Novatek HDAL Design Specification - hd_videoprocess

Table of Content

Difference Table (for IPC only)

Item	NT9668X	NT9852X
IN_STAMP_ATTR	Only for RGB565	For ARGB4444、ARGB1555、RGB565
IN_MASK_ATTR	Hollow mask is not supported in builtin	If hollow masks are set in builtin, only mask 0/2/4/6 are available and corresponding 1/3/5/7 should be left unused. Ext masks are free of this restriction

Item	NT9668X	NT9852X
DevCfg.pipe	RAWALL YUVALL - - PANO360 PANO360_4V COLOR_ONLY SCALE_ONLY GDC_ONLY	RAWALL YUVALL RAWCAP YUVCAP PANO360 - COLOR_ONLY SCALE_ONLY GDC_ONLY
PathCfg.in_func [input 0]	n/a	DIRECT
PathCfg.out_func [output 0]	n/a	LOWLATENCY
PathCfg.out_func [output 1]	n/a	LOWLATENCY
PathCfg.out_func [output 2]	n/a	LOWLATENCY
PathCfg.out_func [output 3]	n/a	n/a
PathCfg.out_func [output 4]	n/a	LOWLATENCY

for DevCfg.pipe = RAWALL

Item	NT9668X	NT9852X
Ctrl.func	WDR	WDR

	SHDR DEFOG 3DNR COLORNR	SHDR DEFOG 3DNR COLORNR
Ctrl.ref_path_3dnr	YES	YES
Ctrl.trig_time_lowlatency	n/a	YES
OSG path	YES	YES
IN.dim,w, h	w need 4 align h need 4 align	w need 4 align h need 4 align
OUT.dim,w, h	w need 4 align h need 4 align	w need 4 align h need 4 align
IN scale to OUT[0]	31.99x ~ 1/15.99x	31.99x ~ 1/15.99x (****)
IN scale to OUT[1]	31.99x ~ 1/15.99x	31.99x ~ 1/15.99x
IN scale to OUT[2]	31.99x ~ 1/15.99x	31.99x ~ 1/15.99x
IN scale to OUT[3]	31.99x ~ 1/15.99x	31.99x ~ 1/15.99x
IN scale to OUT[4]	31.99x ~ 1/15.99x	1x (*)
IN. pxlfmt [input 0]	RAW NRX	RAW NRX
OUT.pxlfmt [output 0]	YUV444 YUV422 YUV420 YUV444_PLANER YUV422_PLANER YUV420_PLANER Y8 YUV420_NVX1 RGB888_PLANER (**)	YUV420 YUV420_PLANER Y8 YUV420_NVX2 (****)
OUT.pxlfmt [output 1]	YUV422 YUV420 Y8	YUV420 Y8
OUT.pxlfmt [output 2]	YUV422 YUV420 Y8	YUV420 Y8
OUT.pxlfmt [output 3]	Y8	Y8

OUT.pxlfmt [output 4]	YUV422 YUV420 Y8	YUV420 (*) YUV420_NVX2 (*)
IN_CROP [input 0]	YES	YES (****)
OUT_CROP [output 0]	YES	YES (****)
OUT_CROP [output 1]	YES	YES
OUT_CROP [output 2]	YES	YES
OUT_CROP [output 3]	YES	YES
OUT_CROP [output 4]	YES	NO (*)

(*) OUT.dim.w,h is equal to IN.dim.w,h

(**) if use RGB888_PLANER, only support 1 out.

(****) if OUT.pxlfmt is YUV420_NVX2, and IN.dim.w > 2688 or ISP enable MSTRP Mode, cannot support Scale Down & Crop

for DevCfg.ppipe = YUVALL

Item	NT9668X	NT9852X
Ctrl.func	3DNR COLORNR	WDR DEFOG 3DNR COLORNR
Ctrl.ref_path_3dnr	YES	YES
Ctrl.trig_time_lowlatency	n/a	n/a
OSG path	YES	YES
IN.dim,w, h	w need 4 align h need 4 align	w need 4 align h need 4 align
OUT.dim,w, h	w need 4 align h need 4 align	w need 4 align h need 4 align
IN scale to OUT[0]	31.99x ~ 1/15.99x	31.99x ~ 1/15.99x
IN scale to OUT[1]	31.99x ~ 1/15.99x	31.99x ~ 1/15.99x

IN scale to OUT[2]	31.99x ~ 1/15.99x	31.99x ~ 1/15.99x
IN scale to OUT[3]	31.99x ~ 1/15.99x	31.99x ~ 1/15.99x
IN scale to OUT[4]	31.99x ~ 1/15.99x	1x (*)
IN. pxlfmt [input 0]	YUV422 YUV420 YUV444_PLANER YUV422_PLANER YUV420_PLANER Y8 YUV420_NVX1 RGB888_PLANER (**)	YUV422 YUV420
OUT.pxlfmt [output 0]	YUV444 YUV422 YUV420 YUV444_PLANER YUV422_PLANER YUV420_PLANER Y8 YUV420_NVX1 RGB888_PLANER (**)	YUV420 YUV420_PLANER Y8 YUV420_NVX2
OUT.pxlfmt [output 1]	YUV422 YUV420 Y8	YUV420 Y8
OUT.pxlfmt [output 2]	YUV422 YUV420 Y8	YUV420 Y8
OUT.pxlfmt [output 3]	Y8	Y8
OUT.pxlfmt [output 4]	YUV422 YUV420 Y8	YUV420 (*) YUV420_NVX2 (*)
IN_CROP [input 0]	YES	YES
OUT_CROP [output 0]	YES	YES
OUT_CROP	YES	YES

[output 1]		
OUT_CROP [output 2]	YES	YES
OUT_CROP [output 3]	YES	YES
OUT_CROP [output 4]	YES	NO (*)

(*) OUT.dim.w,h is equal to IN.dim.w,h

(**) if use RGB888_PLANER, only support 1 out.

for DevCfg.ppipe = GDC_ONLY

Item	NT9668X	NT9852X
Ctrl.func	n/a	WDR
Ctrl.ref_path_3dnr	n/a	n/a
Ctrl.trig_time_lowlaten cy	n/a	n/a
OSG path	YES	n/a
IN.dim,w, h	w need 4 align h need 4 align	w need 4 align h need 4 align
OUT.dim,w, h	w need 4 align h need 4 align	w need 4 align h need 4 align
IN scale to OUT[0]~[4]	n/a	n/a
IN. pxlfmt [input 0]	YUV422 YUV420	YUV422 YUV420
OUT.pxlfmt [output 0]	YUV422 (***) YUV420 (***)	YUV422 (***) YUV420 (***)
OUT.pxlfmt [output 1]	n/a	n/a
OUT.pxlfmt [output 2]	n/a	n/a
OUT.pxlfmt [output 3]	n/a	n/a
OUT.pxlfmt [output 4]	n/a	n/a
IN_CROP [input 0]	n/a	n/a

OUT_CROP [output 0]	n/a	n/a
OUT_CROP [output 1]	n/a	n/a
OUT_CROP [output 2]	n/a	n/a
OUT_CROP [output 3]	n/a	n/a
OUT_CROP [output 4]	n/a	n/a

(***) OUT.pxlfmt must equal to IN.pxlfmt

for DevCfg.ppipe = COLOR_ONLY

Item	NT9668X	NT9852X
Ctrl.func	n/a	DEFOG
Ctrl.ref_path_3dnr	n/a	n/a
Ctrl.trig_time_lowlatency	n/a	n/a
OSG path	YES	YES
IN.dim,w, h	w need 4 align h need 2 align	w need 4 align h need 2 align
OUT.dim,w, h	w need 4 align h need 4 align	w need 4 align h need 4 align
IN scale to OUT[0]~[4]	n/a	n/a
IN. pxlfmt [input 0]	YUV444 YUV422 YUV420 Y8	YUV444 YUV422 YUV420 Y8
OUT.pxlfmt [output 0]	YUV444 YUV422 YUV420 Y8	YUV444 YUV422 YUV420 Y8
OUT.pxlfmt [output 1]	n/a	n/a
OUT.pxlfmt [output 2]	n/a	n/a

OUT.pxlfmt [output 3]	n/a	n/a
OUT.pxlfmt [output 4]	n/a	n/a
IN_CROP [input 0]	n/a	n/a
OUT_CROP [output 0]	n/a	n/a
OUT_CROP [output 1]	n/a	n/a
OUT_CROP [output 2]	n/a	n/a
OUT_CROP [output 3]	n/a	n/a
OUT_CROP [output 4]	n/a	n/a

for DevCfg.ppipe = SCALE_ONLY

Item	NT9668X	NT9852X
Ctrl.func	3DNR COLORNR	3DNR COLORNR
Ctrl.ref_path_3dnr	YES	YES
Ctrl.trig_time_lowlatency	n/a	YES
OSG path	YES	YES
IN.dim,w, h	w need 4 align h need 4 align	w need 4 align h need 4 align
OUT.dim,w, h	w need 4 align h need 4 align	w need 4 align h need 4 align
IN scale to OUT[0]	31.99x ~ 1/15.99x	31.99x ~ 1/15.99x
IN scale to OUT[1]	31.99x ~ 1/15.99x	31.99x ~ 1/15.99x
IN scale to OUT[2]	31.99x ~ 1/15.99x	31.99x ~ 1/15.99x
IN scale to OUT[3]	31.99x ~ 1/15.99x	31.99x ~ 1/15.99x
IN scale to OUT[4]	31.99x ~ 1/15.99x	1x (*)
IN. pxlfmt [input 0]	YUV422 YUV420	YUV420 YUV420_PLANER

