enumeration variable
enOutputType	Format of output pixel
bActive_Infor_Present	Whether to validate the current information frame
enBarInfo	BAR information enumeration variable
enScanInfo	Scanning information enumeration variable
enColorimetry	Color space matrix
enAspectRatio	Aspect ratio enumeration variable
enActiveAspectRatio	Enable aspect ratio enumeration variable
enPictureScaling	Picture scaling enumeration
enRGBQuantization	RGB color space range
bIsITContent	Whether IT contents are included
u32PixelRepetition	Pixel repetition mode
enContentType	Content type
enYCCQuantization	YCC color space range
u32LineNEndofTopBar	End line of the top border

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
u32LineNStartofBotBar	Start line of the bottom border
u32PixelNEndofLeftBar	End pixel of the left border
u32PixelNStartofRightBar	Start pixel of the right border

[Note]

For details about the enumeration members of each member variable, see the Hi_comm_hdmi.h file.

[See Also]

None

HI_HDMI_AUD_INFOFRAME_VER1_S

[Description]

Defines the audio information frame.

[Syntax]

```
typedef struct hiUNF_HDMI_AUD_INFOFRAME_VER1_S
{
    HI_U32                      u32ChannelCount;
    HI_HDMI_CODING_TYPE_E         enCodingType;
    HI_U32                      u32SampleSize;
    HI_U32                      u32SamplingFrequency;
    HI_U32                      u32ChannelAlloc;
    HI_U32                      u32LevelShift;
    HI_BOOL                     u32DownmixInhibit;
}HI_HDMI_AUD_INFOFRAME_VER1_S;
```

[Member]

Member	Description
u32ChannelCount	Number of channels
enCodingType	Audio data coding type
u32SampleSize	Sampling bit width. Typically, it is set to 0. 0: relies on the stream header 1: 16 bits 2: 20 bits 3: 24 bits
u32SamplingFrequency	Sampling frequency. It is recommended that this parameter be set to 0.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Member	Description
	0: relies on the stream header 1: 32 kHz 2: 44.1 kHz 3: 48 kHz 4: 88.2 kHz 5: 96 kHz 6: 176.4 kHz 7: 192 kHz
u32ChannelAlloc	Channel allocation. It is set to 0.
u32LevelShift	It is set to 0.
u32DownmixInhibit	It is set to 0.

[Note]

None

[See Also]

None

HI_HDMI_SPD_INFOFRAME_S

[Description]

Defines the SPD information frame.

[Syntax]

```
typedef struct hiUNF_HDMI_SPD_INFOFRAME_S
{
    HI_U8                   u8VendorName[8];
    HI_U8                   u8ProductDescription[16];
}HI_HDMI_SPD_INFOFRAME_S;
```

[Member]

Member	Description
u8VendorName[8]	Vendor name
u8ProductDescription[16]	Product descriptor

[Note]

None

[See Also]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

- HI_MPI_HDMI_SetInfoFrame
- HI_MPI_HDMI_GetInfoFrame

HI_HDMI_MPEGSOURCE_INFOFRAME_S

[Description]

Defines the MPEG source information frame.

[Syntax]

```
typedef struct hiUNF_HDMI_MPEGSOURCE_INFOFRAME_S
{
    HI_U32                      u32MPEGBitRate;
    HI_BOOL                     bIsFieldRepeated;
}HI_HDMI_MPEGSOURCE_INFOFRAME_S;
```

[Member]

Member	Description
u32MPEGBitRate	MPEG bit rate
bIsFieldRepeated	Field repeater flag

[Note]

None

[See Also]

- HI_MPI_HDMI_SetInfoFrame
- HI_MPI_HDMI_GetInfoFrame

HI_HDMI_VENDORSPEC_INFOFRAME_S

[Description]

Defines the vendor specific information frame.

[Syntax]

```
typedef struct hiUNF_HDMI_VENDORSPEC_INFOFRAME_S
{
    HI_U32                      u32RegistrationId;
}HI_HDMI_VENDORSPEC_INFOFRAME_S;
```

[Member]

Member	Description
u32RegistrationId	Registration ID

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Note]

None

[See Also]

- HI_MPI_HDMI_SetInfoFrame
- HI_MPI_HDMI_GetInfoFrame

HI_HMDI_INFORFRAME_UNIT_U

[Description]

Defines the HDMI information frame union.

[Syntax]

```
typedef union hiUNF_HDMI_INFOFRAME_UNIT_U
{
    HI_HDMI_AVI_INFOFRAME_VER2_S   stAVIInfoFrame;
    HI_HDMI_AUD_INFOFRAME_VER1_S   stAUDInfoFrame;
    HI_HDMI_SPD_INFOFRAME_S        stSPDInfoFrame;
    HI_HDMI_MPEGSOURCE_INFOFRAME_S stMPEGSourceInfoFrame;
    HI_HDMI_VENDORSPEC_INFOFRAME_S stVendorSpecInfoFrame;
}HI_HMDI_INFORFRAME_UNIT_U;
```

[Member]

Member	Description
stAVIInfoFrame	Data structure of the AVI information frame
stAUDInfoFrame	Data structure of the audio information frame
stSPDInfoFrame	Data structure of the SPD information frame
stMPEGSourceInfoFrame	Data structure of the MPEG information frame
stVendorSpecInfoFrame	Data structure of the vendor-specified information frame

[Note]

None

[See Also]

None

HI_HDMI_INFOFRAME_S

[Description]

Defines the HDMI information frame.

[Syntax]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
typedef struct hiUNF_HDMI_INFOFRAME_S
{
    HI_HDMI_INFOFRAME_TYPE_E    enInfoFrameType;  /* InfoFrame type */
    HI_HMDI_INFORFRAME_UNIT_U   unInforUnit;      /* InfoFrame date */
}HI_HDMI_INFOFRAME_S;
```

[Member]

Member	Description
enInfoFrameType	Information frame type enumeration variable
unInforUnit	Information frame union variable

[Note]

None

[See Also]

- HI_MPI_HDMI_SetInfoFrame
- HI_MPI_HDMI_GetInfoFrame

12.5 Error Codes

Table 12-1 describes the error codes of the HDMI APIs

Table 12-1 Error codes for HDMI APIs

Error Code	Macro Definition	Description
0xA0288001	HI_ERR_HDMI_NOT_INIT	The HDMI is not initialized.
0xA0288002	HI_ERR_HDMI_INVALID_PARA	The parameter is invalid.
0xA0288003	HI_ERR_HDMI_NUL_PTR	The pointer is null.
0xA0288004	HI_ERR_HDMI_DEV_NOT_OPEN	The HDMI is disabled.
0xA0288005	HI_ERR_HDMI_DEV_NOT_CONNECT	The HDMI is disconnected.
0xA0288006	HI_ERR_HDMI_READ_SINK_FAILED	The HDMI fails to read the sink.
0xA0288007	HI_ERR_HDMI_INIT_ALREADY	The HDMI is initialized.
0xA0288008	HI_ERR_HDMI_CALLBACK_ALREADY	The HDMI callback is registered.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Error Code	Macro Definition	Description
0xA0288009	HI_ERR_HDMI_INVALID_CALLBACK	The callback function is invalid.
0xA028800A	HI_ERR_HDMI_FEATURE_NO_USPPORT	Not supported.
0xA028800B	HI_ERR_HDMI_BUS_BUSY	Bus busy flag.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Contents

13 Proc Debugging Information 13-1
13.1 Overview 13-1
13.2 SYS 13-2
13.3 VB 13-4
13.4 LOG 13-7
13.5 CHNL 13-9
13.6 DSU 13-11
13.7 GROUP 13-16
13.8 H264E 13-21
13.9 JPEGE 13-26
13.10 RC 13-29
13.11 REGION 13-34
13.12 VENC 13-39
13.13 VDEC 13-41
13.14 VI 13-47
13.15 VO 13-56
13.16 VPSS 13-67
13.17 IVE 13-77
13.18 VDA 13-78
13.19 AI 13-81
13.20 AO 13-88
13.21 AENC 13-94
13.22 ADEC 13-95
13.23 MPEG4E 13-95
13.24 HDMI 13-98
13.25 ACODEC 13-102

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

13 Proc Debugging Information

13.1 Overview

The debugging information is obtained from the proc file system of Linux. The information reflects the status of the current system and can be used to locate and analyze problems.

[Directory]

/proc/umap

[Checklist]

File	Description
sys	Records the status of the SYS module.
vb	Records the status of the video buffer (VB).
logmpp	Records the debugging level of each module (for internal debugging).
chnl	Records the status of the channel (CHNL) module.
dsu	Records the status of the dedicated scaling unit (DSU).
grp	Records the attributes of the current encoding channel group and the statistics of the current encoding channel.
h264e	Records the encoding attributes, status, and history statistics of each channel during H.264 encoding.
jpege	Records the encoding attributes, status, and history statistics of each channel during JPEG encoding.
rc	Records the stream control attributes, status, and history statistics of each encoding channel.
rgn	Records the region management information about the video overlay on-screen display (OSD).
venc	Records the information about the video encoder.
vdec	Records the information about the video decoder.
vi	Records the information about the video input unit (VIU).

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

File	Description
vo	Records the information about the video output unit (VOU).
vpss	Records the information about the video pre-processing module.
ive	Records the status of the operators of the intelligent video engine (IVE).
vda	Records the status and attributes of the channel whose video detection analysis (VDA) function is enabled.
ai	Records the information about audio input (AI) channels.
ao	Records the information about audio output (AO) channels.
aenc	Records the audio encoding (AENC) information.
adec	Records the audio decoding (ADEC) information.
mpeg4e	Records the encoding attributes, status, and history statistics of each channel during MPEG4 encoding.
hdmi	Records the HDMI information.

[View Methods]

- You can run the **cat** command such as **cat /proc/umap/venc** on the console. You can also run file operation commands such as **cp /proc/umap/ ./ -rf** to copy all the proc files under umap to the current directory.
- You can read the preceding files as common read-only files through applications such as fopen and fread.

NOTE

Note the following when reading the parameter descriptions:

- For the parameter whose value is **0** or **1**, if the mapping between the values and the definitions is not specified, the value **1** indicates affirmative and the value **0** indicates negative.
- For the parameter whose value is **aaa**, **bbb**, or **ccc**, if the mapping between the values and the definitions is not specified, you can identify the parameter definitions based on **aaa**, **bbb**, or **ccc**.

13.2 SYS

[Debugging Information]

```
# cat /proc/umap/sys
[SYS] Version: [Hi3531_MPP_V1.0.0.0 Debug], Build Time[Dec 20 2011, 10:40:10]
System State: 0 (0: initialized; 1: exiting; 2: exited)
-----MEM TABLE-------------------------------------------------------------
   MOD          MODNAME DEV CHN          MMZNAME
     5            grp   1   0          ddr1
     5            grp   3   0          ddr1
     5            grp   5   0          ddr1
     5            grp   7   0          ddr1
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
-----BIND RELATION
TABLE-----------------------------------------------------
FirMod FirDev FirChn  SecMod  SecDev  SecChn  TirMod  TirDev  TirChn SendCnt
rstCnt
vpss    0      2      vo       0       0     null      0       0      0     0
vi     0       0     vpss      0       2      vo       0       0     448    0
```

[Analysis]

This section records the status of the SYS module.

[Parameter Description]

Parameter		Description
System State	initialized	Initialized status
	exiting	Exiting status
	exited	Exited status
MEMTABLE	MOD	Module ID
	MODNAME	Module name
	DEV	Device ID corresponding to a module
	CHN	Channel ID corresponding to a module
	MMZNAME	Media memory zone (MMZ) name. Typically, the MMZ name is the DDR name. If the DDR name is NULL, the MMZ name is not displayed.
BIND RELATION TABLE	FirMod	Level-1 module name in the settings of the binding relationship. Data is transferred from level 1 to level 2.
	FirDev	Level-1 device name in the settings of the binding relationship. Data is transferred from level 1 to level 2.
	FirChn	Level-1 channel name in the settings of the binding relationship. Data is transferred from level 1 to level 2.
	SecMod	Level-2 module name in the settings of the binding relationship. Data is transferred from level 1 to level 2.
	SecDev	Level-2 device name in the settings of the binding relationship. Data is transferred from level 1 to level 2.
	SecChn	Level-2 channel name in the settings of the binding relationship. Data is transferred from level 1 to level 2.
	TirMod	Level-3 module name in the settings of the binding relationship. If there is level-3 binding relationship, data is transferred from level 2 to level 3. If there is no level-3 binding relationship, this parameter is null.
	TirDev	Level-3 device name in the settings of the binding relationship

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	TirChn	Level-3 channel name in the settings of the binding relationship
	SendCnt	Amount of data transferred from level 1 to level 2. The data amount is in the unit of frame. The amount of data transferred from the VPSS to the VOU is always 0, because SYS does not count such data. You need to view the data amount from the proc information about each module.
	rstCnt	Number of times that level 1 resets level 2

13.3 VB

[Debugging Information]

```
# cat /proc/umap/vb
[VB] Version: [Hi3531_MPP_V1.0.0.0 Debug], Build Time[Nov 22 2012, 16:56:02]
-----VB PUB CONFIG------------------------------------------------
Max Count of Pools:       256
-----COMMON POOL CONFIG-------------------------------------------
PoolId      0      1      2      3      4      5      6      7      8
9  …  15
Size    884736 1107968 4149248    3616      0      0      0      0      0
0 … 0
Count       0    128      32       48       0      0      0      0      0
0 …  0
 ------------------------------------------------------------------
PoolId    PhysAddr    VirtAddr    IsComm    BlkSz    BlkCnt     Free
MinFree
0  0x85de0000  0xcf000000        1   1107968      128      124(124)
123
BLK   VIU   VOU   DSU  VENC  VDEC VDA H264E JPEGE MPEGE H264D JPEGD MPEGD
VPSS …
85     0     1     0     0     0     0     0     0     0     0     0     0     0
0   …
86     0     1     0     0     0     0     0     0     0     0     0     0     0
0   …
87     1     0     0     0     0     0     0     0     0     0     0     0     0
0   …
88     1     0     0     0     0     0     0     0     0     0     0     0     0
0   …
Sum      2     2     0     0     0     0     0     0     0     0     0     0     0
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
0   …
--------------------------------------------------------------------
PoolId    PhysAddr    VirtAddr    IsComm    BlkSz   BlkCnt    Free    MinFree
1  0x8e520000  0xd8000000        1   4149248       32       32(32)       32
--------------------------------------------------------------------
PoolId    PhysAddr    VirtAddr    IsComm    BlkSz    BlkCnt    Free   MinFree
2   0x963c0000  0xc49c0000        1     3616       48       48(48)       48
ddr1----------------------------------------------------------------------
PoolId    PhysAddr    VirtAddr    IsComm    BlkSz    BlkCnt     Free
MinFree
3  0xc0000000  0xe0000000        0   4147200        6        2(2)        2
 BLK   VIU   VOU   DSU  VENC  VDEC   VDA H264E JPEGE MPEGE H264D JPEGD MPEGD
VPSS …
4     0     2     0     0     0     0     0     0     0     0     0     0     0
0 …
5     0     2     0     0     0     0     0     0     0     0     0     0     0
0 …
0     0     2     0     0     0     0     0     0     0     0     0     0     0
0 …
1     0     2     0     0     0     0     0     0     0     0     0     0     0
0 …
Sum      0     4     0     0     0     0     0     0     0     0     0     0     0
0     0
```

[Analysis]

This section records the status of VB.

[Parameter Description]

Parameter		Description
VB PUB CONFIG	Max Count of Pools	Maximum number of buffer pools
COMMON POOL CONFIG	PoolId	Handle of a public buffer pool
	Size	Size of the block in a buffer pool
	Count	Number of blocks in a buffer pool
NULL (it indicates the DDR that is not named.)	PoolId	Handle of a public or private buffer pool
	PhysAddr	Start physical address for a public or private buffer pool
	VirtAddr	Start virtual address for a public or private buffer pool
	IsComm	Whether the pool is a public buffer pool Value: {0, 1}

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	BlkSz	Size of the buffer in a buffer pool
	BlkCnt	Number of buffers in a buffer pool
	Free	Number of free buffers in a buffer pool
	MinFree	Minimum number of free buffers since the program runs. If the minimum number is 0, some frames may be lost because the buffers are insufficient.
	BLK	Handle of the buffer in a buffer pool
	VIU/VOU/DSU/VENC/VDEC/VDA/H264E/JPEGE/MPEGE/H264D/JPEGD/MPEGD/VPSS/GRP/MPI/PCIV/AI/AENC/RC	Module name The value indicates the number of times that a buffer in the buffer pool is occupied by the current module. 0: not occupied Other values: number of times that a buffer is occupied
ddr1 (it indicates the DDR that is named ddr1)	PoolId	Handles of a public or private buffer pool
	PhysAddr	Start physical address for a public or private buffer pool
	VirtAddr	Start virtual address for a public or private buffer pool
	IsComm	Whether the pool is a public buffer pool Value: {0, 1}
	BlkSz	Size of the buffer in a public or private buffer pool
	BlkCnt	Number of buffers in a public or private buffer pool
	Free	Number of free buffers in a public or private buffer pool
	MinFree	Minimum number of free buffers since the program runs. If the minimum number is 0, some frames may be lost because the buffers are insufficient.
	BLK	Handle of the buffer in a public or private buffer pool
	VIU/VOU/DSU/VENC/VDEC/VDA/H264E/JPEGE/MPEGE/H264D/JPEGD/MPEGD/VPSS/GRP/MPI/PCIV/AI/AENC/RC	Module name The value indicates the number of times that a buffer in the buffer pool is occupied by the current module. 0: not occupied Other values: number of times that a buffer is occupied

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

13.4 LOG

[Debugging Information]

```
# cat /proc/umap/logmpp
-----LOG BUFFER STATE----------------------------------------------
MaxLen  ReadPos WritePos ButtPos
64(KB)      0    1378   65536
-----CURRENT LOG LEVEL---------------------------------------------
cmpi    :  3
vb      :  3
sys     :  3
chnl    :  3
grp     :  3
venc    :  3
vpss    :  3
vda     :  3
h264e   :  3
jpege   :  3
mpeg4e  :  3
vdec    :  3
h264d   :  3
jpegd   :  3
vo      :  3
vi      :  3
dsu     :  3
rgn     :  3
rc      :  3
sio     :  3
ai      :  3
ao      :  3
aenc    :  3
adec    :  3
pciv    :  3
pcivfmw :  3
ive     :  3
vcmp    :  3
fb      :  3
hdmi    :  3
voie    :  3
```

[Analysis]

This section records the debugging level of each module (for internal debugging).

The **cat /proc/umap/logmpp** command is used to obtain the log level of the Hi35xx.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

[Parameter Description]

Parameter		Description
LOG BUFFER STATE	MaxLen	Log buffer size (in KB)
	ReadPos	Read pointer position
	WritePos	Write pointer position
	ButtPos	Buffer bottom position
CURRENT LOG LEVEL	cmpi	CMPI module name
	vb	VB module name
	sys	SYS module name
	chnl	CHNL module name
	grp	GROUP module name
	venc	VENC module name
	vpps	VPPS module name
	Vda	VDA module name
	h264e	H264E module name
	JPEGe	JPEGe module name
	mpeg4e	MPEG4E module name
	vdec	VDEC module name
	h264d	H264D module name
	JPEGd	JPEGd module name
	vo	VOU name
	vi	VIU name
	dsu	DSU name
	rgn	REGION module name
	rc	RC module name
	sio	SIO module name
	ai	AIU name
	ao	AOU name
	aenc	AENC module name
	adec	ADEC module name
	pciv	PCIV module name
	pcivfmw	Peripheral component interconnect video firmware

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		(PCIVFMW) module name
	ive	IVE module name
	vcmp	Video compress (VCMP) module name
	fb	FB module name
	hdmi	HDMI module name
	voie	VOIE module name

13.5 CHNL

[Debugging Information]

```
# cat /proc/umap/chnl

[CHNL] Version: [Hi3531_MPP_V1.0.0.0 Debug], Build Time[Sep  2 2011,
15:36:02]

