SDK6 API A12 Image Kernel

Version 1.4

February 25, 2015

Confidentiality Notice:

The contents of this document are proprietary and confidential information of Ambarella Inc.

The material in this document is for information only. Ambarella assumes no responsibility for errors or omissions and reserves the right to change, without notice, product specifications, operating characteristics, packaging, ordering, etc. Ambarella assumes no liability for damage resulting from the use of information contained in this document. All brands, product names and company names are trademarks of their respective owners.

US
3101 Jay Street
Ste. 110
Santa Clara, CA 95054, USA
Phone: +1.408.734.8888
Fax: +1.408.734.0788

Taiwan
Suite C1, No. 1, Li-Hsin Road 1
Science-Based Industrial Park
Hsinchu 30078, Taiwan
Phone: +886.3.666.8828
Fax: +886.3.666.1282

Hong Kong
Unit A&B, 18/F, Spectrum Tower
53 Hung To Road
Kwun Tong, Kowloon
Phone: +85.2.2806.8711
Fax: +85.2.2806.8722

Japan - Yokohama
Shin-Yokohama Business Center Bldg. 5th Floor
3-2-6 Shin-Yokohama, Kohoku-ku,
Yokohama, Kanagawa, 222-0033, Japan
Phone: +81.45.548.6150
Fax: +81.45.548.6151

Korea
6 Floor, Hanwon-Bldg.
Sunae-Dong, 6-1, Bundang-Gu
SeongNam-City, Kyunggi-Do
Republic of Korea 463-825
Phone: +031.717.2780
Fax: +031.717.2782

China - Shenzhen
Unit E, 5th Floor
No. 2 Finance Base
8 Ke Fa Road
Shenzhen, 518057, China
Phone: +86.755.3301.0366
Fax: +86.755.3301.0966

China - Shanghai
9th Floor, Park Center
1088 Fangdian Road, Pudong New District
Shanghai 201204, China
Phone: +86.21.6088.0608
Fax: +86.21.6088.0366

I Contents

II Preface

This document provides technical details using a set of consistent typographical conventions to help the user differentiate key concepts at a glance.

Conventions include:

Example	Description
AmbaGuiGen, **DirectUSB** **Save, File > Save** **Power, Reset, Home**	Software names GUI commands and command sequences Computer / Hardware buttons
Flash_IO_control **da**, **status**, **enable**	Register names and register fields. For example, **Flash_IO_control** is the register for global control of Flash I/O, and bit 17 (**da)** is used for DMA acknowledgement.
GPIO81, **CLK_AU**	Hardware external pins
VIL, VIH, VOL, VOH	Hardware pin parameters
INT_O, RXDATA_I	Hardware pin signals
AmbaI2C_Init() **AMBA_I2C_CTRL_s** **AMBA_I2C_CHANNEL_e** **AMBA_GIC_ISR_f** **AMBA_KAL_TASK_t**	API Functions API Structures API Enumerations API Function pointers API Typedef of ThreadX kernel abstraction layer

`DSC_Platform\Tools`
`AmbaI2C.h`
`RetStatus = AmbaI2C_Init();`
	User entries into software dialogues and GUI windows
	File names and paths
	Command line scripting and Code

Table II-1. Typographical Conventions for Technical Documents.

Additional Ambarella typographical conventions include:

- Acronyms are given in UPPER CASE using the default font (e.g., AHB, ARM11 and DDRIO).
- Names of Ambarella documents and publicly available standards, specifications, and databooks appear in *italic* type.

1 Overview

This chapter provides an overview of the Ambarella Image Kernel API. The Overview chapter is organized as follows:

1.1 Overview: Introduction

The Ambarella Image Kernel API is designed for use with the A12 family of digital camera processors. The Image Kernel API allows the configuration of multiple image-tuning parameters. In the A12 environment, the image processing occurs in three domains:

1. CFA domain before demosaicing
2. RGB domain before converting to YUV
3. YUV domain before entering the image encoder

Key features supported by the Image Kernel API include:

- AAA (Auto-exposure, Auto-focus, Auto-white-balance) statistics
- Lens and sensor correction
- Color processing
- Noise removal
- Sharpening

The Image Kernel API document is organized as follows:

1.2 Overview: Image Processing Pipeline Block Diagram

The A12 image processing pipeline is shown below. Each function block has a corresponding API that can be used to configure the desired image quality settings.

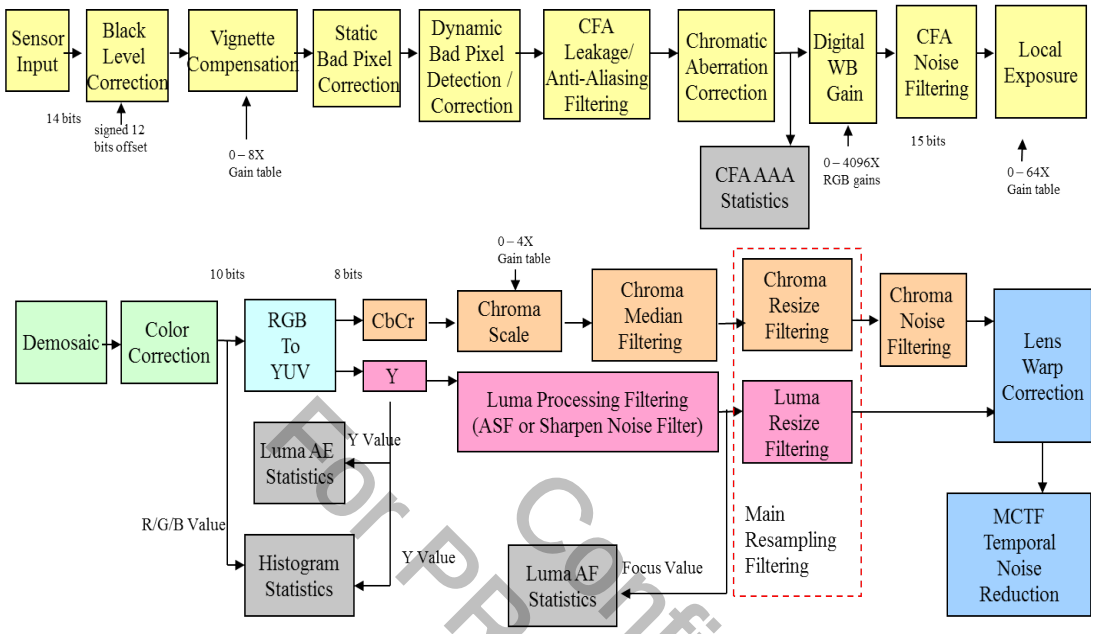

Figure 1-1. Overview: A12 Image Pipeline Diagram.

1.3 Overview: Software Architecture

The overall system architecture is illustrated as follows. The Image Kernel is a component of the DSP Support Package.

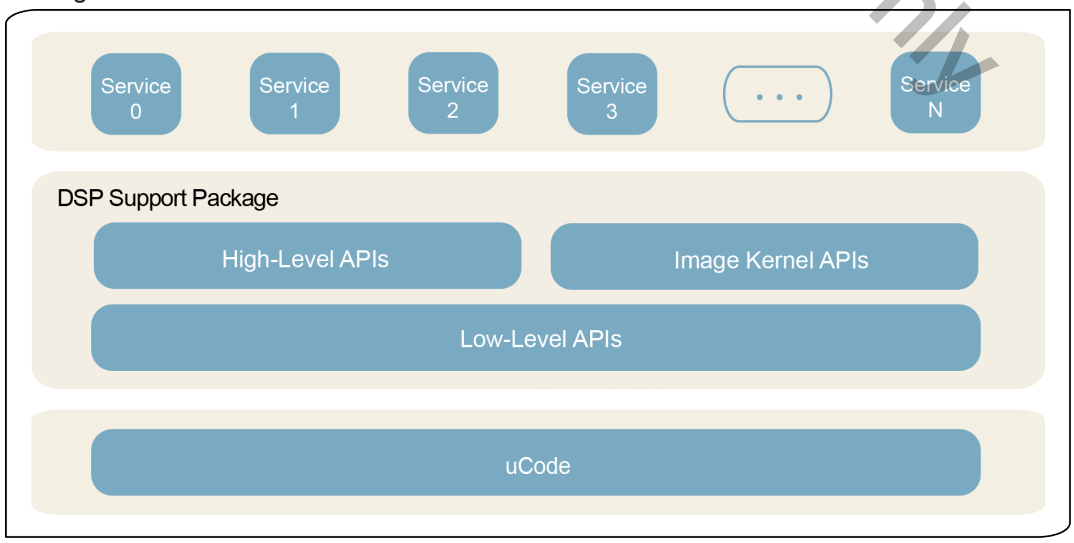

Figure 1-2. Overview: Overall System Architecture.

The Image Kernel architecture is shown in Figure 1-3. Individual pipelines for **Video**, **Still Image**, and **Decode** are designed to support different usage scenarios. Each pipeline includes multiple instance concepts for **context** (`ctx`), **configuration** (`cfg`), and the **batch command buffer** (`batch`). **Context** is the database that stores required settings. **Configuration** stores specific configurations for special algorithms such as Low-ISO or High-ISO. The **batch command buffer** allows the application to group filter settings together for more efficient processing.

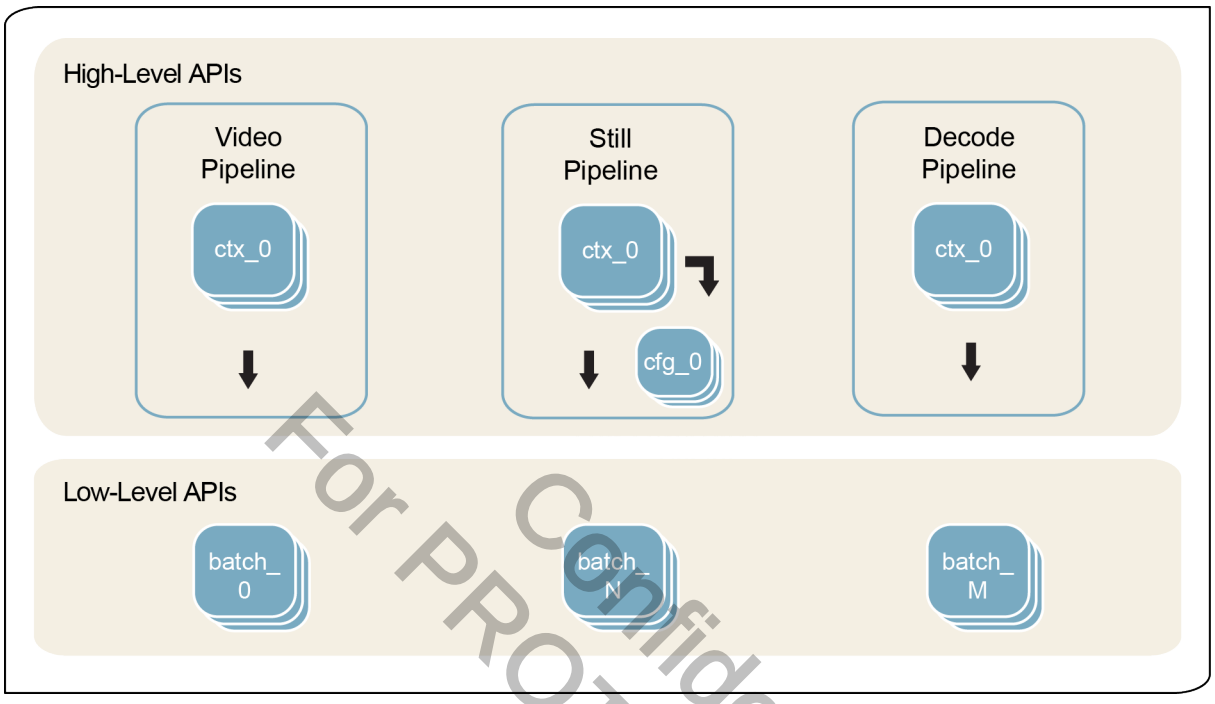

Figure 1-3. Overview: Image Kernel Architecture.

1.4 Overview: Scope of Document

This document focuses strictly on the A12 Image Kernel APIs. Users of this document are assumed to be familiar with the A12 chip hardware, system capabilities, software architecture, and reference applications. The reader is referred to the following for a background overview:

- The chip A12 datasheet provides hardware pin and package details including a feature list with descriptions of chip performance, brief interface descriptions, a complete power-on configuration table and electrical characteristics.
- "*A12 Hardware Programming Reference Manual*" is the primary resource for programming peripheral drivers. It lists software-programmable registers accessible from CPU cores, including detailed information on each field of a register. It also provides overviews of the system memory map, power-on configuration options, and ARM interrupts.
- "*A12 System Hardware*" covers power-on timing and configuration. It provides pin connection details including guidance for unused interfaces and PCB layout.

2 AAA Statistics API

2.1 AAA Statistics: Overview

This chapter introduces the AAA (Auto-exposure, Auto-focus, Auto-white-balance) Statistics APIs.

2.2 AAA Statistics: List of Functions

2.2.1 AmbaDSP_Img3aEnbAaaStat

API Syntax:

AmbaDSP_Img3aEnbAaaStat (AMBA_DSP_IMG_Mode_CFG_s *pMode, AMBA_DSP_IMG_AAA_STAT_ENB_s *pEnbInfo)

Function Description:

- This API is used to enable/disable specified AAA statistics. User will not receive 3a statistics if the statistics feature is not enabled. For the 3a statistic structure, please see Sections 2.2.1.3 and 2.2.1.4 for more details.
-

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 below for more details.
AMBA_DSP_IMG_AAA_STAT_ENB_s	***pEnbInfo**	Address for AAA statistics buffer. Please see Section 2.2.1.2 below for more details.

*Table 2-1. Parameters for AAA Statistics Image Kernel API **AmbaDSP_Img3aEnbAaaStat()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 2-2. Returns for AAA Statistics Image Kernel API **AmbaDSP_Img3aEnbAaaStat()**.*

Example:

None

See Also:

AmbaDSP_Img3aTransferAaaStatData()

2.2.1.1 AmbaDSP_Img3aEnbAaaStat > AMBA_DSP_IMG_MODE_CFG_s

Type	Field	Description
AMBA_DSP_IMG_PIPE_e	**Pipe**	Pipeline. 0: **AMBA_DSP_IMG_PIPE_VIDEO** 1: **AMBA_DSP_IMG_PIPE_STILL** 2: **AMBA_DSP_IMG_PIPE_DEC**
AMBA_DSP_IMG_ALGO_MODE_e	**AlgoMode**	Algorithm mode. 0: **AMBA_DSP_IMG_ALGO_MODE_FAST** 1: **AMBA_DSP_IMG_ALGO_MODE_LISO** 3: **AMBA_DSP_IMG_ALGO_MODE_HISO**
AMBA_DSP_IMG_FUNC_MODE_e	**FuncMode**	Function mode. 0: **AMBA_DSP_IMG_FUNC_MODE_FV** 1: **AMBA_DSP_IMG_FUNC_MODE_QV** 2: **AMBA_DSP_IMG_FUNC_MODE_PIV**
UINT8	**ContextId**	Context **ID**
UINT32	**BatchId**	Batch command buffer **ID**
UINT8	**ConfigId**	Config **ID**

*Table 2-3. Definition of **AMBA_DSP_IMG_MODE_CFG_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aEnbAaaStat()**.*

2.2.1.2 AmbaDSP_Img3aEnbAaaStat > AMBA_DSP_IMG_AAA_STAT_ENB_s

Type	Field	Description
UINT8	**AeAwbEnb**	Enable AE/AWB statistics output: 0: Disable 1: Enable
UINT8	**AfEnb**	Enable AF statistics output: 0: Disable 1: Enable
UINT8	**HisEnb**	Enable histogram statistics output: 0: Disable 1: Enable

*Table 2-4. Definition of **AMBA_DSP_IMG_AAA_STAT_ENB_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aEnbAaaStat()**.*

2.2.1.3 AMBA_DSP_EVENT_RGB_AAA_DATA_s

Type	Field	Description
AMBA_DSP_AAA_HDR_s	**Header**	Header to report real tile size and right-shift bit counts for the statistics. Please refer to Section 2.2.1.5 for more details.
UINT16	**FrameId**	Frame ID
AMBA_DSP_RGB_AF_s	**Af[96]**	AF statistics data for each tile. Please refer to Section 2.2.1.6 for more details

Type	Field	Description
AMBA_DSP_RGB_AE_s	**Ae[96]**	AE statistics data for each tile. Please refer to Section 2.2.1.7 for more details.
AMBA_DSP_RGB_HISTO_s	**Histo**	Histograms for Y, R, G, B. Please refer to Section 2.2.1.8 for more details.

*Table 2-5. Definition of **AMBA_DSP_EVENT_RGB_AAA_DATA_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aEnbAaaStat()**.*

2.2.1.4 AMBA_DSP_EVENT_CFA_AAA_DATA_s

Type	Field	Description
AMBA_DSP_AAA_HDR_s	**Header**	Header to report real tile size and right-shift bit counts for the statistics. Please refer to Section 2.2.1.5 for more details.
UINT16	**FrameId**	Frame ID
AMBA_DSP_CFA_AWB_s	**Awb[1024]**	AWB statistics data for each tile. Please refer to SSection 2.2.1.9 for more details
AMBA_DSP_CFA_AE_s	**Ae[96]**	AF statistics data for each tile. Please refer to Section 2.2.1.10 for more details
AMBA_DSP_CFA_AF_s	**Af[96]**	AF statistics data for each tile. Please refer to Section 2.2.1.11 for more details.
AMBA_DSP_CFA_HISTO_s	**Histo**	Histograms for Y, R, G, B. Please refer to Section 2.2.1.12 for more details.

*Table 2-6. Definition of **AMBA_DSP_EVENT_CFA_AAA_DATA_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aEnbAaaStat()**.*

2.2.1.5 AMBA_DSP_AAA_HDR_s

Type	Field	Description
UINT16	**AwbTileColStart**	The horizontal starting pixel of the first AWB tile. Unit is pixels.
UINT16	**AwbTileRowStart**	The vertical starting pixel of the first AWB tile. Unit is pixels.
UINT16	**AwbTileWidth**	The width of each AWB tile. Unit is pixels.
UINT16	**AwbTileHeight**	The height of each AWB tile. Unit is pixels.
UINT16	**AwbTileActiveWidth**	The active width of each AWB tile. Unit is pixels.
UINT16	**AwbTileActiveHeight**	The active height of each AWB tile. Unit is pixels.
UINT16	**AwbRgbShift**	The right-shift value of AWB RGB statistics. The actual AWB RGB sum value = AWB_RGB_statistics << AwbRgbShift
UINT16	**AwbYShift**	The right-shift value of AWB Y statistics. The actual AWB Y sum value = AWB_Y_statistics << AwbYShift
UINT16	**AwbMinMaxShift**	The right-shift value of AWB count min/max statistics. The actual AWB count min/max number = AWB_count_[min|max]
UINT16	**AeTileColStart**	The horizontal starting pixel of the first AE tile. Unit is pixels.
UINT16	**AeTileRowStart**	The vertical starting pixel of the first AE tile. Unit is pixels.
UINT16	**AeTileWidth**	The width of each AE tile. Unit is pixels.
UINT16	**AeTileHeight**	The height of each AE tile. Unit is pixels.

Type	Field	Description
UINT16	**AeYShift**	The right-shift value of AE YUV Y statistics. The actual AE Y sum value = AE_Y_statistics << AeYShift.
UINT16	**AeLinearYShift**	The right-shift value of AE CFA Y statistics. The actual AE Y sum value = AE_Y_statistics << AeLinearYShift.
UINT16	**AfTileColStart**	The horizontal starting pixel of the first AF tile. Unit is pixels.
UINT16	**AfTileRowStart**	The vertical starting pixel of the first AF tile. Unit is pixels.
UINT16	**AfTileWidth**	The width of each AF tile. Unit is pixels.
UINT16	**AfTileHeight**	The height of each AF tile. Unit is pixels.
UINT16	**AfTileActiveWidth**	The active width of each AF tile. Unit is pixels.
UINT16	**AfTileActiveHeight**	The active height of each AF tile. Unit is pixels.
UINT16	**AfYShift**	The right-shift value of AF YUV Y statistics. The actual AF Y sum value = AF_Y_statistics << AfYShift.
UINT16	**AfCfaYShift**	The right-shift value of AF CFA Y statistics. The actual AF Y sum value = AF_Y_statistics << AfCfaYShift.
UINT8	**AwbTileNumCol**	Typically the tile number of statistics is configured by the user API input, but when the digital zoom ratio is very large the effective pixel number will become very small. A12 will shrink the tile number automatically for better resolution of AAA statistics.
UINT8	**AwbTileNumRow**	
UINT8	**AeTileNumCol**	
UINT8	**AeTileNumRow**	
UINT8	**AfTileNumCol**	
UINT8	**AfTileNumRow**	
UINT8	**TotalSlicesX**	Total slice number in the X direction.
UINT8	**TotalSlicesY**	Total slice number in the Y direction.
UINT8	**SliceIndexX**	Index of slice in the X direction.
UINT8	**SliceIndexY**	Index of slice in the Y direction.
UINT16	**SliceWidth**	The width of slice. Unit is pixels.
UINT16	**SliceHeight**	The height of slice. Unit is pixels.
UINT16	**SliceStartX**	The X-offset of the slice. Unit is pixels.
UINT16	**SliceStartY**	The Y-offset of the slice. Unit is pixels.

*Table 2-7. Definition of **AMBA_DSP_AAA_HDR_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aEnbAaaStat()**.*

2.2.1.6 AMBA_DSP_RGB_AF_s

Type	Field	Description
UINT16	**SumFY**	Sum of luma component regardless of whether the pixel is saturated or right-shifted by **AfYShift**.
UINT16	**SumFV1**	Sum of first gradient measures after **AfTileFv1Shift** rightshift.
UINT16	**SumFV2**	Sum of second gradient measures after **AfTileFv1Shift** rightshift.

*Table 2-8. Definition of **AMBA_DSP_RGB_AF_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aEnbAaaStat()**.*

2.2.1.7 AMBA_DSP_RGB_AE_s

Type	Field	Description
UNIT16	**SumY**	Sums of luma pixels for each AE tile.

*Table 2-9. Definition of **AMBA_DSP_RGB_AE_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aEnbAaaStat()**.*

2.2.1.8 AMBA_DSP_RGB_HISTO_s

Type	Field	Description
UINT32	**HisBinY[64]**	Each histogram consists of 64 bins where each RGB value is right-shifted from full-scale to produce a 6-bit `bin` index and then the corresponding `bin` counter in increment.
UINT32	**HisBinR[64]**	
UINT32	**HisBinG[64]**	
UINT32	**HisBinB[64]**	

*Table 2-10. Definition of **AMBA_DSP_RGB_HISTO_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aEnbAaaStat()**.*

2.2.1.9 AMBA_DSP_CFA_AWB_s

Type	Field	Description
UINT16	**SumR**	Sum of R pixels
UINT16	**SumG**	Sum of G pixels
UINT16	**SumB**	Sum of B pixels
UINT16	**CountMin**	Number of pixels below the minimum threshold
UINT16	**CountMax**	Number of pixels above the maximum threshold

*Table 2-11. Definition of **AMBA_DSP_CFA_AWB_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aEnbAaaStat()**.*

2.2.1.10 AMBA_DSP_CFA_AE_s

Type	Field	Description
UNIT16	**LinY**	Sum of pseudo Y values
UINT16	**CountMin**	Number of pixels below the minimum threshold
UINT16	**CountMax**	Number of pixels above the maximum threshold

*Table 2-12. Definition of **AMBA_DSP_CFA_AE_s** for AAA Statistics Image API **AmbaDSP_Img3aEnbAaaStat()**.*

2.2.1.11 AMBA_DSP_CFA_AF_s

Type	Field	Description
UNIT16	**SumY**	Sum of luma component regardless of whether the pixel is saturated and right-shifted by **AfCfaYShift**.
UINT16	**SumFV1**	Sum of first gradient measures after **AfTileFv1Shift** rightshift.
UINT16	**SumFV2**	Sum of second gradient measures after **AfTileFv2Shift** rightshift.

*Table 2-13. Definition of **AMBA_DSP_CFA_AF_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aEnbAaaStat()**.*

2.2.1.12 AMBA_DSP_CFA_HISTO_s

Type	Field	Description
UINT32	**HisBinR[64]**	Each histogram consists of 64 bins where each RGB value is right-shifted from full scale to produce a 6-bit bin index and then, the corresponding bin counter in increment.
UINT32	**HisBinG[64]**	
UINT32	**HisBinB[64]**	
UINT32	**HisBinY[64]**	

*Table 2-14. Definition of **AMBA_DSP_CFA_HISTO_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aEnbAaaStat()**.*

2.2.2 AmbaDSP_Img3aSetAeStatInfo

API Syntax:

AmbaDSP_Img3aSetAeStatInfo (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_AE_STAT_INFO_s *pAeStat)

Function Description:

- This API is used to define the tile configuration for AE statistics calculation. The parameters are based on a 4096x4096 logical image to specify tiling geometry. The DSP microcode will map this geometry to the actual active image size.
- For A12, two sets of AE statistics are computed based on CFA data or RGB/YUV data respectively. Both sets are computed based on the same set of tile configuration parameters. See Figure 2-1.

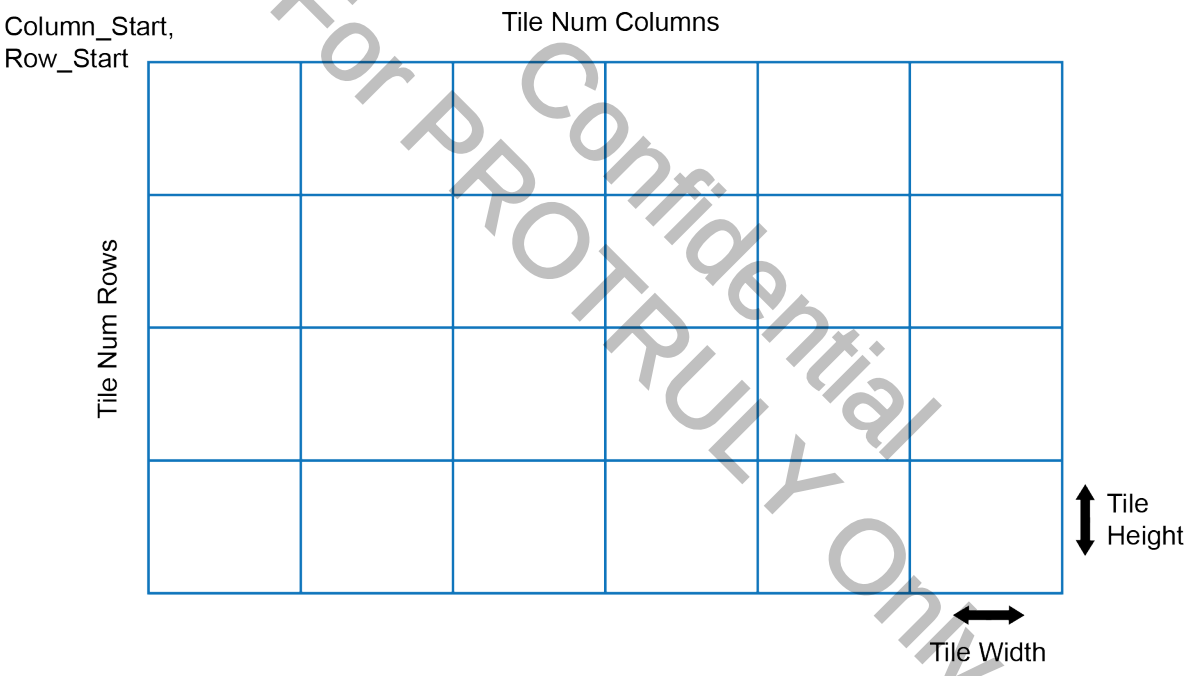

Figure 2-1. Tile Configuration Parameters.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_AE_STAT_INFO_s	***pAeStat**	AE statistics information. Please refer to Section 2.2.2.1 below for details.

*Table 2-15. Parameters for AAA Statistics Image Kernel API **AmbaDSP_Img3aSetAeStatInfo()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 2-16. Returns for AAA Statistics Image Kernel API **AmbaDSP_Img3aSetAeStatInfo()**.*

Example:

None

See Also:

None

2.2.2.1 AmbaDSP_Img3aSetAeStatInfo > AMBA_DSP_IMG_AE_STAT_INFO_s

Type	Field	Description
UINT16	**AeTileNumCol**	1 - 12. Number of tiles horizontally.
UINT16	**AeTileNumRow**	1 - 8. Number of tiles vertically.
UINT16	**AeTileColStart**	Column start for tile configuration
UINT16	**AeTileRowStart**	Row start for tile configuration
UINT16	**AeTileWidth**	1 - 511. Width of each tile.
UINT16	**AeTileHeight**	1 - 511. Height of each tile.
UINT16	**AePixMinValue**	Minimum thresholds for calculating pixel numbers based on CFA data.
UINT16	**AePixMaxValue**	Maximum thresholds for calculating pixel numbers based on CFA data.

*Table 2-17. Definition of **AMBA_DSP_IMG_AE_STAT_INFO_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aSetAeStatInfo()**.*

2.2.3 AmbaDSP_Img3aSetAwbStatInfo

API Syntax:

AmbaDSP_Img3aSetAwbStatInfo (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_AWB_STAT_INFO_s * pAwbStat)

Function Description:

- This API is used to define the tile configuration for AWB statistics calculation. The parameters are based on a 4096x4096 logical image to specify tiling geometry. The DSP microcode will map the geometry to the actual active image size.
- For A12, AWB statistics are computed based on the CFA data. See Figure 2-2.

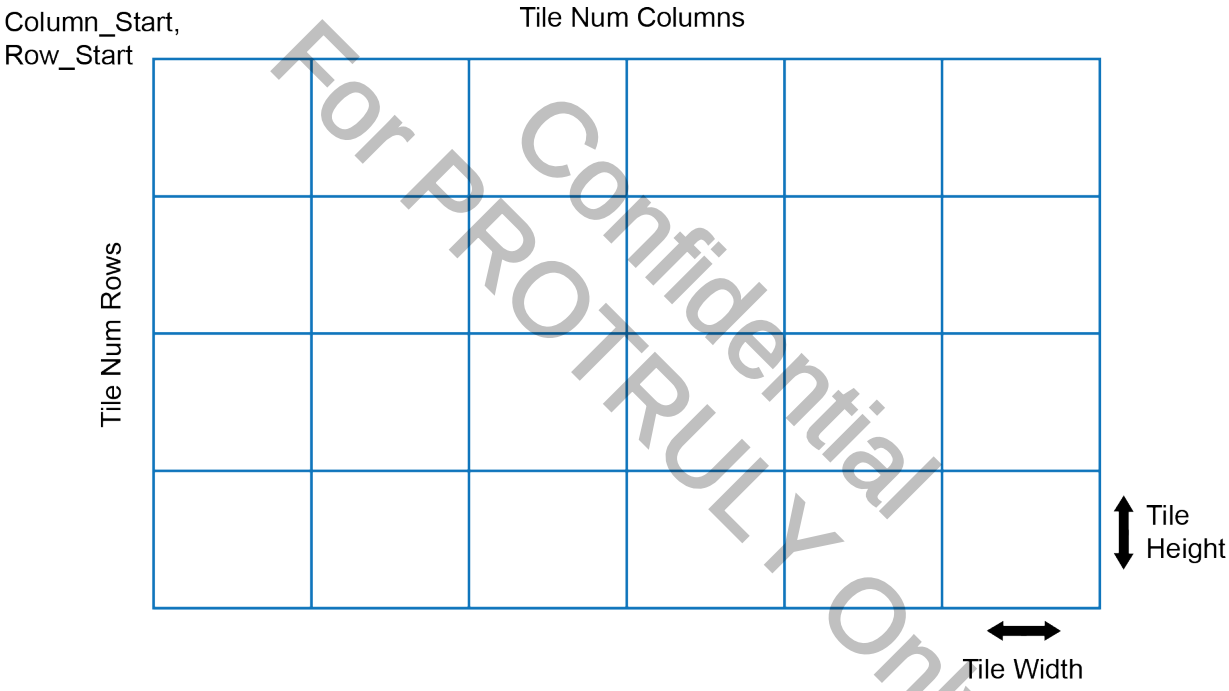

Figure 2-2. Tile Configuration Parameters.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_AWB_STAT_INFO_s	***pAwbStat**	AWB statistics information. Please refer to Section 2.2.3.1 below for more details.

*Table 2-18. Parameters for AAA Statistics Image Kernel API **AmbaDSP_Img3aSetAwbStatInfo()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 2-19. Returns for AAA Statistics Image Kernel API **AmbaDSP_Img3aSetAwbStatInfo()**.*

Example:

None

See Also:

None

2.2.3.1 AmbaDSP_ImgAaaSetAwbStatInfo > AMBA_DSP_IMG_AWB_STAT_INFO_s

Type	Field	Description
UINT16	**AwbTileNumCol**	1 - 32. Number of tiles horizontally.
UINT16	**AwbTileNumRow**	1 - 32. Number of tiles vertically.
UINT16	**AwbTileColStart**	Column start for tile configuration
UINT16	**AwbTileRowStart**	Row start for tile configuration
UINT16	**AwbTileWidth**	1 - 511. Width of each tile.
UINT16	**AwbTileHeight**	1 - 511. Height of each tile.
UINT16	**AwbTileActiveWidth**	1 - 511. Active width and active height define the sub-section of a tile over which AWB statistics are calculated. The active region is cropped starting from the top-left corner of the corresponding tile.
UINT16	**AwbTileActiveHeight**	
UINT32	**AwbPixMinValue**	Minimum thresholds for calculating pixel numbers based on CFA data.
UINT32	**AwbPixMaxValue**	Maximum thresholds for calculating pixel numbers based on CFA data.

*Table 2-20. Definition of **AMBA_DSP_IMG_AWB_STAT_INFO_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aSetAwbStatInfo()**.*

2.2.4 AmbaDSP_Img3aSetAfStatInfo

API Syntax:

AmbaDSP_Img3aSetAfStatInfo (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_AF_STAT_INFO_s * pAfSta)

Function Description:

- This API is used to define the tile configuration for AF statistics calculation. The parameters are based on a 4096x4096 logical image to specify tiling geometry. The DSP microcode will map the geometry to the actual active image size.
- For A12, two sets of AF statistics are computed based on CFA data or RGB/YUV data respectively. Both sets are computed based on the same set of tile configuration parameters. See Figure 2-3.

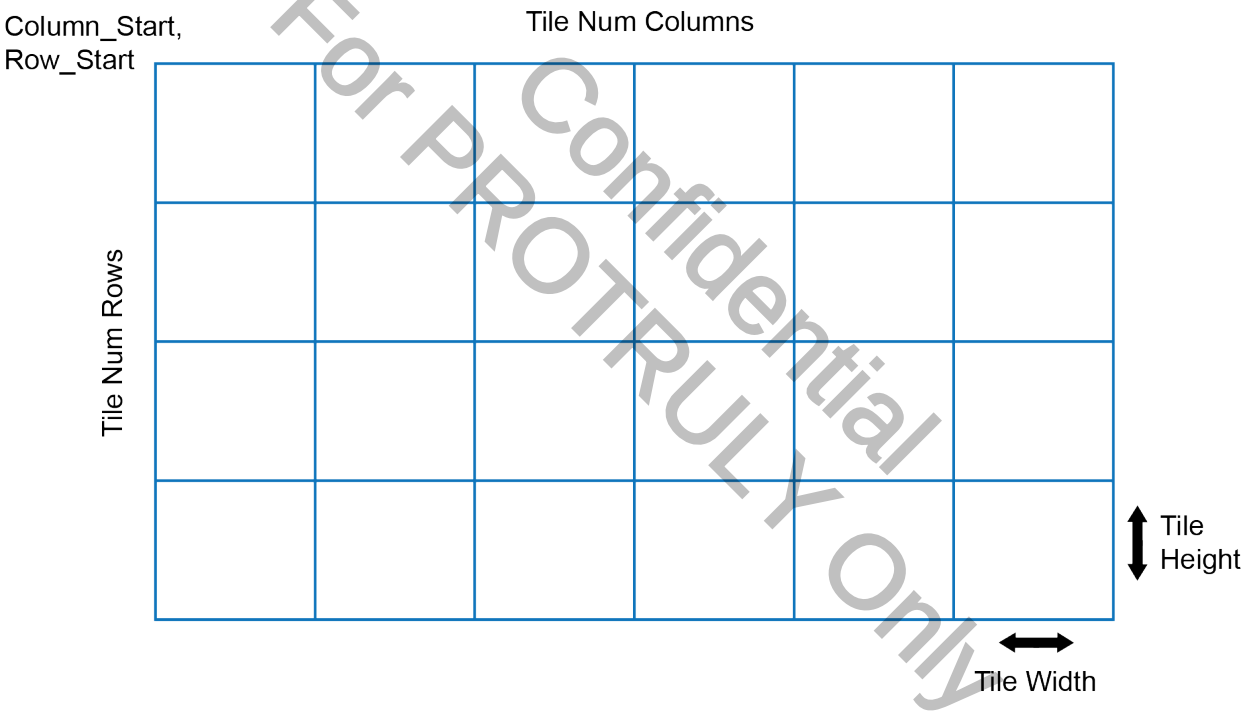

Figure 2-3. Tile Configuration Parameters.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_AF_STAT_INFO_s	***pAfStat**	AF statistics information. See Section 2.2.4.1 below for more details.

*Table 2-21. Parameters for AAA Statistics Image Kernel API **AmbaDSP_Img3aSetAfStatInfo()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success.
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure.
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 2-22. Returns for AAA Statistics Image Kernel API **AmbaDSP_Img3aSetAfStatInfo()**.*

Example:

None

See Also:

None

2.2.4.1 AmbaDSP_Img3aSetAfStatInfo > AMBA_DSP_IMG_AF_STAT_INFO_s

Type	Field	Description
UINT16	**AfTileNumCol**	1 - 16. Number of tiles horizontally.
UINT16	**AfTileNumRow**	1 - 8. Number of tiles vertically.
UINT16	**AfTileColStart**	Column start for tile configuration. Due to a zero initial condition on window boundary to the AF Horizontal IIR filter, a column start value less than 256 is not recommended, as this will cause spikes in the first column.
UINT16	**AfTileRowStart**	Row start for tile configuration
UINT16	**AfTileWidth**	1 - 511 (*1). Width of each tile.
UINT16	**AfTileHeight**	1 - 511 (*1). Height of each tile.
UINT16	**AfTileActiveWidth**	1 - 511 (*1). Active width and active height define the sub-section of a tile over which AWB statistics are calculated. The active region is cropped starting from the top-left corner of the corresponding tile.
UINT16	**AfTileActiveHeight**	

*Table 2-23. Definition of **AMBA_DSP_IMG_AF_STAT_INFO_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aSetAfStatInfo()**.*

2.2.5 AmbaDSP_Img3aSetAfStatExInfo

API Syntax:

AmbaDSP_Img3aSetAfStatExInfo (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_AF_STAT_EX_INFO_s * pAfStatEx)

Function Description:

- This API is used to define the tile configuration for extended AF statistics calculation.
- The horizontal component of the tile focus value is obtained by feeding luma samples line-by-line through an IIR filter and accumulating the absolute value of the filter output over the active region of the AF tile. The overall IIR filter is implemented according to a three-stage cascaded IIR filter structure as follows:

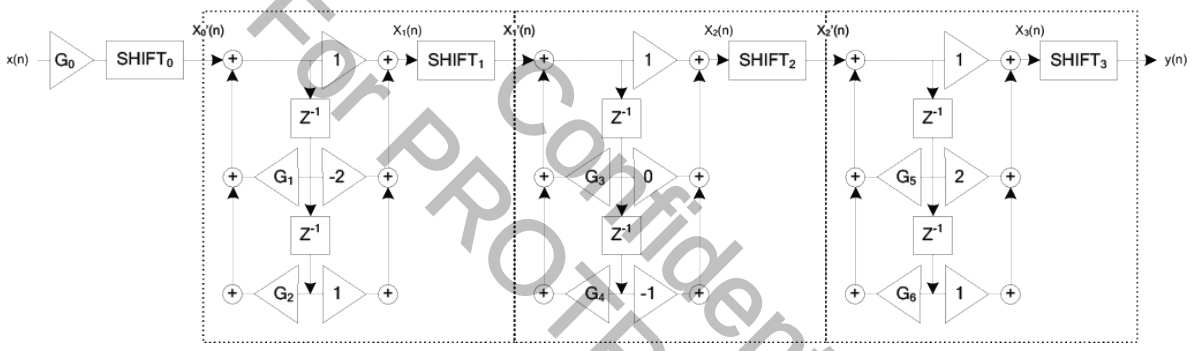

Figure 2-4. Tile Configuration Parameters.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_AF_STAT_EX_INFO_s	***pAfStatEx**	Extended AF statistics information. See Section 2.2.5.1 below for details.

