HiMPP V4.0 Media Processing Software

FAQs

Issue 10

Date 2019-09-12

HiSilicon (Shanghai) Technologies Co., Ltd.

Address:	New R&D Center, 49 Wuhe Road, Bantian, Longgang District, Shenzhen 518129 P. R. China
Website:	http://www.hisilicon.com/en/
Email:	support@hisilicon.com

About This Document

Purpose

This document describes the solutions to the problems that may occur when you use the HiSilicon media processing platform 4.0 (HiMPP V4.0).

NOTE

- Unless otherwise specified, the descriptions of Hi3516E V200 also apply to Hi3518E V300 and Hi3516E V300.
- Unless otherwise specified, the descriptions of Hi3559A V100 also apply to Hi3559C V100.
- Unless otherwise specified, the descriptions of Hi3516C V500 also apply to Hi3556 V200, Hi3559V200, Hi3516A V300, and Hi3516D V300.
- Unless otherwise specified, the descriptions of Hi3556A V100 also apply to Hi3519A V100.

Related Versions

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

Product Name	Version
Hi3559A	V100
Hi3559C	V100
Hi3519A	V100
Hi3556A	V100
Hi3516C	V500
Hi3516D	V300
Hi3516A	V300
Hi3559	V200
Hi3556	V200
Hi3516E	V200
Hi3516E	V300
Hi3518E	V300

Product Name	Version
Hi3516D	V200

Intended Audience

This document is intended for:

- Technical support engineers
- Software development engineers

Conventions

Symbol Conventions

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

Symbol	Description
DANGER	Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.
WARNING	Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.
CAUTION	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury.
NOTICE	Indicates a potentially hazardous situation which, if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results. NOTICE is used to address practices not related to personal injury.
NOTE	Calls attention to important information, best practices and tips. NOTE is used to address information not related to personal injury, equipment damage, and environment deterioration.

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**.

Convention	Description
Italic	Book titles are in *italics*.
`Courier New`	Examples of information displayed on the screen are in `Courier New`.

Change History

Changes between document issues are cumulative. The latest document issue contains all changes made in previous issues.

Issue 10 (2019-09-12)

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

Section 3.3.1.2 is modified.

Issue 09 (2019-07-30)

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

Section 1.2.2 is modified. The description of region management configuration is added.

Sections 1.2.3 and 1.6 are added.

Sections 3.3 and 3.4 are modified.

Issue 08 (2019-06-20)

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

Sections 1.2.2 and 1.2.3 are modified.

Issue 07 (2019-05-30)

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

In section 1.2.2, the ISP module configuration is updated.

Section 3.3.5 is added.

Issue 06 (2019-05-15)

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

Sections 3.6 and 6.6 are added.

Section 2.2 is modified.

Issue 05 (2019-03-15)

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

Sections 1.2.2, 3.4.2, and 3.4.3 are modified.

Issue 04 (2019-02-28)

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

Sections 1.2.2 and 3.4.2 are modified.

Sections 3.5, 6.5 and 9.1.2 are added.

Issue 03 (2019-01-30)

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

Section 1.2.1 is added.

Issue 02 (2019-01-14)

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

Section 3.4.2 is modified.

Section 12.1.5 is added.

Issue 01 (2018-12-12)

This issue is the first official release.

Contents

1 System Control

1.1 Log Information

1.1.1 How Do I View HiMPP Logs?

[Symptom]

How do I view HiMPP logs and change the log level?

[Cause Analysis]

The logs record the error causes, error locations, and system running status during the running of the software development kit (SDK). Logs can help you locate errors.

Currently, the logs are classified into seven levels, and the default level is level 3. A higher log level indicates that more information is recorded. When the level is set to level 7, the information about the running status of the entire system is recorded in logs in real time. The mass information, however, significantly reduces the overall performance of the system. Typically, you are advised to set the log level to level 3. In this case, information is recorded in logs only when errors occur and most errors can be located.

[Solution]

You can run the following commands to obtain logs, and view or change the log level:

- To view the log level of each module, run the **cat /proc/umap/logmpp** command. Then, the log levels of all the modules are listed.
- To change the log level of a module, run the **echo "venc=4" > /proc/umap/logmpp** command. In this command, **venc** is a module name. This name must be the same as that displayed after the **cat** command is executed.
- To change the log levels of all the modules, run the **echo "all=4" > /proc/umap/logmpp** command.
- To obtain logs, run the **cat /dev/logmpp** command. Then, all the log information is displayed. If all the log information is read, the command is blocked until new log information is recorded. Press **Ctrl**+**C** to exit. If you do not want to block the waiting log information, run the **echo wait=0 > /proc/umap/logmpp** command. To use the device node in **/dev/logmpp**, run **open** and **read** commands.

1.2 Memory Usage

1.2.1 How Do I Adjust the Sizes of the Reserved Memory and Thread Stack of the Linux OS?

[Symptom 1]

The message "oom-killer" is displayed during service program running on the Linux operating system (OS).

[Cause Analysis]

The possible causes are:

- The OS memory is insufficient.
- The system reserved memory is too small.

[Solution]

- Increase the OS memory.
- Increase the reserved memory. You can set the size of the reserved memory to 4 MB (which is adjustable) by adding the following commands to **/etc/profile**:
  - **echo 2 >/proc/sys/kernel/randomize_va_space**
  - **echo 4096 >/proc/sys/vm/min_free_kbytes**

[Symptom 2]

The error message "pthread_create: Resource temporarily unavailable" is displayed for a thread creation failure during service program running on the Linux OS.

[Cause Analysis]

The possible causes are:

- The OS memory is insufficient.
- The thread stack space is too large.

[Solution]

- Increase the OS memory.
- Adjust the maximum thread stack space by using either of the following methods:
  - For Linux, run the **ulimit -s** command to change the thread stack size. For example, to set the thread stack size to 1 MB, run the **ulimit -s 1024** command over the serial port. Alternatively, you can add the command to **/etc/profile** so that the stack space size is automatically set to 1 MB upon the next startup.
  - Use **pthread_attr_setstacksize** to change the thread stack size in the program.

1.2.2 How Do I Adjust the Memories Occupied by Media Services?

[Symptom]

Media services require memories for normal running. The memories mainly indicate the media memory zone (MMZ). HiMPP V4.0 allocates memories based on services. When the memories are insufficient, you can adjust the allocated memories.

[Cause Analysis]

The HiSilicon SDK allows you to adjust the allocated memories when the memories are insufficient. This section briefly describes the measures to minimize memory usage. For details, see related documents.

[Solution]

- Check the operating system (OS) memory and MMZ memory.

  For details, see chapter "Allocating and Using the Address Space" in the *Description of the Installation and Upgrade of the Hi35xx SDK*.

- Adjust the memories occupied by SDK services.
  - Entire system

    Ensure that all image resolutions of the chip are integer multiples, for example, 1080p represents 1920 x 1080 and 960H represents 960 x 480, while 960H cannot be 960 x 756 in this case. In addition, the VI module cannot capture 1920 x 1088 images when the VENC module encodes 1920 x 1080 images.

  - Minimum buffer size for each module

    For details, see the *HiMPP V4.0 Media Processing Software Development Reference*.

  - Just enough common video buffer (VB)

    For details, see HI_MPI_VB_SetConfig.

    For details, see chapter 2 "System Control" in the *HiMPP V4.0 Media Processing Software Development Reference*.

    The calculation of the VB size used by the output data of each module is complicated. For details about the calculation formula, see **hi_buffer.h**.

  - Video input (VI)

    For details, see chapter 3 "VI" in the *HiMPP V4.0 Media Processing Software Development Reference*. Run the following command to check the proc information: **cat /proc/umap/vi**

Measure	MPI and Parameter	Benefit	Impact	Note	Proc Information
Enable raw compression.	HI_MPI_VI_CreatePipe: **enCompressMode**	Compared with the non-compression mode, the memory and bandwidth are saved in raw compression mode.	-	Not supported by Hi3516E V200	VI PIPE ATTR1: CompressMode
Enable wrapping in online WDR line mode.	HI_MPI_VI_SetDevAttr: **stWDRAttr**	The frame of buffer does not need to be allocated if wrapping is used.	Improper configuration may lead to an abnormal image effect.	• Closely related to the sensor timing • Not supported by Hi3559A V100	VI DEV ATTR2:WDRMode CacheLine
Enable 3DNR compression.	HI_MPI_VI_CreatePipe: **bNrEn** and **stNrAttr**	Compared with the non-compression mode, the memory and bandwidth are saved in raw compression mode	-	Supported only by Hi3559A/C V100	VI PIPE NR ATTR: CompressMode

Measure	MPI and Parameter	Benefit	Impact	Note	Proc Information
Enable the Early_End mechanism.	HI_MPI_VI_SetPipeFrameInterruptAttr	A frame of buffer can be saved with proper adjustment.	Improper adjustment may lead to an abnormal image effect.	Closely related to the sensor timing	VI PIPE ATTR2: IntType EarlyLine

– Video processing subsystem (VPSS)

For details, see chapter 5 "VPSS" in the *HiMPP V4.0 Media Processing Software Development Reference*. Run the following command to check the proc information: **cat /proc/umap/vpss**

Measure	MPI and Parameter	Benefit	Impact	Note	Proc Information
Enable 3DNR compression.	HI_MPI_VPSS_CreateGrp: **stNrAttr**	Compared with the non-compression mode, the memory and bandwidth are saved in raw compression mode	-	Not supported by Hi3559A/C V100	VPSS GRP ATTR: RefCmp
Enable buffer reuse for the 3DNR reference frame and reconstruction frame.	Adaptive reuse	A frame of buffer can be saved.	-	• For Hi3519A V100, buffer reuse is not supported in 3DNR compression mode or when **enNrMotionMode** is set to **NR_MOTION_MODE_COMPENSATE**. • For Hi3516C V500, buffer reuse is not supported in 3DNR non-compression mode and when **enNrMotionMode** is set to **NR_MOTION_MODE_COMPENSATE**.	-

Measure	MPI and Parameter	Benefit	Impact	Note	Proc Information
Enable the VPSS-VENC low-delay single-buffer mode,	• Low delay: HI_MPI_VPSS_SetLowDelayAttr • Single-buffer mode: HI_MPI_VPSS_SetModParam: **bOneBufForLowDelay**	A frame of buffer can be saved.	Improper configuration of the line number may lead to an abnormal image effect.	-	VPSS CHN LOWDELAY ATTR: Enable LineCnt OneBufEnable
Enable the VPSS-VENC low delay wrapping.	• HI_MPI_VPSS_SetChnBufWrapAttr • HI_MPI_SYS_GetVPSSVENCWrapBufferLine	More memory and bandwidth are saved compared with the low-delay single-buffer mode.	Improper configuration may lead to an abnormal image effect.	• Closely related to the sensor timing • Supported only by Hi3516E V200/Hi3516E V300/Hi3518E V300	VPSS CHN BUF WRAP ATTR: Enable BufLine WrapBufSize
Enable the reuse of the input and output buffers.	HI_MPI_VPSS_EnableBufferShare HI_MPI_VPSS_DisableBufferShare The multiplexing is supported only when certain conditions are met. For details, see the description of HI_MPI_VPSS_EnableBufferShare in chapter "VPSS" of the *HiMPP V4.0 Media Processing Software Development Reference*.	A frame of buffer can be saved.	-	Supported only by Hi3516E V200/Hi3516E V300/Hi3518E V300/Hi3516C V500/Hi3516D V300/Hi3556 V200/Hi3559 V200	VPSS CHN ATTR: bBufferShare
Enable the Early_End mechanism.	HI_MPI_VPSS_SetGrpFrameInterruptAttr	A frame of buffer can be saved with proper adjustment.	Improper adjustment may lead to an abnormal image effect.	• Closely related to the sensor timing • Supported only by Hi3516E V200/Hi3516E V300/Hi3518E V300. In all-online mode with wrapping enabled	FRAME INTERRUPT ATTR: IntType EarlyLine

Measure	MPI and Parameter	Benefit	Impact	Note	Proc Information
				for CH0, you can adjust the line number for Early_End to save one VB for the sub stream. You are advised to adjust the line number in descending order.	
Adjust the number of block nodes according to the scenario.	HI_MPI_VPSS_SetModParam: **u32VpssSplitNodeNum**	The template memory can be saved.	-	Supported only by Hi3559A/C V100	MODULE PARAM: u32VpssSplitNodeNum
Enable HDR according to the scenario.	HI_MPI_VPSS_SetModParam: **bHdrSupport**	The HDR buffer memory can be saved.	-	Supported only by Hi3559A/C V100	MODULE PARAM: bHdrSupport
Disable the backup frame.	HI_MPI_VPSS_EnableBackupFrame and HI_MPI_VPSS_DisableBackupFrame	The input source buffer occupied by each VPSS group is saved by one frame.	When the VO module is paused, the background color of the device is displayed when during image switchover.	-	-
Output YUV compression data from CH0.	HI_MPI_VPSS_SetChnAttr: enCompressMode	The memory and bandwidth are saved compared with the non-compression mode.	-	• Supported only by Hi3516E V200/Hi3516E V300/Hi3518E V300 • YUV compression can be used with H.264/H.265 encoding.	VPSS CHN OUTPUT RESOLUTION: Compress

– Video encoding (VENC)

For details, see chapter 6 "VENC" in the *HiMPP V4.0 Media Processing Software Development Reference*. Run the following commands to check the proc information: **cat /proc/umap/venc**, **cat /proc/umap/h265e**, **cat /proc/umap/h264e**, **cat /proc/umap/ jpege**

Measure	MPI and Parameter	Benefit	Impact	Note	Proc Information
Dynamically switch the encoding resolution.	• HI_MPI_VENC_GetChnAttr • HI_MPI_VENC_SetChnAttr	The VENC channel is not destroyed when the encoding resolution is switched, which reduces memory fragments.	None	After the encoding resolution is switched, all parameters are restored to default values.	-
Allocate the encoding stream buffers in memory reduction mode.	HI_MPI_VENC_SetModParam: • u32H264eMiniBufMode • u32H265eMiniBufMode • u32JpegeMiniBufMode	The stream buffer size can be reduced.	If the memory reduction mode is used, the stream buffer size must be set to an appropriate size. Otherwise, re-encoding or frame discarding occurs due to insufficiency of the stream buffer.	None	MODULE PARAM: MiniBufMode (Supported by VENC/H265E/H264E/JPEGE)
Enable buffer reuse for the reference frame and reconstruction frame.	HI_MPI_VENC_CreateChn: **bRcnRefShareBuf**	About frames (RefNum + 1 – 1.3 x RefNum) of buffer can be saved.	For frame loss or re-encoding caused by exceptions such as jumbo frames, bit rate overshoot, and bit rate buffer full, only the I-frame can be inserted in the next frame.	Not supported by Hi3559A V100 or Hi3519A V100	cRefParam INFO: RcnRefShareBuf (Supported by H265E/H264E)
Enable dynamic	HI_MPI_VENC_SetModParam:	When the GOP mode is	None	None	MODULE PARAM:

