ISP 模块用户指南

文档版本号：SDK-V1.0

发布日期：2019.03.30

商标声明


、全志和其他全志商标均为珠海全志科技股份有限公司的商标。

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、服务或特性等应受全志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，全志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

前 言

概述

产品版本

产品名称	产品版本
V316	V1.0

读者对象

本文档（本指南）主要适用于以下工程师：

- 电子产品设计开发人员

修订记录

版本号	修订日期	修订内容
V1.0	2019-03-30	V316 初始化版本

目 录

1. ISP 图像处理模块

1.1. ISP 简介

ISP 模块主要用于解析 CMOS sensor 输出的图像数据，其主要功能如下：

- 最大分辨率：MIPI CSI (静态图片：16M，视频 8M@30fps)，Parallel CSI (静态图片：5M，视频 1080P@30fps)
- 支持 MIPI、LVDS、HiSPi、BT.656、BT1120 时序
- 支持用于 NTSC 和 PAL 制式的 CCIR656 协议
- 支持格式: YUV422-8bit/10bit, YUV420-8bit/10bit, RAW-8,RAW-10, RAW-12, RGB888, RGB565
- 支持 DDR 采样模式
- 支持黑电平校正、坏点校正、镜头阴影、2D/3D 降噪、色彩增强、3A 等。

ISP 算法包含硬件算法和软件算法库两部分：硬件算法集成在 SOC 上，称为 hawkview ISP；软件算法服务于 ISP 硬件算法，故称为 ISP server。

1.2. ISP Server 简介

ISP Server 模块主要包括 ISP 算法库和 ISP 中间件部分：

ISP 算法库部分，其主要用于在 ISP 运行时图像效果的处理，包括 3A 算法以及一系列 ISP 正常运行所需的基本算法；

ISP 中间件部分，其主要用于控制 ISP 以及 Sensor 驱动、响应 Camera 应用以及 Tuning 工具命令、调度 ISP 相关算法等，包括事件管理、Pipeline 管理、Buffer 管理以及算法调度等模块。

ISP 算法库、中间件、驱动以及 Camera 应用相互关系如下图所示：

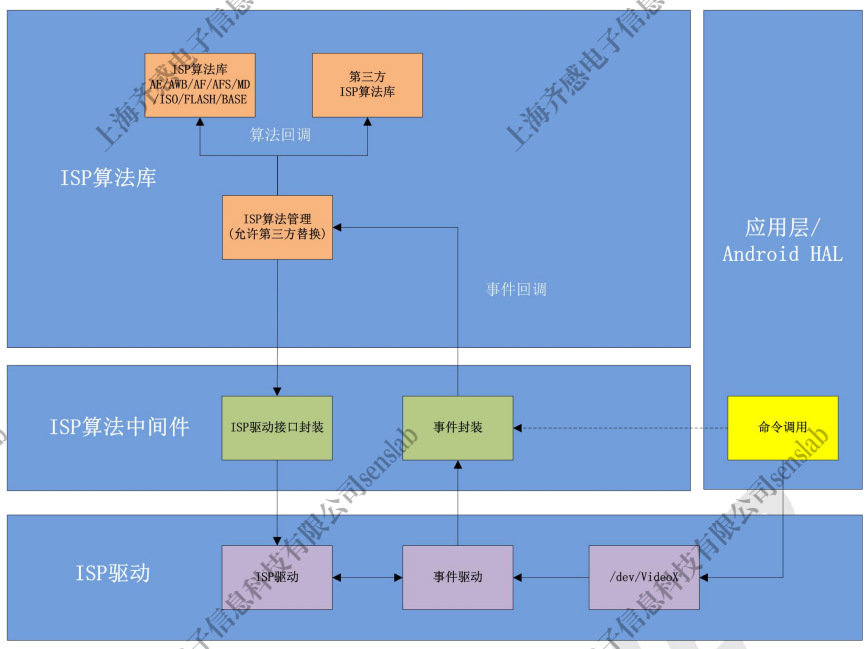

图 1-1 ISP Server 基本框架

1.3. ISP tool 简介

Hawkview ISP 调试工具可以通过局域网络连接单板在线调试 ISP 各个模块的参数，使用标定分析工具进行各类数据分析，使用 rtsp 工具实时预览图像效果等。使用 ISP 调试工具时，需要在小机端运行包含 isp server 的应用，用于与工具交互。