*Table 2-24. Parameters for AAA Statistics Image Kernel API **AmbaDSP_Img3aSetAfStatExInfo()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped

*Table 2-25. Returns for AAA Statistics Image Kernel API **AmbaDSP_Img3aSetAfStatExInfo()**.*

Example:

None

See Also:

None

2.2.5.1 AmbaDSP_ImgAaaSetAfStatExInfo > AMBA_DSP_IMG_AF_STAT_EX_INFO_s

Type	Field	Description
UINT8	**AfHorizontalFilter1Mode**	0: Sum mode 1: Peak mode for FV1cfa, RGB
UINT8	**AfHorizontalFilter1Stage1Enb**	Horizontal Filter 1 Stage 1 enable. Please refer to Figure 2-4.
UINT8	**AfHorizontalFilter1Stage2Enb**	Horizontal Filter 1 Stage 2 enable. Please refer to Figure 2-4.
UINT8	**AfHorizontalFilter1Stage3Enb**	Horizontal Filter 1 Stage 3 enable. Please refer to Figure 2-4.
INT	**AfHorizontalFilter1Gain[7]**	Horizontal Filter 1 Gain values. Please refer to Figure 2-4.
UINT16	**AfHorizontalFilter1Shift[4]**	Horizontal Filter 1 Shift values. Please refer to Figure 2-4.
UINT16	**AfHorizontalFilter1BiasOff**	FV1cfa, RGB horizontal offset for each pixel
UINT16	**AfHorizontalFilter1Thresh**	FV1cfa, RGB horizontal threshold for each pixel
UINT16	**AfVerticalFilter1Thresh**	FV1rgb vertical threshold for each pixel
UINT8	**AfHorizontalFilter2Mode**	0: Sum mode 1: Peak mode for FV2cfa, RGB
UINT8	**AfHorizontalFilter2Stage1Enb**	Horizontal Filter 2 Stage 1 enable. Please refer to Figure 2-4.
UINT8	**AfHorizontalFilter2Stage2Enb**	Horizontal Filter 2 Stage 2 enable. Please refer to Figure 2-4.
UINT8	**AfHorizontalFilter2Stage3Enb**	Horizontal Filter 2 Stage 3 enable. Please refer to Figure 2-4.
INT	**AfHorizontalFilter2Gain[7]**	Horizontal Filter 2 Gain values. Please refer to Figure 2-4.
UINT16	**AfHorizontalFilter2Shift[4]**	Horizontal Filter 2 Shift values. Please refer to Figure 2-4.
UINT16	**AfHorizontalFilter2BiasOff**	FV2cfa, RGB horizontal offset for each pixel
UINT16	**AfHorizontalFilter2Thresh**	FV2cfa, RGB horizontal threshold for each pixel
UINT16	**AfVerticalFilter2Thresh**	FV2rgb vertical threshold for each pixel
UINT16	**AfTileFv1HorizontalShift**	FV1cfa, RGB right-shift horizontal of each tile
UINT16	**AfTileFv1VerticalShift**	FV1rgb right-shift vertical of each tile
UINT16	**AfTileFv1HorizontalWeight**	FV1rgb = AfTileFv1HorizontalWeight*FV1horizontal + AfTileFv1VerticalWeight*FV1vertical
UINT16	**AfTileFv1VerticalWeight**	
UINT16	**AfTileFv2HorizontalShift**	FV2cfa, RGB right-shift horizontal of each tile
UINT16	**AfTileFv2VerticalShift**	FV2,rgb right-shift vertical of each tile
UINT16	**AfTileFv2HorizontalWeight**	FV2rgb= AfTileFv2HorizontalWeight*FV2horizontal + AfTileFv2VerticalWeight*FV2vertical
UINT16	**AfTileFv2VerticalWeight**	

*Table 2-26. Definition of **AMBA_DSP_IMG_AF_STAT_EX_INFO_s** for AAA Statistics Image Kernel API **AmbaDSP_ImgAaaSetAfStatExInfo()**.*

2.2.6 AmbaDSP_Img3aSetAaaStatInfo

API Syntax:

AmbaDSP_Img3aSetAaaStatInfo (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_AAA_STAT_INFO_s * pAaaStat)

Function Description:

- This API is used to define the tile configuration for AWB/AE/AF statistics calculation. The parameters are based on a 4096x4096 logical image to specify tiling geometry. The DSP microcode will map the geometry to the actual active image size. Note that the real active size settings of each AWB/AE/AF tile should not be larger than 512x512; otherwise, overflow could occur in the accumulator. The real tile size can be accessed from the AAA statistics reports.

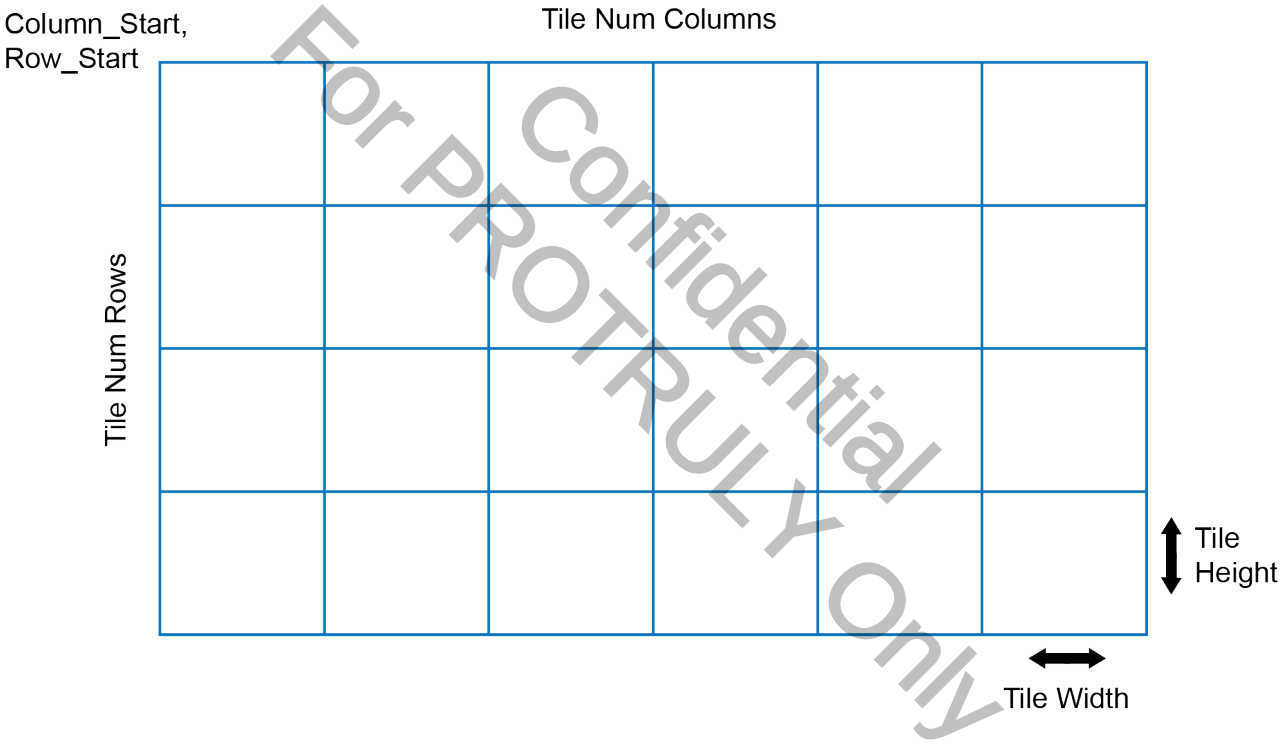

Figure 2-5. Tile Configuration Parameters.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_AAA_STAT_INFO_s	***pAaaStat**	AAA statistics information. See Section 2.2.6.1 below for more details.

*Table 2-27. Parameters for AAA Statistics Image Kernel API **AmbaDSP_Img3aSetAaaStatInfo()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 2-28. Returns for AAA Statistics Image Kernel API **AmbaDSP_Img3aSetAaaStatInfo()**.*

Example

None

See Also:

AmbaDSP_ImgAaaSetAeStatInfo()
AmbaDSP_ImgAaaSetAwbStatInfo()
AmbaDSP_ImgAaaSetAfStatInfo()

2.2.6.1 AmbaDSP_Img3aSetAaaStatInfo > AMBA_DSP_IMG_AAA_STAT_INFO_s

Type	Field	Description
UINT16	**AwbTileNumCol**	Number of tiles horizontally. Maximum: 32
UINT16	**AwbTileNumRow**	Number of tiles vertically. Maximum: 32
UINT16	**AwbTileColStart**	Column start for tile configuration
UINT16	**AwbTileRowStart**	Row start for tile configuration
UINT16	**AwbTileWidth**	Width of each tile. Maximum: 512 (real size)
UINT16	**AwbTileHeight**	Height of each tile Maximum: 512 (real size)
UINT16	**AwbTileActiveWidth**	Active width defines the sub-section of a tile over which AWB statistics are calculated. Maximum: 512 (real size)
UINT16	**AwbTileActiveHeight**	Active height defines the sub-section of a tile over which AWB statistics are calculated. Maximum: 512 (real size)
UINT16	**AwbPixMinValue**	Minimum thresholds for calculating pixel numbers based on CFA data.
UINT16	**AwbPixMaxValue**	Maximum thresholds for calculating pixel numbers based on CFA data.
UINT16	**AeTileNumCol**	Number of tiles horizontally. Maximum: 12
UINT16	**AeTileNumRow**	Number of tiles vertically. Maximum: 8
UINT16	**AeTileColStart**	Column start for tile configuration
UINT16	**AeTileRowStart**	Row start for tile configuration
UINT16	**AeTileWidth**	Width of each tile. Maximum: 512 (real size)
UINT16	**AeTileHeight**	Height of each tile Maximum: 512 (real size)
UINT16	**AePixMinValue**	Minimum thresholds for calculating pixel numbers based on CFA data.
UINT16	**AePixMaxValue**	Maximum thresholds for calculating pixel numbers based on CFA data.
UINT16	**AfTileNumCol**	Number of tiles horizontally. Maximum: 16
UINT16	**AfTileNumRow**	Number of tiles vertically. Maximum: 8

Type	Field	Description
UINT16	**AfTileColStart**	Column start for tile configuration. Due to a zero initial condition on window boundary to the AF Horizontal IIR filter, a column start value less than 256 is not recommended, as this will cause spikes in the first column.
UINT16	**AfTileRowStart**	Row start for tile configuration
UINT16	**AfTileWidth**	Width of each tile. Maximum: 512 (real size)
UINT16	**AfTileHeight**	Height of each tile Maximum: 512 (real size)
UINT16	**AfTileActiveWidth**	Active width defines the sub-section of a tile over which auto-focus statistics are calculated. Maximum: 512 (real size)
UINT16	**AfTileActiveHeight**	Active height defines the sub-section of a tile over which auto-focus statistics are calculated. Maximum: 512 (real size)

*Table 2-29. Definition of **AMBA_DSP_IMG_AAA_STAT_INFO_s** for AAA Statistics Image Kernel API **AmbaDSP_Img3aSetAaaStatInfo()**.*

2.2.7 AmbaDSP_Img3aGetAaaStatInfo

API Syntax:

AmbaDSP_Img3aGetAaaStatInfo (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_AAA_STAT_INFO_s * pAaaStat)

Function Description:

- This API is used to retrieve the tile configuration for AWB/AE/AF statistics calculation.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_AAA_STAT_INFO_s	***pAaaStat**	AAA statistics information. See Section 2.2.6.1 below for more details.

*Table 2-30. Parameters for AAA Statistics Image Kernel API **AmbaDSP_Img3aGetAaaStatInfo()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 2-31. Returns for AAA Statistics Image Kernel API **AmbaDSP_Img3aGetAaaStatInfo()**.*

Example

None

See Also:

AmbaDSP_Img3aSetAeStatInfo
AmbaDSP_Img3aSetAwbStatInfo
AmbaDSP_Img3aSetAfStatInfo
AmbaDSP_Img3aSetAaaStatInfo

3 Common Filter API

3.1 Common Filter: Overview

This chapter introduces the common filter APIs. These APIs are applicable to both video and still ISO as shown in the following table (Table 3-1).

API Name	Video	Still		
	LISO	LISO	3-pass LISO	HISO
AmbaDSP_ImgSetVinSensorInfo	✓	✓	✓	✓
AmbaDSP_ImgGetVinSensorInfo	✓	✓	✓	✓
AmbaDSP_ImgSetStaticBlackLevel	✓	✓	✓	✓
AmbaDSP_ImgGetStaticBlackLevel	✓	✓	✓	✓
AmbaDSP_ImgCalcVignetteCompensation	✓	✓	✓	✓
AmbaDSP_ImgSetVignetteCompensation	✓	✓	✓	✓
AmbaDSP_ImgGetVignetteCompensation	✓	✓	✓	✓
AmbaDSP_ImgSetCfaLeakageFilter	✓	✓	✓	✓
AmbaDSP_ImgGetCfaLeakageFilter	✓	✓	✓	✓
AmbaDSP_ImgSetAntiAliasing	✓	✓	✓	✓
AmbaDSP_ImgGetAntiAliasing	✓	✓	✓	✓
AmbaDSP_ImgSetDynamicBadPixelCorrection	✓	✓	✓	✓
AmbaDSP_ImgGetDynamicBadPixelCorrection	✓	✓	✓	✓
AmbaDSP_ImgSetStaticBadPixelCorrection	✓	✓	✓	✓
AmbaDSP_ImgGetStaticBadPixelCorrection	✓	✓	✓	✓
AmbaDSP_ImgCalcCawarpCompensation	✓	✓	✓	✓
AmbaDSP_ImgSetCawarpCompensation	✓	✓	✓	✓
AmbaDSP_ImgGetCawarpCompensation	✓	✓	✓	✓
AmbaDSP_ImgSetWbGain	✓	✓	✓	✓
AmbaDSP_ImgGetWbGain	✓	✓	✓	✓
AmbaDSP_ImgSetDgainSaturationLevel	✓	✓	✓	✓
AmbaDSP_ImgGetDgainSaturationLevel	✓	✓	✓	✓
AmbaDSP_ImgSetCfaNoiseFilter	✓	✓	✓	✓
AmbaDSP_ImgGetCfaNoiseFilter	✓	✓	✓	✓
AmbaDSP_ImgSetLocalExposure	✓	✓	✓	✓
AmbaDSP_ImgGetLocalExposure	✓	✓	✓	✓
AmbaDSP_ImgSetDeferredBlackLevel	✓	✓	✓	✓
AmbaDSP_ImgGetDeferredBlackLevel	✓	✓	✓	✓
AmbaDSP_ImgSetDemosaic	✓	✓	✓	✓
AmbaDSP_ImgGetDemosaic	✓	✓	✓	✓
AmbaDSP_ImgSetColorCorrectionReg	✓	✓	✓	✓
AmbaDSP_ImgGetColorCorrectionReg	✓	✓	✓	✓
AmbaDSP_ImgSetColorCorrection	✓	✓	✓	✓
AmbaDSP_ImgGetColorCorrection	✓	✓	✓	✓
AmbaDSP_ImgSetToneCurve	✓	✓	✓	✓

API Name	Video	Still		
	LISO	LISO	3-pass LISO	HISO
AmbaDSP_ImgGetToneCurve	✓	✓	✓	✓
AmbaDSP_ImgSetRgbToYuvMatrix	✓	✓	✓	✓
AmbaDSP_ImgGetRgbToYuvMatrix	✓	✓	✓	✓
AmbaDSP_ImgSetChromaScale	✓	✓	✓	✓
AmbaDSP_ImgGetChromaScale	✓	✓	✓	✓
AmbaDSP_ImgSetChromaMedianFilter	✓	✓	✓	✓
AmbaDSP_ImgGetChromaMedianFilter	✓	✓	✓	✓
AmbaDSP_ImgSetColorDependentNoiseReduction	✓	✓	✓	
AmbaDSP_ImgGetColorDependentNoiseReduction	✓	✓	✓	
AmbaDSP_ImgSetLumaProcessingMode	✓	✓	✓	
AmbaDSP_ImgGetLumaProcessingMode	✓	✓	✓	
AmbaDSP_ImgSetAdvanceSpatialFilter	✓	✓	✓	
AmbaDSP_ImgGetAdvanceSpatialFilter	✓	✓	✓	
AmbaDSP_ImgSet1stSharpenNoiseBoth	✓	✓	✓	
AmbaDSP_ImgGet1stSharpenNoiseBoth	✓	✓	✓	
AmbaDSP_ImgSet1stSharpenNoiseNoise	✓	✓	✓	
AmbaDSP_ImgGet1stSharpenNoiseNoise	✓	✓	✓	
AmbaDSP_ImgSet1stSharpenNoiseSharpenFir	✓	✓	✓	
AmbaDSP_ImgGet1stSharpenNoiseSharpenFir	✓	✓	✓	
AmbaDSP_ImgSet1stSharpenNoiseSharpenCoring	✓	✓	✓	
AmbaDSP_ImgGet1stSharpenNoiseSharpenCoring	✓	✓	✓	
AmbaDSP_ImgSet1stSharpenNoiseSharpenCoringIndex-Scale	✓	✓	✓	
AmbaDSP_ImgGet1stSharpenNoiseSharpenCoringIndex-Scale	✓	✓	✓	
AmbaDSP_ImgSet1stSharpenNoiseSharpenMinCorin-gResult	✓	✓	✓	
AmbaDSP_ImgGet1stSharpenNoiseSharpenMinCorin-gResult	✓	✓	✓	
AmbaDSP_ImgSetResamplerCoefAdj	✓			
AmbaDSP_ImgGetResamplerCoefAdj	✓			
AmbaDSP_ImgSetChromaFilter	✓	✓	✓	✓
AmbaDSP_ImgGetChromaFilter	✓	✓	✓	✓
AmbaDSP_ImgSetGbGrMismatch	✓	✓	✓	✓
AmbaDSP_ImgGetGbGrMismatch	✓	✓	✓	✓
AmbaDSP_ImgCalcWarpCompensation	✓	✓	✓	✓
AmbaDSP_ImgSetWarpCompensation	✓	✓	✓	✓
AmbaDSP_ImgGetWarpCompensation	✓	✓	✓	✓
AmbaDSP_ImgSetVideoMctf	✓			
AmbaDSP_ImgGetVideoMctf	✓			
AmbaDSP_ImgSetVideoMctfTemporalAdjust	✓			
AmbaDSP_ImgGetVideoMctfTemporalAdjust	✓			

Table 3-1. Common Filter APIs; Where ✓ Indicates that API Support is Provided

3.2 Common Filter: List of APIs

3.2.1 AmbaDSP_ImgSetVinSensorInfo

API Syntax:

AmbaDSP_ImgSetVinSensorInfo (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SENSOR_INFO_s * pVinSensorInfo)

Function Description:

- This API is used to configure settings for the video input (VIN) sensor.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_SENSOR_INFO_s	***pVinSensorInfo**	VIN sensor information. Please refer to Section 3.2.1.1 below for more details.

*Table 3-2. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetVinSensorInfo()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-3. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetVinSensorInfo()**.*

Example:

None

See Also:

None

3.2.1.1 AmbaDSP_ImgSetVinSensorInfo > AMBA_DSP_IMG_SENSOR_INFO_s

Type	Field	Description
UINT8	**FieldFormat**	This field is intended for use in multi-field readout modes. This is to indicate the line number of the first line readout in field 0. 0 is the first line in field 0 of the readout and represents the top line of the image. 1 is the line below line 0. In progressive readout mode, set this to 0. The rest can be ignored.
UINT8	**Resolution**	Number of bits
UINT8	**Pattern**	0: RG 1: BG 2: GR 3: GB
UINT32	**ReadoutMode**	Sensor readout mode

*Table 3-4. Definition of **AMBA_DSP_IMG_SENSOR_INFO_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetVinSensorInfo()**.*

3.2.2 AmbaDSP_ImgGetVinSensorInfo

API Syntax:

AmbaDSP_ImgGetVinSensorInfo (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SENSOR_INFO_s * pVinSensorInfo)

Function Description:

- This API is used to retrieve video input (VIN) sensor information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_SENSOR_INFO_s	***pVinSensorInfo**	Returned last VIN sensor configuration. Please refer to Section 3.2.1.1 for more details.

*Table 3-5. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetVinSensorInfo()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-6. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetVinSensorInfo()**.*

Example:

None

See Also:

None

3.2.3 AmbaDSP_ImgSetStaticBlackLevel

API Syntax:

AmbaDSP_ImgSetStaticBlackLevel (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_BLACK_CORRECTION_s * pBlackCorr)

Function Description:

- This API is used to set the static black level. The black-level corrections will be added to the raw sensor pixels directly. The result for each pixel is assigned a 15-bit value.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_BLACK_CORRECTION_s	***pBlackCorr**	Static black-level information. Please refer to Section 3.2.3.1 for more details.

*Table 3-7. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetStaticBlackLevel()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-8. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetStaticBlackLevel()**.*

Example:

None

See Also:

None

3.2.3.1 AmbaDSP_ImgSetStaticBlackLevel > AMBA_DSP_IMG_BLACK_CORRECTION_s

Type	Field	Description
INT16	**BlackR**	These signed 12-bit values will be added to Bayer pixels during black-level correction.
INT16	**BlackGr**	
INT16	**BlackGb**	
INT16	**BlackB**	

*Table 3-9. Definition of **AMBA_DSP_IMG_BLACK_CORRECTION_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetStaticBlackLevel()**.*

3.2.4 AmbaDSP_ImgGetStaticBlackLevel

API Syntax:

AmbaDSP_ImgGetStaticBlackLevel (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_BLACK_CORRECTION_s * pBlackCorr)

Function Description:

- This API is used to retrieve the static black-level information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_BLACK_CORRECTION_s	***pBlackCorr**	Static black-level information. Please refer to Section 3.2.3.1 for more details.

*Table 3-10. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetStaticBlackLevel()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-11. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetStaticBlackLevel()**.*

Example:

None

See Also:

None

3.2.5 AmbaDSP_ImgCalcVignetteCompensation

API Syntax:

AmbaDSP_ImgCalcVignetteCompensation (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_VIGNETTE_CALC_INFO_s * pVignetteCaclInfo)

Function Description:

- This API is used to calculate the vignette compensation information from calibration vignette tables. The calibration vignette data contains gain tables (value (1<<GainShift) means 1X gain) for Red, GreenEven, GreenOdd, and Blue channels. Table width and height can be assigned in the input parameters. The Image Kernel will crop the appropriate range in these tables based on the relationship between calibration VIN geometry and current VIN geometry, and resample them according to the hardware format. The result will be stored in the Image Kernel internal context, and it will be issued until API **AmbaDSP_ImgSetVignetteCompensation** is invoked.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_VIGNETTE_CALC_INFO_s	***pVignetteCaclInfo**	Vignette calculation information. Please refer to Section 3.2.5.1 for more details.

*Table 3-12. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgCalcVignetteCompensation()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-13. Returns for Common Filter Image Kernel API **AmbaDSP_ImgCalcVignetteCompensation()**.*

Example:

None

See Also:

AmbaDSP_ImgSetVignetteCompensation()

3.2.5.1 AmbaDSP_ImgCalcVignetteCompensation > AMBA_DSP_IMG_VIGNETTE_CALC_INFO_s

Type	Field	Description
UINT8	**Enb**	0: Disable 1: Enable
UINT8	**GainShift**	7: 3.7 gain format 8: 2.8 gain format 9: 1.9 gain format
UINT8	**VigStrengthEffectMode**	There are two modes for **VigStrength**: 0: Default mode: Gain values of four channels are linearly scaled from unit gain to full gain by **VigStrength**. 1: Keep ratio mode Gain values of four channels are scaled by **VigStrength** but the ratio between every channel gain will be maintained, which helps avoid color shift when the **VigStrength** value is low.
UINT32	**VigStrength**	0 - 65536: Weakest to strongest vignette gain.
AMBA_DSP_IMG_WIN_GEOMETRY_s	**CurrentVinGeo**	Current VIN geometry. Please refer to Section 3.2.5.2 below for more details.
AMBA_DSP_IMG_CALIB_VIGNETTE_INFO_s	**CalibVignetteInfo**	Calibration vignette information. Please refer to Section 3.2.5.3 for more details.

*Table 3-14. Definition of **AMBA_DSP_IMG_VIGNETTE_CALC_INFO_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcVignetteCompensation()**.*

3.2.5.2 AmbaDSP_ImgCalcVignetteCompensation > AMBA_DSP_IMG_WIN_GEOMETRY_s

Type	Field	Description
UINT32	**StartX**	Start position in X direction. Unit is pixels.
UINT32	**StartY**	Start position in Y direction. Unit is pixels.
UINT32	**Width**	Window width. Unit is pixels.
UINT32	**Height**	Window height. Unit is pixels.
AMBA_DSP_IMG_SENSOR_SUBSAMPLING_s	**HSubSample**	Horizontal sample rate. It represents the binning mode of the sensor in horizontal. Please refer to Section 3.2.5.4 below for more details.
AMBA_DSP_IMG_SENSOR_SUBSAMPLING_s	**VSubSample**	Vertical sample rate. It represents the binning mode of the sensor in vertical. Please refer to Section 3.2.5.4 below for more details.

*Table 3-15. Definition of **AMBA_DSP_IMG_WIN_GEOMETRY_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcVignetteCompensation ()**.*

3.2.5.3 AmbaDSP_ImgCalcVignetteCompensation > AMBA_DSP_IMG_CALIB_VIGNETTE_INFO_s

Type	Field	Description
UINT32	**Version**	Vignette calibration table version. Current version number is 0x20130218.
INT32	**TableWidth**	Width of the vignette table
INT32	**TableHeight**	Height of the vignette table
AMBA_DSP_IMG_WIN_GEOMETRY_s	**CalibVinGeo**	The VIN geometry when performing calibration. Please refer to Section 3.2.5.2 for more details.
UINT32	***pVignetteRedGain**	The address of per-channel gain table
UINT32	***pVignetteGreenEvenGain**	The address of per-channel gain table.
UINT32	***pVignetteGreenOddGain**	The address of per-channel gain table.
UINT32	***pVignetteBlueGain**	The address of per-channel gain table.

*Table 3-16. Definition of **AMBA_DSP_IMG_CALIB_VIGNETTE_INFO_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcVignetteCompensation()**.*

3.2.5.4 AmbaDSP_ImgCalcVignetteCompensation > AMBA_DSP_IMG_SENSOR_SUBSAMPLING_s

Type	Field	Description
UINT8	**FactorNum**	Numerator of the sample factor
UINT8	**FactorDen**	Denominator of the sample factor

*Table 3-17. Definition of **AMBA_DSP_IMG_SENSOR_SUBSAMPLING_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcVignetteCompensation()**.*

3.2.6 AmbaDSP_ImgSetVignetteCompensation

API Syntax:

AmbaDSP_ImgSetVignetteCompensation (AMBA_DSP_IMG_MODE_CFG_s *pMode)

Function Description:

- This API is used to set the vignette compensation results computed by the **AmbaDSP_ImgCalcVignetteCompensation** API.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.

*Table 3-18. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetVignetteCompensation()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-19. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetVignetteCompensation()**.*

Example:

None

See Also:

AmbaDSP_ImgCalcVignetteCompensation()

3.2.7 AmbaDSP_ImgGetVignetteCompensation

API Syntax:

AmbaDSP_ImgGetVignetteCompensation (AMBA_DSP_IMG_MODE_CFG_s *pMode, MBA_DSP_IMG_VIGNETTE_CALC_INFO_s * pVignetteCaclInfo)

Function Description:

- This API is used to retrieve the input-calculated vignette compensation parameters of the **AmbaDSP_ImgCalcVignetteCompensation** API.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_VIGNETTE_CALC_INFO_s	***pVignetteCaclInfo**	Returned vignette calculation information. Please refer to Section 3.2.5.1 for more details.

*Table 3-20. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetVignetteCompensation()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-21. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetVignetteCompensation()**.*

Example:

None

See Also:

AmbaDSP_ImgCalcVignetteCompensation()
AmbaDSP_ImgSetVignetteCompensation()

3.2.8 AmbaDSP_ImgSetCfaLeakageFilter

API Syntax:

AmbaDSP_ImgSetCfaLeakageFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CFA_LEAKAGE_FILTER_s * pCfaLeakage)

Function Description:

- This API is used to set the CFA leakage filter. The CFA leakage filter provides model-based red and blue pixel leakage correction for Gr and Gb pixels. The purpose is to prevent a Gr/Gb mismatch from occurring. A linear model is used to correct green pixels that have been corrupted by leakage from neighboring red and blue pixels.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_CFA_LEAKAGE_FILTER_s	***pCfaLeakage**	CFA leakage filter information. Please refer to Section 3.2.8.1 for more details.

*Table 3-22. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetCfaLeakageFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-23. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetCfaLeakageFilter()**.*

Example:

None

See Also:

None

3.2.8.1 AmbaDSP_ImgSetCfaLeakageFilter > AMBA_DSP_IMG_CFA_LEAKAGE_FILTER_s

Type	Field	Description
UINT8	**Enb**	0: Disable 1: Enable
UINT8	**AlphaRr**	GR, corrected = GR + ALPHARR * average(RLEFT, RRIGHT) + ALPHARB * average(BTOP, BBOT). Where ALPHARR and ALPHARB are programmable signed 8-bit values that represent the range [-128/1024, 127/1024] in granularity of 1/1024.
UINT8	**AlphaRb**	
UINT8	**AlphaBr**	GB, corrected = GB + ALPHABB * average(RLEFT, RRIGHT) + ALPHABR * average(BTOP, BBOT). Where ALPHABB and ALPHABR are programmable signed 8-bit values that represent the range [-128/1024, 127/1024] in granularity of 1/1024.
UINT8	**AlphaBb**	
UINT16	**SaturationLevel**	The above equations are applied only for GR and GB pixels that have values less than this saturation level.

*Table 3-24. Definition of **AMBA_DSP_IMG_CFA_LEAKAGE_FILTER_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetCfaLeakageFilter()**.*

3.2.9 AmbaDSP_ImgGetCfaLeakageFilter

API Syntax:

AmbaDSP_ImgGetCfaLeakageFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CFA_LEAKAGE_FILTER_s * pCfaLeakage)

Function Description:

- This API is used to retrieve CFA leakage filter settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_CFA_LEAKAGE_FILTER_s	***pCfaLeakage**	Returned CFA leakage filter information. Please refer to Section 3.2.8.1 for more details.

*Table 3-25. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetCfaLeakageFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-26. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetCfaLeakageFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgSetCfaLeakageFilter()

3.2.10 AmbaDSP_ImgSetAntiAliasing

API Syntax:

AmbaDSP_ImgSetAntiAliasing (AMBA_DSP_IMG_MODE_CFG_s *pMode, UINT8 Strength)

Function Description:

- This API is used to enable or disable the anti-aliasing filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
UINT8	**Strength**	0: Disable 1 - 3: Weakest to strongest filtering

*Table 3-27. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetAntiAliasing()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-28. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetAntiAliasing()**.*

Example:

None

See Also:

None

3.2.11 AmbaDSP_ImgGetAntiAliasing

API Syntax:

AmbaDSP_ImgGetAntiAliasing (AMBA_DSP_IMG_MODE_CFG_s *pMode, UINT8 Strength)

Function Description:

- This API is used to retrieve anti-aliasing filter settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
UINT8	**Strength**	Returned anti aliasing settings. 0: Disable 1 - 3: Weakest to strongest filtering

*Table 3-29. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetAntiAliasing()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-30. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetAntiAliasing()**.*

Example:

None

See Also:

None

3.2.12 AmbaDSP_ImgSetDynamicBadPixelCorrection

API Syntax:

AmbaDSP_ImgSetDynamicBadPixelCorrection (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_DBP_CORRECTION_s * pDbpCorr)

Function Description:

- This API is used to set values for dynamic bad pixel correction. Dynamic bad pixel detection is aimed at finding pixels that grossly deviate from their eight closest same-color neighboring pixels in the current picture.
- There are two detection modes: first-order detection and second-order detection. First-order detection locates isolated bad pixels while second-order detection is a more aggressive mode which can detect clusters of up to two bad pixels. Please note that second-order detection has a higher potential for false detection.
- Dynamic bad pixel detection is less reliable than static bad pixel detection, but it does not require calibration and is useful for dealing with temperature- and gain-dependent hot pixels in particular. Bad pixels are corrected by clamping to the nearest same-color neighbors.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_DBP_CORRECTION_s	***pDbpCorr**	Dynamic bad pixel correction information. Please refer to Section 3.2.12.1 for more details.

*Table 3-31. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetDynamicBadPixelCorrection()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-32. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetDynamicBadPixelCorrection()**.*

Example

None

See Also:

None

3.2.12.1 AmbaDSP_ImgSetDynamicBadPixelCorrection > AMBA_DSP_IMG_DBP_CORRECTION_s

Type	Field	Description
UINT8	**Enb**	0: Disable 1: Hot first-order, dark second-order 2: Hot second-order, dark first-order 3: Hot second-order, dark second-order 4: Hot first-order, dark first-order
UINT8	**HotPixelStrength**	Hot pixel correction strength, 0-10.
UINT8	**DarkPixelStrength**	Dark pixel correction strength 0-10.
UINT8	**CorrectionMethod**	0: Normal correction mode 1: Aggressive correction mode: It may be useful in high-noise situations.

*Table 3-33. Definition of **AMBA_DSP_IMG_DBP_CORRECTION_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetDynamicBadPixelCorrection()**.*

3.2.13 AmbaDSP_ImgGetDynamicBadPixelCorrection

API Syntax:

AmbaDSP_ImgGetDynamicBadPixelCorrection (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_DBP_CORRECTION_s * pDbpCorr)

Function Description:

- This API is used to retrieve dynamic bad pixel correction settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_DBP_CORRECTION_s	***pDbpCorr**	Dynamic bad pixel correction information. Please refer to Section 3.2.12.1 for more details.

*Table 3-34. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetDynamicBadPixelCorrection()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-35. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetDynamicBadPixelCorrection()**.*

Example

None

See Also:

AmbaDSP_ImgSetDynamicBadPixelCorrection()

3.2.14 AmbaDSP_ImgSetStaticBadPixelCorrection

API Syntax:

AmbaDSP_ImgSetStaticBadPixelCorrection (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SBP_CORRECTION_s * pSbpCorr)

Function Description:

- This function is used to mark static bad pixels (SBPs), which are then replaced by a combination of neighboring good pixels of the same color. The bad pixel map can be generated by Ambarella's proprietary calibration tool, and should be stored in non-volatile memory.
- Please take note of the following:
  1. The memory pointed by **SbpBuffer** must be allocated by the user and cannot be released. The Image DSP library will not duplicate the memory pointed to by this variable.
  2. The calibration information matches current sensor binning/scaling modes, and this API will crop the appropriate area based on the relationship between current VIN geometry and calibration geometry.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_SBP_CORRECTION_s	***pSbpCorr**	Static bad pixel correction information. Please refer to Section 3.2.14.1 for more details.

*Table 3-36. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetStaticBadPixelCorrection()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-37. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetStaticBadPixelCorrection()**.*

Example:

None

See Also:

None

3.2.14.1 AmbaDSP_ImgSetStaticBadPixelCorrection > AMBA_DSP_IMG_SBP_CORRECTION_s

Type	Field	Description
UINT8	**Enb**	0: Disable SBP correction. 1: Enable SBP correction.
UINT8	**Reserved[3]**	Reserved for alignment.
AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s	**CurrentVinSensorGeo**	Current VIN sensor geometry. Image Kernel APIs will calculate the offset of the SBP correction map address based on the relationship between current VIN sensor geometry and calibration geometry. Please refer to Section 3.2.14.2 for more details.
AMBA_DSP_IMG_CALIB_SBP_INFO_s	**CalibSbpInfo**	Calibration SBP information. This is where the physical address of calibrated SBP map and sensor geometry information is stored. Please refer to Section 3.2.14.3 for more details.

*Table 3-38. Definition of **AMBA_DSP_IMG_SBP_CORRECTION_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetStaticBadPixelCorrection()**.*

3.2.14.2 AmbaDSP_ImgSetStaticBadPixelCorrection > AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s

Type	Field	Description
UINT32	**StartX**	Offset in X direction. Unit is pixels.
UINT32	**StartY**	Offset in Y direction. Unit is pixels.
UINT32	**Width**	Width. Unit is pixels.
UINT32	**Height**	Height. Unit is pixels.
AMBA_DSP_IMG_SENSOR_SUBSAMPLING_s	**HSubSample**	Horizontal sample rate. It represents the binning mode of the sensor in horizontal. Please refer to Section 3.2.5.4 for more details.
AMBA_DSP_IMG_SENSOR_SUBSAMPLING_s	**VSubSample**	Vertical sample rate. It represents the binning mode of the sensor in vertical. Please refer to Section 3.2.5.4 for more details.

*Table 3-39. Definition of **AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetStaticBadPixelCorrection()**.*

3.2.14.3 AmbaDSP_ImgSetStaticBadPixelCorrection > AMBA_DSP_IMG_CALIB_SBP_INFO_s

Type	Field	Description
UINT32	**Version**	SBP correction information version. Current version number is 0x20130218.
UINT8	***SbpBuffer**	The physical address of the SBP correction map.
AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s	**VinSensorGeo**	Calibration VIN sensor geometry. Image Kernel APIs will calculate the offset of the SBP correction map address based on the relationship between current VIN sensor geometry and calibration geometry. Please refer to Section 3.2.14.2 above for more details.
UINT32	**Reserved**	Reserved for alignment.
UINT32	**Reserved1**	Reserved for alignment.

*Table 3-40. Definition of **AMBA_DSP_IMG_CALIB_SBP_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetStaticBadPixelCorrection()**.*

3.2.15 AmbaDSP_ImgGetStaticBadPixelCorrection

API Syntax:

AmbaDSP_ImgGetStaticBadPixelCorrection (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SBP_CORRECTION_s * pSbpCorr)

Function Description:

- This function is used to retrieve bad pixel correction settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_SBP_CORRECTION_s	***pSbpCorr**	Static bad pixel correction information. Please refer to Section 3.2.14.1 for more details.