Measure	MPI and Parameter	Benefit	Impact	Note	Proc Information
recycle of the reference frame buffer.	**u32FrameBufRecycle**	switched, the extra buffer of multiple reference frames can be dynamically released with the reduction of the reference frames.			FrameBufRecycle (Supported by VENC)
Limit the number of channels supported by the VENC module	• Linux: **VencMaxChnNum** • Huawei LiteOS: VENC_MODULE_PARAMS_S: **u32VencMaxChnNum**	Some of the OS memory can be saved.	None	None	MODULE PARAM: VencMaxChnNum (Supported by VENC)
Use the user VB mode to save buffer for multi-channel encoding at the same resolution.	HI_MPI_VENC_SetModParam: • enH264eVBSource • enH265eVBSource	More frame buffer is saved compared with the private VB mode.	None	None	MODULE PARAM: H265eVBSource (Supported by H265E) MODULE PARAM: H264eVBSource (Supported by H264E)

– Video decoding (VDEC)

For details, see chapter 7 "VDEC" in the *HiMPP V4.0 Media Processing Software Development Reference*. Run the following command to check the proc information: **cat /proc/umap/vdec**

Measure	MPI and Parameter	Benefit	Impact	Note	Proc Information
Use the buffer-saving mode for allocating the decoding stream buffer.	HI_MPI_VDEC_SetModParam: **u32MiniBufMode**	You can set the stream buffer to a smaller value.	-	-	MODULE PARAM: MiniBufMode
Set the	HI_MPI_VDEC_Set	Some of the	If the allocated	The video	MODULE

Measure	MPI and Parameter	Benefit	Impact	Note	Proc Information
maximum capability set of the decoder by scenario.	ModParam: **stVideoModParam** and **stPictureModParam**	MMZ memory can be saved.	memory video decoder for high-definition (VDH) is reduced, the decoding performance may deteriorate, but the function is not affected.	encoding decoding unit (VEDU) does not support the VDH configurations.	PARAM: MaxPicWidth MaxPicHeight MaxSliceNum VdhMsgNum VdhBinSize VdhExtMemLevel and MaxJpegeWidth MaxJpegeHeight SupportProgressive DynamicAllocate CapStrategy
Set the decoding channel capability set by scenario.	HI_MPI_VDEC_SetProtocolParam	Some of the OS memory can be saved.	-	-	CHN VIDEO ATTR & PARAMS: MaxVPS MaxSPS MaxPPS MaxSlice
Disable the TMV when no B-frame streams are involved during H.264 decoding.	HI_MPI_VDEC_CreateChn: **bTemporalMvpEnable**	The TMV buffer can be saved.	-	-	CHN VIDEO ATTR & PARAMS: TemporalMvp
Limit the number of channels supported by the VDEC module	Linux: • VdecMaxChnNum • VfmwMaxChnNum Huawei LiteOS: • VDEC_MODULE_PARAMS_S • VFMW_MODULE_PARAMS_S	Some of the OS memory can be saved.	-	-	MODULE PARAM: VdecMaxChnNum
Set the reference frame to 0 for channels where only I-frames are decoded.	HI_MPI_VDEC_CreateChn: **u32RefFrameNum**	The buffer of the reference frame can be saved.	-	Set the channel decoding mode to I mode. Otherwise, a logmpp error is	CHN VIDEO ATTR & PARAMS: RefNum

Measure	MPI and Parameter	Benefit	Impact	Note	Proc Information
				reported.	

– Any view stitching (AVS)

For details, see chapter 12 "AVS" in the *HiMPP V4.0 Media Processing Software Development Reference*. Run the following command to check the proc information: **cat /proc/umap/avs**

Measure	MPI and Parameter	Benefit	Impact	Note	Proc Information
Set **WorkingSet** by scenario.	HI_MPI_AVS_SetModParam: **u32WorkingSetSize**	Some of the MMZ memory can be saved.	If this parameter is set to a value too small, the image effect is compromised.	-	cat /proc/umap/avs AVS WORKING SET: WorkingSetSize

– Video graphics subsystem (VGS)

For details, see chapter 10 "VGS" in the *HiMPP V4.0 Media Processing Software Development Reference*. Run the following command to check the proc information: **cat /proc/umap/vgs**

Note: The following parameter configuration applies to Linux only. The corresponding parameter structure for Huawei LiteOS is VGS_MODULE_PARAMS_S, which is configured in the VGS_init function in the **sdk_init.c** file.

Measure	Parameter	Benefit	Impact	Note	Proc Information
Limit the number of VGS jobs supported.	max_vgs_job	The default value is **128**. The memory can be saved with the deduction of the parameter value.	The VGS performance is restricted if the number of jobs is too small.	None	MODULE PARAM: max_job_num
Limit the number of VGS tasks supported.	max_vgs_task	The default value is **200**. The memory can be saved with the deduction of the parameter value.	The VGS performance is restricted if the number of tasks is too small.	None	MODULE PARAM: max_task_num
Limit the number of VGS nodes supported.	max_vgs_node	The default value is **200**. The memory can be saved with the deduction of the	The VGS performance is restricted if the number of nodes is	None	MODULE PARAM: max_node_

		parameter value.	too small.		num
Enable or disable the HDR function.	bVgsHdrSupport	The HDR buffer can be saved.	None	Supported only by Hi3559A V100	MODULE PARAM: bVgsHdrSupport

– Geometric distortion correction (GDC)

For details, see chapter 11 "GDC" in the *HiMPP V4.0 Media Processing Software Development Reference*. Run the following command to check the proc information: **cat /proc/umap/gdc**

Note: The following parameter configuration applies to Linux only. The corresponding parameter structure for Huawei LiteOS is GDC_MODULE_PARAMS_S, which is configured in the VGS_init function in the **sdk_init.c** file.

Measure	Parameter	Benefit	Impact	Note	Proc Information
Limit the number of GDC jobs supported.	max_gdc_job	The memory can be saved with the deduction of the parameter value.	If this parameter is set to a value too small, the GDC performance is affected.	The default value of Hi3559A V100 and Hi3519A V100 is **128**. The default value of other chips is **32**.	MODULE PARAM: max_job_num
Limit the number of GDC tasks supported.	max_gdc_task	The memory can be saved with the deduction of the parameter value.	If this parameter is set to a value too small, the GDC performance is affected.	The default value of Hi3559A V100 and Hi3519A V100 is **200**. The default value of other chips is **64**.	MODULE PARAM: max_task_num
Limit the number of GDC nodes supported.	max_gdc_node	The memory can be saved with the deduction of the parameter value.	If this parameter is set to a value too small, the GDC performance is affected.	The default value of Hi3559A V100 and Hi3519A V100 is **200**. The default value of other chips is **64**.	MODULE PARAM: max_node_num

– Video output (VO):

For details, see chapter 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference*. Run the following command to check the proc information: **cat /proc/umap/vo**

Measure	MPI	Benefit	Impact	Note	Proc Information
Set the minimum length	HI_MPI_VO_SetDispBufLen	One frame of buffer can be	The VO display	Hi3516C V500 and Hi3516E	VIDEO LAYER STATUS 3:

Measure	MPI	Benefit	Impact	Note	Proc Information
of the display queue in playback mode to **3**.		saved for the HD device.	smoothness is affected.	V200 do not support the playback mode.	u32BufLen
Set **DispBufLen** to **0** in passthrough mode.	HI_MPI_VO_SetDispBufLen	The display buffer can be saved.		For details about the restrictions on the passthrough mode, see chapter 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference*.	VIDEO LAYER STATUS 3: u32BufLen
Set the multi-region aggregation mode.	• HI_MPI_VO_SetVideoLayerAttr • HI_MPI_VO_SetChnDispPos	Some of the MMZ memory can be saved.	The VO module in aggregation mode does not support scaling.	Not supported by Hi3516C V500 or Hi3516E V200	VIDEO LAYER STATUS 1: ClustMode CHN BASIC INFO: DispX DispY
Use the VO auto magnification function.	HI_MPI_VO_SetVideoLayerAttr	If **stImageSize** is less than **stDispRect**, enable the auto magnification function for VO video layers to save memory and reduce bandwidth.		Supported only by Hi3559A V100	VIDEO LAYER STATUS 1: ImgW ImgH and DispW DispH
Use the buffer saving solution in the VO module for a single region.	HI_MPI_VO_SetModParam HI_MPI_VO_SetDispBufLen HI_MPI_VO_SetVtth HI_MPI_VO_SetVtth2	In single-region non-bypass mode, the minimum number of display buffers can be set to **2**. In single-region bypass mode, the number of buffers for recycling may be 1 less than the previous number.	The number of buffers used for display is reduced by 1.	Supported only by Hi3516C V500 and Hi3516E V200	MODULE PARAM: SaveBufMode VIDEO LAYER STATUS 3: u32BufLen

- Region management:

  For details, see chapter 8 "Region Management" in the *HiMPP V4.0 Media Processing Software Development Reference*. Run the following command to check the proc information: **cat /proc/umap/rgn**

  Most memory-saving solution for diagonal line drawing for Hi3516E V200 in all-online mode: Enable low-delay wrapping for VPSS-VENC. During encoding, use the ARGB 2BPP OSD for diagonal line drawing. Set the diagonal line in the OSD to be opaque while the rest transparent.

Measure	MPI and Parameter	Benefit	Impact	Note	Proc Information
Use the ARGB 2BPP Overlay region.	HI_MPI_RGN_Create: **enPixelFmt** = **PIXEL_FORMAT_ARGB_2BPP**	ARGB 2BPP requires 7/8 less buffer than ARGB 1555 and 15/16 less buffer than ARGB 888.	Only two colors are supported.	Supported only by Hi3516E V200 and Hi3516C V500 When OSDs overlap, an OSD with a higher layer level (larger **u32Layer**) overwrites an OSD with a lower layer level (smaller **u32Layer**).	REGION STATUS OF OVERLAY: PiFmt
Use the ARGB 2BPP OverlayEx region.	HI_MPI_RGN_Create: **enPixelFmt** = **PIXEL_FORMAT_ARGB_2BPP**	ARGB 2BPP requires 7/8 less buffer than ARGB 1555 and 15/16 less buffer than ARGB 888.	Only two colors are supported.	Supported by Hi3516E V200	REGION STATUS OF OVERLAYEX: PiFmt

  – HiFB

    For details, see the *HiFB Development Guide* and *HiFB API Reference*. Run the following command to check the proc information: **cat /proc/umap/hifb0**

    An SoC supporting multiple graphs layers may have **hifb1**, **hifb2**, and so on.

Measure	MPI and Parameter	Benefit	Impact	Note	Proc Information
Set an appropriate size for the physical display buffer at the graphics layer.	• Linux: video • Huawei LiteOS: HIFB_MODULE_PARAMS_S	Setting an appropriate size for the physical display buffer at the graphics layer according to the actual resolution avoids memory waste.	None	None	Mem size
Enable scaling for graphics layers.	FBIOPUT_SCREENSIZE	Some of the MMZ memory can be saved.	None	None	None

– Audio

For details, see chapter 9 "Audio" in the *HiMPP V4.0 Media Processing Software Development Reference*.

Measure	MPI and Parameter	Benefit	Impact	Note	Proc Information
Set the size of the audio frame for the audio input (AI) buffer by scenario.	HI_MPI_AI_SetPubAttr: u32FrmNum	Some of the OS memory can be saved.	Ensure that the buffer size is properly set. Otherwise, exceptions such as frame loss may occur.	-	cat /proc/umap/ai AI DEV ATTR: FrmNum
Set the size of the audio frame for the audio encoding (AENC) buffer by scenario.	HI_MPI_AENC_CreateChn: u32BufSize	Some of the OS memory can be saved.	Ensure that the buffer size is properly set. Otherwise, exceptions such as frame loss may occur.	-	cat /proc/umap/aenc AENC CHN ATTR: BufSize
Set the size of the audio frame for the audio decoding (ADEC) buffer by scenario.	HI_MPI_ADEC_CreateChn: u32BufSize	Some of the OS memory can be saved.	Ensure that the buffer size is properly set. Otherwise, exceptions such as frame loss may occur.	-	cat /proc/umap/adec ADEC CHN ATTR: BufSize
Set the size of the audio frame for the audio output (AO) buffer by scenario.	HI_MPI_AO_SetPubAttr: u32FrmNum	Some of the OS memory can be saved.	Ensure that the buffer size is properly set. Otherwise, exceptions such as frame loss may occur.	-	cat /proc/umap/ao AO DEV ATTR: FrmNum

- ISP module configuration:

  For details, see the *HiISP Development Reference*.

Measure	Module Parameter	Benefit	Impact	Note
Do not use the SpecAwb library. The SpecAwb memory does not need to be initialized.	For Hi3516E V200/Hi3516E V300, the SpecAwb algorithm memory is not allocated by default. For details about how to allocate the memory, see *HiISP Development Reference*.	Some of the OS memory can be saved.	Ensure that the SepcAwb library is not used.	Only Hi3516E V200 supports compilation of the macro switch. For Hi3516E V200/ Hi3516E V300, the SpecAwb memory is not allocated by default.

- Intelligent video engine (IVE)

  For details, see the *HiIVE API Reference*. Run the following command to check the proc information: **cat /proc/umap/ive**

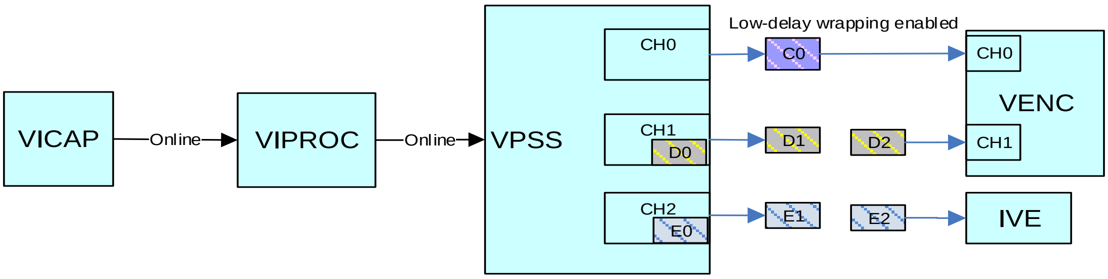
  Note: The following parameter configuration applies to Linux only. The corresponding parameter structure for Huawei LiteOS is IVE_MODULE_PARAMS_S, which is configured in the IVE_init function in the **sdk_init.c** file.

Measure	Module Parameter	Benefit	Impact	Note	Proc Information
Limit the number of IVE nodes supported.	max_node_num	The default value is **512**. The memory can be saved with the deduction of the parameter value.	The IVE performance is restricted if the number of nodes is too small.	-	MODULE PARAM: max_node_num

1.2.3 Common VB Allocation

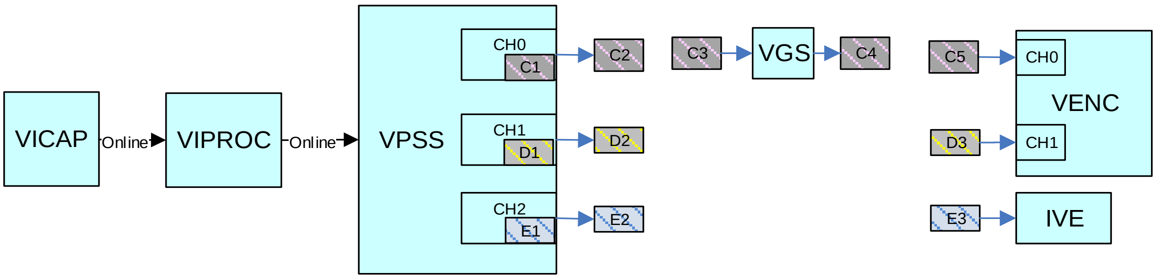
This section describes how to allocate the common VB by taking some scenarios of Hi3516EV200 as an example.