Total Chnl Scheduler Count: 1
-----SCHEDULER 0 INFO-------------------------------------------------
SchdlId VpuNum
  0       1
-----VPU INFO---------------------------------------------------------
VpuId   Name    State   IntCnt  TimeCnt   VpuCnt  ErrCnt  InqCnt  StartOk
StartNo Config   Reset
0       VEUD_0  RUN     5252    312    4940      0     7735    4941      0
233       0
-----CHNL STATE------------------------------------------------------
ChnlId  Type    TskNum  State   InqCnt  StartOk StartNo  IntPro
0       H264E     6     IDLE    304     155      0        4940
-----CHNL PERF-------------------------------------------------------
ChnlId  Type    StartCost IntCost IntCostL  HWCost  HWCycle HWCostL HWCostAcc
0        H264E   2829      4354       6352    22234     0     25635     3
-----CHNL CURRENT RUN STATE------------------------------------------
VpuId   VpuName ChnlId  Type
0       VEUD_0    27    H264E
```

[Analysis]

This section records the status of the CHNL module.

[Parameter Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
SCHEDULER 0 INFO	SchdlId	Scheduler ID
	VpuNum	Number of video processing units (VPUs) managed by the scheduler
VPU INFO	VpuId	VPU ID
	Name	VPU name
	State	Running status of a VPU
	IntCnt	Number of times that the VPU is interrupted by the software and hardware
	TmCnt	Number of times that the VPU is interrupted by the software
	VpuCnt	Number of times that the VPU is interrupted by the hardware
	ErrCnt	Number of times that the VPU reports incorrect interrupt information
	InqCnt	Number of times that the VPU query tasks
	StartOk	Number of times that the VPU starts successfully
	StartNo	Number of times that the VPU fails to start
	Config	Number of times that the VPU is successfully configured
	Reset	Number of times that the VPU crashes
CHNL STATE	ChnlId	ID of the channel that is registered in the current scheduler
	Type	Channel type
	TskNum	Number of tasks in the current channel
	State	Status of the current task
	InqCnt	Number of times that the current task is queried
	StartOk	Number of times that the current task is started
	StartNo	Number of times that the current task fails to start
	IntPro	Number of times that the interrupt processing function processes the current task
CHNL PERF	ChnlId	ID of the channel that is registered in the current scheduler
	Type	Channel type
	StartCost	Time spent in starting the current task (in μs)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	IntCost	Time spent by the interrupt processing function for processing the current task (in μs)
	IntCostL	Maximum time spent by the interrupt processing function for processing the current task (in μs)
	HWCost	Time that the chip spent in processing the current task (in μs)
	HWCycle	Number of cycles for the current task
	HWCostL	Maximum time that the chip spent in processing a task slice in the currently scheduled channel (in μs)
	HWCostAcc	Number of times that the chip spent more than 30 ms in processing a task slice in the currently scheduled channel
CHNL CURRENT RUN STATE	VpuId	VPU ID
	VpuName	VPU name
	ChnlId	ID of the channel that is registered in the current scheduler
	Type	Channel type

13.6 DSU

[Debugging Information]

```
# cat /proc/umap/dsu

[DSU] Version: [Hi3531_MPP_V1.0.0.0 Debug], Build Time[Sep 15 2011, 19:41:34]

-----MODULE PARAM-----------------------------------------------------------
 max_job_num
         40
-----RECENT JOB INFO--------------------------------------------------------
   SeqNo   ModNameJobHdl TaskNum   State  InSize    OutSize CostTime
       0     VOU      13       4      ProcOk 1658880  414720   10533
       1     VOU      12       4      ProcOk 1658880  414720   10599
       2     VOU      11       4      ProcOk 1658880  414720   10583
       3     VOU      10       4      ProcOk 1658880  414720   10542
       4     VOU       9       4      ProcOk 1658880  414720   10573
       5     VOU       8       4      ProcOk 1658880  414720   10538
       6     VOU       7       4      ProcOk 1658880  414720   10605
       7     VOU       6       4      ProcOk 1658880  414720   10564
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
-----MAX WASTE TIME JOB INFO-----------------------------------------
   SeqNo   ModNameJobHdl TaskNum   State  InSize OutSize CostTime
       8     VOU      23       4   ProcOk    1658880  414720   11005

-----DSU JOB STATUS---------------------------------------------------
   Success      Fail    Cancel  AllJobNd    FreeNd   BeginNd  EndJobNd
ProcNd
      1294         0         0       40       39         0         0           1

-------DSU TASK STATUS-----------------------------------------------------
   Success      Fail    Cancel AllTaskNd    FreeNd    BusyNd
     10349         0         0      320       312         8

-----------DSU INT STATUS-------------------------------------------
    IntNum     IntTm        FifoDepth    HalProcTm
      1294        52               1        314

-------DSU MEM REQ STATUS ---------------------------------------------
     ReqOk    FreeOk   ReqFail  FreeFail
         0         0         0         0
-------DSU TASKLET STATUS ---------------------------------------------
    TslNum MaxTslNum     TslTm  MaxTslTm
        13        13     23265     24034

-----DSU JOB STATISTIC FOR EACH MOD-----------------------------------
     ModId   ModName    JobNum    MaxJob     psNum   MaxPNum
        15        vo        13        13         0         0
```

[Analysis]

This section records the information about the DSU, including the number of recently completed jobs, the most time-consuming job, history statistics, and interrupt information.

[Parameter Description]

Parameter		Description
MODULE PARAM	max_job_num	Maximum number of jobs. The default value is 40. This parameter value needs to be set during driver loading only when you want to change the number of jobs supported by the DSU.
RECENT JOB INFO	SeqNo	Print sequence Value range: [0, 7]
	ModName	Name of the module that submits a job
	JobHdl	Handle ID of a job, for debugging

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		Value range: [0, 39]
	TaskNum	Number of tasks in a job Value range: [0, 320)
	State	Status of a job Value range: {ProcNok, Procok, Procing} ProcNok: The job fails to be handled. Procok: The job is handled successfully. Procing: The job is being handled by the hardware.
	InSize	Total size of the input pictures involved in all the tasks of a job The size is expressed by pixel. Each time a task is added to the job, the value of InSize increases by adding the size of the input pictures of the task. The scaling performance of the DSU depends on the total sizes of the input pictures and output pictures (InSize/OutSize). In general, the larger the InSize and OutSize of a job, the longer (CostTime) the job is handled.
	OutSize	Total size of the output pictures involved in all the tasks of a job The size is expressed by pixel. Each time a task is added to the job, the value of OutSize increases by adding the size of the output pictures of the task.
	CostTime	Period spent in submitting and handling a job The period includes the processing time of the software, hardware, and interrupt service routine (ISR) relevant to the job, in μs. If the DSU performance is insufficient (for example, too many scaling tasks are submitted or the performance limit is exceeded), the period exceeds the expected value. For example, in the PAL preview scenario, if a job is handled more than 40 ms, the previewed pictures are paused intermittently. In this case, you can check whether the DSU performance limit is exceeded.
MAX WASTE TIME JOB INFO	The items are the same as those of RECENT JOB INFO.	Information of the most time-consuming job among 500 jobs The members are the same as those of RECENT JOB INFO. For details, see the preceding description. When a job consumes more time or the number of jobs is greater than 500, the value of this item is updated. The value reflects the recent performance of the DSU and indicates whether some jobs are not handled in a timely manner by the DSU.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
DSU JOB STATUS	Success	Number of jobs that are handled successfully The number increases by 1 when a job is successfully handled by the hardware.
	Fail	Number of jobs that fail to be handled The number increases by 1 when the DSU fails to submit a job to the driver layer. If the number increases, you can view the log to determine the failure causes.
	Cancel	Number of jobs that are proactively canceled by users The number increases by 1 each time users call the cancelJob interface. If a task fails to be added to a job, users call the cancelJob interface. If the number increases, you can view the log to determine the failure causes.
	AllJobNd	Number of DSU job nodes. This number is consistent with the value of max_job_num and is typically set to 40.
	FreeNd	Number of free job nodes
	BeginNd	Number of jobs that are created but not submitted (EndJob interface)
	EndJobNd	Number of jobs that are submitted (EndJob interface) but not transferred to the hardware
	ProcNd	Number of jobs that are being handled by the hardware
DSU TASK STATUS	Success	Number of tasks that are handled successfully A job consists of one or more scaling tasks. If a job is handled successfully, its tasks are all handled successfully. Therefore, this number increases faster than the Success item of the job. When a job is successfully handled by the hardware, the number increases by adding the tasks of the job.
	Fail	Number of tasks that fail to be handled If a job fails to be handled, its tasks all fail to be handled. When a job fails to be handled, the number increases by adding the number of failed tasks. If the number increases, you can view the log to determine the failure causes.
	Cancel	Number of tasks that are proactively canceled by users When users call the cancelJob interface to cancel a job, all the tasks of the job are canceled and the number increases. If the number increases, you can view the log to determine the failure causes.
	AllJobNd	Number of DSU task nodes. This number is typically set to 320.
	FreeNd	Number of free task nodes

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	BusyNd	Number of tasks that are added to a job
DSU INT STATUS	IntNum	Number of DSU interrupts This number increases by 1 when a job is handled by the hardware.
	IntTm	Time of processing a DSU interrupt, in μs The time includes the time of processing a DSU interrupt and calling a callback interface.
	FifoDepth	FIFO depth of the hardware, for internal debugging
	HalProcTm	Time (in μs) of submitting a job to the driver layer, for internal debugging.
DSU MEM REQ STATUS	ReqOk	Times of applying for VBs successfully The number increases by 1 each time a VB is applied successfully. Note: The DSU applies for or releases a VB only when users enable the OSD blending function or changes the video format to package.
	FreeOk	Number of times that VBs are released successfully The number increases by 1 each time a VB is released successfully. The number of ReqOk must be consistent with the value of ReqOk. This indicates that the memory is not leaked.
	ReqFail	Number of times that VBs fail to be applied The number is 0 in general and increases by 1 each time a VB fails to be applied.
	FreeFail	Number of times that VBs fail to be released The number is 0 in general and increases by 1 each time a VB fails to be released.
DSU TASKLET STATUS	TslNum	Number of times that tasklet is performed in the last second
	MaxTslNum	Historical maximum number of times that tasklet is performed in one second
	TslTm	Time in which the CPU is occupied when tasklet is performed in the last second, in μs
	MaxTslTm	Historical maximum time in which the CPU is occupied when tasklet is performed in one second, in μs
DSU JOB STATISTIC FOR EACH MODULE (the information about the	ModId	Module ID
	ModName	Module name
	JobNum	Number of times that the module calls DSU jobs (including bypass and non-bypass jobs) in the last second
	MaxJob	Historical maximum number of times that the module calls

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
modules that are not called is not displayed.)		DSU jobs in one second
	psNum	Number of times that the module calls DSU jobs in the last second
	MaxPNum	Historical number of times that the module calls DSU bypass jobs in one second

13.7 GROUP

[Debugging Information]

```
cat /proc/umap/grp

[GRP] Version: [Hi3531_MPP_V1.0.0.0 Debug], Build Time[Apr  9 2012, 15:52:20]

-----GROUP COLOR2GREY CONF----------------------------------------------
Enabled  MaxWidth MaxHeight
0          0        0
-----GROUP CHN ATTR-----------------------------------------------------
 NO.   ChnId  EnType   Width  Height   C2GEn VeStr  OsdStr SrcFr  TarFr
Timeref
 0      0     96        704    576      0     YES     YES     -1     -1
20065876
PixFmt   PicAddr
YUV420  0x8773f000
-------GROUP CHN RECEIVE STAT-----------------------------------------
NO.      Start    StartEx    RecvLeft    EncLeft
0          1          0          0          0
-------GROUP VPSS QUERY------------------------------------------------
 NO.   Query   QueryOk   QueryFR   Invld    Full  VbFail    Fail InfoErr  Stop
0    60074037 60074037   0       50041107    0    31600     0      0       0
-------GROUP SEND1-----------------------------------------------------
NO.  VpssSnd VInfErr OthrSnd OInfErr    Send    Stop    Full CropErr
DrectSnd  SizeErr
0    10001330  1          0       0      10001329   0         0     0
10001329    0
---------GROUP SEND2----------------------------------------------------
NO.  SendDsu  StartOk StartFail IntOk  IntFail SrcAdd  SrcSub   HistAdd
HistSub
0      0         0        0      0       0        0       0        0
0
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
DestAdd   DestSub
0          0
-------GROUP PIC QUEUE STATE-----------------------------------------
NO.    Free    busy     dsu
0       5       1       0
---------GROUP CHNL INFO-----------------------------------------------
NO.     Inq         InqOk      Start        StartOk    Config     GrpInt
0    69949490    10001328    10001328     9455744     9455744     9455744
-------GROUP CROP INFO-------------------------------------------------
NO.  CropEn  StartX  StartY   Width  Height
0     NO       0       0       0       0
```

[Analysis]

This section records the configurations and status of the current encoding channel group.

[Parameter Description]

Parameter		Description
GROUP COLOR2GREY CONF	Enabled	Global parameter that determines whether all channel groups allow color pictures to be converted into gray pictures for encoding.
	MaxWidth	Maximum width of a channel group that supports the color-to-gray function. If a channel group allows color picture to be converted into gray pictures for encoding (C2Gen = 1), the group width must be less than or equal to MaxWidth.
	MaxHeight	Maximum height of a channel group that supports the color-to-gray function. If a channel group allows color picture to be converted into gray pictures for encoding (C2Gen = 1), the group height must be less than or equal to MaxHeight.
GROUP CHN ATTR	NO.	GROUP channel ID
	ChnId	Encoding channel number registered in the GROUP
	EnType	Encoding channel type in GROUP, such as h264 and jpege
	Width	Width of the encoding channel in a GROUP channel
	Height	Height of the encoding channel in a GROUP channel
	C2GEn	Whether a channel group allows color pictures to be converted into gray pictures for encoding
	VeStart	Whether to start encoding
	OsdStart	Whether to enable OSD
	SrcFr	Source frame rate (input frame rate) for controlling the GROUP frame rate

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	TarFr	Target frame rate for controlling the GROUP frame rate
	Timeref	Timeref of the latest frame in the busy queue
	PixFmt	Format of the current frame that is being encoded
	PicAdr	Address of the current frame that is being encoded
GROUP CHN RECEIVE STAT	NO.	GROUP channel ID
	Start	Whether the channel group starts to receive pictures
	StartEx	Whether the channel group starts to receive pictures based on the frame number
	RecvLeft	Number of frames that are to be received by GROUP. This parameter is used with HI_MPI_VENC_StartRecvPicEx().
	EncLeft	Number of frames that are to be encoded by GROUP. This parameter is used with HI_MPI_VENC_StartRecvPicEx().
GROUP VPSS QUERY	NO.	GROUP channel ID
	Query	Total number of times that the VPSS queries the GROUP
	Query Ok	Number of times that the VPSS successfully queries the GROUP
	Query FR	Number of times the GROUP discards frames due to frame rate control, which is queried by the VPSS
	Invld	Number of times that the source picture of the GROUP is empty, which is queried by the VPSS
	Full	Number of times that the picture queue of the GROUP is full, which is queried by the VPSS
	VbFail	Number of times that VBs fail to be requested, which is queries by the VPSS
	Fail	Number of times that the VPSS fails to query the GROUP
	InfoErr	Number of times that the VPSS fails to query the GROUP due to incorrect information
	Stop	Number of times that the VPSS fails to query the GROUP because the GROUP stops receiving pictures
GROUP SEND1	NO.	GROUP channel ID
	VpssSnd	Number of times that the VPSS transmits pictures
	VInfErr	Number of times that the VPSS fails to transmit pictures due to incorrect picture information
	OthrSnd	Number of times that other modules transmit pictures
	OInfErr	Number of times that other modules fail to transmit pictures due to incorrect picture information

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	Send	Number of times that external modules transmit pictures properly
	Stop	Number of times that the GROUP stops receiving pictures when external modules are transmitting pictures
	Full	Number of times that the picture queue is full when external modules are transmitting pictures
	CropErr	Number of times that picture cropping fails
	DrectSnd	Number of times that the external modules transmit pictures to the picture busy queue
	SizeErr	Number of times that the GROUP refuses to receive pictures. This item increases when the width or height of the picture transmitted by the front-end is greater than the width or height of the GROUP.
GROUP SEND2	NO	GROUP channel ID
	SendDsu	Number of times that pictures are transmitted to the DSU
	StrtOk	Number of times that the DSU starts successfully
	StartFail	Number of times that the DSU fails to start
	IntOk	Number of interrupts indicating that the DSU tasks are performed successfully
	IntFail	Number of interrupts indicating that DSU tasks fail to be performed
	SrcAdd	Number of times that the count of using the source picture VBs increases
	SrcSub	Number of times that the count of using the source picture VBs decreases
	HistAdd	Number of times that the count of using the histogram VBs increases
	HistSub	Number of times that the count of using the histogram VBs decreases
	DestAdd	Number of times that the count of using the target picture VBs increases
	DestSub	Number of times that the count of using the target picture VBs decreases
GROUP PIC QUEUE STATE	NO	GROUP channel ID
	Free	Number of the nodes in the GROUP free queue (number of frames allowed)
	Busy	Number of the nodes in the GROUP busy queue (number of received and used frames)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	Dsu	Number of frames being processed by the DSU
GROUP CHNL INFO	GNO.	GROUP channel group ID
	Inq	Number of times that Chnl is queried
	InqOk	Number of times that Chnl queries successfully
	Start	Number of times that Chnl starts encoding
	StartOk	Number of times that Chnl starts encoding successfully
	Config	Number of times that Chnl configures encoding hardware
	GrpInt	Number of times that Chnl receives interrupt
GROUP CROP INFO	NO.	GROUP channel ID
	CropEn	Whether to enable the group cropping function of the encoding channel
	StartX	Start horizontal coordinate of the cropped region
	StartY	Start vertical coordinate of the cropped region
	Width	Width of the cropped region
	Height	Height of the cropped region

13.8 H264E

[Debugging Information]