	YUV444_PLANER YUV422_PLANER YUV420_PLANER Y8 YUV420_NVX1 RGB888_PLANER (**)	Y8
OUT.pxlfmt [output 0]	YUV444 YUV422 YUV420 YUV444_PLANER YUV422_PLANER YUV420_PLANER Y8 YUV420_NVX1 RGB888_PLANER (**)	YUV420 YUV420_PLANER Y8 YUV420_NVX2
OUT.pxlfmt [output 1]	YUV422 YUV420 Y8	YUV420 Y8
OUT.pxlfmt [output 2]	YUV422 YUV420 Y8	YUV420 Y8
OUT.pxlfmt [output 3]	Y8	Y8
OUT.pxlfmt [output 4]	YUV422 YUV420 Y8	YUV420 (*) YUV420_NVX2 (*)
IN_CROP [input 0]	n/a	n/a
OUT_CROP [output 0]	YES	YES
OUT_CROP [output 1]	YES	YES
OUT_CROP [output 2]	YES	YES
OUT_CROP [output 3]	YES	YES

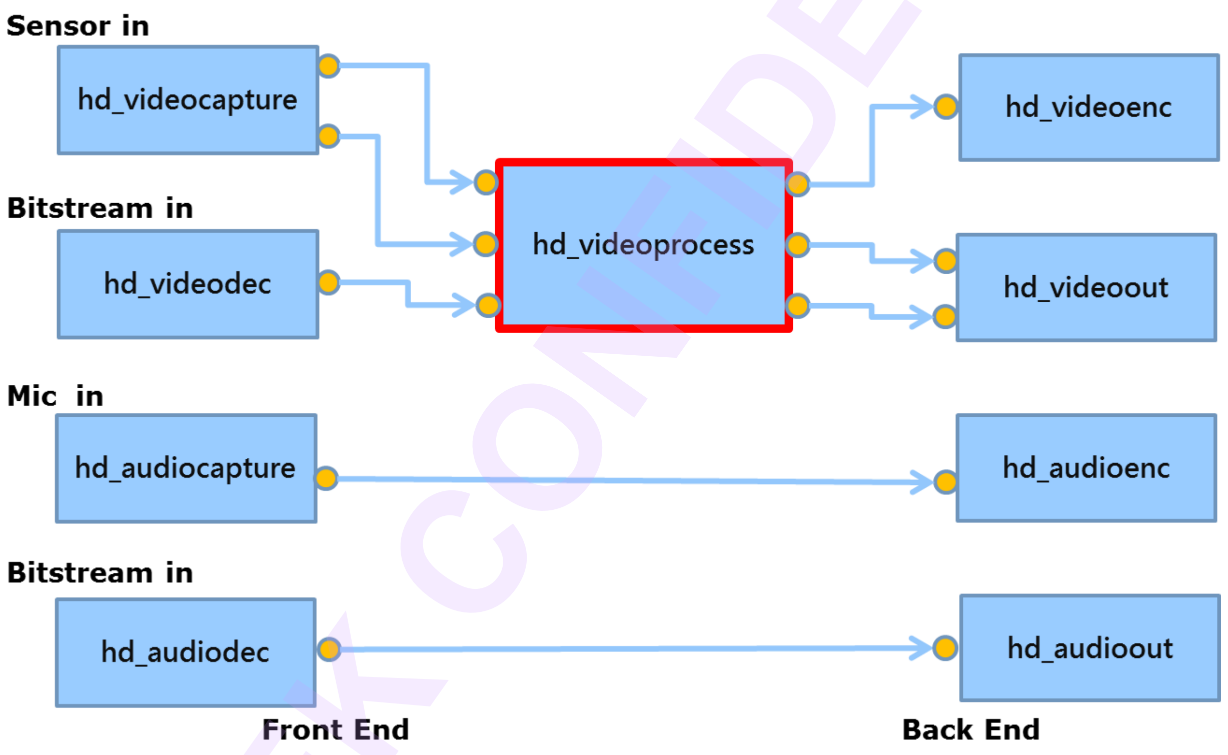
OUT_CROP [output 4]	YES	NO (*)

(*) OUT.dim.w,h is equal to IN.dim.w,h

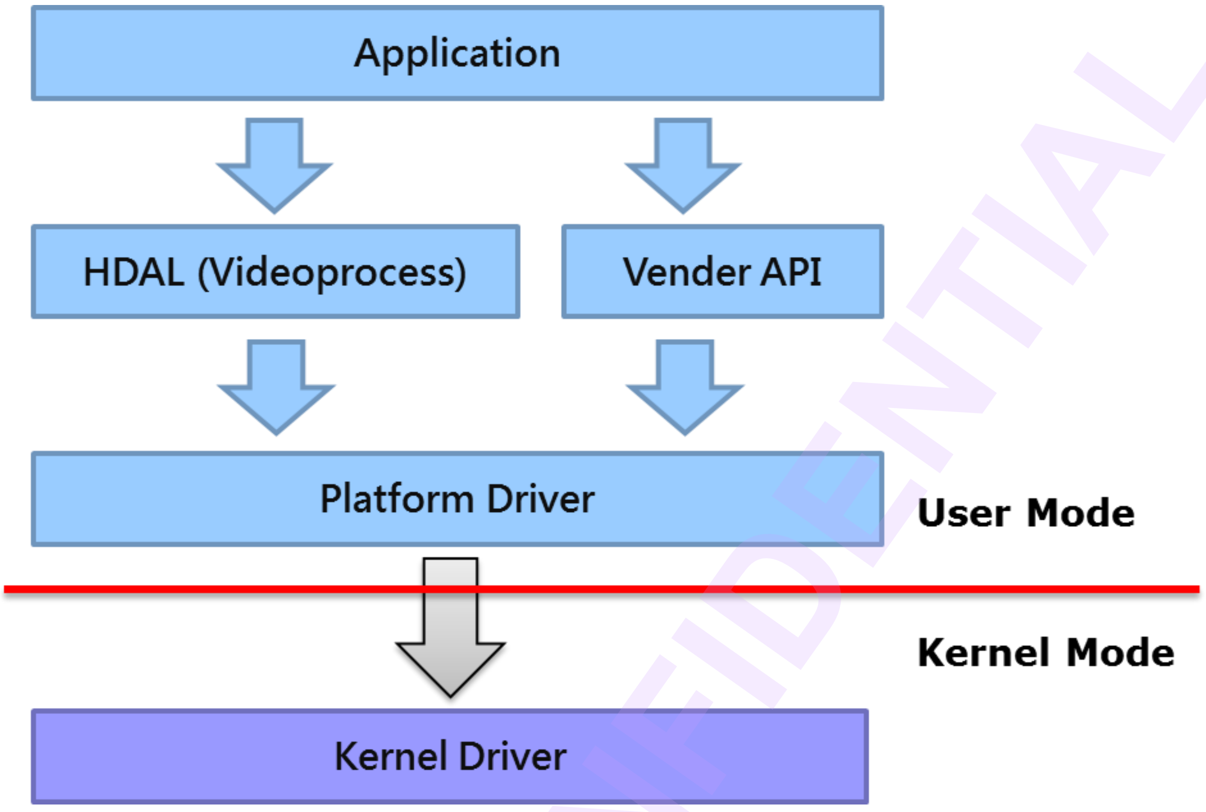
(**) if use RGB888_PLANER, only support 1 out.

1 Introduction

The major purpose of hd_videoprocess is to get YUV frame data from upper unit, and controls the video process engine to process image, including scaling, noise reduction, rotate, PIP, cropping, then return the YUV frame data which can be used for displaying and encoding. This document will talk about the red block in the following diagram. The device driver is not the main point in this document.

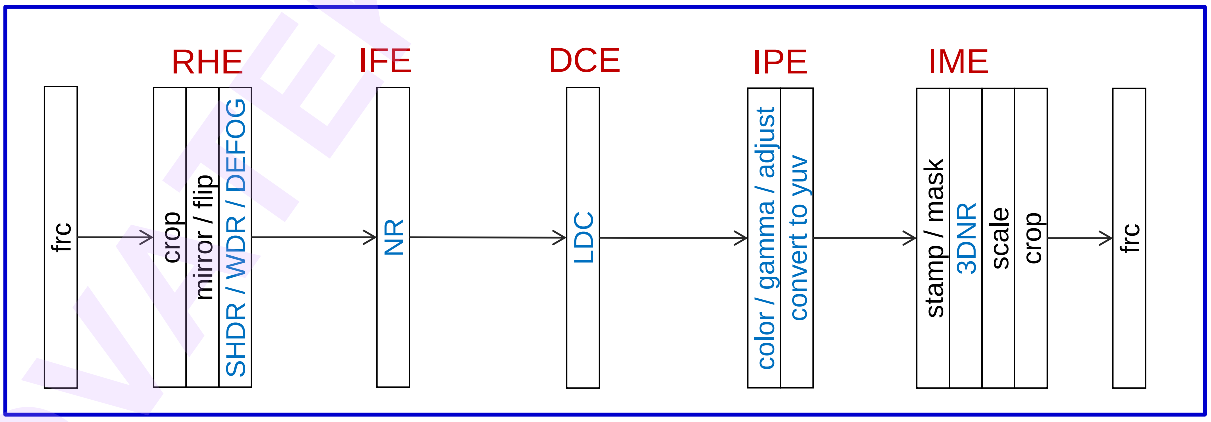


Module diagram is shown as below:



1.1 Block Diagram

1.1.1 IPC Block Diagram



NT9668X:

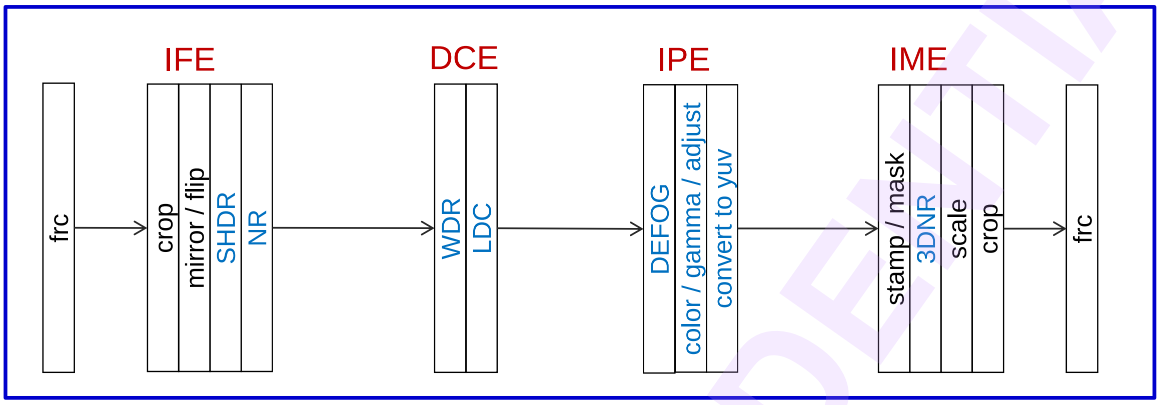
for pipe = RAWALL, it supports function of RHE/IFE/DCE/IPE/IME.

for pipe = YUVALL, it supports function of DCE/IPE/IME.

for pipe = GDC, it supports function of DCE only.

for pipe = COLOR, it supports function of IPE only.

for pipe = SCALE, it supports function of IME only.