Figure 1-1 All-online scenario with low-delay wrapping enabled



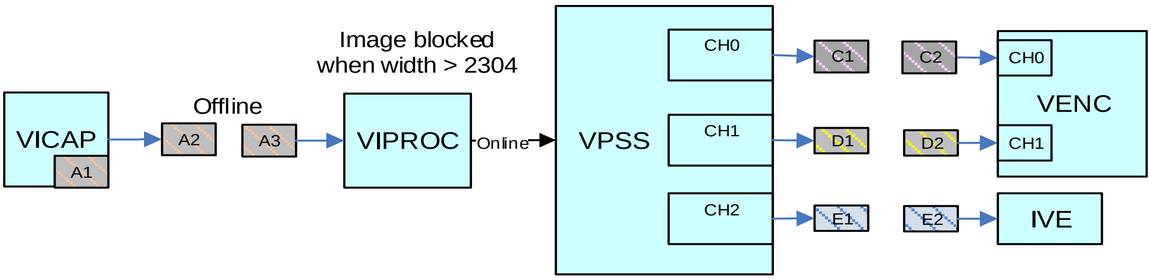
In the all-online scenario with low-delay wrapping enabled, one wrapped VB (the size is related to the line number and less than the frame size) is required by CH0 of the VPSS, and three wrapped VBs by CH1 and CH2 of the VPSS. Adjust the line number of VPSS Early_End by calling HI_MPI_VPSS_SetGrpFrameInterruptAttr. (You are advised to adjust the line number in descending order until frame loss is eliminated. If frame loss persists, it indicates that the Early_End mechanism is not applicable to this timing.) In ideal cases, CH1 and CH2 each require only two VBs. The number of VBs for CH2 also depends on the frame rate control. At a low frame rate, one more VB may be saved.

Figure 1-2 All-online scenario with rotation or LDC enabled in VPSS CH0



In the all-online scenario with rotation or LDC enabled in VPSS CH0, the number of main-stream VBs between the VPSS, VG, and VENC depends on the processing speeds of the three modules. The number of required VBs falls within the range of 3–5. In this case, you are advised to reduce the line number of the VPSS Early_End mechanism by 1 each time by calling HI_MPI_VPSS_SetGrpFrameInterruptAttr. In ideal cases, one VB can be reduced for VPSS CH0, that is, the number of required main-stream VBs falls within the range of 2–4.

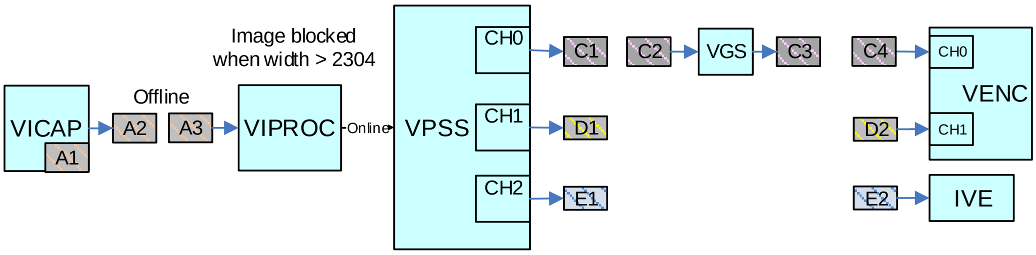
Figure 1-3 Block division scenario in offline slave mode with rotation and LDC disabled



When the image width is greater than 2304 pixels, the VICAP and VIPROC must be offline, but VIPROC-VPSS can be online. In this scenario, three raw VBs (that is, VBs for raw data, see A1, A2, and A3 in Figure 1-3) need to be allocated. CH0, CH1 and CH2 of the VPSS each require two VBs. Adjust the line number of VPSS Early_End by calling HI_MPI_VI_SetPipeFrameInterruptAttr. (You are advised to adjust the line number in descending order.) In ideal cases, only two raw VBs are required by the VICAP and VIPROC.

Enable low-delay for the VPSS CH0 by calling HI_MPI_VPSS_SetLowDelayAttr to speed up the VB rotation between the VPSS and VENC. In this case, only one VB needs to be allocated between VPSS CH0 and the VENC main stream.

Figure 1-4 Block division scenario in offline slave mode with rotation and LDC enabled in VPSS CH0



When the image width is greater than 2304 pixels, the VICAP and VIPROC modules must be offline, but VIPROC-VPSS can be online. In this scenario, three raw VBs (see A1, A2, and A3 in Figure 1-4) are required. In this case, you are advised to reduce the line number of the VPSS Early_End mechanism by 1 each time by calling HI_MPI_VI_SetPipeFrameInterruptAttr. In ideal cases, only two raw VBs are required by the VICAP and VIPROC. The number of main-stream VBs between the VPSS, VG, and VENC depends on the processing speeds of the three modules. The number of VBs falls within the range of 2–4.

1.2.4 CMA

For Hi3519A V100, Hi3516C V500, Hi3516D V300, and Hi3516A V300, the contiguous memory allocator (CMA) is enabled by default. After the CMA is enabled, the system reserves some memory by default. Only a part of the reserved memory may be used.

To save memory, you can use either of the following methods:

- Adjust the memory reserved by the system.

  Run the **cat /proc/meminfo** command to view the reserved CMA memory and memory usage. In the command output, **CmaTotal** indicates the CMA memory reserved by the system, and **CmaFree** indicates the remaining memory.

  CmaTotal: 16384 kB

  CmaFree: 16068 kB

  You can modify the kernel configuration to adjust the size of the reserved memory: **Device Drivers > Generic Driver Options > Size inMega Bytes**

  After modifying the kernel configuration, recompile the kernel.

- Disable the CMA directly.

  If the CMA function is not required, you can disable the CMA by modifying the kernel configuration: **Kernel Features > Contiguous Memory Allocator**

  After modifying the kernel configuration, recompile the kernel and **hi_osal.ko**.

1.3 Performance

1.3.1 What Do I Do If the CPU Usage Statistics Obtained by Running the Top Command Fluctuates?

[Symptom]

The CPU usage statistics obtained by running the **top** command is not accurate and fluctuates greatly in small service scenarios.

[Cause Analysis]

The default frequency of the Linux kernel in this version is 100 Hz. That is, the interval for calculating the CPU usage is 10 ms. The time granularity is coarse, which results in low statistical precision and large fluctuations in the statistics.

[Solution]

To improve the precision of the CPU usage statistics, you can change the kernel frequency to 1000 Hz.

1.3.2 Precautions for Binding Interrupts to Different CPUs

To bind interrupts to CPUs, you are advised to:

- Perform the CPU binding operation before the service is running. Do not dynamically switch the binding during service running.
- Bind multiple cores of the same module to the same CPU.
- Identify the module with many interrupts and bind it to another CPU. For example, if there are many network interrupts, separate the network module from media services.

1.3.3 Decoding Performance

[Symptom]

Hi3519AV100 cannot decode data at the full frame rate. In the playback scenario, VO **ChnRpt** occurs.

[Cause Analysis]

- The decoding thread has high requirements on real-time performance. If there are a large number of system threads and the decoding threads cannot be scheduled in a timely manner, the hardware utilization is affected.
- The output efficiency of tile output is higher than that of linear output.

[Solution]

- Change the value of **VDEC_SET_SCHEDULER** in **hi_usr.c** to **1** to increase the priority of the decoding thread.
- Call the HI_MPI_VDEC_SetChnParam MPI to change the video output format to the tile format.

1.4 Tailoring

1.4.1 How Do I Reduce the Application Size When the Static Libraries Are Used?

[Symptom]

The application uses only a small part of the functions in the **libmpi.a** library. However, the application needs to link to the associated library files such as **vqev2**. As a result, the application size is too large.

[Cause Analysis]

The application needs to link to all the functions defined in the MPI libraries by default when it links to the MPI libraries. Therefore, the application needs to use other libraries associated with the MPI libraries.

[Solution]

When the libraries of the HiMPP are generated, add the **-ffunction-sections** compilation option to **Makefile.param**. When the application links to the MPI libraries during compilation, add the **-Wl,-gc-sections** compilation option to **Makefile**. This deletes the functions that are not used and reduces the application size significantly.

1.5 How Do I Configure Pin Multiplexing, Clock Gating, and System Control?

In the single-Linux multi-core solution, the pin multiplexing (pinmux), pin drive capability, clock gating (clk), and system control (sysctl) are configured in **drv/interdrv/sysconfig.c**. You can modify the configurations as required and compile them into **sys_config.ko**. The configurations take effect after the .ko file is loaded.

In the dual-system (Linux+Huawei LiteOS) solution, the preceding items are configured in **mpp/out/liteos/single/init/sdk_initl.c**.

For Hi3516E V200: The pin multiplexing (pinmux), pin drive capability, clock gating (clk), and system control (sysctl) are configured in **drv/interdrv/sysconfig.c**. You can modify the configurations as required and compile them into **sys_config.ko**. The configuration takes effect after the .ko file is loaded.

1.6 Recompiling the .ko Drivers After Modifying the Kernel Options

[Symptom]

The customer needs to modify kernel options. After some kernel options are modified, the .ko drivers need to be recompiled.

[Analysis]

[Solution]

- The customer has recompiled the kernel after modifying the kernel options.
- The **KERNEL_ROOT** variable in the **Makefile.linux.param** file in **smp/a53_linux/mpp** has been modified so that the kernel path points to the recompiled kernel path.
- The **drv/extdrv**, **drv/interdrv**, **osal**, **mpp/component**, and **mpp/obj** sub-directories in the **smp/a53_linux** directory have been recompiled, respectively. The generated .ko

drivers are automatically copied to **smp/a53_linux/mpp/ko**. Note that the old .ko drivers will be overwritten.

- For details about how to compile the **osdrv/components/ipcm/ipcm/**, see *readme_en.txt*. The generated files after compilation include **hi_ipcm.ko** and **hi_virt-tty.ko**. You need to manually copy them to **smp/a53_linux/mpp/ko**.

2 MIPI

2.1 MIPI Frequency

How Do I Query the Relationship Between the Transfer Frequency of MIPI Lanes and VI Processing Frequency?

[Symptom]

When multiple MIPI lanes are used for data transfer, how do I query the relationship between the transfer frequency of MIPI lanes and VI processing frequency, and how do I calculate the maximum transfer frequency of each lane?

[Solution]

The Hi3516A receives data from multiple lanes over the MIPI, converts data into the internal timing, and transmits the timing to the VIU for processing. The total amount of data transferred by multiple lanes remains unchanged, as shown in the following equation:

VI_Freq x Pix_Width = Lane_Num x MIPI_Freq

Where **VI_Freq** indicates the frequency of the VI working clock, **Pix_Width** indicates the pixel bit width, **Lane_Num** indicates the number of lanes used for data transfer, and **MIPI_Freq** indicates the maximum RX frequency of each lane.

MIPI_Freq is calculated as follows: MIPI_Freq = VI_Freq x Pix_Width/Lane_Num. For example, if the frequency of the VI working clock is 250 MHz, the MIPI data is in RAW12 format, and four lanes are used for data transfer, **MIPI_Freq** is 750 (250 x 12/4).

That is, the maximum transfer frequency of each lane is 750 MHz.

2.2 Timing

Timing Requirements

When the MIPI receives the front-end timing, it is required that valid data in a row cannot be interrupted. Otherwise, the image becomes abnormal. Therefore, if you need to end the transmission of an image frame and immediately start the next frame, do not interrupt in the middle of valid data in a row.

hs_exit Adjustment

If the connected clock lane is in non-continuous mode (that is, the clock is disabled in the blanking interval) and the blanking interval of the sensor is small, a large value of hs_exit would affect the detection of the future burst by MIPI_RX, leading to a sensor interconnection failure (generally, the width and height are detected incorrectly and lines are lost). In this case, you can adjust the value of the cil_cyc_clk_hs_exit register to reduce the wait period of MIPI_RX detection. The minimum value of cil_cyc_clk_hs_exit is **14**.

Take Hi3516E V200 as an example. Modify bits 16–21 of **MIPI_FSMO_VALUE** in the **mipi_rx_hal.c** file.

```
#define MIPI_FSMO_VALUE (0x003f1d0c)
```

For example, if cil_cyc_clk_hs_exit is set to **14**, set the value to **0xe1d0c** in the macro definition.

3 VI

3.1 DIS

3.1.1 Implementing the ISP-DIS Functions

The image sensor processor (ISP) uses two-axis digital image stabilization (DIS). After the DIS function is enabled by calling HI_MPI_ISP_SetDISAttr, the VI module calculates the offset of the image jitter using the DIS algorithm. After the image is sent to the VPSS, the offset is cropped by using the VPSS group cropping function. If the image resolution is insufficient, the image can be amplified in the channel. In this way, the image after DIS is obtained.

3.2 WDR Function

3.2.1 Quadra WDR Precautions

You may use sensors that support Quadra WDR, such as Sony IMX294. In the WDR scenario, VI needs to write a long exposure frame to the DDR and then read the long exposure frame when a short exposure frame is transmitted for WDR combination. In the Quadra WDR timing, there is no line difference between transmissions of a long exposure frame and a short exposure frame. Therefore, the VI has a high latency requirement on the bus. If the latency of the bus does not meet the requirement, it may result in low bandwidth.

3.3 VI Timing Configuration

For transferring YUV data to the VI module, you need to configure pin multiplexing and clock selection (which is implemented through the transfer of the **-sensor** parameter in the **load** script. For details, see **sysconfig.c**). Then, configure the MIPI, VI device, and VI pipes. The differences are as follows.

3.3.1 BT.1120

3.3.1.1 MIPI Configuration

Determine whether the MIPI needs to be configured by referring to the *MIPI User Guide*. Hi3559A V100 is used as an example.