```
# cat /proc/umap/h264e
[H264E] Version: [Hi3531_MPP_V1.0.0.0 ], Build Time[Dec  3 2014, 11:34:44]
-----MODULE PARAM-----------------------------------------------------
ContiBuffer  H264eVBSource   H264eRcnEqualRef    H264eMiniBufMode
       0            1           0                       0
-----CHN ATTR---------------------------------------------------------
ID  MaxWidth MaxHeight  PicWidth PicHeight   profile   MainStr   RefMode
BufSize   ByFrame   MaxStrCnt  PoolId
0     1920     1080      1920      1080        hp       Yes       1X
4147200      No          200       4
-----PICTURE INFO-----------------------------------------------------
ID    EncdStart   EncdSucceed    Lost    Disc    Skip  BufLeak   PSkip
Recode     RlsStr  UnrdStr   GetVBFail
0          161          161         0       0       0       0       0      0
152       9          0
-----STREAM BUFFER----------------------------------------------------
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
ID     Base    RdTail    RdHead      WrTail     WrHead     DataLen   BufFree
0     0xe7400000  0xbb500     0xea800     0xea800    0xea800    0    3955904
-----RefParam INFO-------------------------------------------
     ID    EnPred      Base   Enhance     EnIDR
      0      Yes        1        0      Yes
-----ROI INFO-----------------------------------------------------------
ID     index    bAbsQp    Qp     width    height    startx    starty
-----Syntax INFO-------------------------------------------------------
ID   slcsplt   slcmode   slcsize
0      No       N/A      N/A
-----Inter & Intra prediction INFO---------------------------------
ID   profile  HWsize  VWsize  P16x16   P16x8   P8x16    P8x8   MvExt  I16x16
Inxn  Ipcm
0     hp      9      4     Yes     Yes     Yes     Yes     Yes     Yes     Yes
No
-----Syntax INFO-------------------------------------------------------
ID   profile   EntrpyI   EntrpyP  Itrans  ptrans QMatrix   POC   DblkIdc
alpha    beta
0       hp     cabac     cabac     all     all     No      2      0     0      0
```

[Analysis]

During multi-channel H.264 encoding, the upper-layer software may not fetch streams in a timely manner. As a result, the stream buffer is insufficient, which causes frame loss. You can view the Disc, BufLeak, and UnrdStr parameters to check whether frames are lost. BufLeak indicates the number of frames that are discarded when the software detects stream buffer insufficiency. Disc indicates the number of frames that are discarded when the hardware detects stream buffer insufficiency. BufLeak and Disc indicate the total number of discarded frames due to stream buffer insufficiency. UnrdStr indicates the number of remaining frames in the stream buffer. If the value of UnrdStr is great, streams are not fetched in a timely manner.

[Parameter Description]

Parameter		Description
MODULE PARAM	ContiBuffer	Whether the virtual address for the user-mode buffer is consecutive 0: inconsecutive 1: consecutive
	H264eVBSource	Allocation mode of the buffers for storing encoded frames 1: PrivateVB pool mode 2: UserVB pool mode
	H264eRcnEqualRef	Whether the reference frame shares the memory for storing the luminance data with the reconstruction frame 0: no

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		1: yes
	H264eMiniBufMode	Configuration mode of the stream buffer 0: normal mode 1: memory reduction mode
CHN ATTR	ID	Channel ID
	MaxWidth	Maximum width of the encoding channel (in pixel)
	MaxHeight	Maximum height of the encoding channel (in pixel)
	PicWidth	Width of the picture to be encoded (in pixel)
	PicHeight	Height of the picture to be encoded (in pixel)
	profile	Encoding channel profile Base: baseline Mp: main profile Hp: high profile
	MainStr	Large stream Value: {0, 1}
	RefMode	Frame skipping reference mode of an encoding channel 1X: 1x frame skipping reference mode 2X: 2x frame skipping reference mode 4X: 4x frame skipping reference mode
	BufSize	Stream buffer size (in byte)
	ByFrame	Whether to obtain streams by frame Value: {0, 1}
	MaxStrCnt	Maximum number of frames in the stream buffer Default value: 0x1FFFFFFF
	PoolId	VB pool that the frame buffer for the current encoding channel belongs to
PICTURE INFO	ID	Channel ID
	EncdStart	Number of received pictures This value increases by 1 after the pictures to be encoded are received.
	EncdSucceed	Number of frames that are encoded successfully
	Lost	Number of frames that are discarded during encoding. The discarded frames include: • Number of frames that are discarded due to encoder exceptions or jumbo frames. That is, the

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		number of frames is the value of Disc. • Number of frames that are discarded for controlling the frame rate. The frame rate is controlled by the rate controller (RC). That is, the number is the value of Skip. • Number of frames that are discarded due to stream buffer insufficiency. That is, the number is the value of BufLeak.
	Disc	Number of frames discarded for the following reasons: • The encoder detects stream buffer insufficiency during encoding. • Jumbo frames occur. • An exception occurs in the encoder. For example, the encoder crashes. The first reason is the most common one. If the value of Disc continuously increases, it is probably that the upper-layer software does not fetch streams in a timely manner, which causes buffer insufficiency.
	Skip	Number of frames that are discarded for controlling the frame rate during encoding
	BufLeak	Number of discarded frames due to stream buffer insufficiency The number increases by 1 when the encoder detects stream buffer insufficiency and then discards a frame before encoding. The number is not 0 when streams are not fetched in a timely manner.
	Recode	Number of frames that are re-encoded by the encoder
	UnrdStr	Number of frames that are not obtained by users
	RlsStr	Number of frames that are obtained and released by users
	GetVBFail	Number of times that the VB fails to be obtained
STREAM BUFFER	ID	Channel ID
	base	Base address for a stream buffer
	RdTail	Read tail pointer
	RdHead	Read head pointer
	WrTail	Write tail pointer
	WrHead	Write head pointer

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	datalen	Data length
	buffree	Free buffer size
RefParam INFO	ID	Channel ID
	EnPred	Whether some frames at the base layer are referenced by other frames at the base layer
	Base	Base layer period
	Enhance	Enhance layer period
	EnIDR	IDR frame enable
ROI INFO	ID	Channel ID
	index	Region of interest (ROI) index
	bAbsQp	Whether the ROI uses the absolute quantizer parameter (QP) mode
	Qp	QP value configured by the ROI
	width	ROI width (in pixel)
	height	ROI height (in pixel)
	startx	Start horizontal coordinate of the ROI (in pixel)
	starty	Start vertical coordinate of the ROI (in pixel)
Syntax INFO	ID	Channel ID
	slcsplt	Whether to split with slice
	slcmode	Slice split mode • ByteNum: split by byte. • MbLine: split by macroblock line.
	slcsize	Size of each slice When the slice is split by byte, it indicates the number of bytes of each slice. When the slice is split by macroblock line, it indicates the number of macroblock lines of each slice.
Inter & Intra prediction INFO	ID	Channel ID
	profile	Encoding channel profile type: • Base: baseline. • MP: main profile. • HP: high profile.
	HWsize	Size of the horizontal search window

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	VWsize	Size of the vertical search window
	P16x16	Whether to enable inter16x16 prediction mode
	P16x8	Whether to enable inter16x8 prediction mode
	P8x16	Whether to enable inter8x16 prediction mode
	P8x8	Whether to enable inter8x8 prediction mode
	MvExt	Whether to extend the boundaries when search reaches the boundaries of the pictures
	I16x16	Whether to enable the intra16x16 prediction mode
	Inxn	Whether to enable intra8x8 and intra4x4 prediction modes
	Ipcm	Whether to enable the Ipcm prediction mode
Syntax INFO	ID	Channel ID
	profile	Encoding channel profile type: • Base: baseline. • MP: main profile. • HP: high profile.
	EntrpyI	Entropy encoding mode of the I frame, context-based adaptive binary arithmetic coding (CABAC) or context-based adaptive variable-length coding (CAVLC)
	EntrpyP	Entropy encoding mode of the P frame, CABAC or CAVLC
	Itrans	Transform mode used by I frame: • All: The trans8x8 mode and trans4x4 mode are enabled simultaneously. • 4x4: The trans4x4 mode is enabled. • 8x8: The trans8x8 mode is enabled.
	ptrans	Transform mode used by P frame: • All: The trans8x8 mode and trans4x4 mode are enabled simultaneously. • 4x4: The trans4x4 mode is enabled. • 8x8: The trans8x8 mode is enabled.
	QMatrix	Whether a customized quantization table is used
	POC	Value of the syntax element pic_order_cnt_type
	DblkIdc	Value of the syntax element disable_deblocking_filter_idc
	alpha	Value of the syntax element

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		slice_alpha_c0_offset_div2
	beta	Value of the syntax element slice_beta_offset_div2

13.9 JPEGE

[Debugging Information]

```
[JPEGE] Version: [Hi3531_MPP_V1.0.0.0], Build Time[Dec  3 2014, 11:34:45]
-----MODULE PARAM-----------------------------------------------------
ContiBuffer    JpegeMiniBufMode
0                   0
-----ATTRIBUTE-------------------------------------------------------
ID  bMjpeg   PicType  MaxWidth MaxHeight  PicWidth PicHeight   ViFld   bMain
ByFrm    MCU Qfactor
0    Yes    YUV420      1920     1080      704      576        No     Yes
Yes    0     48
-----STATUS1---------------------------------------------------------
ID            BufLen       FreeLen        StrmCnt        MaxStrm
0            4149248        775744         59             200
-----STATUS2---------------------------------------------------------
ID    PicRec    PicCoded    PicDroped    PicDisc    NoStmCnt    NoBuf
RcFail    FullInt    PicRecode
0    812      643        168        0      0       168     0     0    0
-----STREAM BUFFER---------------------------------------------------
ID     Base          RdTail  RdHead  WrTail  WrHead  BufLen  DataLen
BufFree
0     0xcb400000     788480  11904   11904   12672   4149248    0    775744
```

[Analysis]

This section records the encoding attributes, status, and history statistics of each channel during JPEG encoding. A maximum of 64 encoding channels are supported. The information is used to locate problems such as system blocking and frame loss.

[Parameter Description]

Parameter		Description
MODULE PARAM	ContiBuffer	Whether the virtual address for the user-mode buffer is consecutive 0: inconsecutive 1: consecutive
	JpegeMiniBuf	Configuration mode of the stream buffer

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	Mode	0: normal mode 1: memory reduction mode
ATTRIBUTE	ID	Channel ID
	bMjpeg	MJPEG encoding No: JPEG snapshot Yes: MJPEG encoding
	PicType	Picture type: YUV422 or YUV420
	MaxWidth	Maximum width of the encoding channel (in pixel)
	MaxHeight	Maximum height of the encoding channel (in pixel)
	PicWidth	Picture width
	PicHeight	Picture height
	ViFld	Attributes of a VI picture 0: frame 1: field
	MCU	Number of minimum coded units (MCUs) in each embedded CPU subsystem (ECS)
	ByFrm	Whether to obtain streams by frame Value: {0, 1}
	bMain	Large stream Value: {0, 1}
	Qfactor	Channel Qfactor
	ContiBuffer	Whether the virtual address for the user-mode stream buffer is consecutive 0: inconsecutive 1: consecutive
Status1	ID	Channel ID
	BufLen	Total length of the stream buffer
	FreeLen	Length of the free stream buffer
	StrmCnt	Number of frames in the current stream buffer
	MaxStrm	Maximum number of frames in the stream buffer Default value: 100,000
Status2	ID	Channel ID
	PicRec	Number of pictures that are transmitted for encoding
	PicCoded	Number of frames that are successfully encoded

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	PicDroped	Total number of discarded pictures before encoding, including the discarded pictures of Nostmcnt, NoBuf, and RcFail
	PicDisc	Number of frames discarded for the following reasons: • The encoder detects stream buffer insufficiency during encoding. • Jumbo frames occur. • An exception occurs in the encoder. For example, the encoder crashes. The first reason is the most common one. If the value of Disc continuously increases, it is probably that the upper-layer software does not fetch streams in a timely manner, which causes buffer insufficiency.
	Nostmcnt	Number of times that frames are discarded because the number of frames in the stream buffer exceeds upper limit
	NoBuf	Number of times that the system detects stream buffer insufficiency (which leads to frame loss) The number increases by 1 when the encoder detects stream buffer insufficiency and then discards a frame before encoding. The number is not 0 when streams are not fetched in a timely manner.
	RcFail	Number of times that the frame rate or bit rate fails to be controlled (which leads to frame loss)
	FullInt	Number of times that the hardware reports the interrupt indicating full buffer If the encoder detects stream buffer insufficiency during encoding, it will report a full buffer interrupt. The number is not 0 when streams are not fetched in a timely manner.
	PicRecode	Number of jumbo frames that are encoded again
STREAM BUFFER	ID	Channel ID
	base	Base address for a stream buffer
	RdTail	Read tail pointer
	RdHead	Read head pointer
	WrTail	Write tail pointer
	WrHead	Write head pointer
	Buflen	Stream buffer length
	datalen	Data length

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	buffree	Free buffer size

13.10 RC

[Debugging Information]

```
[RC] Version: [Hi3531_MPP_V1.0.D.0 ], Build Time[Mar 27 2015, 17:01:18]
----- BASE PARAMS -------------------------------------------------
ID    Gop  StatTm  ViFr      TrgFr ProType  RcMode  Br(kbps)  FluLev  IQp
PQp   MinQp  MinIQp   MaxQp
0     30       1    30       30/0      96   CBRV2      4096       0  N/A  N/A
10      10      51
-----RUN COMM PARAM 1----------------------------------------
    ID   QpDelta                    ThrdI(12)
     0        2  [  5,  5,  5, 10, 10, 10,255,255,255,255,255,255]
-----RUN COMM PARAM 2----------------------------------------------
    ID     PARAM                    ThrdP(12)
     0   DEFAULT  [  5,  5,  5,255,255,255,255,255,255,255,255,255]

-----RUN COMM PARAM 3----------------------------------------
    ID   bLost     LostThr  LostMode   LostGap      Priority
     0       0    83886080    NORMAl         0       BitRate

-----RUN CBR PARAM1 -----------------------------------------------
ID  MinIprop  MaxIprop   MaxQp   MaxStQp   MinQp  MaxPPDltQp  MaxIPDltQp
IPDltQp  QLevel  MaxReEncTimes
0        1       20     51       51     10          3          5       6
3            2

-----RUN CBR PARAM2 -------------------------------------------------------
    ID   SprFrmMod   SprIFrm   SprPFrm   RQRtio0   RQRtio1   RQRtio2
RQRtio3   RQRtio4   RQRtio5   RQRtio6   RQRtio7
     0       None    500000    500000       50       75       75       80
80       80       80        0

-----RUN VBR PARAM -------------------------------------------------
    ID   DltQp   ChgPs   SprFrmMod   SprIFrm   SprPFrm  MinIprop  MaxIprop

-----RUN INFO-----------------------------------------------------
ID   InsBr(kbps)   InsFr    WatL   CfgBt(kb)  RealBt(kb)   MaxRlBt(kb)
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
StartQp   MinQp   MaxQp
0         3567     29    4600        107        146          936       24
10     51

-----RUN INFO2-------------------------------------------------------
ID    IpRt      Tr   Delta   TbTar     Tb

-----IMG INFO-------------------------------------------------------
    ID   InterRl   IntraRl  InterSum  IntraSum    MvCnt   InterMb    Status
```

The following is the proc information in VBR mode:

```
# cat /proc/umap/rc

[RC] Version: [Hi3531_MPP_V1.0.D.0 ], Build Time[Mar 31 2015, 16:30:08]
----- BASE PARAMS ---------------------------------------------------
ID    Gop  StatTm  ViFr      TrgFr ProType  RcMode  Br(kbps)  FluLev  IQp
PQp   MinQp  MinIQp   MaxQp
0     25       1    25       25/0      96     VBR      6144     N/A  N/A  N/A
10      10      40
-----RUN COMM PARAM 1-----------------------------------------------
ID   QpDelta                    ThrdI(12)
0         0  [255,255,255,255,255,255,255,255,255,255,255,255]
-----RUN COMM PARAM 2-----------------------------------------------
ID     PARAM                    ThrdP(12)
0   DEFAULT  [255,255,255,255,255,255,255,255,255,255,255,255]
-----RUN COMM PARAM 3-------------------------------------------
ID   bLost     LostThr  LostMode   LostGap      Priority
0       0    83886080    NORMAl         0     SuperFrame
-----RUN CBR PARAM1 -------------------------------------------------
ID  MinIprop  MaxIprop   MaxQp   MaxStQp   MinQp  MaxPPDltQp  MaxIPDltQp
IPDltQp  QLevel  MaxReEncTimes

-----RUN CBR PARAM2 -------------------------------------------------
ID   SprFrmMod   SprIFrm   SprPFrm   RQRtio0   RQRtio1   RQRtio2   RQRtio3
RQRtio4   RQRtio5   RQRtio6   RQRtio7

-----RUN VBR PARAM -------------------------------------------------
    ID   DltQp   ChgPs   SprFrmMod   SprIFrm   SprPFrm  MinIprop  MaxIprop
     0      2     90        None    500000    500000         1        20
-----RUN INFO-------------------------------------------------------
    ID   InsBr(kbps)   InsFr    WatL   CfgBt(kb)  RealBt(kb)   MaxRlBt(kb)
StartQp   MinQp   MaxQp
     0           0      0     384          0          0            0        0
0      0
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
-----RUN INFO2-----------------------------------------------------
    ID    IpRt      Tr   Delta   TbTar      Tb
     0       0       0       0       0       0
-----IMG INFO------------------------------------------------------
    ID   InterRl   IntraRl  InterSum  IntraSum    MvCnt   InterMb    Status
     0        0        0        0        0        0        0        7
```

The following is proc information in FixQp mode:

```
[RC] Version: [Hi3531_MPP_V1.0.D.0 ], Build Time[Mar 27 2015, 17:01:18]
----- BASE PARAMS -------------------------------------------------------
    ID    Gop  StatTm  ViFr      TrgFr ProType  RcMode  Br(kbps)  FluLev
IQp  PQp   MinQp  MinIQp   MaxQp
     0    25       1    25       25/0     96   FIXQP      N/A    N/A   20
23    N/A    N/A    N/A
-----RUN COMM PARAM 1----------------------------------------------------
    ID   QpDelta                    ThrdI(12)
     0        0  [255,255,255,255,255,255,255,255,255,255,255,255]
-----RUN COMM PARAM 2----------------------------------------------------
    ID    PARAM                    ThrdP(12)
     0   DEFAULT  [255,255,255,255,255,255,255,255,255,255,255,255]
-----RUN COMM PARAM 3----------------------------------------------------
    ID   bLost    LostThr  LostMode   LostGap      Priority
     0       0    83886080    NORMAl        0    SuperFrame
-----RUN CBR PARAM1 -----------------------------------------------
    ID  MinIprop  MaxIprop   MaxQp   MaxStQp   MinQp  MaxPPDltQp  MaxIPDltQp
IPDltQp  QLevel  MaxReEncTimes
-----RUN CBR PARAM2 -----------------------------------------------
    ID   SprFrmMod   SprIFrm   SprPFrm   RQRtio0   RQRtio1   RQRtio2
RQRtio3   RQRtio4   RQRtio5   RQRtio6   RQRtio7
-----RUN VBR PARAM ------------------------------------------
    ID   DltQp   ChgPs   SprFrmMod   SprIFrm   SprPFrm  MinIprop  MaxIprop
-----RUN INFO-----------------------------------------------------
ID   InsBr(kbps)   InsFr    WatL   CfgBt(kb)  RealBt(kb)   MaxRlBt(kb)
StartQp   MinQp   MaxQp
0           0      0      0          0          0           0        0
0      0
-----RUN INFO2----------------------------------------------------
    ID    IpRt      Tr   Delta   TbTar      Tb
-----IMG INFO-----------------------------------------------------
    ID   InterRl   IntraRl  InterSum  IntraSum    MvCnt   InterMb    Status
```

[Analysis]

This section records the bit rate control information during encoding.

[Parameter Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
BASE PARAMS	ID	ID of a VENC channel
	Gop	Encoding group of pictures (GOP)
	StatTm	Bit rate statistics (in second)
	ViFr	Frame rate for transmitting pictures by the VIU
	TrgFr	Target frame rate for encoding
	ProType	Encoding type
	RcMode	Bit rate control mode (CBR, VBR, or FixQp)
	Br (kbit/s)	In CBR mode, it indicates the average bit rate. In VBR mode, it indicates the maximum bit rate. The unit of bit rate is kbit/s.
	FluLev	Fluctuation level, valid only in CBR mode
	IQp	I frame QP, valid only in FixQp mode
	PQp	P frame QP, valid only in FixQp mode
	MinQp	Minimum QP, valid only in VBR mode
	MinIQp	Minimum QP of the I frame, valid in CBR and VBR modes
	MaxQp	Maximum QP, valid only in VBR mode
RUN COMM PARAM 1		For details, see the description of VENC_RC_PARAM_S in chapter 6 "VENC."
RUN COMM PARAM 2		For details, see the description of VENC_RC_PARAM_S in chapter 6 "VENC."
RUN COMM PARAM 3	ChnId	Channel ID
	bLost	Whether to discard frames when the instantaneous bit rate exceeds the threshold
	LostThr	Instantaneous bit rate threshold
	LostMode	Frame discarding policy
	LostGap	Maximum number of consecutive discarded frames
	Priority	RC priority BitRate: The target bit rate takes priority. SuperFrame: The jumbo frame takes priority.
RUN CBR PARAM		For details, see the descriptions of VENC_PARAM_H264_CBR_S and VENC_PARAM_H264_CBRV2_S in chapter 6 "VENC."
RUN VBR PARAM		For details, see the description of VENC_PARAM_H264_VBR_S in chapter 6

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		"VENC."
RUN INFO	ChnId	ID of a VENC channel
	InsBr (kbit/s)	Instant bit rate (in kbit/s)
	InsFr	Instant frame rate
	WatL	Bit rate threshold (for internal debugging)
	CfgBt(kb)	Target size of the current frame (in kbit)
	RealBt(kb)	Actual size of the streams in the previous frame (in kbit)
	MaxRlBt(kb)	Maximum size of the streams in a frame (in kbit)
	StartQp	Start QP
	MinQp	Minimum QP
	MaxQp	Maximum QP
RUN INFO2	ChnId	ID of a VENC channel
	IpRt	Ratio of the I frame size to the P frame size. The value must be divided by 64.
	Tr	Target bit rate of the current frame that is adjusted based on the instant bit rate
	Delta	Threshold reduction factor (for internal debugging)
	TbTar	Target threshold bit rate
	Tb	Target bit rate of the current frame that is adjusted based on the threshold bit rate
IMG INFO	ChnId	ID of a VENC channel
	InterRl	Coefficient related to the inter-frame information
	IntraRl	Coefficient related to the intra-frame information
	InterSum	MAD value of the VPP module
	IntraSum	Mean square error (MSE) value of the VPP module
	MvCnt	Sum of MV vectors
	InterMb	Macroblock proportion of the inter-frame mode
	Status	Scenario mode

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

13.11 REGION

[Debugging Information]