*Table 3-41. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetStaticBadPixelCorrection()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-42. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetStaticBadPixelCorrection()**.*

Example:

None

See Also:

AmbaDSP_ImgSetStaticBadPixelCorrection()

3.2.16 AmbaDSP_ImgCalcCawarpCompensation

API Syntax:

AmbaDSP_ImgCalcCawarpCompensation (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CAWARP_CALC_INFO_s * pCaCalcInfo)

Function Description:

- This API is used to calculate the **chromatic aberration warp** (**cawarp**) compensation information from calibration cawarp tables. The calibration cawarp data contain a cawarp compensation table array (Each vector is sign 4.5 format), with blue and red scale factors. The Image Kernel will crop the appropriate range of calibration tables based on the relationship between calibration VIN geometry and current VIN geometry, and resample them to hardware format. The result will be stored in image-kernel internal context, and it will be issued until API **AmbaDSP_ImgSetCawarpCompensation** is invoked.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_CAWARP_CALC_INFO_s	***pCaCalcInfo**	Cawarp calculation information. Please refer to Section 3.2.16.1 for more details.

*Table 3-43. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgCalcCawarpCompensation()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success.
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure.
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-44. Returns for Common Filter Image Kernel API **AmbaDSP_ImgCalcCawarpCompensation()**.*

Example:

None

See Also:

AmbaDSP_ImgSetCawarpCompensation()

3.2.16.1 AmbaDSP_ImgCalcCawarpCompensation > AMBA_DSP_IMG_CAWARP_CALC_INFO_s

Type	Field	Description
UINT8	**CaWarpEnb**	0: Disable cawarp. 1: Enable cawarp
UINT8	**Reserved**	Reserved for alignment.
UINT16	**Reserved1**	Reserved for alignment.
AMBA_DSP_IMG_CALIB_CAWARP_INFO_s	**CalibCaWarpInfo**	Calibration cawarp information. This is where the calibrated cawarp table grid numbers, tile size, and sensor geometry information are stored. Please refer to Section 3.2.16.2 for more details.
AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s	**VinSensorGeo**	Current VIN sensor geometry. Image Kernel APIs will calculate the warp table based on the relationship between current VIN sensor geometry and calibration geometry. Please refer to Section 3.2.16.3 for more details.
AMBA_DSP_IMG_WIN_DIMENSION_s	**R2rOutWinDim**	If raw-to-raw scaling is enabled, user is required to assign raw-to-raw output dimension. Unit is pixels. Please refer to Section 3.2.16.4 for more details.
AMBA_DSP_IMG_WIN_GEOMETRY_s	**DmyWinGeo**	Dummy window geometry. Unit is pixels. StartX, StartY, and Width, and Height must be even-numbered values. The margins between dummy window and actual window are for EIS rolling-shutter correction usage. Please refer to Section 3.2.16.5 for more details.
AMBA_DSP_IMG_WIN_DIMENSION_s	**CfaWinDim**	CFA prescaler output window dimension. Unit is pixels. Width and Height must be even-numbered values. Please refer to Section 3.2.16.4 for more details.

*Table 3-45. Definition of **AMBA_DSP_IMG_CAWARP_CALC_INFO_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcCawarpCompensation()**.*

3.2.16.2 AmbaDSP_ImgCalcCawarpCompensation > AMBA_DSP_IMG_CALIB_CAWARP_INFO_s

Type	Field	Description
UINT32	**Version**	Cawarp calibration info version. Current version number is 0x20130125.
INT32	**HorGridNum**	Horizontal grid number
INT32	**VerGridNum**	Vertical grid number
INT32	**TileWidthExp**	Tile width exponent. Tile width is 2^TileWidthExp. Note that (HorGridNum-1) << TileWidthExp must be larger or equal to VinSensorGeo.Width.
INT32	**TileHeightExp**	Tile height exponent. Tile height is 2^TileHeightExp. Note that (VerGridNum-1) << TileHeightExp must be larger or equal to VinSensorGeo.Height.
AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s	**VinSensorGeo**	VIN sensor geometry when calibrating. Please refer to Section 3.2.16.3 for more details.
UINT32	**RedScaleFactor**	Red scale factor. Red channel uses (RedScaleFactor * CawarpTableVector) / 256.
UINT32	**BlueScaleFactor**	Blue scale factor. Blue channel uses (BlueScaleFactor * CawarpTableVector) / 256.
UINT32	**Reserved1**	Reserved bytes for extension.
AMBA_DSP_IMG_GRID_POINT_s	***pCaWarp**	Calibration cawarp vectors pointer. It should be pointed to a memory with size (HorGridNum*VerGridNum*2 Bytes), and each vector is with sign 4.5 format. Please refer to Section 3.2.16.6 for more details.

*Table 3-46. Definition of **AMBA_DSP_IMG_CALIB_CAWARP_INFO_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcCawarpCompensation()**.*

3.2.16.3 AmbaDSP_ImgCalcCawarpCompensation > AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s

Type	Field	Description
UINT32	**StartX**	Offset in X direction. Unit is pixels.
UINT32	**StartY**	Offset in Y direction. Unit is pixels.
UINT32	**Width**	Width. Unit is pixels.
UINT32	**Height**	Height. Unit is pixels.
AMBA_DSP_IMG_SENSOR_SUBSAMPLING_s	**HSubSample**	Horizontal sample rate. It represents the binning mode of the sensor in horizontal. Please refer to Section 3.2.5.4 for more details.
AMBA_DSP_IMG_SENSOR_SUBSAMPLING_s	**VSubSample**	Vertical sample rate. It represents the binning mode of the sensor in vertical. Please refer to Section 3.2.5.4 for more details.

*Table 3-47. Definition of **AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcCawarpCompensation()**.*

3.2.16.4 AmbaDSP_ImgCalcCawarpCompensation > AMBA_DSP_IMG_WIN_DIMENSION_s

Type	Field	Description
UINT32	**Width**	Width. Unit is pixels.
UINT32	**Height**	Height. Unit is pixels.

*Table 3-48. Definition of **AMBA_DSP_IMG_WIN_DIMENSION_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcCawarpCompensation()**.*

3.2.16.5 AmbaDSP_ImgCalcCawarpCompensation > AMBA_DSP_IMG_WIN_GEOMETRY_s

Type	Field	Description
UINT32	**StartX**	Offset in X direction. Unit is pixels.
UINT32	**StartY**	Offset in Y direction. Unit is pixels.
UINT32	**Width**	Width. Unit is pixels.
UINT32	**Height**	Height. Unit is pixels.

*Table 3-49. Definition of **AMBA_DSP_IMG_WIN_GEOMETRY_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcCawarpCompensation()**.*

3.2.16.6 AmbaDSP_ImgCalcCawarpCompensation > AMBA_DSP_IMG_GRID_POINT_s

Type	Field	Description
INT16	**X**	Horizontal vector
INT16	**Y**	Vertical vector

*Table 3-50. Definition of **AMBA_DSP_IMG_GRID_POINT_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcCawarpCompensation()**.*

3.2.17 AmbaDSP_ImgSetCawarpCompensation

API Syntax:

AmbaDSP_ImgSetCawarpCompensation (AMBA_DSP_IMG_MODE_CFG_s *pMode)

Function Description:

- This API is used to issue the chromatic aberration warp compensation results computed by the **AmbaDSP_ImgCalcCawarpCompensation** API.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.

*Table 3-51. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetCawarpCompensation()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-52. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetCawarpCompensation()**.*

Example:

None

See Also:

AmbaDSP_ImgCalcCawarpCompensation()

3.2.18 AmbaDSP_ImgGetCawarpCompensation

API Syntax:

AmbaDSP_ImgGetCawarpCompensation (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CAWARP_CALC_INFO_s * pCaCalcInfo)

Function Description:

- This API is used to retrieve the chromatic aberration warp (cawarp) compensation information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_CAWARP_CALC_INFO_s	***pCaGetInfo**	Get cawarp compensation information. Please refer to Section 3.2.16.1 for more details.

*Table 3-53. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetCawarpCompensation()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-54. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetCawarpCompensation()**.*

Example:

None

See Also:

AmbaDSP_ImgCalcCawarpCompensation()
AmbaDSP_ImgSetCawarpCompensation()

3.2.19 AmbaDSP_ImgSetWbGain

API Syntax:

AmbaDSP_ImgSetWbGain (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_WB_GAIN_s * pWbGains)

Function Description:

- This API is used to set White Balance (WB) gains for various preset modes.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_WB_GAIN_s	***pWbGains**	WB gain information. Please refer to Section 3.2.19.1 for more details.

*Table 3-55. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetWbGain()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-56. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetWbGain()**.*

Example:

None

See Also:

None

3.2.19.1 AmbaDSP_ImgSetWbGain > AMBA_DSP_IMG_WB_GAIN_s

Type	Field	Description
UINT32	**GainR**	These three color gains are expressed in fixed-point representation and the actual values are obtained by (gain/ 4096).
UINT32	**GainG**	
UINT32	**GainB**	

*Table 3-57. Definition of **AMBA_DSP_IMG_WB_GAIN_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetWbGain()**.*

3.2.20 AmbaDSP_ImgGetWbGain

API Syntax:

AmbaDSP_ImgGetWbGain (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_WB_GAIN_s * pWbGains)

Function Description:

- This API is used to retrieve White Balance (WB) gain information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_WB_GAIN_s	***pWbGains**	WB gain information. Please refer to Section 3.2.19.1 for more details.

*Table 3-58. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetWbGain()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-59. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetWbGain()**.*

Example:

None

See Also:

AmbaDSP_ImgSetWbGain()

3.2.21 AmbaDSP_ImgSetDgainSaturationLevel

API Syntax:

AmbaDSP_ImgSetDgainSaturationLevel (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_DGAIN_SATURATION_s * pDgainSat)

Function Description:

- This API is used to set the digital gain saturation level. It is used to define the maximum pixel value allowed after digital gain multiplication. Those multiplied values will be clipped to be no greater than this saturation value.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_DGAIN_SATURATION_s	***pDgainSat**	Digital gain saturation information. Please refer to Section 3.2.21.1 for more details.

*Table 3-60. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetDgainSaturationLevel()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-61. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetDgainSaturationLevel()**.*

Example:

None

See Also:

None

3.2.21.1 AmbaDSP_ImgSetDgainSaturationLevel > AMBA_DSP_IMG_DGAIN_SATURATION_s

Type	Field	Description
UINT32	**LevelRed**	Saturation level of different color channels. Valid range is 15 bits (0-32767). If ((raw_pixel_value * dgain) > **Level**), the output of pixel value after dgain stage will be clamped to **Level**.
UINT32	**LevelGreenEven**	
UINT32	**LevelGreenOdd**	
UINT32	**LevelGreenBlue**	

*Table 3-62. Definition of **AMBA_DSP_IMG_DGAIN_SATURATION_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetDgainSaturationLevel()**.*

3.2.22 AmbaDSP_ImgGetDgainSaturationLevel

API Syntax:

AmbaDSP_ImgGetDgainSaturationLevel (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_DGAIN_SATURATION_s * pDgainSat)

Function Description:

- This API is used to retrieve digital gain saturation level information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_DGAIN_SATURATION_s	***pDgainSat**	Returned digital gain saturation information. Please refer to Section 3.2.21.1 for details.

*Table 3-63. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetDgainSaturationLevel()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-64. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetDgainSaturationLevel()**.*

Example:

None

See Also:

AmbaDSP_ImgSetDgainSaturationLevel()

3.2.23 AmbaDSP_ImgSetCfaNoiseFilter

API Syntax:

AmbaDSP_ImgSetCfaNoiseFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CFA_NOISE_FILTER_s * pCfaNoise)

Function Description:

- This API is used to set the CFA domain noise filter, an edge-preserving non-linear filter that improves demosaic performance under noisy conditions. The CFA noise filter processes each Bayer color component separately. The goal is to reduce noise without causing excessive blurring or smearing. Each pixel is classified as normal or fine-detail. A fine-detail pixel differs in value from most pixels in its support region. Different filter settings are applied for normal and fine-detail pixels.
- **NoiseLevel** is set to the average noise level expected for the 14-bit sensor input (which is a function of the sensor settings). This value may be experimentally determined at each ISO level by measuring the noise standard-deviation. Internally, the red and blue noise levels are adjusted based on white-balance gains, so the application does not need to change these values based on current white-balance parameters.
- **ExtentRegular** determines the size of the filter support region. Higher values indicate wider support. This parameter is in effect when the pixel is classified as normal. A wider support results in more filtering, while a smaller support retains more details.
- **Extent_Fine** determines the size of the filter support region. Higher values indicate wider support. This parameter is in effect when the pixel is classified as fine-detail.
- **Strength_Fine** determines the filtering strength to be used for fine-detail pixels. 0 indicates the same strength as regular pixels, while 256 indicates the maximum strength. A higher strength tends to reduce fine detail, but also removes impulse, or salt-pepper, type noise. It is recommended that a high **strength_fine** be used together with a smaller **extent_fine** to reduce impulse noise without causing excessive blurring.
- **OriginalBlendStrength** determines the weight of the center (original) pixel. Higher values increase the weight of the center pixel.
- **Selectivity** determines the weight of the pixels closer to the center in support region. Higher value increases the weight of the pixels close to the center.

The filtered output pixel is a weighted sum of pixels that fall within the threshold (NoiseLevel), and center pixel by definition always falls within the threshold. If you set original blend strength (and selectivity) to zero, then every pixel in the neighborhood has equal weight.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_CFA_NOISE_FILTER_s	***pCfaNoise**	CFA noise filter information. Please refer to Section 3.2.23.1 for more details.

*Table 3-65. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetCfaNoiseFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-66. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetCfaNoiseFilter()**.*

Example:

None

See Also:

None

3.2.23.1 AmbaDSP_ImgSetCfaNoiseFilter > AMBA_DSP_IMG_CFA_NOISE_FILTER_s

Type	Field	Description
UINT8	**Enb**	0: Disable 1: Enable
UINT16	**NoiseLevel[3]**	R, G, B channel maps to array index of 0, 1, 2 respectively. The range is 0 - 8192.
UINT16	**OriginalBlendStr[3]**	R, G, B channel maps to array index of 0, 1, 2 respectively. The range is 0 - 256.
UINT16	**ExtentRegular[3]**	R, G, B channel maps to array index of 0, 1, 2 respectively. The range is 0 - 256.
UINT16	**ExtentFine[3]**	R, G, B channel maps to array index of 0, 1, 2 respectively. The range is 0 - 256.
UINT16	**StrengthFine[3]**	R, G, B channel maps to array index of 0, 1, 2 respectively. The range is 0 - 256.
UINT16	**SelectivityRegular**	The value could be 0, 50, 100, 150, 200, 250.
UINT16	**SelectivityFine**	The value could be 0, 50, 100, 150, 200, 250.

*Table 3-67. Definition of **AMBA_DSP_IMG_CFA_NOISE_FILTER_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetCfaNoiseFilter()**.*

3.2.24 AmbaDSP_ImgGetCfaNoiseFilter

API Syntax:

AmbaDSP_ImgGetCfaNoiseFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CFA_NOISE_FILTER_s * pCfaNoise)

Function Description:

- This API is used to retrieve the CFA domain noise filter settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_CFA_NOISE_FILTER_s	***pCfaNoise**	CFA noise filter information. Please refer to Section 3.2.23.1 for more details.

*Table 3-68. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetCfaNoiseFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-69. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetCfaNoiseFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgSetCfaNoiseFilter()

3.2.25 AmbaDSP_ImgSetLocalExposure

API Syntax:

AmbaDSP_ImgSetLocalExposure (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LOCAL_EXPOSURE_s * pLocalExposure)

Function Description:

- This API is used to set local exposure. The local exposure stage dynamically adjusts the gain of the current pixel based on the average luma value around the current pixel. This can be used to improve the contrast in shadow regions by increasing local gain. The radius controls the neighboring tile size. As this operation is performed in the CFA domain, luma value is calculated as a weighted sum of R, G and B values, and the weights are specified as 1.4-bit fixed-point numbers.
- The radius index has legal range from 0 to 6. The mapping of this radius index to the actual square pixel region used for luma accumulation is as follows:
  - Index 0 → 4x4 pixels
  - index 1 → 6x6 pixels
  - index 2 → 8x8 pixels
  - index 3 → 10x10 pixels
  - index 4 → 12x12 pixels
  - index 5 → 14x14 pixels
  - index 6 → 16x16 pixels
- Note that the region always contains an integral number of 2x2 Bayer quads.
- The **LumaWeightShift** is used to scale-down the luma summation in order to clamp the average luma value to an 8-bit index.
- The gain curve takes this index value as an input, and maps it to a 2.10-bit multiplier output (i.e, 1024 corresponds to unity gain). The shift value should be computed so that the desired shadow range after the summation falls within a limit of 256. The last value in the table should be 1024, because mid-tone and highlight values will be clamped and will use the last index for local gain.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_LOCAL_EXPOSURE_s	***pLocalExposure**	Local exposure information. Please refer to Section 3.2.25.1 for more details.

*Table 3-70. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetLocalExposure()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-71. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetLocalExposure()**.*

Example:

The local exposure gain is a function of the **average luma level** over the region of pixels. The range of this **average luma level** is 15 bits in the underlying hardware. Thus, for the choice of region size, the weighted summation of the R, G, and B pixels over that region should undergo an appropriate **LOCAL_EXPOSURE_LUMA_SUM_SHIFT** that will utilize the 15-bit range.

The following is an example:

LocalExposure.Radius	4
LocalExposure.LumaWeightRed	28
LocalExposure.LumaWeightGreen	28
LocalExposure.LumaWeightBlue	28
LocalExpousre.LumaWeightShift	7

In this example, a radius index of 4 equates to a 12x12 region (144 pixels). Each RGB Bayer domain pixel has a 14-bit dynamic range at the input of this stage. The LumaWeightRed, the LumaWeightGreen, and the LumaWeightBlue are 28, which equates to a 1.75x multiplier for each R, G and B pixel. Therefore, summing over the 144 pixels, this luma summation would have a range of 2^14 * 144 *1.75 = 15.75 * 2^18.

Assigning **LOCAL_EXPOSURE_LUMA_SUM_SHIFT** a value of 7 will bring the range down to 15.75 *2^11. This is between 2^14 and 2^15, and is the correct choice.

Please note that all four pixels have the same luma result for each Bayer quad. For each Bayer quad, there are two G, one R and one B pixel. Equal weighting of R, G and B is equivalent to a 0.5G + 0.25R + 0.25B ratio for each Bayer quad pseudo-luma calculation.

The 15-bit range of **AvgLuma** is the input to the gain curve function. The gain curve function has an input of 256 entries, where each entry corresponds to a bin size of 2^15/256, or 128 units of **AvgLuma**. The output of the gain function is a 6.10-bit gain value; e.g., 1024 corresponds to unity gain, 4096 corresponds to 4x gain, and 0 is set to black.

See Also:

None

3.2.25.1 AmbaDSP_ImgSetLocalExposure > AMBA_DSP_IMG_LOCAL_EXPOSURE_s

Type	Field	Description
UINT8	**Enb**	0: Disable 1: Enable
UINT8	**Radius**	Valid value is between 0 and 6 for 4x4 to 16x16 pixel regions.
UINT8	**LumaWeightRed**	Unsigned 5 bits
UINT8	**LumaWeightGreen**	Unsigned 5 bits
UINT8	**LumaWeightBlue**	Unsigned 5 bits
UINT8	**LumaWeightShift**	Local exposure Luma sum shift. Unsigned 5 bits.
UINT16	**GainCurveTable[256]**	Unsigned 12 bits

*Table 3-72. Definition of **AMBA_DSP_IMG_LOCAL_EXPOSURE_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetLocalExposure()**.*

3.2.26 AmbaDSP_ImgGetLocalExposure

API Syntax:

AmbaDSP_ImgGetLocalExposure (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LOCAL_EXPOSURE_s * pLocalExposure)

Function Description:

- This API is used to retrieve the local exposure settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_LOCAL_EXPOSURE_ss	***pLocalExposure**	Local exposure information. Please refer to Section 3.2.25.1 for more details.

*Table 3-73. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetLocalExposure()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-74. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetLocalExposure()**.*

Example:

None

See Also:

AmbaDSP_ImgSetLocalExposure()

3.2.27 AmbaDSP_ImgSetDeferredBlackLevel

API Syntax:

AmbaDSP_ImgSetDeferredBlackLevel (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_DEF_BLC_s * pDefBlc)

Function Description:

- This API is used to set the deferred black level.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_DEF_BLC_s	***pDefBlc**	Deferred black-level information. Please refer to Section 3.2.27.1 below for more details.

*Table 3-75. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetDeferredBlackLevel()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-76. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetDeferredBlackLevel()**.*

Example:

None

See Also:

None

3.2.27.1 AmbaDSP_ImgSetDeferredBlackLevel > AMBA_DSP_IMG_DEF_BLC_s

Type	Field	Description
UINT8	**Enb**	0: Disable 1: Enable

*Table 3-77. Definition of **AMBA_DSP_IMG_DEF_BLC_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetDeferredBlackLevel()**.*

3.2.28 AmbaDSP_ImgGetDeferredBlackLevel

API Syntax:

AmbaDSP_ImgGetDeferredBlackLevel (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_DEF_BLC_s * pDefBlc)

Function Description:

- This API is used to retrieve the deferred black-level settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_DEF_BLC_s	***pDefBlc**	Returned deferred black-level information. Please refer to Section 3.2.27.1 for more details.

*Table 3-78. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetDeferredBlackLevel()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-79. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetDeferredBlackLevel()**.*

Example:

None

See Also:

AmbaDSP_ImgSetDeferredBlackLevel()

3.2.29 AmbaDSP_ImgSetDemosaic

API Syntax:

AmbaDSP_ImgSetDemosaic (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_DEMOSAIC_s * pDemosaic)

Function Description:

- This API is used to configure demosaic settings.
- This API contains Gradient registers and Activity registers. The Gradient register is used to select between isotropic and directional interpolation for the green channel. Interpolations for the red and blue components are dependent on, and therefore follow, the green channel's interpolation. When directional interpolation has been selected, the Activity registers are used to select the directional interpolation type: constant-hue or straight-average.
- Therefore, the possible combination of interpolations are:
  - directional, constant-hue
  - directional, straight-average
  - isotropic (always straight-average)
- There are four demosaic tuning registers. Two are Gradient registers and two are Activity registers.

1. **grad_noise_thresh**:
   - This register is used to select between directional interpolation and isotropic interpolation based on a Gradient measure. When the Gradient measure falls below **grad_noise_thresh**, isotropic interpolation is chosen to reduce noise. When the Gradient measure rises above **grad_noise_thresh**, directional interpolation is chosen to avoid zipper artifacts in edge areas. This register is primarily used to achieve a balance between smoothing and "noodle-noise" in noisy flat areas, as well as a balance between zipper artifacts and proper directional interpolation for edge areas.
   - Other tunable registers are used to achieve a balance between increasing fine detail and introducing speckle artifacts.
2. **grad_clip_thresh**:
   - This register is used to determine whether to clip (or bound) the green interpolation so that it falls within its neighboring 4 green pixels' range, based upon a Gradient threshold. If the gradient measure falls below this programmed threshold, green is clipped to the range of its neighbors. If the Gradient measure rises above the threshold, no clipping occurs. This register is used to achieve a balance between increasing resolution on high-constrast fine diagonal lines and introducing speckle artifacts.

- The two Activity registers are used to determine whether to use constant-hue interpolation or straight-average interpolation based on an Activity measure. The higher the activity, the more constant-hue interpolation is favored. This results in an increase in clarity for fine-detailed areas close to neutral gray; however, speckle artifacts near high-saturated-color boundaries may occur.

3. **activity_difference_thresh**:
   - **activity** counts the number of pixel differences in a neighborhood that are larger than **activity_difference_thresh**. Therefore, larger values of **activity_difference_thresh** will tend to decrease the **activity** count.

4. **activity_thresh**:
   - If **activity count** is >= **activity_thresh**, constant-hue interpolation is used to achieve finer detail interpolation.
   - If **activity count** is < **activity_thresh**, straight-average (conservative) interpolation is used to achieve smoother and usually less noisy interpolation.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_DEMOSAIC_s	***pDemosaic**	Demosaic information. Please refer to Section 3.2.29.1 below for more details.

*Table 3-80. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetDemosaic()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-81. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetDemosaic()**.*

Example:

None

See Also:

None

3.2.29.1 AmbaDSP_ImgSetDemosaic > AMBA_DSP_IMG_DEMOSAIC_s

Type	Field	Description
UINT16	**ActivityThresh**	0-31. Activity threshold
UINT16	**ActivityDifferenceThresh**	0-16383. Activity difference threshold
UINT16	**GradClipThresh**	0-4095. Gradient clip threshold
UINT16	**GradNoiseThresh**	0-32767. Gradient noise threshold
-	-	-
-	-	-
-	-	-
-	-	-

*Table 3-82. Definition of **AMBA_DSP_IMG_DEMOSAIC_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetDemosaic()**.*

3.2.30 AmbaDSP_ImgGetDemosaic

API Syntax:

AmbaDSP_ImgGetDemosaic (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_DEMOSAIC_s * pDemosaic)

Function Description:

- This API is used to retrieve demosaic settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_DEMOSAIC_s	***pDemosaic**	Returned demosaic information. Please refer to Section 3.2.29.1 for more details.

*Table 3-83. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetDemosaic()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-84. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetDemosaic()**.*

Example:

None

See Also:

AmbaDSP_ImgSetDemosaic()

3.2.31 AmbaDSP_ImgSetColorCorrectionReg

API Syntax:

AmbaDSP_ImgSetColorCorrectionReg (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_COLOR_CORRECTION_REG_s * pColorCorrReg)

Function Description:

- This API is used to set the color-correction register filter. The color-correction stage performs a general 3-D mapping from 15-bit linear RGB to the 10-bit non-linear RGB output space to provide flexible color tuning. This mapping also handles mapping of out-of-gamut colors from the white balance gain stage back into gamut. Ambarella provides tools to generate the necessary register settings and 3D matrix for this API. The color-correction register setting is global to different color temperatures or still-capture modes; thus it should be set only once during initialization.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_COLOR_CORRECTION_REG_s	***pColorCorrReg**	Color-correction register setting information. Please refer to Section 3.2.31.1 for more details.

*Table 3-85. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetColorCorrectionReg()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-86. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetColorCorrectionReg()**.*

Example:

None

See Also:

None

3.2.31.1 AmbaDSP_ImgSetColorCorrectionReg > AMBA_DSP_IMG_COLOR_CORRECTION_REG_s

Type	Field	Description
UINT32	**RegSettingAddr**	Color-correction register setting DRAM address pointing to a 18752 bytes memory of color-correction register.

*Table 3-87. Definition of **AMBA_DSP_IMG_COLOR_CORRECTION_REG_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetColorCorrectionReg()**.*

3.2.32 AmbaDSP_ImgGetColorCorrectionReg

API Syntax:

AmbaDSP_ImgGetColorCorrectionReg (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_COLOR_CORRECTION_REG_s * pColorCorrReg)

Function Description:

- This API is used to retrieve color-correction register filter settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_COLOR_CORRECTION_REG_s	***pColorCorrReg**	Color-correction table information. Please refer to Section 3.2.31.1 for more details.

*Table 3-88. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetColorCorrectionReg()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success.
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure.

*Table 3-89. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetColorCorrectionReg()**.*

Example:

None

See Also:

AmbaDSP_ImgSetColorCorrectionReg()

3.2.33 AmbaDSP_ImgSetColorCorrection

API Syntax:

AmbaDSP_ImgSetColorCorrection (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_COLOR_CORRECTION_s * pColorCorr)

Function Description:

- This API is used to set the color-correction table. During the color-correction stage, 3-D mapping from 15-bit linear RGB to the 10-bit non-linear RGB output space is performed. The purpose of this mapping is to provide flexible color tuning, and to map out-of-gamut colors from the white balance gain stage back into gamut. Ambarella provides tools to generate the necessary register settings and 3D matrix table for this API.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_COLOR_CORRECTION_REG_s	***pColorCorr**	Color-correction register setting information. Please refer to Section 3.2.31.1 for more details.

*Table 3-90. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetColorCorrection()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-91. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetColorCorrection()**.*

Example:

None

See Also:

AmbaDSP_ImgGetColorCorrectionReg()

3.2.33.1 AmbaDSP_ImgSetColorCorrection > AMBA_DSP_IMG_COLOR_CORRECTION_s

Type	Field	Description
UINT32	**MatrixThreeDTableAddr**	3D table address pointing to the 17536 bytes of color-correction 3D matrix memory.

*Table 3-92. Definition of **AMBA_DSP_IMG_COLOR_CORRECTION_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetColorCorrection()**.*

3.2.34 AmbaDSP_ImgGetColorCorrection

API Syntax:

AmbaDSP_ImgNmGetColorCorrection (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_COLOR_CORRECTION_s * pColorCorr)

Function Description:

- This API is used to retrieve the color-correction table.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_COLOR_CORRECTION_s	***pColorCorr**	Color-correction register setting information. Please refer to Section 3.2.33.1 for more details.

*Table 3-93. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetColorCorrection()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-94. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetColorCorrection()**.*

Example:

None

See Also:

AmbaDSP_ImgSetColorCorrection()

3.2.35 AmbaDSP_ImgSetToneCurve

API Syntax:

AmbaDSP_ImgSetToneCurve (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_TONE_CURVE_s * pToneCurve)

Function Description:

- This API is used to modify the tone curve embedded in the 3D table by using a secondary gain curve. It may be used to dynamically change the contrast based on the input statistics, for instance, in order to implement an auto-knee feature.
- The feature is implemented using three tone-curve tables, one per color. There are 256 entries, and each entry is 10 bits in 8.2 fixed-point format.
- Tone curve has to be monotonic increasing.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_TONE_CURVE_s	***pToneCurve**	Tone curve tables. Please refer to Section 3.2.35.1 for more details.

*Table 3-95. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetToneCurve()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-96. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetToneCurve()**.*

Example:

None

See Also:

None

3.2.35.1 AmbaDSP_ImgSetToneCurve > AMBA_DSP_IMG_TONE_CURVE_s

Type	Field	Description
UINT16	**ToneCurveRed[256]**	This table consists of 256 values for color component red. Each value occupies 16 bits, but only the 10 least significant bits are valid. Each value is in the range of [0, 255] represented by 8.2 bits.
UINT16	**ToneCurveGreen[256]**	
UINT16	**ToneCurveBlue[256]**	

*Table 3-97. Definition of **AMBA_DSP_IMG_TONE_CURVE_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetToneCurve()**.*

3.2.36 AmbaDSP_ImgGetToneCurve

API Syntax:

AmbaDSP_ImgGetToneCurve (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_TONE_CURVE_s * pToneCurve)

Function Description:

- This API is used to retrieve the tone curve settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_TONE_CURVE_s	***pToneCurve**	Tone curve tables. Please refer to Section 3.2.35.1 for more details.

*Table 3-98. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetToneCurve()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-99. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetToneCurve()**.*

Example:

None

See Also:

AmbaDSP_ImgSetToneCurve()

3.2.37 AmbaDSP_ImgSetRgbToYuvMatrix

API Syntax:

AmbaDSP_ImgSetRgbToYuvMatrix (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_RGB_TO_YUV_s * pRgbToYuv)

Function Description:

- This API is used to set the RGB-to-YUV matrix. The function converts a given RGB input into a YUV signal.

$$\begin{bmatrix} Y' \\ U' \\ V' \end{bmatrix} = \begin{bmatrix} m[0] & m[1] & m[2] \\ m[3] & m[4] & m[5] \\ m[6] & m[7] & m[8] \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} Yoff \\ Uoff \\ Voff \end{bmatrix}$$

- The final converted YUV signal is in the range of [0,255].

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_RGB_TO_YUV_s	***pRgbToYuv**	RGB-to-YUV matrix. Please refer to Section 3.2.39.1 below for more details.

*Table 3-100. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetRgbToYuvMatrix()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-101. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetRgbToYuvMatrix()**.*

Example:

None

See Also:

None

3.2.37.1 AmbaDSP_ImgSetRgbToYuvMatrix > AMBA_DSP_IMG_RGB_TO_YUV_s

Type	Field	Description
INT16	**MatrixValue[9]**	Matrix coefficients for RGB-to-YUV color space conversion. Each matrix coefficient is in the range (-4,4) and is represented by 13 bits (sign + 2.10 bits).
INT16	**OffsetY**	Represented by 11 bits (sign + 10 bits) covering the range [-1024, 1023].
INT16	**OffsetU**	Represented by 9 bits (sign + 8 bits) covering the range [-256,255].
INT16	**OffsetV**	Represented by 9 bits (sign + 8 bits) covering the range [-256,255].

*Table 3-102. Definition of **AMBA_DSP_IMG_RGB_TO_YUV_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetRgbToYuvMatrix()**.*

3.2.38 AmbaDSP_ImgGetRgbToYuvMatrix

API Syntax:

AmbaDSP_ImgGetRgbToYuvMatrix (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_RGB_TO_YUV_s * pRgbToYuv)

Function Description:

- This API is used to retrieve the RGB-to-YUV matrix.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_RGB_TO_YUV_s	***pRgbToYuv**	RGB-to-YUV matrix. Please refer to Section 3.2.39.1 for more details.

*Table 3-103. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetRgbToYuvMatrix()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-104. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetRgbToYuvMatrix()**.*

Example:

None

See Also:

AmbaDSP_ImgSetRgbToYuvMatrix()

3.2.39 AmbaDSP_ImgSetChromaScale

API Syntax:

AmbaDSP_ImgSetChromaScale (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CHROMA_SCALE_s * pChromaScale)

Function Description:

- This API is used to set the chroma saturation of each pixel based on luma (Y). Each Y value is used to access a 128-entry look-up table (LUT) which generates a 2.10-bit scaling value. U and V are multiplied by the scaling-value then adjusted to ensure that the final YUV triple is a legal RGB value. To ensure the YUV values can be transformed to a legal RGB value range, the YUV-to-RGB mapping of the display device must be considered. Consequently, users are asked to provide the display device type in the parameter **Enb**.
- This function may be used to increase the chroma saturation in shadow areas for low-ISO settings (low-noise conditions). It may also be used to decrease the shadow chroma saturation in high-ISO settings, as chroma noise may be dominant under these conditions.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_CHROMA_SCALE_s	***pChromaScale**	Chroma scale information. Please refer to Section 3.2.41.1 below for more details.

*Table 3-105. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetChromaScale()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-106. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetChromaScale()**.*

Example:

None

See Also:

None

3.2.39.1 AmbaDSP_ImgSetChromaScale > AMBA_DSP_IMG_CHROMA_SCALE_s

Type	Field	Description
UINT8	**Enb**	0: Disable 1: Enable, PC-style display device 2: Enable, HDTV-style display device
UINT16	**GainCurve[128]**	128 16-bit entries. Each entry is unsigned 12-bit.

*Table 3-107. Definition of **AMBA_DSP_IMG_CHROMA_SCALE_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetChromaScale()**.*

3.2.40 AmbaDSP_ImgGetChromaScale

API Syntax:

AmbaDSP_ImgGetChromaScale (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CHROMA_SCALE_s * pChromaScale)

Function Description:

- This API is used to retrieve the chroma scale settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_CHROMA_SCALE_s	***pChromaScale**	Chroma scale information. Please refer to Section 3.2.41.1 for more details.

*Table 3-108. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetChromaScale()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-109. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetChromaScale()**.*

Example:

None

See Also:

None

3.2.41 AmbaDSP_ImgSetChromaMedianFilter

API Syntax:

AmbaDSP_ImgSetChromaMedianFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CHROMA_MEDIAN_FILTER_s * pChromaMedian)

Function Description:

- This API is used to set the chroma median filter. This filter is useful in reducing false-color artifacts.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_CHROMA_MEDIAN_FILTER_s	***pChromaMedian**	Chroma median filter information. Please refer to Section 3.2.41.1 below for more details.

*Table 3-110. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetChromaMedianFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-111. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetChromaMedianFilter()**.*

Example:

None

See Also:

None

3.2.41.1 AmbaDSP_ImgSetChromaMedianFilter > AMBA_DSP_IMG_CHROMA_MEDIAN_FILTER_s

Type	Field	Description
int	**Enb**	0: Disable 1: Enable
UINT16	**CbAdaptiveStrength**	0 – 256. Chroma median filter adaptive strength for Cb/Cr component.
UINT16	**CrAdaptiveStrength**	
UINT8	**CbNonAdaptiveStrength**	0 – 31. Chroma median filter non-adaptive strength for Cb/Cr component.
UINT8	**CrNonAdaptiveStrength**	
UINT16	**CbAdaptiveAmount**	0 – 256
UINT16	**CrAdaptiveAmount**	0 – 256

*Table 3-112. Definition of **AMBA_DSP_IMG_CHROMA_MEDIAN_FILTER_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetChromaMedianFilter()**.*

3.2.42 AmbaDSP_ImgGetChromaMedianFilter

API Syntax:

AmbaDSP_ImgGetChromaMedianFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CHROMA_MEDIAN_FILTER_s * pChromaMedian)

Function Description:

- This API is used to retrieve the chroma median filter settings. This filter is useful in reducing false-color artifacts.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_CHROMA_MEDIAN_FILTER_s	***pChromaMedian**	Chroma median filter information. Please refer to Section 3.2.41.1 for more details.

*Table 3-113. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetChromaMedianFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-114. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetChromaMedianFilter()**.*

Example:

None

See Also:

None

3.2.43 AmbaDSP_ImgSetColorDependentNoiseReduction

API Syntax:

AmbaDSP_ImgSetColorDependentNoiseReduction (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CDNR_INFO_s * pCdnr)

Function Description:

- This API is used to reduce noise in colors where color-correction can cause noise amplification.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_CDNR_INFO_s	***pCdnr**	Color-dependent noise reduction information. Please refer to Section 3.2.43.1 for more details.

*Table 3-115. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetColorDependentNoiseReduction()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-116. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetColorDependentNoiseReduction()**.*

Example:

None

See Also:

None

3.2.43.1 AmbaDSP_ImgSetColorDependentNoiseReduction > AMBA_DSP_IMG_CDNR_INFO_s

Type	Field	Description
INT32	**CdnrMode**	0: Off 1: On
-	-	-

*Table 3-117. Definition of **AMBA_DSP_IMG_CDNR_INFO_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetColorDependentNoiseReduction()**.*

3.2.44 AmbaDSP_ImgGetColorDependentNoiseReduction

API Syntax:

AmbaDSP_ImgGetColorDependentNoiseReduction (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CDNR_INFO_s * pCdnr)

Function Description:

- This API is used to retrieve the color-dependent noise reduction settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_CDNR_INFO_s	***pCdnr**	Color-dependent noise reduction information. Please refer to Section 3.2.43.1 for more details.

*Table 3-118. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetColorDependentNoiseReduction()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-119. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetColorDependentNoiseReduction()**.*

Example:

None

See Also:

AmbaDSP_ImgSetColorDependentNoiseReduction()

3.2.45 AmbaDSP_ImgSetLumaProcessingMode

API Syntax:

AmbaDSP_ImgSetLumaProcessingMode (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SHP_A_SELECT_e Select)

Function Description:

- This API is used to determine running LISO pipeline, 2-pass LISO pipeline, or 3-pass LISO pipeline. It can also program a customizable block included in the A12 IDSP. In the LISO and 2-pass LISO pipeline, this block can be programmed to perform sharpening or noise filtering operations. When programmed to sharpening mode, **AmbaDSP_ImgSet1stSharpenNoise** Xxx APIs are valid and **AmbaDSP_ImgSetAdvanceSpatialFilter** is invalid; when programed to noise-filtering mode, **AmbaDSP_ImgSetAdvanceSpatialFilter** is valid and **AmbaDSP_ImgSet1stSharpenNoise** Xxx APIs are invalid.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
-	-	-
AMBA_DSP_IMG_LISO_PROCESS_SELECT_s	***pLisoProcessSelect**	LISO process select information. Please refer to Section 3.2.45.1 for more details.