For channels 0 and 1, BT.1120 is independent of the MIPI, so no configuration is required. For channel 2, set the MIPI input mode to **INPUT_MODE_BT1120**, and **devno** to the required CMOS number.

```
combo_dev_attr_t MIPI_BT1120_ATTR =
{
    .devno = 2,
    .input_mode = INPUT_MODE_BT1120,
    .data_rate = DATA_RATE_X1,
    .img_rect = {0, 0, 1920, 1080},

    {
        .mipi_attr =
        {
            DATA_TYPE_RAW_12BIT,
            HI_MIPI_WDR_MODE_NONE,
            {0, 1, 2, 3, -1, -1, -1, -1}
        }
    }
};
```

3.3.1.2 VI Device Configuration

- Interface mode: **VI_MODE_BT1120_STANDARD**
- Mask settings: **au32ComponentMask[0] = 0xFF000000; au32ComponentMask[1] = 0x00FF0000**
- Scanning format: Only **VI_SCAN_PROGRESSIVE** is supported.
- UV sequence: **VI_DATA_SEQ_VUVU** or **VI_DATA_SEQ_UVUV** (determined based on the actual input timing)
- Data type: **VI_DATA_TYPE_YUV** (because YUV data is input through BT.1120)
- When Hi3516C V500 is configured with BT.1120, VI device 1 must be used. The pin multiplexing configuration is encapsulated in the **sysconfig.c** file. You can set the pin multiplexing using the script loading command (SoC specific). For details, see **load3516cv500** in the **ko** directory.

```
VI_DEV_ATTR_S DEV_BT1120_ATTR =
{
    VI_MODE_BT1120_STANDARD,
    VI_WORK_MODE_1Multiplex,
    {0xFF000000,    0x00FF0000},
    VI_SCAN_PROGRESSIVE,
    { -1, -1, -1, -1},
```



```
    VI_DATA_SEQ_VUVU,

    {
        VI_VSYNC_PULSE, VI_VSYNC_NEG_LOW, VI_HSYNC_VALID_SINGNAL,
VI_HSYNC_NEG_HIGH, VI_VSYNC_VALID_SINGAL, VI_VSYNC_VALID_NEG_HIGH,

        {
            0,            1920,        0,
            0,            1080,        0,
            0,            0,           0
        }
    },
    VI_DATA_TYPE_YUV,

    HI_FALSE,

    {1920 , 1080},

    {
        {
            {1920 , 1080},
        },
        {
            VI_REPHASE_MODE_NONE,
            VI_REPHASE_MODE_NONE
        }
    },

    {
        WDR_MODE_NONE,
        1080
    },

    DATA_RATE_X1
};
```

3.3.1.3 VI Pipe Configuration

- **bIspBypass** of the VI pipe is set to **HI_TRUE**.
- The pixel format of the VI pipe is set to **PIXEL_FORMAT_YVU_SEMIPLANAR_422**.
- **nBitWidth** of the VI pipe is set to **DATA_BITWIDTH_8**.

```
VI_PIPE_ATTR_S PIPE_BT1120_ATTR =
{
```

```
    VI_PIPE_BYPASS_NONE, HI_FALSE, HI_TRUE,
    1920, 1080,
    PIXEL_FORMAT_YVU_SEMIPLANAR_422,
    COMPRESS_MODE_NONE,
    DATA_BITWIDTH_8,
    HI_FALSE,
    {
        PIXEL_FORMAT_YVU_SEMIPLANAR_422,
        DATA_BITWIDTH_8,
        VI_NR_REF_FROM_RFR,
        COMPRESS_MODE_NONE
    },
    HI_FALSE,
    {-1, -1}
};
```

3.3.2 BT.656

3.3.2.1 MIPI Configuration

Determine whether the MIPI needs to be configured by referring to the *MIPI User Guide*. Hi3559A V100 is used as an example.

For channels 0 and 1, BT.656 is independent of the MIPI, so no configuration is required. For channel 2, set the MIPI input mode to **INPUT_MODE_BT656**, and **devno** to the required CMOS number.

```
combo_dev_attr_t MIPI_BT656_ATTR =
{
    .devno = 2,
    .input_mode = INPUT_MODE_BT656,
    .data_rate = DATA_RATE_X1,
    .img_rect = {0, 0, 720, 576},

    {
        .mipi_attr =
        {
            DATA_TYPE_RAW_12BIT,
            HI_MIPI_WDR_MODE_NONE,
            {0, 1, 2, 3, -1, -1, -1, -1}
        }
    }
};
```

3.3.2.2 VI Device Configuration

- Interface mode: **VI_MODE_BT656**
- Mask setting: **au32ComponentMask[0] = 0xFF000000**
- Scanning format: Only **VI_SCAN_PROGRESSIVE** is supported.
- UV sequence: **VI_DATA_SEQ_UYVY**, **VI_DATA_SEQ_VYUY**, **VI_DATA_SEQ_YUYV**, or **VI_DATA_SEQ_YVYU** is determined based on the actual input timing.
- Data type: **VI_DATA_TYPE_YUV** is used because YUV data is input in BT.656.

```
VI_DEV_ATTR_S DEV_BT656_ATTR =
{
    VI_MODE_BT656,
    VI_WORK_MODE_1Multiplex,
    {0xFF000000,    0x00FF0000},
    VI_SCAN_PROGRESSIVE,
    { -1, -1, -1, -1},
    VI_DATA_SEQ_YUYV,

    {
        VI_VSYNC_PULSE, VI_VSYNC_NEG_LOW, VI_HSYNC_VALID_SINGNAL,
VI_HSYNC_NEG_HIGH, VI_VSYNC_VALID_SINGAL, VI_VSYNC_VALID_NEG_HIGH,

        {
            0,            720,        0,
            0,            576,        0,
            0,            0,           0
        }
    },
    VI_DATA_TYPE_YUV,

    HI_FALSE,

    {720, 576},

    {
        {
            {720, 576},
        },
        {
            VI_REPHASE_MODE_NONE,
            VI_REPHASE_MODE_NONE
        }
    },
```



```
    {
        WDR_MODE_NONE,
        576
    },

    DATA_RATE_X1
};
```

3.3.2.3 VI Pipe Configuration

- **bIspBypass** of the VI pipe is set to **HI_TRUE**.
- The pixel format of the VI pipe is set to **PIXEL_FORMAT_YVU_SEMIPLANAR_422**.
- **nBitWidth** of the VI pipe is set to **DATA_BITWIDTH_8**.

```
VI_PIPE_ATTR_S PIPE_BT656_ATTR =
{
    VI_PIPE_BYPASS_NONE, HI_FALSE, HI_TRUE,
    720, 576,
    PIXEL_FORMAT_YVU_SEMIPLANAR_422,
    COMPRESS_MODE_NONE,
    DATA_BITWIDTH_8,
    HI_FALSE,
    {
        PIXEL_FORMAT_YVU_SEMIPLANAR_422,
        DATA_BITWIDTH_8,
        VI_NR_REF_FROM_RFR,
        COMPRESS_MODE_NONE
    },
    HI_FALSE,
    {-1, -1}
};
```

3.3.3 BT.601

3.3.3.1 MIPI Configuration

Determine whether the MIPI needs to be configured by referring to the *MIPI User Guide*. Hi3559A V100 is used as an example.

For channels 0 and 1, BT. 601 is independent of the MIPI, so no configuration is required. For channel 2, set the MIPI input mode to **INPUT_MODE_BT601**, and **devno** to the required CMOS number.

```
combo_dev_attr_t MIPI_BT601_ATTR =
{
    .devno = 2,
    .input_mode = INPUT_MODE_BT601,
```



```
    .data_rate = DATA_RATE_X1,
    .img_rect = {0, 0, 720, 576},

    {
        .mipi_attr =
        {
            DATA_TYPE_RAW_12BIT,
            HI_MIPI_WDR_MODE_NONE,
            {0, 1, 2, 3, -1, -1, -1, -1}
        }
    }
};
```

3.3.3.2 VI Device Configuration

- Interface mode: **VI_MODE_BT601**
- Mask setting: **au32ComponentMask[0] = 0xFF000000**
- Scanning format: Only **VI_SCAN_PROGRESSIVE** is supported.
- Timing parameters: For details about the timing parameter configuration, see the description of **VI_SYNC_CFG_S** in Chapter "VI."
- UV sequence: **VI_DATA_SEQ_VUVU** or **VI_DATA_SEQ_UVUV** is determined based on the actual input timing.
- Data type: **VI_DATA_TYPE_YUV** is used because YUV data is input in BT.601.

```
VI_DEV_ATTR_S DEV_BT601_ATTR =
{
    VI_MODE_BT601,
    VI_WORK_MODE_1Multiplex,
    {0xFF000000,    0x00FF0000},
    VI_SCAN_PROGRESSIVE,
    { -1, -1, -1, -1},
    VI_DATA_SEQ_YUYV,

    {
        VI_VSYNC_PULSE, VI_VSYNC_NEG_LOW, VI_HSYNC_VALID_SINGNAL,
VI_HSYNC_NEG_HIGH, VI_VSYNC_VALID_SINGAL, VI_VSYNC_VALID_NEG_HIGH,

        {
            0,            720,        0,
            0,            576,        0,
            0,            0,            0
        }
    },
    VI_DATA_TYPE_YUV,

    HI_FALSE,
```

```
    {720 , 576},

    {
        {
            {720 , 576},
        },
        {
            VI_REPHASE_MODE_NONE,
            VI_REPHASE_MODE_NONE
        }
    },

    {
        WDR_MODE_NONE,
        576
    },

    DATA_RATE_X1
};
```

3.3.3.3 VI Pipe Configuration

- **bIspBypass** of the VI pipe is set to **HI_TRUE**.
- The pixel format of the VI pipe is set to **PIXEL_FORMAT_YVU_SEMIPLANAR_422**.
- **nBitWidth** of the VI pipe is set to **DATA_BITWIDTH_8**.

```
VI_PIPE_ATTR_S PIPE_BT1120_ATTR =
{
    VI_PIPE_BYPASS_NONE, HI_FALSE, HI_TRUE,
    720, 576,
    PIXEL_FORMAT_YVU_SEMIPLANAR_422,
    COMPRESS_MODE_NONE,
    DATA_BITWIDTH_8,
    HI_FALSE,
    {
        PIXEL_FORMAT_YVU_SEMIPLANAR_422,
        DATA_BITWIDTH_8,
        VI_NR_REF_FROM_RFR,
        COMPRESS_MODE_NONE
    },
    HI_FALSE,
    {-1, -1}
};
```

3.3.4 MIPI_YUV

3.3.4.1 MIPI Configuration

- The MIPI input mode is set to INPUT_MODE_MIPI.
- The input_data_type of the MIPI attribute is set based on the input data type. If YUV422 data is input, the data type is set to DATA_TYPE_YUV422_8BIT. If legacy YUV420 data is input, the data type is set to DATA_TYPE_YUV420_8BIT_LEGACY. If normal YUV420 data is input, the data type is set to DATA_TYPE_YUV420_8BIT_NORMAL.

```
ombo_dev_attr_t MIPI_YUV422_ATTR =
{
    .devno = 0,
    .input_mode = INPUT_MODE_MIPI,
    .data_rate = DATA_RATE_X1,
    .img_rect = {0, 0, 1920, 1080},

    {
        .mipi_attr =
        {
            DATA_TYPE_YUV422_8BIT,
            HI_MIPI_WDR_MODE_NONE,
            {0, 1, 2, 3, -1, -1, -1, -1}
        }
    }
};
```

3.3.4.2 VI Device Configuration

- Interface mode: It is set base on the input data type. If YUV422 data is input, the data type is set to VI_MODE_MIPI_YUV422. If legacy YUV420 data is input, the data type is set to VI_MODE_MIPI_YUV420_LEGACY. If normal YUV420 data is input, the data type is set to VI_MODE_MIPI_YUV420_NORMAL.
- Mask settings: au32ComponentMask[0] = 0xFF000000; au32ComponentMask[1] = 0x00FF0000
- Scanning format: Only VI_SCAN_PROGRESSIVE is supported.
- UV sequence: VI_DATA_SEQ_VUVU or VI_DATA_SEQ_UVUV is determined based on the actual input timing.
- Data type: VI_DATA_TYPE_YUV is used because YUV data is input in MIPI.

```
VI_DEV_ATTR_S DEV_MIPI_YUV422_ATTR =
{
    VI_MODE_MIPI_YUV422,
    VI_WORK_MODE_1Multiplex,
    {0xFF000000,    0x00FF0000},
    VI_SCAN_PROGRESSIVE,
    { -1, -1, -1, -1},
    VI_DATA_SEQ_VUVU,
```



```
    {
        VI_VSYNC_PULSE, VI_VSYNC_NEG_LOW, VI_HSYNC_VALID_SINGNAL,
VI_HSYNC_NEG_HIGH, VI_VSYNC_VALID_SINGAL, VI_VSYNC_VALID_NEG_HIGH,

        {
            0,              1920,         0,
            0,              1080,         0,
            0,              0,            0
        }
    },
    VI_DATA_TYPE_YUV,

    HI_FALSE,

    {1920 , 1080},

    {
        {
            {1920 , 1080},
        },
        {
            VI_REPHASE_MODE_NONE,
            VI_REPHASE_MODE_NONE
        }
    },

    {
        WDR_MODE_NONE,
        1080
    },

    DATA_RATE_X1
};
```

3.3.4.3 VI Pipe Configuration

- **bIspBypass** of the VI pipe is set to **HI_TRUE**.
- The pixel format of the VI pipe is set to **PIXEL_FORMAT_YVU_SEMIPLANAR_422**.
- **nBitWidth** of the VI pipe is set to **DATA_BITWIDTH_8**.

```
VI_PIPE_ATTR_S PIPE_BT1120_ATTR =
{
    VI_PIPE_BYPASS_NONE, HI_FALSE, HI_TRUE,
```



```
    1920, 1080,
    PIXEL_FORMAT_YVU_SEMIPLANAR_422,
    COMPRESS_MODE_NONE,
    DATA_BITWIDTH_8,
    HI_FALSE,
    {
        PIXEL_FORMAT_YVU_SEMIPLANAR_422,
        DATA_BITWIDTH_8,
        VI_NR_REF_FROM_RFR,
        COMPRESS_MODE_NONE
    },
    HI_FALSE,
    {-1, -1}
};
```

3.3.5 LVDS

3.3.5.1 LVDS Configuration

- The MIPI input mode is set to INPUT_MODE_LVDS, INPUT_MODE_SUBLVDS, or INPUT_MODE_HISPI.
- The input_data_type of the LVDS attribute is set based on the input data type.
- The input data rate must be the same as that of the VI device. Here, MIPI_DATA_RATE_X2 is used as an example.

```
combo_dev_attr_t LVDS_12BIT_ATTR =
{
    .devno                  = 0,
    .input_mode              = INPUT_MODE_LVDS,
    .data_rate               = MIPI_DATA_RATE_X2,
    .img_rect                = {0, 0, 7840, 4320},
    .lvds_attr               =
    {
        .input_data_type      = DATA_TYPE_RAW_12BIT,
        .wdr_mode            = HI_WDR_MODE_NONE,
        .sync_mode           = LVDS_SYNC_MODE_SAV,
        .vsync_attr          = {LVDS_VSYNC_NORMAL, 0, 0},
        .fid_attr            = {LVDS_FID_NONE, HI_TRUE},
        .data_endian         = LVDS_ENDIAN_LITTLE,
        .sync_code_endian     = LVDS_ENDIAN_LITTLE,
        .lane_id             = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15},
        .sync_code           =
        {
            /* each vc has 4 params, sync_code[i]:
               sync_mode is SYNC_MODE_SOF: SOF, EOF, SOL, EOL
```

```
             sync_mode is SYNC_MODE_SAV: invalid sav, invalid eav, valid
sav, valid eav  */
          {   {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00}
          },

          {   {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00}
          },

          {   {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00}
          },

          {   {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00}
          },

          {   {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00}
          },

          {   {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00}
          },

          {   {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00},
              {0x200, 0x300, 0x400, 0xC00}
          },
```

```
        {   {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00}
        },

        {   {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00}
        },

        {   {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00}
        },

        {   {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00}
        },

        {   {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00}
        },

        {   {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00}
        },

        {   {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00}
        },

        {   {0x200, 0x300, 0x400, 0xC00},
            {0x200, 0x300, 0x400, 0xC00},
```



```
                {0x200, 0x300, 0x400, 0xC00},
                {0x200, 0x300, 0x400, 0xC00}
            },

            {   {0x200, 0x300, 0x400, 0xC00},
                {0x200, 0x300, 0x400, 0xC00},
                {0x200, 0x300, 0x400, 0xC00},
                {0x200, 0x300, 0x400, 0xC00}
            },
         }
    }
};
```