```
# cat /proc/umap/rgn

[RGN] Version: [Hi3531_MPP_V1.0.0.0 Debug], Build Time[Sep 16 2011, 11:38:03]

-----REGION STATUS OF OVERLAY-----------------------------------------
 Hdl type Used PiFmt   W   H BgColor     Phy     Virt  Stride
   1    0    0     8 180 144    fc00 87c7e800 c90ed800    384

-----REGION CHN STATUS OF OVERLAY---------------------------------------
 Hdl typ mod dev chn bSh   X   Y AphF AphB lay bAQp QP bInv InvW InvH Luma ChnM
0    0   5    0   0    1  64  64  128    0    0    0   0    1   16    16   50
0

-----REGION STATUS OF COVER-------------------------------------------
Hdl type Used
0    1    0

-----REGION CHN STATUS OF COVER-----------------------------------------
 Hdl type  mod  dev  chn bShow   X   Y   W   H   Field Color   lay
   0    1   16    0    0     0 100  82 260 260    frame f800     1
-----REGION STATUS OF COVEREX-----------------------------------------
 Hdl type Used
0     2    0

-----REGION CHN STATUS OF COVEREX---------------------------------------
Hdl type  mod  dev  chn bShow   X   Y   W   H   Field  Color   lay
0     2   16    0    0     1    96  96  64  64  frame   ff     3

-----REGION STATUS OF OVERLAYEX-----------------------------------------
Hdl type Used PiFmt   W   H   BgColor     Phy     Virt      Stride
0     3     0     8    64  64    fc     8f15d000  c4cc0000    128

-----REGION CHN STATUS OF OVERLAYEX-------------------------------------
 Hdl typ mod dev chn Field bSh   X   Y  AphF AphB lay
0    3   16    0    0   frame  1   96  12  255  255   3
```

[Analysis]

This section records the information about the current region.

[Parameter Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
REGION STATUS OF OVERLAY	Hdl	Overlay handle ID
	type	Overlay type. Its value is 0.
	Used	Number of times that resources are being used
	PiFmt	Overlay pixel format. For details, see PIXEL_FORMAT_E.
	W	Overlay width (in pixel)
	H	Overlay height (in pixel)
	BgColor	Overlay background color
	Phy	Physical address for the memory occupied by the Overlay region
	Virt	Virtual address for the memory occupied by the Overlay region
	Stride	Overlay memory stride (in byte)
REGION CHN STATUS OF OVERLAY	Hdl	Overlay handle ID
	type	Overlay type. Its value is 0
	mod	Module ID. The ID of the GROUP module is 5.
	dev	Device ID
	dSh	Whether to show the overlay region in the channel 0: hide 1: show
	chn	Channel ID
	X	Horizontal coordinate of the start position of the overlay region in the channel
	Y	Vertical coordinate of the start position of the overlay region in the channel
	AphF	Foreground alpha displayed in the channel
	AphB	Background alpha displayed in the channel
	lay	Layer displayed in the channel
	bAQp	Whether the value is an absolute QP value 0: relative QP value 1: absolute QP value
	QP	QP parameters for the encoding module
	bInv	Whether the inverse color function is enabled
	InvW	Width of the basic unit for color inversion

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	InvH	Height of the basic unit for color inversion
	Luma	Luminance threshold during color inversion
	ChnM	Color inversion mode 0: trigger color inversion when the video background luminance is less than the luminance threshold. 1: trigger color inversion when the video background luminance is greater than the luminance threshold.
REGION CHN STATUS OF COVER	Hdl	Cover handle ID
	type	Cover type. Its value is 1.
	Used	Number of times that resources are being used
REGION CHN STATUS OF COVER	Hdl	Cover handle ID
	type	Cover type. Its value is 1.
	mod	Module ID. The VIU ID is 16.
	dev	Device ID
	chn	Channel ID
	bShow	Whether to show the VICover region in the channel 0: hide 1: show
	X	Horizontal coordinate of the start position of the Cover region in the channel
	Y	Vertical coordinate of the start position of the Cover region in the channel
	W	Cover width (in pixel)
	H	Cover height (in pixel)
	Field	Attribute indicating that a region is overlaid with a frame or field frame: A region is overlaid with a frame. top: A region is overlaid with a top field. botm A region is overlaid with a bottom field.
	Color	Cover color
	lay	Layer displayed in the channel
REGION CHN STATUS OF COVEREX	Hdl	CoverEx handle ID
	type	CoverEx type. Its value is 2.
	Used	Number of times that resources are being used

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
REGION CHN STATUS OF COVEREX	Hdl	VICoverEx handle ID
	type	VICoverEx type. Its value is 2.
	mod	Module ID. The VIU ID is 16.
	dev	Device ID
	chn	Channel ID
	bShow	Whether to show the VICoverEx region in the channel 0: hide 1: show
	X	Horizontal coordinate of the start position of the CoverEx region in the channel
	Y	Vertical coordinate of the start position of the CoverEx region in the channel
	W	CoverEx width (in pixel)
	H	CoverEx height (in pixel)
	Field	Attribute indicating that a region is overlaid with a frame or field frame: A region is overlaid with a frame. top: A region is overlaid with a top field. botm A region is overlaid with a bottom field.
	Color	CoverEx color
	lay	Layer displayed in the channel
REGION STATUS OF OVERLAYEX	Hdl	OverlayEx handle ID
	type	OverlayEx type. Its value is 0.
	Used	Number of times that resources are being used
	PiFmt	OverlayEx pixel format. For details, see PIXEL_FORMAT_E.
	W	OverlayEx width (in pixel)
	H	OverlayEx height (in pixel)
	BgColor	OverlayEx background color
	Phy	Physical address for the memory occupied by the OverlayEx region
	Virt	Virtual address for the memory occupied by the OverlayEx region
	Stride	OverlayEx memory stride (in byte)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
REGION CHN STATUS OF OVERLAYEX	Hdl	OverlayEx handle ID
	type	OverlayEx type. Its value is 0.
	mod	Module ID. The VIU ID is 16.
	dev	Device ID
	chn	Channel ID
	Field	Attribute indicating that a region is overlaid with a frame or field frame: A region is overlaid with a frame. top: A region is overlaid with a top field. botm A region is overlaid with a bottom field.
	bSh	Whether to show the VIOverlay region in the channel 0: hide 1: show
	X	Horizontal coordinate of the start position of the VIOverlay region in the channel
	Y	Vertical coordinate of the start position of the VIOverlay region in the channel
	AphF	Foreground alpha displayed in the channel
	AphB	Background alpha displayed in the channel
	lay	Layer displayed in the channel

13.12 VENC

[Debugging Information]

```
# cat /proc/umap/venc
[VENC] Version: [Hi3531_MPP_V1.0.0.0 Debug], Build Time[Sep  6 2011,
09:51:16]
-----MODULE PARAM-----------------------------------------------------
   BuffCache
      0
-----VENC CHN ATTR----------------------------------------------------
NO.   Width  Height  Type Field VIField StreamType  ByFrame  BlockFlag
Sequence
0     720    576     96    0      0        1       1        1         2ef
Registered  LeftBytes  LeftFrm  CurPacks
1              0       0        0
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
-----VENC STREAM STATE---------------------------------------------
NO.   FreeCnt   BusyCnt   UserCnt   UserGet   UserRls  GetTimes  Interval
FrameRate
0        4        0         0       875        875      850      15        25
```

[Analysis]

This section records the attributes and status of the current VENC channel.

[Parameter Description]

Parameter		Description
MODULE PARAM	BuffCache	Mapping mode of the physical address for the stream buffer 0: non-cache mapping 1: cache mapping
VENC CHN ATTR	NO	Channel ID
	Width	Width of an encoding channel
	Height	Height of an encoding channel
	Type	Encoding channel type
	Field	Encoding by frame or by field 0: encoding by frame 1: encoding by field
	VIField	Frame/field flag of an input picture 0: frame mode 1: field mode
	StreamType	Large/Small stream flag 0: small stream 1: large stream
	ByFrame	Mode of obtaining streams 0: by packet 1: by frame
	BlockFlag	Flag of obtaining streams in block or non-block mode 0: block mode 1: non-block mode
	Sequence	Sequence number When the streams are obtained by frame, it represents the frame sequence number. When the streams are obtained by packet, it represents the packet sequence number.
	Registered	Whether the current channel is registered in a channel group

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		Value: {0, 1}
	LeftPics	Number of pictures to be encoded
	LeftBytes	Remaining bytes in a stream buffer
	LeftFrm	Number of remaining stream frames in a stream buffer
	CurPacks	Number of stream packets in the current frame
VENC CHN STATE	NO	Channel ID
	FreeCnt	Number of free nodes in a stream buffer
	BusyCnt	Number of busy nodes in a stream buffer
	UserCnt	Number of user nodes in a stream buffer This number increases by 1 each time users obtain a frame successfully. This number decreases by 1 each time users release a frame successfully.
	UserGet	Number of stream packets from which streams are obtained successfully
	UserRls	Number of stream packets whose streams are released successfully
	GetTimes	Number of times that streams are obtained
	Interval	Interval between two frames that are obtained from the encoder (in μs)
	FrameRate	Number of frames obtained from the encoder in one second, that is, frame rate of the encoder

13.13 VDEC

[Debugging Information]

```
[VDEC] Version: [Hi3531_MPP_V1.0.0.0 Debug], Build Time[Jan 7 2013, 11:30:57]

-----MODULE
PARAM-----------------------------------------------------------------
obey_minCR     MiniBufMode
0              0
-----CHN ATTR &
PARAMS----------------------------------------------------------
  ID  TYPE   Prior    MaxW    MaxH   Width  Height   StrmInputMode   STATE
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
PREVMODE
   0  H264       5     720     576     640     480   FRAME/BLOCK     START
0
  ID  RefNum  SupportB  DispNum  BufSize  SCDBufSize  MaxSlice  MaxSPS
MaxPPS
   0    4        No       1     1658880    622080     100      15      12
  ID  ErrThr   StrmThr   DecMode   OutMode   DnrDisp     DnrTf
   0       0        0        IP       Dec       N/A       N/A

-----CHN
STATE-----------------------------------------------------------------
  ID  PrtclErr  StrmUnSP StrmError RefNumErr  PicSizeErr  fmterror
PicBufSizeErr   RlsFail  Notify
   0      120       20494       583          0          0           0        0
0        31393  ID  fps  TimerCnt   BufFLen   DataLen   UsrFLen    UsrLen
ptsBufF   ptsBufU
   0    0     2868    1856885        0     5088        0       40        0

----- Detail Stream STATE
------------------------------------------------------------
  ID MpiSndNum   MpiSndLen  VdecNum  VdecLen  FmGetNum   FmGetLen  FmRlsNum
FmRlsLen   FmLstGet  FmRlsFail
   0    9852      133272073   9852    147780   9923      133419853   9922
133410498
9355        0

----- Detail FrameStore STATE
-------------------------------------------------
  ID  FmNewPic GetFromFm  RlsToFm Discard UsrSnd KerSnd KerRls FreeNode
BusyNode UserNode MeetEnd FrmInVdec
   0       0         0          0       0       0       0       0       40
0
 0        9852        0

----- Detail UserData STATE
-----------------------------------------------------
ID    MpiGet      MpiGetLen    MpiRls      MpiRlsLen   Discard     DiscardLen
GetFromFm   GetFromFmLen     UsrFLen     UsrLen
0        0            0         0            0          0            0
0        0            5088      0
```

[Analysis]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

This section records the status and attributes of the current decoding channel. A maximum of 64 decoding channels are provided. This information is used to check the attributes and statistics of the current decoding channel.

[Parameter Description]

Parameter		Description
MODULE PARAM	obey_minCR	Whether H.264 streams comply with the minCR conventions in the protocol 0: do not comply(default value) 1: comply. In this case, the value of minCR is 2. When a channel is being created, the maximum frame size is determined based on the value of obey_minCR. 0: The maximum frame size is u32BufSize/2. 1: The maximum frame size is u32PicWidth x u32PicHeight x 1.5/2. If there are jumbo frames in streams (the frame size is greater than u32PicWidth x u32PicHeight x 1.5/2), set obey_minCR to 0.
	MiniBufMode	Configuration mode of the stream buffer 0: common mode 1: memory reduction mode (valid only when streams are decoded by frame)
CHN ATTR PARAMS	ID	Channel ID
	Type	Decoding channel type PT_H264 PT_MJPEG PT_JPEG
	Prior	Decoding channel priority
	MaxW	Maximum width of a decoded picture
	MaxH	Maximum height of a decoded picture
	Width	Decoded picture width
	Height	Decoded picture height
	StrmInputMode	Mode of sending streams for a decoding channel. The modes are classified into two types: FRAME (send by frame) and STREAM (send by stream) BLOCK (block mode) and NOBLOCK (non-block mode)
	STATE	Whether the encoding channel starts to receive streams START: The channel starts to receive streams. STOP: The channel stops receiving streams.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	PREVMODE	Whether the mode is the decoding preview mode 0: playback mode 1: preview mode
	RefNum	Maximum number of reference frames. This parameter is valid for all decoding channels except the JPEG decoding channel.
	SupportB	Whether to support decoding of the B frame
	DispNum	Number of frames occupied by the display end
	BufSize	Size of the internal stream buffer of the VDEC module (in byte)
	SCDBufSize	Size of the stream buffer for storing the streams split by the start code detect (SCD) module. The buffer is in the VFMW and its unit is byte.
	MaxSlice	Maximum number of slices in each frame. This parameter is valid only for the H.264 protocol.
	MaxSPS	Maximum number of sequence parameter sets (SPSs) in each frame. This parameter is valid only for the H.264 protocol.
	MaxPPS	Maximum number of picture parameter sets (PPSs) in each frame. This parameter is valid only for the H.264 protocol.
	ErrThr	Threshold of stream errors
	StrmThr	Threshold for discarding frames before decoding
	DecMode	Decoding mode
	OutMode	Output sequence of decoded pictures
	DnrDisp	Reserved
	DnrTf	Reserved
CHN STATE	ID	Channel ID
	PrtclErr	Number of times that the decoder reports protocol errors
	StrmUnSP	Number of times that the decoder reports the unsupported specifications
	StrmError	Number of times that the decoder reports stream syntax errors
	RefNumErr	Number of times that the decoder reports the message indicating that the number of stream reference frames exceeds the configured value
	PicSizeErr	Number of times that the decoder reports the message indicating that the picture size exceeds the decoding

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		channel size
	fmterror	Unsupported format
	PicBufSizeErr	The frame storage buffer size is insufficient during decoding of JPEG pictures in YUV422 or YUV444 format
	RlsFail	Picture release error
	fmterror	Unsupported format
	Notify	Number of times that the internal events of the decoder are reported
	ID	Channel ID
	fps	Actual decoding frame rate
	TimerCnt	Number of times that the timer in the decoder starts
	BufFLen	Available space of the VDEC stream buffer (in byte)
	DataLen	Data length of the VDEC stream buffer (in byte)
	UsrFLen	Available space of the VDEC user data buffer (in byte)
	UsrLen	Data length of the VDEC user data buffer (in byte)
	ptsBufF	Number of available VDEC queues when streams are sent by frame
	ptsBufU	Number of used VDEC queues when streams are sent by frame
Detail Stream STATE	ID	Channel ID
	MpiSndNum	Number of times that streams are sent
	MpiSndLen	Total length of sent streams (in byte)
	VdecNum	Number of times that an end code is added in the VDEC module. This parameter is valid only when streams are sent by frame.
	VdecLen	Total length of increased streams in the VDEC module (in byte)
	FmGetNum	Number of times that the VFMW obtains streams
	FmGetLen	Total length of streams obtained by the VFMW (in byte)
	FmRlsNum	Number of times that the VFMW releases streams
	FmRlsLen	Total length of streams released by the VFMW (in byte)
	FmLstGet	Length of the last stream obtained by the VFMW (in byte)
	FmRlsFail	Number of times that the VFMW fails to release streams

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
Detail FrameStore STATE	ID	Channel ID
	FmNewPic	Number of frames decoded by the VFMW
	GetFromFm	Number of frames obtained from the VFMW
	RlsToFm	Number of frames released to the VFMW
	Discard	Number of frames discarded by the VDEC module
	UsrSnd	Number of frames sent by the VDEC module in user state
	KerSnd	Number of frames that are sent to other modules by the VDEC module in kernel state
	KerRls	Number of frames that are released to the VDEC module by users or other modules
	FreeNode	Number of free frame nodes
	BusyNode	Number of frame nodes that are not sent
	UserNode	Number of sent frame nodes
	MeetEnd	Number of times that the VFMW parses an end code
	FrmInVdec	Number of frames buffered in the decoder. The frames include the frames that are not decoded and the frames that are decoded but not sent.
Detail UserData STATE	ID	Channel ID
	MpiGet	Number of times that users obtain user data
	MpiGetLen	Length of data obtained by users
	MpiRls	Number of times that users release user data
	MpiRlsLen	Length of data released by users
	Discard	Number of times that the VDEC module discards user data
	DiscardLen	Length of user data discarded by the VDEC module
	GetFromFm	Number of times that the VDEC module obtains user data
	GetFromFmLen	Length of user data obtained by the VDEC module
	UsrFLen	Available size of the user data buffer
	UsrLen	Used size of the user data buffer

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

13.14 VI

[Debugging Information]

```
# cat /proc/umap/vi

[VIU] Version: [Hi3531_MPP_V1.0.0.0 Debug], Build Time: [Apr 12 2012,
14:54:51]

-----MODULE
PARAM-----------------------------------------------------------------
detect_err_frame drop_err_frame stop_int_level  max_cas_gap
          0          0                 0         28000

-----VI DEV
ATTR------------------------------------------------------------------
 Dev   IntfM  WkM  ComMsk0  ComMsk1 ScanM AD0 AD1 AD2 AD3   Seq  DPath DType
DRev
   0   BT656 4Mux ff000000       0    I   -1   -1  -1  -1   UYVY ByPass   YUV
N

-----VI HIGH DEV
ATTR------------------------------------------------------------------
 Dev  InputM  WkM  ComMsk0  ComMsk1 ScanM AD0 AD1 AD2 AD3   Seq CombM CompM ClkM
Fix FldP DPath DType DRev

-----VI PHYCHN
ATTR------------------------------------------------------------------
 PhyChn CapX CapY CapW CapH DstW DstH CapSel Mirror  Flip IntEn PixFom SrcRat
DstRat
     0    0    0    720   576 720   576   both     N      N     Y    sp422
25     6

-----VI PHYCHN MINOR
ATTR------------------------------------------------------------------
 PhyChn CapX CapY  CapW  CapH  DstW  DstH CapSel Mirror   Flip PixFom MixCap
DwScal
     0    0    0    360    288   360   288   both     N      N     sp422
Y     N

-----VI PHYCHN STATUS
1-----------------------------------------------------------------
 PhyChn  Dev  IntCnt  VbFail  LosInt  TopLos  BotLos BufCnt  IntT  SendT
Field  Stride
     0    0    277      0       1       0       1       2      105    25
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
intl    368

-----VI PHYCHN STATUS
2-----------------------------------------------------
 PhyChn  MaxIntT IntGapT MaxGapT  OverCnt LIntCnt  ThrCnt AutoDis CasAutD
TmgErr        0          208    20069   20074        1      1        25
0        0      0
-----VI PHYCHN STATUS
2-----------------------------------------------------
ccErrN    IntRat
2         24
-----VI PHYCHN VBI
ATTR----------------------------------------------------------
 PhyChn   VbiId      X      Y   Local    Len   CasErrN