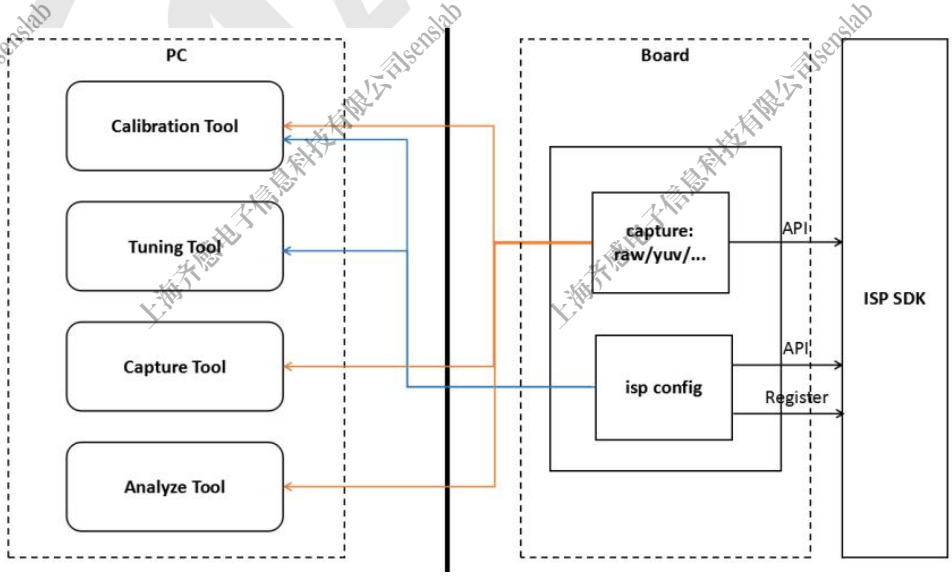

图 1-2 ISP 调试工具框图

1.4. ISP 第三方 3A 算法库

ISP server 支持客户使用自己的 3A 算法库替换原有的 3A 算法库，开发者可以在 isp server 中拿到 3A 的统计值，然后输出给 isp server 对应的运算结果即可。如 AWB 需要输出 RGB 的增益；AE 需要输出 sensor 的曝光时间、模拟增益、数字增益、光圈值等；AF 需要返回对焦马达的位置值。

2. API 接口

2.1 系统控制

系统控制部分主要是启动 ISP Firmware、运行 ISP Firmware、退出 ISP Firmware. 设置 ISP 模块开关。

AW_MPI_ISP_Init

AW_MPI_ISP_Run

AW_MPI_ISP_Stop

AW_MPI_ISP_Exit

AW_MPI_ISP_SetModuleOnOff

AW_MPI_ISP_GetModuleOnOff

AW_MPI_ISP_Init

【目的】

初始化 ISP 设备

AW_S32 AW_MPI_ISP_Init();

【参数】

参数	描述
IspDev	ISP 设备 ID 号

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

在进行 ISP 相关操作前应该首先调用此接口，保证 ISP 设备已经初始化。AW_MPI_ISP_Run 前只调用一次。

【举例】

```
/*declaration*/
int ret = 0;
/* init ISP device*/
ret = AW_MPI_ISP_Init();
```

```
if (AW_SUCCESS != ret)
{
return -1;
}
```

AW_MPI_ISP_Run

【目的】

运行 ISP 设备。

【语法】

AW_S32 AW_MPI_ISP_Run(ISP_DEV IspDev);

【参数】

参数	描述
IspDev	ISP 设备 ID 号

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

在进行 ISP 相关操作前应该首先调用此接口，保证 ISP 设备已经初始化。

【举例】

```
/*declaration*/
int ret = 0, isp_id = 0;
/* init ISP device*/
ret = AW_MPI_ISP_Run(isp_id);
if (AW_SUCCESS != ret)
{
return -1;
}
```

AW_MPI_ISP_Stop

【目的】

ISP 设备退出。

【语法】

AW_S32 AW_MPI_ISP_Stop(ISP_DEV IspDev);