*Table 3-120. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetLumaProcessingMode()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-121. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetLumaProcessingMode()**.*

Example:

None

See Also:

None

3.2.45.1 AmbaDSP_ImgSetLumaProcessingMode > AMBA_DSP_IMG_LISO_PROCESS_SELECT_s

Type	Field	Description
UINT8	**AdvancedFeaturesEnable**	0: LISO 1: 3-passes LISO 2: 2-passes LISO
UINT8	**UseSharpenNotAsf**	This parameter is only used when AdvancedFeaturesEnable is 0 or 1. 0: Program sharpen as advanced spatial filter 1: Program sharpen as sharpen filter

*Table 3-122. Definition of **AMBA_DSP_IMG_CDNR_INFO_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetLumaProcessingMode()**.*

3.2.46 AmbaDSP_ImgGetLumaProcessingMode

API Syntax:

AmbaDSP_ImgGetLumaProcessingMode (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SHP_A_SELECT_e Select)

Function Description:

- This API is used to program a customizable block included in the A12 IDSP. In the video pipeline, this block can be programmed to perform sharpening or noise filtering operations. When programmed to sharpening mode, **AmbaDSP_ImgSet1stSharpenNoise** Xxx APIs are valid and **AmbaDSP_ImgSetAdvanceSpatialFilter** is invalid; when programed to noise-filtering mode, **AmbaDSP_ImgSetAdvanceSpatialFilter** is valid and **AmbaDSP_ImgSet1stSharpenNoise** Xxx APIs are invalid.
- For some digital effects such as Drawing, this block can be programmed to perform edge extraction.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
-	-	-
AMBA_DSP_IMG_LISO_PROCESS_SELECT_s	***pLisoProcessSelect**	Returned LISO process select information. Please refer to Section 3.2.45.1 for details.

*Table 3-123. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetLumaProcessingMode()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-124. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetLumaProcessingMode()**.*

Example:

None

See Also:

AmbaDSP_ImgSetLumaProcessingMode

3.2.47 AmbaDSP_ImgSetAdvanceSpatialFilter

API Syntax:

AmbaDSP_ImgSetAdvanceSpatialFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_ASF_INFO_s * pAsf)

Function Description:

- This API is used to set the advanced spatial filter. This filter functions as follows:

1. An edge direction and edge amount are determined.
2. The sample is filtered parallel to the edge.
3. The sample is filtered isotropically.
4. The result of number 2 and number 3 are blended; the more the sample seems to lie on an edge, the more often number 2 results are chosen.
5. The result of number 4 is linearly combined with the input sample based on the difference between them.
6. The result of number 5 is linearly combined with the input sample based on level, so that the fractional amount of filtering performed is based on level. The output of number 6 is as follows:

   Output of number 6 = Output of number 5 * S / 64 + original sample * (1 – S / 64).

7. The change in the final output (compared to the input pixel) is limited based on the maximum change amount. The final result is clamped to [original sample – max_change, original sample + max_change].

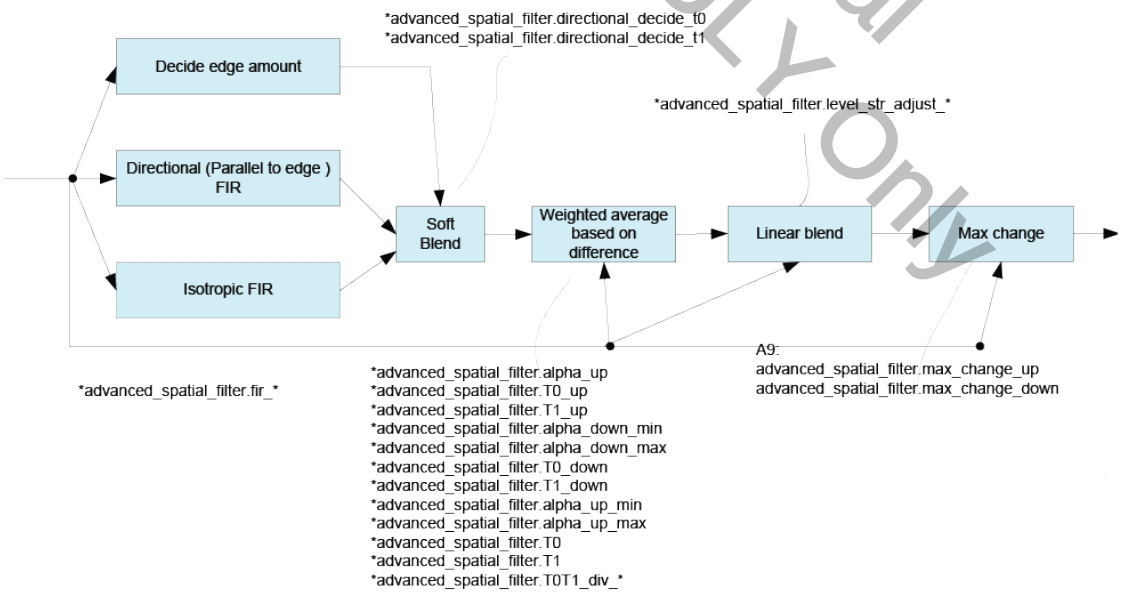

Figure 3-1. Advanced Spatial Filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_ASF_INFO_s	***pAsf**	Advanced spatial filter information. Please refer to Section 3.2.47.1 for more details.

*Table 3-125. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetAdvanceSpatialFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-126. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetAdvanceSpatialFilter()**.*

Example:

None

See Also:

None

3.2.47.1 AmbaDSP_ImgSetAdvanceSpatialFilter > AMBA_DSP_IMG_ASF_INFO_s

A12 edge-detection is controlled by **wide_edge_detect**. Increasing this parameter enhances sensitivity and increases edge-direction decisions for wide high-contrast edges, regardless of proximity to an edge. Smaller values enhance sensitivity and increase edge-direction decisions for closely spaced lines and finer details. The effect of this parameter is generally subtle in nature; however, note that the following issues can occur.

- When the value of **wide_edge_detect** is small, there may be an increase in artifacts a few pixels away from high-contrast edges. For example, on the dark side of an edge, white spots or lines may be introduced a few pixels away from the edge.

- When the value of **wide_edge_detect** is large, there may be an increase in the blurring of closely spaced lines or a reduction in fine detail.

For each pixel, if an edge score falls below **directional_decide_t0** it is filtered isotropically. It is filtered directionally if it rises above **directional_decide_t1**. If an edge score falls between these two values, a combined isotropic and directional filtering approach is used.

To fully disable the filter, set **MaxChangeUp** / **MaxChangeDown** = 0.

Type	Field	Description
UINT8	**Enb**	0 - 1
AMBA_DSP_IMG_FIR_s	**Fir**	Please refer to Section 3.2.47.2. for more details.
UINT8	**DirectionalDecideT0**	0 - 255.
UINT8	**DirectionalDecideT1**	0 - 255.
AMBA_DSP_IMG_FULL_ADAPTATION_s	**Adapt**	Please refer to Section 3.2.47.3 for more details.
AMBA_DSP_IMG_LEVEL_s	**LevelStrAdjust**	Please refer to Section 3.2.47.4 for more details.
AMBA_DSP_IMG_LEVEL_s	**T0T1Div**	Please refer to Section 3.2.47.4 for more details.
UINT8	**MaxChangeUp**	0 - 255.
UINT8	**MaxChangeDown**	0 - 255.

*Table 3-127. Definition of **AMBA_DSP_IMG_ASF_INFO_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetAdvanceSpatialFilter()**.*

3.2.47.2 AmbaDSP_ImgSetAdvanceSpatialFilter > AMBA_DSP_IMG_FIR_s

There are five ways to specify filter strength (Specify 0-4). The table below lists filter strength options.

Specify	Directions	Parameters Used	Description
0	Isotropic only	**StrengthISO**	Single strength determines finite impulse response (FIR) size.
1	Isotropic only	**Coefs**	Only isotropic but fully manual
2	Isotropic and Directional	**StrengthISO StrengthDir**	One strength for isotropic, one strength for directional.
3	Isotropic and Directional	**PerDirFirIsoStrengths PerDirFirDirStrengths PerDirFirDirAmounts**	For each direction, the user specifies an isotropic strength, a directional strength, and the amount to blend isotropic and directional.
4	Isotropic and Directional	**Coefs**	Fully manual

Table 3-128. Filter Strength Options.

Please refer to the following notes when specifying filter strength.

- When Specify 1 is used, the number of coefficient used in **Coefs** is 10.
- The finite impulse response (FIR) coefficients should sum to 0.
- Taps are specified in units of 1/256, with units as shown above.

								Coefficient	Bits	Range
0	1	2	3	2	1	0				
1	4	5	6	5	4	1		9	9	[-256, 255]
2	5	7	8	7	5	2		8	8	[-128, 127]
3	6	8	9	8	6	3		7	7	[-64, 63]
2	5	7	8	7	5	2		4-6	6	[-32, 31]
1	4	5	6	5	4	1		0-3	5	[-16, 15]
0	1	2	3	2	1	0				

Figure 3-2. Finite Impulse Response (FIR) Coefficient.

- When Specify 4 is used, the number of the coefficient is 9*25 = 225.
- In Specify 2 mode, the strength of the directional filter (step 2) is determined by **dir_strength**, and the strength of the isotropic filter (step 3) is determined by **iso_strength**.
- In Specify 3 mode, the strength for each direction can be determined separately. In addition, some directions can use a combination of directional and isotropic filtering. Specifically, strengths for isotropic (direction 0) and 8 directions (1-8) are set. The direction for each edge is shown below:

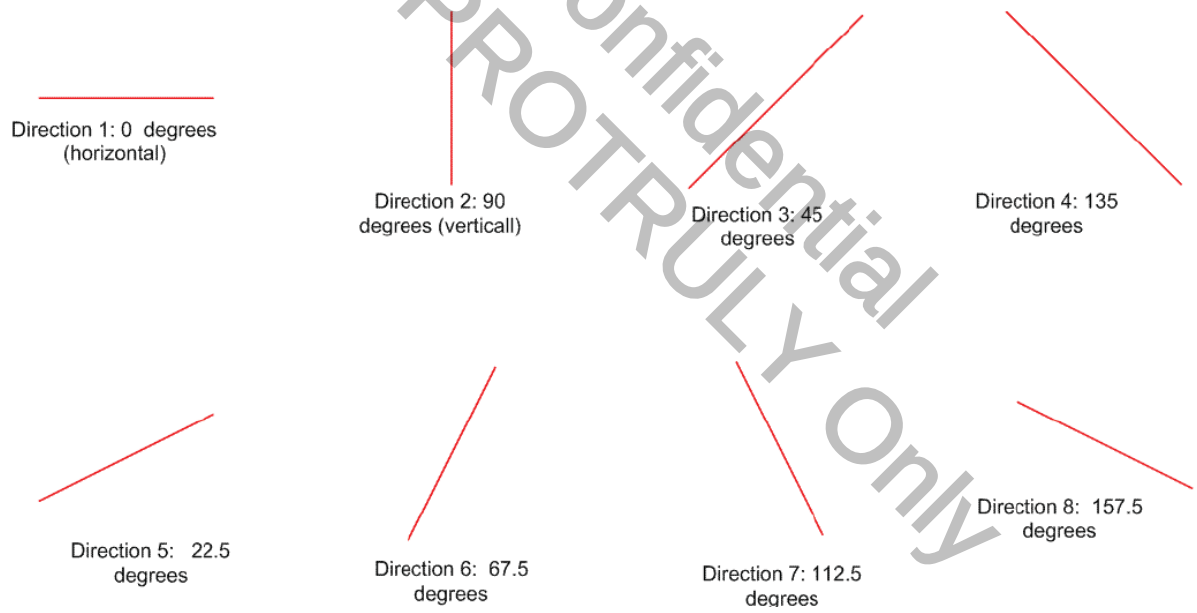

Figure 3-3. Direction For Each Edge.

The directional filter for edge K is a weighted combination of:

Directional filter of strength dir_strengths[K], with weight dir_amounts[K]/256.
An isotropic filter of strength iso_strength[K], with weight 1 - dir_amounts[K]/256.

Type	Field	Description
UINT8	**Specify**	0-4
UINT16	**PerDirFirIsoStrengths[9]**	0-256 (specify = 3)
UINT16	**PerDirFirDirStrengths[9]**	0-256 (specify = 3)
UINT16	**PerDirFirDirAmounts[9]**	0-256 (specify = 3)
INT16	**Coefs[9][25]**	0-1023 (specify = 1, 4)
UINT16	**StrengthIso**	0-256 (specify = 0, 2)
UINT16	**StrengthDir**	0-256 (specify = 2)
UINT8	**WideEdgeDetect**	0-8. Only used for AdvanceSpatialFilter.
UINT16	**EdgeThresh**	0-2047. Only used for HighIsoSetHighIsoFreqRecover.

*Table 3-129. Definition of **AMBA_DSP_IMG_FIR_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetAdvanceSpatialFilter()**.*

3.2.47.3 AmbaDSP_ImgSetAdvanceSpatialFilter > AMBA_DSP_IMG_FULL_ADAPTATION_s

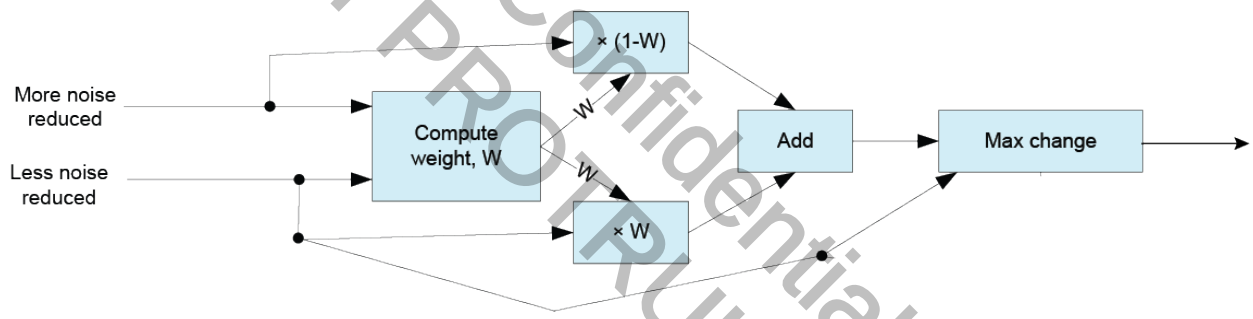

Figure 3-4. Adaptation.

Adaptation takes a weighted average of a less noise-reduced signal and more noise-reduced signal (see Figure 3-4) and then limits the maximum change from the less noise-reduced of the two. The concept behind this approach is that if the noise reduction process introduced a very large change it is most likely due to the removal of the underlying signal, which should be restored.

The first step is to create a weighted average of the less noise-reduced signal and more noise-reduced signal. This is computed based on the difference between the two signals, the alpha_min, alpha_max, T0 and T1 parameters.

In addition, T0T1_div can be used to adjust the weight computation based on level. If a T0T1_div level control set is available for the filter, then T0 and T1 are divided by the result of level adaptation. This can also be viewed as multiplying the result of the difference between the two signals when computing the weight (i.e., either moving both T0 and T1 in the above charts or scaling the X axis; the effect is identical). For T0T1_div, a higher high_strength, low_strength and mid_strength will result in a lower T0 and T1 (i.e., assigning a higher weight for the less noise-reduced signal, resulting in less filtering.)

Type	Field	Description
UINT8	**AlphaMinUp**	0 - 8
UINT8	**AlphaMaxUp**	0 - 8
UINT8	**T0Up**	0 - 252, even only.
UINT8	**T1Up**	2 - 254, even only.
UINT8	**AlphaMinDown**	0 - 8
UINT8	**AlphaMaxDown**	0 - 8
UINT8	**T0Down**	0 - 252, even only.
UINT8	**T1Down**	2 - 254, even only.
-	-	-

*Table 3-130. Definition of **AMBA_DSP_IMG_FULL_ADAPTATION_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetAdvanceSpatialFilter()**.*

3.2.47.4 AmbaDSP_ImgSetAdvanceSpatialFilter > AMBA_DSP_IMG_LEVEL_s

For each of these sets of controls, the definition of shadow, mid-tone and highlight levels, as well as the transitions between them, is determined by Low, LowDelta, High, and HighDelta. Strength value of 0 has no effect.

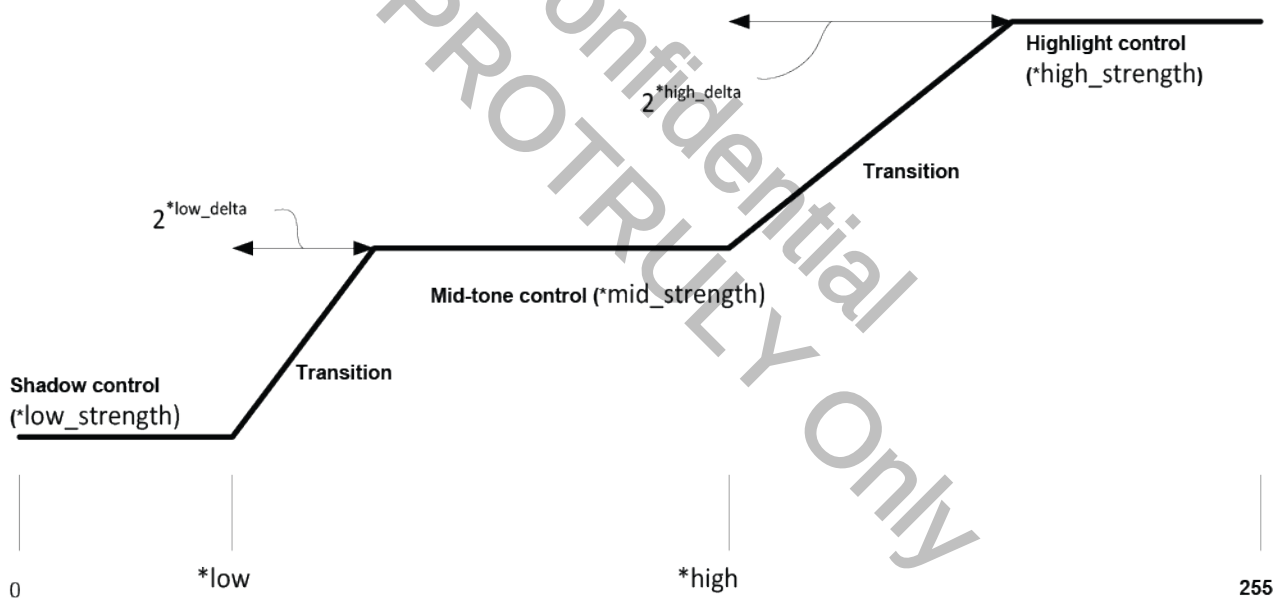

Figure 3-5. Level Control Parameters.

Type	Field	Description
UINT8	**Low**	0 - 255
UINT8	**LowDelta**	0 - 7
UINT8	**LowStrength**	0 - 255 (0-64 for LevelStrAdjust of ASF, 0-16 for LevelStrAdjust of Sharpen)
UINT8	**MidStrength**	0 - 255 (0-64 for LevelStrAdjust of ASF, 0-16 for LevelStrAdjust of Sharpen)
UINT8	**High**	0 - 255
UINT8	**HighDelta**	0 - 7
UINT8	**HighStrength**	0 - 255 (0-64 for LevelStrAdjust of ASF, 0-16 for LevelStrAdjust of Sharpen)
-	-	-

*Table 3-131. Definition of **AMBA_DSP_IMG_LEVEL_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetAdvanceSpatialFilter()**.*

3.2.48 AmbaDSP_ImgGetAdvanceSpatialFilter

API Syntax:

AmbaDSP_ImgGetAdvanceSpatialFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_ASF_INFO_s * pAsf)

Function Description:

- This API is used to retrieve advanced spatial filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_ASF_INFO_s	***pAsf**	Advanced spatial filter information. Please refer to Section 3.2.49.1 for more details.

*Table 3-132. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetAdvanceSpatialFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-133. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetAdvanceSpatialFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgSetAdvanceSpatialFilter()

3.2.49 AmbaDSP_ImgSet1stSharpenNoiseBoth

API Syntax:

AmbaDSP_ImgSet1stSharpenNoiseBoth (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SHARPEN_BOTH_s * pSharpenBoth)

Function Description:

- This API is used to set the Sharpen filter.

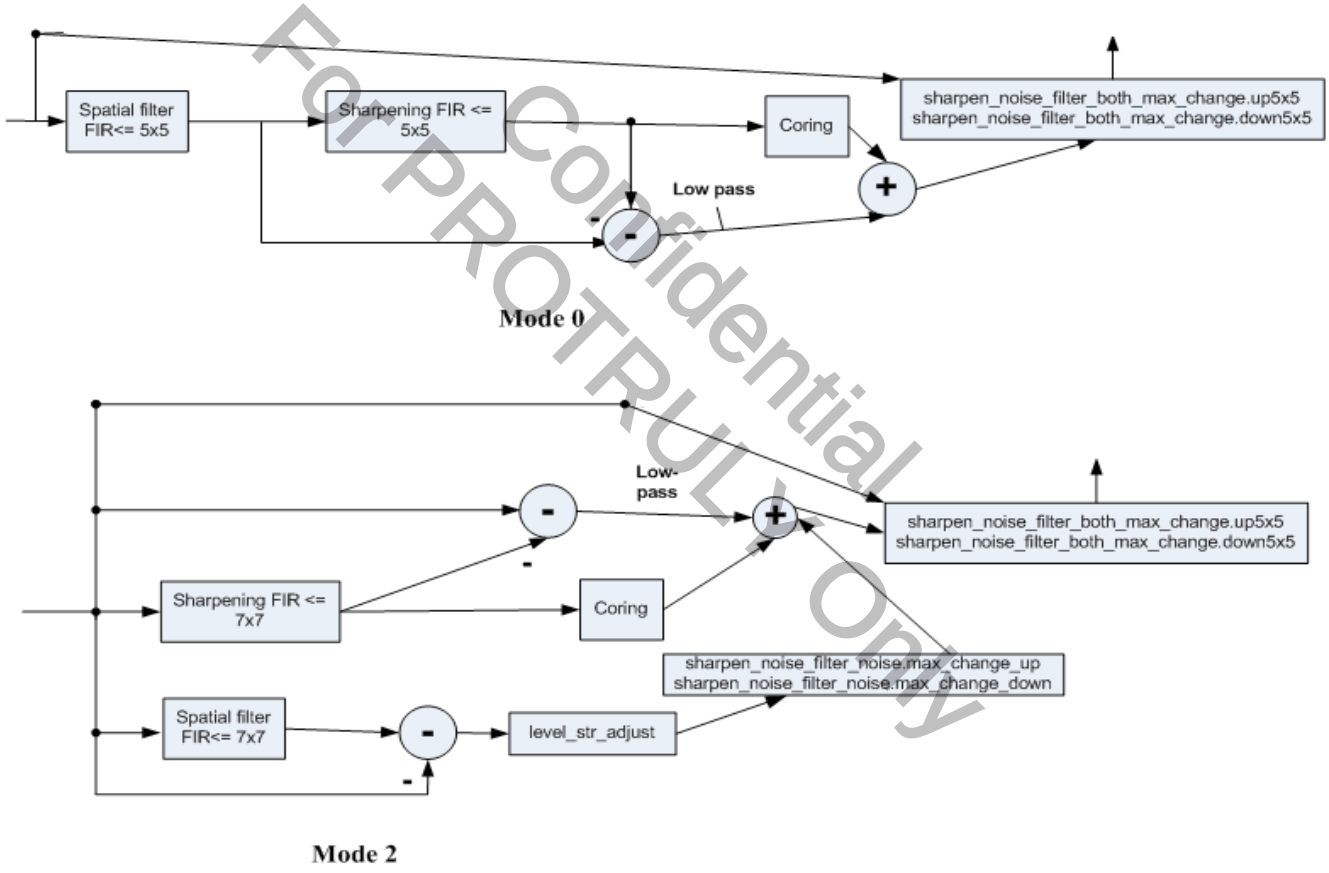

Figure 3-6. Sharpen and Spatial Filter.

- When sharpening is performed, it is performed in conjunction with a spatial filter, which can either occur before sharpening (Mode 0) or work in parallel with it (Mode 2).

- **edge_threshold** is used for spatial filtering in both Mode 0 and Mode 2, and also for the direction/ isotropic decision for directional sharpening in Mode 2.

- A12 edge-detection is controlled by **wide_edge_detect**. Increasing this parameter enhances sensitivity and increases edge-direction decisions for wide high-contrast edges, regardless of proximity to an edge. Smaller values enhance sensitivity and increase edge-direction decisions for closely spaced lines and finer details. The effect of this parameter is generally subtle in nature; however, note that the following issues can occur.
  - When the value of **wide_edge_detect** is small, there may be an increase in artifacts a few pixels away from high-contrast edges. For example, on the dark side of an edge, white spots or lines may be introduced a few pixels away from the edge.
  - When the value of **wide_edge_detect** is large, there may be an increase in the blurring of closely spaced lines or a reduction in fine detail.
- For each pixel, the user determines a wide edge score (W) and narrow edge score (N). edge_score = wide_edge_detect * W + (8- wide_edge_detect) * N. N uses pixel-to-pixel differences, W uses 2-pixel-away differences.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_SHARPEN_BOTH_s	***pSharpenBoth**	Sharpen maximum change information. Please refer to Section 3.2.49.1 for more details.

*Table 3-134. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseBoth()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-135. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseBoth()**.*

Example:

None

See Also:

AmbaDSP_ImgGet1stSharpenNoiseBoth()

3.2.49.1 AmbaDSP_ImgSet1stSharpenNoiseBoth > AMBA_DSP_IMG_SHARPEN_BOTH_s

Type	Field	Description
UINT8	**Enable**	0 -1
UINT16	**EdgeThresh**	0 - 2047
UINT8	**WideEdgeDetect**	0 - 8
-	-	-
-	-	-
UINT8	**MaxChangeUp5x5**	0-255
UINT8	**MaxChangeDown5x5**	Up5x5 and Down5x5 control the maximum change upward and downward, respectively, relative to the 5x5 maximum around the pixel being filtered.
UINT8	**Mode**	0 or 2

*Table 3-136. Definition of **AMBA_DSP_IMG_SHARPEN_BOTH_s** for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseBoth**().*

3.2.50 AmbaDSP_ImgGet1stSharpenNoiseBoth

API Syntax:

AmbaDSP_ImgGet1stSharpenNoiseBoth (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SHARPEN_BOTH_s * pSharpenBoth)

Function Description:

- This API is used to retrieve sharpen filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_SHARPEN_BOTH_s	***pSharpenBoth**	Returned sharpen maximum change information. Please refer to Section 3.2.49.1 for details.

*Table 3-137. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGet1stSharpenNoiseBoth()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-138. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGet1stSharpenNoiseBoth()**.*

Example:

None

See Also:

AmbaDSP_ImgSet1stSharpenNoiseBoth()

3.2.51 AmbaDSP_ImgSet1stSharpenNoiseNoise

API Syntax:

AmbaDSP_ImgSet1stSharpenNoiseNoise (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SHARPEN_NOISE_s * pSharpenNoise)

Function Description:

- This API is used to set the Sharpen noise filter.
- The spatial filter finite impulse response (FIR) is a 5x5 or 7x7 filter (depending on **1stSharpenNoiseBoth.mode**: 5x5 for Mode 0 and 7x7 for Mode 2). The format for specifying 5x5 FIRs is the same as that for setting 7x7 FIRs; however, some parameters are constrained.
- In the Mode 2 diagram, the bottom path is for spatial filtering. **noise.max_change** can be used to control its spatial filter output and **noise.SpatialFir** can be used to change the 7x7 spatial filter FIR.
- Level strength adjust is similar to the ***level_str_adjust_*** parameters in the advanced spatial filter, with the exception that the maximum strength is 16 (as opposed to 64).

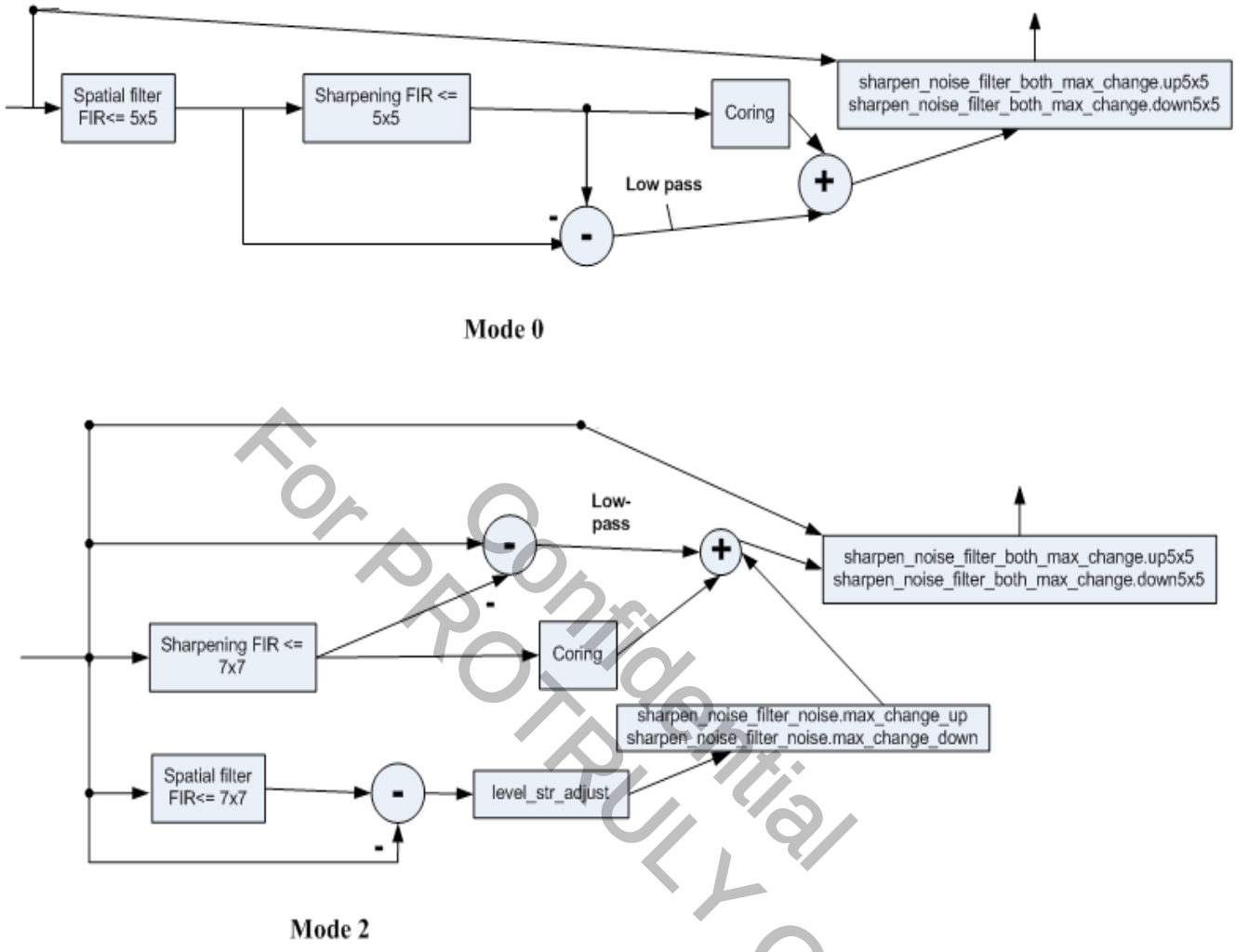

Figure 3-7. Sharpen and Spatial Filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_SHARP-EN_NOISE_s	***pSharpenNoise**	TBD. Please refer to Section 3.2.51.1 for more details.

*Table 3-139. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseNoise()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-140. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseNoise()**.*

Example:

None

See Also:

AmbaDSP_ImgGet1stSharpenNoiseNoise()

3.2.51.1 AmbaDSP_ImgSet1stSharpenNoiseNoise > AMBA_DSP_IMG_SHARPEN_NOISE_s

Type	Parameter	Description
UINT8	**MaxChangeUp**	0 - 255
UINT8	**MaxChangeDown**	0 - 255
AMBA_DSP_IMG_FIR_s	**SpatialFir**	Please refer to Section 3.2.49.2 for definition.
AMBA_DSP_IMG_LEVEL_s	**LevelStrAdjust**	Please refer to Section 3.2.49.4 for definition.
UINT8	**LevelStrAdjustNotT0T1-Level-Based**	0: Use **T0, T1, AlphaMin, AlphaMax** for level control. 1: Use **LevelStrAdjust** for level control.
UINT8	**T0**	0-255
UINT8	**T1**	0-255
UINT8	**AlphaMin**	0-16
UINT8	**AlphaMax**	0-16

*Table 3-141. Definition of **AMBA_DSP_IMG_SHARPEN_NOISE_s** for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseNoise()**.*

3.2.52 AmbaDSP_ImgGet1stSharpenNoiseNoise

API Syntax:

AmbaDSP_ImgGet1stSharpenNoiseNoise (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SHARPEN_NOISE_s * pSharpenNoise)

Function Description:

- This API is used to retrieve information regarding the Sharpen noise filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_SHARP-EN_NOISE_s	***pSharpenNoise**	TBD. Please refer to Section 3.2.51.1 for more details.

*Table 3-142. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGet1stSharpenNoiseNoise()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-143. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGet1stSharpenNoiseNoise()**.*

Example:

None

See Also:

AmbaDSP_ImgSet1stSharpenNoiseNoise()

3.2.53 AmbaDSP_ImgSet1stSharpenNoiseSharpenFir

API Syntax:

AmbaDSP_ImgSet1stSharpenNoiseSharpenFir (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_FIR_s * pFir)

Function Description:

- This API is used to set the finite impulse response (FIR) table of the Sharpen filter while the **AmbaDSP_ImgSet1stSharpenNoiseCoring** API sets the coring table.
- The FIR is a 5x5 or 7x7 filter (depending on **1stSharpenNoiseBoth.mode**: 5x5 for mode 0 and 7x7 for mode 2) which can be used to implement a high-pass filter. The output of the FIR is used to look-up the coring table for coring multipliers. The multiplier is then multiplied by the FIR output. The FIR output is also used to subtract from the original input to generate a low-pass signal, which is eventually added to the coring output.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_FIR_s	***pFir**	FIR filter information. Please refer to Section 3.2.47.2 for more details.

*Table 3-144. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseSharpenFir()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-145. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseSharpenFir()**.*

Example:

None

See Also:

AmbaDSP_ImgGet1stSharpenNoiseSharpenFir()

3.2.54 AmbaDSP_ImgGet1stSharpenNoiseSharpenFir

API Syntax:

AmbaDSP_ImgGet1stSharpenNoiseSharpenFir (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_FIR_s * pFir)

Function Description:

- This API is used to retrieve the finite impulse response (FIR) table of the Sharpen filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_FIR_s	***pFir**	Returned FIR filter information. Please refer to Section 3.2.47.2 for more details.

*Table 3-146. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGet1stSharpenNoiseSharpenFir()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-147. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGet1stSharpenNoiseSharpenFir()**.*

Example:

None

See Also:

AmbaDSP_ImgSet1stSharpenNoiseSharpenFir()

3.2.55 AmbaDSP_ImgSet1stSharpenNoiseSharpenCoring

API Syntax:

AmbaDSP_ImgSet1stSharpenNoiseSharpenCoring (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CORING_s * pCoring)

Function Description:

- This API is used to set the coring table of the Sharpen filter. The output of the FIR is used to look-up coring multipliers from the coring table. The multiplier is then multiplied by the FIR output. There are 256 entries in the coring table to cover the full range of FIR outputs, with positive and negative outputs treated separately. Interpolation is used between entries.
- A coring table entry of 0 maps to the smallest FIR output (i.e., the largest negative value). A coring table entry of 128 maps to a FIR output equal to 0. A coring table entry of 255 maps to the largest positive FIR output value. The 256 coring table entries are 2.3 bits. Thus, an entry value of 8 results in unity gain. A value less than 8 attenuates the high-pass signal, which results in noise reduction and reduced sharpness.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_CORING_s	***pCoring**	Coring table information. Please refer to Section 3.2.55.1 for more details.

*Table 3-148. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseSharpenCoring()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-149. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseSharpenCoring()**.*

Example:

None

See Also:

AmbaDSP_ImgGet1stSharpenNoiseSharpenCoring()

3.2.55.1 AmbaDSP_ImgSetSharpenACoring > AMBA_DSP_IMG_CORING_s

Type	Parameter	Description
UINT8	**Coring[256]**	Unsigned 5 bits. Each entry is in 2.3 bit format.
UINT8	**FractionalBits**	1-3. Fractional bit number. 1: Coring entry is 4.1 format. 2: Coring entry is 3.2 format. 3: Coring entry is 2.3 format.

*Table 3-150. Definition of **AMBA_DSP_IMG_CORING_s** for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseCoring**().*

3.2.56 AmbaDSP_ImgGet1stSharpenNoiseSharpenCoring

API Syntax:

AmbaDSP_ImgGet1stSharpenNoiseSharpenCoring (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CORING_s * pCoring)

Function Description:

- This API is used to retrieve the coring table of the Sharpen filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_CORING_s	***pCoring**	Coring table information. Please refer to Section 3.2.55.1 for more details.

*Table 3-151. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGet1stSharpenNoiseSharpenCoring()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-152. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGet1stSharpenNoiseSharpenCoring()**.*

Example:

None

See Also:

AmbaDSP_ImgSet1stSharpenNoiseSharpenCoring()

3.2.57 AmbaDSP_ImgSet1stSharpenNoiseSharpenCoringIndexScale

API Syntax:

AmbaDSP_ImgSet1stSharpenNoiseSharpenCoringIndexScale (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s * pLevel)

Function Description:

- This API is used to set the scale factor for the coring index in shadow, mid-tone and high-light (HIGH) areas. The definition of the level control diagram is the same as previously specified. Referring to the chart below, the chosen strength value will shift the original coring table index. Strength greater than 16 indicates the use of a higher edge contract index. Strength less than 16 indicates the use of a lower edge contract index. A strength value of 16 has no effect.

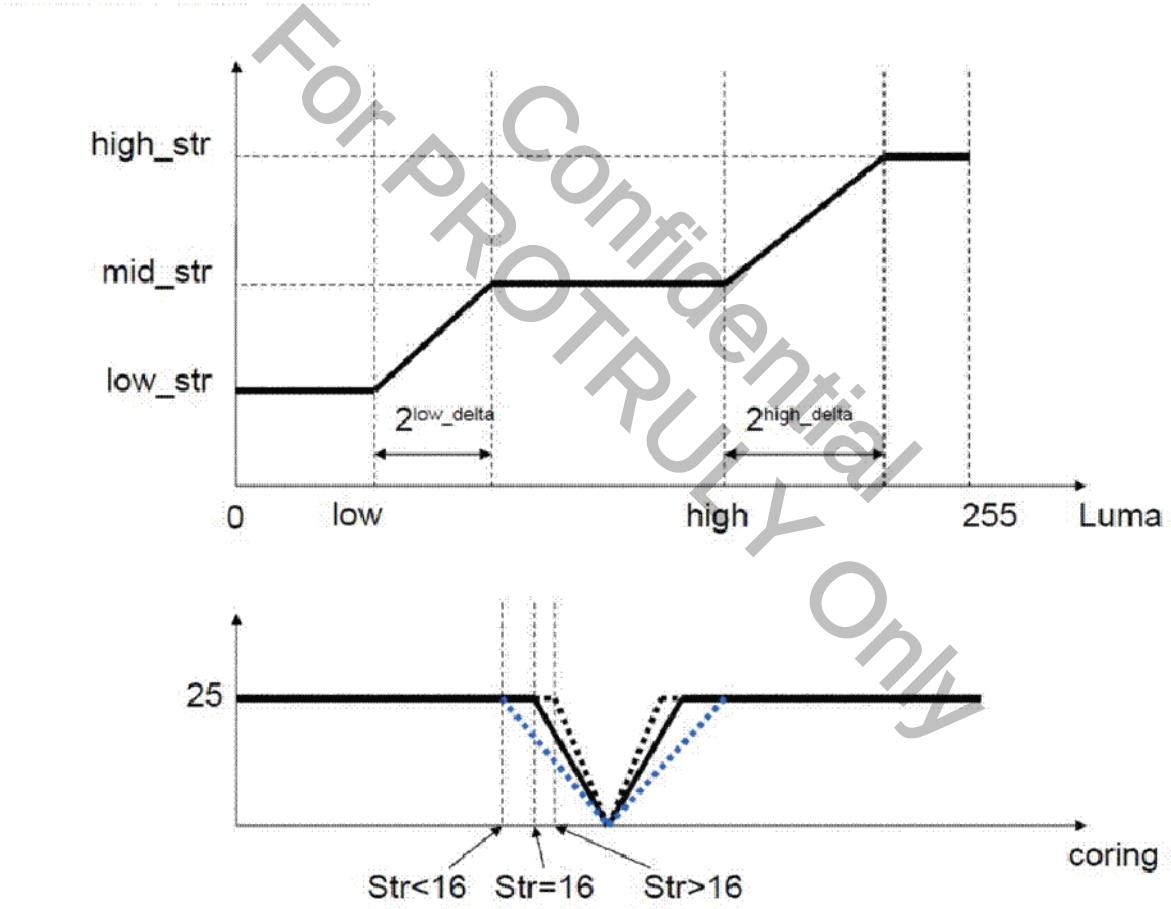

Figure 3-8. Coring Index Scale.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	Coring index scale information. Please refer to Section 3.2.47.4 for more details.