-----VI CASCADE CHN
ATTR----------------------------------------------------------
 CasChn    Dev  PhyChn  SrcRat  DstRat

-----USER PIC
INFO----------------------------------------------------------------
UpicID  Width  Height  Stride  Field  PixForm  PoolID  PhyAddr  bUpdate

-----VI CHN
STATUS--------------------------------------------------------------
ViChn   bEnUsrP   FrmTime   FrmRate   SendCnt    SwLost   Rotate     422to420
0        N         39990       25      275        0        NONE      N

-----VI CHN CALL DSU STATUS
1-------------------------------------------------
ViChn UsrBgnNOk  UsrCancel  UsrEndOk UsrCbOk  CvrBgnNOk  CvrCancel  CvrEndOk
CvrCbOk

-----VI CHN CALL DSU STATUS
2-------------------------------------------------
ViChn  OsdBgnNOk  OsdCancel OsdEndOk  OsdCbOk  ScaleNOk  SclCancel  SclEndOk
SclCbOk

-----VI CHN CALL DSU STATUS
3-------------------------------------------------
ViChn  RotateNOk  RotCancel  RotEndOk  RotCbOk  LDCNOk  LDCCancel  LDCEndOk
LDCCbOk
```

[Analysis]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

This section records the attributes and status of the current VI device and channel.

[Parameter Description]

Parameter		Description
MODULE PARAM	detect_err_frame	Policy of discarding detected error frames in real time when the signal is unstable > 0: When the number of consecutive error frames detected is greater than the parameter value, the system considers that an incorrect timing is configured, and the subsequent frames are not discarded. 0: default value. The detected error frames are discarded in real time. < 0: Error frame detection is disabled.
	drop_err_frame	When the system detects that the current frame is an error frame, the system considers the subsequent frames as error frames and discards them. 0: default value. The function of discarding consecutive frames is disabled and only the current error frame is discarded. > 0: Consecutive **drop_err_frame** frames (including the current frame) are discarded when the system detects a picture error no matter whether the subsequent frames are correct.
	stop_int_level	When the interval between two interrupts is less than the parameter value, interrupt response is automatically disabled, ensuring that applications are responded. The default value is 0, and the unit is μs.
	max_cas_gap	In VI-VO cascade 1080p60 mode, VI interrupts are delayed when the service volume is large. In this case, frames need to be discarded; otherwise, crosstalk occurs. Therefore, when the interval between two interrupts is greater than the parameter value, the frame is discarded. The default value is 28000, and the unit is μs.
VI DEV ATTR (for details, see VI_DEV_ATTR_S.)	Dev	Device ID
	IntfM	Input mode
	WkM	Operating mode
	ComMsk0	Component 0 mask
	ComMsk1	Component 1 mask
	ScanM	Scan mode (interlaced input or progressive input) Value: {I, P}

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	AD0	AD ID
	AD1	AD ID
	AD2	AD ID
	AD3	AD ID
	Seq	Data sequence Value: {VUVU, UVUV, UYVY, VYUY, YUYV, YVYU}
	DPath	ISP enable. The default value is bypass indicating disabled (the DVR does not support the ISP).
	DType	Input data type. The default value is YUV.
	DRev	Whether the ADC or sensor reversely connects to the VI interface. The default value is N indicating no reverse.
VI HIGH DEV ATTR (for details, see VI_DEV_ATTR_S.)	Dev	Device ID
	InputM	Input mode
	WkM	Operating mode
	ComMsk0	Component 0 mask
	ComMsk1	Component 1 mask
	ScanM	Scan mode (interlaced input or progressive input) Value: {I, P}
	AD0	AD ID
	AD1	AD ID
	AD2	AD ID
	AD3	AD ID
	Seq	Data sequence Value: {VUVU, UVUV, UYVY, VYUY, YUYV, YVYU}
	CombM	Component composite mode or separation mode Value: {COMP, SEPAR}
	CompM	Single-component or dual-component mode Value: {SING, DOUB}
	ClkM	Clock mode Value: {UP, DOWN}
	Fix	Configuration of the most significant bit (MSB) of the timing reference code

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		Value: {0, 1}
	FldP	Field indicator mode of the timing reference code Value: {STD, NSTD}
	DPath	ISP enable. The default value is bypass indicating disabled (the DVR does not support the ISP).
	DType	Input data type. The default value is YUV.
	DRev	Whether the ADC or sensor reversely connects to the VI interface. The default value is N indicating no reverse.
VI PHYCHN ATTR (for details, see VI_CHN_ATTR_S.)	PhyChn	ID of a physical VI channel
	CapX	Horizontal coordinate of the start position of the capture area
	CapY	Vertical coordinate of the start position of the capture area
	CapW	Width of the capture region
	CapH	Height of the capture region
	DstW	Target picture width (obtained by scaling the captured picture)
	DstH	Target picture height (obtained by scaling the captured picture)
	CapSel	Frame/field select Value: {top, bottom, both}
	Mirror	Whether to enable the mirror function Value: {Y, N}
	Filp	Whether to enable the flip function Value: {Y, N}
	IntEn	Whether to respond to interrupts Value: {Y, N}
	PixFom	Pixel format Value: {sp420, sp422}
	SrcRat	Source frame rate for controlling the capture frame rate of a physical VI channel The value is set based on the value of VI_CHN_ATTR_S.
	DstRa	Target frame rate for controlling the capture frame rate of a physical VI channel The value is set based on the value of

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		VI_CHN_ATTR_S.
VI PHYCHN MINOR ATTR (Compared with the primary attribute, the secondary attribute includes MixCap and DwScal but does not include SrcRat, IntEn, and DstRa.)	MixCap	Whether to select the mixing capture mode Value: {Y, N}
	DwScal	Whether the current output picture is scaled in the VIU Value: {Y, N}
VI PHYCHN STATUS 1	Dev	ID of a VI device bound to a physical VI channel
	PhyChn	ID of a physical VI channel
	IntCnt	Interrupt count The value increases by 1 each time the frame is interrupted. If a VI channel does not properly work, the value sometimes does not increase.
	VBFail	Number of times that VBs fail to be obtained Before capturing a frame, the VI channel obtains the VB. If the VB fails to be obtained, the value increases by 1. Typically, the value should be 0. Otherwise, the configuration of the public VB is incorrect.
	LosInt	Number of times that interrupts are lost The VI channel needs to prepare for capturing pictures at the intervals of the initial one or two interrupts. It is normal that the value of this item is 1 or 2.
	TopLos	Number of lost top field interrupts
	BotLos	Number of lost bottom field interrupts
	BufCnt	Number of picture VBs occupied by the VI channel
	IntT	Time of processing interrupts (in μs)
	SendT	Time of transmitting pictures during interrupt processing (in μs)
	Field	Frame/field flag of a picture frm: progressive picture intl: interlaced picture
	Stride	Picture stride
VI PHYCHN	PhyChn	ID of a physical VI channel

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
STATUS 2	MaxIntT	Maximum time used for processing an interrupt
	IntGapT	Interval between adjacent interrupts
	MaxGapT	Maximum interval between adjacent interrupts
	OverCnt	Number of times that the interval between two adjacent interrupts exceeds 40 ms
	LIntCnt	Number of times that the time used for processing an interrupt exceeds 10 ms
	AutoDis	Number of times that interrupt mask is automatically disabled due to too many interrupts
	CasAutD	Number of frames discarded due to interrupt delay in VI-VO cascade mode
	TmgErr	Number of interrupts generated due to exceptions such as the buffer overflow
	ThrCnt	Number of times that frames or fields are lost. This parameter value is 1 in most cases when the program just starts. It is normal if this parameter value does not increment subsequently.
	ccErrN	Number of interrupts indicating that data capture is not complete. If data is captured at a non-full frame rate or VBFail increases, ccErrN also increases. If data is captured at a full frame rate and VBFail does not increase but ccErrN increases, check the input timing or input line connection.
	IntRat	Capture frame rate determined by VI interrupts
VI PHYCHN VBI ATTR	PhyChn	ID of a physical VI channel
	VbiId	VBI region ID
	X	Horizontal coordinate of the VBI start position
	Y	Vertical coordinate of the VBI start position
	Local	VBI position
	Len	VBI data length. It is in the unit of word (four bytes).
	CasErrN	Number of incorrect picture IDs detected in VI-VO cascade mode
VI CASCADE CHN ATTR	CasChn	ID of a VI cascade channel
	Dev	ID of the VI device whose cascade function is enabled
	PhyChn	ID of the physical VI channel whose cascade function is enabled

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	SrcRat	Source frame rate for controlling the capture frame rate of a VI cascade channel
	DstRat	Target frame rate for controlling the capture frame rate of a VI cascade channel
USER PIC INFO	UPicID	ID of the user picture used by a channel
	Width	Width of the channel that uses a user picture
	Height	Height of the channel that uses a user picture
	Stride	Stride of the channel that uses a user picture
	Field	Frame/field select The value is fixed at frm.
	PixForm	Pixel format Value: {sp420, sp422}
	PoolID	ID of the pool corresponding to a user picture
	PhyAddr	Physical address for the memory for the pool corresponding to a user picture
	bUpdate	Whether to update the user picture
VI CHN STATUS	ViChn	ID of a VI channel
	bEnUsrP	User picture insertion enable
	FrmTime	Interval of video frames (in μs)
	FrmRate	Capture frame rate of a VI channel
	SendCnt	Number of times that a VI channel transmits captured pictures
	SwLost	Number of times that a VI channel discards frames when transmitting pictures
	Rotate	Rotation angle. It is 90º, 180º, or 270º.
	422to420	Whether to enable the function of converting a YUV422 picture into a YUV420 picture Value range: {Y, N}
VI CHN CALL DSU STATUS Information when the VI channel calls the DSU (Usr corresponds to user pictures, Cvr corresponds to COVEREX_RGN,	ViChn	ID of a VI channel
	UsrBgnNOk	Number of times that the DSU begin job fails to be called
	UsrCancel	Number of times that cancel jobs are called
	UsrEndOk	Number of times DSU jobs are successfully submitted
	UsrCbOk	Number of times that the DSU is successfully called

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
and Osd corresponds to OVERLAYEX_RGN.)		back
	CvrBgnNOk	Number of times that the DSU begin job fails to be called
	CvrCancel	Number of times that the cancel job is called
	CvrEndOk	Number of times that DSU jobs are successfully submitted
	CvrCbOk	Number of times that the DSU is successfully called back
VI CHN CALL DSU STATUS 2 Osd corresponds to OVERLAYEX_RGN. Scl indicates scaling.	ViChn	ID of a VI channel
	OsdBgnNOk	Number of times that the DSU begin job fails to be called
	OsdCancel	Number of times that the cancel job is called
	OsdEndOk	Number of times that DSU jobs are successfully submitted
	OsdCbOk	Number of times that the DSU is successfully called back
	ScaleNOk	Number of times that the DSU begin job fails to be called
	SclCancel	Number of times that the cancel job is called
	SclEndOk	Number of times that DSU jobs are successfully submitted
	SclCbOk	Number of times that the DSU is successfully called back
VI CHN CALL DSU STATUS 3 The VI channel calls the DSU information. Rot indicates rotation, and LDC indicates lens distortion correction.	ViChn	ID of a VI channel
	RotateNOk	Number of times that the DSU begin job fails to be called
	RotCancel	Number of times that the cancel job is called
	RotEndOk	Number of times that DSU jobs are successfully submitted
	RotCbOk	Number of times that the DSU is successfully called back
	LDCNOk	Number of times that the DSU begin job fails to be called
	LDCCancel	Number of times that the cancel job is called
	LDCEndOk	Number of times that DSU jobs are successfully submitted
	LDCCbOk	Number of times that the DSU is successfully called

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		back

13.15 VO

[Debugging Information]

```
# cat /proc/umap/vo
[VOU] Version: [Hi3531_MPP_V1.0.0.0 Debug], Build Time[Sep 13 2011, 19:44:04]
-----MODULE PARAM-----------------------------------------------------

-----DEV CONFIG-------------------------------------------------------------
 DevId  DevEn   Mux1  Mux2  Mux3     InfSync   BkClr  DevFrt DispFrt DoubFrm
Toleration  Vtth
 0       y  BT1120    720P@50     ff     50     25      n   10000000   240

-----DEV VDAC STATUS----------------------------------------
 DevId    VDAC
0      -

-----DEV VIDEO STATUS---------------------------------------------
 DevId VideoEn     PiP  PixFmt    ImgW    ImgH   DispW   DispH
    0      y       n      420    1280     720    1280     720

-----DEV VIDEO STATUS 2-------------------------------------------
 DevId VideoEn     PiP EnChNum   Matrix    Luma    Cont     Hue    Satu
    0      y       n      1           0     50      50       50     50

-----DEV VIDEO STATUS 3-----------------------------------------------
DevId  SetBeg  SetEnd   WBCEn bCasCfg  bCasEn bCasSlv  CasRgn CasMode CasPatn
0       n       n       n       n       n       n                       0

-----GRP STATUS-------------------------------------------------------------
 GrpId OnDev ChNum FulFr TgtFr Start Resrv   GrpGap            BasePts

-----CHN BASE INFO----------------------------------------------------
DevId ChnId  ChnEn Prio DeFlk  ChnX  ChnY  ChnW  ChnH WinX  WinY  WinW  WinH
DispX DispY bSnap Field DoubFrm bCas CcPos  bRcvMatch
0     0      y     1      n      0     0     640    360   0     0     640    320
-1    -1   n     both   n  n     0

-----CHN PLAY INFO----------------------------------------------------
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
DevId ChnId Batch  Show Pause  Step Revrs Refsh Thrshd ChnFrt ChnGap
0     0      n     y     n      n     n     n       3     25     40000

-----CHN PLAY INFO----------------------------------------------------
DevId ChnId   DisplayPts       PrePts      CurrPts     ScalePts     SetPts
0     0    49320948834   49320948834   49321028288            0          0

-----HD CHN STATUS1---------------------------------------------------
DevId ChnId  QCnt NewDo OldDo  LCnt  SCnt ChRpt BusyN ShouD Dsped  Area AreaT
0     0     17    12     0      0     11     7     0     1      1      0     0

-----HD CHN STATUS2---------------------------------------------------
 DevId ChnId  Buf1  Buf2  Buf3  DBuf Stat1 Stat2 QNodeAddr SNodeAddr  DispAddr
     0     0    UseF  UseF    F     2    end    end   97291000   973e2800
973e2800

-----SD CHN STATUS----------------------------------------------------
DevId ChnId   Job  Task  LCnt  SCnt  ChRpt  DRpt CBusy DBusy ShouD Dsped b2Scl
ChnAddr DispAddr
-----CHN OTHER INFO---------------------------------------------------
DevId ChnId bZoom ZmTyp ZoomX ZoomY ZoomW ZoomH  SrcW  SrcH FltTp  HFlt VlFlt
VcFlt
0     0     n      0      0      0      0      0      0     0     0      0     0    0