NT9652X:

for pipe = RAWALL, it supports function of IFE/DCE/IPE/IME.

for pipe = YUVALL, it supports function of DCE/IPE/IME.

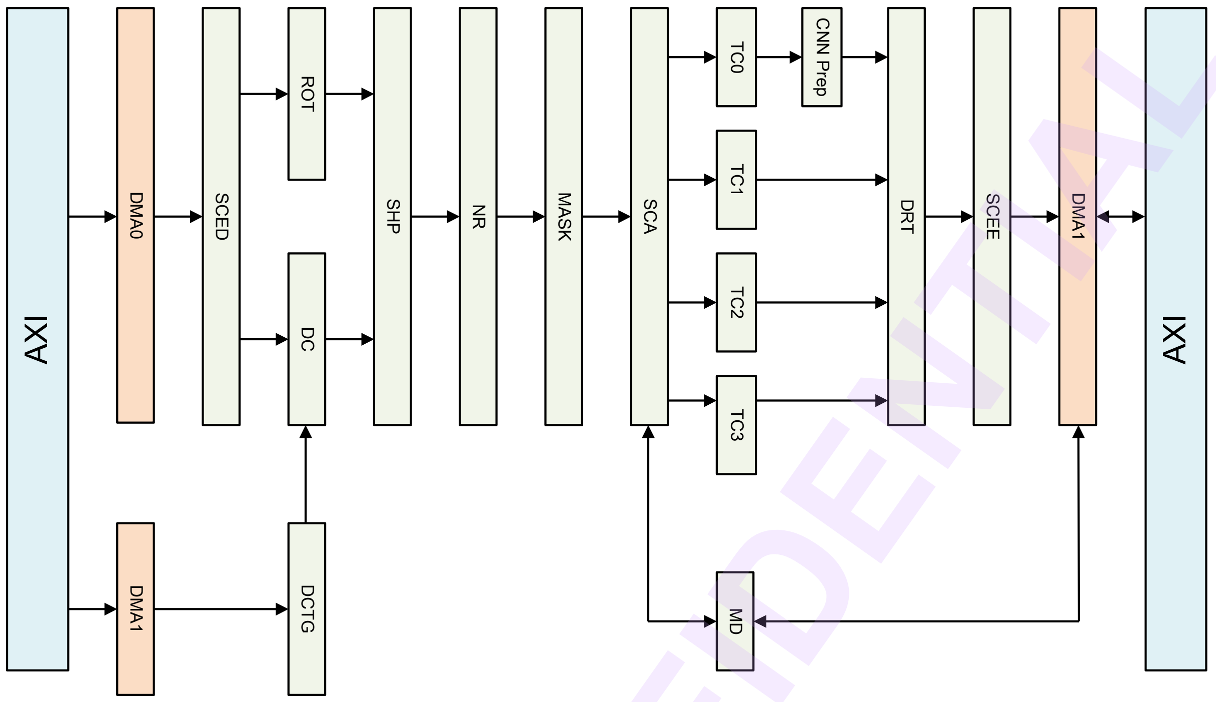
for pipe = GDC, it supports function of DCE only.

for pipe = COLOR, it supports function of IPE only.

for pipe = SCALE, it supports function of IME only.

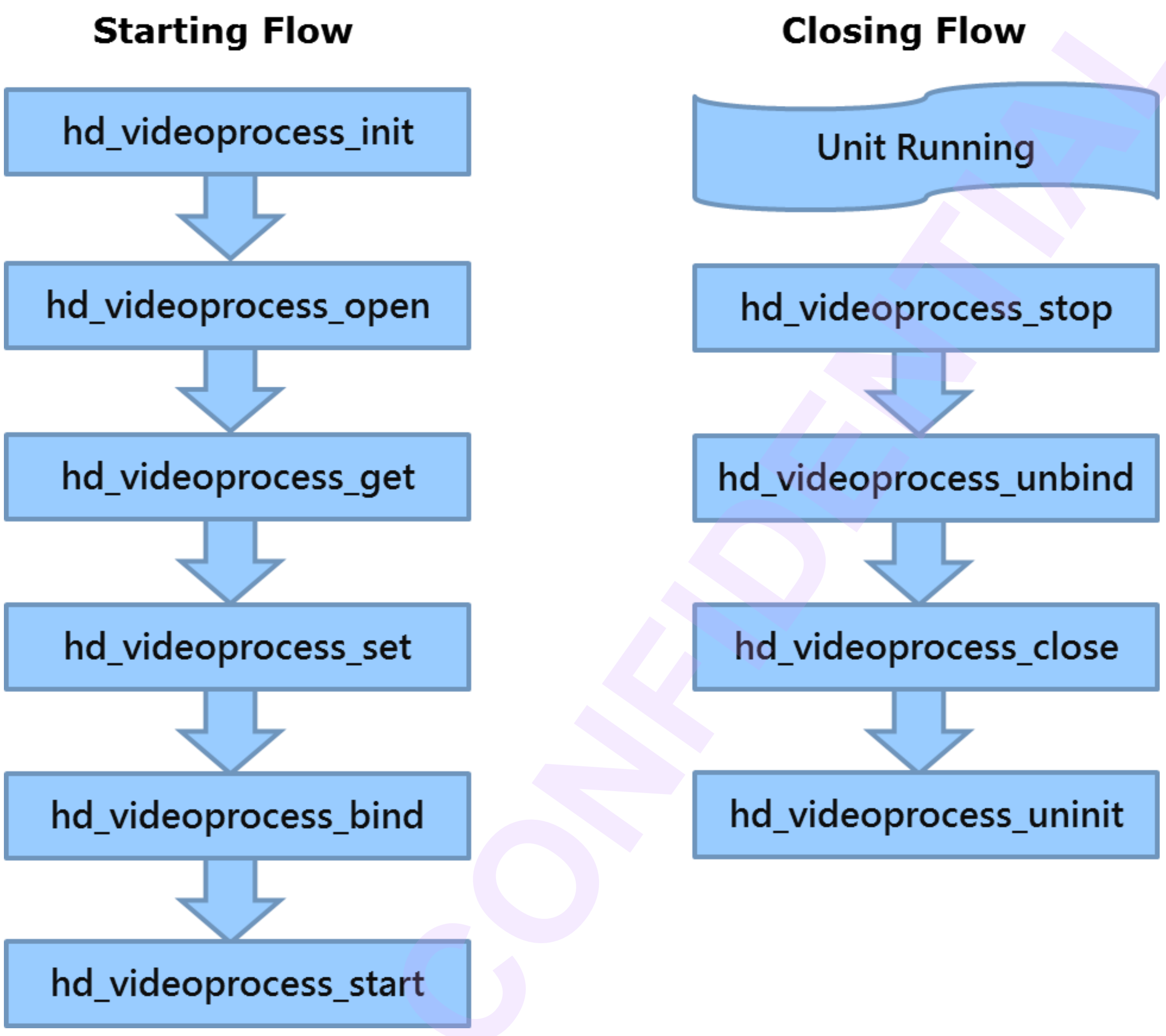
1.1.2 NVR Block Diagram

The block diagram of video process engine is shown as below:



1.2 Basic Flow

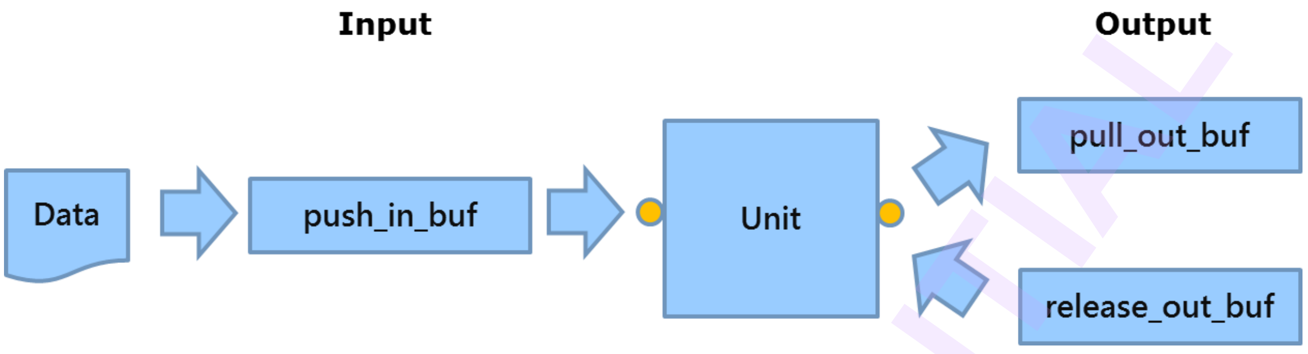
The call sequence is needed to be done correctly for the unit. The standard starting flows of most modules are init, open, get, set and start. The standard closing flows of most modules are stop, unbind, close and uninit. The basic flow is shown as below.



Now, below section in this chapter is mainly about what things to do in those functions above.

