抓拍

使用指南

文档版本　01

发布日期　2018-12-21

商标声明

 、HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市海思半导体有限公司

地址：　深圳市龙岗区坂田华为基地华为电气生产中心　邮编：518129

网址：　http://www.hisilicon.com

客户服务电话：　+86-755-28788858

客户服务传真：　+86-755-28357515

客户服务邮箱：　support@hisilicon.com

前 言

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3559A	V100ES
Hi3559A	V100
Hi3559C	V100
Hi3519A	V100
Hi3556A	V100
Hi3516C	V500
Hi3516D	V300
Hi3559	V200
Hi3556	V200

说明

- 未有特殊说明，Hi3559CV100 与 Hi3559AV100 内容一致。
- 未有特殊说明，Hi3519AV100 与 Hi3556AV100 内容一致。
- 未有特殊说明，Hi3516DV300、Hi3559V200、Hi3556V200 与 Hi3516CV500 内容一致。

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

文档版本 01 (2018-12-21)

1.5 小节，删除 HI_MPI_SNAP_SetBNRRawDumpAttr~HI_MPI_SNAP_ReleaseBNRRaw

1.6 小节，删除 BNR_DUMP_ATTR_S

SNAP_PRO_AUTO_PARAM_S 和 SNAP_PRO_MANUAL_PARAM_S【成员】涉及修改

文档版本 00B11 (2018-11-13)

1.5 小节，HI_MPI_SNAP_SetPipeAttr【注意】涉及修改

1.6 小节，SNAP_PRO_AUTO_PARAM_S 的【注意】涉及修改

2.3 小节涉及修改

文档版本 00B10 (2018-09-29)

第 10 次临时版本发布。

添加 Hi3516CV500、Hi3516DV300、Hi3559V200 和 Hi3556V200 相关内容

文档版本 00B09 (2018-08-08)

第 9 次临时版本发布。

1.3.4 小节中的注意涉及修改

文档版本 00B08 (2018-06-15)

第 8 次临时版本发布。

2.4 小节，HI_MPI_PHOTO_AlgInit【注意】涉及修改

文档版本 00B07 (2018-05-15)

第 7 次临时版本发布。

1.5 小节，HI_MPI_SNAP_GetBNRRaw 和 HI_MPI_SNAP_ReleaseBNRRaw【注意】涉及修改

1.6 小节，ISP_PRO_SHARPEN_PARAM_S 和 ISP_PRO_BNR_PARAM_S【成员】涉及修改

2.5 小节，PHOTO_MFNR_COEF_S, PHOTO_MFNR_ATTR_S 和 PHOTO_MFNR_3DNR_ISO_STRATEGY_S 的【定义】和【成员】涉及修改

新增 PHOTO_MIN_WIDTH~PHOTO_MAX_HEIGHT

新增第 3 章

文档版本 00B06 (2018-04-04)

第 6 次临时版本发布。

2.5 小节，新增 PHOTO_IMAGE_FUSION_PARAM_S；删除 PHOTO_MFNR_GHOST_DETECTION_S

修改 PHOTO_HDR_COEF_S 和 PHOTO_MFNR_3DNR_ISO_STRATEGY_S【定义】和【成员】

文档版本 00B05 (2018-03-15)

第 5 次临时版本发布。

2.5 小节涉及修改

文档版本 00B04 (2018-02-10)

第 4 次临时版本发布。

1.2 小节涉及修改

新增第 2 章

文档版本 00B03 (2018-01-15)

第 3 次临时版本发布。

1.3 小节涉及修改

1.5 小节，HI_MPI_SNAP_SetProBNRParam 和 HI_MPI_SNAP_GetProBNRParam 涉及修改

1.6 小节，ISP_PRO_BNR_PARAM_S 和 ISP_PRO_SHARPEN_PARAM_S 涉及修改

文档版本 00B02 (2017-11-15)

第 2 次临时版本发布。

1.5 小节，HI_MPI_SNAP_SetPipeAttr、HI_MPI_SNAP_EnablePipe 的【注意】涉及修改

文档版本 00B01 (2017-08-28)

第 1 次临时版本发布。

目 录

1 消费类抓拍方案使用指南 1-1
1.1 概述 1-1
1.2 重要概念 1-1
1.3 抓拍数据通路 1-1
1.3.1 双 pipe 离线模式拍照 1-2
1.3.2 双 pipe 在线模式拍照 1-3
1.3.3 单 pipe 离线模式拍照 1-3
1.3.4 单 pipe 在线模式拍照 1-4
1.3.5 并行模式拍照 1-5
1.3.6 ZSL 模式拍照 1-5
1.3.7 拼接模式拍照 1-5
1.4 功能描述 1-6
1.4.1 连拍时的帧率控制 1-6
1.5 API 参考 1-6
1.6 数据类型 1-16
2 拍照后处理算法 2-1
2.1 概述 2-1
2.2 重要概念 2-1
2.3 功能描述 2-2
2.4 API 参考 2-2
2.5 数据类型 2-7
2.6 错误码 2-29
3 监控类抓拍方案使用指南 3-1
3.1 概述 3-1
3.2 数据通路 3-1
3.3 程序调用流程 3-2

插图目录

图 1-1 双 pipe 离线拍照数据通路.....1-2
图 1-2 双 pipe 在线拍照数据通路.....1-3
图 1-3 单 pipe 离线拍照数据通路.....1-4
图 1-4 单 pipe 在线拍照数据通路.....1-4
图 1-5 拼接模式拍照数据通路.....1-5
图 3-1 监控类抓拍机典型数据通路.....3-1

1 消费类抓拍方案使用指南

1.1 概述

消费类抓拍方案主要面向消费类电子产品中的拍照功能，支持 Normal、PRO 的抓拍模式，可以抓拍单张或者多张不同曝光时间的照片。消费类抓拍方案也支持 HDR、SFNR、MFNR 和 DE 的后处理算法。

抓拍的数据通路分为单 pipe 和双 pipe，每个 pipe 可以在线，也可以离线，每种数据通路适用的场景都有些差异。

1.2 重要概念

- 单 pipe 模式

  拍照和预览使用同一 ISP 通路。

- 双 pipe 模式

  拍照和预览使用不同的 ISP 通路。

- PRO(Professional)模式

  专业模式拍照，这种模式下 ISP 会控制 sensor 曝光，得到多张曝光时间和增益可调的图片。可以用于 HDR 算法做多张不同曝光的照片合成，也可以用于拍摄固定曝光时间的照片。

- ZSL(Zero Shutter Lag)

  零延时拍照。可以减少因为快门延迟或其他因素导致的延时，可以拍摄到触发拍照瞬间的图像。

1.3 抓拍数据通路

VI 的 pipe 工作模式分为离线模式、在线模式和并行模式，拍照的数据通路建立在 VI 之上，所以也分这三种模式。

在拍照的场景中，一般视频预览和抓拍的分辨率是不一样的；而且拍照的 ISP 效果处理要对人脸肤色等做优化处理，也会和视频预览通路的不一样。所以拍照的数据通路又分为单 pipe 和双 pipe 两种。

另外消费类电子产品中还有 ZSL 模式拍照和拼接拍照这两种特殊场景下的拍照通路。

注意，VI 和 VPSS 之间的在线、离线关系只影响拍照的 YUV 输出的位置，不影响拍照的控制流，所以下文说的在线和离线，都是指的拍照的那个 VI pipe 是在线或者离线。

综上，拍照的数据通路就会有很多种，每种数据通路的适用场景不一样，我们推荐客户采用双 pipe 离线模式的拍照方案，这种方案在功耗控制上是最优的，拍照的耗时也较短。

下面介绍每种数据通路的优缺点。

1.3.1 双 pipe 离线模式拍照

双 pipe 离线模式拍照的数据通路如图 1-1 所示。

图1-1 双 pipe 离线拍照数据通路

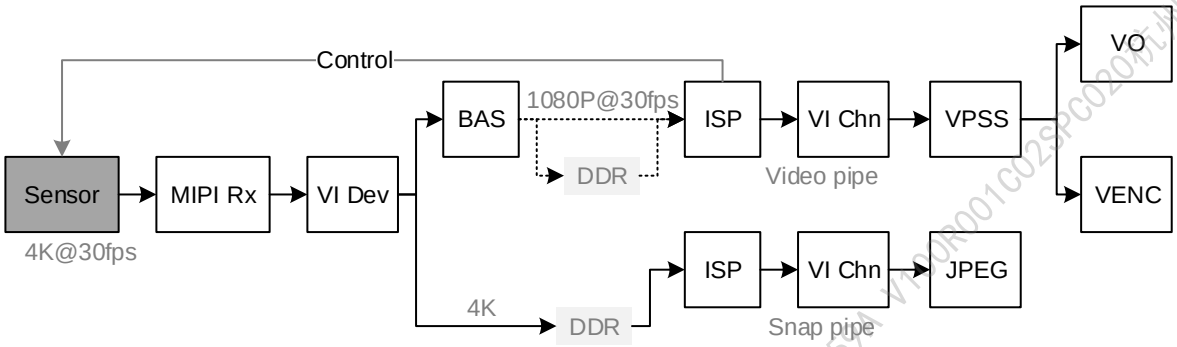


图 1-1 所标示的分辨率只是示意分辨率，实际分辨率根据客户不同的场景可能会有差异。后面的数据通路中所标示分辨率也是同样的意思。

双 pipe 离线的数据通路是一个 sensor 进来的数据，经过 VI Dev 时序解析后，分别绑定到 2 个不同的 pipe，上面的 pipe 用于视频预览和录像，下面的 pipe 用于拍照。视频预览和录像的 pipe 可以在线，也可以离线；拍照的 pipe 是离线的。