3.3.5.2 VI Device Configuration

- Interface mode: It is set to VI_MODE_LVDS or VI_MODE_HISPI based on the input interface type.
- Mask setting: au32ComponentMask[0] = 0xFFF00000

```
VI_DEV_ATTR_S DEV_LVDS_RAW12_ATTR =
{
    VI_MODE_LVDS,
    VI_WORK_MODE_1Multiplex,
    {0xFFF00000,    0x0},
    VI_SCAN_PROGRESSIVE,
    {-1, -1, -1, -1},
    VI_DATA_SEQ_YUYV,

    {
        VI_VSYNC_PULSE, VI_VSYNC_NEG_LOW,
VI_HSYNC_VALID_SINGNAL,VI_HSYNC_NEG_HIGH,VI_VSYNC_VALID_SINGAL,VI_VSYNC_V
ALID_NEG_HIGH,
    {0,            1280,       0,
     0,            720,        0,
     0,            0,            0}
    },
    VI_DATA_TYPE_RGB,
    HI_FALSE,
    {7680 , 4320},
    {
        {
            {7680 , 4320},
        },
        {
            VI_REPHASE_MODE_NONE,
            VI_REPHASE_MODE_NONE
```

```
        }
    },
    {
        WDR_MODE_NONE,
        4320
    },
    DATA_RATE_X2
};
```

3.3.5.3 VI Pipe Configuration

- **bIspBypass** of the VI pipe is set to **HI_FALSE**.
- The pixel format of the VI pipe is set to **PIXEL_FORMAT_RGB_BAYER_12BPP**.
- **nBitWidth** of the VI pipe is set to **DATA_BITWIDTH_12**.

```
VI_PIPE_ATTR_S PIPE_LVDS_ATTR =
{
    VI_PIPE_BYPASS_NONE, HI_FALSE, HI_FALSE,
    7680, 4320,
    PIXEL_FORMAT_RGB_BAYER_12BPP,
    COMPRESS_MODE_LINE,
    DATA_BITWIDTH_12,
    HI_TRUE,
    {
        PIXEL_FORMAT_YVU_SEMIPLANAR_420,
        DATA_BITWIDTH_10,
        VI_NR_REF_FROM_RFR,
        COMPRESS_MODE_NONE
    },
    HI_FALSE,
    {-1, -1}
};
```

3.4 VI Interrupt Types

The collection preparation and sending of the VI module are performed in interrupts. For different interrupt types, the time for collecting and sending data, the occupied DDR resources, and the usage restrictions are different.

- In VICAP-VIPROC-VPSS online mode, call HI_MPI_VPSS_SetGrpFrameInterruptAttr to set the interrupt type.
- In VICAP and VIPROC offline mode, or VICAP-VIPROC online while VIPROC and VPSS offline mode, call HI_MPI_VI_SetPipeFrameInterruptAttr to set the interrupt type.

3.4.1 FRAME_INTERRUPT_START

Interrupt handling: The collection preparation and sending are completed in the SOF interrupt.

DDR usage: During the collection, the VB occupation period is fixed at two frames.

Advantages: The frame collection is complete and correct.

Disadvantages: The VB occupation period is long.

Debugging method: None

3.4.2 FRAME_INTERRUPT_EARLY

Interrupt handling: The collection preparation and sending are completed in the early report interrupt.

DDR usage: During the collection, the VB occupation period is fixed at one frame.

Advantages: The VB occupation period is very short. This interrupt type helps save the memory when the memory is insufficient.

Disadvantages:

- A frame is sent to the BE for processing after being collected until the line indicated by **u32EarlyLine**. Therefore, if the value of **u32EarlyLine** is small, problems may occur.
- If the processing speed of the BE module equals the collection speed, errors such as artifacts may occur.
- Line compression and frame compression are not used because they may cause exceptions in the VIPROC.
- The snapshot channel cannot be used.
- When **u32EarlyLine** is set to a small value, image flickers occur in the ISP.

Instructions: Adjust the value of **u32EarlyLine** to ensure that the downstream module keep pace with the frame collection. Otherwise, exceptions such as erratic display may occur. A larger value of **u32EarlyLine** indicates a lower probability that the processing of the downstream module overtakes frame collection. However, if the value of **u32EarlyLine** is too large, the buffer saving feature may be compromised.

For example, if the VI module is offline, the VIPROC may report an error interrupt when raw data of line compression or frame compression is collected. **u32EarlyLine** depends on the timing, frame rate, and BE module performance. Assume that the time for collecting a frame by the VI module is **T1** (that is, from the SOF to the EOF, excluding the blanking interval), the image height is **H**, and the time for processing a frame by the BE module (such as VIPROC, VPSS, VENC) is **T2**, then:

(T1/H) x (H – u32EarlyLine) ≤ T2

3.4.3 FRAME_INTERRUPT_EARLY_END

Interrupt handling: The collection preparation is performed in the early report interrupt, while the frame is sent in the EOF interrupt.

DDR usage: During the collection, the VB occupation period is from one frame to two frames.

Advantages:

- The frame collection is complete and correct.

- In a low frame rate, data can be collected in a timely manner when it is in a discontinuous sequence.

Disadvantages:

- The VB occupation period is long. When **u32EarlyLine** is set to the image height minus 1, the VB occupation period is fixed at one frame.
- A larger number of interrupts cause increasing CPU usage.
- If the blanking interval of the sensor output timing is small, image flickers may occur during ISP processing.

Debugging method: When the back blanking interval is large, it is strongly recommended that **u32EarlyLine** be set to the image height minus 1. If the back blanking interval is very small so that the EOF interrupt overlaps with the SOF interrupt, adjust **u32EarlyLine** to a value smaller than the image height until no VI frame is lost.

3.5 PAL/NTSC Standard Switchover

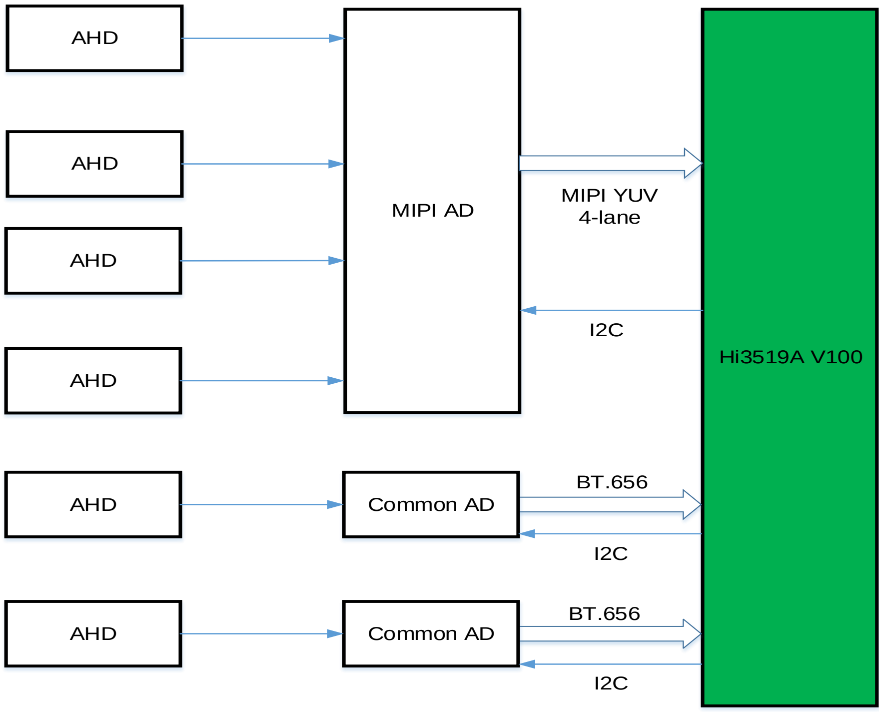
Setting an ISP Interface to Adjust the Output Frame Rate

During service running, you only need to call the HI_MPI_ISP_SetPubAttr interface to set **f32FrameRate** in the public attribute. For the NTSC standard, set **f32FrameRate** to **30**. For the PAL standard, set **f32FrameRate** to **25**.

3.6 Hi3519A V100 Connecting to 6-Lane YUV

3.6.1 4-to-1 MIPI AD + 2 x Separate BT.656

Figure 3-1 6-lane YUV solution



- The MIPI AD connects to four AHD lanes in the front end, with YUV data output through VI pipes 2–5.
- The two BT.656 lanes receive BT656_PACKED_YUV data from the VI, with data output through VI pipes 0–1 in the 8-bit raw format and then converted into the SP422 format using the IVE module in DMA mode. (For details about the format conversion, see the sample provided for the IVE module). The configuration is as follows:
  - Figure 3-2 shows the MIPI configuration.

Figure 3-2 MIPI configuration

```
combo_dev_attr_t BT656_2M_30FPS_ATTR =
{
    .devno = 0,
    .input_mode = INPUT_MODE_BT656,
    .data_rate = MIPI_DATA_RATE_X1,
    .img_rect = {0, 0, 1920, 1080},
};
```

  - Figure 3-3 shows the VI device attribute configuration.

Figure 3-3 VI device attribute configuration

```
VI_DEV_ATTR_S DEV_ATTR_BT656_BASE =
{
    VI_MODE_BT656_PACKED_YUV,
    VI_WORK_MODE_1Multiplex,
    {0xFF000000,    0x0},
    VI_SCAN_PROGRESSIVE,
    {-1, -1, -1, -1},
    VI_DATA_SEQ_YUYV,

    {
    /*port_vsync   port_vsync_neg     port_hsync        port_hsync_neg        */
    VI_VSYNC_PULSE, VI_VSYNC_NEG_LOW, VI_HSYNC_VALID_SINGNAL,VI_HSYNC_NEG_HIGH,VI_VSYNC_VALID_SINGAL,VI_VSYNC_VALID_NEG_HIGH,

    /*hsync_hfb    hsync_act    hsync_hhb*/
    {0,            1280,        0,
    /*vsync0_vhb vsync0_act vsync0_hhb*/
     0,            720,        0,
    /*vsync1_vhb vsync1_act vsync1_hhb*/
     0,            0,            0}
    },
    VI_DATA_TYPE_YUV,
    HI_FALSE,
    {1920 , 1080},
    {
        {
            {1920 , 1080},
        },
        {
            VI_REPHASE_MODE_NONE,
            VI_REPHASE_MODE_NONE
        }
    },
    {
        WDR_MODE_NONE,
        1080
    },
    DATA_RATE_X1
};
```

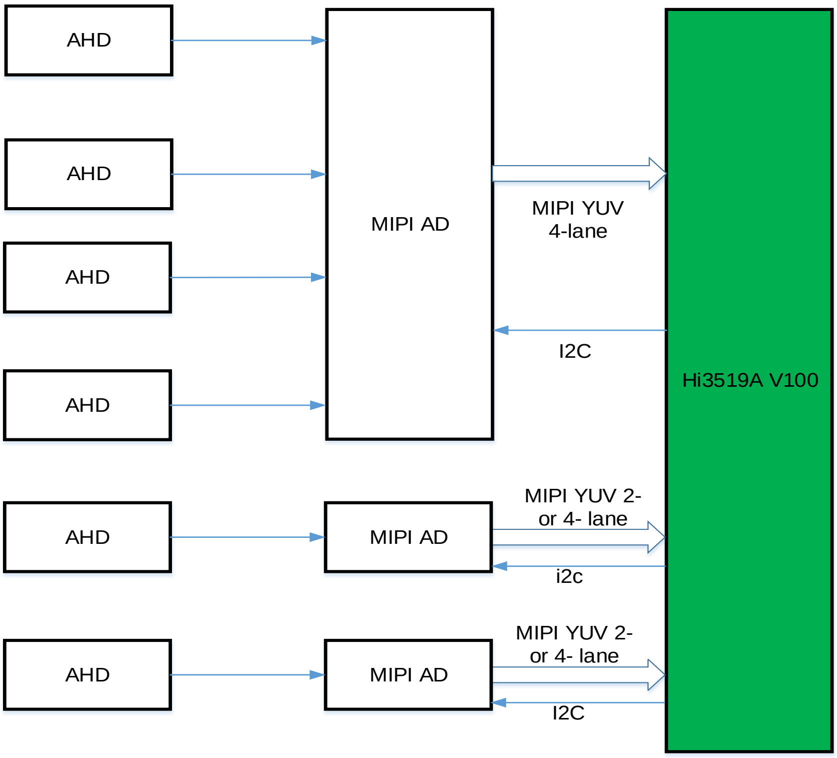
– Figure 3-4 shows the VI pipe attribute configuration. (Note that the width must be twice the actual width.)

Figure 3-4 VI pipe attribute configuration

```
static VI_PIPE_ATTR_S PIPE_ATTR_1920x1080_RAW8_420_3DNR_RFR =
{
    VI_PIPE_BYPASS_BE, HI_TRUE, HI_FALSE,
    1920*2, 1080,
    PIXEL_FORMAT_RGB_BAYER_8BPP,
    COMPRESS_MODE_NONE,
    DATA_BITWIDTH_8,
    HI_TRUE,
    {
        PIXEL_FORMAT_YVU_SEMIPLANAR_420,
        DATA_BITWIDTH_8,
        VI_NR_REF_FROM_RFR,
        COMPRESS_MODE_NONE
    },
    HI_FALSE,
    {-1, -1}
};
```

3.6.2 4-to-1 MIPI AD + 2 x Separate MIPI AD

Figure 3-5 6-lane YUV solution



- The MIPI AD connects to four AHD lanes in the front end, with YUV data output through VI pipes 2–5.
- The other two separate MIPI lanes receive YUV422 packed data from the VI, with data output through VI pipes 0–1 in the 16-bit raw format and then converted into the SP422 format using the IVE module in DMA mode. (For details about the format conversion, see the sample provided for the IVE module). The configuration is as follows:
  - Figure 3-6 shows the MIPI configuration.

Figure 3-6 MIPI configuration

```
combo_dev_attr_t MIPI_4lane_CHN0_2M_SP420_ATTR =
{
    .devno = 0,
    .input_mode = INPUT_MODE_MIPI,
    .data_rate = MIPI_DATA_RATE_X1,
    .img_rect = {0, 0, 1920, 1080},

    {
        .mipi_attr =
        {
            DATA_TYPE_YUV422_PACKED,
            HI_MIPI_WDR_MODE_NONE,
            {0, 1, -1, -1, -1, -1, -1, -1},
        }
    }
};
```

– Figure 3-7 shows the VI device attribute configuration.