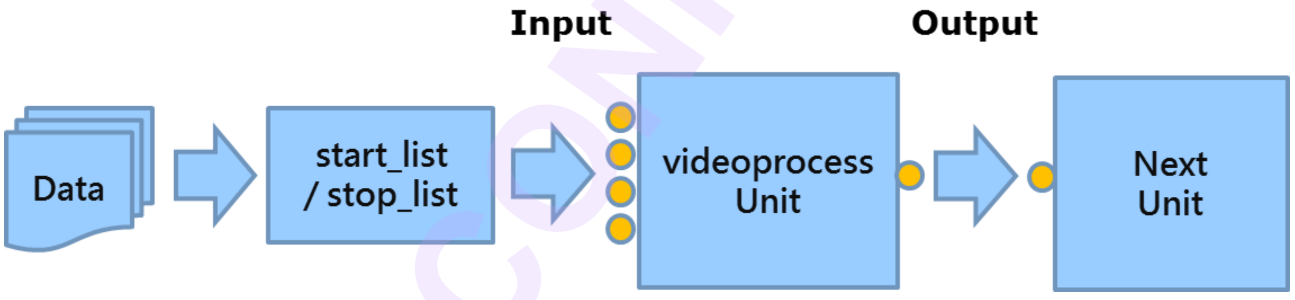
1.3 Single Trigger Operation

Single trigger operation is used to trigger the unit to do one job, such as to grab one YUV frame from video capture; or encode one frame to bitstream by using video encoder. There are two types of functions for the input port and output port. The sequence for input port is new, push and release; the sequence for output port is pull and release. The flow is shown as below.



1.4 Multi List Operation

Multi list operation is used to send mulit bitstream simultaneously, it is very efficiency in the multi channels case. The flow is shown as below:



2 Functions Definition

2.1 hd_videoprocess_init

[Description]
Initialize the unit

[Syntax]
HD_RESULT hd_videoproc_init(VOID);

[Parameter]

Value	Description
VOID	Not available

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure

2.2 hd_videoprocess_open

[Description]
Open the unit

[Syntax]
HD_RESULT hd_videoproc_open(HD_IN_ID in_id, HD_OUT_ID out_id, HD_PATH_ID* p_path_id)

[Parameter]

Value	Description
in_id	id of input port.

out_id	id of output port.
p_path_id	pointer of the path id

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure

[Note]

For OSG:

1. There are two kinds of OSG : ext and non-ext. ext poses less position limitation but consumes more CPU/DMA. ext is ideal for OSG with small resolution and high position flexibility.
2. DIS and Crop will change OSGs' position, making OSGs drift. It's recommend not to using videoprocess's OSG if DIS or Crop is enabled

2.3 hd_videoprocess_get

[Description]

Get parameters from unit by path id

[Syntax]

HD_RESULT hd_videoproc_get(HD_PATH_ID path_id,
HD_VIDEOPROC_PARAM_ID id, VOID* p_param)

[Parameter]

Value	Description
path_id	the path id
id	id of parameters
p_param	pointer of parameters

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure
HD_ERR_NOT_SUPPORT	Not support this parameter

2.4 hd_videoprocess_set

[Description]

Set parameters to unit by path id

[Syntax]

HD_RESULT hd_videoproc_set(HD_PATH_ID path_id, HD_VIDEOPROC_PARAM_ID id, VOID* p_param)

[Parameter]

Value	Description
path_id	the path id
id	id of parameters
p_param	pointer of parameters

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure
HD_ERR_NOT_SUPPORT	Not support this parameter

2.5 hd_videoprocess_bind

[Description]

Bind this unit with destination unit

[Syntax]

HD_RESULT hd_videoproc_bind(HD_OUT_ID out_id, HD_IN_ID dest_in_id)

[Parameter]

Value	Description
out_id	id of output port.
dest_in_id	id of input port.

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure

2.6 hd_videoprocess_start

[Description]
Start the unit

[Syntax]
HD_RESULT hd_videoproc_start(HD_PATH_ID path_id)

[Parameter]

Value	Description
path_id	pointer of the path id

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure

[Difference]

Chip	Description
NT9668x	All functions are supported.
NT98313	All functions are supported.

2.7 hd_videoprocess_stop

[Description]
Stop the unit

[Syntax]
HD_RESULT hd_videoproc_stop(HD_PATH_ID path_id)

[Parameter]

Value	Description
path_id	pointer of the path id

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure

2.8 hd_videoprocess_unbind

[Description]
Unbind the unit

[Syntax]
HD_RESULT hd_videoproc_open(HD_IN_ID in_id, HD_OUT_ID out_id, HD_PATH_ID* p_path_id)

[Parameter]

Value	Description
in_id	id of input port.
out_id	id of output port.
p_path_id	pointer of the path id

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure

2.9 hd_videoprocess_close

[Description]
Close the unit

[Syntax]
HD_RESULT hd_videoproc_close(HD_PATH_ID path_id)

[Parameter]

Value	Description
path_id	pointer of the path id

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure

[Note]

For OSG:

1. OSGs will keep registered buffer until they are closed. Only after this API returns can application safely access/reclaim the buffer.

2.10 hd_videoprocess_uninit

[Description]

Uninitialize the unit

[Syntax]

HD_RESULT hd_videoproc_uninit(VOID);

[Parameter]

Value	Description
VOID	Not available

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure

2.11 hd_videoprocess_push_in_buf

[Description]
Push the video buffer to unit

[Syntax]
HD_RESULT hd_videoproc_push_in_buf(HD_PATH_ID path_id, HD_VIDEO_FRAME* p_in_video_frame, HD_VIDEO_FRAME* p_user_out_video_frame, INT32 wait_ms);

[Parameter]

Value	Description
path_id	the path id
p_in_video_frame	pointer of the input video buffer
p_user_out_video_frame	pointer of the output video buffer
wait_ms	timeout value in ms

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure

2.12 hd_videoprocess_pull_out_buf

[Description]
Pull the video buffer from unit

[Syntax]
HD_RESULT hd_videoproc_pull_out_buf(HD_PATH_ID path_id, HD_VIDEO_FRAME* p_video_frame, INT32 wait_ms);

[Parameter]

Value	Description
path_id	the path id
p_video_frame	pointer of the output video buffer
wait_ms	timeout value in ms

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure

2.13 hd_videoprocess_release_out_buf

[Description]

Release the video frame buffer which is get from unit

[Syntax]

HD_RESULT hd_videoproc_release_out_buf(HD_PATH_ID path_id, HD_VIDEO_FRAME* p_video_frame)

[Parameter]

Value	Description
path_id	the path id
p_video_frame	pointer of the output video buffer

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure

2.14 hd_videoproc_start_list

[Description]

Start to send multi YUV frame to the unit

[Syntax]

HD_RESULT hd_videoproc_start_list(HD_PATH_ID *path_id, UINT num);

[Parameter]

Value	Description
path_id	the path id
num	number of YUV frame

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure

[Difference]

Chip	Description
IPC	Not supported.
NVR	All functions are supported.

2.15 hd_videoproc_stop_list

[Description]

Stop to send multi YUV frame to the unit

[Syntax]

HD_RESULT hd_videoproc_stop_list(HD_PATH_ID *path_id, UINT num);

[Parameter]

Value	Description
path_id	the path id
num	number of YUV frame

[Return Value]

Value	Description
HD_OK	Success
HD_ERR_NG	Failure

[Difference]

Chip	Description
IPC	Not supported.
NVR	All functions are supported.

3 Parameter IDs and Data Structure Definition

3.1 Parameters IDs

The videoprocess provides the following parameter IDs:

- HD_VIDEOPROC_PARAM_DEVCOUNT
  - NVR/IPC. support get with ctrl path
  - using HD_DEVCOUNT struct (device id max count)
- HD_VIDEOPROC_PARAM_SYSCAPS
  - NVR/IPC. support get with ctrl path
  - using HD_VIDEOPROC_SYSCAPS struct (system capabilitiy)
- HD_VIDEOPROC_PARAM_SYSINFO
  - NVR/IPC. support get with ctrl path
  - using HD_VIDEOPROC_SYSINFO struct (system infomation)
- HD_VIDEOPROC_PARAM_DEV_CONFIG
  - NVR/IPC only. support set with ctrl path
  - using HD_VIDEOPROC_DEV_CONFIG struct (device device config)
- HD_VIDEOPROC_PARAM_CTRL
  - NVR/IPC. support get/set with ctrl path
  - using HD_VIDEOPROC_CTRL struct (effect of whole device)
- HD_VIDEOPROC_PARAM_IN
  - IPC only. support get/set with i/o path
  - using HD_VIDEOPROC_IN struct (input frame paramter)
- HD_VIDEOPROC_PARAM_IN_FRC
  - IPC only. support get/set with i/o path
  - using HD_VIDEOPROC_FRC struct (input crop parameter)
- HD_VIDEOPROC_PARAM_IN_CROP
  - NVR/IPC. support get/set with i/o path
  - using HD_VIDEOPROC_CROP struct (input crop parameter)
- HD_VIDEOPROC_PARAM_IN_CROP_PSR
  - NVR/IPC. support get/set with i/o path
  - using HD_VIDEOPROC_CROP struct (input crop parameter)
- HD_VIDEOPROC_PARAM_OUT
  - NVR/IPC. support get/set with i/o path

- ☐ using HD_VIDEOPROC_OUT struct (output frame paramter)