【参数】

参数	描述
IspDev	ISP 设备 ID 号

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

ISP 设备退出，不能再对 ISP 设备进行操作

【举例】

```
/*declaration*/
int ret = 0, isp_id = 0;
/* init ISP device*/
ret = AW_MPI_ISP_Stop(isp_id);
if (AW_SUCCESS != ret)
{
return -1;
}
```

AW_MPI_ISP_Exit

【目的】

ISP 设备退出。

【语法】

AW_S32 AW_MPI_ISP_Exit();

【参数】

参数	描述
IspDev	ISP 设备 ID 号

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

ISP 设备退出，不能再对 ISP 设备进行操作。AW_MPI_ISP_Exit 全部退出后只调用一次。

【举例】

```
/*declaration*/
int ret = 0, isp_id = 0;
/* init ISP device*/
ret = AW_MPI_ISP_Exit();
if (AW_SUCCESS != ret)
{
return -1;
}
```

AW_MPI_ISP_SetModuleOnOff

【目的】

开启或者关闭 ISP Pipeline 的功能模块。

【语法】

AW_S32 AW_MPI_ISP_SetModuleOnOff(ISP_DEV IspDev, ISP_MODULE_ONOFF *pstIspModuleOnOff);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
ISP_MODULE_ONOFF	pstIspModuleOnOff 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

开启 ISP Pipeline 的功能模块

【举例】

AW_MPI_ISP_GetModuleOnOff

【目的】

获取 ISP Pipeline 的功能模块开关信息。

【语法】

AW_S32 AW_MPI_ISP_GetModuleOnOff(ISP_DEV IspDev, ISP_MODULE_ONOFF *pstIspModuleOnOff);

【参数】

参数	描述
IspDev	ISP 设备 ID 号

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

2.2 AE

Hawkview ISP AE 实现的功能是：通过测光系统获取的统计信息计算出图像亮度，并与目标亮度对比得出 sensor 的曝光时间和模拟增益以及镜头的光圈值以及 ISP 的数字增益，并配置给对应的硬件模块，从而获得期望的图像亮度。AE 算法支持光圈优先、快门优先、增益优先三种模式，同时 AE 算法也支持去工频闪烁。

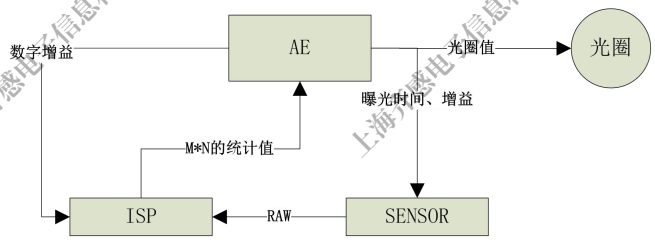

图 2.1 AE 模块流程

AW_MPI_ISP_AE_SetMode

AW_MPI_ISP_AE_GetMode

AW_MPI_ISP_AE_SetExposureBias

AW_MPI_ISP_AE_GetExposureBias

AW_MPI_ISP_AE_SetExposure

AW_MPI_ISP_AE_GetExposure

AW_MPI_ISP_AE_SetGain

AW_MPI_ISP_AE_GetGain

AW_MPI_ISP_AE_SetMode

【目的】

设置 AE 自动、手动模式。

【语法】

AW_S32 AW_MPI_ISP_AE_SetMode(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	0：自动曝光 1：手动曝光 动态属性

返回值	描述
AW_SUCCESS	成功
错误码	

【返回值】

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_AE_GetMode

【目的】

获取 AE 自动、手动模式。

【语法】

AW_S32 AW_MPI_ISP_AE_GetMode(ISP_DEV IspDev, int *Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	0：自动曝光 1：手动曝光

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_AE_SetExposureBias

【目的】

设置曝光偏移。

【语法】

AW_S32 AW_MPI_ISP_AE_SetExposureBias(ISP_DEV IspDev, int Value);

【参数】

参数	描述

IspDev	ISP 设备 ID 号
int Value	曝光偏移值 [0~7] 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_AE_GetExposureBias

【目的】

获取曝光偏移。

【语法】

AW_S32 AW_MPI_ISP_AE_GetExposureBias(ISP_DEV IspDev, int *Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	曝光偏移值 [1~8]

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_AE_SetExposure

【目的】

设置曝光时间。

【语法】

AW_S32 AW_MPI_ISP_AE_SetExposure(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	曝光时间 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_AE_GetExposure

【目的】

获取曝光时间。

【语法】

AW_S32 AW_MPI_ISP_AE_GetExposure(ISP_DEV IspDev, int *Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	曝光时间

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_AE_SetGain

【目的】

设置增益。

【语法】

AW_S32 AW_MPI_ISP_AE_SetGain(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	设置增益 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_AE_GetGain

【目的】

获取增益。

【语法】

AW_S32 AW_MPI_ISP_AE_GetGain(ISP_DEV IspDev, int *Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	获取增益

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

2.3 AWB

Hawkview ISP AWB 实现的功能是：通过 ISP 获取的统计信息计算出 Rgain 和 Bgain，并与调试工具测量的色温曲线进行对比得到对应的色温以及当前色温下的 RGB 各自的增益，从而获取标准日光下的图像颜色。AWB 算法支持色调的偏好设置，支持蓝天、草地、肤色等特殊场景的处理。

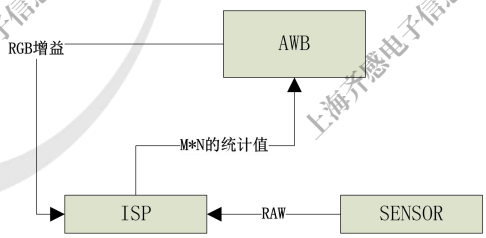

图 3-1 AWB 模块流程

AW_MPI_ISP_AWB_SetMode

AW_MPI_ISP_AWB_GetMode

AW_MPI_ISP_AWB_SetColorTemp

AW_MPI_ISP_AWB_GetColorTemp

AW_MPI_ISP_AWB_SetRGain

AW_MPI_ISP_AWB_GetRGain

AW_MPI_ISP_AWB_SetBGain

AW_MPI_ISP_AWB_GetBGain

AW_MPI_ISP_AWB_SetMode

【目的】

设置 AWB 自动、手动模式。

【语法】

AW_S32 AW_MPI_ISP_AWB_SetMode(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	0：自动白平衡 1：手动白平衡 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_AWB_GetMode

【目的】

获取 AWB 自动、手动模式。

【语法】

AW_S32 AW_MPI_ISP_AWB_GetMode(ISP_DEV IspDev, int *Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	0：自动白平衡 1：手动白平衡

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_AWB_SetColorTemp

【目的】

设置色温参数。

【语法】

AW_S32 AW_MPI_ISP_AWB_SetColorTemp(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	手动白平衡才生效 2：白炽灯 3：荧光灯 4：高亮荧光灯 5：日出 6：日光 7：闪光灯 8：多云

	9：阴天 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_AWB_GetColorTemp

【目的】

获取色温参数。

【语法】

AW_S32 AW_MPI_ISP_AWB_GetColorTemp(ISP_DEV IspDev, int *Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	手动白平衡才生效 2：白炽灯 3：荧光灯 4：高亮荧光灯 5：日出 6：日光 7：闪光灯 8：多云 9：阴天

【返回值】

返回值	描述

AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_AWB_SetRGain

【目的】

设置 R Gain。暂时不支持

【语法】

AW_S32 AW_MPI_ISP_AWB_SetRGain(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
Value	RGain 的值 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_AWB_GetRGain

【目的】

获取 R Gain。暂时不支持

【语法】

AW_S32 AW_MPI_ISP_AWB_GetRGain(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_AWB_SetBGain

【目的】

设置 B Gain。暂时不支持

【语法】

AW_S32 AW_MPI_ISP_AWB_SetBGain(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
Value	BGain 的值 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_AWB_GetBGain

【目的】

获取 B Gain。暂时不支持

【语法】

AW_S32 AW_MPI_ISP_AWB_GetBGain(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

2.4 ISP 接口

函数列表

AW_MPI_ISP_SetFlicker//0:disable,1:50,2:60,3:auto

AW_MPI_ISP_GetFlicker

AW_MPI_ISP_SetBrightness

AW_MPI_ISP_GetBrightness

AW_MPI_ISP_SetContrast

AW_MPI_ISP_GetContrast

AW_MPI_ISP_SetSaturation

AW_MPI_ISP_GetSaturation

AW_MPI_ISP_SetSharpness

AW_MPI_ISP_GetSharpness

AW_MPI_ISP_SetPltmWDR

AW_MPI_ISP_GetPltmWDR

AW_MPI_ISP_SetNRAttr

AW_MPI_ISP_GetNRAttr

AW_MPI_ISP_Set3NRAttr

AW_MPI_ISP_Get3NRAttr

AW_MPI_ISP_SetFlicker

【目的】

设置光源频率。

【语法】

AW_S32 AW_MPI_ISP_SetFlicker(ISP_DEV IspDev, int Value); //0:disable,1:50,2:60,3:auto

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	1:50Hz 2:60Hz 3:自动 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_GetFlicker

【目的】

获取光源频率。

【语法】

AW_S32 AW_MPI_ISP_GetFlicker(ISP_DEV IspDev, int *Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	1:50Hz 2:60Hz 3:自动

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_SetBrightness

【目的】

设置亮度。

【语法】

AW_S32 AW_MPI_ISP_SetBrightness(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	亮度值[-126, 126] 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_GetBrightness

【目的】

获取亮度。

【语法】

AW_S32 AW_MPI_ISP_GetBrightness(ISP_DEV IspDev, int *Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	亮度值[-126, 126]

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_SetContrast

【目的】

设置对比度。

【语法】

AW_S32 AW_MPI_ISP_SetContrast(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	对比度值[-126, 126] 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_GetContrast

【目的】

获取对比度。

【语法】

AW_S32 AW_MPI_ISP_GetContrast(ISP_DEV IspDev, int *Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	对比度值[-126, 126]

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_SetSaturation

【目的】

设置饱和度。

【语法】

AW_S32 AW_MPI_ISP_SetSaturation(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	饱和度值[-256, 512] 动态属性

【返回值】512

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_GetSaturation

【目的】

获取饱和度。

【语法】

AW_S32 AW_MPI_ISP_GetSaturation(ISP_DEV IspDev, int *Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	饱和度值[-256, 512]

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_SetSharpness

【目的】

设置锐利度。

【语法】

AW_S32 AW_MPI_ISP_SetSharpness(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	锐利度值[0,10] 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_GetSharpness

【目的】

获取锐利度。

【语法】

AW_S32 AW_MPI_ISP_GetSharpness(ISP_DEV IspDev, int *Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	锐利度值[0,10]

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_SetPltmWDR

【目的】

设置 Dynamic 数字宽动态属性。

【语法】

AW_S32 AW_MPI_ISP_SetPltmWDR(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号

int Value	宽动态[0, 255] 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_GetPltmWDR

【目的】

获取 Dynamic 数字宽动态属性。

【语法】

AW_S32 AW_MPI_ISP_GetPltmWDR(ISP_DEV IspDev, int *Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	宽动态[0, 255]

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_SetNRAttr

【目的】

设置 2DNR 属性参数。

【语法】

AW_S32 AW_MPI_ISP_SetNRAttr(ISP_DEV IspDev, int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	2D 降噪[0, 100] 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_GetNRAttr

【目的】

获取 2DNR 属性参数。

【语法】

AW_S32 AW_MPI_ISP_GetNRAttr(ISP_DEV IspDev, int *Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	2D 降噪[0, 100]

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_Set3NRAttr

【目的】

设置 3DNR 属性参数。

【语法】

AW_S32 AW_MPI_ISP_Set3NRAttr(ISP_DEV IspDev,int Value);

【参数】

参数	描述
IspDev	ISP 设备 ID 号
int Value	3D 降噪[0, 100] 动态属性

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

AW_MPI_ISP_Get3NRAttr

【目的】

获取 3DNR 属性参数。

【语法】

AW_S32 AW_MPI_ISP_Get3NRAttr(ISP_DEV IspDev, int *Value);

【参数】

参数	描述

IspDev	ISP 设备 ID 号
int Value	3D 降噪[0, 100]

【返回值】

返回值	描述
AW_SUCCESS	成功
错误码	

【需求】

头文件：mpi_isp.h

库文件：libmpp.a

【注意】

【举例】

2.5. 数据结构

2.6. 错误码

错误码	宏定义	描述

3. Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner.

The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This datasheet neither states nor implies warranty of any kind, including fitness for any particular application.