Figure 3-7 VI device attribute configuration

```
VI_DEV_ATTR_S DEV_ATTR_MIPI_SP422_2M_BASE =
{
    VI_MODE_MIPI,
    VI_WORK_MODE_1Multiplex,
    {0xFFFF0000,    0x000000},
    VI_SCAN_PROGRESSIVE,
    {-1, -1, -1, -1},
    VI_DATA_SEQ_UYVY,

    {
    /*port_vsync   port_vsync_neg     port_hsync        port_hsync_neg        */
    VI_VSYNC_PULSE, VI_VSYNC_NEG_LOW, VI_HSYNC_VALID_SINGNAL,VI_HSYNC_NEG_HIGH,VI_VSYNC_VALID_SINGAL,VI_VSYNC_VALID_NEG_HIGH,

    /*hsync_hfb    hsync_act    hsync_hhb*/
    {0,            1280,        0,
    /*vsync0_vhb vsync0_act vsync0_hhb*/
     0,            720,        0,
    /*vsync1_vhb vsync1_act vsync1_hhb*/
     0,            0,            0}
    },
    VI_DATA_TYPE_YUV,
    HI_FALSE,
    {1920 , 1080},
    {
        {
            {1920 , 1080},
        },
        {
            VI_REPHASE_MODE_NONE,
            VI_REPHASE_MODE_NONE
        }
    },
    {
        WDR_MODE_NONE,
        1080
    },
    DATA_RATE_X1
};
```

– Figure 3-8 shows the PIPE attribute configuration.

Figure 3-8 VI pipe attribute configuration

```
static VI_PIPE_ATTR_S PIPE_ATTR_1920x1080_SP420_3DNR_RFR =
{
    VI_PIPE_BYPASS_BE, HI_FALSE, HI_FALSE,
    1920, 1080,
    PIXEL_FORMAT_RGB_BAYER_16BPP,
    COMPRESS_MODE_NONE,
    DATA_BITWIDTH_8,
    HI_FALSE,
    {
        PIXEL_FORMAT_YVU_SEMIPLANAR_420,
        DATA_BITWIDTH_8,
        VI_NR_REF_FROM_RFR,
        COMPRESS_MODE_NONE
    },
    HI_FALSE,
    {-1, -1}
};
```

4 Fisheye

4.1 Fisheye Correction

4.1.1 How Do I Implement Fisheye Correction of More than Four Cells by Using Hi3559A V100/Hi3556A V100?

[Symptom]

In the system binding channel, the VI/VPSS interface HI_MPI_VI_SetExtChnFisheye/HI_MPI_VPSS_SetExtChnFisheye allows you to combine only two to four cells (at most four cells).

[Cause Analysis]

The implementation of the VI/VPSS interface for combining more than four cells complicates the system. In addition, the GDC performance of Hi3559A V100/Hi3556A V100 is insufficient.

[Solution]

Obtain pictures from the VI/VPSS module, implement fisheye correction by calling HI_MPI_GDC_AddCorrectionTask, and then transfer the pictures back to the system channel (VI/VPSS/VO module). The configuration of the LMF parameters can also be implemented by calling the fisheye interface HI_MPI_GDC_SetConfig.

NOTE

For details, see the fourth fisheye sample. The GDC performance may be insufficient.

5 LDC

5.1 Distortion Correction

5.1.1 Comparison Between VI and VPSS LDC

In VI/VPSS offline scenario, the VI interface and LDC interfaces of the VPSS both support lens distortion correction (LDC). Because LDC changes the noise form of images, you are advised to process images using LDC (that is, VPSS LDC) and then 3DNR to achieve better image effects. If the VPSS LDC is used, for example, the VPSS has multiple channels in the service scenario, the VGS/GDC is called to perform LDC in each channel. Compared with the VI LDC, the VGS/GDC performance and memory resources are adversely affected in the VPSS LDC. If you want to obtain better images and the performance and memory resources are sufficient, you are advised to use the VPSS for LDC. In other scenarios, you are advised to use the VI for LDC.

In the VI/VPSS online scenario, only LDC of the VPSS can be enabled. LDC of the VI is unavailable.

6 Audio

6.1 How Do I Play the Audio Streams Encoded by HiSilicon on the PC?

6.1.1 How Do I Play G711/G726/ADPCM Audio Streams Encoded by HiSilicon on the PC?

[Symptom]

The G711/G726/ADPCM audio streams encoded by HiSilicon cannot be played directly by using software on the PC.

[Cause Analysis]

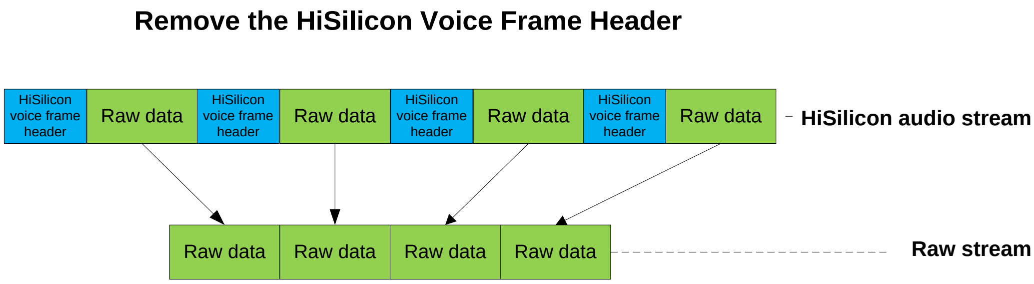
A HiSilicon voice frame header is added at the beginning of each frame in the audio streams encoded by HiSilicon.

For details, see section 9.2.2.3 "Structure of the HiSilicon Voice Frame" in the *HiMPP V4.0 Media Processing Software Development Reference.*

[Solution]

Remove the HiSilicon voice frame header at the beginning of each frame, add the WAV header to the frames in the raw stream, and play the streams using software on the PC. Figure 6-1 shows how to remove the HiSilicon voice frame header.

Figure 6-1 Remove the HiSilicon voice frame header



The reference code for removing the HiSilicon voice frame header is as follows:

```
int HisiVoiceGetRawStream(short *Hisivoicedata, short *outdata, int
hisisamplelen)
{
    int len = 0, outlen = 0;
    short *copyHisidata, *copyoutdata;
    int copysamplelen = 0;
    copysamplelen = hisisamplelen;
    copyHisidata = Hisivoicedata;
    copyoutdata = outdata;
    while(copysamplelen > 2)
    {
        len = copyHisidata[1]&0x00ff;
        copysamplelen -= 2;
        copyHisidata += 2;
        if(copysamplelen < len)
        {
            break;
        }
        memcpy(copyoutdata, copyHisidata, len * sizeof(short));
        copyoutdata += len;
        copyHisidata += len;
        copysamplelen -= len;
        outlen += len;
    }
    return outlen;
}
```

NOTE

- The audio streams in ADPCM_DVI4 or ADPCM_ORG_DVI4 format are used for network transfer over the Real-time Transport Protocol (RTP) and cannot be played by the client programs on the PC. For details, see the RFC35551 standard.
- The method of adding the WAV header is not provided in this document. You can add the WAV header by following the WAV header standard. For details, see the reference links https://msdn.microsoft.com/en-us/library/dd390970(v=vs.85).aspx and http://www.moon-soft.com/program/FORMAT/windows/wavec.htm.

6.2 How Do I Play Standard Audio Streams on HiSilicon Chips?

6.2.1 How Do I Play Standard G711/G726/ADPCM Audio Streams on HiSilicon Chips?

[Symptom]

The standard G711/G726/ADPCM audio streams cannot be played directly on HiSilicon chips.

[Cause Analysis]

To ensure that the previous-generation chips are compatible, the audio streams can be played on HiSilicon chips only after the HiSilicon voice frame header is added at the beginning of each frame in the raw audio streams.

[Solution]

To play G711/G726/ADPCM audio streams on HiSilicon chips, obtain the raw stream data, add the HiSilicon voice frame header at the beginning of each frame based on the frame data length **PerSampleLen**.

1. Obtain the raw stream data. Remove the WAV header if the WAV header is added to the frame.
2. Obtain the data length of each frame (**PersampleLen**, a short number).

Table 6-1 Data length of each frame

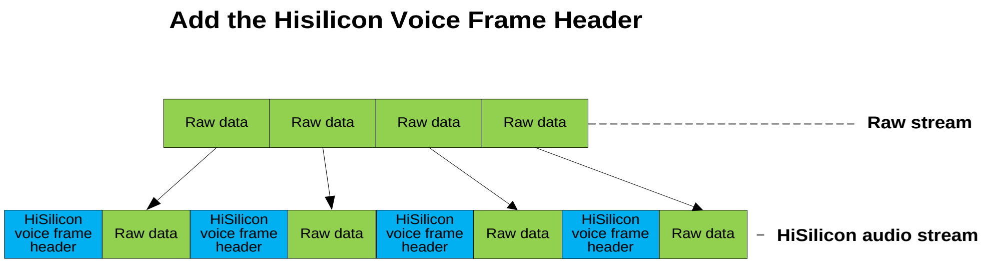
Encoding Format	Data Length of Each Frame	Remarks
G711	*N* x 40	*N* is a positive integer ranging from 1 to 5.
G726 (16 kbit/s)	*N* x 10	*N* is a positive integer ranging from 1 to 5.
G726 (24 kbit/s)	*N* x 15	*N* is a positive integer ranging from 1 to 5.
G726 (32 kbit/s)	*N* x 20	*N* is a positive integer ranging from 1 to 5.
G726 (40 kbit/s)	*N* x 25	*N* is a positive integer ranging from 1 to 5.
IMA ADPCM	Number of bytes in each block/2	The number of bytes in each block indicates the number of bytes in the encoded IMA ADPCM data of each block, corresponding to **nblockalign** (0x20−0x21, 2-byte) of the IMA ADPCM WAV header.

NOTE

- Of all the ADPCM formats, only the IMA ADPCM format is supported. The number of bytes in each sampling point (**wbitspersample**) must be 4.
- If the WAV header is added to the frames in ADPCM streams, the number of bytes in each block can be obtained from the WAV header. For the raw ADPCM streams, the number of bytes in each block must be obtained from the provider of the streams.
- Only the mono-channel encoding format is supported.

3. Add the HiSilicon voice frame header, as shown in Figure 6-2.

Figure 6-2 Adding the HiSilicon voice frame header



The reference code for adding the HiSilicon voice frame header is as follows:

```
int HisiVoiceAddHisiHeader(short *inputdata, short *Hisivoicedata, int
PersampleLen,int inputsamplelen)
{
    int len = 0, outlen = 0;
    short HisiHeader[2];
    short *copyHisidata, *copyinputdata;
    int copysamplelen = 0;
    HisiHeader[0] = (short)(0x001<<8) & (0x0300);
    HisiHeader[1] = PersampleLen & 0x00ff;
    copysamplelen = inputsamplelen;
    copyHisidata = Hisivoicedata;
    copyinputdata = inputdata;
    while(copysamplelen >= PersampleLen)
    {
        memcpy(copyHisidata, HisiHeader, 2 * sizeof(short));
        outlen += 2;
        copyHisidata += 2;
        memcpy(copyHisidata, copyinputdata, PersampleLen * sizeof(short));
        copyinputdata += PersampleLen;
        copyHisidata += PersampleLen;
        copysamplelen -= PersampleLen;
        outlen += PersampleLen;
    }
    return outlen;
}
```

6.3 What Do I Do If High-Frequency Information Is Lost After VQE Is Enabled?

6.3.1 What Do I Do If High-Frequency Information Is Lost After VQE Is Enabled?

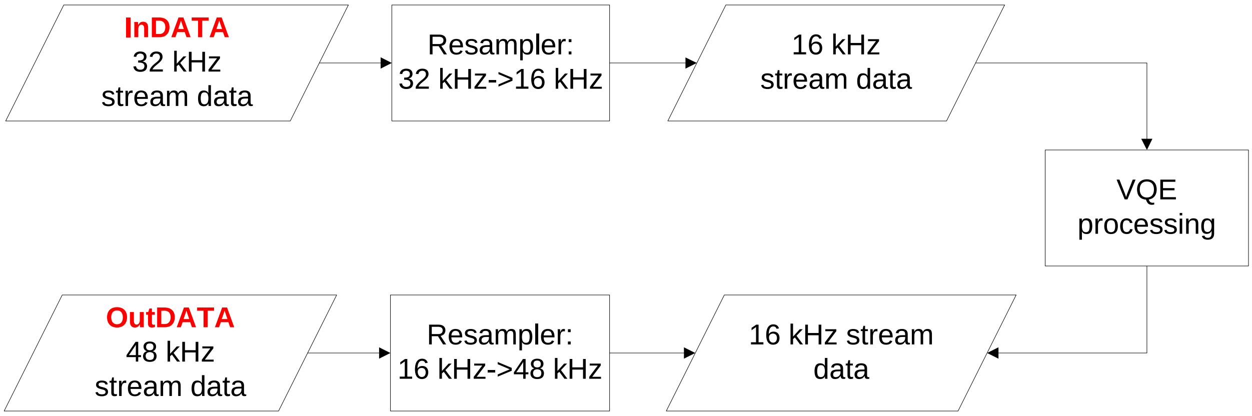
[Symptom]

When the AI sampling rate (**AISampleRate**) is 32 kHz, the voice quality enhancement (VQE) working sampling rate (**VQEWorkSampleRate**) is 16 kHz, output sampling rate (**ResOutSampleRate**) is 48 kHz, and the VQE and resampling functions are enabled, the 8 kHz or higher-frequency information is lost according to the analysis result of the output sequence.

[Cause Analysis]

In actual applications, the HiVQE supports only the 8 kHz and 16 kHz working sampling rates. To meet customer requirements, the resampling layer is encapsulated in the HiVQE to support any standard sampling rate from 8 kHz to 48 kHz. When **AISampleRate**, **VQEWorkSampleRate**, and **ResOutSampleRate** are set to **48 kHz**, **16 kHz**, and **48 kHz** respectively, the resampling layer resamples data from 32 kHz to 16 kHz, and then resamples data from 16 kHz to 48 kHz for output after VQE processing. Figure 6-3 shows the VQE processing flow.

Figure 6-3 VQE processing flow



When data is resampled from 32 kHz to 16 kHz, 8 kHz or higher-frequency information is lost.

In the current application scenario, the frequency band information of the output sequence is as follows:

- When VQE and resampling are disabled, the information within the frequency band of 0 to (**AISampleRate**/2) is output. For example, if the AI sampling rate (**AISampleRate**) is 48 kHz, the frequency band of the output information is 0 kHz to 24 kHz.
- When VQE is disabled and resampling is enabled, the information within the frequency band of 0 to min(**AISampleRate**, **ResOutSampleRate**)/2 is output. For example, if the AI sampling rate (**AISampleRate**) is 16 kHz and the output sampling rate

(**ResOutSampleRate**) is 32 kHz, min(**AISampleRate**, **ResOutSampleRate**) is 16 kHz. Therefore, the frequency band of the output information is 0 kHz to 8 kHz.

- When VQE is enabled and resampling is disabled, the information within the frequency band of 0 to min(**AISampleRate**, **VQEWorkSampleRate**)/2 is output. For example, if the AI sampling rate (**AISampleRate**) is 32 kHz and the VQE working sampling rate (**VQEWorkSampleRate**) is 16 kHz, min(**AISampleRate**, **VQEWorkSampleRate**) is 16 kHz. Therefore, the frequency band of the output information is 0 kHz to 8 kHz.
- When VQE and resampling are enabled, the information within the frequency band of 0 to min(**AISampleRate**, **VQEWorkSampleRate**, **ResOutSampleRate**)/2 is output. For example, if the AI sampling rate (**AISampleRate**) is 32 kHz, the VQE working sampling rate (**VQEWorkSampleRate**) is 16 kHz, and the output sampling rate (**ResOutSampleRate**) is 48 kHz, min(**AISampleRate**, **VQEWorkSampleRate**, **ResOutSampleRate**) is 16 kHz. Therefore, the frequency band of the output information is 0 kHz to 8 kHz.