*Table 3-153. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseSharpenCoringIndexScale()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-154. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseSharpenCoringIndexScale()**.*

Example:

After interpolating based on level to obtain an overall_level in [0, 255], the index is scaled from the center of the coring table by overall_level / 16. For example, with no scaling, the user would be at position 130.5 – i.e., 2.5 from the center. Refer to the following four cases:

1. overall_level = 16. Use 130.5; i.e., average of entries 130 and 131
2. overall_level = 0. Use entry 128
3. overall_level = 40. Scale 2.5 by 40/16 to retrieve 2.5 * 40 / 16 = 6.25. Use ¾ of entry 134 plus ¼ of entry 135.
4. overall_level = 8. Scale 2.5 by 8/16 to retrieve 2.5 * 8 / 16 = 1.25. Use ¾ of entry 129 plus ¼ of entry 130.

See Also:

AmbaDSP_ImgGet1stSharpenNoiseSharpenCoringIndexScale()

3.2.58 AmbaDSP_ImgGet1stSharpenNoiseSharpenCoringIndexScale

API Syntax:

AmbaDSP_ImgGet1stSharpenNoiseSharpenCoringIndexScale (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s * pLevel)

Function Description:

- This API is used to retrieve the scale factor for the coring index.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	Coring index scale information. Please refer to Section 3.2.47.4 for more details.

*Table 3-155. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGet1stSharpenNoiseSharpenCoringIndexScale()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-156. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGet1stSharpenNoiseSharpenCoringIndexScale()**.*

Example:

None

See Also:

AmbaDSP_ImgSet1stSharpenNoiseSharpenCoringIndexScale()

3.2.59 AmbaDSP_ImgSet1stSharpenNoiseSharpenMinCoringResult

API Syntax:

AmbaDSP_ImgSet1stSharpenNoiseSharpenMinCoringResult (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s * pLevel)

Function Description:

- This API is used to set the minimum coring multiplier.

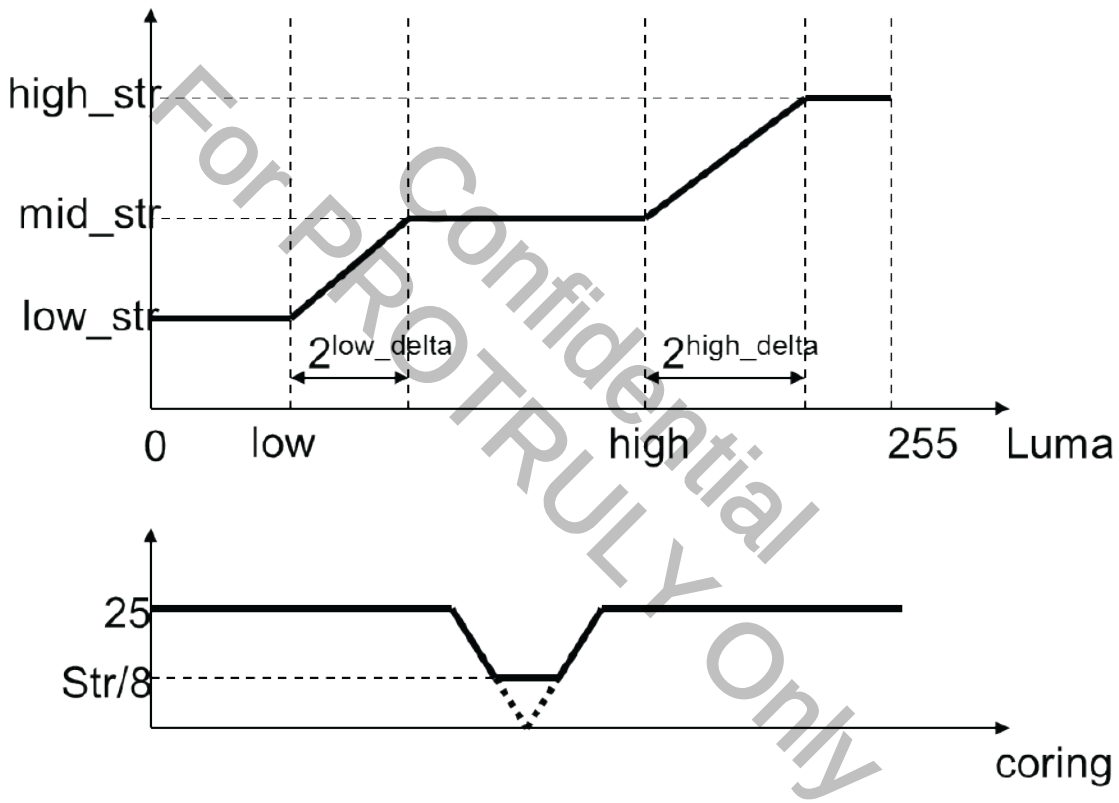

Figure 3-9. Settings in Minimum Coring Multiplier.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	Level minimum information. Please refer to Section 3.2.47.4 for more details.

*Table 3-157. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseSharpenMinCoringResult()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

Table 3-158. Returns for Common Filter Image Kernel API ***AmbaDSP_ImgSet1stSharpenNoiseSharpenMinCoringResult()***.

Example:

None

See Also:

None

3.2.60 AmbaDSP_ImgGet1stSharpenNoiseSharpenMinCoringResult

API Syntax:

AmbaDSP_ImgGet1stSharpenNoiseSharpenMinCoringResult (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s * pLevel)

Function Description:

- This API is used to retrieve the minimum coring multiplier.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	Level minimum information. Please refer to Section 3.2.47.4 for more details.

*Table 3-159. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGet1stSharpenNoiseSharpenMinCoringResult()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-160. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGet1stSharpenNoiseSharpenMinCoringResult()**.*

Example:

None

See Also:

AmbaDSP_ImgSet1stSharpenNoiseSharpenMinCoringResult()

3.2.61 AmbaDSP_ImgSet1stSharpenNoiseSharpenScaleCoring

API Syntax:

AmbaDSP_ImgSet1stSharpenNoiseSharpenScaleCorin (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s * p Level)

Function Description:

- This API is used to set the scale factor of the coring table.
- new_coring = old_coring * (level strength) / 16

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_LEVEL_s	***p Level**	Please refer to Section 3.2.47.4 for more details.

*Table 3-161. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseSharpenScaleCoring()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-162. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSet1stSharpenNoiseSharpenScaleCoring()**.*

Example:

None

See Also:

AmbaDSP_ImgGet1stSharpenNoiseSharpenScaleCoring()

3.2.62 AmbaDSP_ImgGet1stSharpenNoiseSharpenScaleCoring

API Syntax:

AmbaDSP_ImgGet1stSharpenNoiseSharpenScaleCoring (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s * pLevel)

Function Description:

- This API is used to retrieve the scale factor of the coring table.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	Coring scale information. Please refer to Section 3.2.47.4 for more details.

*Table 3-163. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGet1stSharpenNoiseSharpenScaleCoring()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-164. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGet1stSharpenNoiseSharpenScaleCoring()**.*

Example:

None

See Also:

None

- **edge_threshold** is used for spatial filtering in both Mode 0 and Mode 2, and also for the directional sharpening direction/isotropic determination in Mode 2.
- A12 edge-detection is controlled by **wide_edge_detect**. Increasing this parameter enhances sensitivity and increases edge-direction decisions for wide high-contrast edges, regardless of proximity to an edge. Smaller values enhance sensitivity and increase edge-direction decisions for closely spaced lines and finer details. The effect of this parameter is generally subtle in nature; however, note that the following issues can occur.
  - When the value of **wide_edge_detect** is small, there may be an increase in artifacts a few pixels away from high-contrast edges. For example, on the dark side of an edge, white spots or lines may be introduced a few pixels away from the edge.
  - When the value of **wide_edge_detect** is large, there may be an increase in the blurring of closely spaced lines or a reduction in fine detail.
- For each pixel, the user determines a wide edge score (W) and narrow edge score (N). edge_score = wide_edge_detect * W + (8- wide_edge_detect) * N. N uses pixel-to-pixel differences, W uses 2-pixel away differences.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_SHARPEN_BOTH_s	***p Level**	Sharpen information. Please refer to Section 3.2.47.4 below for details.

*Table 3-165. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetFinalSharpenNoiseBoth()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-166. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetFinalSharpenNoiseBoth()**.*

Example:

None

See Also:

AmbaDSP_ImgGetFinalSharpenNoiseBoth()

3.2.62.1 AmbaDSP_ImgSetFinalSharpenNoiseBoth > AMBA_DSP_IMG_SHARPEN_BOTH_s

Type	Parameter	Description
UINT8	**Enable**	0 - 1
UINT16	**EdgeThresh**	0 - 2047
UINT8	**WideEdgeDetect**	0 - 8
UINT8	**MaxChangeup5x5**	0-255
UINT8	**MaxChangeDown5x5**	Up5x5 and Down5x5 control the maximum change upward and sownward, respectively, relative to the 5x5 maximum around the pixel being filtered.
UINT8	**Mode**	0 or 2

*Table 3-167. Definition of **AMBA_DSP_IMG_SHARPEN_BOTH_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetFinalSharpenNoiseBoth()**.*

3.2.62.2 AmbaDSP_ImgSetFinalSharpenNoiseSharpenCoring > AMBA_DSP_IMG_CORING_s

Type	Field	Description
UINT8	**Coring[256]**	Unsigned 5 bits. Each entry is

*Table 3-168. Definition of **AMBA_DSP_IMG_CORING_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetFinalSharpenNoiseSharpenCoring()**.*

3.2.63 AmbaDSP_ImgSetResamplerCoefAdj

API Syntax:

AmbaDSP_ImgSetResamplerCoefAdj (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_RESAMPLER_COEF_ADJ_s * pResamplerCoefAdj)

Function Description:

- This API is used to set the resampler coefficient. It can be adjusted to act as a low-pass filter and to cause intentional blurring. This can be used for advanced digital effects which require defocus, such as a “wedding effect”.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_RESAM-PLER_COEF_ADJ_s	***pResamplerCoefAdj**	Resampler coefficient adjustment information. Please refer to Section 3.2.63.1 below for details.

*Table 3-169. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetResamplerCoefAdj()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-170. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetResamplerCoefAdj()**.*

Example:

None

See Also:

None

3.2.63.1 AmbaDSP_ImgSetResamplerCoefAdj > AMBA_DSP_IMG_RESAMPLER_COEF_ADJ_s

Type	Field	Description
UINT32	**ControlFlag**	0x0: Blackman 0x1: RECTWIN 0x2: COEFF_M2 0x4: COEFF_M4 0x10: COEFF_LP_STRONG 0x20: COEFF_LP_MEDIUM
UINT16	**ResamplerSelect**	0x1: CFA 0x2: MAIN 0x4: PRV_A 0x8: PRV_B 0x10: PRV_C 0x20: CFA_V 0x40: MAIN_V 0x80: PRV_A_V 0x100: PRV_B_V 0x200: PRV_C_V
UINT16	**Mode**	0: Always 1: One frame

*Table 3-171. Definition of **AMBA_DSP_IMG_RESAMPLER_COEF_ADJ_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetResamplerCoefAdj()**.*

3.2.64 AmbaDSP_ImgGetResamplerCoefAdj

API Syntax:

AmbaDSP_ImgGetResamplerCoefAdj (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_RESAMPLER_COEF_ADJ_s * pResamplerCoefAdj)

Function Description:

- This API is used to retrieve the resampler coefficient adjustment settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_RESAMPLER_COEF_ADJ_s	***pResamplerCoefAdj**	Returned resampler coefficient adjustment information. Please refer to Section 3.2.63.1 for details.

*Table 3-172. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetResamplerCoefAdj()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-173. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetResamplerCoefAdj()**.*

Example:

None

See Also:

AmbaDSP_ImgSetResamplerCoefAdj()

3.2.65 AmbaDSP_ImgSetChromaFilter

API Syntax:

AmbaDSP_ImgSetChromaFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CHROMA_FILTER_s * pChromaFilter)

Function Description:

- This API is used to configure the chroma filter.
- Each sample is classified as normal or fine detail. A fine-detail picture differs in value from most pixels in its support region. Different filter settings are applied for normal and fine-detail pixels.
- **NoiseLevel** is set to the average noise level expected for the chroma data. Increasing **NoiseLevel** increases filtering strength.
- **Radius** determines the spatial extent of the filter. Increasing **radius** increases filtering strength.
- **OriginalBlendStrength** determines the weight of the center (original) pixel. Higher values increase the weight of the center pixel.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_FILTER_s	***pChromaFilter**	Chroma filter information. Please refer to Section 3.2.65.1 below for details.

*Table 3-174. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetChromaFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-175. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetChromaFilter()**.*

Example:

None

See Also:

None

3.2.65.1 AmbaDSP_ImgSetChromaFilter > AMBA_DSP_IMG_CHROMA_FILTER_s

Type	Field	Description
UINT8	**Enable**	0 - 1
UINT8	**NoiseLevelCb**	0 - 255
UINT8	**NoiseLevelCr**	0 - 255
UINT16	**OriginalBlendStrengthCb**	0 - 256
UINT16	**OriginalBlendStrengthCr**	0 - 256
UINT16	**Radius**	32, 64, 128

*Table 3-176. Definition of **AMBA_DSP_IMG_CHROMA_FILTER_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetChromaFilter()**.*

3.2.66 AmbaDSP_ImgGetChromaFilter

API Syntax:

AmbaDSP_ImgGetChromaFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CHROMA_FILTER_s * pChromaFilter)

Function Description:

- This API is used to retrieve chroma filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_FILTER_s	***pChromaFilter**	Returned chroma filter information. Please refer to Section 3.2.65.1 for details.

*Table 3-177. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetChromaFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-178. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetChromaFilter()**.*

Example:

None

See Also:

None

3.2.67 AmbaDSP_ImgSetGbGrMismatch

API Syntax:

AmbaDSP_ImgSetGbGrMismatch (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_GBGR_MISMATCH_s * pGbGrMismatch)

Function Description:

- Strong GbGr correction functions as follows. First, it detects whether there is a consistent mismatch in a small area (if **narrow_enable** = 1) or if there is mismatch in a wide area (**WideEnable** = 1). If either condition occurs, the center pixels are averaged with the average of the neighboring four green pixels.
- Narrow detection is off / on with enable, but otherwise exerts no control.
- Wide detection passes if the following is true:
  - A measure of mismatch is greater than **WideThresh**, so increasing **WideThresh** weakens the filter.
  - A measure of how likely the system mismatch is caused by a true signal is less than **WideSafety**, so increasing wide_safety increases filter strength.
  - For a pixel on which GbGr correction is performed, the CFA noise filter is disabled. Refer to the diagram on the following page.

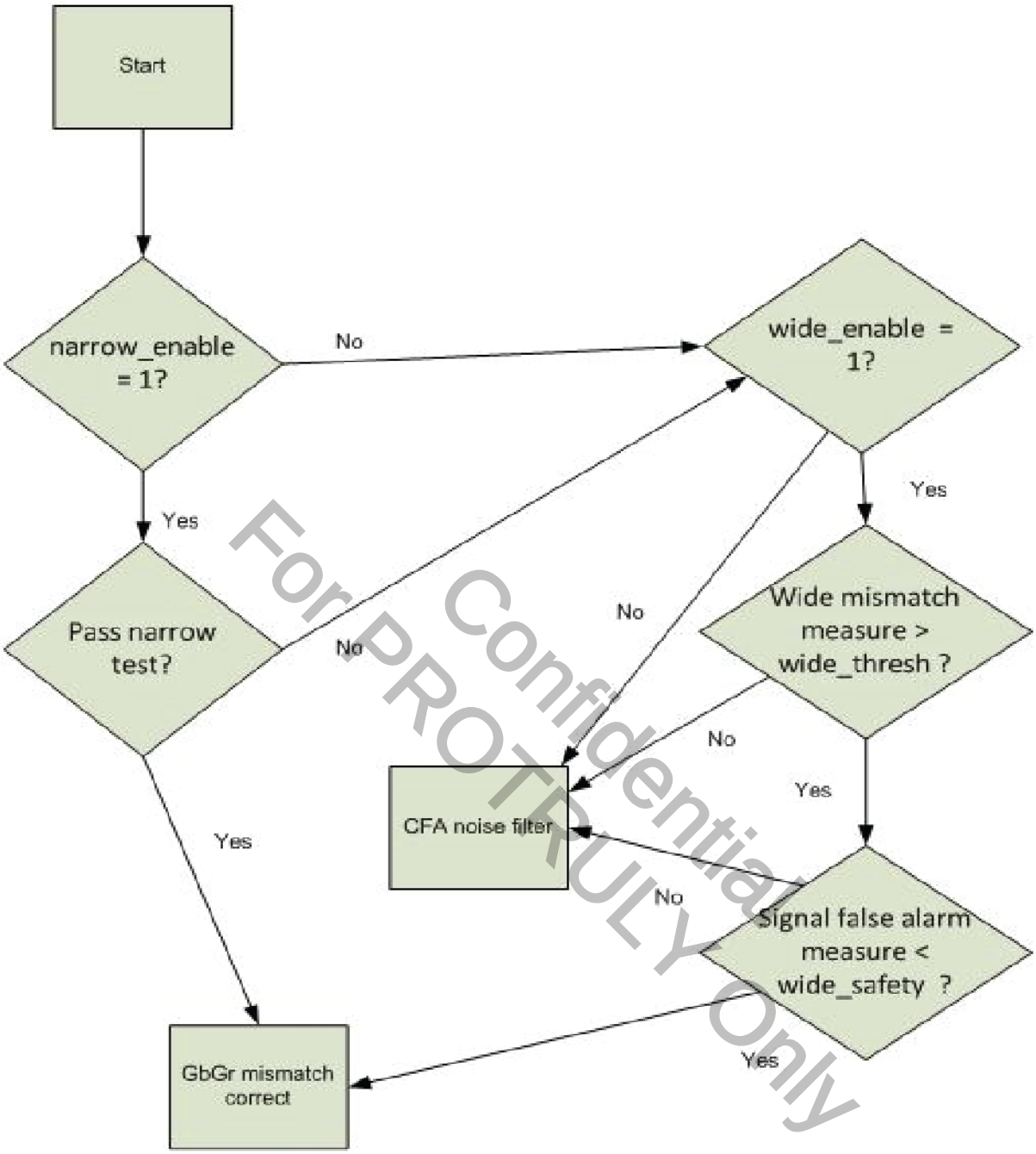

Figure 3-10. GBGR Correction.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_GBGR_MISMATCH_s	***pGbGrMismatch**	Returned chroma filter information. Please refer to Section 3.2.67.1 for details.

*Table 3-179. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetGbGrMismatch()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically.

*Table 3-180. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetGbGrMismatch()**.*

Example:

None

See Also:

None

3.2.67.1 AmbaDSP_ImgSetGbGrMismatch > AMBA_DSP_IMG_GBGR_MISMATCH_s

Type	Field	Description
UINT8	**NarrowEnable**	0: Disable 1: Enable
UINT8	**WideEnable**	0: Disable 1: Enable
UINT16	**WideSafety**	0 - 256. See description.
UINT16	**WideThresh**	0 - 256. See description.

*Table 3-181. Definition of **AMBA_DSP_IMG_GBGR_MISMATCH_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetGbGrMismatch()**.*

3.2.68 AmbaDSP_ImgGetGbGrMismatch

API Syntax:

AmbaDSP_ImgGetGbGrMismatch (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_GBGR_MISMATCH * pGbGrMismatch)

Function Description:

- This API is used to retrieve GbGr mismatch information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_GBGR_MISMATCH_s	***pGbGrMismatch**	Returned chroma filter information. Please refer to Section 3.2.67.1 for details.

*Table 3-182. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetGbGrMismatch()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-183. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetGbGrMismatch()**.*

Example:

None

See Also:

None

3.2.69 AmbaDSP_ImgCalcWarpCompensation

API Syntax:

AmbaDSP_ImgCalcWarpCompensation (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_WARP_CALC_INFO_s * pWarpCalcInfo)

Function Description:

- This API is used to calculate warp compensation information from the calibration warp table. The calibration warp data contains a warp compensation table array (Each vector is sign 11.4 format). The Image Kernel will crop the appropriate range of calibration tables based on the relationship between calibration VIN geometry and current VIN geometry, and resample them to the hardware format. The result will be stored in the Image Kernel internal context, and will be issued until API **AmbaDSP_ImgSetWarpCompensation** is invoked.
- This API also determines the image DSP pipeline scaling and cropping configuration, where a digital zoom effect can be implemented through those window combinations. Please refer to Table 3-6 for definition.

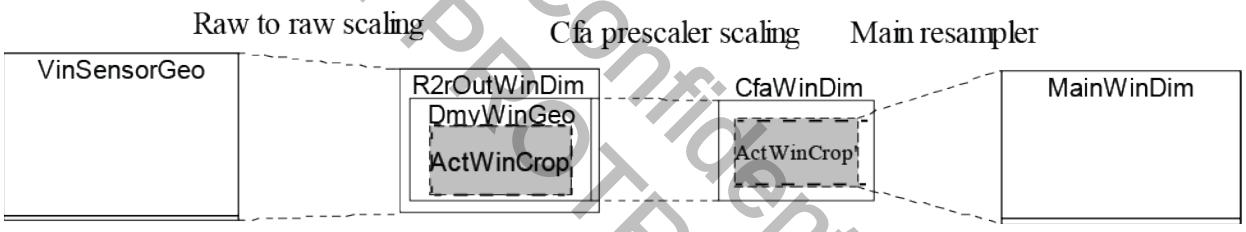

Figure 3-11. Warp Table.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_WARP_CALC_INFO_s	***pWarpCalcInfo**	Warp calculation information. Please refer to Section 3.2.69.1 for details.

*Table 3-184. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgCalcWarpCompensation()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-185. Returns for Common Filter Image Kernel API **AmbaDSP_ImgCalcWarpCompensation()**.*

See Also:

AmbaDSP_ImgSetWarpCompensation()

3.2.69.1 AmbaDSP_ImgCalcWarpCompensation > AMBA_DSP_IMG_WARP_CALC_INFO_s

Type	Field	Description
UINT32	**WarpEnb**	0: Disable warp. 1: Enable warp.
UINT32	**Control**	Control flag. 1: All scaling is performed in section 2.
AMBA_DSP_IMG_CALIB_WARP_INFO_s	**CalibWarpInfo**	Calibration warp information. This is where the calibrated warp table grid numbers, tile size, and sensor geometry information are stored. Please refer to Section 3.2.69.2 below for details.
AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s	**VinSensorGeo**	Current VIN sensor geometry. Image Kernel APIs will calculate the warp table based on the relationship between current VIN sensor geometry and calibration geometry. Please refer to Section 3.2.69.3 below for details.
AMBA_DSP_IMG_WIN_DIMENSION_s	**R2rOutWinDim**	If raw-to-raw scaling is enabled, it is required to assign the raw-to-raw output dimension. Unit is pixels. Please refer to Section 3.2.69.4 below for details.
AMBA_DSP_IMG_WIN_GEOMETRY_s	**DmyWinGeo**	Dummy window geometry. Unit is pixels. StartX, StartY, and Width, and Height must be even values. The margins between the dummy window and actual window are for EIS rolling-shutter correction usage. Please refer to Section 3.2.69.5 below for details.
AMBA_DSP_IMG_WIN_DIMENSION_s	**CfaWinDim**	CFA prescaler output window dimension. Unit is pixels. Width and Height must be even-numbered values. Please refer to Section 3.2.69.4 below for details.
AMBA_DSP_IMG_WIN_CO-ORDINATES _s	**ActWinCrop**	Actual window coordinates, indicating the FOV. These coordinates lie on the dummy window domain but not on the CFA window domain. In 16.16 format. Please refer to Section 3.2.69.6 below for details.
AMBA_DSP_IMG_WIN_DIMENSION_s	**MainWinDim**	Main window dimension. Unit is pixels. Please refer to Section 3.2.69.4 below for details.
AMBA_DSP_IMG_WIN_DIMENSION_s	**PrevWinDim[2]**	Preview window dimension. 0 for Preview A and 1 for Preview B. Unit is pixels. Please refer to Section 3.2.69.4 below for details.
AMBA_DSP_IMG_WIN_DIMENSION_s	**ScreennailDim**	Screennail window dimension. Unit is pixels. Please refer to Section 3.2.69.4 below for definition.
AMBA_DSP_IMG_WIN_DIMENSION_s	**ThumbnailDim**	Thumbnail window dimension. Unit is pixels. Please refer to Section 3.2.69.4 below for details.
INT32	**HorSkewPhaseInc**	Horizontal skew phase increment. For EIS.

*Table 3-186. Definition of **AMBA_DSP_IMG_WARP_CALC_INFO_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcWarpCompensation()**.*

3.2.69.2 AmbaDSP_ImgCalcWarpCompensation > AMBA_DSP_IMG_CALIB_WARP_INFO_s

Type	Field	Description
UINT32	**Version**	Warp calibration information version. Current version number is 0x20130101.
INT32	**HorGridNum**	Horizontal grid number
INT32	**VerGridNum**	Vertical grid number
INT32	**TileWidthExp**	Tile width exponent. Tile width is 2^**TileWidthExp**. Note that (**HorGridNum**-1) << **TileWidthExp** must be greater than or equal to **VinSensorGeo.Width**.
INT32	**TileHeightExp**	Tile height exponent. Tile height is 2^**TileHeightExp**. Note that (**VerGridNum**-1) << **TileHeightExp** must be greater than or equal to **VinSensorGeo.Height**.
AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s	**VinSensorGeo**	VIN sensor geometry when calibrating. Please refer to Section 3.2.69.3 below for details.
UINT32	**Reserved**	Reserved bytes for extension.
UINT32	**Reserved1**	Reserved bytes for extension.
UINT32	**Reserved2**	Reserved bytes for extension.
AMBA_DSP_IMG_GRID_POINT_s	***pWarp**	Calibration warp vectors pointer. It should be pointed to a memory with size (**HorGridNum***VerGridNum*2 Bytes). Each vector is 11.4 format. Please refer to Section 3.2.69.7 below for details.

*Table 3-187. Definition of **AMBA_DSP_IMG_CALIB_WARP_INFO_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcWarpCompensation()**.*

3.2.69.3 AmbaDSP_ImgCalcWarpCompensation > AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s

Type	Field	Description
UINT32	**StartX**	Offset in X direction. Unit is pixels.
UINT32	**StartY**	Offset in Y direction. Unit is pixels.
UINT32	**Width**	Width. Unit is pixels.
UINT32	**Height**	Height. Unit is pixels.
AMBA_DSP_IMG_SENSOR_SUBSAMPLING_s	**HSubSample**	Horizontal sample rate. It represents the binning mode of the sensor in horizontal. Please refer to Section 3.2.5.4 for details.
AMBA_DSP_IMG_SENSOR_SUBSAMPLING_s	**VSubSample**	Vertical sample rate. It represents the binning mode of the sensor in vertical. Please refer to Section 3.2.5.4 for details.

*Table 3-188. Definition of **AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcWarpCompensation()**.*

3.2.69.4 AmbaDSP_ImgCalcWarpCompensation > AMBA_DSP_IMG_WIN_DIMENSION_s

Type	Field	Description
UINT32	**Width**	Width. Unit is pixels.
UINT32	**Height**	Height. Unit is pixels.

*Table 3-189. Definition of **AMBA_DSP_IMG_WIN_DIMENSION_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcWarpCompensation()**.*

3.2.69.5 AmbaDSP_ImgCalcWarpCompensation > AMBA_DSP_IMG_WIN_GEOMETRY_s

Type	Field	Description
UINT32	**StartX**	Offset in X direction. Unit is pixels.
UINT32	**StartY**	Offset in Y direction. Unit is pixels.
UINT32	**Width**	Width. Unit is pixels.
UINT32	**Height**	Height. Unit is pixels.

*Table 3-190. Definition of **AMBA_DSP_IMG_WIN_GEOMETRY_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcWarpCompensation()**.*

3.2.69.6 AmbaDSP_ImgCalcWarpCompensation > AMBA_DSP_IMG_WIN_COORDINTATES_s

Type	Field	Description
UINT32	**LeftTopX**	X position of top-left point. 16.16 format.
UINT32	**LeftTopY**	Y position of top-left point. 16.16 format.
UINT32	**RightBotX**	X position of bottom-right point. 16.16 format.
UINT32	**RightBotY**	Y position of bottom-right point. 16.16 format.

*Table 3-191. Definition of **AMBA_DSP_IMG_WIN_COORDINTATES_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcWarpCompensation()**.*

3.2.69.7 AmbaDSP_ImgCalcWarpCompensation > AMBA_DSP_IMG_GRID_POINT_s

Type	Field	Description
INT16	**X**	Horizontal vector
INT16	**Y**	Vertical vector

*Table 3-192. Definition of **AMBA_DSP_IMG_GRID_POINT_s** for Common Filter Image Kernel API **AmbaDSP_ImgCalcWarpCompensation()**.*

3.2.70 AmbaDSP_ImgSetWarpCompensation

API Syntax:

AmbaDSP_ImgSetWarpCompensation (AMBA_DSP_IMG_MODE_CFG_s *pMode)

Function Description:

- This API is used to trigger the warp settings calculated by **AmbaDSP_ImgCalcWarpCompensation()**.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.

*Table 3-193. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetWarpCompensation()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-194. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetWarpCompensation()**.*

Example:

None

See Also:

AmbaDSP_ImgCalcWarpCompensation()

3.2.71 AmbaDSP_ImgGetWarpCompensation

API Syntax:

AmbaDSP_ImgGetWarpCompensation (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_WARP_CALC_INFO_s * pWarpCalcInfo)

Function Description:

- This API is used to retrieve the warp compensation settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_WARP_CALC_INFO_s	***pWarpCalcInfo**	Returned Warp information. Please refer to Section 3.2.69.1 for details.

*Table 3-195. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetWarpCompensation()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-196. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetWarpCompensation()**.*

Example:

None

See Also:

AmbaDSP_ImgCalcWarpCompensation()
AmbaDSP_ImgSetWarpCompensation()

3.2.72 AmbaDSP_ImgSetVideoMctf

API Syntax:

AmbaDSP_ImgSetVideoMctf (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_VIDEO_MCTF_INFO_s * pMctfInfo)

Function Description:

- This API is used to set the motion-compensated temporal filtering (MCTF) in video mode. Each sample is combined with a sample from the previous picture in a three-step process.
- When Temporal Adjust is disabled, threshold/alpha[0-2] control Y, Cb, and Cr channels, threshold/alpha[3] is invalid.
- When Temporal adjust is enabled, threshold/alpha[0-3] are used for different TA score. Y, Cb, and Cr share same parameters.

  1. A weight W is computed using the following parameters:

     threshold_0, threshold_1, threshold_2, threshold_3
     alpha1, alpha2, alpha3

  2. A preliminary filtered sample is computed as:

     preliminary = ((W-256) * previous + W * current) / 256

  3. The final filtered sample is computed similarly to the preliminary filtered sample; however, the current sample is limited to **y_max_change**, **u_max_change**, or **v_max_change**, depending on the component of the sample.

- The **radius** parameter affects how many neighboring samples are used in the 3D and spatial filters. If **radius** is equal to 0, then 5 samples are used. If **radius** equals 256, then 25 samples are used. The **level_adjust** parameter increases the strength of MCTF (3D and spatial) in shadows and weaker in highlights. Once MCTF is complete, the pre- and post-MCTF luma data are combined, resulting in a more natural looking image. The strength of the combination is controlled by **combined_str_y**; 0 has no effect, and larger values can enhance the natural look of the image but can also weaken noise reduction.

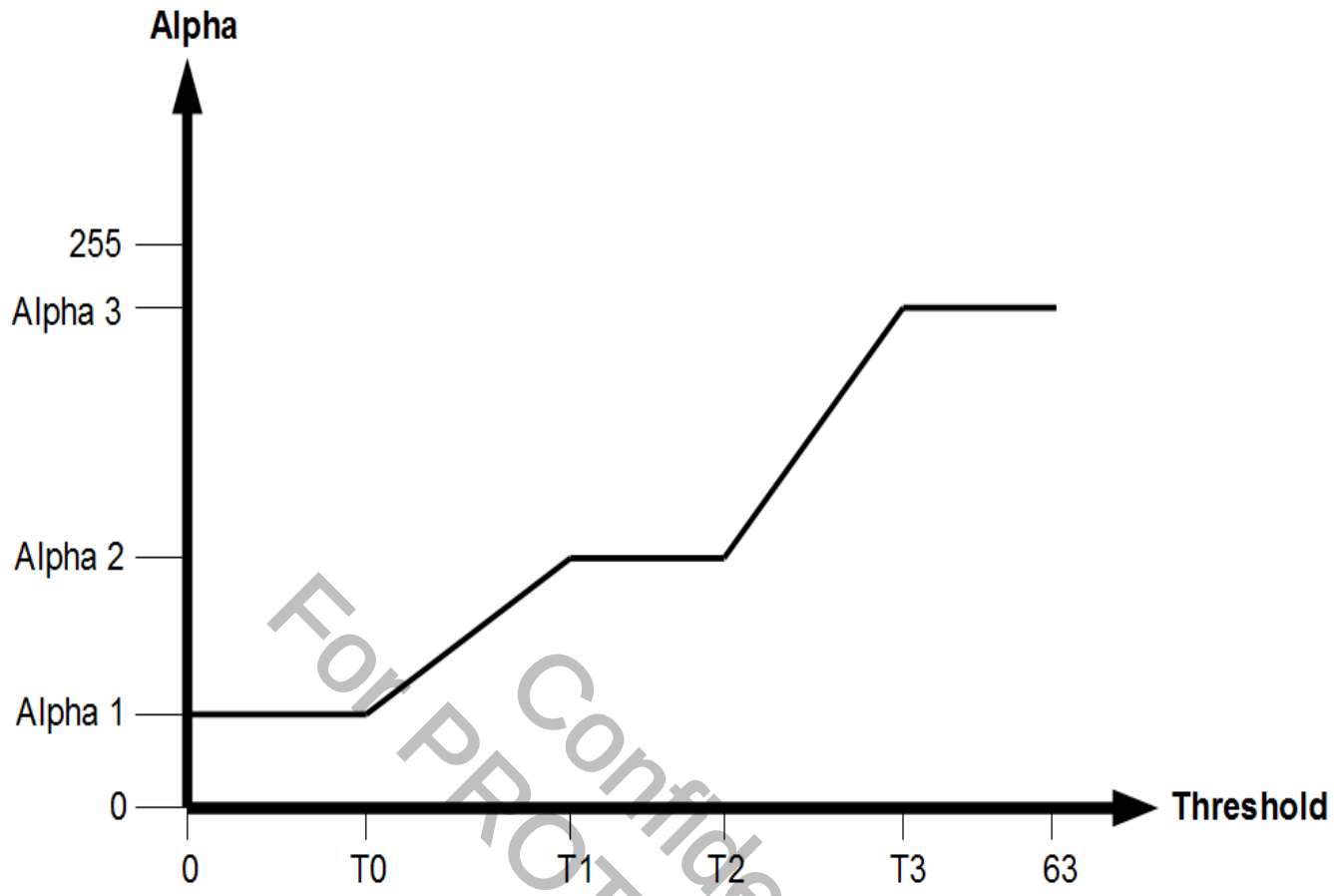

Figure 3-12. MCTF.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_VIDEO_MCTF_INFO_s	***pMctfInfo**	MCTF filter information. Please refer to Section 3.2.72.1 below for details.

*Table 3-197. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetVideoMctf()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure.
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 3-198. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetVideoMctf()**.*

Example:

None

See Also:

None

3.2.72.1 AmbaDSP_ImgSetVideoMctf > AMBA_DSP_IMG_VIDEO_MCTF_INFO_s

Type	Field	Description
UINT8	**Enable**	0: Disable Mctf function 1: Enable Mctf function
UINT16	**YMaxChange**	0-255. Larger value means stronger filter
UINT16	**UMaxChange**	0-255. Larger value means stronger filter
UINT16	**VMaxChange**	0-255. Larger value means stronger filter
UINT8	**WeightingBasedOnLocalMotion**	0-1
UINT8	**Threshold0[4]**	0-63. Larger value means stronger filter
UINT8	**Threshold1[4]**	0-63. Larger value means stronger filter
UINT8	**Threshold2[4]**	0-63. Larger value means stronger filter
UINT8	**Threshold3[4]**	0-63. Larger value means stronger filter
UINT16	**Alpha1[4]**	0-255. Smaller value means stronger filter
UINT16	**Alpha2[4]**	0-255. Smaller value means stronger filter
UINT16	**Alpha3[4]**	0-255. Smaller value means stronger filter

*Table 3-199. Definition of **AMBA_DSP_IMG_VIDEO_MCTF_INFO_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetVideoMctf()**.*

3.2.73 AmbaDSP_ImgGetVideoMctf

API Syntax:

AmbaDSP_ImgGetVideoMctf (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_VIDEO_MCTF_INFO_s * pMctfInfo)

Function Description:

- This API is used to retrieve the motion-compensated temporal filtering (MCTF) settings in video mode.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_VIDEO_MCTF_INFO_s	***pMctfInfo**	Returned MCTF filter information. Please refer to Section 3.2.72.1 for details.

*Table 3-200. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetVideoMctf()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-201. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetVideoMctf()**.*

Example:

None

See Also:

AmbaDSP_ImgSetVideoMctf()

3.2.74 AmbaDSP_ImgSetVideoMctfTemporalAdjust

API Syntax:

AmbaDSP_ImgSetVideoMctfTemporalAdjust (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_VIDEO_MCTF_TEMPORAL_ADJUST_s * pMctfTemporalAdjust)

Function Description:

- This API is used to set the temporal adjust (TA) filtering (MCTF) in video mode. The purpose of TA is that adjust MCTF strength for all pixels in a block based on block motion (score is calculated according to user-defined parameters.).
- A Motion score is computed between frames of current and **MotionDetectionDelay** frames before.
- Motion scores are not only affected by motions but also noises based on different luma levels (DC).
- Use **DcMap** to calibrate motion scores for removing factor of noise levels. The motion score is multiplied by the resulting strength indexed by dc level, which is a 1.7 number (128 -> no change)
- SmoothDetection is for smoothing the motion score with neighboring blocks. It is used for reducing block effect. Increasing SmoothDetection is increasing the smooth strength.
- If max score over multiple frames is less than **FramesCombineThresh**, let TA = 0. It means being determined still area.
- Otherwise, map current score with linear mapping and clamp TA score to [**min**, 3].
- **Sub** and **Mul** are for the linear mapping. **Sub** is for removing background noise and **Mul** is for normalizing the calculated score.
- Finally, threshold/alpha[0-3] are used for different TA score per block. The formula is as follow:

  Filtered pixel = (target * alpha + reference * (256-alpha))/256

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_VIDEO_MCTF_TEMPORAL_ADJUST_s	***pMctfTemporalAdjust**	Temporal Adjust filter information. Please refer to Section 3.2.74.1 for details.