预览和录像的分辨率一般比较小，所以经过了 BAS，做了 Bayer Scale，主要目的是减小上面的 pipe 处理的分辨率，从而降低功耗。DV 产品中预览用的 LCD 屏分辨率一般都很小，但需要预览通路一直存在。

拍照的分辨率一般都比较大，但用户并不会一直拍照，所以下面的 pipe 用于拍照，一般是在客户需要拍照时才启动下面的 pipe 的通路。

Sensor 的曝光控制是由上面的视频 pipe 的 ISP 来控制的。

用户设置拍照相关的属性和触发拍照接口，用的 pipe 号都是下面那个拍照用的 pipe，内部的数据同步由 VI 和 ISP 的驱动来完成。

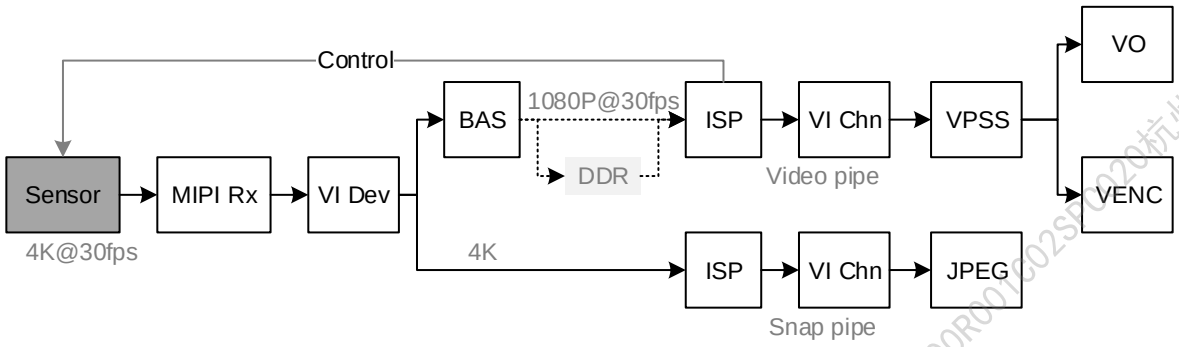
这种数据通路可以用于 NORMAL 和 PRO 模式的拍照。PRO 模式是在用户调用 HI_MPI_SNAP_TriggerPipe 接口之后才开始控制 sensor 进行长短曝光的。

- 双 pipe 离线时，拍照的 pipe 在用户调用 HI_MPI_SNAP_TriggerPipe 接口时才打开中断做数据处理，并且只处理抓拍的那几帧数据，抓拍完成后 VI 内部自动关闭拍照那个 pipe 的中断，停止数据处理。这样处理是为了降低拍照那个 pipe 的功耗。
- 双 pipe 离线时，用户拍摄正常曝光的照片，从调用 HI_MPI_SNAP_TriggerPipe 接口到 VI 输出第一张正常 YUV 数据，理论上的耗时在 3 帧左右，可以满足大部分客户场景的需求。所以，我们推荐客户采用双 pipe 离线的数据通路来实现拍照的方案。

1.3.2 双 pipe 在线模式拍照

双 pipe 在线模式拍照的数据通路如图 1-2 所示。

图1-2 双 pipe 在线拍照数据通路



双 pipe 在线拍照的数据通路和双 pipe 离线的类似，区别在于拍照的那个 pipe 是在线的。

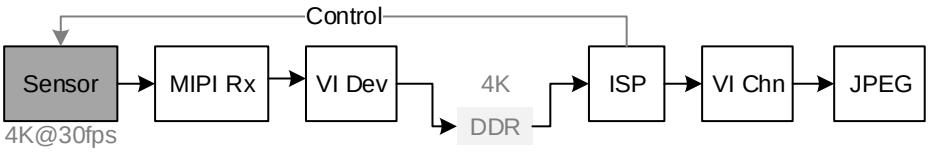
- 双 pipe 在线拍照时，用户调用 HI_MPI_SNAP_EnablePipe 之后就会有数据流处理，但 VI 并不输出 YUV，只有在用户调用 HI_MPI_SNAP_TriggerPipe 之后才输出 YUV。这样处理主要是为了满足对拍照的响应时间有更高要求的场景。但是因为拍照那个 pipe 在 HI_MPI_SNAP_EnablePipe 之后就开始处理数据，功耗上就会比双 pipe 离线拍照通路要大。
- 双 pipe 在线拍照，用户拍摄正常曝光的照片，从调用 HI_MPI_SNAP_TriggerPipe 接口到 VI 输出第一张 YUV，理论上的耗时不超过一帧的曝光时间。前提条件是 HI_MPI_SNAP_EnablePipe 和 HI_MPI_SNAP_TriggerPipe 这两个接口调用的间隔要大于两帧的曝光时间，ISP 的图像效果处理上也有这两帧间隔的要求。

这种数据通路可以拍摄 NORMAL 和 PRO 模式的照片。

1.3.3 单 pipe 离线模式拍照

单 pipe 离线模式拍照的数据通路如图 1-3 所示。

图1-3 单 pipe 离线拍照数据通路



- 单 pipe 离线数据通路是视频预览和拍照共用一个 pipe。
- 单 pipe 离线模式拍照使用的方法是平时只有视频预览时整个通路可以采用比较小的分辨率，只有切换到拍照模式时才将 sensor 和 VI 等通路切换到大的拍照分辨率。这样可以节省一定的功耗。
- 单 pipe 离线模式拍照应用的场景是当客户的产品有多个 sensor 同时输入将 VI pipe 全部占用，无法做到一个 sensor 输入绑定两个 pipe 时，就需要用这种模式来拍照。

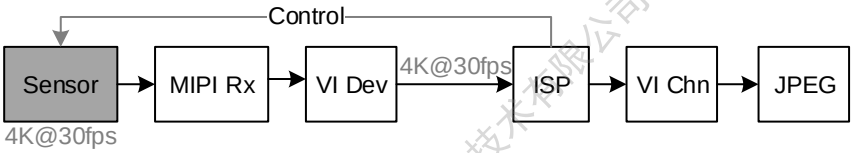
这种数据通路可以拍摄 NORMAL 和 PRO 模式的照片。

当录像和拍照的分辨率一样，客户又不需要对拍照的 ISP 做特殊的调节时，可以从单 pipe 的视频流里面取 YUV 编成 jpeg 来实现拍照的方案。这种方案并不需要 VI 和 ISP 的驱动做额外的事情，所以不需要调拍照相关的 MPI 接口。

1.3.4 单 pipe 在线模式拍照

单 pipe 在线模式拍照的数据通路如图 1-4 所示。

图1-4 单 pipe 在线拍照数据通路



单 pipe 在线拍照的数据通路和单 pipe 离线的类似，区别是数据通路是在线的。这种数据通路可以拍摄 NORMAL 和 PRO 模式的照片。

注意

- Hi3559AV100 及 Hi3559CV100 的 VI 模块目前只支持最多 2 个 pipe 同时在线，如果多 sensor 输入的场景，拍照使用在线时，其他的 sensor 要离线处理。
- Hi3519AV100 及 Hi3556AV100 的 VI 模块目前只支持 1 个 pipe 在线，如果多于 1 个 sensor 输入的场景，所有 sensor 要离线处理。

1.3.5 并行模式拍照

由于 VI_PROC 性能最大是 4K@60fps，在 VI_CAP 输入大数据量的时序超过 4K@60fps 时，需要采用并行模式。在这种模式下，也支持调用 MPI 接口来拍照。