- HD_VIDEOPROC_PARAM_OUT_FRC
  - ☐ IPC only. support get/set with i/o path
  - ☐ using HD_VIDEOPROC_FRC struct (output crop parameter)
- HD_VIDEOPROC_PARAM_OUT_CROP
  - ☐ NVR/IPC. support get/set with i/o path
  - ☐ using HD_VIDEOPROC_CROP struct (output crop parameter)
- HD_VIDEOPROC_PARAM_OUT_EX
  - ☐ IPC only. support get/set with i/o path
  - ☐ using HD_VIDEOPROC_OUT_EX struct (output frame paramter)
- HD_VIDEOPROC_PARAM_OUT_EX_CROP
  - ☐ IPC only. support get/set with i/o path
  - ☐ using HD_VIDEOPROC_CROP struct (output crop parameter)
- HD_VIDEOPROC_PARAM_IN_STAMP_BUF
  - ☐ IPC only. support set with i/stamp path
  - ☐ using HD_OSG_STAMP_BUF struct (stamp buffer parameter)
- HD_VIDEOPROC_PARAM_IN_STAMP_IMG
  - ☐ IPC only. support set with i/stamp path
  - ☐ using HD_OSG_STAMP_IMG struct (stamp image parameter)
- HD_VIDEOPROC_PARAM_IN_STAMP_ATTR
  - ☐ IPC only. support get/set with i/stamp path
  - ☐ using HD_OSG_STAMP_ATTR struct (stamp display attribute)
- HD_VIDEOPROC_PARAM_IN_MASK_ATTR
  - ☐ IPC only. support get/set with i/mask path
  - ☐ using HD_OSG_MASK_ATTR struct (mask display attribute)
- HD_VIDEOPROC_PARAM_IN_MOSAIC_ATTR
  - ☐ IPC only. support get/set with i/mask path
  - ☐ using HD_OSG_MOSAIC_ATTR struct (mosaic display attribute)
- HD_VIDEOPROC_PARAM_PATTERN_IMG
  - ☐ NVR only. support get/set with ctrl path
  - ☐ using HD_VIDEOPROC_PATTERN_IMG struct (pattern parameter)
- HD_VIDEOPROC_PARAM_PATTERN_SELECT
  - ☐ NVR only. support get/set with ctrl path
  - ☐ using HD_VIDEOPROC_PATTERN_SELECT struct (pattern parameter)
- HD_VIDEOPROC_PARAM_VPEMASK_ATTR
  - ☐ NVR only. support get/set with ctrl path
  - ☐ using HD_VIDEOPROC_VPEMASK_ONEINFO struct (pattern parameter)

- HD_VIDEOPROC_PARAM_SCA_WK_BUF
  - NVR only. support get/set with ctrl path
  - using HD_VIDEOPROC_SCA_BUF_INFO struct (pattern parameter)

☐ Data Structure Definition

3.1.1 HD_VIDEOPROC_SYSCAPS

[Description]

System capability

[Parameter]

Value	Description
dev_id	device id
chip_id	chip id of this device
max_in_count	max count of input of this device
max_out_count	max count of output of this device
dev_caps	capability of device, combine caps of HD_DEVICE_CAPS and HD_VIDEOPROC_DEVCAPS
in_caps	capability of input, combine caps of HD_VIDEO_CAPS and HD_VIDEOPROC_INCAPS
out_caps	capability of output, combine caps of HD_VIDEO_CAPS and HD_VIDEOPROC_OUTCAPS
max_w_scaleup_ratio	max scaling up ratio of width
max_w_scaledown_ratio	max scaling down ratio of width
max_h_scaleup_ratio	max scaling up ratio of height
max_h_scaledown_ratio	max scaling down ratio of height
max_in_stamp	max in stamp
max_in_stamp_ex	max in stamp extension
max_in_mask	max in mask
max_in_mask_ex	max in mask externsion

3.1.2 HD_VIDEOPROC_SYSINFO

[Description]

System information

[Parameter]

Value	Description
dev_id	device id
cur_in_fps	current input fps
cur_out_fps	current output fps

3.1.3 HD_VIDEOPROC_DEV_CONFIG

[Description]
Device configuration

[Parameter]

Value	Description
pipe	IPC only. pipeline setting
iq_id	IPC only. IQ id
ctrl_max	IPC only. maximum control settings
in_max	IPC only. maximum input settings
data_pool	NVR only. pool memory information

3.1.4 HD_VIDEOPROC_CTRL

[Description]
Control function

[Parameter]

Value	Description
func	additional function of HD_CTRL (whole device)

3.1.5 HD_VIDEOPROC_IN

[Description]
Input frame

[Parameter]

Value	Description
func	IPC only. additional function of in
dim	IPC only. input dim w,h
pxlfmt	IPC only. input pixel format
dir	IPC only. output direction like mirror/flip
frc	IPC only. input frame-control

3.1.6 HD_VIDEOPROC_CROP

[Description]

input crop or output crop

[Parameter]

Value	Description
mode	NVR/IPC. crop mode
win	NVR/IPC. crop window x,y,w,h

3.1.7 HD_VIDEOPROC_FRC

[Description]

input crop or output crop

[Parameter]

Value	Description
frc	IPC. frame rate control

3.1.8 HD_VIDEOPROC_OUT

[Description]

Physical output frame

[Parameter]

Value	Description

func	IPC only. additional function of in
dim	IPC only. input dim w,h
pxlfmt	IPC only. input pixel format
dir	IPC only. output direction like mirror/flip
frc	IPC only. input frame-control

3.1.9 HD_VIDEOPROC_OUT_EX

[Description]
Extended output frame

[Parameter]

Value	Description
src_id	IPC only. select one of physical out
dim	IPC only. input dim w,h
pxlfmt	IPC only. input pixel format
frc	IPC only. input frame-control

3.1.10 HD_VIDEOPROC_PARAM_IN_STAMP_BUF

[Description]
Stamp buffer settings

[Parameter]

Value	Description
type	NVR/IPC. ping pong buffer or single buffer, using HD_OSG_BUF_TYPE
size	NVR/IPC. buffer's size in byte
p_addr	NVR/IPC. buffer's physical address
ddr_id	NVR only. p_addr's ddr id

[Note]
For IPCam:
1. Different OSGs can share the same buffer to save memory
2. Double buffer requires "2 * max OSG resolution * sizeof(short)" while single buffer

requires only “max OSG resolution* sizeof(short)”. But single buffer suffers from blinking when image is updated.

3. The starting address and length should be 4bytes aligned.
4. In D2D mode, only stamp 1 and 4 are available

3.1.11 HD_VIDEOPROC_PARAM_IN_STAMP_IMG

[Description]
Stamp image settings

[Parameter]

Value	Description
fmt	NVR/IPC. RGB565/ARGB1555/ARGB4444/ARGB8888/ Using HD_VIDEO_PXLFMT struct
dim	NVR/IPC. image's width and height, using HD_DIM struct
p_addr	NVR/IPC. image's bitmap content
ddr_id	NVR only. p_addr's ddr id

[Note]

For IPCam:

1. Only RGB565/ARGB1555/ARGB4444 are supported
2. Image width and height are best to be multiple of 2 for best compatibility.
3. In addition to the whole image width and height, every color area(e.g. timestamp and border)'s width and height should be multiple of 2.
4. hd_videoprocess_get retrieves free buffer(not accessed by hardware) for OSG of ping pong buffer
5. In D2D mode, only stamp 1 and 4 are available

3.1.12 HD_VIDEOPROC_PARAM_IN_STAMP_ATTR

[Description]

Stamp attr settings

[Parameter]

Value	Description
align_type	NVR only. to which corner is stamp aligned Using HD_OSG_ALIGN_TYPE struct
alpha	NVR/IPC. (DISP)alpha value
position	NVR/IPC. (DISP)stamp's x,y position, using HD_IPOINT struct
colorkey_en	IPC only. is colorkey used to filter background
colorkey_val	IPC only. filtered background color
qp_en	Not used
qp_fix	Not used

qp_val	Not used
layer	Not used
region	Not used
gcac_enable	Not used
gcac_blk_width	Not used
gcac_blk_height	Not used

[Note]
For IPCam:

1. align_type is not supported
2. For ARGB4444, alpha field is not applicable. For ARGB1555. alpha[3..0] is for pixels of A = 0 and alpha[7..4] is for pixels of A = 1.
3. X y are best to be multiple of 2 for best compatibility
4. In D2D mode, only stamp 1 and 4 are available

3.1.13 HD_VIDEOPROC_PARAM_IN_MASK_ATTR

[Description]
Mask attribute settings

[Parameter]

Value	Description
type	NVR/IPC. mask is solid or hollow. Using HD_OSG_MASK_TYPE
color	IPC. mask color in rgb, NVR.mask palette index
alpha	NVR/IPC. mask transparency
position	NVR/IPC. 4 vertices' position, using HD_UPOINT struct
thickness	IPC only. border width for hollow mask

[Note]
For IPCam:

1. position[0] should be the top left. Others should be in clockwise order.
2. thickness should be multiple of 2
3. Hollow mask takes more time to complete than solid mask. Don't set over 4 hollow masks in a path.

3.1.14 HD_VIDEOPROC_PARAM_IN_MOSAIC_ATTR

[Description]
Mosaic attribute settings

[Parameter]

Value	Description
Type	NVR only. mask is solid or inversion. Using HD_OSG_MASK_TYPE struct
Alpha	NVR only. mask alpha blending. range: 0 ~ 256 (0: foreground, 256: background)
mosaic_blk_w	NVR/IPC. witdh of internal block
mosaic_blk_h	NVR/IPC. height of internal block
position	NVR/IPC. 4 vertices' position, using HD_UPOINT struct

[Note]
For IPCam:
1. Block size supports 8*8, 16*16, 32*32, 64*64
2. All mosaics in a frame must use the same block size
3. position[0] should be the top left. Others should be in clockwise order.
4. HD_VIDEOPROC_FUNC_MOSAIC must be turned on.