NOTE

- After the sampling rate is configured, the maximum frequency of the output information is half of the sampling rate.
- The standard sampling rates from 8 kHz to 48 kHz are supported, including 8 kHz, 11.025 kHz, 12 kHz, 16 kHz, 22.05 kHz, 32 kHz, 44.1 kHz, and 48 kHz.
- The processing flow of the audio output unit (AOU) is similar to that of the audio input unit (AIU).

6.4 What Do I Do If Abnormal Amplitude Frequency Responses Occur During the Embedded Audio Codec Output (AO Output)

[Symptom]

During the testing of the amplitude frequency responses for the embedded audio codec (DAC) output (AO), the amplitude frequency response is attenuated significantly in the frequency bands higher than 2 kHz.

[Cause Analysis]

This case is caused by the enabled de-emphasis function of the dacl_deemph and dacr_deemph bits for the control register of audio codec. (For details, see the Hi35*XX* IP camera SoC data sheet.) The de-emphasis function is relative to the pre-emphasis function, and is modification on pre-emphasis. If the audio signal input to the AO channel is pre-emphasized, the signal can be restored to the normal amplitude frequency response when the de-emphasis function is enabled. If the audio signal input to the AO channel is not pre-emphasized, the amplitude frequency response will be affected when the de-emphasis function is enabled in dacl_deemph and dacr_deemph (the bits are not set to 00). This test is performed when the HiVQE function is disabled. During the test, the data input to the AO channel is not pre-emphasized. However, the de-emphasis function is enabled in the dacl_deemph and dacr_deemph bits of the control register for the audio codec (the bits are not set to 00). As a result, this issue occurs.

[Solution]

For the AI channel, the pre-emphasis function of the audio codec control register is disabled. Therefore, the de-emphasis function needs to be disabled by default in the dacl_deemph and dacr_deemph bits of the control register by setting the two bits to **00**. Note that the pre-emphasis function and the de-emphasis function should be used in pairs.

[Note]

If the audio codec control register does not contain the dacl_deemph and dacr_deemph bits, the de-emphasis function is not supported. In this case, the default configuration should be applied and additional configuration is not allowed.

6.5 Static Library Registration

- For a chip that supports the VQE static library registration function, select the required VQE and resampling module based on the actual application scenario, and then register the selected module with the audio system by calling HI_MPI_AUDIO_RegisterVQEModule interface. For details, see chapter 9 "Audio" in the *HiMPP V4.0 Media Processing Software Development Reference*.
- For a chip that supports the AAC static library registration function, determine whether to use the SBRENC and SBRDEC modules based on the actual application scenario. When the EAAC or EAACPLUS codec type is used, you must implement static registration for the SBRENC and SBRDEC modules before registering the codec. For details, see the *Audio Components API Reference*.

6.6 What Do I Do If Pop Tones Occur When the Built-in Codec Module Is Loaded?

[Symptom]

When the built-in Codec module is loaded, the audio output end of the board has pop tones.

[Cause Analysis]

Take Hi3516C V500 as an example. The demute operation of the power amplifier (PA) on the demo board is implemented in the sys_config module. The corresponding function is ampunmute. However, the built-in Codec module is loaded after the sys_config module. The PA has performed the demute operation when the Codec module is loaded. As a result, pop tones occur.

[Solution]

Taking Hi3516CV500 as an example, the demute operation of the PA in the sys_config module has been moved to after the insert_audio operation in the **load3516cv500** script.

7 Low Power Consumption

7.1 What Do I Do If the Frequency Is Frequently Modulated During Dynamic Frequency Modulation of the Low-Power Module?

[Symptom]

When the dynamic frequency modulation policy (such as the on demand policy) is used after the low-power module **hi35xx_pm.ko** is loaded, the frequency is frequently modulated.

[Cause Analysis]

The Linux kernel uses the 100 Hz frequency by default, that is, 10 ms statistical period. The statistical time granularity is coarse, which leads to low precision in statistics. In this case, the CPU load statistics fluctuates significantly. Linux implements dynamic voltage and frequency scaling (DVFS) in each statistical cycle based on the CPU load statistics, which results in frequent frequency modulation of the low-power module.

[Solution]

Change the frequency of the Linux kernel to 1000 Hz to improve the statistical precision, or increase the statistical cycle of the low-power module. For example, you can run the following command (the unit of the statistical cycle is μs) to change the statistical cycle of the on demand policy to 1s:

echo 1000000 >/sys/devices/system/cpu/cpufreq/ondemand/sampling_rate

8 LCD Debugging

8.1 Supported LCDs

The LCD compatibility depends on whether the LCD interface type matches the chip capability.

For details about the LCD interface types supported by the chip, see chapter 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference.*

8.2 LCD Debugging Sequence

8.2.1 Confirming Pin Multiplexing Configurations

You need to confirm that all pins for connecting the LCD are correctly configured as VO-related functions, and the driving capabilities of these pins are properly configured. For details about the pin configurations, see the Hi35*XX* IP camera SoC data sheet and the **sys_config.c** file in the release package.

8.2.2 Confirming User Timings

Currently the SDK provides only one kind of timing for each interface type. For example, for the VO_INTF_LCD_8BIT interface, only VO_OUTPUT_320X240_60 is provided. In addition, the timing is valid for only a specific LCD model. For example, VO_OUTPUT_320X240_60 applies to the LCDs with the driver IC OTA5182. Therefore, user timings are required when LCDs are debugged in most cases.

When configuring the public attributes of the output by calling HI_MPI_VO_SetPubAttr, select the correct LCD interface type and the VO_OUTPUT_USER interface timing, and then configure the user timing structure based on the requirements of the LCD.

```
typedef struct tagVO_SYNC_INFO_S
{
    HI_BOOL  bSynm;     /* sync mode(0:timing,as BT.656; 1:signal,as LCD)
*/
    HI_BOOL  bIop;      /* interlaced or progressive display(0:i; 1:p) */
    HI_U8    u8Intfb;   /* interlace bit width while output */
    HI_U16   u16Vact ;  /* vertical active area */
```

```
    HI_U16   u16Vbb;    /* vertical back blank porch */
    HI_U16   u16Vfb;    /* vertical front blank porch */
    HI_U16   u16Hact;   /* horizontal active area */
    HI_U16   u16Hbb;    /* horizontal back blank porch */
    HI_U16   u16Hfb;    /* horizontal front blank porch */
    HI_U16   u16Hmid;   /* bottom horizontal active area */
    HI_U16   u16Bvact;  /* bottom vertical active area */
    HI_U16   u16Bvbb;   /* bottom vertical back blank porch */
    HI_U16   u16Bvfb;   /* bottom vertical front blank porch */
    HI_U16   u16Hpw;    /* horizontal pulse width */
    HI_U16   u16Vpw;    /* vertical pulse width */
    HI_BOOL  bIdv;      /* inverse data valid of output */
    HI_BOOL  bIhs;      /* inverse horizontal synch signal */
    HI_BOOL  bIvs;      /* inverse vertical synch signal */
} VO_SYNC_INFO_S;
```

Table 8-1 describes the parameters.

Table 8-1 Parameter description

Parameter	Description
bSynm	Sync mode. Set it to **1** for LCDs, indicating signal synchronization.
bIop	**0** indicates interlaced, and **1** indicates progressive. Set it to **1** for LCDs typically.
u8Intfb	Invalid parameter, which can be ignored
u16Vact	Vertical active region. It indicates the vertical active region of the top field in interlaced output mode. The unit is row.
u16Vbb	Vertical blank back porch. It indicates the vertical blank back porch of the top field in interlaced output mode. The unit is row.
u16Vfb	Vertical blank front porch. It indicates the vertical blank front porch of the top field in interlaced output mode. The unit is row.
u16Hact	Horizontal active region. The unit is pixel.
u16Hbb	Horizontal blank back porch. The unit is pixel.
u16Hfb	Horizontal blank front porch. The unit is pixel.
u16Hmid	Valid pixel value of bottom field vertical synchronization
u16Bvact	Vertical active region of the bottom field, which is valid in interlaced mode. The unit is row.
u16Bvbb	Vertical blank back porch of the bottom field, which is valid in interlaced mode. The unit is row.
u16Bvfb	Vertical blank front porch of the bottom field, which is valid in interlaced mode. The unit is row.

Parameter	Description
u16Hpw	Width of the horizontal sync signal. The unit is pixel.
u16Vpw	Width of the vertical sync signal. The unit is row.
bIdv	Polarity of the data validity signal. **0** indicates active high, and **1** indicates active low.
bIhs	Polarity of the horizontal validity signal. **0** indicates active high, and **1** indicates active low.
bIvs	Polarity of the vertical validity signal. **0** indicates active high, and **1** indicates active low.

For details about how to configure the user timings, see the related LCD screen document. Note that the unit of each value is consistent with the requirement.

8.2.3 Configuring the Device Frame Rate

When the user timing is used, you need to call the HI_MPI_VO_SetDevFrameRate MPI to configure the device frame rate. For details about this MPI, see chapter 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference*.

8.2.4 Confirming Clock Configurations

In addition to configuring the user timing and frame rate, you need to configure the clock and frequency division ratio of the user timing by calling the HI_MPI_VO_SetUserIntfSyncInfo MPI.

For details about this MPI, see chapter 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference*. The following uses Hi3519A V100 as an example to describe how to configure this MPI.

The clock source of the user timing can be obtained from the PLL or LCD frequency divider. If the PLL is selected as the clock source, you need to configure parameters **u32Fbdiv**, **u32Frac**, **u32Refdiv**, **u32Postdiv1**, and **u32Postdiv2** of the PLL. For the meanings of the five parameters, see the description of the PLL configuration in the "System" chapter in the Hi35*XX* IP camera SoC data sheet. Configure the five parameters properly to obtain the desired clock. If the LCD frequency divider is used as the clock source, you need to set the **u32LcdMClkDiv** parameter. For details, see the description of the LCD clock register in the "System" chapter in the Hi35*XX* IP camera SoC data sheet.

The **bClkReverse** parameter in the HI_MPI_VO_SetUserIntfSyncInfo MPI can be used to reverse the VDP clock and adjust the VDP clock phase.

In the configuration of the HI_MPI_VO_SetUserIntfSyncInfo MPI, you need to confirm the frequency division ratio, which is the ratio of the VDP output clock (chip output clock) to the HD channel clock. The HD channel outputs one pixel during one clock beat. However, the LCD requires multiple clock beats to output a pixel. For example:

- If the required data sequence of an 8-bit serial LCD is the RGB sequence, that is, three clock beats are required to transfer the R, G, and B data for one pixel, set the frequency division ratio to 3.

- For a 16-bit serial LCD, one clock beat is required to output one pixel. In this case, set the frequency division ratio to 1.

9 VO

9.1 How Do I Configure the VO User Timing?

9.1.1 Configuring the Timing Structure

In the HI_MPI_VO_SetPubAttr interface, set **pstPubAttr-> enIntfSync** to **VO_OUTPUT_USER**, and then configure the stSyncInfo structure. The parameters in stSyncInfo are listed as follows:

```
typedef struct tagVO_SYNC_INFO_S
{
    HI_BOOL  bSynm;     /* sync mode(0:timing,as BT.656; 1:signal,as LCD) 
*/
    HI_BOOL  bIop;      /* interlaced or progressive display(0:i; 1:p) */
    HI_U8    u8Intfb;   /* interlace bit width while output */
    HI_U16   u16Vact ;  /* vertical active area */
    HI_U16   u16Vbb;    /* vertical back blank porch */
    HI_U16   u16Vfb;    /* vertical front blank porch */
    HI_U16   u16Hact;   /* horizontal  active area */
    HI_U16   u16Hbb;    /* horizontal  back blank porch */
    HI_U16   u16Hfb;    /* horizontal  front blank porch */
    HI_U16   u16Hmid;   /* bottom horizontal  active area */
    HI_U16   u16Bvact;  /* bottom vertical active area */
    HI_U16   u16Bvbb;   /* bottom vertical back blank porch */
    HI_U16   u16Bvfb;   /* bottom vertical front blank porch */
    HI_U16   u16Hpw;    /* horizontal pulse width */
    HI_U16   u16Vpw;    /* vertical pulse width */
    HI_BOOL  bIdv;      /* inverse data valid of output */
    HI_BOOL  bIhs;      /* inverse horizontal synch signal */
    HI_BOOL  bIvs;      /* inverse vertical synch signal */
} VO_SYNC_INFO_S;
```

Table 9-1 describes the parameters.

Table 9-1 Parameter description

Parameter	Description
bSynm	Synchronization mode. This parameter is set to **1** for the LCD, indicating that signals are synchronized.
bIop	The value **0** indicates the interlaced mode, and **1** indicates the progressive mode. This parameter is set to **1** for the LCD.
u8Intfb	Invalid parameter, which can be ignored
u16Vact	Vertical active area (unit: line). It indicates the top vertical active area in interlaced output mode.
u16Vbb	Vertical blanking back porch (unit: line). It indicates the top vertical blanking back porch in interlaced output mode.
u16Vfb	Vertical blanking front porch (unit: line). It indicates the top vertical blanking front porch in interlaced output mode.
u16Hact	Horizontal active area (unit: pixel)
u16Hbb	Horizontal blanking back porch (unit: pixel)
u16Hfb	Horizontal blanking front porch (unit: pixel)
u16Hmid	Bottom vertical sync active pixel
u16Bvact	Bottom vertical active area (unit: line). It is active in interlaced mode.
u16Bvbb	Bottom vertical blanking back porch (unit: line). It is active in interlaced mode.
u16Bvfb	Bottom vertical blanking front porch (unit: line). It is active in interlaced mode.
u16Hpw	Width of the horizontal sync signal (unit: pixel)
u16Vpw	Width of the vertical sync signal (unit: line)
bIdv	Polarity of the data valid signal, active high when set to **0** and active low when set to **1**
bIhs	Polarity of the horizontal valid signal, active high when set to **0** and active low when set to **1**
bIvs	Polarity of the vertical valid signal, active high when set to **0** and active low when set to **1**

Take the configuration of the 384 x 288P@25 fps user timing of the Hi3519A V100 as an example. The following lists the settings of stSyncInfo:

pstPubAttr->stSyncInfo.bSynm = 0;

pstPubAttr->stSyncInfo.bIop = 1;

pstPubAttr->stSyncInfo.u8Intfb = 0;


```
pstPubAttr->stSyncInfo.u16Vact = 288;
pstPubAttr->stSyncInfo.u16Vbb = 200;
pstPubAttr->stSyncInfo.u16Vfb = 112;

pstPubAttr->stSyncInfo.u16Hact = 384;
pstPubAttr->stSyncInfo.u16Hbb = 300;
pstPubAttr->stSyncInfo.u16Hfb = 216;

pstPubAttr->stSyncInfo.u16Hmid = 1;
pstPubAttr->stSyncInfo.u16Bvact = 1;
pstPubAttr->stSyncInfo.u16Bvbb = 1;
pstPubAttr->stSyncInfo.u16Bvfb = 1;

pstPubAttr->stSyncInfo.u16Hpw = 4;
pstPubAttr->stSyncInfo.u16Vpw = 5;

pstPubAttr->stSyncInfo.bIdv = 0;
pstPubAttr->stSyncInfo.bIhs = 0;
pstPubAttr->stSyncInfo.bIvs = 0;
```

The formula for calculating the VO clock frequency is as follows:

VO clock frequency = (Valid width + Horizontal back blanking + Horizontal front blanking) x (Valid height + Vertical back blanking + Vertical front blanking) x Frame rate

You need to configure the horizontal and vertical front and back blanking lengths to match the configured VO clock.