*Table 3-202. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgSetVideoMctfTemporalAdjust()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-203. Returns for Common Filter Image Kernel API **AmbaDSP_ImgSetVideoMctfTemporalAdjust()**.*

Example:

None

See Also:

None

3.2.74.1 AmbaDSP_ImgSetVideoMctfTemporalAdjust > AMBA_DSP_IMG_VIDEO_MCTF_TEMPORAL_ADJUST_s

Type	Field	Description
UINT8	**Enable**	0: Disable Temporal Adjust function 1: Enable Temporal Adjust function
UINT8	**FramesCombineThresh**	0-255
UINT8	**Min**	0-3
UINT8	**MotionDetectionDelay**	1-10
UINT16	**Mul**	0-65535
UINT8	**Sub**	0-255
UINT8	**SmoothDetection**	1-67
UINT8	**MotionDetectionDcMapHigh**	0-255
UINT8	**MotionDetectionDcMapHigh-Delta**	0-7
UINT8	**MotionDetectionDcMapHigh-Strength**	0-255
UINT8	**MotionDetectionDcMapLow**	0-255
UINT8	**MotionDetectionDcMa-pLowDelta**	0-7
UINT8	**MotionDetectionDcMapLow-Strength**	0-255
UINT8	**MotionDetectionDcMapMid-Strength**	0-255

*Table 3-204. Definition of **AMBA_DSP_IMG_VIDEO_MCTF_TEMPORAL_ADJUST_s** for Common Filter Image Kernel API **AmbaDSP_ImgSetVideoMctfTemporalAdjust()***

3.2.75 AmbaDSP_ImgGetVideoMctfTemporalAdjust

API Syntax:

AmbaDSP_ImgGetVideoMctfTemporalAdjust (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_VIDEO_MCTF_TEMPORAL_ADJUST_s * pMctfTemporalAdjust)

Function Description:

- This API is used to retrieve the temporal adjust filtering (MCTF) settings in video mode.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_VIDEO_MCTF_TEMPORAL_ADJUST_s	***pMctfTemporalAdjust**	Returned Temporal Adjust filter information. Please refer to Section 3.2.74.1 for details.

*Table 3-205. Parameters for Common Filter Image Kernel API **AmbaDSP_ImgGetVideoMctfTemporalAdjust()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 3-206. Returns for Common Filter Image Kernel API **AmbaDSP_ImgGetVideoMctfTemporalAdjust()**.*

Example:

None

See Also:

None

4 HISO APIs

4.1 HISO APIs: Overview

This chapter introduces the HighISO filter APIs. A12 HighISO flow chart is as follows:

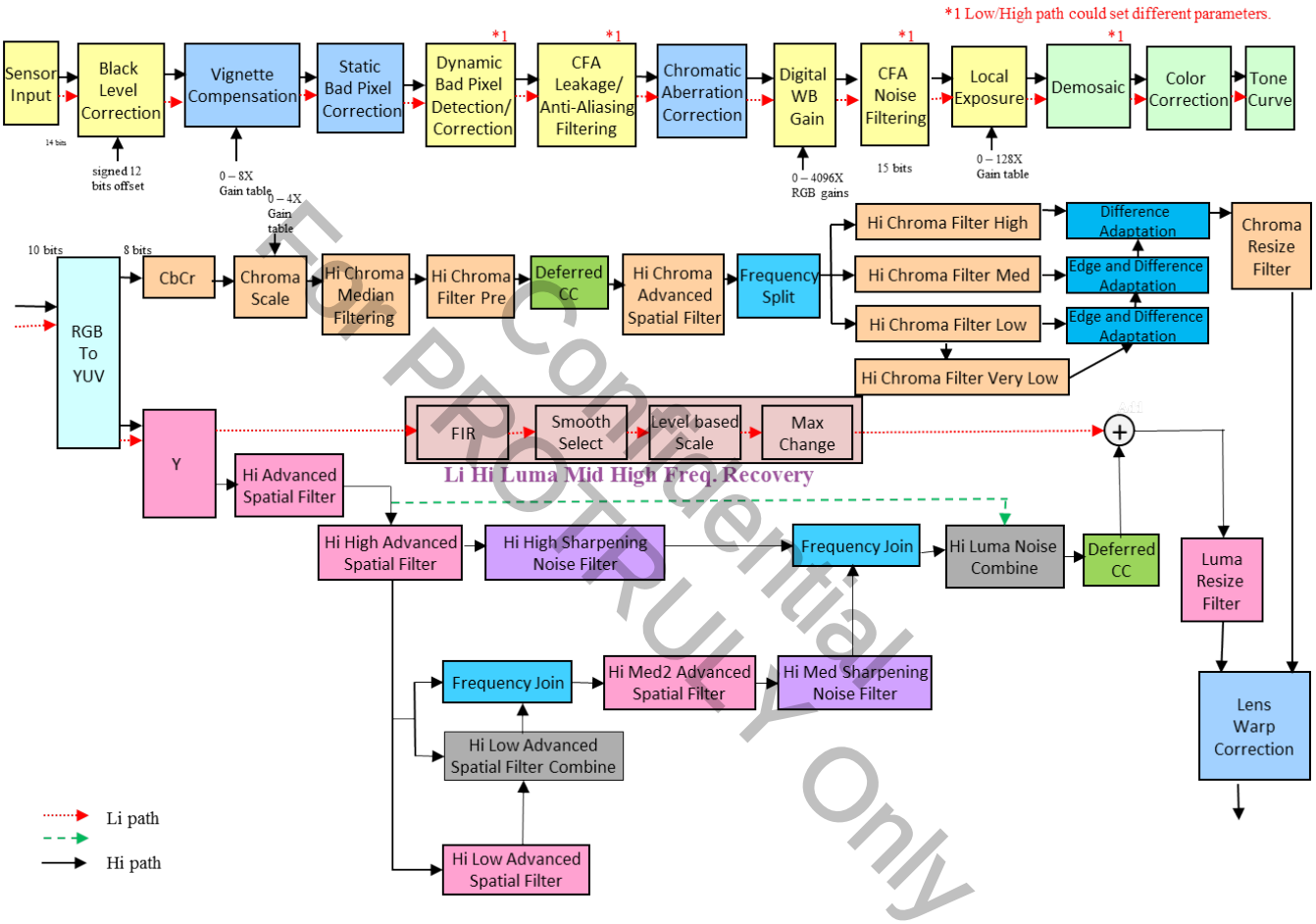

Figure 4-1. A12 HighISO Flow Chart.

4.2 HISO APIs: List of APIs

4.2.1 AmbaDSP_ImgSetHighIsoAntiAliasing

API Syntax:

AmbaDSP_ImgSetHighIsoAntiAliasing (AMBA_DSP_IMG_MODE_CFG_s *pMode, UINT8 Strength)

Function Description:

- This API is used to enable or disable the HISO anti-aliasing filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
UINT 8	**Strength**	HISO anti-aliasing strength: 0: Disable 1 - 3: Weakest to the strongest filtering

*Table 4-1. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoAntiAliasing()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-2. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoAntiAliasing()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoAntiAliasing

4.2.2 AmbaDSP_ImgGetHighIsoAntiAliasing

API Syntax:

AmbaDSP_ImgGetHighIsoAntiAliasing (AMBA_DSP_IMG_MODE_CFG_s *pMode, UINT8 *pStrength)

Function Description:

- This API is used to retrieve HISO anti-aliasing filter settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
UINT 8	***pStrength**	Returns HISO anti-aliasing settings. 0: Disable 1 - 3: Weakest to strongest filtering

*Table 4-3. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoAntiAliasing()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-4. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoAntiAliasing()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoAntiAliasing

4.2.3 AmbaDSP_ImgSetHighIsoCfaLeakageFilter

API Syntax:

AmbaDSP_ImgSetHighIsoCfaLeakageFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CFA_LEAKAGE_FILTER_s *pCfaLeakage)

Function Description:

- This API is used to set the HISO CFA leakage filter. The CFA leakage filter provides model-based red and blue pixel leakage correction on Gr and Gb. The purpose is to prevent a Gr/Gb mismatch from occurring. A linear model is used to correct green pixels that have been corrupted by leakage from neighboring red and blue pixels.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CFA_LEAKAGE_FILTER_s	***pCfaLeakage**	HISO CFA leakage filter information. Please refer to Section 4.2.3.1 for details.

*Table 4-5. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoCfaLeakageFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-6. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoCfaLeakageFilter()**.*

Example:

None

See Also:

None

4.2.3.1 AmbaDSP_ImgSetHighIsoCfaLeakageFilter > AMBA_DSP_IMG_CFA_LEAKAGE_FILTER_s

Type	Field	Description
UINT8	**Enb**	0: Disable 1: Enable
UINT8	**AlphaRr**	GR, corrected = GR + ALPHARR * average(RLEFT, RRIGHT) + ALPHARB * average(BTOP, BBOT). Where ALPHARR and ALPHARB are programmable signed 8-bit values that represent the range [-128/1024, 127/1024] in granularity of 1/1024.
UINT8	**AlphaRb**	
UINT8	**AlphaBr**	GB, corrected = GB + ALPHABB * average(RLEFT, RRIGHT) + ALPHABR * average(BTOP, BBOT). Where ALPHABB and ALPHABR are programmable signed 8-bit values that represent the range [-128/1024, 127/1024] in granularity of 1/1024.
UINT8	**AlphaBb**	
UINT16	**SaturationLevel**	The above equations are applied only for GR and GB pixels that have values less than this saturation level.

*Table 4-7. Definition of **AMBA_DSP_IMG_CFA_LEAKAGE_FILTER_s** for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoCfaLeakageFilter()**.*

4.2.4 AmbaDSP_ImgGetHighIsoCfaLeakageFilter

API Syntax:

AmbaDSP_ImgGetHighIsoCfaLeakageFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CFA_LEAKAGE_FILTER_s *pCfaLeakage)

Function Description:

- This API is used to retrieve the HISO CFA leakage filter settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CFA_LEAKAGE_FILTER_s	***pCfaLeakage**	Returns HISO CFA leakage filter information. Please refer to Section 4.2.3.1 for details.

*Table 4-8. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoCfaLeakageFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-9. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoCfaLeakageFilter()**.*

Example:

None

See Also:

None

4.2.5 AmbaDSP_ImgSetHighIsoDynamicBadPixelCorrection

API Syntax:

AmbaDSP_ImgSetHighIsoDynamicBadPixelCorrection (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_DBP_CORRECTION_s *pDbpCorr)

Function Description:

- This API is used to set values for HISO dynamic bad pixel correction. Dynamic bad pixel detection is aimed at finding pixels that grossly deviate from their eight closest same-color neighboring pixels in the current picture.
- There are two detection modes: first-order detection and second-order detection. First-order detection detects isolated bad pixels while second-order detection is a more aggressive mode which can detect clusters of up to two bad pixels. However, please note that second-order detection has a higher potential for false detection.
- Dynamic bad pixel detection is less reliable than static bad pixel detection, but it does not require calibration and is useful for dealing with temperature- and gain-dependent hot pixels in particular. Bad pixels are corrected by clamping to the nearest same-color neighbors.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_DBP_CORRECTION_s	***pDbpCorr**	Dynamic bad pixel correction information. Please refer to Section 4.2.5.1 for details.

*Table 4-10. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoDynamicBadPixelCorrection()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-11. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoDynamicBadPixelCorrection()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoDynamicBadPixelCorrection

4.2.5.1 AmbaDSP_ImgSetHighIsoDynamicBadPixelCorrection > AMBA_DSP_IMG_DBP_CORRECTION_s

Type	Field	Description
UINT8	**Enb**	0: Disable 1: Hot first-order, dark second-order 2: Hot second-order, dark first-order 3: Hot second-order, dark second-order 4: Hot first-order, dark first-order
UINT8	**HotPixelStrength**	Hot pixel correction strength, 0 - 10.
UINT8	**DarkPixelStrength**	Dark pixel correction strength, 0 - 10.
UINT8	**CorrectionMethod**	0: Normal correction mode 1: Aggressive correction mode: may be useful in high-noise situations.

*Table 4-12. Definition of **AMBA_DSP_IMG_DBP_CORRECTION_s** for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoDynamicBadPixelCorrection()**.*

4.2.6 AmbaDSP_ImgGetHighIsoDynamicBadPixelCorrection

API Syntax:

AmbaDSP_ImgGetHighIsoDynamicBadPixelCorrection (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_DBP_CORRECTION_s *pDbpCorr)

Function Description:

- This API is used to retrieve HISO dynamic bad pixel correction settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_DBP_CORRECTION_s	***pDbpCorr**	HISO dynamic bad pixel correction information. Please refer to Section 4.2.5.1 for details.

*Table 4-13. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoDynamicBadPixelCorrection()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-14. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoDynamicBadPixelCorrection()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoDynamicBadPixelCorrection

4.2.7 AmbaDSP_ImgSetHighIsoCfaNoiseFilter

API Syntax:

AmbaDSP_ImgSetHighIsoCfaNoiseFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CFA_NOISE_FILTER_s *pCfaNoise)

Function Description:

- This API is used to set the HISO CFA domain noise filter, an edge-preserving non-linear filter that improves demosaic performance under noisy conditions. The CFA noise filter processes each Bayer color component separately. The goal is to reduce noise without causing excessive blurring or smearing. Each pixel is classified as normal or fine-detail. A fine-detail picture differs in value from most pixels in its support region. Different filter settings are applied for normal and fine-detail pixels.
- **NoiseLevel** is set to the average noise level expected for the 14-bit sensor input (which is a function of the sensor settings). This value may be experimentally determined at each ISO level by measuring the noise standard-deviation. Internally, the red and blue noise levels are adjusted based on white-balance gains, so the application does not need to change these values based on current white-balance parameters.
- **OriginalBlendStrength** determines how much of the pre-filtered input value is blended into the output. 0 indicates no blending, or maximum noise reduction. Higher levels of blending tend to maintain more high-frequency texture at the expense of noise reduction, and may be more appropriate for low-ISO cases where noise reduction is less important.
- **ExtentRegular** determines the size of the filter support region. Higher values indicate wider support.
- This parameter is in effect when the pixel is classified as normal. A wider support results in more filtering, while a smaller support retains more details.
- **Extent_Fine** determines the size of the filter support region. Higher values indicate wider support. This parameter is in effect when the pixel is classified as fine-detail.
- **Strength_Fine** determines the filtering strength to be used for fine-detail pixels. 0 indicates the same strength as regular pixels, while 256 indicates the maximum strength. A higher strength tends to reduce fine detail, but also removes impulse, or salt-pepper, type noise. It is recommended that a high **strength_fine** be used together with a smaller **extent_fine** to reduce impulse noise without causing excessive blurring.
- **Selectivity** determines the frequency response of the filter. Higher values assign a stronger weight to closer neighboring pixels, while a value of zero assigns the same weight to all pixels in the support region. It is recommended that the selectivity be decreased for noisier inputs (high ISO).

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CFA_NOISE_FILTER_s	***pCfaNoise**	CFA noise filter information. Please refer to Section 4.2.7.1 for details.

*Table 4-15. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoCfaNoiseFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-16. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoCfaNoiseFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoCfaNoiseFilter

4.2.7.1 AmbaDSP_ImgSetHighIsoCfaNoiseFilter > AMBA_DSP_IMG_CFA_NOISE_FILTER_s

Type	Field	Description
UINT8	**Enb**	0: Disable 1: Enable
UINT16	**NoiseLevel[3]**	R, G, B channel maps to array index of 0, 1, 2 respectively. The range is 0 - 8192.
UINT16	**OriginalBlendStr[3]**	R, G, B channel maps to array index of 0, 1, 2 respectively. The range is 0 - 256.
UINT16	**ExtentRegular[3]**	R, G, B channel maps to array index of 0, 1, 2 respectively. The range is 0 - 256.
UINT16	**ExtentFine[3]**	R, G, B channel maps to array index of 0, 1, 2 respectively. The range is 0 - 256.
UINT16	**StrengthFine[3]**	R, G, B channel maps to array index of 0, 1, 2 respectively. The range is 0 - 256.
UINT16	**SelectivityRegular**	The range is 0 - 256.
UINT16	**SelectivityFine**	The range is 0 - 256.

*Table 4-17. Definition of **AMBA_DSP_IMG_CFA_NOISE_FILTER_s** for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoCfaNoiseFilter()**.*

4.2.8 AmbaDSP_ImgGetHighIsoCfaNoiseFilter

API Syntax:

AmbaDSP_ImgGetHighIsoCfaNoiseFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CFA_NOISE_FILTER_s *pCfaNoise)

Function Description:

- This API is used to retrieve the HISO CFA domain noise filter settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CFA_NOISE_FILTER_s	***pCfaNoise**	CFA leakage filter information. Please refer to Section 4.2.7.1 for details.

*Table 4-18. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoCfaNoiseFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-19. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoCfaNoiseFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoCfaNoiseFilter

4.2.9 AmbaDSP_ImgSetHighIsoGbGrMismatch

API Syntax:

AmbaDSP_ImgSetHighIsoGbGrMismatch (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_GBGR_MISMATCH_s *pGbGrMismatch)

Function Description:

- This API is used to set the HISO GbGrMismatch information.

Strong GbGr correction functions as follows. First, it detects whether there is a consistent mismatch in a small area (if narrow_enable = 1) or if there is mismatch in a wide area (WideEnable = 1). If either condition occurs, the center pixels are averaged with the average of the neighboring four green pixels.

Narrow detection is off / on with enable, but otherwise exerts no control.
Wide detection passes if the following is true:

- A measure of mismatch is greater than WideThresh, so increasing WideThresh weakens the filter.
- A measure of how likely the system mismatch is caused by a true signal is less than WideSafety, so increasing wide_safety increases filter strength.
- For a pixel on which GbGr correction is performed, the CFA noise filter is disabled. Refer to the diagram on the following page.

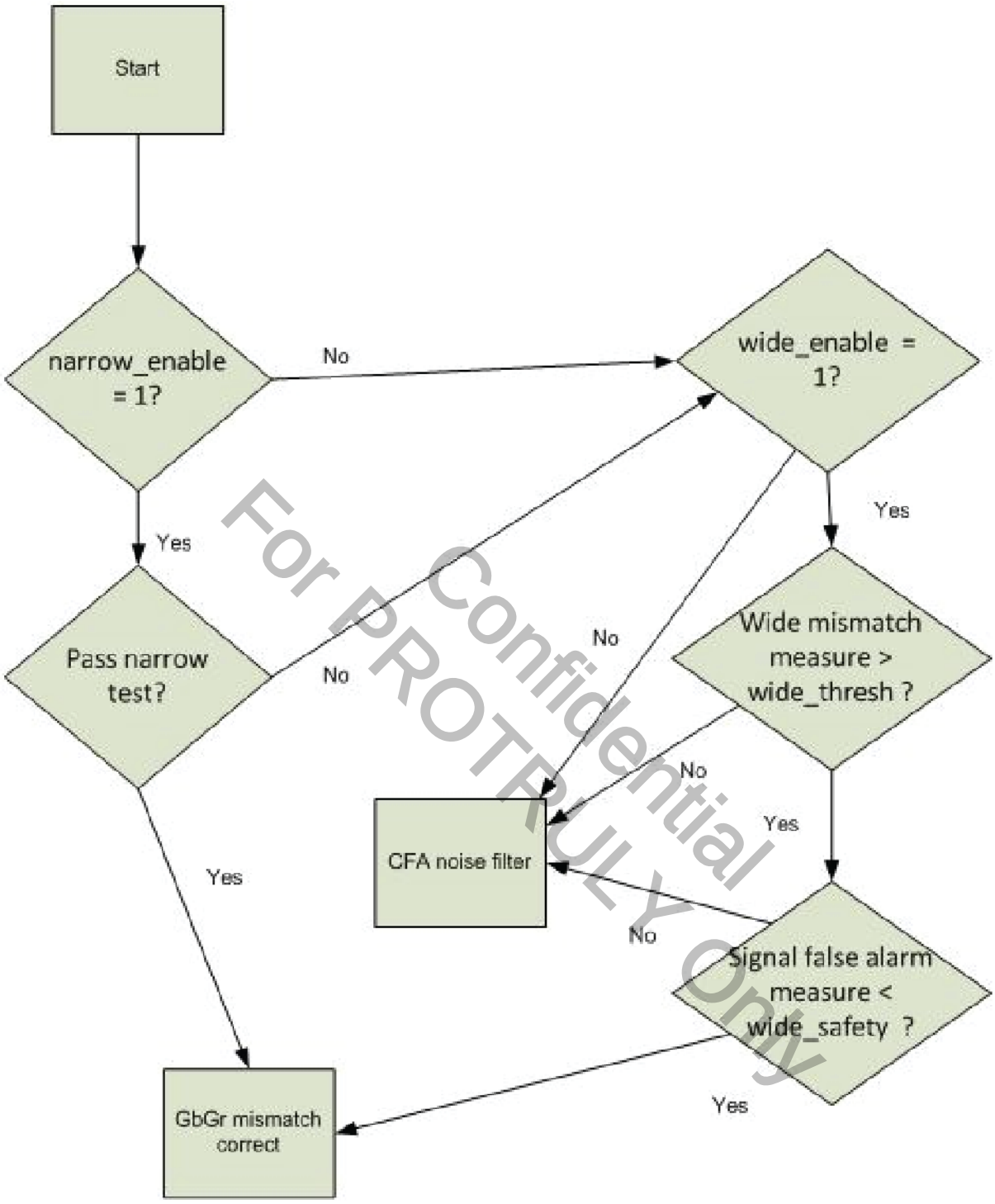

Figure 4-2. GbGr Correction.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_GBGR_MISMATCH_s	***pGbGrMismatch**	HISO GbGrMismatch information. Please refer to Section 4.2.9.1 for details.

*Table 4-20. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoGbGrMismatch()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-21. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoGbGrMismatch()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoGbGrMismatch

4.2.9.1 AmbaDSP_ImgSetHighIsoGbGrMismatch > AMBA_DSP_IMG_GBGR_MISMATCH_s

Type	Field	Description
UINT8	**NarrowEnable**	0: Disable 1: Enable
UINT8	**WideEnable**	0: Disable 1: Enable
UINT16	**WideSafety**	0-256, see description.
UINT16	**WideThresh**	0-256, see description.

*Table 4-22. Definition of **AMBA_DSP_IMG_GBGR_MISMATCH_s** for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoGbGrMismatch()**.*

4.2.10 AmbaDSP_ImgGetHighIsoGbGrMismatch

API Syntax:

AmbaDSP_ImgGetHighIsoGbGrMismatch (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_GBGR_MISMATCH_s *pGbGrMismatch)

Function Description:

- This API is used to retrieve the HISO GbGrMismatch settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_GBGR_MISMATCH_s	***pGbGrMismatch**	HISO GbGrMismatch information. Please refer to Section 4.2.9.1 for details.

*Table 4-23. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoGbGrMismatch()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-24. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoGbGrMismatch()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoGbGrMismatch

4.2.11 AmbaDSP_ImgSetHighIsoDemosaic

API Syntax:

AmbaDSP_ImgSetHighIsoDemosaic (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_DEMOSAIC_s *pDemosaic)

Function Description:

- This API is used to set HISO demosaic settings.
- There is a “gradient” measure used to decide between isotropic vs directional interpolation. And this measure is only directly used for determining green channel’s interpolation. The red and blue component interpolations are dependent-on and follows the green interpolation. When ‘directional’ interpolation has been decided, there is another ‘activity’ measure used to decide between constant-hue versus straight-average interpolation, for the type of directional interpolation.
- So the possible combination of interpolations are:
  - directional with constant hue
  - directional with straight average
  - isotropic (always straight average)
- There are four demosaic tuning registers:

1. grad_noise_thresh:
   - It is what the user have mainly been tuning before to control directional interpolation vs isotropic interpolation based on a gradient measure. When the gradient measure is below “grad_noise_thresh”, isotropic interpolation is chosen to reduce noise. When the gradient measure is above “grad_noise_thresh”,directional interpolation is chosen to avoid zipper artifacts on edges. This register is primarily responsible to tradeoff smoothing versus noodle-noise in noisy flat areas, and zipper artifacts vs correct directional interpolation on edge areas.
   - The other registers that can possibly be tuned all relate in some way to getting better fine detail vs getting more possible speckle artifacts:
2. grad_clip_thresh:
   - It controls when to clip(or bound) the green interpolation to be within its neighboring 4 green pixels’ range, based upon a gradient threhold. If the gradient measure is below this programmed threshold, green gets clipped to the range of its neighbors. If the gradient measure is above the threshold, no clipping occurs. This will basically trade off having higher resolution on high contrast fine diagonal lines vs. possible speckle artifacts.

- The two “activity” registers control whether to use constant hue interpolation vs straight average interpolation based on an “activity” measure. The higher the activity, the more tendency towards constant hue interpolation, and thus clearer detail in fine detailed areas close to neutral gray, but more prone to speckle artifacts near high-saturated-color boundaries.

3. activity_difference_thresh:

- “activity” counts the number of pixel differences in a neighborhood larger than “activity_differ- ence_ thresh”. So larger values of “activity_difference_thresh” will tend to make the “activity” count smaller.

4. activity_thresh:
   - If “activity count” is >= “activity_thresh”, constant hue interpolation is used to achieve finer detail interpolation.
   - If “activity count” is < “activity_thresh”, straight-average (conservative)interpolation is used to achieve smoother and usually less noisy interpolation.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_DEMOSAIC_s	***pDemosaic**	Demosaic information. Please refer to Section 4.2.11.1 for details.

*Table 4-25. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoDemosaic()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-26. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoDemosaic()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoDemosaic

4.2.11.1 AmbaDSP_ImgSetHighIsoDemosaic > AMBA_DSP_IMG_DEMOSAIC_s

Type	Field	Description
UINT16	**ActivityThresh**	0-31. Activity threshold
UINT16	**ActivityDifferenceThresh**	0-16383. Activity difference threshold
UINT16	**GradClipThresh**	0-4095. Gradient clip threshold
UINT16	**GradNoiseThresh**	0-32767. Gradient noise threshold

*Table 4-27. Definition of **AMBA_DSP_IMG_DEMOSAIC_s** for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoDemosaic()**.*

4.2.12 AmbaDSP_ImgGetHighIsoDemosaic

API Syntax:

AmbaDSP_ImgGetHighIsoDemosaic (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_DEMOSAIC_s *pDemosaic)

Function Description:

- This API is used to retrieve HISO demosaic settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_DEMOSAIC_s	***pDemosaic**	Returned demosaic information. Please refer to Section 4.2.11.1 for details.

*Table 4-28. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoDemosaic()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-29. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoDemosaic()**.*

Example

None

See Also:

AmbaDSP_ImgSetHighIsoDemosaic

4.2.13 AmbaDSP_ImgSetHighIsoChromaMedianFilter

API Syntax:

AmbaDSP_ImgSetHighIsoChromaMedianFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CHROMA_MEDIAN_FILTER_s *pChromaMedian)

Function Description:

- This API is used to set the HISO chroma median filter. This filter is useful in reducing false-color artifacts.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_FILTER_s	***pChromaFilter**	Chroma filter information. Please refer to Section 4.2.13.1 for details.

*Table 4-30. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoChromaMedianFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-31. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoChromaMedianFilter()**.*

Example

None

See Also:

AmbaDSP_ImgGetHighIsoChromaMedianFilter

4.2.13.1 AmbaDSP_ImgSetHighIsoChromaMedianFilter > AMBA_DSP_IMG_CHROMA_MEDIAN_FILTER_s

Type	Field	Description
int	**Enb**	0: Disable 1: Enable
UINT16	**CbAdaptiveStrength**	0 – 256. Chroma median filter adaptive strength for Cb/Cr component.
UINT16	**CrAdaptiveStrength**	
UINT8	**CbNonAdaptiveStrength**	0 – 256. Chroma median filter non-adaptive strength for Cb/Cr component.
UINT8	**CrNonAdaptiveStrength**	
UINT16	**CbAdaptiveAmount**	0 – 256
UINT16	**CrAdaptiveAmount**	0 – 256

*Table 4-32. Definition of **AMBA_DSP_IMG_CHROMA_MEDIAN_FILTER_s** for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoChromaMedianFilter()**.*

4.2.14 AmbaDSP_ImgGetHighIsoChromaMedianFilter

API Syntax:

AmbaDSP_ImgGetHighIsoChromaMedianFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CHROMA_MEDIAN_FILTER_s *pChromaMedian)

Function Description:

- This API is used to retrieve the HISO chroma median filter settings. This filter is useful in reducing false- color artifacts.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_FILTER_s	***pChromaMedian**	Chroma filter information. Please refer to Section 4.2.13.1 for details.

*Table 4-33. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoChromaMedianFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-34. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoChromaMedianFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoChromaMedianFilter

4.2.15 AmbaDSP_ImgSetHighIsoDeferColorCorrection

API Syntax:

AmbaDSP_ImgSetHighIsoDeferColorCorrection (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_DEFER_COLOR_CORRECTION_s *pDeferColorCorrection)

Function Description:

- This API is used to set the HISO deferred color correction.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_DEFER_COLOR_CORRECTION_s	***pDeferColorCorrection**	HISO deferred color correction information. Please refer to Section 4.2.15.1 for details.

*Table 4-35. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoDeferColorCorrection()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-36. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoDeferColorCorrection()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoDeferColorCorrection

4.2.15.1 AmbaDSP_ImgSetHighIsoDeferColorCorrection > AMBA_DSP_IMG_DEFER_COLOR_CORRECTION_s

Type	Field	Description
UINT8	**Enable**	0: Disable 1: Enable

*Table 4-37. Definition of **AMBA_DSP_IMG_DEFER_COLOR_CORRECTION_s** for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoDeferColorCorrection()**.*

4.2.16 AmbaDSP_ImgGetHighIsoDeferColorCorrection

API Syntax:

AmbaDSP_ImgGetHighIsoDeferColorCorrection (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_DEFER_COLOR_CORRECTION_s *pDeferColorCorrection)

Function Description:

- This API is used to retrieve HISO deferred color correction information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	**Mode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_DEFER_COLOR_CORRECTION_s	***pDeferColorCorrection**	HISO deferred color correction information. Please refer to Section 4.2.15.1 for details.

*Table 4-38. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoDeferColorCorrection()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success.
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure.

*Table 4-39. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoDeferColorCorrection()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoDeferColorCorrection

4.2.17 AmbaDSP_ImgSetHighIsoHighSharpenNoiseBoth

API Syntax:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseBoth (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SHARPEN_BOTH_s *pSharpenBoth)

Function Description:

- This API is used to set the HISO high frequency sharpen noise filter.

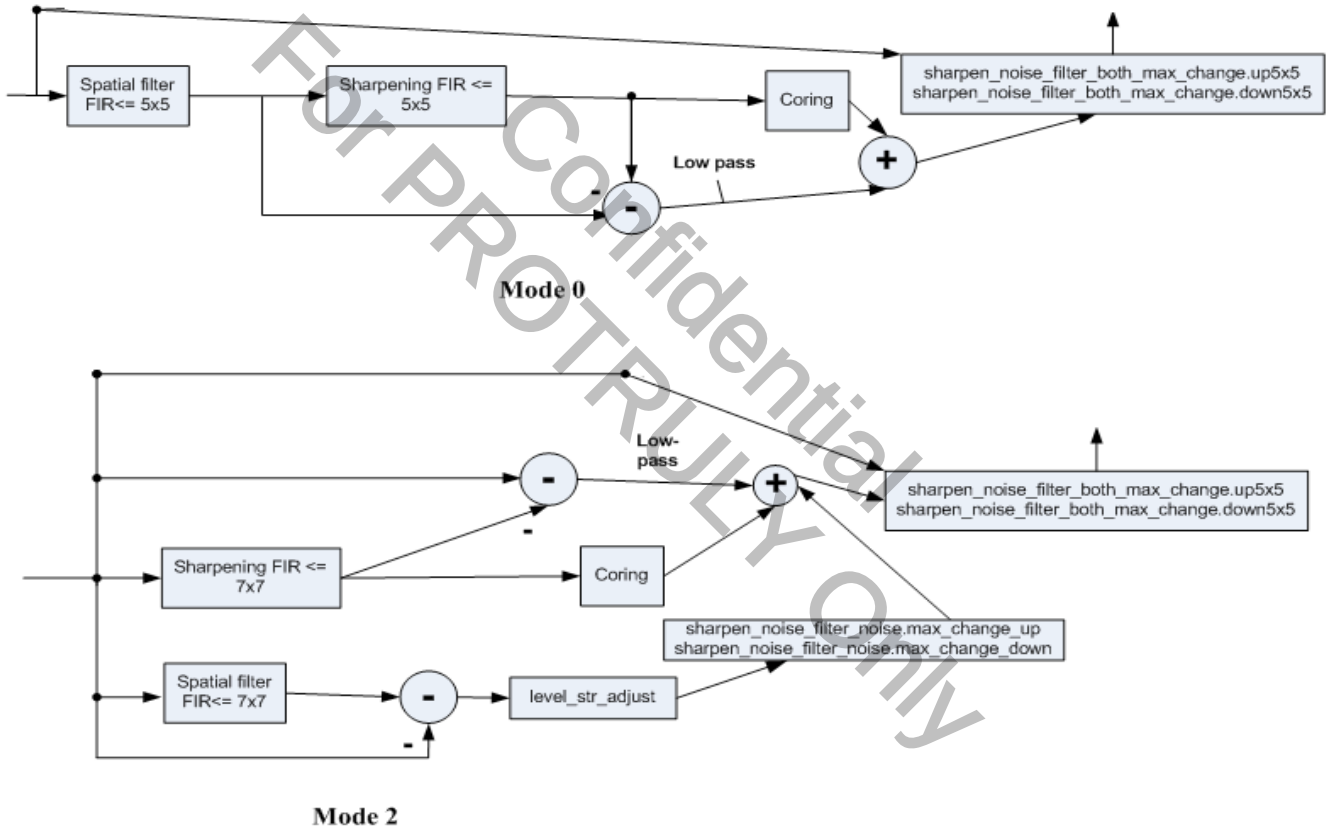

Figure 4-3. Sharpen and Spatial Filter.

- When sharpening is performed, it is performed in conjunction with a spatial filter, which can either come before sharpening (mode 0) or work in parallel with it (mode 2).
- **edge_threshold** is used for spatial filter of both mode 0 and mode 2, and also the directional sharpening's direction/isotropic decision in mode 2.

  The edge detection is controlled by **wide_edge_detect**. Increasing this parameter increases sensitivity and correct edge direction decision for wide high-contrast edges, even when not so close to the edge. Smaller values increase sensitivity and correct edge direction decision for closely spaced lines and finer details.

- Smaller **wide_edge_detect**: Artifacts a few pixels away from high contrast edges.
- Larger **wide_edge_detect**: Blurred out finely spaced lines or not so clear fine details.
- For each pixel, the user determines a wide edge score (W) and narrow edge score (N). edge_score = wide_edge_detect * W + (8- wide_edge_detect) * N. N uses pixel-to-pixel differences, W uses 2-pixel away differences.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_SHARPEN_BOTH_s	***pSharpenBoth**	HISO high frequency sharpen noise both information. Please refer to Section 4.2.17.1 for details.

*Table 4-40. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseBoth()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-41. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseBoth()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseBoth

4.2.17.1 AmbaDSP_ImgSetHighIsoHighSharpenNoiseBoth > AMBA_DSP_IMG_SHARPEN_BOTH_s

Type	Field	Description
UINT8	**Enable**	0 -1
UINT8	**EdgeThresh**	0 - 2047
UINT8	**WideEdgeDetect**	0 - 8
UINT8	**MaxChangeUp5x5**	0-255 Up5x5 and Down5x5 control the maximum change upward and downward, respectively, relative to the 5x5 maximum around the pixel being filtered.
UINT8	**MaxChangeDown5x5**	
UINT8	**Mode**	0 or 2

*Table 4-42. Definition of **AMBA_DSP_IMG_SHARPEN_BOTH_s** for HISO Image Kernel API **AmbaDSP_ImgSetWbGain()**.*

4.2.18 AmbaDSP_ImgGetHighIsoHighSharpenNoiseBoth

API Syntax:

AmbaDSP_ImgGetHighIsoHighSharpenNoiseBoth (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SHARPEN_BOTH_s *pSharpenBoth)

Function Description:

- This API is used to retrieve the HISO high frequency sharpen filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_SHARPEN_BOTH_s	***pSharpenBoth**	Returned HISO high frequency sharpen noise filter information. Please refer to Section 4.2.17.1 for details.

*Table 4-43. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighSharpenNoiseBoth()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-44. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighSharpenNoiseBoth()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseBoth

4.2.19 AmbaDSP_ImgSetHighIsoHighSharpenNoiseNoise

API Syntax:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseNoise (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SHARPEN_NOISE_s *pSharpenNoise)

Function Description:

- This API is used to set the HISO high frequency noise filter of sharpen.
- Mode actually controls both spatial filtering and luma sharpening configuration. When sharpening is performed, it is performed in conjunction with a spatial filter, which can either come before sharpening (mode 0) or work in parallel with it (mode 2).
- The Spatial filter FIR is a 5x5 or 7x7 filter (depending on **1stSharpenNoiseBoth.mode**, 5x5 for mode 0 and 7x7 for mode 2). The format for specifying 5x5 FIRs is the same as for setting 7x7 FIRs, but some parameters are constrained.
- In the mode 2 diagram, the bottom path is for spatial filtering, user can use **noise.max_change** to control its spatial filter output and also to use noise.SpatialFir to change the 7x7 spatial filter FIR.
- Level strength adjust is similar the same named parameters (***level_str_adjust_***) in advanced spatial filter, except that the maximum strength is 16 (not 64).

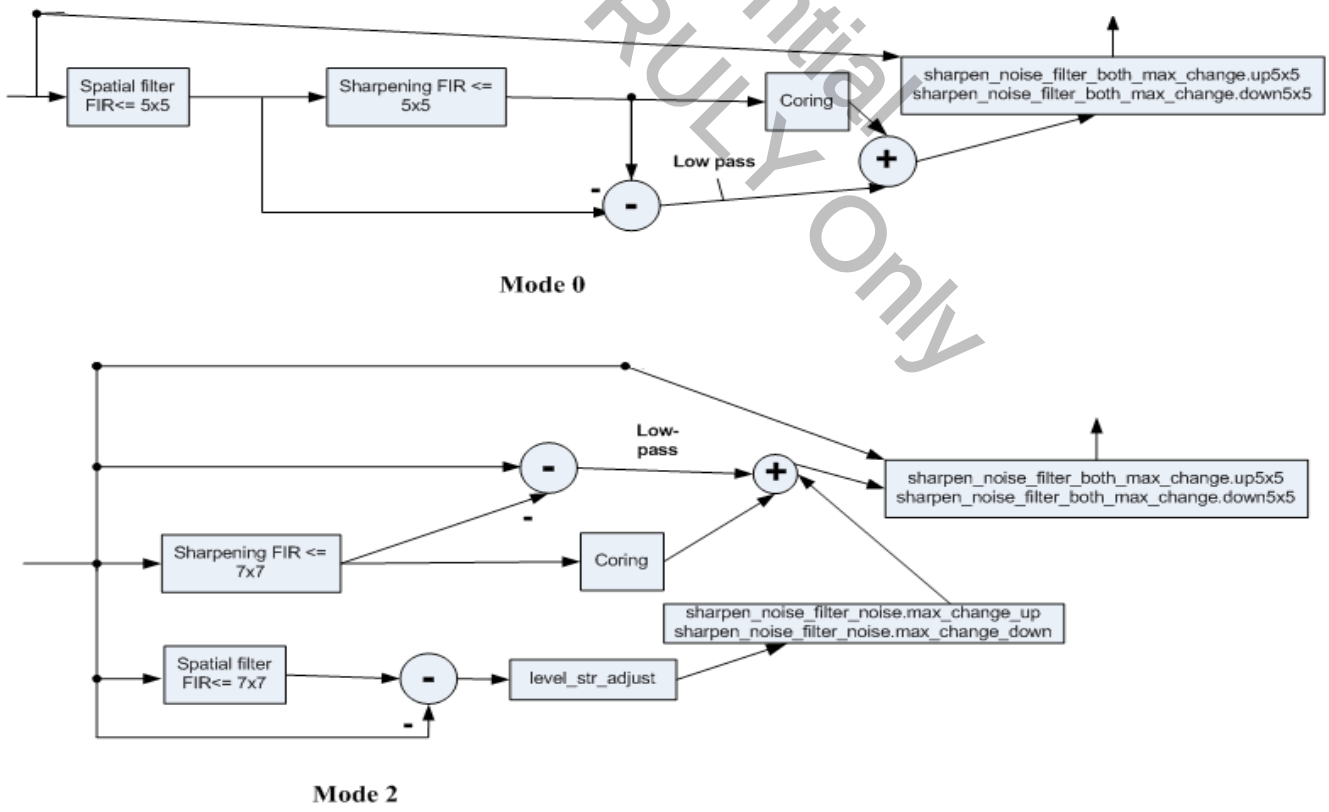

Figure 4-4. Sharpen and Spatial Filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_SHARP-EN_NOISE_s	***pSharpenNoise**	HISO high frequency noise filter information. Please refer to Section 4.2.19.1 for details.

*Table 4-45. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseNoise()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-46. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseNoise()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoHighSharpenNoiseNoise

4.2.19.1 AmbaDSP_ImgSetHighIsoHighSharpenNoiseNoise > AMBA_DSP_IMG_SHARPEN_NOISE_s

Type	Field	Description
UINT8	**MaxChangeUp**	0 - 255
UINT8	**MaxChangeDown**	0 - 255
AMBA_DSP_IMG_FIR_s	**SpatialFir**	Please refer to Section 4.2.19.2 for definition.
AMBA_DSP_IMG_LEVEL_s	**LevelStrAdjust**	Please refer to Section 4.2.19.3 for definition.
UINT8	**LevelStrAdjustNotOT1-Level-Based**	0: Use **T0, T1, AlphaMin, AlphaMax** for level control. 1: Use **LevelStrAdjust** for level control
UINT8	**T0**	0 - 255
UINT8	**T1**	0 - 255
UINT8	**AlphaMin**	0 - 16
UINT8	**AlphaMax**	0 - 16

*Table 4-47. Definition of **AMBA_DSP_IMG_SHARPEN_NOISE_s** for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseNoise()**.*

4.2.19.2 AmbaDSP_ImgSetHighIsoHighSharpenNoiseNoise > AMBA_DSP_IMG_FIR_s

There are 5 ways (specify is from 0 to 4) to determine the strength of these filters. The table below lists the options for “specify”.

Specify	Directions	Params used	Description
0	ISO only	StrengthISO	Single strength deter- mines FIR size.
1	ISO only	Coefs	Only isotropic but fully manual.
2	ISO + dir	StrengthISO Strength-Dir	One strength for isotropic, one strength for directional.
3	ISO + dir	PerDirFirIsoStrengths PerDirFirDirStrengths PerDirFirDirAmounts	For each direction, the user specifies an isotropic strength, a directional strength and the amount to blend isotropic and directional.
4	ISO + dir	Coefs	Fully manual

Figure 4-5. Specify.

When specify 1 is used, the number of coefficient used in Coefs is 10. The FIR coefficients should sum to 0. Taps are specified in units of 1/256, with units as shown above.

0	1	2	3	2	1	0
1	4	5	6	5	4	1
2	5	7	8	7	5	2
3	6	8	9	8	6	3
2	5	7	8	7	5	2
1	4	5	6	5	4	1
0	1	2	3	2	1	0

Coefficient	Bits	Range
9	9	[-256, 255]
8	8	[-128, 127]
7	7	[-64, 63]
4-6	6	[-32, 31]
0-3	5	[-16, 15]

Figure 4-6. FIR Coefficient.

And when specify 4 is used, the number of coefficient is 9*25 = 225.
In specify =2 mode, the strength of the directional filter (step 2) is determined by **dir_strength**, and the strength of the isotropic filter (step 3) is determined by **iso_strength**.
If specify = 3, then the strength for each direction can be determined separately. Also, some directions can use a combination of directional and isotropic filtering.
Specifically, strengths for isotropic (direction 0) and 8 directions (1-8) are set. The direction for each edge is shown below:

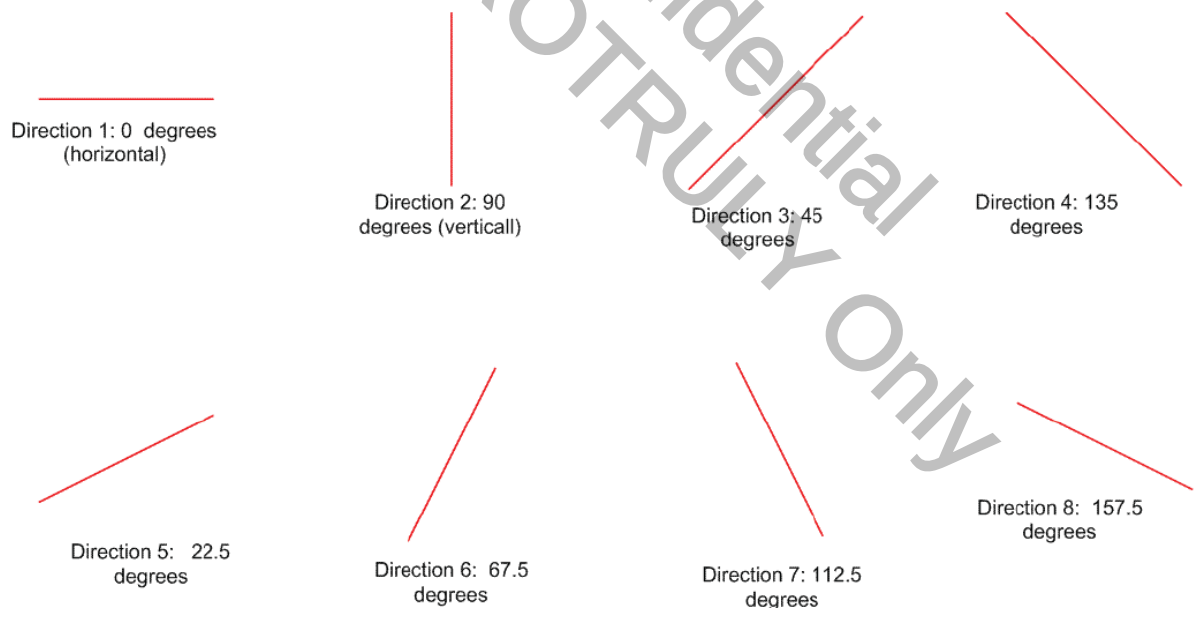

Figure 4-7. Direction for Each Edge.

The directional filter for edge K is a weighted combination of the following:
Directional filter of strength dir_strengths[K], with weight dir_amounts[K]/256.
An isotropic filter of strength iso_strength[K], with weight 1 - dir_amounts[K]/256.

Type	Field	Description
UINT8	**Specify**	0 - 4
UINT16	**PerDirFirIsoStrengths[9]**	0 - 256 (specify = 3)
UINT16	**PerDirFirDirStrengths[9]**	0-256 (specify = 3)
UINT16	**PerDirFirDirAmounts[9]**	0-256 (specify = 3)
UINT8	**Coefs[9][25]**	0-255 (specify = 1, 4)
UINT16	**StrengthIso**	0-256 (specify = 0, 2)
UINT16	**StrengthDir**	0-256 (specify = 2)

*Table 4-48. Definition of **AMBA_DSP_IMG_FIR_s** for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseNoise()**.*

4.2.19.3 AmbaDSP_ImgSetHighIsoHighSharpenNoiseNoise > AMBA_DSP_IMG_SHARPEN_NOISE_s > AMBA_DSP_IMG_LEVEL_s

For each of these sets of controls, the definition of shadow, mid-tone and highlight levels, as well as the transitions between them, is determined by Low, LowDelta, High, and HighDelta. Strength value of 0 is of no effect.

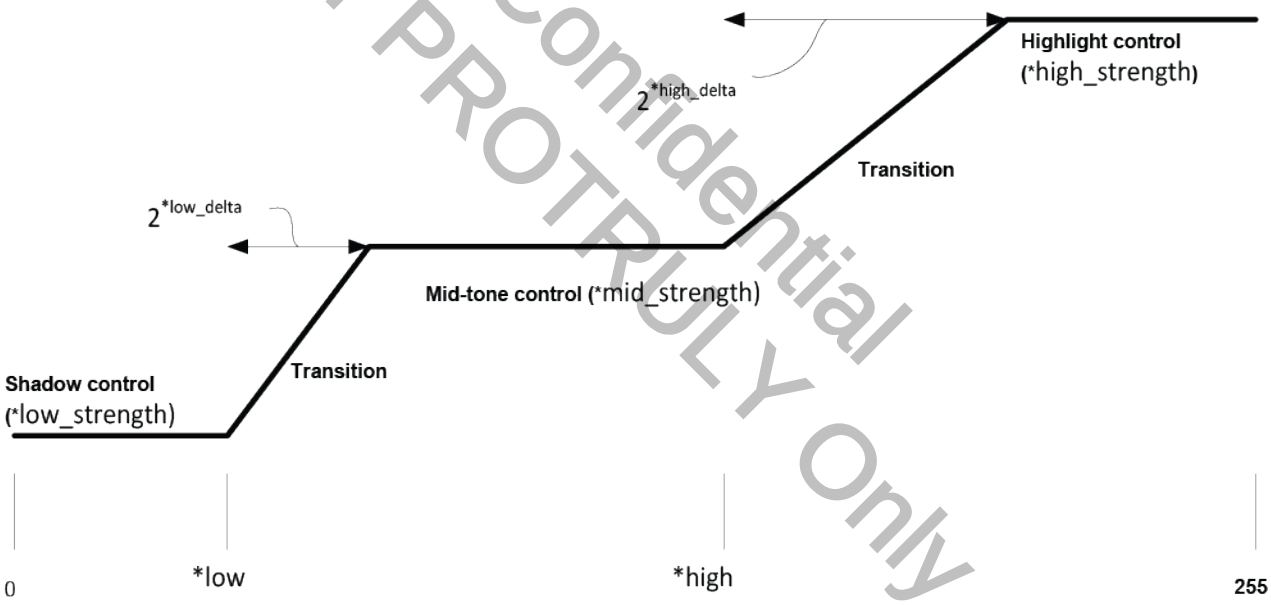

Figure 4-8. Level Control Parameters.

Type	Field	Description
UINT8	**Low**	0 - 255
UINT8	**LowDelta**	0 - 7
UINT8	**LowStrength**	0 - 255 (0-64 for LevelStrAdjust of ASF)
UINT8	**MidStrength**	0 - 255 (0-64 for LevelStrAdjust of ASF)
UINT8	**High**	0 - 255
UINT8	**HighDelta**	0 - 7
UINT8	**HighStrength**	0 - 255 (0-64 for LevelStrAdjust of ASF)

*Table 4-49. Definition of **AMBA_DSP_IMG_LEVEL_s** for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseNoise()**.*

4.2.20 AmbaDSP_ImgGetHighIsoHighSharpenNoiseNoise

API Syntax:

AmbaDSP_ImgGetHighIsoHighSharpenNoiseNoise (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SHARPEN_NOISE_s *pSharpenNoise)

Function Description:

- This API is used to retrieve the HISO high frequency noise filter of Sharpen.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_SHARP-EN_NOISE_s	***pSharpenNoise**	HISO high frequency sharpen noise filter. Please refer to Section 4.2.19.1 for details.

*Table 4-50. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighSharpenNoiseNoise()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-51. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighSharpenNoiseNoise()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseNoise

4.2.21 AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenCoring

API Syntax:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenCoring (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_ IMG_CORING_s * pCoring)

Function Description:

- This API is used to set the coring table of the HISO high frequency sharpen filter. The output of the FIR is used to lookup coring multipliers from the coring table. The multiplier is then multiplied by the FIR output. There are 256 entries in the coring table to cover the full range of FIR outputs, with positive and negative outputs treated separately. Interpolation is used between entries.
- A coring table entry of 0 maps to the smallest FIR output (i.e., the largest negative value). A coring table entry of 128 maps to a FIR output equal to 0. A coring table entry of 255 maps to the largest positive FIR output value. The 256 coring table entries are 2.3 bits. Thus, an entry value of 8 results in unity gain. A value less than 8 attenuates the high-pass signal, which results in noise reduction and reduced sharpness.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CORING_s	***pCoring**	CFA noise filter information. Please refer to Section 4.2.21.1 for details.

*Table 4-52. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenCoring()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-53. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenCoring()**.*

Example:

None

See Also:

None

4.2.21.1 AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenCoring > AMBA_DSP_IMG_CORING_s

Type	Field	Description
UINT8	**Coring[256]**	Unsigned 5 bits. Each entry is controlled by FractionalBits below.
UINT8	**FractionalBits**	1-3. Fractional bit number. 1: Coring entry is 4.1 format. 2: Coring entry is 3.2 format. 3: Coring entry is 2.3 format.

*Table 4-54. Definition of **AMBA_DSP_IMG_CORING_s** for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenCoring()**.*

4.2.22 AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenCoring

API Syntax:

AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenCoring (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CORING_s *pCoring)

Function Description:

- This API is used to retrieve the coring table of the HISO high frequency sharpen filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CORING_s	***pCoring**	HISO high frequency coring table information. Please refer to Section 4.2.21.1 for details.

*Table 4-55. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenCoring()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-56. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenCoring()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenCoring

4.2.23 AmbaDSP_ImgSetLocalExposure

API Syntax:

AmbaDSP_ImgSetLocalExposure (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_FIR_s *pSharpenFir)

Function Description:

- This API is used to set the finite impulse response (FIR) table of the HISO high frequency Sharpen filter while the **AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenCoring** API sets the coring table.
- The FIR is a 5x5 or 7x7 filter (depending on **HighIsoHighSharpenNoiseBoth.mode**, 5x5 for mode 0 and 7x7 for mode 2) which can be used to implement a high-pass filter. The output of the FIR is used to look up the coring table for coring multipliers. The multiplier is then multiplied by the FIR output. The FIR output is also used to subtract from the original input to generate a low-pass signal, which is eventually added to the coring output.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_FIR_s	***pFir**	HISO high frequency FIR filter information. Please refer to Section 4.2.19.2 for details.

*Table 4-57. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetLocalExposure()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-58. Returns for HISO Image Kernel API **AmbaDSP_ImgSetLocalExposure()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenFir

4.2.24 AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenFir

API Syntax:

AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenFir (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_FIR_s *pSharpenFir)

Function Description:

- This API is used to retrieve the finite impulse response (FIR) table of the HISO high frequency Sharpen filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_FIR_s	***pFir**	Returned HISO high frequency FIR filter information. Please refer to Section 4.2.19.2 for details.

*Table 4-59. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenFir()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-60. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenFir()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenFir

4.2.25 AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenCoringIndexScale

API Syntax:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenCoringIndexScale (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s *pLevel)

Function Description:

- This API is used to set the scale factor of coring index in shadow, mid-tone, and high-light (HIGH) areas. The definition of level control diagram is the same as previously specified. Strength value of 16 is of no effect. Refer to the following chart, the strength will shift original coring table index. Str >16 is to use higher edge contract index. Str<16 is to use lower edge contract index.

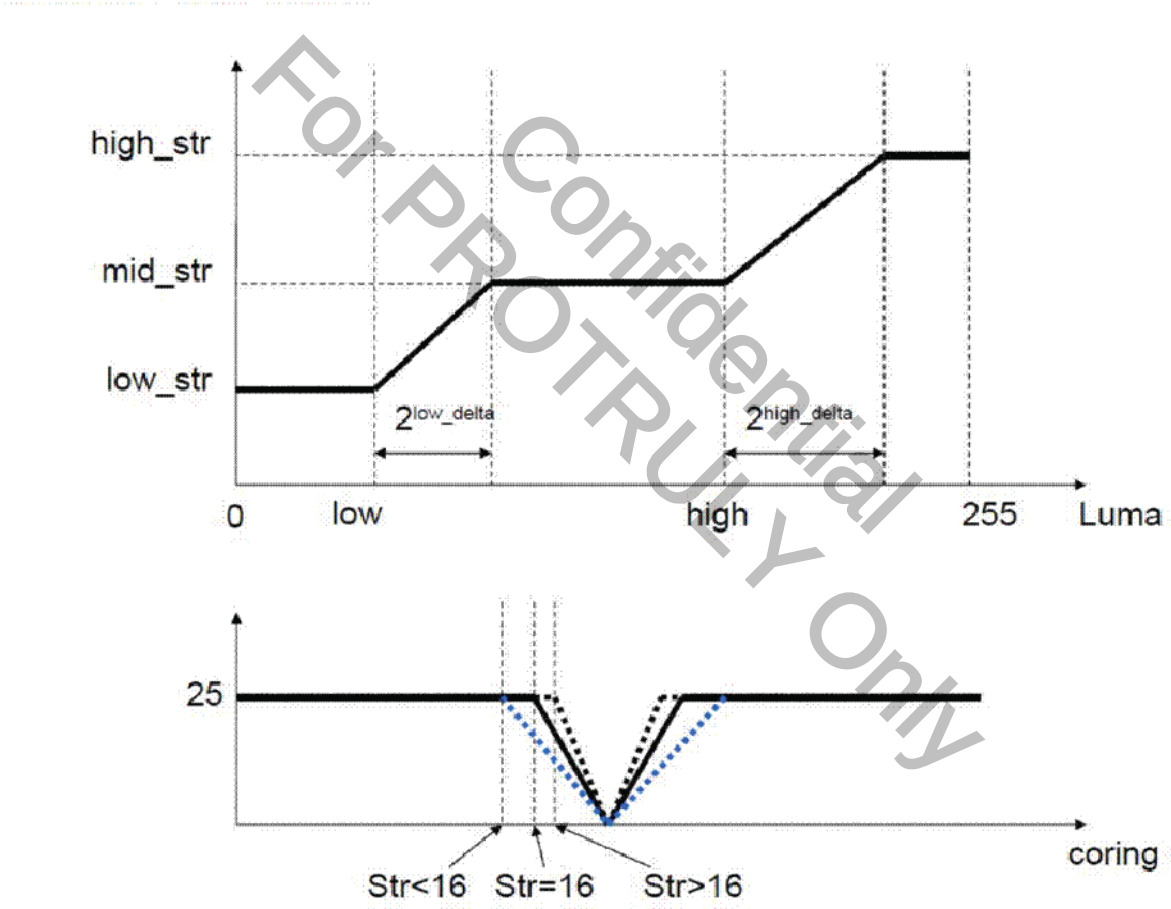

Figure 4-9. Coring Index Scale.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	HISO high frequency coring index scale information. Please refer to Section 4.2.19.3 for details.

*Table 4-61. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenCoringIndexScale()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-62. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenCoringIndexScale()**.*

Example:

After interpolating to arrive at an **overall_level** in [0, 255], the index is scaled from the center of the coring table by **overall_level** / 16. For instance, with an **overall_level** of 16, the index would be positioned at 130.5 (i.e., 2.5 from center). Please refer to the following four examples.

1. **overall_level** = 16. Use 130.5; i.e., the average of entries 130 and 131.
2. **overall_level** = 0. Use entry 128.
3. **overall_level** = 40. Scale 2.5 by 40/16 to retrieve 2.5 * 40 / 16 = 6.25. Use ¾ of entry 134, plus ¼ of entry 135.
4. **overall_level** = 8. Scale 2.5 by 8/16 to retrieve 2.5 * 8 / 16 = 1.25. Use ¾ of entry 129, plus ¼ of entry 130.

See Also:

AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenCoringIndexScale

4.2.26 AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenCoringIndexScale

API Syntax:

AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenCoringIndexScale (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s *pLevel)

Function Description:

- This API is used to retrieve HISO high frequency scale factor of coring index.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	HISO high frequency coring index scale information. Please refer to Section 4.2.19.3 for details.

*Table 4-63. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenCoringIndexScale()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-64. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenCoringIndexScale()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenCoringIndexScale

4.2.27 AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenMinCoringResult

API Syntax:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenMinCoringResult (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s *pLevel)

Function Description:

- This API is used to set the HISO high frequency minimum coring multiplier.

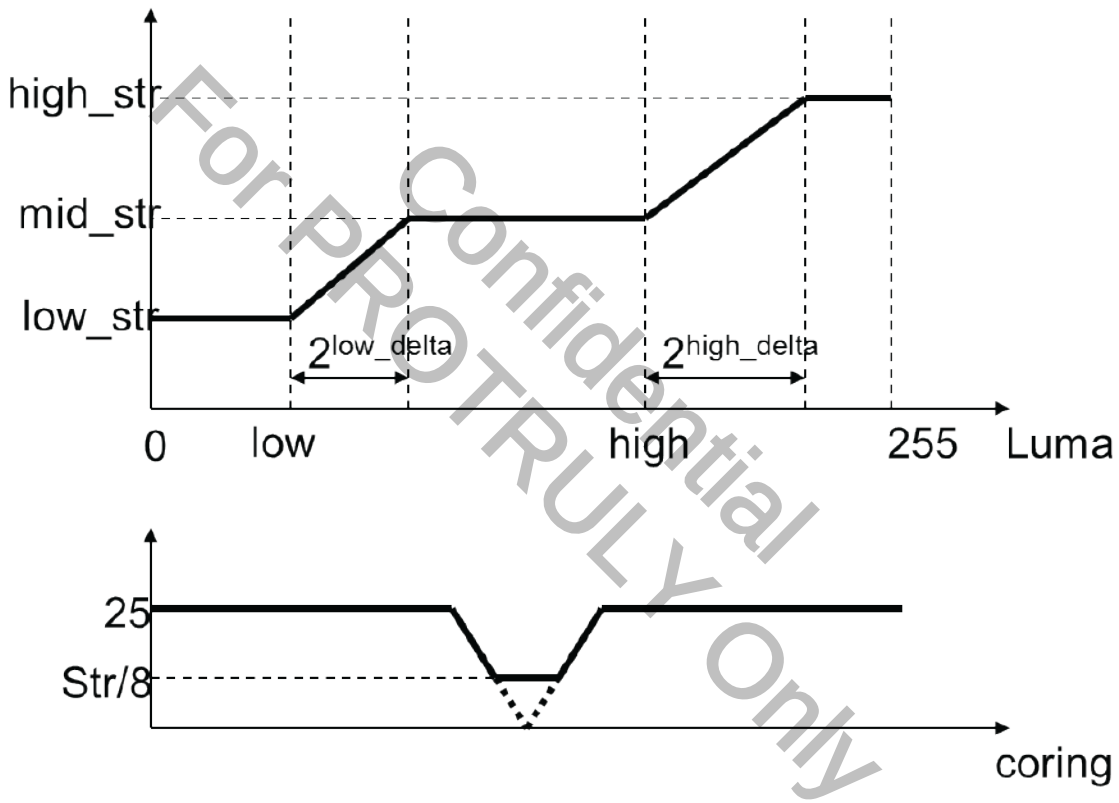

Figure 4-10. Settings in Minimum Coring Multiplier.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	HISO high frequency level minimum information. Please refer to Section 4.2.19.3 for details.

*Table 4-65. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenMinCoringResult()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-66. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenMinCoringResult()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenMinCoringResult

4.2.28 AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenMinCoringResult

API Syntax:

AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenMinCoringResult (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s *pLevel)

Function Description:

- This API is used to retrieve the HISO high frequency minimum coring multiplier.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	HISO high frequency level minimum information. Please refer to Section 4.2.19.3 for details.

*Table 4-67. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenMinCoringResult()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-68. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenMinCoringResult()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenMinCoringResult

4.2.29 AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenScaleCoring

API Syntax:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenScaleCoring (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s *pLevel)

Function Description:

- This API is used to set the HISO high frequency scale factor of the coring table.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	HISO high frequency level minimum information. Please refer to Section 4.2.19.3 for details.

*Table 4-69. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenScaleCoring()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-70. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenScaleCoring()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenScaleCoring

4.2.30 AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenScaleCoring

API Syntax:

AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenScaleCoring (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s *pLevel)

Function Description:

- This API is used to retrieve the HISO high frequency scale factor of the coring table.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	HISO high frequency level minimum information. Please refer to Section 4.2.19.3 for details.

*Table 4-71. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenScaleCoring()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success.
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure.

*Table 4-72. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighSharpenNoiseSharpenScaleCoring()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoHighSharpenNoiseSharpenScaleCoring

4.2.31 AmbaDSP_ImgSetHighIsoMedSharpenNoiseBoth

API Syntax:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseBoth (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SHARPEN_BOTH_s *pSharpenBoth)

Function Description:

- This API is used to set the HISO medium frequency sharpen noise filter. Please refer to Section 4.2.17 for details.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_SHARP-EN_BOTH_s	***pSharpenBoth**	HISO medium frequency sharpen noise both information. Please refer to Section 4.2.17.1 for details.

*Table 4-73. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMedSharpenNoiseBoth()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-74. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMedSharpenNoiseBoth()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoMedSharpenNoiseBoth

4.2.32 AmbaDSP_ImgGetHighIsoMedSharpenNoiseBoth

API Syntax:

AmbaDSP_ImgGetHighIsoMedSharpenNoiseBoth (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SHARPEN_BOTH_s *pSharpenBoth)

Function Description:

- This API is used to retrieve the HISO medium frequency sharpen filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_SHARPEN_BOTH_s	***pSharpenBoth**	Returned HISO medium frequency sharpen noise filter information. Please refer to Section 4.2.17.1 for details.

*Table 4-75. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMedSharpenNoiseBoth()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-76. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMedSharpenNoiseBoth()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseBoth

4.2.33 AmbaDSP_ImgSetHighIsoMedSharpenNoiseNoise

API Syntax:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseNoise (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SHARPEN_NOISE_s *pSharpenNoise)

Function Description:

- This API is used to set the HISO medium frequency noise filter of sharpen. Please refer to Section 4.2.19 for details.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_SHARP-EN_NOISE_s	***pSharpenNoise**	HISO medium frequency sharpen noise filter information. Please refer to Section 4.2.19.1 for details.

*Table 4-77. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMedSharpenNoiseNoise()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-78. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMedSharpenNoiseNoise()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseNoise

4.2.34 AmbaDSP_ImgGetHighIsoMedSharpenNoiseNoise

API Syntax:

AmbaDSP_ImgGetHighIsoMedSharpenNoiseNoise (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SHARPEN_NOISE_s *pSharpenNoise)

Function Description:

- This API is used to retrieve the the HISO medium frequency noise filter of Sharpen.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_SHARP-EN_NOISE_s	***pSharpenNoise**	HISO medium frequency sharpen noise filter information. Please refer to Section 4.2.19.1 for details.

*Table 4-79. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMedSharpenNoiseNoise()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-80. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMedSharpenNoiseNoise()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseNoise

4.2.35 AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenCoring

API Syntax:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenCoring (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CORING_s *pCoring)

Function Description:

- This API is used to set the coring table of the HISO medium frequency sharpen filter. Please refer to Section 4.2.21 for detailed description.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CORING_s	***pCoring**	HISO medium frequency sharpen coring table information. Please refer to Section 4.2.21.1 for details.

*Table 4-81. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenCoring()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-82. Returns for HISO Image Kernel **AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenCoring()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenCoring

4.2.36 AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenCoring

API Syntax:

AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenCoring (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CORING_s *pCoring)

Function Description:

- This API is used to retrieve specific the coring table of the HISO medium frequency sharpen filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	**Mode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CORING_s	***pCoring**	HISO medium frequency coring table information. Please refer to Section 4.2.21.1 for details.

*Table 4-83. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenCoring()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-84. Returns for HISO Image Kernel **AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenCoring()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenCoring

4.2.37 AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenFir

API Syntax:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenFir (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_FIR_s *pSharpenFir)

Function Description:

- This API is used to set the set the finite impulse response (FIR) table of the HISO medium frequency Sharpen filter while the **AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenCoring** API sets the coring table. For detailed description, please refer to Section 4.2.23.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_FIR_s	***pFir**	HISO medium frequency FIR filter information. Please refer to Section 4.2.19.2 for details.

*Table 4-85. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenFir()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-86. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenFir()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenFir

4.2.38 AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenFir

API Syntax:

AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenFir (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_FIR_s *pSharpenFir)

Function Description:

- This API is used to retrieve the the finite impulse response (FIR) table of the HISO medium frequency Sharpen filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_FIR_s	***pFir**	Returned HISO medium frequency FIR filter information. Please refer to Section 4.2.19.2 for details.

*Table 4-87. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenFir()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-88. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenFir()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenFir

4.2.39 AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenCoringIndexScale

API Syntax:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenCoringIndexScale (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s *pLevel)

Function Description:

- This API is used to set the scale factor of coring index in shadow, mid-tone and high-light (HIGH) areas in HISO medium frequency sharpen filter. Please refer to Section 4.2.25 for detailed description.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	Chroma scale information. Please refer to Section 4.2.19.3 for details.

*Table 4-89. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenCoringIndexScale()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-90. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenCoringIndexScale()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenCoringIndexScale

4.2.40 AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenCoringIndexScale

API Syntax:

AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenCoringIndexScale (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s *pLevel)

Function Description:

- This API is used to retrieve the HISO medium frequency scale factor of coring index.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	HISO medium frequency coring index scale information. Please refer to Section 4.2.19.3 for details.

*Table 4-91. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenCoringIndexScale()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-92. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenCoringIndexScale()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenCoringIndexScale

4.2.41 AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenMinCoringResult

API Syntax:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenMinCoringResult (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s *pLevel)

Function Description:

- This API is used to set the HISO medium frequency minimum coring multiplier. Please refer to Section 4.2.27 for details.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	HISO medium frequency level minimum information. Please refer to Section 4.2.19.3 for details.

*Table 4-93. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenMinCoringResult()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-94. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenMinCoringResult()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenMinCoringResult

4.2.42 AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenMinCoringResult

API Syntax:

AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenMinCoringResult (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s *pLevel)

Function Description:

- This API is used to retrieve the HISO medium frequency minimum coring multiplier.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	HISO medium frequency level minimum information. Please refer to Section 4.2.19.3 for details.

*Table 4-95. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenMinCoringResult()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-96. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenMinCoringResult()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenMinCoringResult

4.2.43 AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenScaleCoring

API Syntax:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenScaleCoring (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s *pLevel)

Function Description:

- This API is used to set the HISO medium frequency scale factor of the coring table.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	HISO medium frequency level minimum information. Please refer to Section 4.2.19.3 for details.

*Table 4-97. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenScaleCoring()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP(0x10000000)	Success, but input parameter with invalid value is automatically clamped

*Table 4-98. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenScaleCoring()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenScaleCoring

4.2.44 AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenScaleCoring

API Syntax:

AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenScaleCoring (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_LEVEL_s *pLevel)

Function Description:

- This API is used to retrieve the HISO medium frequency scale factor of the coring table.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_LEVEL_s	***pLevel**	HISO medium frequency level minimum information. Please refer to Section 4.2.19.3 for details.

*Table 4-99. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenScaleCoring()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-100. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMedSharpenNoiseSharpenScaleCoring()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoMedSharpenNoiseSharpenScaleCoring

4.2.45 AmbaDSP_ImgSetHighIsoAdvanceSpatialFilter

API Syntax:

AmbaDSP_ImgSetHighIsoAdvanceSpatialFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_ASF_INFO_s *pAsf)

Function Description:

- This API is used to set the HISO advanced spatial filter. The filters work as follows:
  1. An edge direction and the edge amount are determined.
  2. The sample is filtered parallel to the edge.
  3. The sample is filtered isotropically.
  4. The result of number 2 and number 3 are blended; the more the sample seems to lie on an edge, the more number of 2 is chosen.
  5. The result of number 4 is linearly combined with the input sample based on difference between number 4 and input sample.
  6. The result of number 5 is linearly combined with the input sample based on level, so that the fractional amount of filtering done is based on level. The output of step 6 is: output of number 5 * S / 64 + original sample * (1 – S / 64).
  7. The change in the final output (compared to the input pixel) is limited based on the maximum change amount. The final result is clamped to [original sample – max_change, original sample + max_change].

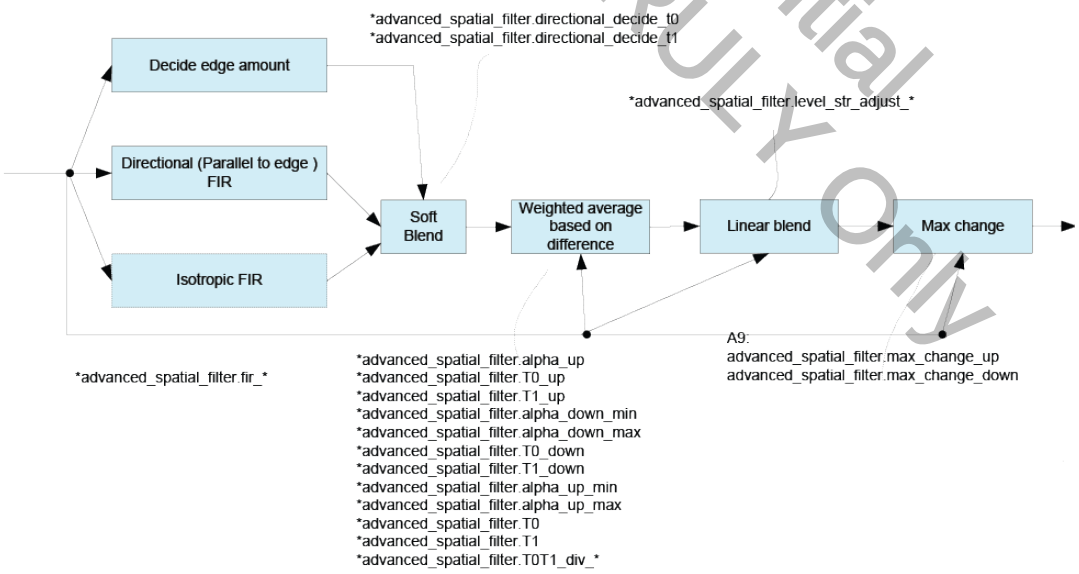

Figure 4-11. Advanced Spatial Filter:

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_ASF_INFO_s	***pAsf**	HISO Advanced spatial filter information. Please refer to Section 4.2.45.1 for details.

*Table 4-101. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoAdvanceSpatialFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success.
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure.
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-102. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoAdvanceSpatialFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoAdvanceSpatialFilter

4.2.45.1 AmbaDSP_ImgSetHighIsoAdvanceSpatialFilter > AMBA_DSP_IMG_ASF_INFO_s

The edge detection is controlled by **wide_edge_detect**. Increasing this parameter increases sensitivity and increases edge direction decisions for wide high-contrast edges, regardless of proximity to the edge. Smaller values enhance sensitivity and increase edge-direction decisions for closely spaced lines and finer details. The effect of this parameter is generally subtle in nature, however, note that the following issues can occur.

- When the value **wide_edge_detect** is small, there may be an increase in artifacts a few pixels away from high-contrast edges. For example, on the dark side of an edge, white spots or lines may be introduced a few pixels away from the edge.

- When the value of **wide_edge_detect** is large, there may be an increase in the blurring of closely spaced lines or a reduction in fine details.

For each pixel, if an edge score is below **directional_decide_t0,** it is filtered isotropically. It is filtered directionally if it rises above **directional_decide_t1**. If an edge score falls between these two values, a combined isotropic and directional filtering approach is used.

To fully disable the filter, set **MaxChangeUp/ MaxChangeDown** = 0.

Type	Field	Description
UINT8	**Enb**	0 - 1
AMBA_DSP_IMG_FIR_s	**Fir**	Please refer to Section 4.2.19.2 for definition.
UINT8	**DirectionalDecideT0**	0 - 255
UINT8	**DirectionalDecideT1**	0 - 255
AMBA_DSP_IMG_FULL_ADAPTATION_s	**Adapt**	Please refer to Section 4.2.45.2 below for definition.
AMBA_DSP_IMG_LEVEL_s	**LevelStrAdjust**	Please refer to Section 4.2.19.3 for definition.
AMBA_DSP_IMG_LEVEL_s	**T0T1Div**	Please refer to Section 4.2.19.3 for definition.
UINT8	**MaxChangeUp**	0 - 255
UINT8	**MaxChangeDown**	0 - 255

*Table 4-103. Definition of **AMBA_DSP_IMG_ASF_INFO_s** for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoAdvanceSpatialFilter()**.*

4.2.45.2 AmbaDSP_ImgSetHighIsoAdvanceSpatialFilter > AMBA_DSP_IMG_ASF_INFO_s > AMBA_DSP_IMG_FULL_ADAPTATION_s

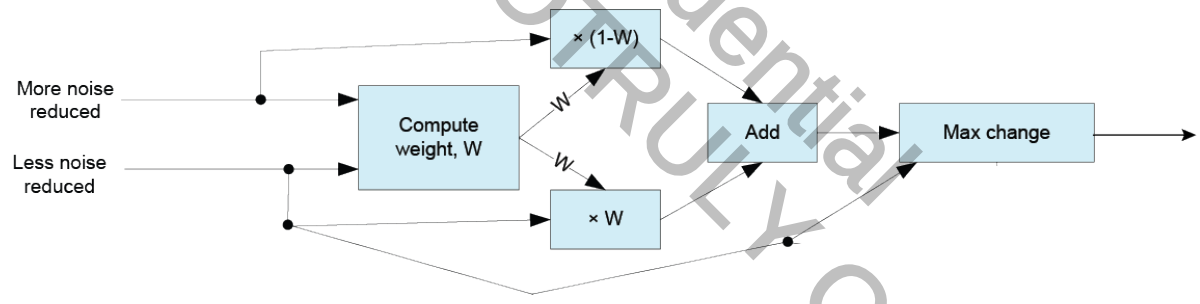

Figure 4-12. Adaptation.

Adaptation takes a weighted average of a less noise reduced signal and more noise reduced signal and then limits the maximum change from the less noise reduced. The underlying concept is that if the noise reduction made a very large change it is more likely due to removing underlying signal removal that should be restored.

The first step is to create a weighted average of the less noise reduced signal and more noise reduced signal. This is computed based on the difference between the two signals, the alpha_min, alpha_max, T0 and T1 parameters.

In addition, T0T1_div can be used to adjust the weight computation based on level. If a T0T1_div level control set is available for filter, then T0 and T1 are divided by the result of level adaptation. This can also be viewed as multiplying result of the difference between the two signals when computing the weight. (I.e., either moving both T0 and T1 in the above charts or scaling the X axis; the effect is identical). For T0T1_div, higher high_strength, low_strength and mid_strength will mean lower T0 and T1; i.e., more likely to choose a higher weight for the less noise reduced, so the user gets less filtering.

Type	Field	Description
UINT8	**AlphaMinUp**	0 - 8
UINT8	**AlphaMaxUp**	0 - 8
UINT8	**T0Up**	0 - 252
UINT8	**T1Up**	2 - 254, even only
UINT8	**AlphaMinDown**	0 - 8
UINT8	**AlphaMaxDown**	0 - 8
UINT8	**T0Down**	0 - 252
UINT8	**T1Down**	2 - 254, even only

*Table 4-104. Definition of **AMBA_DSP_IMG_FULL_ADAPTATION_s** for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoAdvanceSpatialFilter()**.*

4.2.46 AmbaDSP_ImgGetHighIsoAdvanceSpatialFilter

API Syntax:

AmbaDSP_ImgGetHighIsoAdvanceSpatialFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_ASF_INFO_s *pAsf)

Function Description:

- This API is used to retrieve the HISO advanced spatial filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_ASF_INFO_s	***pAsf**	HISO Advanced spatial filter information. Please refer to Section 4.2.45.1 for details.

*Table 4-105. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoAdvanceSpatialFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success.
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure.

*Table 4-106. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoAdvanceSpatialFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoAdvanceSpatialFilter

4.2.47 AmbaDSP_ImgSetHighIsoHighAdvanceSpatialFilter

API Syntax:

AmbaDSP_ImgSetHighIsoHighAdvanceSpatialFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_ASF_INFO_s *pAsf)

Function Description:

- This API is used to set the HISO high frequency advanced spatial filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_ASF_INFO_s	***pAsf**	HISO high frequency adavnced spatial filter information. Please refer to Section 4.2.45.1 for details.

*Table 4-107. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoHighAdvanceSpatialFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-108. Returns for HISO Image kerne API **AmbaDSP_ImgSetHighIsoHighAdvanceSpatialFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoHighAdvanceSpatialFilter

4.2.48 AmbaDSP_ImgGetHighIsoHighAdvanceSpatialFilter

API Syntax:

AmbaDSP_ImgGetHighIsoHighAdvanceSpatialFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_ IMG_ASF_ INFO_s * pAsf)

Function Description:

- This API is used to retrieve the HISO high frequency advanced spatial filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_ASF_INFO_s	***pAsf**	HISO high frequency advanced spatial filter information. Please refer to Section 4.2.45.1 for details.

*Table 4-109. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoHighAdvanceSpatialFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-110. Returns for HISO Image kerne API **AmbaDSP_ImgGetHighIsoHighAdvanceSpatialFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoHighAdvanceSpatialFilter

4.2.49 AmbaDSP_ImgSetHighIsoLowAdvanceSpatialFilter

API Syntax:

AmbaDSP_ImgSetHighIsoLowAdvanceSpatialFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_ASF_INFO_s *pAsf)

Function Description:

- This API is used to set the HISO low frequency advanced spatial filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_ASF_INFO_s	***pAsf**	HISO low frequency advanced spatial filter information. Please refer to Section 4.2.45.1 for details.

*Table 4-111. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoLowAdvanceSpatialFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-112. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoLowAdvanceSpatialFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoLowAdvanceSpatialFilter

4.2.50 AmbaDSP_ImgGetHighIsoLowAdvanceSpatialFilter

API Syntax:

AmbaDSP_ImgGetHighIsoLowAdvanceSpatialFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_ IMG_ASF_ INFO_s * pAsf)

Function Description:

- This API is used to retrieve the HISO low frequency advanced spatial filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_ASF_INFO_s	***pAsf**	HISO low frequency advanced spatial filter information. Please refer to Section 4.2.45.1 for details.

*Table 4-113. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoLowAdvanceSpatialFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-114. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoLowAdvanceSpatialFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoLowAdvanceSpatialFilter

4.2.51 AmbaDSP_ImgSetHighIsoMedAdvanceSpatialFilter

API Syntax:

AmbaDSP_ImgSetHighIsoMedAdvanceSpatialFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_ASF_INFO_s *pAsf)

Function Description:

- This API is used to set the HISO medium frequency advanced spatial filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_ASF_INFO_s	***pAsf**	HISO medium frequency advanced spatial filter information. Please refer to Section 4.2.45.1 for details.

*Table 4-115. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMed1AdvanceSpatialFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-116. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoMed1AdvanceSpatialFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoMed1AdvanceSpatialFilter

4.2.52 AmbaDSP_ImgGetHighIsoMedAdvanceSpatialFilter

API Syntax:

AmbaDSP_ImgGetHighIsoMedAdvanceSpatialFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_ASF_INFO_s *pAsf)

Function Description:

- This API is used to retrieve the retrieve HISO medium frequency advanced spatial filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_ASF_INFO_s	***pAsf**	HISO medium frequency Advanced spatial filter information. Please refer to Section 4.2.45.1 for details.

*Table 4-117. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMed1AdvanceSpatialFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-118. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoMed1AdvanceSpatialFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoMed1AdvanceSpatialFilter

4.2.53 AmbaDSP_ImgSetHighIsoChromaAdvanceSpatialFilter

API Syntax:

AmbaDSP_ImgSetHighIsoChromaAdvanceSpatialFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_ASF_INFO_s * pAsf)

Function Description:

- This API is used to set the HISO advanced spatial filter.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_ASF_INFO_s	***pAsf**	HISO chroma Advanced spatial filter information. Please refer to Section 4.2.45.1 for details.

*Table 4-119. Parameters for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoChromaAdvanceSpatialFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-120. Returns for HISO Image Kernel API **AmbaDSP_ImgSetHighIsoChromaAdvanceSpatialFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgGetHighIsoChromaAdvanceSpatialFilter

4.2.54 AmbaDSP_ImgGetHighIsoChromaAdvanceSpatialFilter

API Syntax:

AmbaDSP_ImgGetHighIsoChromaAdvanceSpatialFilter (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_ASF_INFO_s *pAsf)

Function Description:

- This API is used to retrieve HISO chroma advanced spatial filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_ASF_INFO_s	***pAsf**	HISO chroma Advanced spatial filter information. Please refer to Section 4.2.45.1 for details.

*Table 4-121. Parameters for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoChromaAdvanceSpatialFilter()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure.

*Table 4-122. Returns for HISO Image Kernel API **AmbaDSP_ImgGetHighIsoChromaAdvanceSpatialFilter()**.*

Example:

None

See Also:

AmbaDSP_ImgSetHighIsoChromaAdvanceSpatialFilter

4.2.55 AmbaDSP_ImgHighIsoSetChromaFilterHigh

API Syntax:

AmbaDSP_ImgHighIsoSetChromaFilterHigh (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CHROMA_FILTER_s *pChromaFilterHigh)

Function Description:

- This API is used to set HISO high frequency chroma filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_FILTER_s	***pChromaFilterHigh**	HISO high frequency chroma filter information. Please refer to Section 4.2.55.1 for details.

*Table 4-123. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterHigh()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-124. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterHigh()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoGetChromaFilterHigh

4.2.55.1 AmbaDSP_ImgHighIsoSetChromaFilterHigh > AMBA_DSP_IMG_CHROMA_FILTER_s

Type	Field	Description
UINT8	**Enable**	0 - 1
UINT8	**NoiseLevelCb**	0 - 255
UINT8	**NoiseLevelCr**	0 - 255
UINT16	**OriginalBlendStrengthCb**	0 - 256.

Type	Field	Description
UINT16	**OriginalBlendStrengthCr**	0 - 256.
UINT16	**Radius**	32, 64, 128

*Table 4-125. Definition of **AMBA_DSP_IMG_CHROMA_FILTER_s** for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterHigh()**.*

4.2.56 AmbaDSP_ImgHighIsoGetChromaFilterHigh

API Syntax:

AmbaDSP_ImgHighIsoGetChromaFilterHigh (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_CHROMA_FILTER_s *pChromaFilterHigh)

Function Description:

- This API is used to retrieve the HISO high frequency chroma filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_FILTER_s	***pChromaFilterHigh**	HISO high frequency chroma filter. Please refer to Section 4.2.55.1 for details.

*Table 4-126. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterHigh()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-127. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterHigh()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoSetChromaFilterHigh

4.2.57 AmbaDSP_ImgHighIsoSetChromaFilterLowVeryLow

API Syntax:

AmbaDSP_ImgHighIsoSetChromaFilterLowVeryLow (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_LOW_VERY_LOW_FILTER_s *pChromaFilterLowVeryLow)

Function Description:

- This API is used to set HISO low and very low frequency chroma filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_FILTER_s	***pChromaFilterLowVeryLow**	HISO low and very low frequency chroma filter information. Please refer to Section 4.2.55.1 for details.

*Table 4-128. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterLowVeryLow()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success.
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure.
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-129. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterLowVeryLow()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoGetChromaFilterLowVeryLow

4.2.58 AmbaDSP_ImgHighIsoGetChromaFilterLowVeryLow

API Syntax:

AmbaDSP_ImgHighIsoGetChromaFilterLowVeryLow (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_LOW_VERY_LOW_FILTER_s *pChromaFilterLowVeryLow)

Function Description:

- This API is used to retrieve the HISO low and very low frequency chroma filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_FILTER_s	pChromaFilterLowVeryLow	Returned HISO low and very low frequency chroma filter information. Please refer to Section 4.2.55.1 for details.

*Table 4-130. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterLowVeryLow()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-131. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterLowVeryLow()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoSetChromaFilterLowVeryLow

4.2.59 AmbaDSP_ImgHighIsoSetChromaFilterPre

API Syntax:

AmbaDSP_ImgHighIsoSetChromaFilterPre (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_FILTER_s *pChromaFilterPre)

Function Description:

- This API is used to set HISO pre-chroma filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_HISO_CHROMA_FILTER_s	***pChromaFilterPre**	HISO pre chroma filter information. Please refer to Section 4.2.55.1 for details.

*Table 4-132. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterPre()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-133. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterPre()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoGetChromaFilterPre

4.2.60 AmbaDSP_ImgHighIsoGetChromaFilterPre

API Syntax:

AmbaDSP_ImgHighIsoGetChromaFilterPre (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_FILTER_s *pChromaFilterPre)

Function Description:

- This API is used to retrieve the HISO pre-chroma filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_HISO_CHROMA_FILTER_s	***pChromaFilterPre**	Returned HISO pre-chroma filter information. Please refer to Section 4.2.55.1 for details.

*Table 4-134. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterPre()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-135. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterPre()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoSetChromaFilterPre

4.2.61 AmbaDSP_ImgHighIsoSetChromaFilterMed

API Syntax:

AmbaDSP_ImgHighIsoSetChromaFilterMed (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_FILTER_s *pChromaFilterMed)

Function Description:

- This API is used to set the HISO medium frequency chroma filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_HISO_CHROMA_FILTER_s	***pChromaFilterMed**	HISO medium frequency chroma filter information. Please refer to Section 4.2.55.1 for details.

*Table 4-136. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterMed()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-137. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterMed()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoGetChromaFilterMed

4.2.62 AmbaDSP_ImgHighIsoGetChromaFilterMed

API Syntax:

AmbaDSP_ImgHighIsoGetChromaFilterMed (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_FILTER_s *pChromaFilterMed)

Function Description:

- This API is used to retrieve the HISO medium frequency chroma filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_FILTER_s	***pChromaFilterMed**	Returned HISO medium frequency chroma filter information. Please refer to Section 4.2.55.1 for details.

*Table 4-138. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterMed()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-139. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterMed()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoSetChromaFilterMed

4.2.63 AmbaDSP_ImgHighIsoSetChromaFilterLow

API Syntax:

AmbaDSP_ImgHighIsoSetChromaFilterLow (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_FILTER_s *pChromaFilterLow)

Function Description:

- This API is used to set HISO low frequency chroma filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_FILTER_s	***pChromaFilterLow**	HISO low frequency chroma filter information. Please refer to Section 4.2.55.1 for details.

*Table 4-140. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterLow()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-141. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterLow()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoGetChromaFilterLow

4.2.64 AmbaDSP_ImgHighIsoGetChromaFilterLow

API Syntax:

AmbaDSP_ImgHighIsoGetChromaFilterLow (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_FILTER_s *pChromaFilterLow)

Function Description:

- This API is used to retrieve the HISO low frequency chroma filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_FILTER_s	***pChromaFilterLow**	Returned HISO low frequency chroma filter information. Please refer to Section 4.2.55.1 for details.

*Table 4-142. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterLow()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-143. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterLow()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoSetChromaFilterLow

4.2.65 AmbaDSP_ImgHighIsoSetChromaFilterVeryLow

API Syntax:

AmbaDSP_ImgHighIsoSetChromaFilterVeryLow (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_FILTER_s *pChromaFilterVeryLow)

Function Description:

- This API is used to set the HISO very low frequency chroma filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_FILTER_s	***pChromaFilterVeryLow**	HISO very low frequency chroma filter information.

*Table 4-144. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterVeryLow()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-145. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterVeryLow()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoGetChromaFilterVeryLow

4.2.66 AmbaDSP_ImgHighIsoGetChromaFilterVeryLow

API Syntax:

AmbaDSP_ImgHighIsoGetChromaFilterVeryLow (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_FILTER_s *pChromaFilterVeryLow)

Function Description:

- This API is used to retrieve the HISO very low frequency chroma filter information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_CHROMA_FILTER_s	***pChromaFilterVeryLow**	Returned HISO very low frequency chroma filter information. Please refer to Section 4.2.55.1 for details.

*Table 4-146. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterVeryLow()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-147. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterVeryLow()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoSetChromaFilterVeryLow

4.2.67 AmbaDSP_ImgHighIsoSetLumaNoiseCombine

API Syntax:

AmbaDSP_ImgHighIsoSetLumaNoiseCombine (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_LUMA_FILTER_COMBINE_s *pLumaNoiseCombine)

Function Description:

- This API is used to set the HISO luma noise combine information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_HISO_LUMA_FILTER_COMBINE_s	***pLumaNoiseCombine**	HISO luma noise combine information. Please refer to Section 4.2.67.1 for details.

*Table 4-148. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetLumaNoiseCombine()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-149. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetLumaNoiseCombine()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoGetLumaNoiseCombine

4.2.67.1 AmbaDSP_ImgHighIsoSetLumaNoiseCombine > AMBA_DSP_IMG_HISO_LUMA_FILTER_COMBINE_s

Type	Field	Description
UINT8	**T0**	0:63
UINT8	**T1**	0:63
UINT8	**AlphaMax**	0:255
UINT8	**AlphaMin**	0:255
UINT8	**MaxChange**	0:255

*Table 4-150. Definition of **AMBA_DSP_IMG_HISO_LUMA_FILTER_COMBINE_s** for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetLumaNoiseCombine()**.*

4.2.68 AmbaDSP_ImgHighIsoGetLumaNoiseCombine

API Syntax:

AmbaDSP_ImgHighIsoGetLumaNoiseCombine (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_LUMA_FILTER_COMBINE_s *pLumaNoiseCombine)

Function Description:

- This API is used to set the HISO luma noise combine information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to section 2.2.1.1 for details.
AMBA_DSP_IMG_HISO_LUMA_FILTER_COMBINE_s	***pLumaNoiseCombine**	Luma noise combine information. Please refer to Section 4.2.67.1 for details.

*Table 4-151. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetLumaNoiseCombine()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-152. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetLumaNoiseCombine()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoSetLumaNoiseCombine

4.2.69 AmbaDSP_ImgHighIsoSetLowASFCombine

API Syntax:

AmbaDSP_ImgHighIsoSetLowASFCombine (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_LUMA_FILTER_COMBINE_s *pLowASFCombine)

Function Description:

- This API is used to set the HISO Low ASF combine information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_HISO_LUMA_FILTER_COMBINE_s	***pLowASFCombine**	HISO luma Low ASF combine information. Please refer to Section 4.2.67.1 for details.

*Table 4-153. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetLowASFCombine()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-154. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetLowASFCombine()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoGetLowASFCombine

4.2.70 AmbaDSP_ImgHighIsoGetLowASFCombine

API Syntax:

AmbaDSP_ImgHighIsoGetLowASFCombine (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_LUMA_FILTER_COMBINE_s *pLowASFCombine)

Function Description:

- This API is used to retrieve the HISO Low ASF combine information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_HISO_LUMA_FILTER_COMBINE_s	***pLowASFCombine**	HISO Low ASF combine information. Please refer to Section 4.2.67.1 for details.

*Table 4-155. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetLowASFCombine()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-156. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetLowASFCombine()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoSetLowASFCombine

4.2.71 AmbaDSP_ImgHighIsoSetChromaFilterMedCombine

API Syntax:

AmbaDSP_ImgHighIsoSetChromaFilterMedCombine (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_FILTER_COMBINE_s *pChromaFilterMedCombine)

Function Description:

- This API is used to set the HISO chroma filter medium frequency combine information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for definition.
AMBA_DSP_IMG_HISO_CHROMA_FILTER_COMBINE_s	***pChromaFilterMedCombine**	HISO chroma filter medium frequency combine information. Please refer to Section 4.2.71.1 for details.

*Table 4-157. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterMedCombine()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-158. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterMedCombine()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoGetChromaFilterMedCombine

4.2.71.1 AmbaDSP_ImgHighIsoSetChromaFilterMedCombine > AMBA_DSP_IMG_HISO_CHROMA_FILTER_COMBINE_s

Type	Field	Description
UINT8	**T0Cb**	0:63
UINT8	**T0Cr**	0:63
UINT8	**T1Cb**	0:63
UINT8	**T1Cr**	0:63
UINT8	**AlphaMaxCb**	0:255
UINT8	**AlphaMaxCr**	0:255
UINT8	**AlphaMinCb**	0:255
UINT8	**AlphaMinCr**	0:255
UINT8	**MaxChangeCb**	0:255
UINT8	**MaxChangeCr**	0:255

*Table 4-159. Definition of **AMBA_DSP_IMG_HISO_CHROMA_FILTER_COMBINE_s** for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterMedCombine()**.*

4.2.72 AmbaDSP_ImgHighIsoGetChromaFilterMedCombine

API Syntax:

AmbaDSP_ImgHighIsoGetChromaFilterMedCombine (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_FILTER_COMBINE_s *pChromaFilterMedCombine)

Function Description:

- This API is used to retrieve the HISO chroma filter medium frequency combine information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_HISO_CHROMA_FILTER_COMBINE_s	***pChromaFilterMedCombine**	Returned HISO chroma filter medium frequency combine information. Please refer to Section 4.2.71.1 for details.

*Table 4-160. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterMedCombine()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-161. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterMedCombine()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoSetChromaFilterMedCombine

4.2.73 AmbaDSP_ImgHighIsoSetChromaFilterLowCombine

API Syntax:

AmbaDSP_ImgHighIsoSetChromaFilterLowCombine (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_FILTER_COMBINE_s *pChromaFilterLowCombine)

Function Description:

- This API is used to set the HISO chroma filter low frequency combine information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_HISO_CHROMA_FILTER_COMBINE_s	***pChromaFilterLowCombine**	HISO chroma filter low frequency combine information. Please refer to Section 4.2.71.1 for details.

*Table 4-162. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterLowCombine()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-163. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterLowCombine()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoGetChromaFilterLowCombine

4.2.74 AmbaDSP_ImgHighIsoSetChromaFilterLowCombine

API Syntax:

AmbaDSP_ImgHighIsoSetChromaFilterLowCombine (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_FILTER_COMBINE_s *pChromaFilterLowCombine)

Function Description:

- This API is used to retrieve the HISO chroma filter low frequency combine information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_HISO_CHROMA_FILTER_COMBINE_s	***pChromaFilterLowCombine**	Returned HISO chroma filter low frequency combine information. Please refer to Section 4.2.71.1 for details.

*Table 4-164. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterLowCombine()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-165. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterLowCombine()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoGetChromaFilterLowCombine

4.2.75 AmbaDSP_ImgHighIsoSetChromaFilterVeryLowCombine

API Syntax:

AmbaDSP_ImgHighIsoSetChromaFilter VeryLowCombine (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_FILTER_COMBINE_s *pChromaFilterVeryLowCombine)

Function Description:

- This API is used to retrieve the HISO chroma filter very low frequency combine information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_HISO_CHROMA_FILTER_COMBINE_s	***pChromaFilterVeryLowCombine**	Returned HISO chroma filter very low frequency combine information. Please refer to Section 4.2.71.1 for details.

*Table 4-166. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterVeryLowCombine()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-167. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetChromaFilterVeryLowCombine()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoGetChromaFilteVeryLowCombine

4.2.76 AmbaDSP_ImgHighIsoGetChromaFilterVeryLowCombine

API Syntax:

AmbaDSP_ImgHighIsoGetChromaFilterVeryLowCombine (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_CHROMA_FILTER_COMBINE_s *pChromaFilterVeryLowCombine)

Function Description:

- This API is used to retrieve the HISO chroma filter very low frequency combine information.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_HISO_CHROMA_FILTER_COMBINE_s	***pChromaFilterVeryLowCombine**	Returned HISO chroma filter very low frequency combine information. Please refer to Section 4.2.71.1 for details.

*Table 4-168. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterVeryLowCombine()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-169. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetChromaFilterVeryLowCombine()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoSetChromaFilteVeryLowCombine

4.2.77 AmbaDSP_ImgHighIsoSetHighIsoFreqRecover

API Syntax:

AmbaDSP_ImgHighIsoSetHighIsoFreqRecover (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_FREQ_RECOVER_s *pHighIsoFreqRecover)

Function Description:

- This API is used to set the HISO frequency recover settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_HISO_FREQ_RECOVER_s	***pHighIsoFreqRecover**	HISO frequency recover settings. Please refer to Section 4.2.77.1 for details.

*Table 4-170. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetHighIsoFreqRecover()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure
AMBA_DSP_IMG_RVAL_CLAMP (0x10000000)	Success, but input parameter with invalid value is automatically clamped.

*Table 4-171. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetHighIsoFreqRecover()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoGetHighIsoFreqRecover

4.2.77.1 AmbaDSP_ImgHighIsoSetHighIsoFreqRecover > AMBA_DSP_IMG_HISO_FREQ_RECOVER_s

Type	Field	Description
AMBA_DSP_IMG_FIR_s	**Fir**	FIR filter information. Please refer to Section 4.2.19.2 for more details
UINT8	**SmoothSelect[256]**	0:31
UINT8	**MaxDown**	0:255
UINT8	**MaxUp**	0:255
AMBA_DSP_IMG_LEVEL_s	**Level**	Level information. Please refer to Section 4.2.19.3 for details.

*Table 4-172. Definition of **AMBA_DSP_IMG_HISO_FREQ_RECOVER_s** for HISO Image Kernel API **AmbaDSP_ImgHighIsoSetHighIsoFreqRecover()**.*

4.2.78 AmbaDSP_ImgHighIsoGetHighIsoFreqRecover

API Syntax:

AmbaDSP_ImgHighIsoGetHighIsoFreqRecover (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_HISO_FREQ_RECOVER_s *pHighIsoFreqRecover)

Function Description:

- This API is used to retrieve the HISO frequency recover settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for details.
AMBA_DSP_IMG_HISO_FREQ_RECOVER_s	***pHighIsoFreqRecover**	Returned HISO frequency recover settings. Please refer to Section 4.2.77.1 for details.

*Table 4-173. Parameters for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetHighIsoFreqRecover()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 4-174. Returns for HISO Image Kernel API **AmbaDSP_ImgHighIsoGetHighIsoFreqRecover()**.*

Example:

None

See Also:

AmbaDSP_ImgHighIsoSetHighIsoFreqRecover

5 Utility API

5.1 Utility: Overview

This chapter introduces the Utility API. The Utility API registers are used to initialize flow control, still-capture flow control, and debugging flow control.

5.2 Utility: List of Functions

5.2.1 AmbaDSP_ImgInitArch

API Syntax:

AmbaDSP_ImgInitArch (AMBA_DSP_IMG_ARCH_INFO_s *pArchInfo)

Function Description:

- This function is used to initialize the Image Kernel architecture.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_ARCH_INFO_s	***pArchInfo**	Image Kernel architecture initialization information. Please refer to Section 5.2.1.1 below for more details.

*Table 5-1. Parameters for Utility Image Kernel API **AmbaDSP_ImgInitArch()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 5-2. Returns for Utility Image Kernel API **AmbaDSP_ImgInitArch()**.*

Example:

None

See Also:

None

5.2.1.1 AmbaDSP_ImgInitArch > AMBA_DSP_IMG_ARCH_INFO_s

Type	Field	Description
UINT8	***pWorkBuf**	Image Kernel working buffer address
UINT32	**BufSize**	Image Kernel working buffer size. For a single video pipeline with 1 `context`, and a single still pipeline with 1 `context` and 1 `config`, the required working buffer size is 509212 Bytes.
UINT32	**PipeNum**	Pipeline number. Currently the API supports the video pipeline, still pipeline, and decode pipeline; therefore, the maximum value is 3.
AMBA_DSP_IMG_PIPE_INFO_s	***pPipeInfo[8]**	Pipeline information pointer. Please refer to Section 5.2.1.2 below for more details.

*Table 5-3. Definition of **AMBA_DSP_IMG_ARCH_INFO_s** for Utility Image Kernel API **AmbaDSP_ImgInitArch()**.*

5.2.1.2 AmbaDSP_ImgInitArch > AMBA_DSP_IMG_PIPE_INFO_s

Type	Field	Description
AMBA_DSP_IMG_PIPE_e	**Pipe**	Pipeline type
UINT8	**CtxBufNum**	`Context` buffer number.
UINT8	**CfgBufNum**	`Config` buffer number.

*Table 5-4. Definition of **AMBA_DSP_IMG_PIPE_INFO_s** for Utility Image Kernel API **AmbaDSP_ImgInitArch()**.*

5.2.2 AmbaDSP_ImgInitCtx

API Syntax:

AmbaDSP_ImgInitCtx (UINT8 InitMode, UINT8 DefblcEnb, AMBA_DSP_IMG_CTX_INFO_s *pDestCtx, AMBA_DSP_IMG_CTX_INFO_s *pSrcCtx)

Function Description:

- This function is used to initialize the `context`. Three `init` modes are supported, **HARD_RESET**, **SOFT_RESET**, and **CLONE** modes.
  - **HARD_RESET (0)** allows the user to reset `context` settings, including assigning initial values to all data.
  - **CLONE (1)** allows the user to clone certain `context` settings.
  - **SOFT_RESET(2)** allows the user to enable an `init` flag in the Image Kernel, and will resend the command to the DSP again even if the settings remain the same.
- **DefblcEnb** flag is used in **HARD_RESET** mode only.

Parameters:

Type	Parameter	Description
UINT8	**InitMode**	0: Hard Reset. All data in the `context` will be reset. 1: Clone from **SrcCtx** to **DestCtx**. 2: Soft Reset. An `init` flag will be set; commands will be issued even the settings are the same.
UINT8	**DefblcEnb**	0: Disable deferred black level. 1: Enable deferred black level. Note that this field only is available under Hard Reset mode.
AMBA_DSP_IMG_CTX_INFO_s	***pDestCtx**	Destination `context` pointer. In Hard Reset and Soft Reset modes, this represents the target `context` that is to be reset. In Clone mode, this represents the target `context` for the cloned source settings. Please refer to Section 5.2.2.1 below for more details.
AMBA_DSP_IMG_CTX_INFO_s	***pSrcCtx**	Source `context` pointer. This field is only available in Clone mode. It represents the source `context` to be cloned. Please refer to Section 5.2.2.1 below for more details.

*Table 5-5. Parameters for Utility Image Kernel API **AmbaDSP_ImgInitCtx()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 5-6. Returns for Utility Image Kernel API **AmbaDSP_ImgInitCtx()**.*

Example:

None

See Also:

None

5.2.2.1 AmbaDSP_ImgInitCtx > AMBA_DSP_IMG_CTX_INFO_s

Type	Field	Description
AMBA_DSP_IMG_PIPE_e	**Pipe**	Pipeline type
-	-	-
-	-	-
UINT8	**CtxId**	Context index

*Table 5-7. Definition of **AMBA_DSP_IMG_CTX_INFO_s** for Utility Image Kernel API **AmbaDSP_ImgInitCtx()**.*

5.2.3 AmbaDSP_ImgInitCfg

API Syntax:

AmbaDSP_ImgInitCfg (AMBA_DSP_IMG_CFG_INFO_s *pCfgInfo, AMBA_DSP_IMG_ALGO_MODE_e AlgoMode)

Function Description:

- This API is used to initialize the `config` to use selected algorithm modes. Depending on the **AlgoMode** used, the memory layout and initial values will vary. Once the `config` is initialized, then **PreExe** or **PostExe** actions can be performed.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_CFG_INFO_s	***pCfgInfo**	Config information. Please refer to Section 5.2.3.1 below for more details.
AMBA_DSP_IMG_ALGO_MODE_e	**AlgoMode**	The algorithm mode to initialize.

*Table 5-8. Parameters for Utility Image Kernel API **AmbaDSP_ImgInitCfg()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 5-9. Returns for Utility Image Kernel API **AmbaDSP_ImgInitCfg()**.*

Example:

None

See Also:

None

5.2.3.1 AmbaDSP_ImgInitCfg > AMBA_DSP_IMG_CFG_INFO_s

Type	Field	Description
AMBA_DSP_IMG_PIPE_e	**Pipe**	Pipeline type
UINT8	**CfgId**	`Config` index

*Table 5-10. Definition of **AMBA_DSP_IMG_CFG_INFO_s** for Utility Image Kernel API **AmbaDSP_ImgInitCfg()**.*

5.2.4 AmbaDSP_ImgPreExeCfg

API Syntax:

AmbaDSP_ImgPreExeCfg (AMBA_DSP_IMG_MODE_CFG_s *pMode, UINT32 ExeMode)

Function Description:

- This API is used to generate settings registers for specific algorithm modes using `context ids` from the pipeline. The generated settings are sufficient for DSP initialization once capture starts, but please note that **PostExe** must be invoked prior to the Raw-to-YUV stage.
- Two execution modes are supported:
  - Fast Execute Mode (0): The contents in the table of the `context` are not copied to `config`. Instead, `config` simply stores and points to the table address in the `context`.
  - Full Copy Mode (1): The contents in the table of the `context` are copied to `config`. This mode is slower; however, it can help maintain the integrity of table content if other API filters modify this content before DSP processing completes.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details. Image Kernel will generate settings registers of certain **AlgoModes** from CtxId to CfgId.
UINT32	**ExeMode**	0: Fast Execute Mode 1: Full Copy Mode

*Table 5-11. Parameters for Utility Image Kernel API **AmbaDSP_ImgPreExeCfg()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 5-12. Returns for Utility Image Kernel API **AmbaDSP_ImgPreExeCfg()**.*

Example:

None

See Also:

AmbaDSP_ImgInitCfg()
AmbaDSP_ImgPostExeCfg()

5.2.5 AmbaDSP_ImgPostExeCfg

API Syntax:

AmbaDSP_ImgPostExeCfg (AMBA_DSP_IMG_MODE_CFG_s *pMode, UINT32 ExeMode)

Function Description:

- This API is used to generate settings registers for specific algorithm modes using `context ids` from the pipeline. The generated settings are sufficient for DSP Raw-to-YUV processing.
- Two execute modes are supported:
  - Fast Execute Mode (0): The contents in the table of the `context` are not copied to `config`. Instead, `config` simply stores and points to the table address in the `context`.
  - Full Copy Mode (1): The contents in the table of the `context` are copied to `config`. This mode is slower; however, it can help maintain the integrity of table content if other API filters modify this content before DSP processing completes.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details. Image Kernel will generate settings registers of certain **AlgoModes** from CtxId to CfgId.
UINT32	**ExeMode**	0: Fast Execute Mode 1: Full Copy Mode

*Table 5-13. Parameters for Utility Image Kernel API **AmbaDSP_ImgPostExeCfg()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 5-14. Returns for Utility Image Kernel API **AmbaDSP_ImgPostExeCfg()**.*

Example:

None

See Also:

AmbaDSP_ImgInitCfg()
AmbaDSP_ImgPreExeCfg()

5.2.6 AmbaDSP_ImgGetCfgStatus

API Syntax:

AmbaDSP_ImgGetCfgStatus (AMBA_DSP_IMG_CFG_INFO_s *pCfgInfo, AMBA_DSP_IMG_CFG_STATUS_s *pStatus)

Function Description:

- This API is used to retrieve `config` status including:
  - `Config` address: This address is required to be assigned to the DSP when performing Raw-to-YUV processing.
  - `Config` state: Indicates whether the `config` is initialized (i.e., **AmbaDSP_ImgInitCfg** is invoked), pre-executed (i.e., **AmbaDSP_ImgPreExeCfg** is invoked), or post-executed (i.e., **AmbaDSP_ImgPostExeCfg** is invoked)
  - `Config` **AlgoMode**: Indicates the algorithm mode of the `config`.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_CFG_INFO_s	***pCfgInfo**	`Config` information. Please refer to Section 5.2.3.1 for more details.
AMBA_DSP_IMG_CFG_STATUS_s	***pStatus**	Returned `config` status pointer. Please refer to Section 5.2.6.1 below for more details.

*Table 5-15. Parameters for Utility Image Kernel API **AmbaDSP_ImgGetCfgStatus()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 5-16. Returns for Utility Image Kernel API **AmbaDSP_ImgGetCfgStatus()**.*

Example:

None

See Also:

AmbaDSP_ImgInitCfg()
AmbaDSP_ImgPreExeCfg()
AmbaDSP_ImgPostExeCfg()

5.2.6.1 AmbaDSP_ImgGetCfgStatus > AMBA_DSP_IMG_CFG_STATUS_s

Type	Field	Description
UINT32	**Addr**	Pipeline type
AMBA_DSP_IMG_CFG_STATE_e	**State**	`Config` state: 0: IDLE state. `Config` is not initialized yet. (**AmbaDSP_ImgInitCfg** is not invoked yet) 1: INIT state. `Config` is initialized but not executed yet. (**AmbaDSP_ImgInitCfg** is invoked) 2: PREEXE state. `Config` is pre-executed. (**AmbaDSP_ImgPreExeCfg** is invoked) 2: POSTEXE state. `Config` is post-executed. (**AmbaDSP_ImgPostExeCfg** is invoked)
AMBA_DSP_IMG_ALGO_MODE_e	**AlgoMode**	Algorithm mode: 0: **AMBA_DSP_IMG_ALGO_MODE_FAST** 1: **AMBA_DSP_IMG_ALGO_MODE_LISO** 3: **AMBA_DSP_IMG_ALGO_MODE_HISO**

*Table 5-17. Definition of **AMBA_DSP_IMG_CFG_STATUS_** for Utility Image Kernel API **AmbaDSP_ImgGetCfgStatus()**.*

5.2.7 AmbaDSP_ImgSetSizeInfo

API Syntax:

AmbaDSP_ImgSetSizeInfo (AMBA_DSP_IMG_MODE_CFG_s *pMode, AMBA_DSP_IMG_SIZE_INFO_s *pSizeInfo)

Function Description:

- This API is used to set the image size information required by the ISO algorithm. This API must be invoked prior to **AmbaDSP_ImgPreExeCfg** or **AmbaDSP_ImgPostExeCfg**.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	***pMode**	Mode. Please refer to Section 2.2.1.1 for more details.
AMBA_DSP_IMG_SIZE_INFO_s	***pSizeInfo**	Size information. Please refer to Section 5.2.7.1 below for more details.

*Table 5-18. Parameters for Utility Image Kernel API **AmbaDSP_ImgSetSizeInfo()**.*

Returns:

Return	Description
AMBA_DSP_IMG_RVAL_SUCCESS (0)	Success
AMBA_DSP_IMG_RVAL_ERROR (-1)	Failure

*Table 5-19. Returns for Utility Image Kernel API **AmbaDSP_ImgSetSizeInfo()**.*

Example:

None

See Also:

AmbaDSP_ImgPreExeCfg()
AmbaDSP_ImgPostExeCfg()

5.2.7.1 AmbaDSP_ImgSetSizeInfo > AMBA_DSP_IMG_SIZE_INFO_s

Type	Field	Description
UINT16	**WidthIn**	Input raw data width
UINT16	**HeightIn**	Input raw data height
UINT16	**WidthMain**	Main image width
UINT16	**HeightMain**	Main image height
UINT16	**WidthPrevA**	Preview A output width
UINT16	**HeightPrevA**	Preview A output height
UINT16	**WidthPrevB**	Preview B output width
UINT16	**HeightPrevB**	Preview B output height
UINT16	**WidthScrn**	Screennail width
UINT16	**HeightScrn**	Screennail height
UINT16	**WidthQvRaw**	Quickview raw width
UINT16	**HeightQvRaw**	Quickview raw height

*Table 5-20. Definition of **AMBA_DSP_IMG_SIZE_INFO_s** for Utility Image Kernel API **AmbaDSP_ImgSetSizeInfo()**.*

Appendix 1 Additional Resources

Related resources include:

- *A12 Datasheet (chip specific title)*
- *A12 Hardware Programming Reference Manual*
- *A12 System Hardware*
- *SDK6 AN ADC and IR Input*
- *SDK6 AN Custom Audio Codec Driver*
- *SDK6 AN Custom Image Sensor Driver*
- *SDK6 AN Custom LCD Panel Driver*

Please contact an Ambarella representative for a full list of related resources.

Appendix 2 Important Notice

All Ambarella design specifications, datasheets, drawings, files, and other documents (together and separately, “materials”) are provided on an “*as is*” basis, and Ambarella makes no warranties, expressed, implied, statutory, or otherwise with respect to the materials, and expressly disclaims all implied warranties of noninfringement, merchantability, and fitness for a particular purpose. The information contained herein is believed to be accurate and reliable. However, Ambarella assumes no responsibility for the consequences of use of such information.

Ambarella Incorporated reserves the right to correct, modify, enhance, improve, and otherwise change its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

All products are sold subject to Ambarella’s terms and conditions of sale supplied at the time of order acknowledgment. Ambarella warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with its standard warranty. Testing and other quality control techniques are used to the extent Ambarella deems necessary to support this warranty.

Ambarella assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using Ambarella components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

Ambarella does not warrant or represent that any license, either expressed or implied, is granted under any Ambarella patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Ambarella products or services are used. Information published by Ambarella regarding third-party products or services does not constitute a license from Ambarella to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from Ambarella under the patents or other intellectual property of Ambarella.

Reproduction of information from Ambarella documents is not permissible without prior approval from Ambarella.

Ambarella products are not authorized for use in safety-critical applications (such as life support) where a failure of the product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Customers acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of Ambarella products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by Ambarella. Further, Customers must fully indemnify Ambarella and its representatives against any damages arising out of the use of Ambarella products in such safety-critical applications.

Ambarella products are neither designed nor intended for use in military/aerospace applications or environments. Customers acknowledge and agree that any such use of Ambarella products is solely at the Customer’s risk, and they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

Appendix 3 Revision History

NOTE: Page numbers for previous drafts may differ from page numbers in the current version.

Version	Date	Comments
1.0	12 AUG 2014	Preliminary Release
1.1	22 SEP 2014	Updated Sections 3.2.23, 3.2.23.1, 3.2.25, 3.2.41.1, 3.2.45, 3.2.45.1, 3.2.46, 3.2.49, 3.2.49.1, 3.2.51, 3.2.61, 3.2.63, 3.2.63.1, 3.2.65, 3.2.71, 3.2.75, 3.2.79 Updated Table 3-131
1.2	9 DECEMBER 2014	Updated Section 3.2.35 Updated Figures 3-6, 3-7, 3-10, 3-11, 4-3, and 4-4
1.3	23 JANUARY 2015	Updated Section 3.2.79 Updated Figure 3-16
1.4	25 FEBRUARY 2015	Deleted Sections 3.2.63~3.2.76 Deleted Sections 3.2.81~3.2.84 Updated Section 3.2.90 Added Sections 3.2.74, 3.2.75

Table A3-1. Revision History.