3.1.15 HD_VIDEOPROC_PATTERN_IMG

[Description]
Pattern image setting

[Parameter]

Value	Description
index	NVR only. pattern index
image	NVR only. pattern image

3.1.16 HD_VIDEOPROC_PATTERN_SELECT

[Description]
Pattern image select

[Parameter]

Value	Description
index	NVR only. pattern index select, set VPE_PATTERN_SEL_DIABLE to disable
rect	NVR only. destination window ratio (0 ~ 100)
bg_color_sel	NVR only. background color select, using HD_VIDEOPROC_PALETTE

3.1.17 HD_VIDEOPROC_VPEMASK_ONEINFO

[Description]
Mask info

[Parameter]

Value	Description
index	NVR only. pattern index, 0 to disable
mask_idx	index = priority 0>1>2>3>4>5>6>7
mask_area	0:inside, 1:outside, 2:line
point	NVR only. 4 point
mosaic_en	use original image or mosaic image in mask area
alpha	alpha blending 0~256, only effect at bitmap = 0,1

3.1.18 HD_VIDEOPROC_SCA_BUF_INFO

[Description]
SCA buffer info

[Parameter]

Value	Description
ddr_id	ID of DDR

pbuf_addr	Physical address of buffer
pbuf_size	Size of buffer

4 Trouble shooting

The hd_videoprocess provides a useful feature to debug, it is called debug menu.

4.1 Proc Command of IPC

4.1.1 dump info

User can cat info file to dump module’s status.

```
------------------------- VIDEOPROC 0  PATH & BIND ----------------------------
in    out   state bind_src           bind_dest
0     0     START VIDEOCAP_0_OUT_0     VIDEOOUT_0_IN_0
------------------------- VIDEOPROC 0  DEV CONFIG ----------------------------
mode  pipe   isp_id
      RAWALL 0
ctrl_max    func 3dnr_ref
            00000000 0
in_max   w     h     pxlfmt
         1920  1080  RAW12
------------------------- VIDEOPROC 0  CTRL ----------------------------------
ctrl    func 3dnr_ref
        00000000 0
------------------------- VIDEOPROC 0  IN FRAME ------------------------------
in    w     h     pxlfmt  frc   dir   crop
0     1920  1080 0      1/1   ....  OFF:{0,0,0,0}
------------------------- VIDEOPROC 0  OUT FRAME -----------------------------
out   w     h     pxlfmt  frc   dir   crop
0     1920  1080 0      1/1   ....  OFF:{0,0,0,0}
------------------------- VIDEOPROC 0  IN WORK STATUS -------------------------
in    PUSH  drop  wrn   err   PROC  drop  wrn   err   REL
0     30    0     0     0     30    0     0     0     30
------------------------- VIDEOPROC 0  OUT WORK STATUS ------------------------
out   NEW   drop  wrn   err   PROC  drop  wrn   err   PUSH  drop  wrn   err
```

```
0     30     0     0     0     30    0     0     0     30     0     0     0

------------------------ VIDEOPROC 0  USER WORK STATUS -----------------------

out   PULL  drop  wrn   err   REL

0     30     0     0     0     0
```

[PATH & BIND]

Status	Description	Value
in	input id of path	0
out	output id of path	0 ~ [max_out_count]
state	state of path	OFF/OPEN/START (default OFF)
bind_src	current binding source of input	bind: [module]_[device_id]_OUT_[output_id] not-bind: (null)
bind_dest	current binding source of output	bind: [module]_[device_id]_IN_[input_id] not-bind: (null)

[DEV CONFIG]

Value	Description	value
pipe	current pipe mode	OFF: (none) RAWALL: all raw 2 yuv process YUVALL: all yuv 2 yuv process COLOR: color only process SCALE: scaling only process
isp_id	current isp id	0 ~ 15
func	maximum function combination	see enum in HD_VIDEOPROC_CTRLFUNC
3dnr_ref	3dnr reference path	(don't care)
w	maximum input dimension width	16 ~ 65532: user assign width default 0 (n/a)
h	maximum input dimension height	16 ~ 65532: user assign width default 0 (n/a)
pxlfmt	maximum input pixel format	enum: user assign pixel format see HD_VIDEO_PXLFMT default 0 (n/a)

[CTRL]

Value	Description	value
func	current function	see enum in HD_VIDEOPROC_CTRLFUNC
3dnr_ref	current 3dnr reference path	0 ~ 4

[CONFIG]

Value	Description	value
max	maximum input id	0
w	maximum input dimension width	16 ~ 65532: user assign width default 0 (n/a)
h	maximum input dimension height	16 ~ 65532: user assign width default 0 (n/a)
pxlfmt	maximum input pixel format	enum: user assign pixel format see HD_VIDEO_PXLFMT default 0 (n/a)

[IN FRAME]

Value	Description	value
in	input id	0
w	current input dimension width	0: auto reference 16 ~ 65532: user assign width default 0 (auto reference)
h	current input dimension height	0: auto reference 16 ~ 65532: user assign height default 0 (auto reference)
pxlfmt	current input pixel format	0: auto reference enum: user assign pixel format see HD_VIDEO_PXLFMT default 0 (auto reference)
frc	current input frame control (down sampling)	[dest_frame_rate]/[src_frame_rate] see HD_VIDEO_FRC default 1/1 (disable)
dir	current input direction	.: (none)

		X: mirror Y: flip R: rotate 90 degree L: rotate 270 degree see HD_VIDEO_DIR default: . (none)

crop	current input crop mode and range	OFF/ON
		{x,y,w,h}
		default: OFF,{0,0,0,0}

[OUT FRAME]

Value	Description	value
out	output id	0
w	current output dimension width	0: auto reference 16 ~ 65532: user assign width default 0 (auto reference)
h	current output dimension height	0: auto reference 16 ~ 65532: user assign height default 0 (auto reference)
pxlfmt	current output pixel format	0: auto reference others: user assign pixel format see HD_VIDEO_PXLFMT default 0 (auto reference)
frc	current output frame control (down sampling)	[dest_frame_rate]/[src_frame_rate] see HD_VIDEO_FRC default 1/1 (disable)
dir	current output direction	.: (none) X: mirror Y: flip R: rotate 90 degree L: rotate 270 degree see HD_VIDEO_DIR default: . (none)

crop	current output crop mode and range	OFF/ON/AUTO
		{x,y,w,h}
		default: OFF,{0,0,0,0}

[IN WORK STATUS]

Value	Description	value
in	input id	0
PUSH	start job counter of input push stage (per second)	
drop	dropping job counter of input push stage (per second)	
wrn	cancel by warning job counter of input push stage (per second)	
err	cancel by error job counter of input push stage (per second)	
PROC	start job counter of input process stage (per second)	
drop	dropping job counter of input process stage (per second)	
wrn	cancel by warning job counter of input process stage (per second)	
err	cancel by error job counter of input process stage (per second)	
REL	finish job counter of input release stage (per second)	

[OUT WORK STATUS]

Value	Description	value
out	output id	0~15
NEW	start job counter of output new stage (per second)	
drop	dropping job counter of output new stage (per second)	
wrn	cancel by warning job counter of output new stage (per second)	
err	cancel by error job counter of output new stage (per second)	
PROC	start job counter of output process	

	stage (per second)	
drop	dropping job counter of output process stage (per second)	
wrn	cancel by warning job counter of output process stage (per second)	
err	cancel by error job counter of output process stage (per second)	
PUSH	start job counter of output push stage (per second)	
drop	dropping job counter of output push stage (per second)	
wrn	cancel by warning job counter of output push stage (per second)	
err	cancel by error job counter of output push stage (per second)	

[USER WORK STATUS]

Value	Description	value
out	output id	0~15
PULL	start job counter of user pull stage (per second)	
skip	skipping job counter of user pull stage (per second)	
wrn	cancel by warning job counter of user pull stage (per second)	
err	cancel by error job counter of user pull stage (per second)	
REL	finish job counter of user release stage (per second)	

4.1.2 debug command

User can enter cmd to show "debug log" of path binding, state changing, and parameter setting log.

NOTE: For example, this will enable log of device 0 and output 0 with all actions..

```
$ echo debug d0 p0 mfff > /proc/hdal/vprc/cmd


root@NVTEVM:~$ hd_video_liveview 2


HDAL_VERSION: 00010001:00010001
[  925.382091]  "vdoprc0".out[0]: cmd OPEN
[  925.388869]  "vdoprc0".out[0]: cmd CONNECT

[  925.394718]  "vdoprc0".out[0]: cmd RDYSYNC
[  925.563696]  "vdoprc0".out[0]: cmd RDYSYNC
[  925.568708]  "vdoprc0".out[0]: cmd START
Enter q to exit
q
[  935.563274]  "vdoprc0".out[0]: cmd STOP
[  935.583352]  "vdoprc0".out[0]: cmd DISCONNECT
[  935.595161]  "vdoprc0".out[0]: cmd CLOSE
```

NOTE: For example, this will stop “debug log”..

```
$ echo debug d0 p0 > /proc/hdal/vprc/cmd
[935.595971] debug i/o end
```

4.1.3 trace command

User can enter cmd to show “trace log” of detail state changing, and detail parameter setting log.
NOTE: For example, this will enable log of device 0 and output 0 with all actions..

```
$ echo trace d0 p0 mfff > /proc/hdal/vprc/cmd


root@NVTEVM:~$ hd_video_liveview 2


(TBD)
```

NOTE: For example, this will stop “trace log”..

```
$ echo trace d0 p0 > /proc/hdal/vprc/cmd
```

```
(TBD)
```

4.1.4 probe command

User can enter cmd to show continuous “probe log” of new, add, release and other action log of each processing data.