In this example, the VO clock frequency is calculated as follows: VO clock frequency = (384 + 300 + 216) x (288 + 200 + 112) x 25 = 13500000. That is, the VO clock frequency should be set to 13.5 MHz.

The calculation formula is as follows:

VO clock frequency = (Valid width + Horizontal back blanking + Horizontal front blanking) x (Valid height + Vertical back blanking + Vertical front blanking) x Frame rate

9.1.2 Configuring the Clock Size

Configure the clock and frequency division ratio of the user timing by calling the HI_MPI_VO_SetUserIntfSyncInfo interface.

For details about this MPI, see chapter 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference*.

9.2 Image Switching

9.2.1 Channel Attribute Changes

Image switching: In display state, the display position and size of a channel change.

9.2.2 Recommended Implementation Method

If the channel is enabled or displayed, you are advised to perform the following steps to modify the channel attributes by calling HI_MPI_VO_SetChnAttr (for example, the channel ID is *chn-x*):

Step 1 Start batch processing for the layer where the channel is located by calling HI_MPI_VO_BatchBegin.

Step 2 Hide all channels by calling HI_MPI_VO_HideChn.

Step 3 Set attributes for the target channel *chn-x* (or multiple channels) by calling HI_MPI_VO_SetChnAttr.

Step 4 Display all channels by calling HI_MPI_VO_ShowChn.

Step 5 End the batch processing for the layer where the channel is located by calling HI_MPI_VO_BatchEnd.

----End

The following describes the reference process:

```
/*
* n --> m : change n chns to m chns.
* Set attributes for the target channels chn-0 to chn-m.
*/
SetChnMAttr()
{
/* batch begin  */
     s32Ret = HI_MPI_VO_BatchBegin(0);
     if (HI_SUCCESS != s32Ret)
     {
     SAMPLE_PRT("HI_MPI_VO_BatchBegin(0) failed!\n");
     }
     /* hide all n chns */
     for(i=0;i<n;i++)
     {
       s32Ret = HI_MPI_VO_HideChn(0, i);
       if (HI_SUCCESS != s32Ret)
       {
           SAMPLE_PRT("HI_MPI_VO_HideChn(0,%d) failed!\n",i);
        }
     }
     /* change all m chns’s attr */
```

```
      for(j=0;j<m;j++)
      {
         s32Ret = HI_MPI_VO_SetChnAttr(0, j, &stSetChnAttr);
          if (HI_SUCCESS != s32Ret)
          {
            SAMPLE_PRT("HI_MPI_VO_SetChnAttr(0,%d) failed!\n",j);
   }
}

          /* enable all m chns */
for(j=0;j<m;j++)
{
         s32Ret = HI_MPI_VO_EnableChn(0, j);
          if (HI_SUCCESS != s32Ret)
          {
             SAMPLE_PRT("HI_MPI_VO_EnableChn (0,%d) failed!\n",j);
          }
}

         /* show  all m chns*/
         for(i=0;i<n;i++)
         {
              s32Ret = HI_MPI_VO_ShowChn(0, i);
              if (HI_SUCCESS != s32Ret)
              {
              SAMPLE_PRT("HI_MPI_VO_ShowChn(0,%d) failed!\n",i);
              }
          }
          /* batch end  */
         s32Ret = HI_MPI_VO_BatchEnd(0);
         if (HI_SUCCESS != s32Ret)
         {
         SAMPLE_PRT("HI_MPI_VO_BatchEnd(0) failed!\n");
         }
}
```

9.3 Video Synchronization Solution

Video synchronization implements video synchronization output for different VO devices of the same chip or VO devices of different chips. In video synchronization scenarios, the multi-channel decoding result is sent to the VPSS and then to multiple VO devices for stitching. To ensure the stitching effect, you need to synchronize the videos of all VO devices.

9.3.1 Implementation Principle

The basic implementation principle of the video synchronization solution is to ensure that the VO video frame and the VO clock are synchronized.

- Clock synchronization:

  Clock synchronization is controlled by the module parameter **bDevClkExtEn**. The default value is **0**. Set the parameter to **1** before system initialization. It indicates that the interface clock of the VO device is user-defined. After the service is started, you can disable and then enable the VO device clock to ensure the clock synchronization output of each device.

- TX frame synchronization:

  The hi_user driver provides a response function for the VO device interrupt. You can set the listening response mechanism, so that the driver sends video frames to the VO device only after the interrupt is reported. You can also add set more TX frame synchronization mechanisms.

- Device switch:

  On the basis of clock synchronization, the external functions VOU_DRV_EnableDev and VOU_DRV_DisableDev are added as the master display switches for all devices. Use these functions only when the interface clocks are enabled.

- The declaration formats of the caller are as follows:extern void VOU_DRV_EnableDev(int VoDev)
- extern void VOU_DRV_DisableDev(int VoDev)

9.3.2 Recommended Operation Procedure

To implement the video synchronization solution, perform the following steps:

Step 1 Enable the device interface clock before the service is started.

Step 2 Start the service and set the module parameter **bDevClkExtEn** to **1** before system initialization. Call the MPI according to the standard VO process if the VO service is started for the first time.

Step 3 Disable each VO device by calling VOU_DRV_DisableDev.

Step 4 Before the decoding starts, simultaneously enable and then simultaneously disable the interface clocks of all VO devices by setting the corresponding bit of each interface clock, to ensure clock synchronization. Take Hi3559A V100 as an example, the interface clock of DHD0 is controlled by register 0x12010124[6] while the interface clock of DHD1 is controlled by register 0x12010124[4][7].

Step 5 Call VOU_DRV_EnableDev to start each device to start decoding and sending frames.

Step 6 Listen to the clock deviation and repeat Steps 3, 4, and 5. A clock deviation may occur after the VO devices run for a long time.

---End

NOTICE

This solution is valid only for Hi3559A V100.

10 VENC

10.1 Precautions for Configuring the JPEG Quantization Table

Currently, if **u32Qfactor** is too low for JPEG encoding, encoded JPEG images will have color cast or other symptoms. The cause is that the chrominance quantization step is too large. You can call the HI_MPI_VENC_SetJpegParam interface to modify the chrominance quantization table and limit the chrominance quantization step to avoid color cast or other symptoms. For details about the relationship between the specific **u32Qfactor** value and the quantization table, see the RFC 2435 standard. Modifying the chrominance quantization table may result in the increase of the JPEG image capacity. You need to balance image quality and the JPEG image capacity.

11 HDMI

Hi3559A V100 HDMI Precautions

The Hi3559A V100 hardware Timer11 and the corresponding interrupt source will be occupied by the HDMI. Do not use Timer11. Otherwise, the HDMI may not work properly.

12 Others

12.1 Dynamic Library

12.1.1 What Do I Do If the Dynamic Libraries Cannot Be Used When the Application Is Statically Compiled?

[Symptom]

The file systems and executable programs of customer A are statically compiled. As a result, the dynamic libraries in the SDK cannot be used.

[Cause Analysis]

The current arm-linux-gcc version supports the static compilation, dynamic compilation, and semi-static compilation.

- In static compilation mode (compilation options: **-static**, **-pthread**, **-lrt**, and **-ldl**), the **libc**, **libpthread**, **librt**, and **libdl** libraries are all compiled to the executable program. The static compilation does not depend on any system dynamic library and is implemented independently. However, the dynamic libraries cannot be used in this mode.
- In dynamic compilation mode (common compilation), the system dynamic libraries under **/lib** are linked. Therefore, the compiled program depends on the system dynamic libraries. The advantage of the dynamic compilation is that the system dynamic libraries can be shared by multiple executable programs such as the BusyBox and Himount under **/bin**.
- In semi-static compilation mode (compilation options: **-static-libgcc**, **-static-libstdc++**, **-L**, **-pthread**, **-lrt**, and **-ldl**), the **libgcc** and **libstdc++** libraries are compiled to the executable program. Other system libraries still depend on the system dynamic libraries. In this mode, the dynamic libraries can be used, but the **libc**, **libpthread**, **librt**, and **libdl** files still need to be placed under the system directory.

[Solution]

Adopt the dynamic compilation and place the system files that the dynamic libraries depend on (including **ld-uClibc.so.0**, **libc.so.0**, **libpthread.so.0**, **librt.so.0**, and **libdl.so.0**) under **/lib**.

12.1.2 What Do I Do If a Redefinition Error Occurs When libupvqe.a and libdnvqe.a Are Used for Dynamic Compilation?

[Symptom]

A redefinition error occurred when customer B compiled the audio component libraries **libupvqe.a** and **libdnvqe.a** into a dynamic library. The compilation statement is as follows:

```
$(CC) -shared -o $@ -L. -Wl,--whole-archive libupvqe.a libdnvqe.a -Wl,--
no-whole-archive
```

[Cause Analysis]

libupvqe.a and **libdnvqe.a** share some functional modules to implement code reuse and modularization, and save the file space when ELF files are generated after compilation.

The static libraries can be compiled into the dynamic library in any of the following ways:

- Directly use the **-l** compilation option. The compilation statement is as follows:

```
$(CC) -shared -o libshare.so -L. -lupvqe -ldnvqe
```

This is a link compilation method. The function symbols of the static libraries are not linked to the **libshare.so** library generated after compilation.

- Use the **-Wl,--whole-archive** compilation option. The compilation statement is as follows:

```
$(CC) -shared -o $@ -L. -Wl,--whole-archive libupvqe.a libdnvqe.a -Wl,--
no-whole-archive
```

In this method, the function symbols of the static libraries are compiled to **libshare.so**. However, **libupvqe.a** and **libdnvqe.a** cannot have functions with the same name.

- Split the .a files into multiple .o files respectively, and then compile the .o files into the .so file. The compilation statement is as follows:

```
LIB_PATH = ./
EXTERN_OBJ_DIR = ./EXTERN_OBJ
LIBUPVQE_NAME = libupvqe.a
LIBDNVQE_NAME = libdnvqe.a
EXTERN_OBJ = $(EXTERN_OBJ_DIR)/*.o
all: pre_mk $(TARGET) pre_clr
pre_mk:
    @mkdir -p $(EXTERN_OBJ_DIR);
    @cp $(LIB_PATH)$(LIBUPVQE_NAME) $(EXTERN_OBJ_DIR);
    @cd $(EXTERN_OBJ_DIR); $(AR) -x $(LIBUPVQE_NAME);
    @cp $(LIB_PATH)$(LIBDNVQE_NAME) $(EXTERN_OBJ_DIR);
    @cd $(EXTERN_OBJ_DIR); $(AR) -x $(LIBDNVQE_NAME);

$(TARGET):
    #$(CC) -shared -o $@ -L. libupvqe.so libdnvqe.so
    $(CC) -shared -o $@ -L. $(EXTERN_OBJ)

pre_clr:
@rm -rf $(EXTERN_OBJ_DIR);
```

NOTE

In this method, **libupvqe.a** and **libdnvqe.a** are split into .o files respectively, and then the .o files are compiled into the .so file. The function symbols of the static libraries are compiled to the .so file, and no function name conflict occurs.

[Solution]

Customer B used the second compilation method. A redefinition error occurred because **libupvqe.a** and **libdnvqe.a** have functions with the same name. To solve this issue, customer B can use the first or third compilation method to generate the **libshare.so** file.

12.1.3 Dependency Relationships Between Module .ko Files

- Each loaded .ko file is explicitly dependent. When the **lsmod** command is executed to check the relationship, the **Used by** flag is found. These .ko files need to be loaded in sequence and unloaded in the reverse sequence.
- Some module .ko files are implicitly dependent. For example, some public basic .ko files (**mmz.ko**, **hi_media.ko**, **hi35xx_base.ko**, **hi35 xx _sys.ko**, **hi35 xx _tde.ko**, and **hi35 xx _region.ko**) need to be loaded first. If these .ko files are separately unloaded and then loaded, exceptions may be caused. If exceptions occur, unload and then load these modules in sequence.
- Some common scheduling module .ko files, such as **hi35xx_chnl.ko**, are called by the encoding module, region module, and other modules. If these .ko files are unloaded, the encoding module and region module may become abnormal. The **hi35xx_aio.ko** for the basic audio module is not explicitly depended by but is indispensable to other .ko files of the audio module.

12.1.4 SPI Driver Specifications

There are many peripherals that use SPI interfaces, such as sensors. Generally, the SPI or I^2C interfaces are used to configure the sensor register for chips. The SPI interfaces are used as an example here. For sensor configurations, the IPC package provides two SPI drivers, **hi_sensor_spi.ko** and **hi_ssp_sony.ko**. Take the sensor configuration using the Sony SPI as an example to describe the difference between the two SPI drivers.

- **hi_sensor_spi.ko** is the standard SPI driver in the kernel. However, the sensor may not be configuration in a timely manner when the system is busy.
- **hi_ssp_sony.ko** is the SPI driver developed by HiSilicon. It is not standard and is currently used by the CMV50000 sensor.

12.1.5 How Do I Dynamically Load a DLL in Huawei LiteOS?

Huawei LiteOS supports the dynamic loading of a dynamic link library (DLL). For details, see the *Huawei LiteOS V200R002C00 Developer Guide*.

The audio module is used as an example. To use functions such as VQE, AAC encoding and decoding, and resampling, DLLs such as **libaaccomm.so**, **libhive_common.so**, and **libhive_RES.so** are required. To dynamically load a DLL, perform the following steps:

Step 1 Save the required DLLs to the same directory on the server, for example, **/home/audio/lib_so/**.

Step 2 Run the following command to extract the external function symbols of the DLLs using the **sym.sh** script, which is stored in **$(LITEOS)/tools/scripts/dynload_tools** of the Huawei LiteOS SDK:

```
cd $( LITEOS)/tools/scripts/dynload_tools
```

```
./sym.sh /home/audio/lib_so
```

$(LITEOS) indicates the root directory of the Huawei LiteOS SDK, which must be specified by the user.

Step 3 Use **makefile** of the dynload tool in the Huawei LiteOS SDK to generate the **los_dynload_gsymbol.o** file. **makefile** is stored in **$(LITEOS)/kernel/extended/dynload** and the **los_dynload_gsymbol.o** file is stored in **$(LITEOS)/out/$(CHIP)/obj/kernel/extended/dynload/src** of the Huawei LiteOS SDK. Run the following command:

```
cd $( LITEOS)/kernel/extended/dynload
make export LITEOSTOPDIR=$( LITEOS)
```

$(CHIP) indicates the chip model.

During compilation of Huawei LiteOS apps, before using a DLL, you have to call the LOS_PathAdd interface to specify a path for loading the DLL on the board and save the DLL file to the path.

Step 4 When compiling a bin system image of Huawei LiteOS, link the **los_dynload_gsymbol.o** file.

----End