这种数据通路可以拍摄 NORMAL 和 PRO 模式的照片。

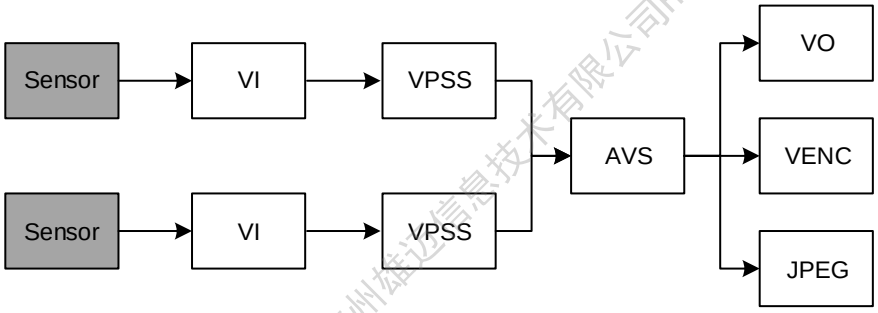
1.3.6 ZSL 模式拍照

ZSL 模式的拍照通路和双 pipe 离线模式的拍照通路一样，区别是 VI 驱动内部会缓存一个 RAW 数据的队列。在调用 HI_MPI_SNAP_EnablePipe 接口后，VI 内部就开始缓存 RAW 数据，调用 HI_MPI_SNAP_TriggerPipe 接口会选择 ZSL 的拍照帧，然后将拍照的帧送给 ISP 做处理。

ZSL 模式拍照，只支持拍摄 NORMAL 模式的照片。

1.3.7 拼接模式拍照

图1-5 拼接模式拍照数据通路



拼接模式拍照数据通路如图 1-5 所示，拼接的 sensor 个数可以是多个，拼接功能是由 AVS 模块来完成的。

拼接模式拍照只支持拍摄单张 NORMAL 模式的照片。

用户调用 HI_MPI_SNAP_MultiTrigger 拍照，VI 驱动内部会将同一时刻不同 sensor 的那帧数据标记为抓拍帧，用户可以通过 VENC 的 HI_MPI_VENC_SetJpegEncodeMode 接口编码有抓拍标记的那帧数据为 jpeg。

1.4 功能描述

1.4.1 连拍时的帧率控制

连拍时可以做帧率控制，是通过 HI_MPI_VI_CreatePipe 或者 HI_MPI_VI_SetPipeAttr 接口设置的 VI_PIPE_ATTR_S 中的帧率控制来实现的。

说明

以上接口请参考《HiMPP V4.0 媒体处理软件开发参考》视频输入章节。

1.5 API 参考

该模块提供以下 MPI：

- HI_MPI_SNAP_SetPipeAttr：设置拍照属性。
- HI_MPI_SNAP_GetPipeAttr：获取拍照属性。
- HI_MPI_SNAP_EnablePipe：使能拍照的 pipe。
- HI_MPI_SNAP_DisablePipe：停止拍照的 pipe。
- HI_MPI_SNAP_TriggerPipe：触发抓拍。
- HI_MPI_SNAP_MultiTrigger：拼接拍照的触发。
- HI_MPI_SNAP_SetProSharpenParam：设置拍照 pipe 的 sharpen 参数。
- HI_MPI_SNAP_GetProSharpenParam：获取拍照 pipe 的 sharpen 参数。
- HI_MPI_SNAP_SetProBNRParam：设置拍照 pipe 的 BNR 参数。
- HI_MPI_SNAP_GetProBNRParam：获取拍照 pipe 的 BNR 参数。

HI_MPI_SNAP_SetPipeAttr

【描述】

设置拍照的属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
ViPipe	VI 的 pipe 号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
pstSnapAttr	拍照参数的属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 VI 模块的错误码。

【芯片差异】

无。

【需求】

- 头文件：hi_comm_snap.h、mpi_snap.h
- 库文件：libmpi.a

【注意】

- PIPE 必须已创建。
- 拍照参数必须合法，具体请参见 SNAP_ATTR_S。
- WDR 模式不支持拍照。

【举例】

无

【相关主题】

HI_MPI_SNAP_GetPipeAttr

HI_MPI_SNAP_GetPipeAttr

【描述】

获取拍照的属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
ViPipe	VI 的 pipe 号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
pstSnapAttr	拍照参数的属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 VI 模块的错误码。

【芯片差异】

无。

【需求】

- 头文件：hi_comm_snap.h、mpi_snap.h
- 库文件：libmpi.a

【注意】

PIPE 必须已创建。

【举例】

无

【相关主题】

HI_MPI_SNAP_SetPipeAttr

HI_MPI_SNAP_EnablePipe

【描述】

使能拍照的 pipe。

【语法】

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

【参数】

参数名称	描述	输入/输出
ViPipe	VI 的 pipe 号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 VI 模块的错误码。

【芯片差异】

无。

【需求】

- 头文件：hi_comm_snap.h、mpi_snap.h
- 库文件：libmpi.a

【注意】

- PIPE 必须已创建。
- 拍照的属性必须已经设置。
- 单 PIPE 拍照场景，不支持此接口，调用 HI_MPI_VI_StartPipe 之后直接 trigger 即可拍照；
- 双 pipe 拍照场景，只能调用 HI_MPI_SNAP_EnablePipe 使能拍照 PIPE，不能调用 HI_MPI_VI_StartPipe 使能拍照 PIPE，HI_MPI_VI_StartPipe 的详细功能描述，请参考《HiMPP V4.0 媒体处理软件开发参考》中 VI 章节的说明。
- 不支持重复使能拍照的 PIPE。
- 双 PIPE 拍照时，HI_MPI_SNAP_EnablePipe 和 HI_MPI_SNAP_TriggerPipe 两个接口的调用间隔要大于 2 帧的曝光时间，才能使拍照的 ISP 处理效果正常。

【举例】

无

【相关主题】

HI_MPI_SNAP_SetPipeAttr

HI_MPI_SNAP_DisablePipe

【描述】

停止拍照的 pipe，也可以用于中断正在拍照的数据流。

【语法】

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

【参数】

参数名称	描述	输入/输出
ViPipe	VI 的 pipe 号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 VI 模块的错误码。

【芯片差异】

无。

【需求】

- 头文件：hi_comm_snap.h、mpi_snap.h
- 库文件：libmpi.a

【注意】

- PIPE 必须已创建。
- 单 pipe 拍照场景，不支持此接口。

【举例】

无

【相关主题】

HI_MPI_SNAP_EnablePipe

HI_MPI_SNAP_TriggerPipe

【描述】

触发拍照。

【语法】

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

【参数】

参数名称	描述	输入/输出
ViPipe	VI 的 pipe 号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 VI 模块的错误码。

【芯片差异】

无。

【需求】

- 头文件：hi_comm_snap.h、mpi_snap.h
- 库文件：libmpi.a

【注意】

- PIPE 必须已创建。
- 拍照 PIPE 必须已使能。
- 正在拍照过程中，不能再次触发拍照。

【举例】

无

【相关主题】

HI_MPI_SNAP_EnablePipe

HI_MPI_SNAP_MultiTrigger

【描述】

拼接场景下触发多路 sensor 同时拍照。

【语法】

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

【参数】

参数名称	描述	输入/输出
StitchGrp	拼接组号。 VI_STITCH_GRP 的定义请参考《HiMPP V4.0 媒体处理软件开发参考》中系统控制章节的介绍。 取值范围：[0, VI_MAX_STITCH_GRP_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 VI 模块的错误码。

【芯片差异】

无。

【需求】

- 头文件：hi_comm_snap.h、mpi_snap.h
- 库文件：libmpi.a

【注意】

- VI 的拼接组属性必须已设置。
- 拼接组属性中的 bStitch 必须已使能。
- 拼接组中每个 PIPE 必须已使能。
- 正在抓拍拼接照片的过程中，不能再次调用该接口。
- 拼接拍照是标记视频流里面的一帧为抓拍帧，不需要调用 HI_MPI_SNAP_SetPipeAttr、HI_MPI_SNAP_EnablePipe 等拍照接口。
- Hi3516CV500 不支持此接口。

【举例】

无

【相关主题】

无

HI_MPI_SNAP_SetProSharpenParam

【描述】

设置 PRO 模式下拍照的 sharpen 属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
ViPipe	VI 的 pipe 号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
pstIspShpParam	ISP 的 sharpen 参数结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 VI/ISP 模块的错误码。

【芯片差异】

无。

【需求】

- 头文件：hi_comm_snap.h、mpi_snap.h
- 库文件：libmpi.a

【注意】

PIPE 必须已创建。

【举例】

无

【相关主题】

无

HI_MPI_SNAP_GetProSharpenParam

【描述】

获取拍照的 sharpen 属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
ViPipe	VI 的 pipe 号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
pstIspShpParam	ISP 的 sharpen 参数结构体指针。	输出

【返回值】

返回值	描述
0	成功。

返回值	描述
非 0	失败，其值为 VI/ISP 模块的错误码。

【芯片差异】

无。

【需求】

- 头文件：hi_comm_snap.h、mpi_snap.h
- 库文件：libmpi.a

【注意】

PIPE 必须已创建。

【举例】

无

【相关主题】

无

HI_MPI_SNAP_SetProBNRParam

【描述】

设置 PRO 模式下拍照的 BNR 属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
ViPipe	VI 的 pipe 号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
pstNrParma	NR 参数结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 VI/ISP 模块的错误码。