-----Wbc INFO---------------------------------------------------------
 DevId  WbcW  WbcH PixFmt FrmRat RealRat  Mode  DataSrc  Depth  NotFin
-----GRAPHICS LAYER--------------------------------------------
Layer BindDev
   HC0       0
-----LAYER CSC PARAM-------------------------------------------------
LAYERID  Matrix    Luma    Cont     Hue    Satu
     0        5      50      50       50     50
-----VGA PARAM-------------------------------------------------------
DevId    Matrix  Luma    Cont     Hue    Satu    Gain
```

[Analysis]

This section records the usage and attributes of the VOU, including the status of the device, video layer, synchronization group, and channel. By checking the debugging information, you can obtain the status of the VOU for easy debugging or testing.

[Parameter Description]

Parameter		Description
DEV CONFIG	DevId	Device ID Value range: [0, VO_MAX_DEV_NUM)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	DevEn	Whether to enable the VOU n: disabled y: enabled
	Mux1/Mux2/ Mux3	Interface type Value: VO_INTF_CVBS, VO_INTF_YPBPR, VO_INTF_VGA, VO_INTF_BT656, VO_INTF_BT1120, or VO_INTF_HDMI
	InfSync	Interface timing Value range: [0, VO_OUTPUT_BUTT)
	BkClr	Background color of a device, in RGB888 format
	DevFrt	Frame rate of a device, that is, refresh rate. The frame rate is related to the timing.
	DispFrt	Target display frame rate, that is, the frame rate of combining pictures Value range: (0, DevFrt]
	DoubFrm	Whether to double the frame rate for the device n: no y: yes
	Toleration	Device tolerance, which is related to frame rate control
	Vtth	Vertical timing interrupt threshold of a device
DEV VDAC STATUS	DevId	Device ID Value range: [0, VO_MAX_DEV_NUM)
	VDAC	Device load detection enable Y: enabled N: disabled The chip does not support load detection.
DEV VIDEO STATUS	DevId	Device ID Value range: [0, VO_MAX_DEV_NUM)
	VideoEn	Video layer enable n: disabled y: enabled
	PiP	Picture-in-picture (PIP) layer enable n: disabled y: enabled
	PixFmt	Pixel format of an input picture Only PIXEL_FORMAT_YUV_SEMIPLANAR_422 and PIXEL_FORMAT_YUV_SEMIPLANAR_420 are

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		supported.
	ImgW	Width of a video layer canvas
	ImgH	Height of a video layer canvas
	DispW	Width of a displayed area
	DispH	Height of a displayed area
DEV VIDEO STATUS 2	DevId	Device ID Value range: [0, VO_MAX_DEV_NUM)
	VideoEn	Video layer enable n: disabled y: enabled
	PiP	PIP layer enable n: disabled y: enabled
	EnChNum	Number of times that channels are enabled, that is, the number of enabled channels of a device Value range: [0, VO_MAX_CHN_NUM)
	Matrix	Color space conversion (CSC) matrix select Value range: [0, 2]
	Luma	Luminance The value ranges from 0 to 100 and the default value is 50.
	Cont	Contrast The value ranges from 0 to 100 and the default value is 50.
	Hue	Hue The value ranges from 0 to 100 and the default value is 50.
	Satu	Saturation The value ranges from 0 to 100 and the default value is 50.
DEV VIDEO STATUS 3	DevId	Device ID Value range: [0, VO_MAX_DEV_NUM)
	SetBeg	Whether SetAttrBegin of a device is set n: no y: yes
	SetEnd	Whether SetAttrEnd of a device is set n: no y: yes
	WBCEn	Whether the video writeback (WBC) function is enabled for

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		a device n: disabled y: enabled
	bCasCfg	Whether the cascade attribute of a device is set n: no y: yes
	bCasEn	Whether to enable the cascade function of a device n: disabled y: enabled
	bCasSlv	Device cascade mode n: master mode y: slave mode
	CasRgn	Cascade region mode of a device 32Rgn: 32-region mode 64Rgn: 64-region mode
	CasMode	Device cascade channel mode Single: single-channel mode Dual: dual-channel mode
	CasPatn	Device cascade pattern Value range: [0, 128)
GRP STATUS	–	Not supported
CHN BASE INFO	DevId	Device ID Value range: [0, VO_MAX_DEV_NUM)
	ChnId	Channel ID Value range: [0, VO_MAX_CHN_NUM)
	ChnEn	Channel enable Y: enabled N: disabled
	Prio	Channel priority Value range: [0, VO_MAX_CHN_NUM)
	DeFlk	Whether to perform anti-flicker. On the interlaced device, when a single-field picture or a frame is zoomed out, the anti-flicker function needs to be enabled. In other cases, anti-flicker function is disabled. On the progressive output device, the anti-flicker function is disabled. n: disabled y: enabled

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	ChnX	Start horizontal coordinate of a channel
	ChnY	Start vertical coordinate of a channel
	ChnW	Channel width
	ChnH	Channel height
	WinX	Start horizontal coordinate of the channel display window (relative coordinate. The start point in the channel attributes serves as the coordinate origin.)
	WinY	Start vertical coordinate of the channel display window (relative coordinate. The start point in the channel attributes serves as the coordinate origin.)
	WinW	Width of the channel display window
	WinH	Height of the channel display window
	DispX	Horizontal coordinate of the channel display position on the PIP layer of the HD device. This parameter is invalid for the channel on the PIP layer of the non-HD device.
	DispY	Vertical coordinate of the channel display position on the PIP layer of the HD device. This parameter is invalid for the channel on the PIP layer of the non-HD device.
	bSnap	Whether to enable channel snapshot n: no y: yes
	Field	Frame/field information. This item is used to set the frame/field flag of a picture. Value range: [0, VO_FIELD_BUTT)
	DoubFrm	Whether to enable the double-frame-rate function of the channel n: no y: yes
	bCas	Whether to cascade channels n: no y: yes
	CcPos	Cascade position. This parameter is not supported.
	bRcvMatch	Whether to enable the frame rate match function of a specified VO channel 0: no 1: yes
CHN PLAY INFO1	DevId	Device ID Value range: [0, VO_MAX_DEV_NUM)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	ChnId	Channel ID Value range: [0, VO_MAX_CHN_NUM)
	Batch	Whether the channel processes data in batches n: no y: yes
	Show	Whether to show a channel n: hide y: show
	Pause	Whether to pause a channel n: disabled y: enabled
	Step	Whether to step n: disabled y: enabled
	Revrs	Whether to enable reverse playing n: disabled y: enabled
	Refsh	Channel refresh enable n: disabled y: enabled
	Thrshd	Channel display threshold For the HD device, the threshold is the maximum number of frames to be displayed in the channel. For the SD device, the threshold is the maximum number of frames to be received in the channel buffer queue.
	ChnFrt	Frame rate of a channel. This value reflects the playing and control mode of a channel.
	ChnGap	Frame gap of channel (in µs). The gap is in inverse proportion to the frame rate of the channel (in µs).
CHN PLAY INFO2	DevId	Device ID Value range: [0, VO_MAX_DEV_NUM)
	ChnId	Channel ID Value range: [0, VO_MAX_CHN_NUM)
	DisplayPts	PTS of the frame being displayed (in µs)
	PrePts	PTS of the picture of the node at the end of the channel busy queue (in µs)
	CurrPts	PTS of the pictures that are queried by the VPSS (in µs)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	ScalePts	Target PTS, that is, PTS of the next frame (in μs)
	SetPts	Size of HD picture switching and PTS OF the last old frame (in μs)
HD CHN STATUS1	DevId	Device ID Value range: [0, VO_MAX_DEV_NUM)
	ChnId	Channel ID Value range: [0, VO_MAX_CHN_NUM)
	QCnt	Number of times the VPSS queries VO channel
	NewDo	Number of times that New+Do is returned
	OldDo	Number of times that Old+Do is returned
	LCnt	Number of times that frames are discarded
	SCnt	Number of times that the channel receives frames transmitted by other modules. If the value is 0, the VOU does not receive pictures; if the value does not increase, the VOU cannot receive pictures.
	BusyN	Number of nodes in the busy queue
	ChRpt	Number of times that the channel displays repeated pictures. If the channel is paused, this value is continuously increased.
	ShouD	Number of times that the current picture should be displayed
	Dsped	Number of times that the current picture has been displayed. If the channel is paused, this value is continuously increased.
	Area	Display area of the video layer corresponding to the channel
	AreaT	Display area of the video layer resorted after the channel size is switched
HD CHN STATUS2	DevId	Device ID Value range: [0, VO_MAX_DEV_NUM)
	ChnId	Channel ID Value range: [0, VO_MAX_CHN_NUM)
	Buf1/Buf2/Buf3	Used status of channel box 1/2/3 Value range: {UseF, F, E}. UseF indicates that the box is filled with memory and is used. F indicates that the box is filled with memory, but is not used. E indicates that the box is not filled with memory and is empty.
	DBuf	ID of the box used by the current displayed frame in the channel

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	Stat1	Status of channel size switching end: Channel size switching is completed. ing: The VPSS is processing the pictures with new resolutions after channel size switching. seted: The pictures with new resolutions are received and the channel can be switched. begin: The channel size starts to be switched.
	Stat2	Status of the channel from hidden to shown end: The status of the channel is changed from hidden to shown. ing: There are pictures to be displayed in the channel queue. begin: The show flag starts to be processed.
	QNodeAddr	Physical address for the picture that is returned to the VPSS by the VOU when the VPSS queries
	SNodeAddr	Physical address for the picture that is received by the VOU from other modules
	DispAddr	Physical address for current picture
SD CHN STATUS	DevId	Device ID Value range: [0, VO_MAX_DEV_NUM)
	ChnId	Channel ID Value range: [0, VO_MAX_CHN_NUM)
	Job	Number of tasks for the DSU. This value indicates the number of combination tasks that are being handled by the DSU.
	Task	Number of tasks in the DSU
	LCnt	Number of times that the channel discards the frames sent by other modules
	SCnt	Number of times that the channel receives frames sent by other modules. If the value is 0, the VO channel does not receive pictures; if the value does not increase, the VO channel cannot receive pictures.
	ChRpt	Number of times that the channel displays repeated pictures. If the channel is paused, this value is continuously increased.
	DRpt	Number of times that spliced pictures are displayed repeatedly
	CBusy	Number of nodes in the channel busy queue
	DBusy	Number of displayed busy nodes in the queue
	ShouD	Number of times that the current picture should be

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		displayed
	Dsped	Number of times that the current picture has been displayed. If the channel is paused, this value increases continuously.
	b2Scl	Whether the displayed picture in the channel is processed by using level-2 scaling Y: yes N: no
	ChnAddr	Physical address for the current picture in the channel
	DispAddr	Physical address for the picture that is displayed at the video layer
CHN OTHER INFO	DevId	Device ID Value range: [0, VO_MAX_DEV_NUM)
	ChnId	Channel ID Value range: [0, VO_MAX_CHN_NUM)
	bZoom	Whether to perform partial enlargement n: no y: yes
	ZmTyp	Partial enlargement type, enlargement by region or ratio Value range: [0, VOU_ZOOM_IN_BUTT)
	ZoomX	Start horizontal coordinate of a partially enlarged region. If the partially enlarged region is specified by ratio, the value is the coordinate of the source picture after processing by ratio and alignment.
	ZoomY	Start vertical coordinate of a partially enlarged region. If the partially enlarged region is specified by ratio, the value is the coordinate of the source picture after processing by ratio and alignment.
	ZoomW	Width of a partially enlarged region. If the partially enlarged region is specified by ratio, the value is the coordinate of the source picture after processing by ratio and alignment.
	ZoomH	Height of a partially enlarged region. If the partially enlarged region is specified by ratio, the value is the coordinate of the source picture after processing by ratio and alignment.
	SrcW	Width of the source channel picture
	SrcH	Height of the source channel picture
	FltTp	Type of the channel filter, that is, the type of the DSU filter Value range: [0, FILTER_PARAM_TYPE_BUTT)

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	HFlt	Horizontal filtering coefficient of a channel, that is, the horizontal DSU coefficient Value range: [0, DSU_HSCALE_FILTER_BUTT)
	VlFlt	Vertical luminance filtering coefficient of a channel, that is, the vertical luminance coefficient of the DSU Value range: [0, DSU_VSCALE_FILTER_BUTT)
	VcFlt	Vertical chrominance filtering coefficient of a channel, that is, the vertical chrominance coefficient of the DSU Value range: [0, DSU_VSCALE_FILTER_BUTT)
Wbc INFO	DevId	Device ID Note that only DHD1 supports video WBC.
	WbcW	Target picture width for video WBC. The maximum width is 720.
	WbcH	Target picture height for video WBC. The maximum width is 576.
	PixFmt	Pixel format for video WBC Only PIXEL_FORMAT_YUV_SEMIPLANAR_422 and PIXEL_FORMAT_YUV_SEMIPLANAR_420 are supported.
	FrmRat	Target frame rate for video WBC
	RealRat	Average frame rate for video WBC. This value is calculated every 10 seconds.
	Mode	Video WBC mode
	DataSrc	Video WBC source Mixer: The pictures from the video layer and graphics layer are mixed and then written back. Video: Only the pictures from the video layer are written back.
	Depth	Video WBC depth
	NotFin	Number of unfinished frames after video WBC
GRAPHICS LAYER	Layer	Graphics layer Value: G4, HC0, or HC1
	BindDev	ID of the device bound to a graphics layer
LAYER CSC PARAM	LAYERID	ID of the graphics layer bound to a device Value range: [0, VOU_GRAPHICS_LAYER_NUM)
	Matrix	VGA CSC matrix select Value range: [5, 6]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	Luma	Luminance Value range: [0, 100] Default value: 50
	Cont	Contrast Value range: [0, 100] Default value: 50
	Hue	Hue Value range: [0, 100] Default value: 50
	Satu	Saturation Value range: [0, 100] Default value: 50
VGA PARAM	DevId	Device ID Value range: [0, VO_MAX_DEV_NUM)
	Matrix	VGA CSC matrix select Value range: [3, 4]
	Luma	Luminance Value range: [0, 100] Default value: 50
	Cont	Contrast Value range: [0, 100] Default value: 50
	Hue	Hue Value range: [0, 100] Default value: 50
	Satu	Saturation Value range: [0, 100] Default value: 50
	Gain	DAC gain Value range: [0, 63] Default value: 48

13.16 VPSS

[Debugging Information]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
# cat /proc/umap/vpss

[VPSS] Version: [Hi3520D_MPP_V0.0.0.0 Debug], Build Time[Feb 18 2013,
16:46:55]

-------------------------VPSS GRP ATTR-------------------------------------
  GrpID    MaxW    MaxH  PixFmt DieMode    DrEn    DbEn    IeEn    NrEn
HistEn    Field
     0     720     576 semi422       0       0       0       1       1       0
BOTH

----------------------VPSS GRP FRAME CONTROL INFO-------------------------
  GrpID bFilter   Width  Height    SrcFRate    DstFRate
     0       0       0       0          -1          -1

-------------------------VPSS GRP PARAM-------------------------------------
  GrpID     Lum    Cont    Dark  Bright   IeStr IeSharp   SfStr   TfStr
Motion   DiStr  ChrmRg     NrW   SfWin DisMode
     0       0      64       0       0       4       0      24       8       0
0       0       0       0       0

-------------------------VPSS CHN PARAM--------------------------------
  GrpID   ChnID   ChnSp   ChnSf   ChnTf
     0       0      64       0       0
     0       1      64       0       0
     0       2      64       0       0
     0       3      64       0       0

----------------------VPSS GRP PRESCALE INFO-------------------------
  GrpID bPreScl  CapSel   Width  Height   totde
     0     OFF    BOTH       0       0       0

-------------------------VPSS CHN ATTR-------------------------------------
  GrpId  PhyChnId  Enable    SpEn FrmWkEn      LW      RW      TW      BW
     0       0       1       1       1       0       2       6      14
     0       1       1       1       1       0       2       6      14
     0       2       1       1       1       0       2       6      14
     0       3       1       1       1       0       2       6      14
     0       4       1       -       -       -       -       -       -

-------------------------VPSS CROP INFO-------------------------------------
GrpId  CropEn  CoorTp   CoorX   CoorY   Width  Height  CapSel    OriW    OriH
TrimWid TrimHgt
     0       1     ABS       0       0     640     480    BOTH     704     576
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
640     480

-------------------------VPSS GRP PIC QUEUE-------------------------
GrpID FreeLen BusyLen   Delay   Backup
0       6       0       0         1
-----------------------VPSS GRP WORK STATUS-------------------------
   GrpID        RecvPic  ViLost    VdecLost          NewDo   OldDo NewUnDo
OldUnDo  NoHist StartFl  bStart  CostTm MaxCostTm
      0           463       0          0           463       0       0
1386       0       0       1   15482    15807

-----------------------VPSS CHN WORK STATUS-------------------------
   GrpID   ChnID    WorkMode   Depth         SendOk  bConfident     bDouble
CapSel
      0       0       AUTO       0           461          1         --
BOTH
      0       1       AUTO       0           461          1         --
BOTH
      0       2       AUTO       0           461          1          0
--
      0       3       AUTO       0           461          1         --
--

----------------------VPSS CHN OUTPUT RESOLUTION-----------------------
   GrpID   ChnID  Enable   Width  Height  pixfmt
      0       0       1     640     480      19
      0       1       1     352     288      19
      0       2       1     640     360      19
      0       3       1     640     360      19


----------------------TIMER WORK STATUS-------------------------
    CntPerSec  MaxCntPerSec       CostTm    MostCostTm  CostTmPerSec
MCostTmPerSec
          75           76           75         1916        20821        43525

-------------------------DRV WORK
STATUS-----------------------------------
      StartSuc0   StartSuc1        LinkInt     NodeInt   StartErr0
NodeIdErr0   StartErr1  NodeIdErr1      BusErr
           463          0           463          0          0          0
0          0          0

-------------------------DRV NODE QUEUE-----------------------------------
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
 FreeNum WaitNum  Busy00  Busy01    Sel0  Busy10  Busy11    Sel1
    128       0       0       0       1       0       0       0

----------------------INT WORK STATUS-------------------------
    CntPerSec  MaxCntPerSec       CostTm    MostCostTm  CostTmPerSec
MCostTmPerSec
          25           26          482          552        12596        13028
```

[Analysis]

This section records the current attributes and status of the VPSS.

[Parameter Description]

Parameter		Description
VPSS GRP ATTR (including the configured maximum processing capability and enable status of each functional module.)	GrpID	GRP ID Value range: [0, 127]
	MaxW	Maximum process width Value range: [64, 4096]
	MaxH	Maximum process height Value range: [64, 4080]
	PixFmt	Pixel format. semi420 and semi422 are supported.
	DieMode	Deinterlace mode 0: adaptation 1: do not perform de-interlace (DIE) forcibly 2: perform DIE forcibly
	DrEn	Dering enable 0: disabled 1: enabled
	DbEn	Deblock enable 0: disabled 1: enabled
	IeEn	Image enhancement enable 0: disabled 1: enabled
	NrEn	Noise reduction enable 0: disabled 1: enabled
	HistEn	Histogram output 0: disabled

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		1: enabled
	Field	CRP capture field information BOTH: both fields BOTTOM: bottom field TOP: top field
VPSS GRP FRAME CONTROL INFO	GrpID	GRP ID Value range: [0, 127]
	bSizer	Enable for filtering out pictures by picture size
	Width	Picture width for filtering out pictures
	Height	Picture height for filtering out pictures
	SrcFRate	Source frame rate
	DstFRate	Target frame rate
VPSS GRP PARAM	GrpID	GRP ID Valid range: [0, 127]
	Lum	Luminance
	Cont	Contrast
	Dark	Dark region enhancement
	Bright	Bright region enhancement
	IeStr	Image enhancement (IE) strength
	IeSharp	IE sharpness
	SfStr	Strength of spatial-domain noise reduction (NR)
	TfStr	Strength of time-domain NR
	Motion	Motion judgment threshold
	DiStr	DIE strength
	ChrmRg	Chrominance amplitude
	NrW	Weight of time-domain NR
	SfWin	Size of the spatial domain filtering window
	DisMode	Display mode
VPSS CHN PARAM	GrpID	GRP ID Value range: [0, 127]
	ChnID	Channel ID

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		Value range: [0, 4]
	ChnSp	Channel sharpening (SP) strength
	ChnSf	Strength of channel spatial-domain NR
	ChnTf	Strength of channel time-domain NR
VPSS GRP PRESCALE INFO	GrpID	GROUP ID Value range: [0, 127]
	bPreScl	Prescale enable 0: disabled 1: enabled
	CapSel	Captured field select TOP: top field BOTTOM: bottom field BOTH: top and bottom fields
	Width	Width of the target picture to be prescaled
	Height	Height of the target picture to be prescaled
	totde	Number of pictures submitted by the VPSS that are being processed by the TDE
VPSS CHN ATTR	GrpId	GRP ID Value range: [0, 127]
	PhyChnID	Physical channel ID Value range: [0, 4]
	Enable	Channel enable 0: disabled 1: enabled
	SpEn	Sharpening enable 0: disabled 1: enabled
	FrmWkEn	Frame enable 0: disabled 1: enabled
	LW	Width of the left frame The value must be an even ranging from 0 to 14.
	RW	Width of the right frame The value must be an even ranging from 0 to 14.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	TW	Width of the top frame The value must be an even ranging from 0 to 14.
	BW	Width of the bottom frame The value must be an even ranging from 0 to 14.
VPSS CROP INFO	GrpId	GRP ID Value range: [0, 127]
	CropEn	Crop enable 0: disabled 1: enabled
	CoorTp	Coordinate type RIT: relative coordinate ABS: absolute coordinate
	CoorX	Horizontal start coordinate When the coordinate type is relative coordinate, the valid value range is [0, 999]. When the coordinate type is absolute coordinate, the valid value range is [0, MAX_WIDTH].
	CoorY	Vertical start coordinate When the coordinate type is relative coordinate, the valid value range is [0, 999]. When the coordinate type is absolute coordinate, the valid value range is [0, MAX_HEIGHT].
	Width	Width of clip CROP RECT. It cannot exceed the maximum picture width.
	Height	Height of clip CROP RECT. It cannot exceed the maximum picture height.
	CapSel	Captured field select TOP: top field BOTTOM: bottom field BOTH: top and bottom fields
	OriWth	Width of the original picture
	OriHgt	Height of the original picture
	TrimWid	Actual picture width

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	TrimHgt	Actual picture height
VPSS GRP PIC QUEUE	GrpID	GRP ID Value range: [0, 127]
	FreeLen	Number of nodes that can be used
	BusyLen	Number of nodes that can be used
	Delay	Delay queue length
	Backup	Whether to enable backup frames 0: disabled 1: enabled
VPSS GRP WORK STATUS	GrpID	GRP ID Value range: [0, 127]
	RecvPic	Number of pictures the VPSS receives
	ViLost	Number of discarded VI pictures due to full queue
	VdecLost	Number of discarded VDEC pictures due to full queue
	NewDo	Number of NewDo times
	OldDo	Number of OldDo times
	NewUnDo	Number of NewUnDo times
	OldUnDo	Number of OldUnDo times
	NoHist	Number of times that the histogram buffer cannot be applied
	StartFl	Number of times that the start task fails
	bStart	Whether to receive pictures
	CostTm	Current time for completing tasks
	MaxCostTm	Historical maximum time for completing tasks
VPSS CHN WORK STATUS	GrpID	GRP ID Value range: [0, 127]
	ChnID	Channel ID Value range: [0, 4]
	SendOk	Number of times that the VPSS transmits pictures successfully
	WorkMode	Channel working mode

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	Depth	User queue depth
	bConfident	Whether the backend picture processing requirement matches the channel capability
	bDouble	Whether field/frame conversion is being used in the channel
	CapSel	Captured field select TOP: top field BOTTOM: bottom field BOTH: top and bottom fields
VPSS CHN OUTPUT RESOLUTION	GrpID	GRP ID Value range: [0, 127]
	ChnID	Channel ID Value range: [0, 4]
	Enable	Channel enable 0: disabled 1: enabled
	Width	Target picture width (in pixel)
	Height	Target picture height (in pixel)
	pixfmt	Pixel format of the target picture
TIMER WORK STATUS	CntPerSec	Number of times that the timer works in the last second
	MaxCntPerSec	Historical maximum number of times that the timer works in one second
	CostTm	Time that the timer works last time
	MostCostTm	Maximum time that the timer works
	CostTmPerSec	Time that the timer works in the last second
	MCostTmPerSec	Historical maximum time that the timer works in one second
DRV WORK STATUS	StartSuc0	Number of times that VPSS0 is successfully started
	StartSuc1	Number of times that VPSS1 is successfully started
	LinkInt	Number of times that links are interrupted
	NodeInt	Number of times that nodes are interrupted
	StartErr0	Number of times that VPSS0 fails to be

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		started
	NodeIdErr0	Number of times that an error occurs when the VPSS0 node ID is read
	StartErr1	Number of times that VPSS1 fails to be started
	NodeIdErr1	Number of times that an error occurs when the VPSS1 node ID is read
	BusErr	Number of bus error interrupts
DRV NODE QUEUE	FreeNum	Number of available nodes in the free queue
	WaitNum	Number of nodes to be processed in the wait queue
	Busy00	Number of nodes in busy0 queue of VPSS0
	Busy01	Number of nodes in busy1 queue of VPSS0
	Sel0	ID of the busy queue of VPSS0 for operation
	Busy10	Number of nodes in busy0 queue of VPSS1
	Busy11	Number of nodes in busy1 queue of VPSS1
	Sel1	ID of the busy queue of VPSS1 for operation
INT WORK STATUS of the VPSS	CntPerSec	Number of times that the interrupt is performed in the last second
	MaxCntPerSec	Historical maximum number of times that the interrupt is performed in one second
	CostTm	Time that the interrupt is performed last time
	MostCostTm	Maximum time that the interrupt is performed
	CostTmPerSec	Time that the interrupt is performed in the last second
	MCostTmPerSec	Historical maximum time that the interrupt is performed in one second

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

13.17 IVE

[Debugging Information]

```
# cat /proc/umap/ive

[IVE] Version: [Hi3531_MPP_V1.0.0.0 Debug], Build Time[Sep 16 2011, 11:38:03]

-----IVE CONTEXT
ATTR------------------------------------------------------------
 W  B WCI WEI BCI BEI      HndU      TskF LstId TskId bInt HWrap FWrap    IntCnt
 1  0    0   0   0  309      4060      3751      0     197    0      0      0
29
```

[Analysis]

This section records the status of the IVE.

