Snapshot

User Guide

Issue 02

Date 2019-04-30

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

Related Versions

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

Product Name	Version
Hi3559A	V100ES
Hi3559A	V100
Hi3559C	V100
Hi3519A	V100
Hi3556A	V100
Hi3516C	V500
Hi3516D	V300
Hi3516A	V300
Hi3559	V200
Hi3556	V200

NOTE

- Unless otherwise stated, Hi3559C V100 and Hi3559A V100 contents are consistent.
- Unless otherwise stated, Hi3556A V100 and Hi3519A V100 contents are consistent.
- Unless otherwise stated, Hi3516D V300, Hi3516A V300, Hi3559 V200, Hi3556 V200, and Hi3516C V500 contents are consistent.

Change History

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

Issue 03 (2019-07-25)

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

In section 1.5, the **Note** fields of HI_MPI_SNAP_SetPipeAttr and HI_MPI_SNAP_DisablePipe are modified.

In section 2.5, the **Member** field of PHOTO_HDR_ATTR_S is modified.

Issue 02 (2019-04-30)

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

Section 2.1 and section 3.2 are modified.

Issue 01 (2018-12-21)

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

In section 1.5, HI_MPI_SNAP_SetBNRRawDumpAttr to HI_MPI_SNAP_ReleaseBNRRaw are deleted.

In section 1.6, BNR_DUMP_ATTR_S is deleted.

The descriptions in the **Member** fields of SNAP_PRO_AUTO_PARAM_S and SNAP_PRO_MANUAL_PARAM_S are modified.

Issue 00B11 (2018-11-13)

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

In section 1.5, the description in the Note field of HI_MPI_SNAP_SetPipeAttr is modified.

In section 1.6, the description in the Note field of SNAP_PRO_AUTO_PARAM_S is modified.

Section 2.3 is modified.

Issue 00B10 (2018-09-29)

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

The contents related to the Hi3516D V300, Hi3516C V500, Hi3559 V200, and Hi3556 V200 are added.

Issue 00B09 (2018-08-08)

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

In section 1.3.4, the caution part is modified.

Issue 00B08 (2018-06-15)

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

In section 2.4, the description in the **Note** field of HI_MPI_PHOTO_AlgInit is modified.

Issue 00B07 (2018-05-15)

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

In section 1.5, the descriptions in the **Note** fields of HI_MPI_SNAP_GetBNRRaw and HI_MPI_SNAP_ReleaseBNRRaw are modified.

In section 1.6, the descriptions in the Member fields of ISP_PRO_SHARPEN_PARAM_S and ISP_PRO_BNR_PARAM_S are modified.

In section 2.5, the descriptions in the Syntax and Member fields of PHOTO_MFNR_COEF_S, PHOTO_MFNR_ATTR_S, and PHOTO_MFNR_3DNR_ISO_STRATEGY_S are modified.

PHOTO_MIN_WIDTH to PHOTO_MAX_HEIGHT are added.

Chapter 3 is added.

Issue 00B06 (2018-04-04)

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

In section 2.5, PHOTO_IMAGE_FUSION_PARAM_S is added, and PHOTO_MFNR_GHOST_DETECTION_S is deleted.

The descriptions in the Syntax and Member fields of PHOTO_HDR_COEF_S and PHOTO_MFNR_3DNR_ISO_STRATEGY_S are modified.

Issue 00B05 (2018-03-15)

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

Section 2.5 is modified.

Issue 00B04 (2018-02-10)

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

Section 1.2 is modified.

Chapter 2 is added.

Issue 00B03 (2018-01-15)

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

Section 1.3 is modified.

In section 1.5, HI_MPI_SNAP_SetProBNRParam and HI_MPI_SNAP_GetProBNRParam are modified.

In section 1.6, ISP_PRO_BNR_PARAM_S and ISP_PRO_SHARPEN_PARAM_S are modified.

Issue 00B02 (2017-11-15)

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

In section 1.5, the descriptions in the **Note** fields of HI_MPI_SNAP_SetPipeAttr and HI_MPI_SNAP_EnablePipe are modified.

Issue 00B01 (2017-08-28)

This issue is the first draft release.

Contents

Figures

1 Introduction

1.1 Overview

The consumer camera snapshot solution is designed for the photography function of the consumer electronics products. It offers the normal and PRO (professional) capture modes, supporting both single snapshot and multiple snapshots with different exposure times. The consumer camera snapshot solution supports the post-processing algorithms such as high dynamic range (HDR), single-frame noise reduction (SFNR), multi-frame noise reduction (MFNR), and DE.

The consumer camera snapshot datapaths come in two types, the single-pipe datapath and the dual-pipe datapath, with each supporting different scenarios. Each pipe can switch between online and offline processing.

1.2 Important Concepts

- Single-pipe mode

  Shooting and previewing are processed in the same image sensor processor (ISP) datapath.
- Dual-pipe mode

  Shooting and previewing are processed in different ISP datapaths.
- PRO mode

  In this mode, the ISP controls the sensor exposure. Multiple images with adjustable exposure time and gain are generated. The PRO mode can be used by the HDR algorithm to merge photos with different exposure times into one photo, and be used to take photos with fixed exposure time.
- Zero Shutter Lag (ZSL)

  ZSL reduces the delay caused by shutter lag or other factors and allows you to take a photo in the instant of triggering the shutter.

1.3 Snapshot Datapath

There are three pipe modes of the VI: offline mode, online mode and parallel mode. The snapshot datapaths are built on the VI. Therefore, the snapshot datapaths come in the three modes as well.

The resolutions for video previewing and snapshot are always different. The ISP performs a range of optimization such as correcting the facial skin colors in the shooting scenarios. Therefore, shooting and previewing are processed in different ISP datapaths. The shooting datapaths can be further divided into two types: the single-pipe datapath and the dual-pipe datapath.

In addition, consumer electronics products provide two more datapaths for shooting in the ZSL mode and stitch mode.

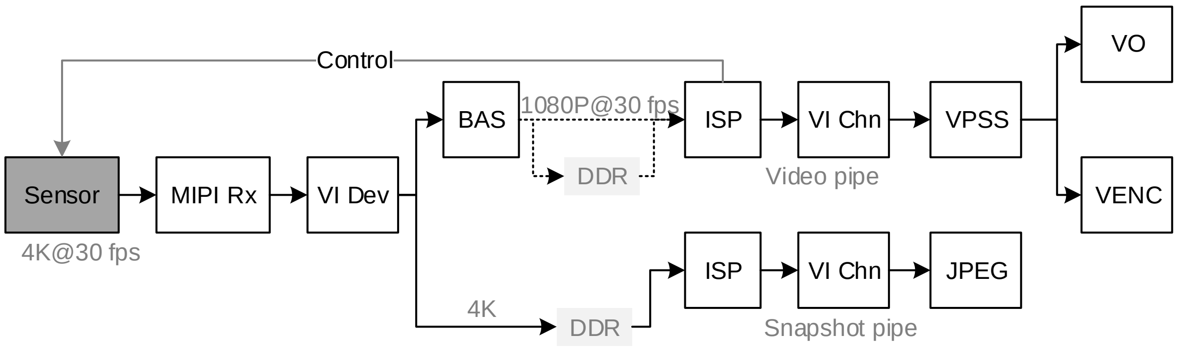
Note that the online and offline relationships between the VI and VPSS affect only the YUV output position, but not the control flow of snapshot. Therefore, the online and offline modes described in the following sections apply only to the VI pipe.

In summary, the shooting datapaths come in various types, with each designed for different scenarios. You are advised to use the offline mode with dual-pipe datapath. This solution offers quick snapshot with lowest power consumption. The advantages and disadvantages of each datapath type are described as follows.

1.3.1 Offline Mode, Dual Pipes

Figure 1-1 shows the dual-pipe shooting datapath in the offline mode.

Figure 1-1 Dual-pipe shooting datapath in the offline shooting mode



The resolutions shown in Figure 1-1 are for reference only. The actual resolutions may differ with the change of scenarios. The same applies to the other figures on datapath in this document.

In the offline mode with dual-pipe datapath, the data, coming from a sensor, will be bound to two different pipes respectively after VI Dev timing analysis. The top pipe is responsible for video previewing and filming and the bottom one is responsible for shooting. The pipe for video previewing and filming supports both online and offline processing, but the shooting pipe supports offline processing only.

Because the resolutions of video previewing and filming are generally low, the process of bayer scaling (BAS) is added to scale down the resolutions to be processed in the pipe, so that the power consumption can be reduced. Although the liquid crystal display (LCD) resolution of a DV is generally low, the previewing datapath cannot be omitted.

Although the resolution of shooting is generally high, users do not shoot all the times. Therefore, the bottom pipe for shooting will be enabled only when the shooting mode is on.

The sensor exposure is controlled by the ISP of the top pipe.

The pipe involved in the shooting attribute setting and shooting operations is the bottom pipe. The internal data is synchronized by the drivers of the VI module and the ISP.

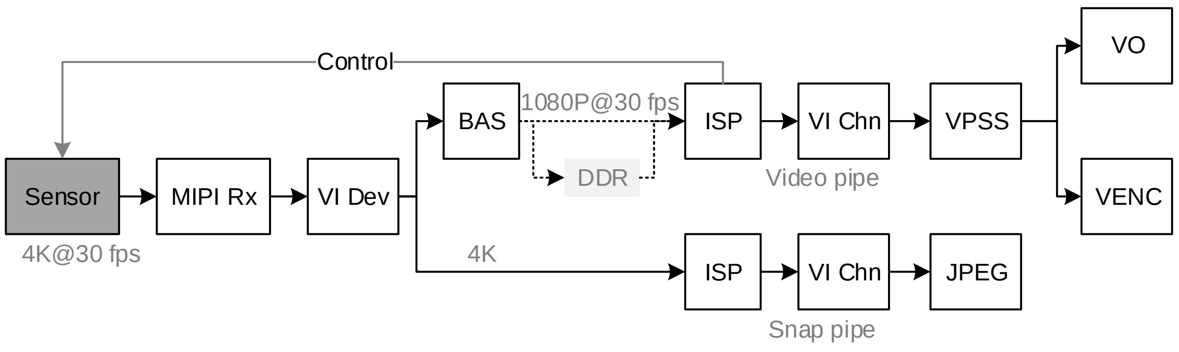
Such datapath can be used for shooting in the normal and PRO modes. In the PRO mode, HI_MPI_SNAP_TriggerPipe should be called to control the sensor to operate long and short exposures.

- For shooting with the dual-pipe datapath in the offline mode, the pipe for shooting enables the interrupt on and begins data processing only when you call HI_MPI_SNAP_TriggerPipe. Only the frames of snapshot will be processed. Upon the completion of a snapshot, the VI module will automatically disable the interrupt of the pipe for shooting and stop data processing. The purpose of such configuration is to reduce power consumption.
- For shooting with the dual-pipe datapath in the offline mode, photos with normal exposure will be generated. Theoretically, it takes about three frames to complete the whole process from calling HI_MPI_SNAP_TriggerPipe to the output of the first image with normal YUV data from the VI module. It is enough to meet customer requirements in most scenarios. Therefore, the dual-pipe datapath in the offline mode is recommended to perform shooting operations.

1.3.2 Online Mode, Dual Pipes

Figure 1-2 shows the dual-pipe shooting datapath in the online mode.

Figure 1-2 Dual-pipe shooting datapath in the online mode



The dual-pipe datapath in the online shooting mode is the same as the dual-pipe datapath in the offline shooting mode, except that the pipe for shooting operates in the online mode.

- In the online shooting mode with the dual-pipe datapath, data stream processing occurs when you call HI_MPI_SNAP_EnablePipe. However, the VI outputs the YUV data only when HI_MPI_SNAP_TriggerPipe is called. It is designed for scenarios where there are stricter requirements on the shooting response time. The pipe for shooting begins data processing when HI_MPI_SNAP_EnablePipe is called. Therefore, the power consumption is greater than shooting with the dual-pipe datapath in the offline mode
- For shooting with the dual-pipe datapath in the online mode, it takes less than the exposure time for one frame to produce a normally exposed photo, including the calling of HI_MPI_SNAP_TriggerPipe and the YUV data output from the VI module. However,

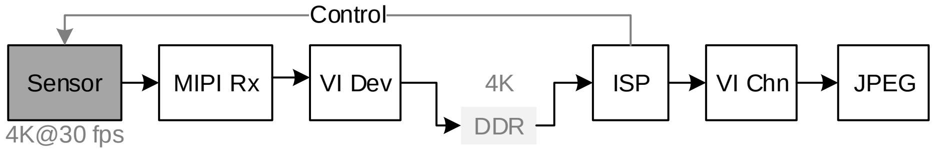
the precondition is that the calling interval between HI_MPI_SNAP_EnablePipe and HI_MPI_SNAP_TriggerPipe must be greater than two frames. The requirement of invocation interval also applies to the image effect processing by the ISP.

Such datapath can be used for shooting in the normal and PRO modes.

1.3.3 Offline Mode, Single Pipe

Figure 1-3 shows the single-pipe shooting datapath in the offline mode.

Figure 1-3 Single-pipe shooting datapath in the offline mode



- In the offline shooting mode with one pipe, video previewing and shooting share one pipe.
- The whole datapath uses low resolutions for video previewing. The datapath of the sensor and the VI uses higher resolutions only when the shooting mode is enabled. The purpose of such configuration is to reduce power consumption.
- The offline shooting mode with one pipe applies to the scenarios where the inputs come from more than one sensor and the inputs fully occupy the VI pipe. In that case, one sensor input cannot be bound to two pipes.

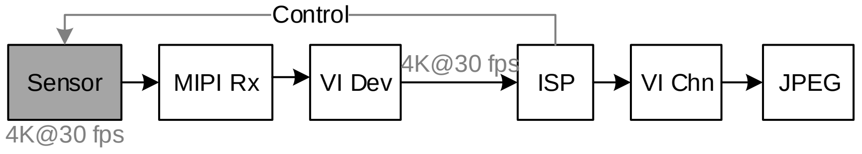
Such datapath can be used for shooting in the normal and PRO modes.

In the cases when the resolutions for recording and shooting are the same and the user ask no special adjustments of the shooting ISP, shooting operations can be realized by encoding the YUV data from the video stream in the single pipe into JPEG format. This solution does not require extra work from the drivers of the VI module and the ISP. Therefore, no adjustment of the shooting-related media processing platform programming interfaces (MPIs) is needed.

1.3.4 Online Mode, Single Pipe

Figure 1-4 shows the single-pipe shooting datapath in the online mode.

Figure 1-4 Single-pipe shooting datapath in the online mode



The single-pipe datapath in the online shooting mode is the same as the single-pipe datapath in the offline shooting mode, except that the pipe operates in the online mode. Such datapath can be used for shooting in the normal and PRO modes.

NOTICE

- The VI module of Hi3559A V100 and Hi3559C V100 supports at most two online pipes for the present. In the scenarios where there are inputs from multiple sensors, if the pipe for shooting operates in online mode, other sensors should operate in offline mode.
- The VI module of Hi3519A V100 and Hi3516C V500 supports only one online pipe. In the scenarios where there are inputs from more than one sensor, all sensors should operate in offline mode.

1.3.5 Parallel Mode

Because the maximum performance of VI_PROC is 4K@60 fps, the parallel mode should be enabled when the timing data input into VI_CAP exceeds 4K@60 fps. This mode also supports shooting by calling MPIs.

Such datapath can be used for shooting in the normal and PRO modes.

1.3.6 ZSL Mode

The shooting datapath of the ZSL mode is the same as that of the offline shooting mode with dual pipes, except that the VI driver caches a queue of raw data. When HI_MPI_SNAP_EnablePipe is called, the VI driver will begin to cache raw data. When HI_MPI_SNAP_TriggerPipe is called, the ZSL snapshot frames will be selected and sent to the ISP for processing.

The ZSL mode can only be used for shooting in the normal mode.

1.3.7 Stitch Mode

Figure 1-5 Shooting datapath in the stitch mode

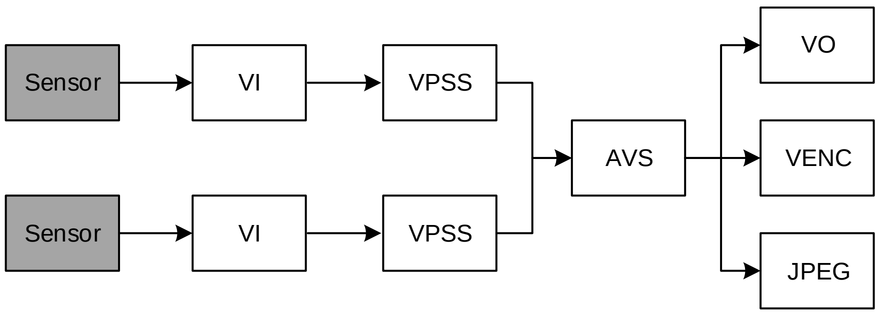


Figure 1-5 shows the shooting datapath in the stitch mode. It can stitch more than one sensor. The stitch function is realized by the any view stitching (AVS) module.

The stitch mode can only be used for single shooting in the normal mode.

When you call HI_MPI_SNAP_MultiTrigger for shooting, the VI driver will tag the frame processed by different sensors at the same time as the snapshot frame. You can encode the snapshot frame into JPEG format by calling the HI_MPI_VENC_SetJpegEncodeMode MPI of the VENC.

1.4 Function Description

1.4.1 Frame Rate Control During Burst Shooting

The frame rate of snapshot can be controlled by the frame rate controller of VI_PIPE_ATTR_S configured by HI_MPI_VI_CreatePipe or HI_MPI_VI_SetPipeAttr.

NOTE

For details about the preceding MPIs, see Chapter 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference*.

1.5 API Reference

This module provides the following MPIs:

- HI_MPI_SNAP_SetPipeAttr: Sets the shooting attributes.
- HI_MPI_SNAP_GetPipeAttr: Obtains the shooting attributes.
- HI_MPI_SNAP_EnablePipe: Enables the pipe for shooting.
- HI_MPI_SNAP_DisablePipe: Disables the pipe for shooting.
- HI_MPI_SNAP_TriggerPipe: Triggers the snapshot operation.
- HI_MPI_SNAP_MultiTrigger: Triggers the stitch shooting operation.
- HI_MPI_SNAP_SetProSharpenParam: Sets the sharpen parameter of the pipe for shooting.
- HI_MPI_SNAP_GetProSharpenParam: Obtains the sharpen parameter of the pipe for shooting.
- HI_MPI_SNAP_SetProBNRParam: Sets the BNR parameter of the pipe for shooting.
- HI_MPI_SNAP_GetProBNRParam: Obtains the BNR parameter of the pipe for shooting.

HI_MPI_SNAP_SetPipeAttr

[Description]

Sets the shooting attributes.

[Syntax]

```
HI_S32 HI_MPI_SNAP_SetPipeAttr(VI_PIPE ViPipe, const SNAP_ATTR_S
*pstSnapAttr);
```

[Parameter]

Parameter	Description	Input/Output
ViPipe	VI pipe ID Value range: [0, VI_MAX_PIPE_NUM)	Input
pstSnapAttr	Pointer to the attribute structure of the shooting parameters	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The value indicates the error code of the VI module.

[Chip Difference]

None

[Requirement]

- Header files: **hi_comm_snap.h** and **mpi_snap.h**
- Library file: **libmpi.a**

[Note]

- A pipe must have been created before this MPI is called.
- The shooting parameters must be valid. For details, see the description of SNAP_ATTR_S.
- Shooting in WDR mode is not supported.
- After HI_MPI_SNAP_EnablePipe is called, only the **stProAttr** attribute of the member **pstSnapAttr** can be modified when HI_MPI_SNAP_SetPipeAttr is called again. If other attributes are modified, an error code is returned.

[Example]

None

[See Also]

HI_MPI_SNAP_GetPipeAttr

HI_MPI_SNAP_GetPipeAttr

[Description]

Obtains the shooting attributes.

[Syntax]

```
HI_S32 HI_MPI_SNAP_GetPipeAttr(VI_PIPE ViPipe, SNAP_ATTR_S *pstSnapAttr);
```

[Parameter]

Parameter	Description	Input/Output
ViPipe	VI pipe ID Value range: [0, VI_MAX_PIPE_NUM)	Input
pstSnapAttr	Pointer to the attribute structure of the shooting parameters	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The value indicates the error code of the VI module.

[Chip Difference]

None

[Requirement]

- Header files: **hi_comm_snap.h** and **mpi_snap.h**
- Library file: **libmpi.a**

[Note]

A pipe must have been created before this MPI is called.

[Example]

None

[See Also]

HI_MPI_SNAP_SetPipeAttr

HI_MPI_SNAP_EnablePipe

[Description]

Enables the pipe for shooting.

[Syntax]

```
HI_S32 HI_MPI_SNAP_EnablePipe(VI_PIPE ViPipe);
```

[Parameter]

Parameter	Description	Input/Output
ViPipe	VI pipe ID Value range: [0, VI_MAX_PIPE_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The value indicates the error code of the VI module.

[Chip Difference]

None

[Requirement]

- Header files: **hi_comm_snap.h** and **mpi_snap.h**
- Library file: **libmpi.a**

[Note]

- A pipe must have been created before this MPI is called.
- The shooting attributes must have been set before this MPI is called.
- This MPI is not supported in single-pipe shooting scenarios. After HI_MPI_VI_StartPipe is called, shooting can be performed immediately after HI_MPI_SNAP_TriggerPipe is called.
- In dual-pipe shooting scenarios, only HI_MPI_SNAP_EnablePipe can be called to enable the shooting pipe. HI_MPI_VI_StartPipe cannot be called to enable the shooting pipe. For details of the functions of HI_MPI_VI_StartPipe, see section 3 "VI" in the *HiMPP V4.0 Media Processing Software Development Reference*.
- The pipe for shooting cannot be repeatedly enabled.
- In the dual-pipe shooting mode, the calling interval between the HI_MPI_SNAP_EnablePipe and HI_MPI_SNAP_TriggerPipe MPIs must be greater than the exposure time of two frames. Otherwise, the processing effect of the ISP cannot be guaranteed.

[Example]

None

[See Also]

HI_MPI_SNAP_SetPipeAttr

HI_MPI_SNAP_DisablePipe

[Description]

Disables the pipe for shooting or interrupts the shooting data stream.

[Syntax]

```
HI_S32 HI_MPI_SNAP_DisablePipe(VI_PIPE ViPipe);
```

[Parameter]

Parameter	Description	Input/Output
ViPipe	VI pipe ID Value range: [0, VI_MAX_PIPE_NUM)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The value indicates the error code of the VI module.

[Chip Difference]

None

[Requirement]

- Header files: **hi_comm_snap.h** and **mpi_snap.h**
- Library file: **libmpi.a**

[Note]

- A pipe must have been created before this MPI is called.
- This MPI is not supported in single-pipe shooting scenarios.
- This MPI cannot be called repeatedly for the same pipe. Otherwise, an error code is returned.

[Example]

None

[See Also]

HI_MPI_SNAP_EnablePipe

HI_MPI_SNAP_TriggerPipe

[Description]

Triggers the snapshot operation.

[Syntax]

```
HI_S32 HI_MPI_SNAP_TriggerPipe(VI_PIPE ViPipe);
```

[Parameter]

Parameter	Description	Input/Output
ViPipe	VI pipe ID Value range: [0, VI_MAX_PIPE_NUM)	Input

[Return Value]

Return Value	Description
0	Success

Return Value	Description
Other values	Failure. The value indicates the error code of the VI module.

[Chip Difference]

None

[Requirement]

- Header files: **hi_comm_snap.h** and **mpi_snap.h**
- Library file: **libmpi.a**

[Note]

- A pipe must have been created before this MPI is called.
- The pipe for shooting must have been enabled before this MPI is called.
- During the snapshot process, snapshot cannot be triggered again.

[Example]

None

[See Also]

HI_MPI_SNAP_EnablePipe

HI_MPI_SNAP_MultiTrigger

[Description]

Triggers multiple sensors to snapshot in the stitch scenarios.

[Syntax]

```
HI_S32 HI_MPI_SNAP_MultiTrigger(VI_STITCH_GRP StitchGrp);
```

[Parameter]

Parameter	Description	Input/Output
StitchGrp	Stitch group number For the definition of VI_STITCH_GRP, see section 2 "System Control" in the *HiMPP V4.0 Media Processing Software Development Reference*. Value range: [0, VI_MAX_STITCH_GRP_NUM)	Input

[Return Value]

Return Value	Description
0	Success

Return Value	Description
Other values	Failure. The value indicates the error code of the VI module.

[Chip Difference]

None

[Requirement]

- Header files: **hi_comm_snap.h** and **mpi_snap.h**
- Library file: **libmpi.a**

[Note]

- The attributes of the stitch group of the VI module must have been set before this MPI is called.
- **bStitch** in the attributes of the stitch group must have been enabled before this MPI is called.
- Each pipe in the stitch group must have been enabled before this MPI is called.
- This MPI cannot be called again during the snapshot process in the stitch mode.
- Shooting in the stitch mode tags one frame of the video stream as the snapshot frame. Therefore, the HI_MPI_SNAP_SetPipeAttr and HI_MPI_SNAP_EnablePipe MPIs do not need to be called for shooting.
- Hi3516C V500 does not support this MPI.

[Example]

None

[See Also]

None

HI_MPI_SNAP_SetProSharpenParam

[Description]

Sets the sharpen attributes in the PRO mode.

[Syntax]

```
HI_S32 HI_MPI_SNAP_SetProSharpenParam(VI_PIPE ViPipe, const
ISP_PRO_SHARPEN_PARAM_S *pstIspShpParam);
```

[Parameter]

Parameter	Description	Input/Output
ViPipe	VI pipe ID Value range: [0, VI_MAX_PIPE_NUM)	Input
pstIspShpParam	Pointer to the sharpen parameter structure of the ISP module	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The value indicates the error code of the VI or ISP module.

[Chip Difference]

None

[Requirement]

- Header files: **hi_comm_snap.h** and **mpi_snap.h**
- Library file: **libmpi.a**

[Note]

A pipe must have been created before this MPI is called.

[Example]

None

[See Also]

None

HI_MPI_SNAP_GetProSharpenParam

[Description]

Obtains the **sharpen** attributes.

[Syntax]

```
HI_S32 HI_MPI_SNAP_GetProSharpenParam(VI_PIPE ViPipe,
ISP_PRO_SHARPEN_PARAM_S *pstIspShpParam);
```

[Parameter]

Parameter	Description	Input/Output
ViPipe	VI pipe ID Value range: [0, VI_MAX_PIPE_NUM)	Input
pstIspShpParam	Pointer to the sharpen parameter structure of ISP module	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The value indicates the error code of the VI or ISP module.

[Chip Difference]

None

[Requirement]

- Header files: **hi_comm_snap.h** and **mpi_snap.h**
- Library file: **libmpi.a**

[Note]

A pipe must have been created before this MPI is called.

[Example]

None

[See Also]

None

HI_MPI_SNAP_SetProBNRParam

[Description]

Sets the BNR attributes in the PRO mode.

[Syntax]

```
HI_S32 HI_MPI_SNAP_SetProBNRParam(VI_PIPE ViPipe, const
ISP_PRO_BNR_PARAM_S *pstNrParma);
```

[Parameter]

Parameter	Description	Input/Output
ViPipe	VI pipe ID Value range: [0, VI_MAX_PIPE_NUM)	Input
pstNrParma	Pointer to the NR parameter structure	Input

[Return Value]

Return Value	Description
0	Success

Return Value	Description
Other values	Failure. The value indicates the error code of the VI or ISP module.

[Chip Difference]

None

[Requirement]

- Header files: **hi_comm_snap.h** and **mpi_snap.h**
- Library file: **libmpi.a**

[Note]

A pipe must have been created before this MPI is called.

[Example]

None

[See Also]

None

HI_MPI_SNAP_GetProBNRParam

[Description]

Obtains the BNR attributes.

[Syntax]

```
HI_S32 HI_MPI_SNAP_GetProBNRParam(VI_PIPE ViPipe, ISP_PRO_BNR_PARAM_S
*pstNrParma);
```

[Parameter]

Parameter	Description	Input/Output
ViPipe	VI pipe ID Value range: [0, VI_MAX_PIPE_NUM)	Input
pstNrParma	Pointer to the NR parameter structure	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The value indicates the error code of the VI or ISP module.

[Chip Difference]

None

[Requirement]

- Header files: **hi_comm_snap.h** and **mpi_snap.h**
- Library file: **libmpi.a**

[Note]

A pipe must have been created before this MPI is called.

[Example]

None

[See Also]

None

1.6 Data Structures

The shooting-related data structures are defined as follows:

- SNAP_ATTR_S: Defines the shooting parameters.
- SNAP_TYPE_E: Defines the enumeration of shooting types.
- SNAP_NORMAL_ATTR_S: Defines the shooting parameters for the normal type.
- SNAP_PRO_ATTR_S: Defines the shooting parameters for the PRO type.
- SNAP_PRO_PARAM_S: Defines the ISP-related parameters for the PRO type.
- SNAP_PRO_AUTO_PARAM_S: Defines the ISP auto mode-related parameters for the PRO type.
- SNAP_PRO_MANUAL_PARAM_S: Defines the ISP manual mode-related parameters for the PRO type.
- PRO_MAX_FRAME_NUM: Defines the macro of the maximum snapshot number for the PRO type.
- ISP_PRO_SHARPEN_PARAM_S: Defines the sharpen parameter.
- ISP_PRO_BNR_PARAM_S: Defines the BNR parameter.

SNAP_ATTR_S

[Description]

Defines the shooting parameters.

```
typedef struct hiSNAP_ATTR_S
{
    SNAP_TYPE_E enSnapType;
    HI_BOOL bLoadCCM;
    union
    {
        SNAP_NORMAL_ATTR_S     stNormalAttr;
        SNAP_PRO_ATTR_S        stProAttr;
    };
} SNAP_ATTR_S;
```

[Member]

Member	Description
enSnapType	Enumeration of shooting types
bLoadCCM	Whether to use external high dynamic range (CCM) values or not. ● **HI_TRUE**: Use the CCM information in the externally inputted ISP_CONFIG_INFO_S. ● **HI_FALSE**: Do not use the CCM information externally inputted, but the CCM information generated by the ISP algorithm.
stNormalAttr	Structure of the shooting parameters for the normal type
stProAttr	Structure of the shooting parameters for the PRO type.

[Note]

None

[See Also]

- HI_MPI_SNAP_SetPipeAttr
- HI_MPI_SNAP_GetPipeAttr

SNAP_TYPE_E

[Description]

Defines the enumeration of shooting types.

[Definition]

```
typedef enum hiSNAP_TYPE_E
{
    SNAP_TYPE_NORMAL,
    SNAP_TYPE_PRO,
    SNAP_TYPE_BUTT
} SNAP_TYPE_E;
```

[Member]

Member	Description
SNAP_TYPE_NORMAL	Normal type. During normal shooting, photos normal exposures are taken.

Member	Description
SNAP_TYPE_PRO	PRO type. During PRO shooting, photos with long and short exposures are taken.

[Note]

None

[See Also]

SNAP_ATTR_S

SNAP_NORMAL_ATTR_S

[Description]

Defines the shooting parameters for the normal type.

[Definition]

```
typedef struct hiSNAP_NORMAL_ATTR_S
{
    HI_U32  u32FrameCnt;
    HI_U32  u32RepeatSendTimes;
    HI_BOOL bZSL;
    HI_U32  u32FrameDepth;
    HI_U32  u32RollbackMs;
    HI_U32  u32Interval;
} SNAP_NORMAL_ATTR_S;
```

[Member]

Member	Description
u32FrameCnt	Number of snapshots Value range: (0, 0xFFFFFFF]
u32RepeatSendTimes	Number of times that first raw frame is sent. When the pipe of the VI module is offline, some ISP algorithms need to repeatedly transmit the first raw frame to generate reference information. Value range: [0, 2]
bZSL	Whether to use the ZSL mode for shooting or not
u32FrameDepth	Buffer queue depth in the ZSL mode Value range: [1, 8]

Member	Description
u32RollbackMs	Rollback time (unit: millisecond) when you call the trigger interface in the ZSL mode Because the ZSL caches have at most eight frames, when the rollback time is longer than the time of the earliest frame in the cache queue, the earliest frame in the cache queue will be selected. When the rollback time is between the two frames in the cache queue, the latter frame will be selected. Value range: [0, 0xFFFFFFFF] Note: Because of the limited length of the cache queue, it is unnecessary to roll back too much.
u32Interval	One more frame rate control can be implemented on the frames in the cache queue in the ZSL mode. This value indicates the interval (unit: number of frames) it takes to take one more frame as the snapshot frame after the first shooting frame in the cache queue is found. Value range: [0, 0xFFFFFFFF]

[Note]

None

[See Also]

SNAP_ATTR_S

SNAP_PRO_ATTR_S

[Description]

Defines the shooting parameters for the PRO type.

[Definition]

```
typedef struct hiSNAP_PRO_ATTR_S
{
    HI_U32  u32FrameCnt;
    HI_U32  u32RepeatSendTimes;
    SNAP_PRO_PARAM_S stProParam;
} SNAP_PRO_ATTR_S;
```

[Member]

Member	Description
u32FrameCnt	Number of snapshots Value range: (0, PRO_MAX_FRAME_NUM]

Member	Description
u32RepeatSendTimes	Number of times that first raw frame is sent. When the pipe of the VI module is offline, some ISP algorithms need to repeatedly transmit the first raw frame to generate reference information. Value range: [0, 2]
stProParam	Structure of the ISP parameters for the PRO type

[Note]

None

[See Also]

SNAP_ATTR_S

SNAP_PRO_PARAM_S

[Description]

Defines the ISP parameters for the PRO type.

[Definition]

```
typedef struct hiSNAP_PRO_PARAM_S
{
    OPERATION_MODE_E enOperationMode;
    SNAP_PRO_AUTO_PARAM_S stAutoParam;
    SNAP_PRO_MANUAL_PARAM_S stManualParam;
} SNAP_PRO_PARAM_S;
```

[Member]

Member	Description
enOperationMode	Mode of setting the enumeration of parameter types: auto mode or manual mode For the definition of OPERATION_MODE_E, see section 2 "System Control" in the *HiMPP V4.0 Media Processing Software Development Reference*.
stAutoParam	Parameter of the ISP auto mode for PRO shooting
stManualParam	Parameter of the ISP manual mode for PRO shooting

[Note]

None

[See Also]

SNAP_PRO_ATTR_S

SNAP_PRO_AUTO_PARAM_S

[Description]

Defines the ISP auto mode-related parameters for the PRO type.

[Definition]

```
typedef struct hiSNAP_PRO_AUTO_PARAM_S
{
   HI_U16 au16ProExpStep[PRO_MAX_FRAME_NUM];
} SNAP_PRO_AUTO_PARAM_S;
```

[Member]

Member	Description
au16ProExpStep	Exposure level of each frame in the ISP auto mode for PRO shooting Value range: [0, 0xFFFF]. For details about the calculation formula, see the following **Note** field. The upper limit of the exposure time range is related to the sensor. If the exposure time calculated based on the configured value exceeds the upper limit supported by the sensor, the upper limit supported by the sensor applies.

[Note]

- **au16HDRExpStep** uses the exposure of the current previewing channel as reference. The gain remains unchanged. The exposure time is adjusted based on the exposure level.

  ExpTime_i = ExpTime_base x au16ProExpStep[i]/256

  ExpTime_base indicates the reference exposure time. **ExpTime_i** indicates the exposure time of the *i*th frame of PRO shooting.

  - When **ExpTime_i** is greater than the configured maximum exposure time, the ISP will automatically enter the slow shutter mode to align the exposure time with **ExpTime_i**.
  - When **ExpTime_i** is less than the configured minimum exposure time, the actual exposure time is equal to the preset minimum exposure time.
  - The reference exposure of the current preview channel refers to the exposure of the AE module in automatic mode. The reference exposure cannot be set manually.

[See Also]

SNAP_PRO_PARAM_S

SNAP_PRO_MANUAL_PARAM_S

[Description]

Defines the ISP manual mode-related parameters for the PRO type.

[Definition]

```
typedef struct hiSNAP_PRO_MANUAL_PARAM_S
{
   HI_U32 au32ManExpTime[PRO_MAX_FRAME_NUM];
   HI_U32 au32ManSysgain[PRO_MAX_FRAME_NUM];
} SNAP_PRO_MANUAL_PARAM_S;
```

[Member]

Member	Description
au32ManExpTime	Exposure time for the ISP manual mode of PRO shooting. The unit is μs. Value range: [0, 0xFFFFFFFF]. The upper limit of the exposure time range is related to the sensor. If the configured value exceeds the upper limit supported by the sensor, the upper limit supported by the sensor applies.
au32ManSysgain	System gain for the ISP manual mode of the PRO type. The precision is 10 bits. Value range: [0x400, 0xFFFFFFFF]. The upper limit of the exposure time range is related to the sensor. If the configured value exceeds the upper limit supported by the sensor, the upper limit supported by the sensor applies.

[Note]

None

[See Also]

SNAP_PRO_PARAM_S

PRO_MAX_FRAME_NUM

[Description]

Defines the macro of the maximum snapshot number for the PRO type.

[Definition]

```
#define PRO_MAX_FRAME_NUM         (8)
```

[Note]

None

[See Also]

SNAP_PRO_ATTR_S

ISP_PRO_SHARPEN_PARAM_S

[Description]

Defines the sharpen parameter.

[Definition]

```
typedef struct hiISP_PRO_SHARPEN_PARAM_S
{
    HI_BOOL bEnable;
    HI_U32  u32ParamNum;
    ISP_SHARPEN_AUTO_ATTR_S *pastShpAttr;
} ISP_PRO_SHARPEN_PARAM_S;
```

[Member]

Member	Description
bEnable	Enable switch of the sharpen parameter in the PRO mode
u32ParamNum	Number of parameter groups Value range: [1, 8]
pastShpAttr	Sharpen parameter in the PRO mode. For details of the ISP_SHARPEN_AUTO_ATTR_S structure, see the descriptions on sharpen in section 6 "IMP" in the *HiISP Development Reference*.

[Note]

None

[See Also]

- HI_MPI_SNAP_SetProSharpenParam
- HI_MPI_SNAP_GetProSharpenParam

ISP_PRO_BNR_PARAM_S

[Description]

Defines the BNR parameter.

[Definition]

```
typedef struct hiISP_PRO_BNR_PARAM_S
{
    HI_BOOL bEnable;
    HI_U32  u32ParamNum;
    ISP_NR_AUTO_ATTR_S *pastNrAttr;
} ISP_PRO_BNR_PARAM_S;
```

[Member]

Member	Description
bEnable	Enable switch of the NR parameter in the PRO mode
u32ParamNum	Number of parameter groups Value range: [1, 8]
pastNrAttr	NR parameter in the PRO mode. For details of ISP_SHARPEN_AUTO_ATTR_S, see the descriptions on the noise reduction algorithm in section 6 "IMP" in the *HiISP Development Reference*.

[Note]

None

[See Also]

- HI_MPI_SNAP_SetProBNRParam
- HI_MPI_SNAP_GetProBNRParam

2 Post-Processing Algorithm after Photographing

2.1 Overview

NOTICE

Hi3516C V500 does not support post-processing algorithms after photographing.

Hi3519A V100 does not support post-processing algorithms after photographing.

Hi3556A V100 supports post-processing algorithms after photographing.

The Photo module integrates the post-processing algorithms after photographing of the consumer camera snapshot solution, including the HDR, MFNR, SFNR, and DE algorithms.

They are software processing algorithms running on the central processing unit (CPU) and digital signal processor (DSP).

2.2 Important Concepts

- High Dynamic Range (HDR)

  The HDR post-processing algorithm can increase the dynamic range of images. Shooting in PRO mode, you can obtain multiple images with adjustable exposure time and gains. Then those images are combined into one image with higher dynamic range by the HDR algorithm. Compared with common images, an HDR image offers a wider dynamic range and a larger amount of image detail.
- SFNR, short for single-frame noise reduction
- MFNR, short for multi-frame noise reduction
- DE, short for detail enhancement

  The DE algorithm can compensate for the detail loss caused by the background noise reduction (BNR) module of the ISP. The DE algorithm requires the input of a YUV image and a raw image. (The raw image is output by the BNR module.) Then, the DE algorithm outputs an enhanced YUV image.

2.3 Function Description

The Photo module operates depending on the DSP resources. The Photo libraries are compiled to the DSP 0 image by default. Therefore, before calling the Photo MPIs, ensure that HI_MPI_SVP_DSP_LoadBin has been called to load the DSP 0 image. For details about the HI_MPI_SVP_DSP_LoadBin MPI, see the *HiSVP API Reference*.

- Currently, only three-to-one HDR combination is supported. Three continuous YUV image frames with different exposure values are input to the Photo module and combined to one HDR YUV image frame as the output. The three input YUV image frames are short exposed, normally exposed, and long exposed in sequence.
- The HDR algorithm can optimize the effect of the face areas. The coordinates of the facial areas in an image need be detected in advance by using the intelligent algorithm for face recognition.
- Currently, only four-to-one MFNR is supported. Four continuous YUV image frames normally exposed are input to the Photo module and are combined to one YUV image frame which is temporal and spatial noise reduced as the output.
- The BNR raw data required by the DE algorithm is obtained through the **HI_MPI_VI_GetPipeBNRRaw** MPI.
- When the MFNR algorithm is complete, run the DE algorithm on the output YUV data. The BNR raw data required by the DE algorithm can be the BNR raw data corresponding to any one of the four YUV image frames input to the MFNR algorithm.
- The strides of the input and output frame data for algorithm processing must be 128-byte aligned, and the width and height of the frame data must be multiples of 8 pixels.
- The algorithms can only process the non-compressed NV21 YUV data (PIXEL_FORMAT_YVU_SEMIPLANAR_420 format) with the DYNAMIC_RANGE_SDR8 dynamic range.
- Only one algorithm can be called at a time. Multi-algorithm concurrent processing is not supported.
- Algorithms in the Photo module depend on the DSP. The current DSP uses the 32-bit address bus, capable of accessing only 4 GB memory address space. Therefore, the public memory and input and output frame buffers used by the algorithms in the Photo module must be located within the 4 GB memory address space. For details about restrictions on the address range, see the *HiSVP Development Guide*.

2.4 API Reference

This module provides the following media processing platform programming interface (MPIs):

- HI_MPI_PHOTO_AlgInit: Initializes a Photo algorithm.
- HI_MPI_PHOTO_AlgDeinit: Deinitializes a Photo algorithm.
- HI_MPI_PHOTO_AlgProcess: Runs an algorithm.
- HI_MPI_PHOTO_SetAlgCoef: Sets the adjustment coefficient for a Photo algorithm.
- HI_MPI_PHOTO_GetAlgCoef: Obtains the adjustment coefficient for a Photo algorithm.

HI_MPI_PHOTO_AlgInit

[Description]

Initializes a Photo algorithm.

[Syntax]

```
HI_S32 HI_MPI_PHOTO_AlgInit(PHOTO_ALG_TYPE_E enAlgType, const
PHOTO_ALG_INIT_S* pstPhotoInit);
```

[Parameter]

Parameter	Description	Input/Output
enAlgType	Enumeration value of the algorithms	Input
pstPhotoInit	Initialization parameter of a Photo algorithm	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The value indicates an error code. For details, see section 2.6 "Error Code."

[Chip Difference]

None

[Requirement]

- Header files: hi_comm_photo.h, mpi_photo.h
- Library file: libmpi_photo.a

[Note]

- Before calling this MPI, ensure that the binary files on the DSP end are successfully loaded.
- The memory transferred by this MPI must have been allocated from the media memory zone (MMZ).
- Different algorithms and different resolutions require different public memory sizes. You are advised to use the HDR_GetPublicMemSize, MFNR_GetPublicMemSize, SFNR_GetPublicMemSize, and DE_GetPublicMemSize functions defined in **hi_comm_photo.h** to obtain the public memory sizes of different algorithms corresponding to different resolutions.

[Example]

None

[See Also]

HI_MPI_PHOTO_AlgDeinit

HI_MPI_PHOTO_AlgDeinit

[Description]

Deinitializes a Photo algorithm.

[Syntax]

```
HI_S32 HI_MPI_PHOTO_AlgDeinit(PHOTO_ALG_TYPE_E enAlgType);
```

[Parameter]

Parameter	Description	Input/Output
enAlgType	Enumeration value of the algorithms	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The value indicates an error code. For details, see section 2.6 "Error Code."

[Chip Difference]

None

[Requirement]

- Header files: hi_comm_photo.h, mpi_photo.h
- Library file: libmpi_photo.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_PHOTO_AlgInit

HI_MPI_PHOTO_AlgProcess

[Description]

Runs an algorithm.

This is a block MPI. A value is returned only after the current frame has been processed.

Only one image frame is input each time this MPI is called. Therefore, for an algorithm that involves multi-frame combination, this MPI must be called multiple times. For example, for the three-to-one high dynamic range (HDR) combination, you must call this MPI three times. The current input frame starts to be processed once this MPI is called.

[Syntax]

```
HI_S32 HI_MPI_PHOTO_AlgProcess(PHOTO_ALG_TYPE_E enAlgType, const
PHOTO_ALG_ATTR_S* pstPhotoAttr);
```

[Parameter]

Parameter	Description	Input/Output
enAlgType	Enumeration value of the algorithms	Input
pstPhotoAttr	Attribute structure of algorithm processing	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The value indicates an error code. For details, see section 2.6 "Error Code."

[Chip Difference]

None

[Requirement]

- Header files: hi_comm_photo.h, mpi_photo.h
- Library file: libmpi_photo.a

[Note]

This MPI can be called only after the corresponding algorithm is successfully initialized.

[Example]

None

[See Also]

HI_MPI_PHOTO_AlgInit

HI_MPI_PHOTO_SetAlgCoef

[Description]

Sets the adjustment coefficient for a Photo algorithm.

This MPI is not mandatory. You may leave this MPI unconfigured, and the default coefficients are to be used.

[Syntax]

```
HI_S32 HI_MPI_PHOTO_SetAlgCoef(PHOTO_ALG_TYPE_E enAlgType, const
PHOTO_ALG_COEF_S* pstAlgCoef);
```

[Parameter]

Parameter	Description	Input/Output
enAlgType	Enumeration value of the algorithms	Input
pstAlgCoef	Structure of the algorithm coefficients	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The value indicates an error code. For details, see section 2.6 "Error Code."

[Chip Difference]

None

[Requirement]

- Header files: hi_comm_photo.h, mpi_photo.h
- Library file: libmpi_photo.a

[Note]

This MPI can be called after an algorithm is initialized and before the algorithm begins to process the first frame.

[Example]

None

[See Also]

HI_MPI_PHOTO_GetAlgCoef

HI_MPI_PHOTO_GetAlgCoef

[Description]

Obtains the adjustment coefficient for a Photo algorithm.

[Syntax]

```
HI_S32 HI_MPI_PHOTO_GetAlgCoef(PHOTO_ALG_TYPE_E enAlgType,
PHOTO_ALG_COEF_S* pstAlgCoef);
```

[Parameter]

Parameter	Description	Input/Output
enAlgType	Enumeration value of the algorithms	Input
pstAlgCoef	Structure of the algorithm coefficients	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The value indicates an error code. For details, see section 2.6 "Error Code."

[Chip Difference]

None

[Requirement]

- Header files: hi_comm_photo.h, mpi_photo.h
- Library file: libmpi_photo.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_PHOTO_SetAlgCoef

2.5 Data Structures

The Photo module provides the following data structures:

- PHOTO_ALG_TYPE_E: Defines the enumerations of the Photo algorithm types.
- PHOTO_ALG_INIT_S: Defines the initialization of a Photo algorithm.
- PHOTO_ALG_ATTR_S: Defines the attribute of Photo algorithm processing
- PHOTO_HDR_ATTR_S: Defines the HDR algorithm processing
- PHOTO_SFNR_ATTR_S: Defines the SFNR algorithm processing
- PHOTO_MFNR_ATTR_S: Defines the MFNR algorithm processing
- PHOTO_DE_ATTR_S: Defines the DE algorithm processing
- PHOTO_FACE_INFO_S: Defines the information of the face areas.
- PHOTO_ALG_COEF_S: Defines the Photo algorithm coefficients.
- PHOTO_HDR_COEF_S: Defines the HDR algorithm coefficients.
- PHOTO_IMAGE_FUSION_PARAM_S: Defines the image fusion parameters.
- PHOTO_DARK_MOTION_DETECTION_PARAM_S: Defines the HDR ghost detection parameters.
- PHOTO_SFNR_COEF_S: Defines the SFNR algorithm coefficients.
- PHOTO_SFNR_ISO_STRATEGY_S: Defines ISO strategies for the SFNR algorithm.

- PHOTO_MFNR_COEF_S: Defines the MFNR algorithm coefficients.
- PHOTO_MFNR_3DNR_PARAM_S: Defines the temporal NR coefficients for the MFNR algorithm.
- PHOTO_MFNR_3DNR_ISO_STRATEGY_S: Defines each ISO level of the temporal NR coefficients for the MFNR algorithm.
- PHOTO_MFNR_2DNR_PARAM_S: Defines the spatial NR coefficients for the MFNR algorithm.
- PHOTO_MFNR_2DNR_ISO_STRATEGY_S: Defines each ISO level for the spatial NR coefficients of the MFNR algorithm.
- PHOTO_DE_COEF_S: Defines the DE algorithm coefficients.
- PHOTO_DE_ISO_STRATEGY_S: Defines the strategy corresponding to each ISO level for the DE algorithm.
- PHOTO_HDR_FRAME_NUM: Defines the number of frames for HDR combination.
- PHOTO_MFNR_FRAME_NUM: Defines the number of frames for MFNR combination.
- PHOTO_HDR_ISO_LEVEL_CNT: Defines the number of ISO levels for adjusting the HDR image effect.
- PHOTO_SFNR_ISO_LEVEL_CNT: Defines the number of ISO levels for adjusting the SFNR image effect.
- PHOTO_MFNR_ISO_LEVEL_CNT: Defines the number of ISO levels for adjusting the MFNR image effect.
- PHOTO_DE_ISO_LEVEL_CNT: Defines the number of ISO levels for adjusting the DE image effect.
- PHOTO_MAX_FACE_NUM: Defines the maximum number of human face regions in algorithm processing.
- PHOTO_MIN_WIDTH: Defines the width of the minimum image resolution supported by algorithm processing.
- PHOTO_MIN_HEIGHT: Defines the height of the minimum image resolution supported by algorithm processing.
- PHOTO_MAX_WIDTH: Defines the width of the maximum image resolution supported by algorithm processing.
- PHOTO_MAX_HEIGHT: Defines the height of the maximum image resolution supported by algorithm processing.

PHOTO_ALG_TYPE_E

[Description]

Defines the enumerations of the Photo algorithm types.

[Definition]

```
typedef enum hiPHOTO_ALG_TYPE_E
{
    PHOTO_ALG_TYPE_HDR   = 0x0,
    PHOTO_ALG_TYPE_SFNR  = 0x1,
    PHOTO_ALG_TYPE_MFNR  = 0x2,
    PHOTO_ALG_TYPE_DE    = 0x3,
    PHOTO_ALG_TYPE_BUTT
} PHOTO_ALG_TYPE_E;
```

[Member]

Member	Description
PHOTO_ALG_TYPE_HDR	HDR algorithm
PHOTO_ALG_TYPE_SFNR	SFNR algorithm
PHOTO_ALG_TYPE_MFNR	MFNR algorithm
PHOTO_ALG_TYPE_DE	DE algorithm

[Note]

None

[See Also]

- HI_MPI_PHOTO_AlgInit
- HI_MPI_PHOTO_AlgDeinit
- HI_MPI_PHOTO_AlgProcess
- HI_MPI_PHOTO_SetAlgCoef
- HI_MPI_PHOTO_GetAlgCoef

PHOTO_ALG_INIT_S

[Description]

Defines the initialization of a Photo algorithm.

[Definition]

```
typedef struct hiPHOTO_ALG_INIT_S
{
   HI_U64     u64PublicMemPhyAddr;
   HI_U64     u64PublicMemVirAddr;
   HI_U32     u32PublicMemSize;
   HI_BOOL    bPrintDebugInfo;
}PHOTO_ALG_INIT_S;
```

[Member]

Member	Description
u64PublicMemPhyAddr	Start physical address of the memory for algorithm processing The memory needs to be allocated from the MMZ in advance.
u64PublicMemVirAddr	Virtual start address of the memory for algorithm processing This member is invalid because it is not used by the module yet.
u32PublicMemSize	Length of the memory zone for algorithm processing

Member	Description
bPrintDebugInfo	Whether to display the debugging information HI_TRUE: Displays the debugging information such as the algorithm version ID. HI_FALSE: Displays only the error information.

[Note]

None

[See Also]

HI_MPI_PHOTO_AlgInit

PHOTO_ALG_ATTR_S

[Description]

Defines the attribute structure of Photo algorithm processing

[Definition]

```
typedef struct hiPHOTO_ALG_ATTR_S
{
    union
    {
        PHOTO_HDR_ATTR_S     stHDRAttr;
        PHOTO_SFNR_ATTR_S    stSFNRAttr;
        PHOTO_MFNR_ATTR_S    stMFNRAttr;
        PHOTO_DE_ATTR_S      stDEAttr;
    };
}PHOTO_ALG_ATTR_S;
```

[Member]

Member	Description
stHDRAttr	HDR algorithm structure
stSFNRAttr	SFNR algorithm structure
stMFNRAttr	MFNR algorithm structure
stDEAttr	DE algorithm structure

[Note]

None

[See Also]

HI_MPI_PHOTO_AlgProcess

PHOTO_HDR_ATTR_S

[Description]

Defines the HDR algorithm processing

[Definition]

```
typedef struct hiPHOTO_HDR_ATTR_S
{
    VIDEO_FRAME_INFO_S  stSrcFrm;
    VIDEO_FRAME_INFO_S  stDesFrm;
    HI_U32              u32FrmIndex;
    HI_U32              u32ISO;
    HI_U32              u32FaceNum;
    PHOTO_FACE_INFO_S   astFaceInfo[PHOTO_MAX_FACE_NUM];
}PHOTO_HDR_ATTR_S;
```

[Member]

Member	Description
stSrcFrm	Information of the image frame input to the HDR algorithm For details about the structure definition, see chapter 2 "System Control" of the *HiMPP V4.0 Media Processing Software Development Reference*.
stDesFrm	Information of the image frame output from the HDR algorithm The frame buffer needs to be allocated from the VB pool in advance. The information needs to be sent when the MFNR processing of the first frame begins. The second and third frames use the same information as the first frame.
u32FrmIndex	Frame sequence number for HDR processing The sequence number starts at **0**. For three-to-one HDR processing, the sequence number of the three frames are **0**, **1**, and **2** in sequence.
u32ISO	ISO level of the current frame for HDR processing The ISO levels of the three frames must be the same.
u32FaceNum	Number of faces detected in the image The number of faces in the second frame (the normally exposed frame) is used. Value range: [0, 10]
astFaceInfo	Information structure of the face areas

[Note]

None

[See Also]

PHOTO_ALG_ATTR_S

PHOTO_SFNR_ATTR_S

[Description]

Defines the SFNR algorithm processing

[Definition]

```
typedef struct hiPHOTO_SFNR_ATTR_S
{
    VIDEO_FRAME_INFO_S  stFrm;
    HI_U32              u32ISO;
}PHOTO_SFNR_ATTR_S;
```

[Member]

Member	Description
stFrm	Information structure of the frame input to the SFNR algorithm After SFNR processing, the output image is written back to the frame buffer of the input image. For details about the structure definition, see chapter 2 "System Control" of the *HiMPP V4.0 Media Processing Software Development Reference*.
u32ISO	ISO level of the current frame for SFNR processing

[Note]

None

[See Also]

PHOTO_ALG_ATTR_S

PHOTO_MFNR_ATTR_S

[Description]

Defines the MFNR algorithm processing

[Definition]

```
typedef struct hiPHOTO_MFNR_ATTR_S
{
    VIDEO_FRAME_INFO_S  stSrcFrm;
    VIDEO_FRAME_INFO_S  stDesFrm;
    VIDEO_FRAME_INFO_S  stRawFrm;
    HI_U32            u32FrmIndex;
    HI_U32            u32ISO;
}PHOTO_MFNR_ATTR_S;
```

[Member]

Member	Description
stSrcFrm	Information of the image frame input to the MFNR algorithm For details about the structure definition, see chapter 2 "System Control" of the *HiMPP V4.0 Media Processing Software Development Reference*.
stDesFrm	• Information of the image frame output from the MFNR algorithm • The frame buffer needs to be allocated from the VB pool in advance. • The information needs to be sent when the MFNR processing of the first frame begins. The other three frames use the same information as the first frame.
stRawFrm	If the DE algorithm is not required after MFNR processing, this structure does not need to be assigned a value. If the DE algorithm is required after MFNR processing, this structure is the structure of the RAW data required by the DE algorithm. In the MFNR processing of four frames, only the RAW data corresponding to the frame whose **u32FrmIndex** is **0** is used. The RAW data structure of the other three frames must be consistent with the RAW data of the frame whose **u32FrmIndex** is **0**.
u32FrmIndex	Frame sequence number for MFNR processing The sequence number starts at **0**. For four-to-one MFNR processing, the sequence numbers of the four frames are 0, 1, 2, and 3 in sequence.
u32ISO	ISO level of the current frame for MFNR processing

[Note]

None

[See Also]

PHOTO_ALG_ATTR_S

PHOTO_DE_ATTR_S

[Description]

Defines the DE algorithm processing

[Definition]

```
typedef struct hiPHOTO_DE_ATTR_S
{
    VIDEO_FRAME_INFO_S  stFrm;
    VIDEO_FRAME_INFO_S  stRawFrm;
    HI_U32             u32ISO;
}PHOTO_DE_ATTR_S;
```

[Member]

Member	Description
stFrm	Information structure of the YUV frame input to the DE algorithm After DE processing, the output image is written back to the frame buffer of the input image. For details about the structure definition, see chapter 2 "System Control" of the *HiMPP V4.0 Media Processing Software Development Reference*.
stRawFrm	Information structure of the raw frame required by the DE algorithm.
u32ISO	ISO level of the current frame for DE processing

[Note]

None

[See Also]

PHOTO_ALG_ATTR_S

PHOTO_FACE_INFO_S

[Description]

Defines the information of the face areas.

[Definition]

```
typedef struct hiPHOTO_FACE_INFO_S
{
    HI_U32  u32Id;
    RECT_S  stFaceRect;
    RECT_S  stLeftEyeRect;
    RECT_S  stRightEyeRect;
    HI_U32  u32BlinkScore;
    HI_U32  u32SmileScore;
}PHOTO_FACE_INFO_S;
```

[Member]

Member	Description
u32Id	ID of the current face area
stFaceRect	Coordinates of the current face area For details about the structure definition, see chapter 2 "System Control" of the *HiMPP V4.0 Media Processing Software Development Reference*.
stLeftEyeRect	Coordinates of the left eye in the current face area This member is reserved and invalid temporarily.
stRightEyeRect	Coordinates of the right eye in the current face area This member is reserved and invalid temporarily.
u32BlinkScore	Value indicating that the eye is how much closed in the current face area This member is reserved and invalid temporarily.
u32SmileScore	Value indicating that the smiling is how much wide in the current face area This member is reserved and invalid temporarily.

[Note]

None

[See Also]

PHOTO_HDR_ATTR_S

PHOTO_ALG_COEF_S

[Description]

Defines the Photo algorithm coefficients.

[Definition]

```
typedef struct hiPHOTO_ALG_COEF_S
{
    union
    {
        PHOTO_HDR_COEF_S    stPhotoHdrCoef;
        PHOTO_SFNR_COEF_S   stPhotoSfnrCoef;
        PHOTO_MFNR_COEF_S   stPhotoMfnrCoef;
        PHOTO_DE_COEF_S     stPhotoDECoef;
    };
}PHOTO_ALG_COEF_S;
```

[Member]

Member	Description
stPhotoHdrCoef	Structure of the HDR algorithm coefficients
stPhotoSfnrCoef	Structure of the SFNR algorithm coefficients
stPhotoMfnrCoef	Structure of the MFNR algorithm coefficients
stPhotoDECoef	Structure of the DE algorithm coefficients

[Note]

None

[See Also]

- HI_MPI_PHOTO_SetAlgCoef
- HI_MPI_PHOTO_GetAlgCoef

PHOTO_HDR_COEF_S

[Description]

Defines the HDR algorithm coefficients.

[Definition]

```
typedef struct hiPHOTO_HDR_COEF_S
{
    HI_S32     s32AjustRatio;
    HI_S32     s32ImageScaleMethod;
    PHOTO_DARK_MOTION_DETECTION_PARAM_S
astMotionDetectionParam[PHOTO_HDR_ISO_LEVEL_CNT];
    PHOTO_IMAGE_FUSION_PARAM_S
astImageFusionParam[PHOTO_HDR_ISO_LEVEL_CNT];
}PHOTO_HDR_COEF_S;
```

[Member]

Member	Description
s32AjustRatio	Attenuation weight of the long exposed and short exposed frames of the face coefficient Value range: [0, 255]
s32ImageScaleMethod	Scaling interpolation mode of the output image 0: Bilinear interpolation 1: Lanczos interpolation

Member	Description
astMotionDetectionParam	Ghost detection parameter
astImageFusionParam	Image fusion parameter

[Note]

None

[See Also]

PHOTO_ALG_COEF_S

PHOTO_IMAGE_FUSION_PARAM_S

[Description]

Defines the image fusion parameters.

[Syntax]

```
typedef struct hiPHOTO_IMAGE_FUSION_PARAM_S
{
    HI_S32      s32IsoSpeed;
    HI_S32      s32PyramidTopSize;
    HI_S32      s32WeightCurveMethod;
    HI_S32      s32WeightCalcMethod;
    HI_S32      s32BlendUVGain;
    HI_S32      s32DetailEnhanceRatioL0;
    HI_S32      s32DetailEnhanceRatioL1;
    HI_FLOAT    f32BlendSigma;
}PHOTO_IMAGE_FUSION_PARAM_S;
```

[Member]

Member	Description
s32IsoSpeed	ISO value of each algorithm level
s32PyramidTopSize	Maximum width and height of the pyramid Value range: [16, 128]
s32WeightCurveMethod	Weight curve 0: The same Gaussian curve is shared. It is a normal distribution curve with brightness as the variable, and the standard deviation is **Blendsigma**. 1: The curves are set separately by segment. The weight curve of each frame is generated separately. **BlendSigma** controls the attenuation of each curve. A larger **BlendSigma** indicates faster attenuation.

Member	Description
s32WeightCalcMethod	Weight LUT 0: The corresponding curves are checked for frames according to the brightness. 1: When **WeightCurveMethod** is set to **1**, the three curves are checked only according to the medium exposure brightness.
s32BlendUVGain	UV enhancement coefficient after blending. Value range: [128, 255]
s32DetailEnhanceRatioL0	Lamination coefficient of layer 0 for pyramid reconstruction Value range: [0, 1024] Large edges are enhanced.
s32DetailEnhanceRatioL1	Lamination coefficient of layer 1 for pyramid reconstruction Value range: [0, 1024] Detail is enhanced.
f32BlendSigma	Standard deviation of a normal distribution curve Value range: [0.00, 4.00]

[Note]

None

[See Also]

PHOTO_HDR_COEF_S

PHOTO_DARK_MOTION_DETECTION_PARAM_S

[Description]

Defines the HDR ghost detection parameters.

[Definition]

```
typedef struct hiPHOTO_DARK_MOTION_DETECTION_PARAM_S
{
    HI_S32      s32IsoSpeed;
    HI_S32      s32MotionLowGray;
    HI_S32      s32MotionHighGray;
    HI_FLOAT     f32MotionRatio;
    HI_S32      s32NightAverageLuma;
}PHOTO_DARK_MOTION_DETECTION_PARAM_S;
```

[Member]

Member	Description
s32IsoSpeed	ISO level of each algorithm level
s32MotionLowGray	Motion detection threshold for the dark regions of the reference frame Value range: [0, 255]
s32MotionHighGray	Motion detection threshold for the bright regions of the reference frame Value range: [0, 255]
f32MotionRatio	Ratio threshold of the motion regions If the ratio is greater than the threshold, motion correction is not performed and the current processing frame is discarded. Value range: [0, 100]
s32NightAverageLuma	Night scene threshold for determining the luminance of the reference frame Value range: [0, 255]

[Note]

None

[See Also]

PHOTO_HDR_COEF_S

PHOTO_SFNR_COEF_S

[Description]

Defines the SFNR algorithm coefficients.

[Definition]

```
typedef struct hiPHOTO_SFNR_COEF_S
{
   PHOTO_SFNR_ISO_STRATEGY_S astIsoStrat[PHOTO_SFNR_ISO_LEVEL_CNT];
}PHOTO_SFNR_COEF_S;
```

[Member]

Member	Description
astIsoStrat	Structure of each ISO level of the SFNR algorithm

[Note]

None

[See Also]

PHOTO_ALG_COEF_S

PHOTO_SFNR_ISO_STRATEGY_S

[Description]

Defines ISO strategies for the SFNR algorithm.

[Definition]

```
typedef struct hiPHOTO_SFNR_ISO_STRATEGY_S
{
    HI_S32 s32IsoValue;
    HI_S32 s32Luma;
    HI_S32 s32Chroma;
    HI_S32 s32LumaHF;
    HI_S32 s32ChromaHF;
}PHOTO_SFNR_ISO_STRATEGY_S;
```

[Member]

Member	Description
s32IsoValue	ISO level of each algorithm level
s32Luma	Low frequency luma NR strength The larger the value is, the stronger the NR strength becomes. Value range: [–100, 0]
s32Chroma	Low frequency chroma NR strength The larger the value is, the stronger the NR strength becomes. Value range: [–100, 0]
s32LumaHF	Medium and high frequency luma NR strength The larger the value is, the stronger the NR strength becomes. Value range: [–100, 0]
s32ChromaHF	Medium and high frequency chroma NR strength The larger the value is, the stronger the NR strength becomes. Value range: [–100, 0]

[Note]

None

[See Also]

PHOTO_SFNR_COEF_S

PHOTO_MFNR_COEF_S

[Description]

Defines the MFNR algorithm coefficients.

[Definition]

```
typedef struct hiPHOTO_MFNR_COEF_S
{
    HI_BOOL bImageScale;
    PHOTO_MFNR_3DNR_PARAM_S st3DNRParam;
    PHOTO_MFNR_2DNR_PARAM_S st2DNRParam;
    HI_BOOL bDEEnable;
    PHOTO_DE_COEF_S stPhotoDECoef;
}PHOTO_MFNR_COEF_S;
```

[Member]

Member	Description
bImageScale	Whether the public region of the MFNR output result is cropped and scaled to the original image size HI_TRUE: Yes. The public region is scaled. HI_FALSE: No. The non-public regions are filled with the color of the adjacent areas.
st3DNRParam	Structure of the temporal NR coefficients for the MFNR algorithm
st2DNRParam	Structure of the spatial NR coefficients for the MFNR algorithm
bDEEnable	Whether to perform the DE algorithm after MFNR processing
stPhotoDECoef	Structure of the DE algorithm coefficients

[Note]

None

[See Also]

PHOTO_ALG_COEF_S

PHOTO_MFNR_3DNR_PARAM_S

[Description]

Defines the temporal NR coefficients for the MFNR algorithm.

[Definition]

```
typedef struct hiPHOTO_MFNR_3DNR_PARAM_S
{
   PHOTO_MFNR_3DNR_ISO_STRATEGY_S
ast3DNRIsoStrat[PHOTO_MFNR_ISO_LEVEL_CNT];
}PHOTO_MFNR_3DNR_PARAM_S;
```

[Member]

Member	Description
ast3DNRIsoStrat	Structure of each ISO level of the temporal NR coefficients for the MFNR algorithm.

[Note]

None

[See Also]

PHOTO_MFNR_COEF_S

PHOTO_MFNR_3DNR_ISO_STRATEGY_S

[Description]

Defines each ISO level of the temporal NR coefficients for the MFNR algorithm.

[Definition]

```
typedef struct hiPHOTO_MFNR_3DNR_ISO_STRATEGY_S
{
   HI_S32 s32IsoValue;
   HI_S32 s32Stnr;
   HI_S32 s32TnrFrmNum;
   HI_S32 s32StnrDarkLess;
   HI_S32 s32StnrGhostLess;
   HI_S32 s32LumaAlpha;
   HI_S32 s32LumaDelta;
   HI_S32 s32ChromaAlpha;
   HI_S32 s32ChromaDelta;
}PHOTO_MFNR_3DNR_ISO_STRATEGY_S;
```

[Member]

Member	Description
s32IsoValue	ISO level of each algorithm level
s32Stnr	Strength ratio of the temporal strength to the spatial strength The larger the ratio is, the stronger the temporal effect becomes. Value range: [0, 255]

Member	Description
s32TnrFrmNum	Number of frames for temporal noise reduction Value range: [1, 4]
s32StnrDarkLess	Strength of the spatial effect in dark regions If the pixel value is less than this member value, stronger spatial effect is used. Value range: [0, 255]
s32StnrGhostLess	Blur intensity of the ghost region. When the value is greater, the region is more blurred. Value range: [0, 255]
s32LumaAlpha	Luminance ratio threshold When the luminance ratio of two frames is greater than the threshold value, denoising or deghosting is performed. Value range: [0, 255]
s32LumaDelta	Luminance difference threshold When the luminance difference of two frames is greater than the threshold value, denoising or deghosting is performed. Value range: [0, 255]
s32ChromaAlpha	Chrominance ratio threshold When the chrominance ratio of two frames is greater than the threshold value, denoising or deghosting is performed. Value range: [0, 255]
s32ChromaDelta	Chrominance difference threshold When the chrominance difference of two frames is greater than the threshold value, denoising or deghosting is performed. Value range: [0, 255]

[Note]

None

[See Also]

PHOTO_MFNR_3DNR_PARAM_S

PHOTO_MFNR_2DNR_PARAM_S

[Description]

Defines the spatial NR coefficients for the MFNR algorithm.

[Definition]

```
typedef struct hiPHOTO_MFNR_2DNR_PARAM_S
{
    PHOTO_MFNR_2DNR_ISO_STRATEGY_S
ast2DNRIsoStrat[PHOTO_MFNR_ISO_LEVEL_CNT];
}PHOTO_MFNR_2DNR_PARAM_S;
```

[Member]

Member	Description
ast2DNRIsoStrat	Structure of each ISO level of the spatial NR coefficients for the MFNR algorithm.

[Note]

None

[See Also]

PHOTO_MFNR_COEF_S

PHOTO_MFNR_2DNR_ISO_STRATEGY_S

[Description]

Defines each ISO level for the spatial NR coefficients of the MFNR algorithm.

[Definition]

```
typedef struct hiPHOTO_MFNR_2DNR_ISO_STRATEGY_S
{
    HI_S32 s32IsoValue;
    HI_S32 s32Luma;
    HI_S32 s32Chroma;
    HI_S32 s32LumaHF2;
    HI_S32 s32DetailMin;
}PHOTO_MFNR_2DNR_ISO_STRATEGY_S;
```

[Member]

Member	Description
s32IsoValue	ISO level of each algorithm level
s32Luma	Low frequency luma NR strength The larger the value is, the stronger the NR strength becomes. Value range: [–100, 0]
s32Chroma	Low frequency chroma NR strength The larger the value is, the stronger the NR strength becomes. Value range: [–100, 0]

Member	Description
s32LumaHF2	Medium and high frequency luma NR strength The larger the value is, the stronger the NR strength becomes. Value range: [–100, 0]
s32DetailMin	High frequency luma detail retaining strength The larger the value is, the larger amount of image detail is retained. Value range: [0, 8]

[Note]

None

[See Also]

PHOTO_MFNR_3DNR_PARAM_S

PHOTO_DE_COEF_S

[Description]

Defines the DE algorithm coefficients.

[Definition]

```
typedef struct hiPHOTO_DE_COEF_S
{
   PHOTO_DE_ISO_STRATEGY_S astDEIsoStrat[PHOTO_DE_ISO_LEVEL_CNT];
}PHOTO_DE_COEF_S;
```

[Member]

Member	Description
astDEIsoStrat	Structure of the strategy corresponding to each ISO level for the DE algorithm.

[Note]

None

[See Also]

PHOTO_ALG_COEF_S

PHOTO_DE_ISO_STRATEGY_S

[Description]

Defines the strategy corresponding to each ISO level for the DE algorithm.

[Definition]

```
typedef struct hiPHOTO_DE_ISO_STRATEGY_S
{
    HI_S32  s32IsoValue;
    HI_S32  s32GlobalGain;
    HI_S32  s32GainLF;
    HI_S32  s32GainHFPos;
    HI_S32  s32GainHFNeg;
    HI_S32  s32LumaScaleX0;
    HI_S32  s32LumaScaleX1;
    HI_S32  s32LumaScaleY1;
    HI_S32  s32SatuGainX0;
    HI_S32  s32SatuGainX1;
    HI_S32  s32SatuGainY1;
}PHOTO_DE_ISO_STRATEGY_S;
```

[Member]

Member	Description
s32IsoValue	ISO level of each algorithm level
s32GlobalGain	Global detail strength coefficient The greater the value is, the stronger the strength becomes. Value range: [0, 255]
s32GainLF	Low frequency ratio The greater the value is, the stronger the strength becomes. Value range: [0, 255]
s32GainHFPos	High frequency white pixel ratio The greater the value, the stronger the strength. Value range: [0, 255]
s32GainHFNeg	High frequency black point pixel ratio The greater the value, the stronger the strength. Value range: [0, 255]
s32LumaScaleX0	Lower threshold of luminance stretching Value range: [0, 255]
s32LumaScaleX1	Upper threshold of luminance stretching Value range: [0, 255]
s32LumaScaleY1	Upper threshold of luminance stretching, corresponding to the attenuation coefficient Value range: [0, 255]

Member	Description
s32SatuGainX0	Lower threshold of saturation Value range: [0, 255]
s32SatuGainX1	Upper threshold of saturation Value range: [0, 255]
s32SatuGainY1	Upper threshold of saturation, corresponding to the attenuation coefficient Value range: [0, 255]

[Note]

None

[See Also]

PHOTO_DE_COEF_S

PHOTO_HDR_FRAME_NUM

[Description]

Defines the number of frames for HDR combination.

[Definition]

```
#define PHOTO_HDR_FRAME_NUM 3
```

[Note]

None

[See Also]

None

PHOTO_MFNR_FRAME_NUM

[Description]

Defines the number of frames for MFNR combination.

[Definition]

```
#define PHOTO_MFNR_FRAME_NUM  4
```

[Note]

None

[See Also]

None

PHOTO_HDR_ISO_LEVEL_CNT

[Description]

Defines the number of ISO levels for adjusting the HDR image effect.

[Definition]

```
#define PHOTO_HDR_ISO_LEVEL_CNT 10
```

[Note]

None

[See Also]

None

PHOTO_SFNR_ISO_LEVEL_CNT

[Description]

Defines the number of ISO levels for adjusting the SFNR image effect.

[Definition]

```
#define PHOTO_SFNR_ISO_LEVEL_CNT   8
```

[Note]

None

[See Also]

None

PHOTO_MFNR_ISO_LEVEL_CNT

[Description]

Defines the number of ISO levels for adjusting the MFNR image effect.

[Definition]

```
#define PHOTO_MFNR_ISO_LEVEL_CNT    8
```

[Note]

None

[See Also]

None

PHOTO_DE_ISO_LEVEL_CNT

[Description]

Defines the number of ISO levels for adjusting the DE image effect.

[Definition]

```
#define PHOTO_DE_ISO_LEVEL_CNT 8
```

[Note]

None

[See Also]

None

PHOTO_MAX_FACE_NUM

[Description]

Defines the maximum number of human face regions in algorithm processing.

[Definition]

```
#define PHOTO_MAX_FACE_NUM  10
```

[Note]

None

[See Also]

None

PHOTO_MIN_WIDTH

[Description]

Defines the width of the minimum image resolution supported by algorithm processing.

[Definition]

```
#define PHOTO_MIN_WIDTH     1280
```

[Note]

None

[See Also]

None

PHOTO_MIN_HEIGHT

[Description]

Defines the height of the minimum image resolution supported by algorithm processing.

[Definition]

```
#define PHOTO_MIN_HEIGHT    720
```

[Note]

None

[See Also]

None

PHOTO_MAX_WIDTH

[Description]

Defines the width of the maximum image resolution supported by algorithm processing.

[Definition]

```
#define PHOTO_MAX_WIDTH     7680
```

[Note]

None

[See Also]

None

PHOTO_MAX_HEIGHT

[Description]

Defines the height of the maximum image resolution supported by algorithm processing.

[Definition]

```
#define PHOTO_MAX_HEIGHT    4320
```

[Note]

None

[See Also]

None

2.6 Error Code

Table 2-1 lists the API error codes of the Photo module.

Table 2-1 API error codes of the Photo module

Error Code	Macro Definition	Description
0xA0268001	HI_ERR_PHOTO_INVALID_DEVID	The device ID of the Photo algorithm is invalid.
0xA0268002	HI_ERR_PHOTO_INVALID_CHNID	The channel ID of the Photo algorithm is invalid.
0xA0268003	HI_ERR_PHOTO_ILLEGAL_PARAM	The configuration of the input parameter is invalid.
0xA0268004	HI_ERR_PHOTO_EXIST	The Photo algorithm already exists.

Error Code	Macro Definition	Description
0xA0268005	HI_ERR_PHOTO_UNEXIST	The Photo algorithm does not exist.
0xA0268006	HI_ERR_PHOTO_NULL_PTR	The pointer to the input parameter is null.
0xA0268007	HI_ERR_PHOTO_NOT_CONFIG	The attribute of the Photo algorithm is not configured.
0xA0268008	HI_ERR_PHOTO_NOT_SUPPORT	The operation is not supported.
0xA0268009	HI_ERR_PHOTO_NOT_PERM	The operation is not allowed.
0xA026800C	HI_ERR_PHOTO_NOMEM	The memory fails to be allocated.
0xA026800D	HI_ERR_PHOTO_NOBUF	The buffer fails to be allocated.
0xA026800E	HI_ERR_PHOTO_BUF_EMPTY	The buffer is empty.
0xA0268010	HI_ERR_PHOTO_NOTREADY	The system is not initialized.
0xA0268012	HI_ERR_PHOTO_BUSY	The system is busy.

3 Surveillance Camera Snapshot Solution User Guide

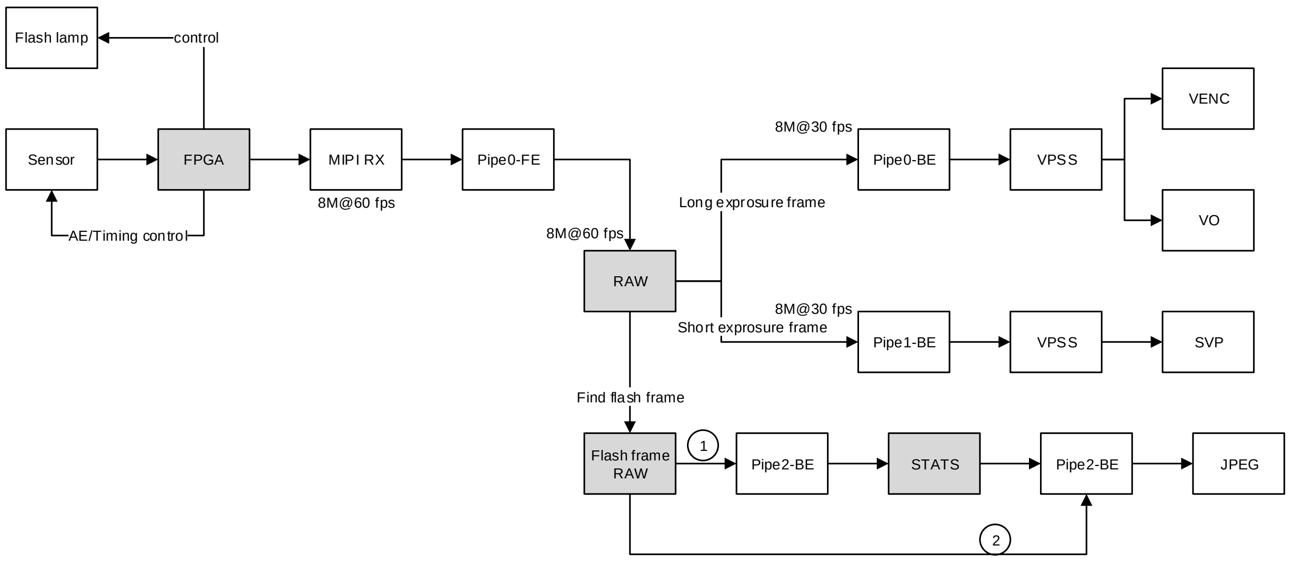
3.1 Overview

The surveillance camera snapshot solution is designed for surveillance products in scenes such as electronic police and checkpoint surveillance. The significant difference between surveillance and consumer camera snapshots is that the former type of cameras can be customized to a higher degree and frequently interact with peripherals.

3.2 Datapath

Figure 3-1 shows the typical datapath of the surveillance camera snapshot solution.

Figure 3-1 Typical datapath of the surveillance camera snapshot solution



A surveillance capture camera needs to control the flashing lights and stroboscopic lights of peripheral devices, and receive signals of traffic lights and snapshot triggering. The timing control and automatic exposure (AE) adjustment of the sensor must be synchronized with the traffic lights. Therefore, the user-defined ISP 3A algorithms are required.

In scenes with higher real-time requirements, the customer-implemented field-programmable gate array (FPGA) is used to control the sensor timing, flashing lights, and stroboscopic lights. The marking of snapshot frames is also implemented by the FPGA. Generally, a few lines of valid data are added after the valid data input by the sensor to identify whether the frames are snapshot frames and provide other user-defined information required by the customer.

The raw data collected from the front end (FE) of pipe 0 needs snapshot frame identification, and then the data will be sent to different pipes for subsequent processing. Before the data is sent to the BE for processing, the customer must tailor the additional lines of user-defined information in the raw data to avoid impact on the image effects processed by the BE. Currently, the VI pipe requires that the width and height data written by the FE must be consistent with those processed by the BE. Therefore, the raw data written by the FE of pipe 0 cannot be sent to the BE of the pipe 0 for processing after tailoring. In this case, a pipe can be created to replace the BE of pipe 0.

Snapshot frames are processed in a different way from video frames. For video frames, the configuration of the ISP image effect parameter can reference the statistics information of the previous frame to calculate the configuration information of the current frame. However, for snapshot frames, there is only one frame and no reference frame. Therefore, the raw data of the snapshot frame must be sent to the BE of pipe 2 twice. The first transmitted data is used to generate the statistics of the snapshot frame, based on which the ISP parameter configuration information is calculated. Then the correct parameter configuration information and the raw data of the snapshot frame are processed by the BE again so that YUV data of the snapshot frame can be correctly generated.

Generally, the ISP algorithm is driven by the start of frame (SOF) interrupt of each frame. In snapshot scenes, however, the raw data is sent to the BE for processing in offline mode, which is not applicable to the SOF interrupt. Therefore, a user program is required to drive the ISP. Before the raw data is sent to the BE for processing, HI_MPI_ISP_RunOnce needs to be called to generate the register configuration information required by the current frame to replace the HI_MPI_ISP_Run function.

HI_MPI_ISP_RunOnce requires the statistics about the previous frame processed by the ISP. Therefore, HI_MPI_ISP_RunOnce can be called only when the previous frame has been processed. Otherwise, the ISP statistics are not up-to-date, and the image effect may not meet the expectation. If the channel data can be successfully obtained by calling HI_MPI_VI_GetChnFrame (in VPSS offline mode) or HI_MPI_VPSS_GetChnFrame (in VPSS online mode), it indicates that the previous frame has been processed.

3.3 Process of Invoking a Program

For details about the program invoking process, see the code in **mpp/sample/traffic_capture**. For details about the interfaces used in programs, see the *HiISP Development Reference* and *HiMPP V4.0 Media Processing Software Development Reference*.