NOTE: For example, this will enable log of device 0 and output 0 with all actions..

```
$ echo probe d0 p0 mffff > /proc/hdal/vprc/cmd


[  350.024501] probe i/o begin: "vdoprc0".out[0], action mask=0xffff
[  350.034948] "vdoprc0".out[0] - NEW - new -- h=9496bfc0 size=002f7600 addr=9496c000 OK
[  350.050397] "vdoprc0".out[0] - NEW - add -- h=9496bfc0 (result=0) OK
[  350.057786] "vdoprc0".out[0] push! -- h=9496bfc0 t=0000000015ec6118 (YUV:
1920x1080.520c0420 9496c000 94b66400 1920 1920)
[  350.070447] "vdoprc0".out[0] - PUSH - rel -- h=9496bfc0 (result=0) OK
[  350.077937] "vdoprc0".out[0] - NEW - new -- h=94f5bfc0 size=002f7600 addr=94f5c000 OK
[  350.093595] "vdoprc0".out[0] - NEW - add -- h=94f5bfc0 (result=0) OK
[  350.101074] "vdoprc0".out[0] - NEW - new -- h=94c63fc0 size=002f7600 addr=94c64000 OK
[  350.110690] "vdoprc0".out[0] push! -- h=94f5bfc0 t=0000000015ece348 (YUV:
1920x1080.520c0420 94f5c000 95156400 1920 1920)
[  350.122684] "vdoprc0".out[0] - PUSH - rel -- h=94f5bfc0 (result=0) OK
[  350.130125] "vdoprc0".out[0] - NEW - add -- h=94c63fc0 (result=0) OK
[  350.137579] "vdoprc0".out[0] - NEW - new -- h=9496bfc0 size=002f7600 addr=9496c000 OK
[  350.147291] "vdoprc0".out[0] push! -- h=94c63fc0 t=0000000015ed6a04 (YUV:
1920x1080.520c0420 94c64000 94e5e400 1920 1920)
[  350.159233] "vdoprc0".out[0] - PUSH - rel -- h=94c63fc0 (result=0) OK
[  350.166668] "vdoprc0".out[0] - NEW - add -- h=9496bfc0 (result=0) OK
[  350.174134] "vdoprc0".out[0] - NEW - new -- h=94f5bfc0 size=002f7600 addr=94f5c000 OK
[  350.183710] "vdoprc0".out[0] push! -- h=9496bfc0 t=0000000015ee0978 (YUV:
1920x1080.520c0420 9496c000 94b66400 1920 1920)
[  350.195640] "vdoprc0".out[0] - PUSH - rel -- h=9496bfc0 (result=0) OK
[  350.203200] "vdoprc0".out[0] - NEW - new -- h=94c63fc0 size=002f7600 addr=94c64000 OK
[  350.212759] "vdoprc0".out[0] - NEW - add -- h=94f5bfc0 (result=0) OK
  :
```

NOTE: For example, this will stop “probe log”..

```
$ echo probe d0 p0 > /proc/hdal/vprc/cmd
[350.212788] probe i/o end
```

4.1.5 perf command

User can enter cmd to show continuous "perf log" of each second.
NOTE: For example, this will enable log of device 0 and output 0 with all actions..

```
$ echo probe d0 p0 mfff > /proc/hdal/vprc/cmd

[  104.850315] perf i/o begin: "vdoprc0".out[0]
[  104.875120] "vdoprc0".out[0] Perf! -- (Video) 0 Frame/sec
[  105.875083] "vdoprc0".out[0] Perf! -- (Video) 30 Frame/sec
[  106.908364] "vdoprc0".out[0] Perf! -- (Video) 30 Frame/sec
[  107.941653] "vdoprc0".out[0] Perf! -- (Video) 30 Frame/sec
[  108.941722] "vdoprc0".out[0] Perf! -- (Video) 30 Frame/sec
[  109.975101] "vdoprc0".out[0] Perf! -- (Video) 30 Frame/sec
[  111.008434] "vdoprc0".out[0] Perf! -- (Video) 30 Frame/sec
[  112.041660] "vdoprc0".out[0] Perf! -- (Video) 30 Frame/sec
[  113.075091] "vdoprc0".out[0] Perf! -- (Video) 30 Frame/sec
  :
```

NOTE: For example, this will stop "perf log"..

```
$ echo perf d0 p0 > /proc/hdal/vprc/cmd
[  113.895742] perf i/o end
```

4.1.6 save command

User can enter cmd to "save data to file" of current processing data.
NOTE: For example, this will save data of device 0 and output 0..

```
$ echo save d0 p0 > /proc/hdal/vprc/cmd

[  128.595608] save i/o begin: "vdoprc0".out[0] count=1
[  128.608674] "vdoprc0".out[0] Save -- h=94c63fc0 t=0000000008b99320 (YUV:
1920x1080.520c0420 94c64000 94e5e400 1920 1920)
[  128.839831] "vdoprc0".out[0] Save -- //mnt//sd//isf_
vdoprc0_out[0]_520c0420_1920_1080_1920_c2602.vdo ok
```

```
[  128.850336] save port end
```

NOTE: The saved vdo file will be a 1920x1080 YUV420 data.

4.2 Debug Menu for IPC

4.2.1 dump info

After enter debug menu, select 7 to enter this module’s sub-menu.
User can select 1 to dump module’s status, just like dump info results of proc command.

4.3 Proc Command of NVR

4.3.1 dump setting

User can cat info file to dump module's status.

```
root@NVTEVM:~$ cat /proc/videograph/hdal_setting
========================= VIDEOPROC 0  PATH & BIND ================================
in      out     state   bind_src              bind_dest
0       0       START   VIDEOCAP_0_OUT_0       VIDEOOUT_0_IN_0
------------------------- VIDEOPROC 0  OUT FRAME -----------------------------
x       y       w       h       bg_w    bg_h
0       0       960     540     1920    1080
pxlfmt
YUV422_ONE
------------------------- VIDEOPROC 0  CONTROL -----------------------------
de-interlace
OFF
------------------------- VIDEOPROC 0  DEV POOL -----------------------------
out     pool    ddr_id  count   max_count
0       0       0       3.0     3.0


========================= VIDEOPROC 1  PATH & BIND ================================
in      out     state   bind_src              bind_dest
0       0       START   VIDEOCAP_1_OUT_0       VIDEOOUT_0_IN_1
------------------------- VIDEOPROC 1  OUT FRAME -----------------------------
x       y       w       h       bg_w    bg_h
960     0       960     540     1920    1080
pxlfmt
YUV422_ONE
------------------------- VIDEOPROC 1  CONTROL -----------------------------
de-interlace
OFF
------------------------- VIDEOPROC 1  DEV POOL -----------------------------
out     pool    ddr_id  count   max_count
0       0       0       3.0     3.0
```

```
========================= VIDEOPROC 2  PATH & BIND ================================
in      out     state   bind_src              bind_dest
0       0       START   VIDEOCAP_2_OUT_0       VIDEOOUT_0_IN_2
------------------------- VIDEOPROC 2  OUT FRAME ----------------------------
x       y       w       h       bg_w    bg_h
0       540     960     540     1920    1080
pxlfmt
YUV422_ONE
------------------------- VIDEOPROC 2  CONTROL ----------------------------
de-interlace
OFF
------------------------- VIDEOPROC 2  DEV POOL ----------------------------
out     pool    ddr_id  count   max_count
0       0       0       3.0     3.0


========================= VIDEOPROC 3  PATH & BIND ================================
in      out     state   bind_src              bind_dest
0       0       START   VIDEOCAP_3_OUT_0       VIDEOOUT_0_IN_3
------------------------- VIDEOPROC 3  OUT FRAME ----------------------------
x       y       w       h       bg_w    bg_h
960     540     960     540     1920    1080
pxlfmt
YUV422_ONE
------------------------- VIDEOPROC 3  CONTROL ----------------------------
de-interlace
OFF
------------------------- VIDEOPROC 3  DEV POOL ----------------------------
out     pool    ddr_id  count   max_count
0       0       0       3.0     3.0


========================= VIDEOPROC 4  PATH & BIND ================================
in      out     state   bind_src              bind_dest
0       0       OPEN    -                     -
------------------------- VIDEOPROC 4  OUT FRAME ----------------------------
x       y       w       h       bg_w    bg_h
0       0       0       0       0       0
pxlfmt
```

```
-
------------------------ VIDEOPROC 4  CONTROL ----------------------------
de-interlace
OFF
------------------------ VIDEOPROC 4  DEV POOL ----------------------------
out     pool    ddr_id  count   max_count
```

[PATH & BIND]

Status	Description	Value
in	input id of path	0
out	output id of path	0 ~ [max_out_count]
state	state of path	OFF/OPEN/START (default OFF)
bind_src	current binding source of input	bind: [module]_[device_id]_OUT_[output_id] not-bind: (null)
bind_dest	current binding source of output	bind: [module]_[device_id]_IN_[input_id] not-bind: (null)

[CTRL]

Value	Description	value
func	current function	see enum in HD_VIDEOPROC_CTRLFUNC
3dnr_ref	current 3dnr reference path	0 ~ 4

[CONFIG]

Value	Description	value
max	maximum input id	0
w	maximum input dimension width	16 ~ 65532: user assign width default 0 (n/a)
h	maximum input dimension height	16 ~ 65532: user assign width default 0 (n/a)
pxlfmt	maximum input pixel format	enum: user assign pixel format see HD_VIDEO_PXLFMT default 0 (n/a)

[OUT FRAME]

Value	Description	value
out	output id	0
w	current output dimension width	0: auto reference 16 ~ 65532: user assign width default 0 (auto reference)
h	current output dimension height	0: auto reference 16 ~ 65532: user assign height default 0 (auto reference)
pxlfmt	current output pixel format	0: auto reference others: user assign pixel format see HD_VIDEO_PXLFMT default 0 (auto reference)
frc	current output frame control (down sampling)	[dest_frame_rate]/[src_frame_rate] see HD_VIDEO_FRC default 1/1 (disable)
dir	current output direction	.: (none) X: mirror Y: flip R: rotate 90 degree L: rotate 270 degree see HD_VIDEO_DIR default: . (none)

crop	current output crop mode and range	OFF/ON/AUTO
		{x,y,w,h}
		default: OFF,{0,0,0,0}

4.4 Debug Menu for NVR

4.4.1 dump info

After enter debug menu, select 07 to enter this module's sub-menu.
User can select 01 to dump module's status shown as below.

```
========================= VIDEOPROC 0  PATH & BIND ================================

in      out     state   bind_src                bind_dest

0       0       START   VIDEOCAP_0_OUT_0        VIDEOOUT_0_IN_0

------------------------- VIDEOPROC 0  OUT FRAME -----------------------------

x       y       w       h       bg_w    bg_h

0       0       1920    1080    3840    2160

pxlfmt

YUV422_ONE

------------------------- VIDEOPROC 0  CONTROL -----------------------------

de-interlace

OFF

------------------------- VIDEOPROC 0  DEV POOL -----------------------------

out     pool    ddr_id  count   max_count

0       0       0       3.0     3.0


========================= VIDEOPROC 1  PATH & BIND ================================

in      out     state   bind_src                bind_dest

0       0       START   VIDEOCAP_1_OUT_0        VIDEOOUT_0_IN_1

------------------------- VIDEOPROC 1  OUT FRAME -----------------------------

x       y       w       h       bg_w    bg_h

1920    0       1920    1080    3840    2160

pxlfmt

YUV422_ONE

------------------------- VIDEOPROC 1  CONTROL -----------------------------

de-interlace

OFF

------------------------- VIDEOPROC 1  DEV POOL -----------------------------

out     pool    ddr_id  count   max_count

0       0       0       3.0     3.0
```

4.5 OSG Proc Command

4.5.1 dump status

cat /proc/hdal/osg/info to show the status of OSG and focus on VIDEOPROC

```
------------------------ VIDEOENC 0 BUFFER -----------------------------------

pid     type    fmt     w       h       addr        size        draw

0       pp      4444    1000    200     13a55000    400000      1

0       pp              0       0       13ab6a80    400000      0


------------------------ VIDEOENC 0 STAMP -------------------------------------

pid     start   x       y       alpha   cken    ckval   layer   rgn

0       1       0       0       255     0       0       0       0

------------------------ VIDEOENC 0 MASK --------------------------------------

pid     start   x       y       w       h       solid   thick   color           alpha

0       1       500     0       100     120     1       0       ff0000          255

------------------------VIDEOPROC 0 MOSAIC ----------------------------------

pid     start   x       y       w       h       blkw    blkh

1       1       700     0       200     120     64      64
```

As above, the debug menu shows buffer, stamp, mask and mosaic configuration of all videoprocess's OSGs. Most values are simply from hd_videoprocess_set and self-explained. pid serves as an internal serial number and is mainly used to associate stamp and buffer information. start reflects if hd_videoprocess_start/hd_videoprocess_stop had been applied to that OSG.

4.5.2 change status

OSG attr can be changed through debug menu while buffer and image can't because buffer and image typically require a buffer which can't be created by shell console. To change an OSG's attr, echo *data* > /proc/hdal/osg/cmd. Below are the format of *data*:

1. For stamp: phase osg pid io start x y alpha cken ckval layer region
   example: to set the 5th stamp of device id 3 of videoprocess to position[1024,512] and layer(1) region(8), run "echo videoprocess stamp 5 3 1 1024 512 255 0 0 1 8"
2. For mask : phase osg pid io start x y w h solid thick color alpha

example: to set the 5th green mask of device id 3 of videoprocess to position[1024,512] and size 256x128, run “echo videoprocess mask 5 3 1 1024 512 256 128 1 0 0x0FF00 255”

3. For mosaic: phase osg pid io start x y w h mosaic_blk_w mosaic_blk_h
   example: to set the 5th mosaic of device id 3 of videoprocess to position[1024,512] and size 256x128, run “echo videoprocess mosaic 5 3 1 1024 512 256 128 32 32”

5 Sample Codes

5.1 user_videoprocess

The user_videoprcess demonstrates how to use the single trigger operation to process the input image.

```
/* Allocate in buffer */
scale_in_buffer.ddr_id = 0;
scale_in_buffer.dim.w = YUV_WIDTH;
scale_in_buffer.dim.h = YUV_HEIGHT;
scale_in_buffer.pxlfmt = HD_VIDEO_PXLFMT_YUV422_ONE;
raw_frame_size = YUV_WIDTH * YUV_HEIGHT * 2;

blk = hd_common_mem_get_block(pool, raw_frame_size, ddr_id);
if (HD_COMMON_MEM_VB_INVALID_BLK == blk) {
     printf("hd_common_mem_get_block fail\r\n");
     ret =  HD_ERR_NG;
     goto exit;
}
scale_in_buffer.phy_addr[0] = hd_common_mem_blk2pa(blk);
if (scale_in_buffer.phy_addr[0] == 0) {
     printf("hd_common_mem_blk2pa fail, blk = %#lx\r\n", blk);
     hd_common_mem_release_block(blk);
     return HD_ERR_NG;
}
scale_in_buffer_va = hd_common_mem_mmap(HD_COMMON_MEM_MEM_TYPE_NONCACHE,
                                        scale_in_buffer.phy_addr[0],
                                        raw_frame_size);

/* Allocate out buffer */
blk = hd_common_mem_get_block(pool, raw_frame_size, ddr_id);
if (HD_COMMON_MEM_VB_INVALID_BLK == blk) {
     printf("hd_common_mem_get_block fail\r\n");
     ret =  HD_ERR_NG;
     goto exit;
}
scale_out_buffer.phy_addr[0] = hd_common_mem_blk2pa(blk);
if (scale_out_buffer.phy_addr[0] == 0) {
     printf("hd_common_mem_blk2pa fail, blk = %#lx\r\n", blk);
     hd_common_mem_release_block(blk);
     return HD_ERR_NG;
}
scale_out_buffer_va = hd_common_mem_mmap(HD_COMMON_MEM_MEM_TYPE_NONCACHE,
                                        scale_out_buffer.phy_addr[0],
                                        raw_frame_size);

/* Set parameters */
crop.win.rect.x = YUV_WIDTH / 4;
crop.win.rect.y = YUV_HEIGHT / 4;
crop.win.rect.w = YUV_WIDTH / 2;
crop.win.rect.h = YUV_HEIGHT / 2;
crop.win.coord.w = YUV_WIDTH;
crop.win.coord.h = YUV_HEIGHT;
ret = hd_videoproc_set(path_id, HD_VIDEOPROC_PARAM_IN_CROP, (void *)&crop);
if (ret != HD_OK) {
     printf("hd_videoproc_set in crop fail\n");
     goto exit;
}
out.rect.x = 0;
out.rect.y = 0;
out.rect.w = MAX_FRAME_WIDTH;
out.rect.h = MAX_FRAME_HEIGHT;
out.bg.w = MAX_FRAME_WIDTH;
out.bg.h = MAX_FRAME_HEIGHT;
out.pxlfmt = HD_VIDEO_PXLFMT_YUV422_ONE;
out.dir = HD_VIDEO_DIR_NONE;
ret = hd_videoproc_set(path_id, HD_VIDEOPROC_PARAM_OUT, (void *)&out);
if (ret != HD_OK) {
     printf("hd_videoproc_set out fail\n");
     goto exit;
}
```

```
/* Push in buffer */
if ((raw_frame_fd = fopen(YUV_FILE, "rb")) == NULL) {
     printf("[ERROR] Open File %s failed!!\n", YUV_FILE);
     goto exit;
}
printf("Load pattern file: %s\n", YUV_FILE);
fread((void *)scale_in_buffer_va, 1, raw_frame_size, raw_frame_fd);
ret = hd_videoproc_push_in_buf(path_id, &scale_in_buffer, &scale_out_buffer, 500);
if (ret != HD_OK) {
     printf("hd_videoproc_push_in_buf fail\n");
     goto exit;
}

/* Pull out buffer */
ret = hd_videoproc_pull_out_buf(path_id, &scale_out_buffer, 500);
if (ret != HD_OK) {
     printf("hd_videoproc_pull_out_buf fail\n");
     goto exit;
} else {
     sprintf(filename, "user_vpe_%ldx%ld_YUV422.yuv",
             scale_out_buffer.dim.w, scale_out_buffer.dim.h);
     save_output(filename, scale_out_buffer_va, FRAME_BUF_SIZE);
}

/* Release in buffer */
hd_common_mem_munmap(scale_in_buffer_va, raw_frame_size);
hd_common_mem_release_block((HD_COMMON_MEM_VB_BLK)scale_in_buffer.phy_addr[0]);

/* Release out buffer */
hd_common_mem_munmap(scale_out_buffer_va, raw_frame_size);
hd_common_mem_release_block((HD_COMMON_MEM_VB_BLK)scale_out_buffer.phy_addr[0]);
```

6 Q&A

1. Can an osd image shared between videoprocess, videoenc and videoout?
   - Yes : Just set the same p_addr value of HD_OSG_STAMP_BUF to videoprocess, videoenc and videoout pathid.
   - Subsequent upate through any pathid with automatically reflect on other pathid
2. Any constrain on osd image and buffer?
   - Width of an image is best to be 4 aligned
   - Height of an image is best to be 2 aligned
   - Buffer address is best to be 128 aligned
3. Ex stamp and mask consume much system resource. Any advice on the number?
   - It's hard to say. System with heavy loading has less margin for ex stamp and mask
   - For a 30fps system, 4 ex stamps or 4 ex masks are safe
4. How to set alpha for an osd image
   - This is conditional to image formats
   - Alpha of argb4444 is completely determined by pixel's 4-bits alpha value
   - Alpha of argb1555 is determined by pixel's 1-bit alpha value and the alpha field of HD_OSG_STAMP_ATTR. If pixel's alpha value is 0, bits 3 ~ bits 0 of HD_OSG_STAMP_ATTR's alpha field determines transparency. If pixel's alpha value is 1, bits 7 ~ bits 4 of HD_OSG_STAMP_ATTR's alpha field determines transparency
   - Alpha of rgb565 is completely determined by the alpha field of HD_OSG_STAMP_ATTR.
5. If an osd is configured with ping pong buffer. Is it possible to directly draw the free buffer?
   - Yes
   - Use HD_VIDEOENC_PARAM_IN_STAMP_IMG to get the physical address of the free buffer
   - Draw this free buffer
   - Apply the drawing by set HD_VIDEOPROC_PARAM_IN_STAMP_IMG. The p_addr field of HD_OSG_STAMP_IMG should be the same physical address.