[Parameter Description]

Parameter		Description
IVE CONTEXT ATTR	W	Waiting queue ID (0 or 1)
	B	ID of the queue being scheduled (0, 1, or –1). The value –1 indicates that the IVE hardware is idle.
	WCI	ID of the first valid task in the waiting queue
	WEI	ID of the last valid task in the waiting queue + 1
	BCI	ID of the first valid task in the queue being scheduled
	BEI	ID of the last valid task in the queue being scheduled + 1
	HndU	ID of the handle of the current task that can be allocated
	TskF	Number of completed tasks
	LstId	ID of the previously completed task
	TskId	ID of the currently completed task
	bInt	bInstant value transferred when users submit tasks last time
	HWrap	Number of times that the user handle ID is wrapped
	FWrap	Number of times that completed tasks are wrapped
	IntCnt	Number of times that the IVE generates interrupts

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

13.18 VDA

[Debugging Information]

```
# cat /proc/umap/vda

[VDA] Version: [Hi3531_MPP_V1.0.0.0 Debug], Build Time[May 29 2012, 09:29:27]
-----VDA CHN ATTR-----------------------------------------------
NO.    W    H  Mode   Alg  MbSz MbBit RfM BufN  Itl BgWt RgnN
  0  720  576     0     1     1     0   0    8    6  128    1
  1  720  576     1     1     0     0   0    0    6  128    1
-----VDA CHN ATTR 2---------------------------------------------
 NO. Rgn    X    Y    W    H SadTh  ArTh  OcTh  UnOcTh  ObjN  Sad  Obj  Plx
   0   0    0    0  720  576   100     0     0       0   128    1    1    1
   1   0    0    0  720  576   100    60     6       2   128    0    0    0
-----VDA STATE--------------------------------------------------------
NO. Rgn    BgPhy BgSrd    RfPhy   RfStd bScdu bFst  OcCt  UnOc     VdaC
bRefUp
0   0       0     0       0        0      1     0     0     0        36       0
1   0       0     0       0        0      1     0     0     0        36       0
-----VDA STATE2------------------------------------------------------
NO.   MemPhy   MemVir MemSz  CRfPhy RgnIx  ViSd VdecSd  VpssSd  UserSd   Recv
CalDsuC LstPicC
0 868ec000 c4918000 22944 862f4000    0    250      0       0        0     36
0        0
1 868f2000 c48fa000   352  862f4000    0    250      0       0        0     36
0        0
```

[Analysis]

This section records the current attributes and status of the VDA.

[Parameter Description]

Parameter		Description
VDA CHN ATTR	NO.	Channel ID
	W	Channel width (in pixel)
	H	Channel height (in pixel)
	Mode	Channel mode 0: motion detection (MD) 1: occlusion detection (OD)
	Alg	Algorithm 0: background algorithm 1: frame reference algorithm

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	MbSz	Macroblock size 0: 8x8 1: 16x16
	MbBit	SAD output bits 0: 8 bits 1: 16 bits
	RfM	Reference frame mode 0: dynamic mode 1: static mode 2: user mode
	BufN	Number of MD result buffers
	Itl	VDA interval (in frame)
	BgWt	Background algorithm refresh weight
	RgnN	Number of regions in a channel The OD channel supports multiple regions.
VDA CHN ATTR 2 The MD channel contains only one region, whereas the OD channel contains multiple regions.	NO.	Channel ID
	Rgn	Region ID
	X	Start horizontal coordinate of a region relative to the channel
	Y	Start vertical coordinate of a region relative to the channel
	W	Region width (in pixel)
	H	Region height (in pixel)
	SadTh	Region SAD alarm threshold
	ArTh	Region alarm area threshold
	OcTh	Region occlusion count alarm threshold
	UnOcTh	Threshold of allowed uncover count when the region occlusion count is collected
	ObjN	Maximum number of output OBJ regions (some parts are motion parts)
	Sad	Whether to output the macroblock SAD value 0: not output 1: output
	Obj	Whether to output OBJ regions 0: not output

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		1: output
	Plx	Whether to output the number of alarm pixels 0: not output 1: output
VDA STATE	NO.	Channel ID
	Rgn	Region ID
	BgPhy	Background address
	BgSrd	Background picture stride
	RfPhy	Address of the reference frame
	RfStd	Reference frame stride
	bScdu	Whether scheduling is being performed
	bFst	Whether it is the initial scheduling
	OcCt	Region occlusion count
	UnOc	Allowed uncover count when the region occlusion count is collected
	VdaC	VDA completion count
	bRefUp	Whether the reference frame is manually updated in user reference frame mode
VDA STATE2	NO.	Channel ID
	MemPhy	Start physical address for the memory occupied by the channel
	MemVir	Start virtual address for the memory occupied by the channel
	MemSz	Size of the memory occupied by the channel
	CRfPhy	Physical address for the public reference frame
	RgnIx	Region that is being scheduled
	ViSd	Number of pictures transmitted by the VI channel
	VdecSd	Number of pictures transmitted by the VDEC channel
	VpssSd	Number of pictures transmitted by the VPSS
	UserSd	Number of pictures transmitted by users
	Recv	Number of pictures received by the VDA channel
	CalDsuC	Number of times that the DSU is called
	LstPicC	Number of discarded pictures due to VDA

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		out-of-order

13.19 AI

The AI debugging information is classified into two types based on the audio interface used by the chip:

[Debugging Information About the SIO Interface]

```
# cat /proc/umap/ai

[AI] Version: [Hi3518_MPP_V1.0.5.0 Debug], Build Time: [Mar 15 2013, 13:07:47]

-----AI DEV ATTR-----------------------------------------------------
AiDev WorkMod   SampR  BitWid  ChnCnt  ClkSel SondMod  PoiNum  ExFlag  FrmNum
0 i2s_mas    8kHZ   16bit      2      1    mono     320      1     30

-----AI DEV STATUS---------------------------------------------------------
AiDev     IntCnt FrmTime  MaxFrmTime  DMAChn  DMAReq TranLen IsrTime
MaxIsrTime   DMAPhy0   DMAPhy1 CurDMAPhy Pingpong    ErrCnt
0        146   39999      40021      0      0    2560     264        415
84ff9000  84ff9a00  84ff9000     1        0

-----AI CHN STATUS---------------------------------------------------
AiDev   AiChn   State    Read   Write     BufFul u32Data0 u32Data1   AecAo
AecFail    bAnr  bResmp  PoiNum   SampR ResmpType
0      0  enable      0      0          0   df0187  1cd01f7 (-1,-1)
0 disable disable      0  (null)    (null)
0      1  enable      0      0      0   6300c7   4c0051 (-1,-1)          0
disable disable      0  (null)    (null)
-----VQE & ResampleEx-----------------------------------------------
AiDev   AiChn    bVqe    bAlc  MinLev  MaxLev MaxGain    bAec AecMode    bAnr
bHpf  bReSp   SamIn  SamOut  WrFile
0      0  enable  enable     -16     -4     12  enable   speaker  enable
disable  disable (null)  (null) disable
```

[Analysis]

This section records the attributes and status of the current AI device and AI channel.

[Parameter Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
AI DEV ATTR (for details, see AIO_ATTR_S.)	AiDev	ID of an AI device
	WorkMod	SIO operating mode i2s_mas: master I^2S mode i2s_sla: slave I^2S mode pcm0_mt: standard master PCM mode pcm0_sl: standard slave PCM mode pcm1_mt: non-standard master PCM mode pcm1_sl: non-standard slave PCM mode
	SampR	Sampling rate Value: {8–48 kHz}
	BitWid	Bit width Value: {8 bits, 16 bits}
	ChnCnt	Maximum number of channels Value {2, 4, 8, 16}
	ClkSel	Clock select
	SondMod	Sound mode mono: mono channel stereo: stereo channel
	PoiNum	Number of sampling points in a frame
	ExFlag	8-bit extension flag
	FrmNum	Number of frames in a buffer
AI DEV STATUS	AiDev	ID of an AI device
	IntCnt	Number of interrupts This number increases by 1 after a frame of audio data is collected. If the number does not increase, the AO device does not properly connect to the external codec.
	FrmTime	Frame interval time (in μs) The actual audio sampling rate can be calculated based on FrmTime. For example, when FrmTime is 40,000 μs and the number of sampling points in a frame is set to 320, the number of sampling points per second is 8000. That is, the sampling rate is 8 kHz.
	MaxFrmTime	Maximum interval between frames This parameter is used to obtain statistical information about the maximum interval between frames from the time the system starts till now.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	DMAChn	DMA channel
	DMAReq	DMA request signal
	TranLen	DMA transfer length (in byte)
	IsrTime	Time of processing a DMA interrupt
	MaxIsrTime	Maximum time used for processing a DMA interrupt This parameter is used to obtain statistical information about the maximum time used for processing a DMA interrupt from the time the system starts till now.
	DMAPhy0	Physical address for DMA buffer 0
	DMAPhy1	Physical address for DMA buffer 1
	CurDMAPhy	Physical address that is being used by the current DMA
	Pingpong	Pingpong memory that is being used by the current software
	ErrCnt	Number of times that the Pingpong memory has errors
AI CHN STATUS	AiDev	ID of an AI device
	AiChn	ID of an AI channel
	State	Channel status orig: initialization enable: enabled disable: disabled
	Read	Read pointer of a channel buffer
	Write	Write pointer of a channel buffer
	BufFul	Number of times that the frame buffer is full
	u32Data0	First 32-bit data segment in a channel buffer
	u32Data1	Second 32-bit data segment in a channel buffer
	AecAo	IDs of the AO device and AO channel for which the audio echo cancellation (AEC) function is enabled. The value **–1** indicates that the AEC function is disabled.
	AecFail	Number of times that echo cancellation fails
	bAnr	Channel denoising enable enable: Denoising is enabled for this channel. disable: Denoising is disabled for this channel.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	bHpf	High-pass filtering enable for the channel enable: enabled disable: disabled
	bResmp	Channel resampling enable enable: Resampling is enabled for this channel. disable: Resampling is disabled for this channel.
	PoiNum	Number of sampling points in the data frames input for resampling when channel resampling is enabled
	SampR	Sampling rate in the data frames input for resampling when channel resampling is enabled
	ResmpType	Channel resampling type
VQE & ResampleEx (VQE and advanced resampling information)	AiDev	ID of an AI device
	AiChn	ID of an AI channel
	bVqe	VQE enable enable: The VQE is enabled. disable: The VQE is disabled.
	bAlc	ALC enable enable: The ALC is enabled. disable: The ALC is disabled.
	MinLev	Minimum target level (in dBm0)
	MaxLev	Maximum target level (in dBm0)
	MaxGain	Maximum gain
	bAec	AEC enable enable: The AEC is enabled. disable: The AEC is disabled.
	AecMode	AECmode. receiver: receiver mode speaker: speaker mode headphone: headphone mode
	bAnr	ANR enable enable: The ANR is enabled. disable: The ANR is disabled.
	bResmp	Advanced resampling enable enable: Resampling is enabled. disable: Resampling is disabled.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	SamIn	Input sampling rate
	SamOut	Output sampling rate after resampling
	WrFile	Saving file enable enable: The saving file function is enabled. disable: The saving file function is disabled.

[Debugging Information About the AIO Interface]

```
# cat /proc/umap/ai

[AI] Version: [Hi3520D_MPP_V0.0.0.0 Debug], Build Time: [Feb 27 2013,
14:01:00]

-----AI DEV ATTR-----------------------------------------------------
AiDev WorkMod   SampR  BitWid  ChnCnt  ClkSel SondMod  PoiNum  ExFlag  FrmNum
0 i2s_sla    8kHZ   16bit      2       0    mono     320       1      30

-----AI DEV STATUS--------------------------------------------------
AiDev      IntCnt     fifoCnt     buffInt FrmTime  MaxFrmTime TranLen IsrTime
MaxIsrTime     CBPhy  CBSize   ROffSet   WOffSet
0       1124          0          0   40026      40081    1280    1628
1733  84e72000    2560        0         0

-----AI CHN STATUS-------------------------------------------
AiDev   AiChn   State    Read   Write     BufFul u32Data0 u32Data1   AecAo
AecFail    bAnr  bResmp  PoiNum   SampR ResmpType
0       0  enable      0       0          0 f376f37d f363f368 (-1,-1)
0 disable disable      0  (null)    (null)
0       1  enable      0       0          0 f46af464 f469f471 (-1,-1)
0 disable disable      0  (null)    (null)
```

[Analysis]

This section records the current attributes and status of the AI device and AI channel.

[Parameter Description]

Parameter		Description
AI DEV ATTR (for details, see AIO_ATTR_S.)	AiDev	ID of an AI device
	WorkMod	AI operating mode i2s_mas: master I^2S mode i2s_sla: slave I^2S mode

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		pcm0_mt: standard master PCM mode pcm0_sl: standard slave PCM mode pcm1_mt: non-standard master PCM mode pcm1_sl: non-standard slave PCM mode
	SampR	Sampling rate Value: {8–48 kHz}
	BitWid	Bit width Value: {8 bits, 16 bits}
	ChnCnt	Maximum number of channels Value {2, 4, 8, 16}
	ClkSel	Clock select
	SondMod	Sound mode mono: mono channel stereo: stereo channel
	PoiNum	Number of sampling points in a frame
	ExFlag	8-bit extension flag
	FrmNum	Number of frames in a buffer
AI DEV STATUS	AiDev	ID of an AI device
	IntCnt	Number of interrupts This number increases by 1 after a frame of audio data is collected. If the number does not increase, the AO device does not properly connect to the external codec.
	fifoCnt	Number of FIFO overflow interrupts This number increases by 1 if overflow occurs when the AI RX FIFO is full.
	buffInt	Number of DMA buffer full interrupts This number increases by 1 when the DMA buffer for storing the data received by the AIU is full.
	FrmTime	Frame interval time (in μs) The actual audio sampling rate can be calculated based on FrmTime. For example, when FrmTime is 40,000 μs and the number of sampling points in a frame is set to 320, the number of sampling points per second is 8000. That is, the sampling rate is 8 kHz.
	MaxFrmTime	Maximum interval between frames This parameter is used to obtain statistical information about the maximum interval between frames from the

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		time the system starts till now.
	TranLen	DMA transfer length (in byte)
	IsrTime	Time of processing a DMA interrupt
	MaxIsrTime	Maximum time used for processing a DMA interrupt This parameter is used to obtain statistical information about the maximum time used for processing a DMA interrupt from the time the system starts till now.
	CBPhy	Physical address for the DMA buffer
	CBSize	DMA buffer size
	ROffSet	Offset of the read pointer of the DMA buffer relative to the start address
	WOffSet	Offset of the write pointer of the DMA buffer relative to the start address
AI CHN STATUS	AiDev	ID of an AI device
	AiChn	ID of an AI channel
	State	Channel status orig: initialization enable: enabled disable: disabled
	Read	Read pointer of a channel buffer
	Write	Write pointer of a channel buffer
	BufFul	Number of times that the frame buffer is full
	u32Data0	First 32-bit data segment in a channel buffer
	u32Data1	Second 32-bit data segment in a channel buffer
	AecAo	IDs of the AO device and AO channel for which the audio echo cancellation (AEC) function is enabled. The value **–1** indicates that the AEC function is disabled.
	AecFail	Number of times that echo cancellation fails
	bAnr	Channel denoising enable enable: Denoising is enabled for this channel. disable: Denoising is disabled for this channel.
	bResmp	Channel resampling enable enable: Resampling is enabled for this channel. disable: Resampling is disabled for this channel.
	PoiNum	Number of sampling points in the data frames input

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		for resampling when channel resampling is enabled
	SampR	Sampling rate in the data frames input for resampling when channel resampling is enabled
	ResmpType	Channel resampling type

13.20 AO

The AO debugging information is classified into two types based on the audio interface used by the chip:

[Debugging Information About the SIO Interface]

```
# cat /proc/umap/ao

[AO] Version: [Hi3518_MPP_V1.0.5.0 Debug], Build Time: [Mar 15 2013, 13:07:47]

-----AO DEV ATTR-----------------------------------------------------
AoDev WorkMod   SampR  BitWid  ChnCnt  ClkSel SondMod  PoiNum  ExFlag  FrmNum
    0 i2s_mas    8kHZ   16bit       2       1    mono     320       1      30

-----AO DEV STATUS-------------------------------------------
AoDev      IntCnt FrmTime  MaxFrmTime  DMAChn  DMAReq TranLen IsrTime
MaxIsrTime   DMAPhy0   DMAPhy1 CurDMAPhy Pingpong    ErrCnt
    0         89   40015      40019      1      1    2560     161        161
85021000  85021a00  85021a10     0        0

-----AO CHN STATUS-----------------------------------------
AoDev   AoChn   State    Read   Write     BufEmp  u32Data0  u32Data1  bResmp
PoiNum   SampR ResmpType   Volume
0       0  enable     29     29          0    f00148     60120 disable      0
(null)    (null)       0
0       1  enable      0      0         89        0       0   disable      0
(null)    (null)
-----AO ResampleEx STATUS-----------------------------------
AoDev   AoChn   bReSp   SamIn  SamOut
0       0      disable  (null)  (null)
```

[Analysis]

This section records the attributes and status of the current AO device.