【芯片差异】

无。

【需求】

- 头文件：hi_comm_snap.h、mpi_snap.h
- 库文件：libmpi.a

【注意】

PIPE 必须已创建。

【举例】

无

【相关主题】

无

HI_MPI_SNAP_GetProBNRParam

【描述】

获取拍照的 BNR 属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
ViPipe	VI 的 pipe 号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
pstNrParma	NR 参数结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 VI/ISP 模块的错误码。

【芯片差异】

无。

【需求】

- 头文件：hi_comm_snap.h、mpi_snap.h
- 库文件：libmpi.a

【注意】

PIPE 必须已创建。

【举例】

无

【相关主题】

无

1.6 数据类型

拍照相关数据类型定义如下：

- SNAP_ATTR_S：定义拍照参数的结构体。
- SNAP_TYPE_E：定义拍照类型的枚举。
- SNAP_NORMAL_ATTR_S：定义 Normal 类型拍照参数的结构体。
- SNAP_PRO_ATTR_S：定义 PRO 类型拍照参数的结构体。
- SNAP_PRO_PARAM_S：定义 PRO 类型 ISP 相关参数结构体。
- SNAP_PRO_AUTO_PARAM_S：定义 PRO 类型 ISP 自动模式参数结构体。
- SNAP_PRO_MANUAL_PARAM_S：定义 PRO 类型 ISP 手动模式参数结构体。
- PRO_MAX_FRAME_NUM：定义 PRO 类型拍照的最大张数的宏。
- ISP_PRO_SHARPEN_PARAM_S：定义 sharpen 参数的结构体。
- ISP_PRO_BNR_PARAM_S：定义 BNR 参数的结构体。

SNAP_ATTR_S

【说明】

定义拍照参数的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
enSnapType	拍照类型的枚举值。
bLoadCCM	是否使用外部的 CCM 值。 • HI_TRUE：使用外部输入的 ISP_CONFIG_INFO_S 中的 CCM 信息。 • HI_FALSE：不使用外部输入的 CCM 信息，用 ISP 算法自己计算生成的 CCM。
stNormalAttr	Normal 类型拍照的参数结构体。
stProAttr	PRO 模式拍照的参数结构体。

【注意事项】

无

【相关数据类型及接口】

- HI_MPI_SNAP_SetPipeAttr
- HI_MPI_SNAP_GetPipeAttr

SNAP_TYPE_E

【说明】

定义拍照类型的枚举。

【定义】

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

【成员】

成员名称	描述
SNAP_TYPE_NORMAL	Normal 类型，可以拍正常曝光的照片。
SNAP_TYPE_PRO	专业类型，可以拍不同长短曝光的照片。

【注意事项】

无

【相关数据类型及接口】

SNAP_ATTR_S

SNAP_NORMAL_ATTR_S

【说明】

定义 Normal 类型拍照参数的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
u32FrameCnt	拍照的张数。 取值范围：(0, 0xFFFFFFFF]。
u32RepeatSendTimes	重复送首帧 RAW 的次数。当 VI 的 pipe 离线时，ISP 里面的某些算法需要将拍照的首帧 RAW 重复送多次，用于生成参考信息。 取值范围：[0, 2]。
bZSL	是否使用 ZSL 模式拍照。
u32FrameDepth	ZSL 模式缓存队列的深度。 取值范围：[1, 8]。

成员名称	描述
u32RollbackMs	ZSL 模式下，用户调用 Trigger 接口时往前回退多少毫秒。 由于 ZSL 的缓存最多有 8 帧，当回退后的时间超过缓存队列里面最早那一帧的时间时，选择缓存最早的那一帧； 当回退之后的时间位于缓存队列里面两帧的中间时，选择较新的那一帧。 取值范围：[0, 0xFFFFFFFF]。 注意：由于缓存队列长度有限，回退的时间太大是没有意义的。
u32Interval	ZSL 模式下，缓存队列里面的帧可以再做一次帧率控制。该值代表在缓存队列里面找到首帧拍照帧后，间隔多少帧之后再取一帧作为拍照帧。 取值范围：[0, 0xFFFFFFFF]。

【注意事项】

无

【相关数据类型及接口】

SNAP_ATTR_S

SNAP_PRO_ATTR_S

【说明】

定义 PRO 类型拍照参数的结构体。

【定义】

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

【成员】

成员名称	描述
u32FrameCnt	拍照的张数。 取值范围：(0, PRO_MAX_FRAME_NUM]。

成员名称	描述
u32RepeatSendTimes	重复送首帧 RAW 的张数。当 VI 的 pipe 离线时，ISP 里面的某些算法需要将拍照的首帧 RAW 重复送多次，用于生成参考信息。 取值范围：[0, 2]。
stProParam	PRO 模式 ISP 参数结构体。

【注意事项】

无

【相关数据类型及接口】

SNAP_ATTR_S

SNAP_PRO_PARAM_S

【说明】

定义 PRO 类型中 ISP 参数的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
enOperationMode	设置参数类型的枚举，自动模式或者手动模式。 OPERATION_MODE_E 定义请参考《HiMPP V4.0 媒体处理软件开发参考》中系统控制章节。
stAutoParam	PRO 拍照时 ISP 自动模式的参数。
stManualParam	PRO 拍照时 ISP 手动模式的参数。

【注意事项】

无

【相关数据类型及接口】

SNAP_PRO_ATTR_S

SNAP_PRO_AUTO_PARAM_S

【说明】

定义 PRO 类型 ISP 自动模式参数结构体。

【定义】

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

【成员】

成员名称	描述
au16ProExpStep	PRO 拍照时 ISP 自动模式每帧的曝光等级。 取值范围：[0, 0xFFFF]，与曝光时间的计算公式见注意事项。 曝光时间范围上限与 sensor 相关，如果按照设置值计算后的曝光时间超过 sensor 的范围上限，则按照 sensor 的上限生效。

【注意事项】

- au16ProExpStep 是以当前预览通道的曝光量为基准，增益保持不变，根据曝光等级调整曝光时间。

  ExpTime_i = ExpTime_base*au16ProExpStep[i]/256

  ExpTime_base 代表基准曝光时间，ExpTime_i 代表专业拍照第 i 帧的曝光时间。

  - 当 ExpTime_i 大于当前设置的最大曝光时间时，ISP 会自动进入慢快门模式，使曝光时间等于 ExpTime_i；
  - 当 ExpTime_i 小于设置的最小曝光时间时，实际曝光时间等于设置的最小曝光时间。
  - 当前预览通路的基准曝光量是指 AE 在自动模式下的曝光量，不支持手动 AE 设置基准曝光量。

【相关数据类型及接口】

SNAP_PRO_PARAM_S

SNAP_PRO_MANUAL_PARAM_S

【说明】

定义 PRO 类型 ISP 手动模式参数结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
au32ManExpTime	PRO 拍照时 ISP 手动模式的曝光时间，单位为微秒。 取值范围：[0, 0xFFFFFFFF]，范围上限与 sensor 相关，如果设置值超过 sensor 的范围上限，则按照 sensor 的上限生效。
au32ManSysgain	PRO 拍照时 ISP 手动模式的系统增益，10bit 精度。 取值范围：[0x400, 0xFFFFFFFF]，范围上限与 sensor 相关，如果设置值超过 sensor 的范围上限，则按照 sensor 的上限生效。

【注意事项】

无

【相关数据类型及接口】

SNAP_PRO_PARAM_S

PRO_MAX_FRAME_NUM

【说明】

定义 PRO 类型拍照的最大张数的宏。

【定义】

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

【注意事项】

无

【相关数据类型及接口】

SNAP_PRO_ATTR_S

ISP_PRO_SHARPEN_PARAM_S

【说明】

定义 sharpen 参数的结构体。

【定义】

```
typedef struct hiISP_PRO_SHARPEN_PARAM_S
```

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

【成员】

成员名称	描述
bEnable	使用专业拍照模式 Sharpen 参数使能。
u32ParamNum	参数组数。 取值范围：[1, 8]
pastShpAttr	专业模式的 Sharpen 参数。 ISP_SHARPEN_AUTO_ATTR_S 结构体请参考《HiISP 开发参考》IMP 章节中的 Sharpen 描述。

【注意事项】

无

【相关数据类型及接口】

- HI_MPI_SNAP_SetProSharpenParam
- HI_MPI_SNAP_GetProSharpenParam

ISP_PRO_BNR_PARAM_S

【说明】

定义 BNR 参数的结构体。

【定义】

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

【成员】

成员名称	描述
bEnable	使用专业拍照模式 NR 参数使能。

成员名称	描述
u32ParamNum	参数组数。 取值范围：[1, 8]
pastNrAttr	专业模式的 NR 参数。ISP_NR_AUTO_ATTR_S 结构体请参考《HiISP 开发参考》IMP 章节中的去噪算法描述。

【注意事项】

无

【相关数据类型及接口】

- HI_MPI_SNAP_SetProBNRParam
- HI_MPI_SNAP_GetProBNRParam

2 拍照后处理算法

2.1 概述

注意

Hi3516CV500 不支持拍照后处理算法。

PHOTO 代表消费类抓拍方案中的拍照后处理算法，包括 HDR、MFNR、SFNR、DE 这几种。

拍照后处理算法是运行在 CPU 和 DSP 上的软件处理算法。

2.2 重要概念

- HDR(High Dynamic Range)

  HDR 图像后处理算法，能提升图像的动态范围；通过拍照的 PRO 模式拍照，得到多张曝光时间和增益可调的图片，然后经过 HDR 算法模块合成为一张具有高动态范围的图像，相比普通的图像，HDR 可以提供更多的动态范围和图像细节。

- SFNR(Single frame noise reduction)

  单帧降噪。

- MFNR(Multi-frame noise reduction)

  多帧降噪。

- DE(Detail enhancement)

  细节增强。ISP 中的 BNR 处理，会造成图像中一些细节信息的丢失，DE 算法可以补偿这些丢失的图像细节信息。DE 算法的输入是一张 YUV 和 BNR 写出的 RAW，输出是一张细节增强过的 YUV。

2.3 功能描述

PHOTO 模块的运行依赖 DSP 的资源，PHOTO 的库默认编译到 DSP0 的镜像中，所以在调用 PHOTO 的接口之前，请确保已经调用了 HI_MPI_SVP_DSP_LoadBin 接口加载了 DSP0 的镜像。（HI_MPI_SVP_DSP_LoadBin 的具体描述，请参考《HiSVP API 参考》中的介绍）

- HDR 合成当前仅支持三合一，就是输入三帧不同曝光的连续 YUV，输出一帧高动态范围的 YUV。三帧的输入顺序依次是短曝光的 YUV，正常曝光的 YUV，长曝光的 YUV。
- HDR 算法支持对人脸区域的图像效果做特殊优化处理。图像中的人脸区域坐标需要提前通过人脸检测的智能算法检测出来。
- MFNR 当前仅支持四合一，输入是四帧连续的正常曝光的 YUV，输出是一帧经过时域和空域降噪的 YUV。
- DE 算法需要的 BNR RAW 数据是通过 HI_MPI_VI_GetPipeBNRRaw 接口获取的。
- MFNR 算法做完后，可以再对输出后的 YUV 做一次 DE 算法处理。DE 算法需要的 BNR RAW 数据可以是 MFNR 输入的四帧 YUV 里面任意一帧 YUV 对应的 BNR RAW。
- 算法处理输入和输出帧数据的 Stride 必须是 128Byte 对齐，并且帧数据的像素宽高必须是 8 的倍数。
- 算法处理的输入 YUV 数据，仅支持处理 NV21 格式（也就是 PIXEL_FORMAT_YVU_SEMIPLANAR_420 格式）的非压缩数据，输入图像的动态范围仅支持 DYNAMIC_RANGE_SDR8 的。
- 同一时刻只能有一个算法处理被调用，不支持多个算法并行处理。
- PHOTO 中算法运行依赖 DSP，由于当前 DSP 使用 32bit 地址总线，只能访问 4GB 的内存地址空间。所以，PHOTO 算法使用的 Public 内存，输入、输出帧存都必须位于 4GB 的内存地址空间中。详细的地址范围限制说明请参考《HiSVP 开发指南》中的描述。

2.4 API 参考

该模块提供以下 MPI：

- HI_MPI_PHOTO_AlgInit：初始化某个 PHOTO 算法。
- HI_MPI_PHOTO_AlgDeinit：去初始化某个 PHOTO 算法。
- HI_MPI_PHOTO_AlgProcess：启动某个 PHOTO 算法的处理。
- HI_MPI_PHOTO_SetAlgCoef：设置某个 PHOTO 算法的图像效果调节系数。
- HI_MPI_PHOTO_GetAlgCoef：获取某个 PHOTO 算法的图像效果调节系数。

HI_MPI_PHOTO_AlgInit

【描述】

初始化某个算法。

【语法】

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

【参数】

参数名称	描述	输入/输出
enAlgType	算法的枚举值。	输入
pstPhotoInit	Photo 算法的初始化参数。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【芯片差异】

无。

【需求】

- 头文件：hi_comm_photo.h、mpi_photo.h
- 库文件：libmpi_photo.a

【注意】

- 调用该接口之前需要确保 DSP 端的 bin 文件已经加载成功。
- 该接口传入的内存是已经从 MMZ 中分配好的。
- 不同算法，不同分辨率需要的 Public 内存大小不同，建议使用 hi_comm_photo.h 中定义的 HDR_GetPublicMemSize，MFNR_GetPublicMemSize，SFNR_GetPublicMemSize，DE_GetPublicMemSize 这几个函数来获取不同分辨率对应的算法 Public 内存大小。

【举例】

无

【相关主题】

HI_MPI_PHOTO_AlgDeinit

HI_MPI_PHOTO_AlgDeinit

【描述】

去初始化某个算法。

【语法】

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

【参数】

参数名称	描述	输入/输出
enAlgType	算法的枚举值。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【芯片差异】

无

【需求】

- 头文件：hi_comm_photo.h、mpi_photo.h
- 库文件：libmpi_photo.a

【注意】

无

【举例】

无

【相关主题】

HI_MPI_PHOTO_AlgInit

HI_MPI_PHOTO_AlgProcess

【描述】

PHOTO 的算法处理。这个接口是阻塞接口，当前帧处理完成之后才返回。

多帧合成的算法需要调用多次，比如 HDR 三合一需要调用三次这个接口，每次传一帧输入的帧数据信息，调用后当前帧就开始处理。

【语法】

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

【参数】

参数名称	描述	输入/输出
enAlgType	算法的枚举值。	输入
pstPhotoAttr	算法处理的属性结构体。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【芯片差异】

无。

【需求】

- 头文件：hi_comm_photo.h、mpi_photo.h
- 库文件：libmpi_photo.a

【注意】

对应算法必须初始化成功后，才能调用该接口。

【举例】

无

【相关主题】

HI_MPI_PHOTO_AlgInit

HI_MPI_PHOTO_SetAlgCoef

【描述】

设置某个 PHOTO 算法的图像效果调节系数。

这个接口不是必设的接口，PHOTO 内部保存了一套默认的系数，如果不设置，就使用默认的值。

【语法】

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

【参数】

参数名称	描述	输入/输出
enAlgType	算法的枚举值。	输入
pstAlgCoef	算法系数的结构体。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【芯片差异】

无。

【需求】

- 头文件：hi_comm_photo.h、mpi_photo.h
- 库文件：libmpi_photo.a

【注意】

该接口可以在算法初始化之后，算法处理第一帧之前调用。

【举例】

无

【相关主题】

HI_MPI_PHOTO_GetAlgCoef

HI_MPI_PHOTO_GetAlgCoef

【描述】

获取某个 PHOTO 算法的图像效果调节系数。

【语法】

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

【参数】

参数名称	描述	输入/输出
enAlgType	算法的枚举值。	输入
pstAlgCoef	算法系数的结构体。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【芯片差异】

无。

【需求】

- 头文件：hi_comm_photo.h、mpi_photo.h
- 库文件：libmpi_photo.a

【注意】

无

【举例】

无

【相关主题】

HI_MPI_PHOTO_SetAlgCoef

2.5 数据类型

PHOTO 相关数据类型定义如下：

- PHOTO_ALG_TYPE_E：定义 PHOTO 算法类型的枚举。
- PHOTO_ALG_INIT_S：定义 PHOTO 算法初始化的结构体。
- PHOTO_ALG_ATTR_S：定义 PHOTO 算法处理的属性结构体。
- PHOTO_HDR_ATTR_S：定义 HDR 算法处理的结构体。
- PHOTO_SFNR_ATTR_S：定义 SFNR 算法处理的结构体。
- PHOTO_MFNR_ATTR_S：定义 MFNR 算法处理的结构体。
- PHOTO_DE_ATTR_S：定义 DE 算法处理的结构体。
- PHOTO_FACE_INFO_S：定义人脸区域信息的结构体。
- PHOTO_ALG_COEF_S：定义 PHOTO 算法系数的结构体。
- PHOTO_HDR_COEF_S：定义 HDR 算法系数的结构体。
- PHOTO_IMAGE_FUSION_PARAM_S：定义图像融合参数的结构体。
- PHOTO_DARK_MOTION_DETECTION_PARAM_S：定义 HDR 中鬼影检测参数的结构体。

- PHOTO_SFNR_COEF_S：定义 SFNR 算法系数的结构体。
- PHOTO_SFNR_ISO_STRATEGY_S：定义 SFNR 算法 ISO 策略的结构体。
- PHOTO_MFNR_COEF_S：定义 MFNR 算法系数的结构体。
- PHOTO_MFNR_3DNR_PARAM_S：定义 MFNR 算法中时域降噪系数的结构体。
- PHOTO_MFNR_3DNR_ISO_STRATEGY_S：定义 MFNR 算法时域降噪系数中每个 ISO 档位的结构体。
- PHOTO_MFNR_2DNR_PARAM_S：定义 MFNR 算法中空域降噪系数的结构体。
- PHOTO_MFNR_2DNR_ISO_STRATEGY_S：定义 MFNR 算法空域降噪系数中每个 ISO 档位的结构体。
- PHOTO_DE_COEF_S：定义 DE 算法系数的结构体。
- PHOTO_DE_ISO_STRATEGY_S：定义 DE 算法每个 ISO 档位对应策略的结构体。
- PHOTO_HDR_FRAME_NUM：定义 HDR 合成的帧数目。
- PHOTO_MFNR_FRAME_NUM：定义 MFNR 合成的帧数目。
- PHOTO_HDR_ISO_LEVEL_CNT：定义调节 HDR 图像效果的 ISO 档位数目。
- PHOTO_SFNR_ISO_LEVEL_CNT：定义调节 SFNR 图像效果的 ISO 档位数目。
- PHOTO_MFNR_ISO_LEVEL_CNT：定义调节 MFNR 图像效果的 ISO 档位数目。
- PHOTO_DE_ISO_LEVEL_CNT：定义调节 DE 图像效果的 ISO 档位数目。
- PHOTO_MAX_FACE_NUM：定义算法处理中最大的人脸区域的数目。
- PHOTO_MIN_WIDTH：定义算法处理支持的最小图像分辨率的宽。
- PHOTO_MIN_HEIGHT：定义算法处理支持的最小图像分辨率的高。
- PHOTO_MAX_WIDTH：定义算法处理支持的最大图像分辨率的宽。
- PHOTO_MAX_HEIGHT：定义算法处理支持的最大图像分辨率的高。

PHOTO_ALG_TYPE_E

【说明】

定义 PHOTO 算法类型的枚举。

【定义】

```
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;
```

【成员】

成员名称	描述
PHOTO_ALG_TYPE_HDR	HDR 合成算法。
PHOTO_ALG_TYPE_SFNR	SFNR，单帧降噪算法。
PHOTO_ALG_TYPE_MFNR	MFNR，多帧降噪算法。
PHOTO_ALG_TYPE_DE	DE，细节增强算法。

【注意事项】

无

【相关数据类型及接口】

- HI_MPI_PHOTO_AlgInit
- HI_MPI_PHOTO_AlgDeinit
- HI_MPI_PHOTO_AlgProcess
- HI_MPI_PHOTO_SetAlgCoef
- HI_MPI_PHOTO_GetAlgCoef

PHOTO_ALG_INIT_S

【说明】

定义 PHOTO 算法初始化的结构体。

【定义】

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

【成员】

成员名称	描述
u64PublicMemPhyAddr	给算法处理的内存起始物理地址。这块内存需要用户提前从 MMZ 区域分配。
u64PublicMemVirAddr	给算法处理的内存起始虚拟地址，这个值当前无效，PHOTO 内部并没有使用。
u32PublicMemSize	给算法处理的内存区域的长度。

成员名称	描述
bPrintDebugInfo	是否开启打印调试信息。 HI_TRUE：打印算法的版本号等调试信息。 HI_FALSE：除异常错误信息外，不打印其他的调试信息。

【注意事项】

无

【相关数据类型及接口】

HI_MPI_PHOTO_AlgInit

PHOTO_ALG_ATTR_S

【说明】

定义 PHOTO 算法处理的属性结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
stHDRAttr	HDR 算法处理的结构体。
stSFNRAttr	SFNR 算法处理的结构体。
stMFNRAttr	MFNR 算法处理的结构体。
stDEAttr	DE 算法处理的结构体。

【注意事项】

无

【相关数据类型及接口】

HI_MPI_PHOTO_AlgProcess

PHOTO_HDR_ATTR_S

【说明】

定义 HDR 算法处理的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
stSrcFrm	每次 HDR 处理的源图像帧信息。（结构体的定义请参考《HiMPP V4.0 媒体处理软件开发参考》文档，系统控制 章节的描述）
stDesFrm	HDR 处理的输出图像帧信息。需要用户提前从 VB 池中分配。 做第一帧 HDR 处理时就需要传入，后面两帧处理时，保存和第一帧的值一致。
u32FrmIndex	HDR 处理的帧序号，从 0 开始。 三帧 HDR 合成，该值应该依次为 0, 1, 2；
u32ISO	当前处理帧的 ISO 值。三帧的 ISO 值应该保持一致。
u32FaceNum	图像中检测到的人脸的个数。以中间那帧正常曝光帧上的人脸个数为准，其他两帧无效。
astFaceInfo	人脸区域的信息的结构体。

【注意事项】

无

【相关数据类型及接口】

PHOTO_ALG_ATTR_S

PHOTO_SFNR_ATTR_S

【说明】

定义 SFNR 算法处理的结构体。

【定义】

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

【成员】

成员名称	描述
stFrm	SFNR 单帧降噪的源图帧信息结构体。 SFNR 处理之后，将输出的图像写回到源图这块帧存。 （结构体的定义请参考《HiMPP V4.0 媒体处理软件开发参考》文档，系统控制 章节的描述）
u32ISO	当前处理帧的 ISO 值。

【注意事项】

无

【相关数据类型及接口】

PHOTO_ALG_ATTR_S

PHOTO_MFNR_ATTR_S

【说明】

定义 MFNR 算法处理的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
stSrcFrm	每次 MFNR 处理的源图像帧信息。 （结构体的定义请参考《HiMPP V4.0 媒体处理软件开发参考》文档，系统控制 章节的描述）
stDesFrm	MFNR 处理的输出图像帧信息。需要用户提前从 VB 池中分配。 做第一帧 MFNR 处理时就需要传入，后面三帧处理时，保存和第一帧的值一致。
stRawFrm	如果 MFNR 处理之后不需要做 DE 算法，那么该结构体不需要赋值。 如果 MFNR 处理之后需要做 DE 算法，那么该结构体为 DE 算法需要的 RAW 数据的结构体。 MFNR 的四帧处理中，只使用了 u32FrmIndex 为 0 的那帧对应的 RAW 数据，其余三帧传入的 RAW 数据结构体需要保持和 u32FrmIndex 为 0 的那帧 RAW 数据一致。
u32FrmIndex	MFNR 处理的帧序号，从 0 开始。 四帧 MFNR 处理，该值应该依次为 0, 1, 2, 3;
u32ISO	当前处理帧的 ISO 值。

【注意事项】

无

【相关数据类型及接口】

PHOTO_ALG_ATTR_S

PHOTO_DE_ATTR_S

【说明】

定义 DE 算法处理的结构体。

【定义】

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

【成员】

成员名称	描述
stFrm	DE 细节增强算法的源图 YUV 数据帧信息结构体。DE 处理之后，将输出的 YUV 图像写回到源图这块帧存。 （结构体的定义请参考《HiMPP V4.0 媒体处理软件开发参考》文档，系统控制 章节的描述）
stRawFrm	DE 细节增强算法需要的 RAW 数据帧信息结构体。
u32ISO	当前处理帧的 ISO 值。

【注意事项】

无

【相关数据类型及接口】

PHOTO_ALG_ATTR_S

PHOTO_FACE_INFO_S

【说明】

定义人脸区域信息的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
u32Id	当前人脸区域的 ID。
stFaceRect	当前人脸区域的坐标信息。 （结构体的定义请参考《HiMPP V4.0 媒体处理软件开发参考》文档，系统控制 章节的描述）
stLeftEyeRect	当前人脸中左眼的坐标信息。保留，暂时无效。
stRightEyeRect	当前人脸中右眼的坐标信息。保留，暂时无效。

成员名称	描述
u32BlinkScore	当前人脸中眼睛闭合程度的值。保留，暂时无效。
u32SmileScore	当前人脸中笑脸检测的程度的值。保留，暂时无效。

【注意事项】

无

【相关数据类型及接口】

PHOTO_HDR_ATTR_S

PHOTO_ALG_COEF_S

【说明】

定义 PHOTO 算法系数的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
stPhotoHdrCoef	HDR 算法系数的结构体。
stPhotoSfnrCoef	SFNR 算法系数的结构体。
stPhotoMfnrCoef	MFNR 算法系数的结构体。
stPhotoDECoef	DE 算法系数的结构体。

【注意事项】

无

【相关数据类型及接口】

- HI_MPI_PHOTO_SetAlgCoef
- HI_MPI_PHOTO_GetAlgCoef

PHOTO_HDR_COEF_S

【说明】

定义 HDR 算法系数的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
s32AjustRatio	人脸系数衰减长曝和短曝帧的权重。 取值范围：[0, 255]。
s32ImageScaleMethod	输出图像缩放插值方式。 0：双线性插值； 1：Lanczos 插值。
astMotionDetectionParam	鬼影检测参数结构体。
astImageFusionParam	图像融合参数的结构体。

【注意事项】

无

【相关数据类型及接口】

PHOTO_ALG_COEF_S

PHOTO_IMAGE_FUSION_PARAM_S

【说明】

定义图像融合参数的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
s32IsoSpeed	算法每个档位的 ISO 值。
s32PyramidTopSize	金字塔最顶层宽/高大小限制 取值范围：[16, 128]。
s32WeightCurveMethod	权重曲线： 0：公用一个高斯曲线 以亮度为变量的正态分布曲线，标准差为 BlendSigma 1：分别设置各自曲线，分段方式 每帧各自生成，每个曲线衰减部分由 BlendSigma 控制，越大衰减越快
s32WeightCalcMethod	权重查表： 0：表示各自帧按照亮度查各自对应曲线 1：表示在权重曲线 WeightCurveMethod=1 时候，只用中曝光亮度查三条曲线
s32BlendUVGain	融合后 UV 增强系数。 取值范围：[128,255]
s32DetailEnhanceRatioL0	金字塔重建，第 0 层回叠系数。 取值范围：[0,1024] 增强大边
s32DetailEnhanceRatioL1	金字塔重建，第 1 层回叠系数。 取值范围：[0,1024] 增强细节
f32BlendSigma	正态分布曲线标准差。 取值范围：[0.00,4.00]

【注意事项】

无

【相关数据类型及接口】

PHOTO_HDR_COEF_S

PHOTO_DARK_MOTION_DETECTION_PARAM_S

【说明】

定义 HDR 中鬼影检测参数的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
s32IsoSpeed	算法每个档位的 ISO 值。
s32MotionLowGray	参考帧暗区，运动检测阈值 取值范围：[0, 255]。
s32MotionHighGray	参考帧亮区，运动检测阈值 取值范围：[0, 255]。
f32MotionRatio	运动区域比例阈值，若大于阈值则不做运动矫正，丢弃当前处理帧 取值范围：[0, 100]。
s32NightAverageLuma	参考帧亮度判断夜景阈值 取值范围：[0, 255]。

【注意事项】

无

【相关数据类型及接口】

PHOTO_HDR_COEF_S

PHOTO_SFNR_COEF_S

【说明】

定义 SFNR 算法系数的结构体。

【定义】

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

【成员】

成员名称	描述
astIsoStrat	算法每 ISO 档位对应的结构体。

【注意事项】

无

【相关数据类型及接口】

PHOTO_ALG_COEF_S

PHOTO_SFNR_ISO_STRATEGY_S

【说明】

定义 SFNR 算法 ISO 策略的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
s32IsoValue	算法每个档位的 ISO 值。

成员名称	描述
s32Luma	亮度低频去噪强度，数字越大去噪强度越强 取值范围：[-100, 0]。
s32Chroma	色度低频去噪强度，数字越大去噪强度越强 取值范围：[-100, 0]。
s32LumaHF	亮度中高频去噪强度，数字越大去噪强度越强 取值范围：[-100, 0]。
s32ChromaHF	色度中高频去噪强度，数字越大去噪强度越强 取值范围：[-100, 0]。

【注意事项】

无

【相关数据类型及接口】

PHOTO_SFNR_COEF_S

PHOTO_MFNR_COEF_S

【说明】

定义 MFNR 算法系数的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
bImageScale	MFNR 输出的结果是否将配准公共区域裁出缩放到原图大小。 HI_TRUE：表示缩放配准的公共区域； HI_FALSE：不缩放，公共区域外采用填充方式；
st3DNRParam	MFNR 算法中时域降噪系数的结构体。

成员名称	描述
st2DNRParam	MFNR 算法中空域降噪系数的结构体。
bDEEnable	MFNR 处理之后，是否再做 DE 算法。
stPhotoDECoef	DE 算法系数的结构体。

【注意事项】

无

【相关数据类型及接口】

PHOTO_ALG_COEF_S

PHOTO_MFNR_3DNR_PARAM_S

【说明】

定义 MFNR 算法中时域降噪系数的结构体。

【定义】

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

【成员】

成员名称	描述
ast3DNRIsoStrat	MFNR 算法时域降噪系数中每个 ISO 档位对应的结构体。

【注意事项】

无

【相关数据类型及接口】

PHOTO_MFNR_COEF_S

PHOTO_MFNR_3DNR_ISO_STRATEGY_S

【说明】

定义 MFNR 算法时域降噪系数中每个 ISO 档位的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
s32IsoValue	算法每个档位的 ISO 值。
s32Stnr	使用时域与空域的强度比，数字越大使用越多时域效果。 取值范围：[0, 255]
s32TnrFrmNum	参与时域降噪的帧数。 取值范围：[1, 4]
s32StnrDarkLess	暗处使用空域效果的强度，像素值小于此值时使用较多空域效果。 取值范围：[0, 255]
s32StnrGhostLess	鬼影区域模糊强度，数字越大越模糊。 取值范围：[0, 255]
s32LumaAlpha	亮度比阈值，当两帧亮度比大于此阈值时，执行去噪或去鬼影。 取值范围：[0, 255]。
s32LumaDelta	亮度差阈值，当两帧亮度之差大于此阈值时，执行去噪或去鬼影。 取值范围：[0, 255]
s32ChromaAlpha	色度比阈值，当两帧色度比大于此阈值时，执行去噪或去鬼影 取值范围：[0, 255]
s32ChromaDelta	色度差阈值，当两帧色度之差大于此阈值时，执行去噪或去鬼影。 取值范围：[0, 255]

【注意事项】

无

【相关数据类型及接口】

PHOTO_MFNR_3DNR_PARAM_S

PHOTO_MFNR_2DNR_PARAM_S

【说明】

定义 MFNR 算法中空域降噪系数的结构体。

【定义】

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

【成员】

成员名称	描述
ast2DNRIsoStrat	MFNR 算法空域降噪系数中每个 ISO 档位对应的结构体。

【注意事项】

无

【相关数据类型及接口】

PHOTO_MFNR_COEF_S

PHOTO_MFNR_2DNR_ISO_STRATEGY_S

【说明】

定义 MFNR 算法空域降噪系数中每个 ISO 档位的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
s32IsoValue	算法每个档位的 ISO 值。
s32Luma	亮度低频去噪强度，数字越大去噪强度越强。 取值范围：[-100, 0]。
s32Chroma	色度低频去噪强度，数字越大去噪强度越强。 取值范围：[-100, 0]。
s32LumaHF2	亮度中高频去噪强度，数字越大去噪强度越强。 取值范围：[-100, 0]。
s32DetailMin	亮度高频保留强度，数字越大保留越多。 取值范围：[0, 8]。

【注意事项】

无

【相关数据类型及接口】

PHOTO_MFNR_3DNR_PARAM_S

PHOTO_DE_COEF_S

【说明】

定义 DE 算法系数的结构体。

【定义】

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

【成员】

成员名称	描述
astDEIsoStrat	DE 算法中每个 ISO 档位对应策略的结构体。

【注意事项】

无

【相关数据类型及接口】

PHOTO_ALG_COEF_S

PHOTO_DE_ISO_STRATEGY_S

【说明】

定义 DE 算法每个 ISO 档位对应策略的结构体。

【定义】

```
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;
```

【成员】

成员名称	描述
s32IsoValue	算法每个档位的 ISO 值。
s32GlobalGain	全局细节强度系数，数字越大越强。 取值范围：[0, 255]。
s32GainLF	低频比例，数字越大越强。 取值范围：[0, 255]。
s32GainHFPos	高频白点比例，数字越大越强。 取值范围：[0, 255]。
s32GainHFNeg	高频黑点比列，数字越大越强。 取值范围：[0, 255]。
s32LumaScaleX0	亮度拉伸低阈值。 取值范围：[0, 255]。
s32LumaScaleX1	亮度拉伸高阈值。 取值范围：[0, 255]。

成员名称	描述
s32LumaScaleY1	亮度拉伸高阈值，对应衰减系数。 取值范围：[0, 255]。
s32SatuGainX0	饱和度低阈值。 取值范围：[0, 255]。
s32SatuGainX1	饱和度高阈值。 取值范围：[0, 255]。
s32SatuGainY1	饱和度高阈值，对应衰减系数。 取值范围：[0, 255]。

【注意事项】

无

【相关数据类型及接口】

PHOTO_DE_COEF_S

PHOTO_HDR_FRAME_NUM

【说明】

定义 HDR 合成的帧数目。

【定义】

```
#define PHOTO_HDR_FRAME_NUM 3
```

【注意事项】

无。

【相关数据类型及接口】

无。

PHOTO_MFNR_FRAME_NUM

【说明】

定义 MFNR 合成的帧数目。

【定义】

```
#define PHOTO_MFNR_FRAME_NUM 4
```

【注意事项】

无。

【相关数据类型及接口】

无。

PHOTO_HDR_ISO_LEVEL_CNT

【说明】

定义调节 HDR 图像效果的 ISO 档位数目。

【定义】

```
#define PHOTO_HDR_ISO_LEVEL_CNT 10
```

【注意事项】

无。

【相关数据类型及接口】

无。

PHOTO_SFNR_ISO_LEVEL_CNT

【说明】

定义调节 SFNR 图像效果的 ISO 档位数目。

【定义】

```
#define PHOTO_SFNR_ISO_LEVEL_CNT 8
```

【注意事项】

无。

【相关数据类型及接口】

无。

PHOTO_MFNR_ISO_LEVEL_CNT

【说明】

定义调节 MFNR 图像效果的 ISO 档位数目。

【定义】

```
#define PHOTO_MFNR_ISO_LEVEL_CNT    8
```

【注意事项】

无。

【相关数据类型及接口】

无。

PHOTO_DE_ISO_LEVEL_CNT

【说明】

定义调节 DE 图像效果的 ISO 档位数目。

【定义】

```
#define PHOTO_DE_ISO_LEVEL_CNT 8
```

【注意事项】

无。

【相关数据类型及接口】

无。

PHOTO_MAX_FACE_NUM

【说明】

定义算法处理中最大的人脸区域的数目。

【定义】

```
#define PHOTO_MAX_FACE_NUM  10
```

【注意事项】

无。

【相关数据类型及接口】

无。

PHOTO_MIN_WIDTH

【说明】

定义算法处理支持的最小图像分辨率的宽。

【定义】

```
#define PHOTO_MIN_WIDTH    1280
```

【注意事项】

无。

【相关数据类型及接口】

无。

PHOTO_MIN_HEIGHT

【说明】

定义算法处理支持的最小图像分辨率的高。

【定义】

```
#define PHOTO_MIN_HEIGHT    720
```

【注意事项】

无。

【相关数据类型及接口】

无。

PHOTO_MAX_WIDTH

【说明】

定义算法处理支持的最大图像分辨率的宽。

【定义】

```
#define PHOTO_MAX_WIDTH    7680
```

【注意事项】

无。

【相关数据类型及接口】

无。

PHOTO_MAX_HEIGHT

【说明】

定义算法处理支持的最大图像分辨率的高。

【定义】

```
#define PHOTO_MAX_HEIGHT    4320
```

【注意事项】

无。

【相关数据类型及接口】

无。

2.6 错误码

PHOTO 模块 API 错误码如表 2-1 所示。

表2-1 PHOTO 模块 API 错误码

错误代码	宏定义	描述
0xA0268001	HI_ERR_PHOTO_INVALID_DEVID	拍照算法设备号无效
0xA0268002	HI_ERR_PHOTO_INVALID_CHNID	拍照算法通道号无效
0xA0268003	HI_ERR_PHOTO_ILLEGAL_PARAM	输入参数设置无效
0xA0268004	HI_ERR_PHOTO_EXIST	拍照算法已存在
0xA0268005	HI_ERR_PHOTO_UNEXIST	拍照算法不存在
0xA0268006	HI_ERR_PHOTO_NULL_PTR	输入参数空指针错误
0xA0268007	HI_ERR_PHOTO_NOT_CONFIG	拍照算法属性未配置
0xA0268008	HI_ERR_PHOTO_NOT_SUPPORT	操作不支持
0xA0268009	HI_ERR_PHOTO_NOT_PERM	操作不允许
0xA026800C	HI_ERR_PHOTO_NOMEM	分配内存失败
0xA026800D	HI_ERR_PHOTO_NOBUF	分配缓存失败
0xA026800E	HI_ERR_PHOTO_BUF_EMPTY	缓存为空
0xA0268010	HI_ERR_PHOTO_NOTREADY	系统未初始化
0xA0268012	HI_ERR_PHOTO_BUSY	系统忙

3 监控类抓拍方案使用指南

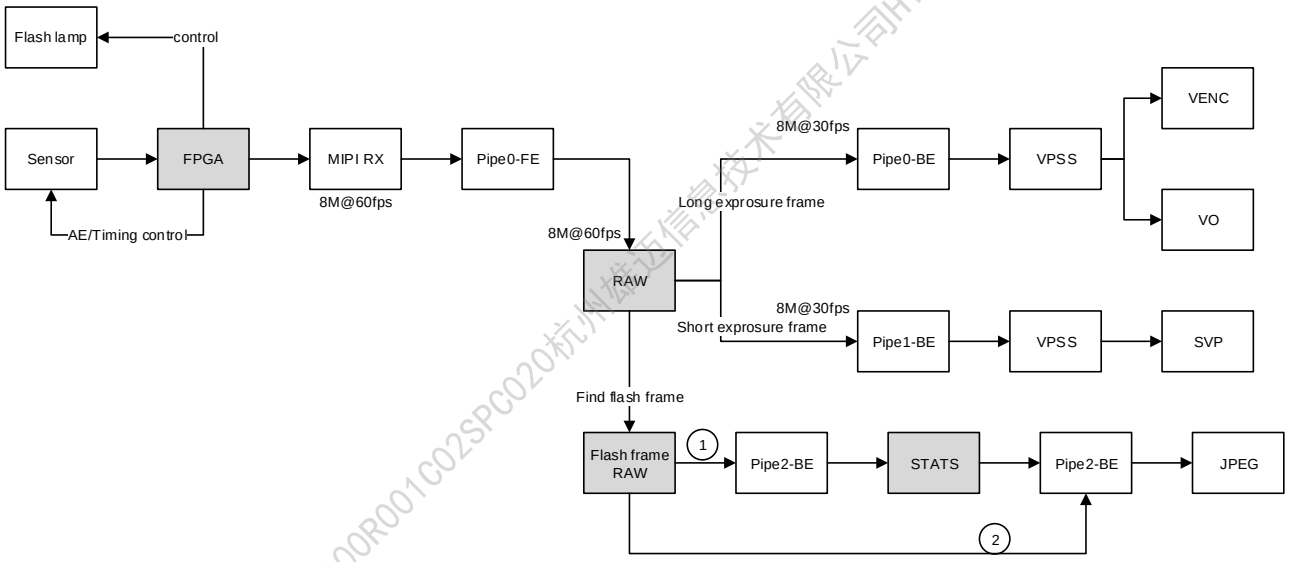
3.1 概述

监控类抓拍方案方案主要是面向监控产品中的电子警察、卡口监控等场景下的抓拍解决方案。监控类抓拍方案和消费类抓拍的显著区别是客户的定制化程度较高，与外设的交互也很多。

3.2 数据通路

监控类抓拍机的产品形态较多，典型的数据通路如图 3-1 所示。

图3-1 监控类抓拍机典型数据通路



监控类抓拍机需要控制外部设备爆闪灯和频闪灯，同时需要接收交通信号灯信号、抓拍触发信号等。sensor 的时序控制和 AE 调节需要和交通信号灯保持同步，因此需要使用客户自定义的 ISP 3A 算法。

实时性要求较高的场景，一般是由客户实现的 FPGA 来完成 sensor 时序的控制，爆闪灯和频闪灯的控制等。抓拍帧的标记也是由 FPGA 来完成，一般是在 sensor 输入的有效数据后额外增加几行有效数据，在里面标识是否是抓拍帧，以及其他一些客户需要的自定义信息。

从 Pipe0-FE 采集的 RAW 数据需要识别抓拍帧，同时送给不同的 Pipe 做后续处理。在送 BE 做处理之前，需要客户自己将 RAW 数据中额外增加的那几行自定义信息裁剪掉，防止对 BE 处理的图像效果有影响。当前 VI Pipe 要求 FE 写出的宽高和 BE 处理的宽高必须一致，所以 Pipe0 那一路 FE 写出的 RAW 经过裁剪后无法再送到 Pipe0 那一路的 BE 做处理，可以再起一个新的 Pipe 来替代 Pipe0-BE。

抓拍帧的处理和视频帧不同，视频帧的 ISP 图像效果参数配置可以参考前一帧的统计信息计算当前帧的配置。但是抓拍帧只有一帧，没用参考帧，因此需要将抓拍帧的 RAW 送到 Pipe2-BE 两次，第一次用于生成抓拍帧的统计信息，然后根据统计信息计算出 ISP 的参数配置信息，然后再用正确的参数配置信息和抓拍帧 RAW 数据再做一次 BE 的处理，才能生成正确的抓拍帧 YUV 数据。

ISP 算法的运行一般是由每帧的帧起始中断来驱动，但是在抓拍机的场景中都是离线送 RAW 数据给 BE 做处理的，并不适合用帧起始中断，因此需要用户程序来驱动 ISP 运行。在每次送 RAW 数据给 BE 做处理之前，需要调用 HI_MPI_ISP_RunOnce 生成当前帧需要的寄存器配置信息，替代 HI_MPI_ISP_Run 函数做的处理。

3.3 程序调用流程

程序调用流程请参考 mpp/sample/ traffic_capture 中的代码实现。

程序中用到的接口请参考《HiISP 开发参考》和《HiMPP V4.0 媒体处理软件开发参考》文档中的描述。