[Parameter Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
AO DEV ATTR (for details, see AIO_ATTR_S.)	AoDev	ID of an AO device
	WorkMod	SIO operating mode i2s_mas: master I^{2}S mode i2s_sla: slave I^{2}S mode pcm0_mt: standard master PCM mode pcm0_sl: standard slave PCM mode pcm1_mt: non-standard master PCM mode pcm1_sl: non-standard slave PCM mode
	SampR	Sampling rate Value: {8–48 kHz}
	BitWid	Bit width Value: {8 bits, 16 bits}
	ChnCnt	Maximum number of channels Value {2, 4, 8, 16}
	ClkSel	Clock select
	SondMod	Sound mode mono: mono channel stereo: stereo channel
	PoiNum	Number of sampling points in a frame
	ExFlag	8-bit extension flag
	FrmNum	Number of frames in a buffer
AO DEV STATUS	AoDev	ID of an AO device
	IntCnt	Number of interrupts This number increases by 1 after a frame of audio data is transmitted. If the number does not increase, the AO device does not properly connect to the external codec.
	FrmTime	Interval between frames (in μs) The actual audio sampling rate can be calculated based on FrmTime. For example, when FrmTime is 40,000 μs and the number of sampling points in a frame is set to 320, the number of sampling points per second is 8000. That is, the sampling rate is 8 kHz.
	MaxFrmTime	Maximum interval between frames This parameter is used to obtain statistical information about the maximum interval between frames from the time the system starts till now.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	DMAChn	DMA channel
	DMAReq	DMA request signal
	TranLen	DMA transfer length (in byte)
	IsrTime	Time used for processing a DMA interrupt
	MaxIsrTime	Maximum time used for processing a DMA interrupt This parameter is used to obtain statistical information about the maximum time used for processing a DMA interrupt from the time the system starts till now.
	DMAPhy0	Physical address for DMA Buffer0
	DMAPhy1	Physical address for DMA Buffer1
	CurDMAPhy	Physical address that is being used by the current DMA
	Pingpong	Pingpong memory that is being used by the current software
	ErrCnt	Number of times that the Pingpong memory has errors
AO CHN STATUS	AoDev	ID of an AO device
	AoChn	ID of an AO channel
	State	Channel status orig: initialization enable: enabled disable: disabled
	Read	Read pointer of a channel buffer
	Write	Write pointer of a channel buffer
	BufEmp	Number of times that the frame buffer is empty
	u32Data0	First 32-bit data segment in a channel buffer
	u32Data1	Second 32-bit data segment in a channel buffer
	bResmp	Channel resampling enable enable: Resampling is enabled for this channel. disable: Resampling is disabled for this channel.
	PoiNum	Number of sampling points in the data frames input for resampling when channel resampling is enabled
	SampR	Sampling rate in the data frames input for resampling when channel resampling is enabled
	ResmpType	Channel resampling type

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	Volume	AO device volume
AO ResampleEx STATUS (AO advanced resampling information)	AoDev	ID of an AO device
	AoChn	ID of an AO channel
	bReSp	Channel resampling enable enable: Resampling is enabled for this channel. disable: Resampling is disabled for this channel.
	SamIn	Input sampling rate
	SamOut	Output sampling rate after resampling

[Debugging Information About the AIO Interface]

```
# cat /proc/umap/ao
[AO] Version: [Hi3520D_MPP_V0.0.0.0 Debug], Build Time: [Feb 27 2013,
14:01:00]

-----AO DEV ATTR-----------------------------------------------------
AoDev WorkMod   SampR  BitWid  ChnCnt  ClkSel SondMod  PoiNum  ExFlag  FrmNum
    0 i2s_sla    8kHZ   16bit       2       0    mono     320       1      30

-----AO DEV STATUS--------------------------------------------------------
AoDev      IntCnt     fifoCnt     buffInt FrmTime  MaxFrmTime TranLen IsrTime
MaxIsrTime     CBPhy  CBSize   ROffSet   WOffSet
0       1192       0          0   40053      40053    1280     738        740
84e99000    3840        0        0

-----AO CHN STATUS--------------------------------------------------------
AoDev   AoChn   State    Read   Write     BufEmp  u32Data0  u32Data1  bResmp
PoiNum   SampR RespmType
0   0  enable     22     22    0  f380f387  f388f37c disable  0  (null)
(null)
```

[Analysis]

This section records the attributes and status of the current AO device.

[Parameter Description]

Parameter		Description
AO DEV ATTR (for details, see AIO_ATTR_S.)	AoDev	ID of an AO device
	WorkMod	AO operating mode i2s_mas: master I^2S mode

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		i2s_sla: slave I^{2}S mode pcm0_mt: standard master PCM mode pcm0_sl: standard slave PCM mode pcm1_mt: non-standard master PCM mode pcm1_sl: non-standard slave PCM mode
	SampR	Sampling rate Value: {8–48 kHz}
	BitWid	Bit width Value: {8 bits, 16 bits}
	ChnCnt	Maximum number of channels Value {2, 4, 8, 16}
	ClkSel	Clock select
	SondMod	Sound mode mono: mono channel stereo: stereo channel
	PoiNum	Number of sampling points in a frame
	ExFlag	8-bit extension flag
	FrmNum	Number of frames in a buffer
AO DEV STATUS	AoDev	ID of an AO device
	IntCnt	Number of interrupts This number increases by 1 after a frame of audio data is transmitted. If the number does not increase, the AO device does not properly connect to the external codec.
	fifoCnt	Number of FIFO underflow interrupts This number increases by 1 if an underflow occurs when the AO TX FIFO is empty.
	buffInt	Number of DMA buffer empty interrupts This number increases by 1 when the DMA buffer for storing the data transmitted by the AOU is empty.
	FrmTime	Interval between frames (in μs) The actual audio sampling rate can be calculated based on FrmTime. For example, when FrmTime is 40,000 μs and the number of sampling points in a frame is set to 320, the number of sampling points per second is 8000. That is, the sampling rate is 8 kHz.
	MaxFrmTime	Maximum interval between frames This parameter is used to obtain statistical information

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		about the maximum interval between frames from the time the system starts till now.
	TranLen	DMA transfer length (in byte)
	IsrTime	Time used for processing a DMA interrupt
	MaxIsrTime	Maximum time used for processing a DMA interrupt This parameter is used to obtain statistical information about the maximum time used for processing a DMA interrupt from the time the system starts till now.
	CBPhy	Physical address for the DMA buffer
	CBSize	DMA buffer size
	ROffSet	Offset of the read pointer of the DMA buffer relative to the start address
	WOffSet	Offset of the write pointer of the DMA buffer relative to the start address
AO CHN STATUS	AoDev	ID of an AO device
	AoChn	ID of an AO channel
	State	Channel status orig: initialization enable: enabled disable: disabled
	Read	Read pointer of a channel buffer
	Write	Write pointer of a channel buffer
	BufEmp	Number of times that the frame buffer is empty
	u32Data0	First 32-bit data segment in a channel buffer
	u32Data1	Second 32-bit data segment in a channel buffer
	bResmp	Channel resampling enable enable: Resampling is enabled for this channel. disable: Resampling is disabled for this channel.
	PoiNum	Number of sampling points in the data frames input for resampling when channel resampling is enabled
	SampR	Sampling rate in the data frames input for resampling when channel resampling is enabled
	ResmpType	Channel resampling type

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

13.21 AENC

[Debugging Information]

```
# cat /proc/umap/aenc
[AENC] Version: [Hi3531_MPP_V1.0.0.0 Debug], Build Time[Nov 30 2011, 21:17:11]
-----AENC CHN ATTR-----------------------------------------------------
ChnId  PlType BufSize   Attr1   Attr2   Attr3   Attr4   Attr5
0   adpcm     30
-----AENC CHN STATUS-------------------------------------------------
ChnId  RcvFrm   EncOk  FrmErr BufFull GetStrm RlsStrm  WtFile
0     316     315      0      0       315     315     disable
```

[Analysis]

This section records the attributes and status of the current AENC channel.

[Parameter Description]

Parameter		Description
Attribute of AENC Channel	ChnId	ID of an AENC channel
	PlType	Type of the encoding protocol
	BufSize	Frame buffer size
	Attr1–Attr5	Attributes related to protocols. The specific attributes depend on the protocol type.
Status of AENC Channel	ChnId	ID of an AENC channel
	RcvFrm	Number of received audio frames
	EncOk	Number of audio frames that are successfully encoded
	EncErr	Number of audio frames that fail to be encoded
	BufFull	Number of times that the audio stream buffer is full. The buffer is full when the audio streams are not fetched in a timely manner. As a result, steams are lost.
	GetStrm	Number of times that users obtain audio streams
	RlsStrm	Number of times that users release audio streams
	WtFile	Saving file enable enable: The saving file function is enabled. disable: The saving file function is disabled.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

13.22 ADEC

[Debugging Information]

```
# cat /proc/umap/adec
[ADEC] Version: [Hi3531_MPP_V1.0.D.0 ], Build Time[Mar 27 2015, 17:01:17]
-----ADEC CHN ATTR-----------------------------------------------------
ChnId  PlType BufSize    SendCnt     GetCnt     PutCnt    AdecMode
SendFail
```

[Analysis]

This section records the attributes and status of the current ADEC channel.

[Parameter Description]

Parameter		Description
Attribute of ADEC Channel	ChnId	Channel ID
	PlType	Type of the decoding protocol
	BufSize	Frame buffer size
	SendCnt	Number of transmitted audio frames that are decoded by the decoder successfully
	GetCnt	Number of stream frames that are obtained by users
	PutCnt	Number of stream frames that are released by users
	AdecMode	Decoding mode PACK: pack decoding mode STREAM: stream decoding mode
	SendFail	Number of times that streams fail to be transmitted

13.23 MPEG4E

[Debugging Information]

```
# cat /proc/umap/mpeg4e
[MPEG4E] Version: [Hi3531_MPP_V1.0.0.0 ], Build Time[Dec  3 2014, 11:34:45]
-----MODULE PARAM------------------------------------------------------
Mpeg4eMiniBufMode
0
-----CHN ATTR---------------------------------------------------------------
ID  Priority  MaxWidth MaxHeight  PicWidth PicHeight   MainStr   VIField
BufSize   ByFrame MaxStrCnt
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
0        0       1920     1080       704      576       Yes       No
4147200      Yes      200

-----PICTURE INFO-----------------------------------------------------
ID   EncdStart      EncdSucceed     Lost   Disc    Skip    BufLeak Recode
RlsStr  UnrdStr
0    1683        1683        0      0      0      0      0    1683      0
-----Syntax INFO------------------------------------------------------
   ID   slcsplt slcmode slcsize HWSize  VWSize
   0    Yes     MbLine    12      0      0
-----STREAM BUFFER----------------------------------------------------
   ID   Base       RdTail     RdHead     WrTail     WrHead     DataLen
BufFree
   0    0xcb400000  0x9ce80     0x9ce80     0x9ce80     0x9ce80       0
4149184
```

[Analysis]

During multi-channel MPEG4 encoding, the upper-layer software may not fetch streams in a timely manner. As a result, the stream buffer is insufficient, which causes frame loss. You can view the Disc, BufLeak, and UnrdStr parameters to check whether frames are lost. BufLeak indicates the number of frames that are discarded when the software detects stream buffer insufficiency. Disc indicates the number of frames that are discarded when the hardware detects stream buffer insufficiency. BufLeak and Disc indicate the total number of discarded frames due to stream buffer insufficiency. UnrdStr indicates the number of remaining frames in the stream buffer. If the value of UnrdStr is great, streams are not fetched in a timely manner.

[Parameter Description]

Parameter		Description
MODULE PARAM	Mpeg4eMiniBufMode	Configuration mode of the stream buffer 0: normal mode 1: memory reduction mode
CHN ATTR	ID	Channel ID
	Pri	Channel priority. This parameter is not used currently.
	MaxWidth	Maximum width of the encoding channel (in pixel)
	MaxHeight	Maximum height of the encoding channel (in pixel)
	PicWidth	Picture width (in pixel)
	PicHeight	Picture height (in pixel)
	MainStr	Whether streams are large stream Value: {0, 1}
	VIField	This parameter is not used currently and its value is fixed at No.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	BufSize	Stream buffer size (in byte)
	ByFrame	Whether to obtain streams by frame Value: {0, 1}
	MaxStrCnt	Maximum number of frames in the stream buffer Default value: 0x1FFFFFFF
PICTURE INFO	ID	Channel ID
	EncdStart	Number of received pictures This value increases by 1 after the pictures to be encoded are received.
	EncdSucceed	Number of frames that are successfully encoded
	Lost	Number of frames that are discarded during encoding. The discarded frames include: • Number of frames discarded due to encoder exceptions or jumbo frames. The number of these frames is the value of Disc. • Frames that are discarded for controlling the frame rate. The frame rate is controlled by the RC. The number of these frames is the value of Skip. • Frames that are discarded due to stream buffer insufficiency. The number of these frames is the value of BufLeak.
	Disc	Number of frames discarded for the following reasons: • The encoder detects stream buffer insufficiency during encoding. • Jumbo frames occur. • An exception occurs in the encoder. For example, the encoder crashes. The first reason is the most common one. If the value of Disc continuously increases, it is probably that the upper-layer software does not fetch streams in a timely manner, which causes buffer insufficiency.
	Skip	Number of frames that are discarded for controlling the frame rate during encoding
	BufLeak	Number of frames that are discarded due to stream buffer insufficiency The number increases by 1 when the encoder detects stream buffer insufficiency and then discards a frame before encoding. The number is not 0 when streams are not fetched in a timely manner.
	Recode	Number of frames that are re-encoded by the encoder

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	RlsStr	Number of frames that are obtained and released by users
	UnrdStr	Number of frames that are not obtained by users
STREAM BUFFER	ID	Channel ID
	Base	Base address for a stream buffer
	RdTail	Read tail pointer
	RdHead	Read head pointer
	WrTail	Write tail pointer
	WrHead	Write head pointer
	DataLen	Data length
	BufFree	Free buffer size

13.24 HDMI

[Debugging Information]

```
# cat /proc/umap/hdmi

[HDMI] Version: [Hi3531_MPP_V1.0.0.0 Debug], Build Time[Nov 26 2012, 17:07:39]

-----MODULE
PARAM-------------------------------------------------------------

-----HDMI
STATE-------------------------------------------------------------------
 DevId  Open Start    Event
    0     y     y    HOT PLUG

############ Hisi HDMI  Dev Stat ############
HPD Status:              IN
TV:                     Power On
EDID Parse Status:        Ok
TV ManufactureName:       DEL
HDMI Sink video capability:
TV support:              1080P_60
TV support:              1080P_50
TV support:              1080P_24
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
TV support:             1080i_60
TV support:             1080i_50
TV support:             720P_60
TV support:             720P_50
TV support:             576P_50
TV support:             480P_60
TV support:             576i_50
TV support:             480i_60
TV support:             640X480_60
TV support:             800X600_60
TV support:             1024X768_60
CEC Status:             OFF
HDMI Output Attribute:
HDCP Encryption:         OFF
HDMI PHY Output:         Enable
Video Output:            Enable
Video Format:            1080P_60
Video Output mode:       YCbCr444
Video DeepColor mode:    off
Audio Output:            Enable
Audio input format:      I2S
Audio sample rate:       48000 Hz
HDMI Mode:              HDMI
AVI Infoframe:           Enable
AUD Infoframe:           Enable
MPG/VendorSpec Infoframe:  Disable
Gamut Metadata Packet:    Disable
Generic Packet Packet:    Disable
AVMUTE:                 Disable
AVI Inforframe:
0x82,0x02,0x0d,0x67,0x50,0xa8,0x00,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,0x00,
AUD Inforframe:
0x84,0x01,0x0a,0x70,0x01,0x00,0x00,0x00,0x00,
EDID Raw Data:
00 ff ff ff ff ff ff 00 10 ac 16 f0 4c 39 44 41
1c 15 01 03 80 34 20 78 ea 1e c5 ae 4f 34 b1 26
0e 50 54 a5 4b 00 81 80 a9 40 d1 00 71 4f 01 01
01 01 01 01 01 01 28 3c 80 a0 70 b0 23 40 30 20
36 00 06 44 21 00 00 1a 00 00 00 ff 00 38 32 57
58 44 31 37 34 41 44 39 4c 0a 00 00 00 fc 00 44
45 4c 4c 20 55 32 34 31 30 0a 20 20 00 00 00 fd
00 38 4c 1e 51 11 00 0a 20 20 20 20 20 20 01 32
02 03 29 f1 50 90 05 04 03 02 07 16 01 1f 12 13
```


HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

```
14 20 15 11 06 23 09 07 07 67 03 0c 00 10 00 38
2d 83 01 00 00 e3 05 03 01 02 3a 80 18 71 38 2d
40 58 2c 45 00 06 44 21 00 00 1e 01 1d 80 18 71
1c 16 20 58 2c 25 00 06 44 21 00 00 9e 01 1d 00
72 51 d0 1e 20 6e 28 55 00 06 44 21 00 00 1e 8c
0a d0 8a 20 e0 2d 10 10 3e 96 00 06 44 21 00 00
18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 3e
#################### END ##################
```

[Analysis]

Records the current running parameters, hot-plug status, and monitor EDID for the HDMI port.

[Parameter Description]

Parameter		Description
HDMI STATE	DevId	HDMI port ID
	Open	Whether the HDMI port is on
	Start	Whether the HDMI port is enabled Value: {y, n} If the hot-plug detection mechanism detects that the HDMI cable is disconnected, the HDMI port status changes to n automatically. If the HDMI cable is connected, the HDMI port status changes to y automatically.
	Event	HDMI hot-plug event HOT PLUG: The HDMI port connects to the monitor. NO PLUG: The HDMI port dos not connect to the monitor through the HDMI cable. EDID FAL: The monitor EDID fails to be read. HDCP FAL: The HDCP authentication fails. HDCP SUC: The HDCP authentication is successful. HDCP RST: The HDCP authentication is reset and must be performed again.
Hisi HDMI Dev Stat	HPD Status	HDMI port status detected by the hot-plug detection mechanism IN: The HDMI port connects to the monitor. OUT: The HDMI port does not connect to the monitor through the HDMI cable.
	TV	Monitor power-on/off status Power off: The monitor is powered off. Power on: The monitor is powered on.
	EDID Parse Status	EDID parsing status

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
		OK: The EDID parsing is successful. Fail: The EDID parsing fails.
	TV ManufactureName	Monitor manufacturer name
	HDMI Sink video capability	Resolution capability set supported by the monitor
	TV support	Resolution supported by the monitor
	CEC Status	Consumer electronics control (CEC) status ON: The CEC is on. If the CEC status is on, the following two subitems are displayed: • CEC Physical Add: physical address for the CEC. • CEC Logical Add: logical address for the CEC. Off: The CEC is off.
	HDMI Output Attribute	HDMI output attributes
	HDCP Encryption	HDMI encryption status Enable: The HDMI encryption is enabled. Disable: The HDMI encryption is disabled.
	HDMI PHY Output	PHY transmitter IP status Enable: The PHY transmitter IP is enabled Disable: The PHY transmitter IP is disabled.
	Video Output	Video output status Enable: The video output is enabled. Disable: The video output is disabled.
	Video Format	Video output timing
	Video Output mode	Video output mode Value: RGB444, YCbCr422, or YCbCr444
	Video DeepColor mode	Video DeepColor mode
	Audio Output	Audio output status Enable: The audio output is enabled. Disable: The audio output is disabled.
	Audio input format	Audio input format Value: I^2S, SPDIF, or HighBitRate
	Audio sample rate	Audio sampling rate
	HDMI Mode	HDMI or DVI mode

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
	AVI Infoframe	AVI information frame status Enable: The AVI information frame is enabled. Disable: The AVI information frame is disabled.
	AUD Infoframe	Audio information frame status Enable: The audio information frame is enabled. Disable: The audio information frame is disabled.
	MPG/VendorSpec Infoframe	MPEG information frame status Enable: The MPEG information frame is enabled. Disable: The MPEG information frame is disabled.
	Gamut Metadata Packet	Gamut metadata transmission status Enable: The Gamut transmission is enabled. Disable: The Gamut transmission is disabled.
	Generic Packet	Generic packet transmission status Enable: The generic packet transmission is enabled. Disable: The generic packet transmission is disabled.
	AVMUTE	Mute status Enable: The mute function is enabled. Disable: The mute function is disabled.
	AVI Inforframe	AVI information frame data
	AUD Inforframe	AUD information frame data
	EDID Raw Data	Raw EDID data

13.25 ACODEC

[Debugging Information]

```
# cat /proc/umap/acodec
[ACODEC] Version: [Hi3518_MPP_V1.0.0.0 Release], Build Time[Aug 14 2014,
14:11:21]
-----ACODEC PARAM-----------------------------------------------------
LGain RGain DacLVol DacRVol AdcLVol AdcRVol MicLMut MicRMut DacLMut DacRMut
Boost
11     11      9      9      30      30       0       0        0
0      1
```

[Parameter Description]

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd

Parameter		Description
ACODEC PARAM	LGain	Input analog gain of the audio-left channel Value range: [0, 0x1f]
	RGain	Input analog gain of the audio-right channel Value range: [0, 0x1f]
	DacLVol	Input digital gain of the audio-left channel Value range: [0, 0x7f]
	DacRVol	Input digital gain of the audio-right channel Value range: [0, 0x7f]
	AdcLVol	Output digital gain of the audio-left channel Value range: [0, 0x7f]
	AdcRVol	Output digital gain of the audio-right channel Value range: [0, 0x7f]
	MicLMut	Input mute status of the audio-left channel 0: disabled 1: enabled
	MicRMut	Input mute status of the audio-right channel 0: disabled 1: enabled
	DacLMut	Output mute status of the audio-left channel 0: disabled 1: enabled
	DacRMut	Output mute status of the audio-right channel 0: disabled 1: enabled
	Boost	BOOST enable status 0: disabled 1: enabled

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd