Ambarella

SDK6 API A12 DSP Support Package

Version 1.1

October 31, 2014

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

1.1 Overview: Introduction

This document provides information on the Digital Signal Processing (DSP) support package Application Programming Interface (API) from Ambarella.

The document is organized as follows:

- (Chapter 2) DSP
- (Chapter 3) Event Handler
- (Chapter 4) Video Output
- (Chapter 5) Video Input
- (Chapter 6) Liveview
- (Chapter 7) Still Capture
- (Chapter 8) Still Decode
- (Chapter 9) Video Encode
- (Chapter 10) Video Decode
- (Chapter 11) Warp Core

Refer to Appendix 1 for a list of related resources.

1.2 Overview: Background

The DSP support package API provides controls for DSP functions on the Ambarella A12 processor, a versatile multi-format (H.264 and JPEG) video codec, and the image processor System on a Chip (SoC). The functions provided in this document enable A12 interface hardware control blocks, including the sensor input interface, the video output interface, and the H.264 and JPEG codecs.

1.3 Overview: Scope of Document

This document focuses strictly on the API for the A12 DSP support package. 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 background information:

- The A12 datasheet provides hardware pin and package details including a feature list with description of chip performance, brief interface descriptions, a complete power-on configuration table, and electrical characteristics.
- The "*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 RM: System Hardware*" covers power-on timing and sequencing. It provides pin connection details including guidance for unused interfaces and PCB layout.

2 DSP

2.1 DSP: Overview

This chapter provides details on the APIs involved in the initialization of the DSP support package. The APIs outlined in this chapter must be called before any other DSP support package APIs can be used.

2.2 DSP: List of Functions

2.2.1 AmbaDSP_Init

API Syntax:

AmbaDSP_Init (AMBA_DSP_SYS_CONFIG_s *pDspSysConfig)

Function Description:

- This function is used to initialize the DSP boot process with a defined configuration.

Parameters:

Type	Parameter	Description
AMBA_DSP_SYS_CONFIG_s	***pDspSysConfig**	Pointer to the DSP system configuration. Please refer to Section 2.2.1.1 for more details.

*Table 2-1. Parameters for AmbaDSP API **AmbaDSP_Init()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-2. Returns for AmbaDSP API **AmbaDSP_Init()**.*

Example:

```
#define AMBA_DSP_VOUT_LCD 0
#define AMBA_DSP_VOUT_TV   1

AMBA_DSP_SYS_CONFIG_s AmbaDSP_SysConfig = {

    .pDebugLogDataArea    = AmbaDSP_DebugLogDataArea,
/* pointer to DSP Debug Log Data area */
    .DebugLogDataAreaSize = sizeof(AmbaDSP_DebugLogDataArea),
/* DSP Debug Log Data area in Bytes */
    .pWorkArea    = AmbaDSP_WorkArea,           /* pointer to DSP Work Data
area */
    .WorkAreaSize = sizeof(AmbaDSP_WorkArea),   /* DSP Work Data area in Bytes
*/

    .VoutMixerConfigLCD.ActiveWidth  = AMBA_LCD_WIDTH,
    .VoutMixerConfigLCD.ActiveHeight = AMBA_LCD_HEIGHT,
    .VoutMixerConfigLCD.VideoHorFlipEnable = 0,
    .VoutMixerConfigLCD.Interlace = 0,
    .VoutMixerConfigLCD.FrameRate = AMBA_DSP_FRAME_RATE_59_94_P,
    .VoutMixerConfigLCD.MixerColorFormat = MIXER_IN_YUV_444_RGB,
    .VoutOsdBufConfigLCD.FieldRepeat = 0,
    .VoutOsdBufConfigLCD.Pitch = AMBA_LCD_WIDTH << 2,
/* 32-bit OSD */
    .VoutOsdBufConfigLCD.pBaseAddr = (void *) AmbaOSD_Buf,
```

```
    .VoutDefaultImgConfig[AMBA_DSP_VOUT_LCD].FieldRepeat = 0,
    .VoutDefaultImgConfig[AMBA_DSP_VOUT_LCD].Pitch = AMBA_LCD_WIDTH,
    .VoutDefaultImgConfig[AMBA_DSP_VOUT_LCD].pBaseAddrY = AmbaDSP_Default-
ImgLumaLCD,
    .VoutDefaultImgConfig[AMBA_DSP_VOUT_LCD].pBaseAddrUV = AmbaDSP_DefaultImg-
ChromaLCD,

    .VoutDefaultImgConfig[AMBA_DSP_VOUT_TV].FieldRepeat = 0,
    .VoutDefaultImgConfig[AMBA_DSP_VOUT_TV].Pitch = AMBA_MAX_TV_WIDTH,
    .VoutDefaultImgConfig[AMBA_DSP_VOUT_TV].pBaseAddrY = AmbaDSP_DefaultImgLu-
maTV,
    .VoutDefaultImgConfig[AMBA_DSP_VOUT_TV].pBaseAddrUV = AmbaDSP_DefaultImg-
ChromaTV,
};


AmbaDSP_Init(&AmbaDSP_SysConfig);
```

See Also:

AmbaDSP_SetWorkArea

2.2.1.1 AmbaDSP_Init > AMBA_DSP_SYS_CONFIG_s

Type	Field	Description
AMBA_DSP_SYS_STATE_e	**SysState**	DSP System state. Please refer to Section 2.2.1.2 for more details.
UINT8	***pDebugLogDataArea**	Pointer to DSP Debug Log Data area
UINT32	**DebugLogDataAreaSize**	DSP Debug Log Data area in Bytes
UINT8	***pWorkArea**	Pointer to DSP Work Data area
UINT32	**WorkAreaSize**	DSP Work Data area in Bytes
AMBA_DSP_VOUT_MIXER_CONFIG_s	**VoutMixerConfigLCD**	Mixer configurations for LCD. Please refer to Section 2.2.1.3 for more details.
AMBA_DSP_VOUT_OSD_BUF_CONFIG_s	**VoutOsdBufConfigLCD**	OSD DRAM Buffer configuration for LCD. Please refer to Section 2.2.1.6 for more details.
AMBA_DSP_VOUT_DEFAULT_IMG_CONFIG_s	**VoutDefaultImgConfig[2]**	Default image configuration for VOUTs. Please refer to Section 2.2.1.7 for more details.

*Table 2-3. Definition of **AMBA_DSP_SYS_CONFIG_s** for AmbaDSP API **AmbaDSP_Init()**.*

2.2.1.2 AmbaDSP_Init > AMBA_DSP_SYS_STATE_e

Type	Description
AMBA_DSP_SYS_STATE_LIVEVIEW	Liveview state
AMBA_DSP_SYS_STATE_PLAYBACK	Still playback state
AMBA_DSP_SYS_STATE_SENSORLESS	Sensor-less encode state; internal use only.
AMBA_NUM_DSP_SYS_STATE	Number of DSP state

*Table 2-4. Definition of **AMBA_DSP_SYS_STATE_e** for AmbaDSP API **AmbaDSP_Init()**.*

2.2.1.3 AmbaDSP_Init > AMBA_DSP_VOUT_MIXER_CONFIG_s

Type	Field	Description
UINT16	**ActiveWidth**	Number of pixels per line
UINT16	**ActiveHeight**	Number of lines
UINT8	**VideoHorReverseEnable**	1: Video Data is horizontally reversed
UINT8	**Interlace**	0: Progressive 1: Interlace
AMBA_DSP_FRAME_RATE_s	**FrameRate**	Frame rate. Please refer to Section 2.2.1.4 for more details.
AMBA_DSP_VOUT_MIXER_COLOR_FORMAT_e	**MixerColorFormat**	Mixer color format. Please refer to Section 2.2.1.5 for more details.

*Table 2-5. Definition of **AMBA_DSP_VOUT_MIXER_CONFIG_s** for AmbaDSP API **AmbaDSP_Init()**.*

2.2.1.4 AmbaDSP_Init > AMBA_DSP_FRAME_RATE_s

Type	Field	Description
UINT8	**Interlace**	0: Progressive 1: Interlace
UINT32	**TimeScale**	Time scale
UINT32	**NumUnitsInTick**	Fps = time scale / number of units in tick / (1 + unit field based flag)

*Table 2-6. Definition of **AMBA_DSP_FRAME_RATE_s** for AmbaDSP API **AmbaDSP_Init()**.*

2.2.1.5 AmbaDSP_Init > AMBA_DSP_VOUT_MIXER_COLOR_FORMAT_e

Type	Description
MIXER_IN_YUV_422	YUV 422
MIXER_IN_YUV_444_RGB	YUV 444 or RGB

*Table 2-7. Definition of **AMBA_DSP_VOUT_MIXER_COLOR_FORMAT_e** for AmbaDSP API **AmbaDSP_Init()**.*

2.2.1.6 AmbaDSP_Init > AMBA_DSP_VOUT_OSD_BUF_CONFIG_s

Type	Field	Description
UINT8	**FieldRepeat**	Repeat the default image for top field and bottom field in interlaced mode
UINT16	**Pitch**	The DRAM pitch (number of bytes) between two contiguous lines
void	***pBaseAddr**	Pointer to the base address of OSD data

*Table 2-8. Definition of **AMBA_DSP_VOUT_OSD_BUF_CONFIG_s** for AmbaDSP API **AmbaDSP_Init()**.*

2.2.1.7 AmbaDSP_Init > AMBA_DSP_VOUT_DEFAULT_IMG_CONFIG_s

Type	Field	Description
UINT8	**FieldRepeat**	Repeat the default image for top field and bottom field in interlaced mode
UINT16	**Pitch**	The DRAM pitch (number of bytes) between two contiguous lines
UINT8	***pBaseAddrY**	Pointer to Luma (Y) data area
UINT8	***pBaseAddrUV**	Pointer to Chroma (UV) data area

*Table 2-9. Definition of **AMBA_DSP_VOUT_DEFAULT_IMG_CONFIG_s** for AmbaDSP API **AmbaDSP_Init()**.*

2.2.2 AmbaDSP_CalldspFreq

API Syntax:

AmbaDSP_CalldspFreq (AMBA_DSP_CAL_IDSP_FREQ_INFO_s*pInfo)

Function Description:

- This function is used to calculate a close-to-minimal IDSP clock frequency for the encode mode.

Parameters:

Type	Parameter	Description
AMBA_DSP_CAL_IDSP_FREQ_INFO_s	***pInfo**	Information to calculate IDSP frequency. Please refer to Section 2.2.2.1 for more details.

*Table 2-10. Parameters for AmbaDSP API **AmbaDSP_CalldspFreq()**.*

Returns:

Return	Description
Frequency	Suggested frequency

*Table 2-11. Returns for AmbaDSP API **AmbaDSP_CalldspFreq()**.*

Example:

None

See Also:

None

2.2.2.1 AmbaDSP_CalldspFreq > AMBA_DSP_CAL_IDSP_FREQ_INFO_s

Type	Field	Description
UINT32	**VinCapWidth**	Vin capture window width
UINT32	**VinCapHeight**	Vin capture window height
AMBA_DSP_FRAME_RATE_s	**VinFramerate**	Vin frame rate in integer
UINT32	**VBlanking**	Number of vertical blank rows
UINT32	**VinOutWidth**	Vin total width (Capture window width + Horizontal blank)
UINT32	**MainYuvWidth**	Main window width
UINT32	**MainYuvHeight**	Main window height
UINT32	**LcdLiveviewWidth**	LCD liveview window width
UINT32	**LcdLiveviewHeight**	LCD liveview windows height
AMBA_DSP_FRAME_RATE_s	**LcdLiveviewFrateRate**	LCD liveview frame rate
UINT32	**TvLiveviewWidth**	TV liveview window width
UINT32	**TvLiveviewHeight**	TV liveview windows height

Type	Field	Description
AMBA_DSP_FRAME_RATE_s	**TvLiveviewFrateRate**	TV liveview frame rate
-	-	-
UINT8:2	**VideoProcMode**	0: Express pipeline 1: Hybrid pipeline
UINT8:3	**VideoAlgoMode**	0: Fast 1: LISO 2: HISO
UINT8:3	**VideoOSMode**	0: Non-oversampling 1: Oversampling

*Table 2-12. Definition of **AMBA_DSP_CAL_IDSP_FREQ_INFO_s** for AmbaDSP API **AmbaDSP_CalIdspFreq()**.*

2.2.3 AmbaDSP_CalCoreFreq

API Syntax:

AmbaDSP_CalCoreFreq (AMBA_DSP_CAL_CORE_FREQ_INFO_s * pInfo)

Function Description:

- This function is used to calculate a close-to-minimal Core clock frequency for the encode mode.

Parameters:

Type	Parameter	Description
AMBA_DSP_CAL_CORE_FREQ_INFO_s	***pInfo**	Information to calculate core frequency. Please refer to Section 2.2.3.1 for details
-	-	-
-	-	-
-	-	-

*Table 2-13. Parameters for AmbaDSP API **AmbaDSP_CalCoreFreq()**.*

Returns:

Return	Description
Frequency	Suggested frequency

*Table 2-14. Returns for AmbaDSP API **AmbaDSP_CalCoreFreq()**.*

Example:

None

See Also:

None

2.2.3.1 AmbaDSP_CalCoreFreq > AMBA_DSP_CAL_CORE_FREQ_INFO_s

Type	Field	Description
UINT32	**NumEncStream**	Number of streams to encode
UINT32	***pFrameSize**	Array of frame width*height
UINT32	***pFrameRate**	Array of frame rate in integer
UINT32	**LcdLiveviewWidth**	LCD liveview window width
UINT32	**LcdLiveviewHeight**	LCD liveview window height
UINT32	**LcdLiveviewFrateRate**	LCD liveview frame rate
UINT32	**TvLiveviewWidth**	TV liveview window width
UINT32	**TvLiveviewHeight**	TV liveview window height
UINT32	**TvLiveviewFrateRate**	TV liveview frame rate

*Table 2-15. Definition of **AMBA_DSP_CAL_CORE_FREQ_INFO_s** for AmbaDSP API **AmbaDSP_CalCoreFreq()**.*

2.2.4 AmbaDSP_Suspend

API Syntax:

AmbaDSP_Suspend (void)

Function Description:

- This function is used to trigger a suspend state for the DSP.

Parameters:

None

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-16. Returns for AmbaDSP API **AmbaDSP_Suspend()**.*

Example:

```
/* System boot done */
INT8 boot_done = 1;

/* press some key */
boot_done ++;
if (boot_done % 2 == 0)
      AmbaDSP_Suspend();
else
      AmbaDSP_Resume();
```

See Also:

None

2.2.5 AmbaDSP_Resume

API Syntax:

AmbaDSP_Resume (void)

Function Description:

- This function is used to resume normal DSP functions after the **AmbaDSP_Suspend** API has been called.

Parameters:

None

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-17. Returns for AmbaDSP API **AmbaDSP_Resume()**.*

Example:

Please refer to the example of Section 2.2.2.

See Also:

AmbaDSP_Suspend

2.2.6 AmbaDSP_SetWorkArea

API Syntax:

AmbaDSP_SetWorkArea (UINT8 *pWorkArea, UINT32 WorkAreaSize)

Function Description:

- This function is used to configure the size of the DSP work area.

Parameters:

Type	Parameter	Description
UINT8	***pWorkArea**	Pointer to DSP Work Data area
UINT32	**WorkAreaSize**	DSP Work Data area size (Bytes)

*Table 2-18. Parameters for AmbaDSP API **AmbaDSP_SetWorkArea()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-19. Returns for AmbaDSP API **AmbaDSP_SetWorkArea()**.*

Example:

None

See Also:

None

3 Event Handler

3.1 Event Handler: Overview

This chapter provides information regarding Event Handler functions.

3.2 Event Handler: List of Functions

3.2.1 AmbaDSP_EventInit

API Syntax:

AmbaDSP_EventInit (void)

Function Description:

- This function is used to initialize the Event Handler module.

Parameters:

None

Returns:

Return	Description
0	Success
- 1	Failure

*Table 3-1. Returns for Event Handler API **AmbaDSP_EventInit()**.*

Example:

None

See Also:

None

3.2.2 AmbaDSP_EventHandlerCtrlReset

API Syntax:

AmbaDSP_EventHandlerCtrlReset (UINT32 EventID)

Function Description:

- This function is used to reset the Event Handler control module settings.

Parameters:

Type	Parameter	Description
UINT32	**EventID**	Reset certain EventID's event handler

*Table 3-2. Parameters for Event Handler API **AmbaDSP_EventHandlerCtrlReset()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 3-3. Returns for Event Handler API **AmbaDSP_EventHandlerCtrlReset()**.*

Example:

None

See Also:

None

3.2.3 AmbaDSP_EventHandlerCtrlConfig

API Syntax:

AmbaDSP_EventHandlerCtrlConfig (AMBA_DSP_EVENT_ID_e EventID, int MaxNumHandler, AMBA_DSP_EVENT_HANDLER_f *pEventHandlers)

Function Description:

- This function is used to configure Event Handler control settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_EVENT_ID_e	**EventID**	Event ID. Please refer to Section 3.2.3.1 for more details.
int	**MaxNumHandler**	Maximum number of Handlers
AMBA_DSP_EVENT_HANDLER_f	***pEventHandlers**	Pointer to the Event Handlers. Please refer to Section 3.2.3.2 for more details.

*Table 3-4. Parameters for Event Handler API **AmbaDSP_EventHandlerCtrlConfig()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 3-5. Returns for Event Handler API **AmbaDSP_EventHandlerCtrlConfig()**.*

Example:

To allow 4 event handlers to listen to one event, the following configuration can be applied.

```
static AMBA_DSP_EVENT_HANDLER_f  AmbaMovieRecordVideoEventHandlers[4];

AmbaDSP_EventHandlerCtrlConfig(AMBA_DSP_EVENT_ID_ENC_CH0_MAIN_PIC_READY,
4,
AmbaMovieRecordVideoEventHandlers);
```

See Also:

None

3.2.3.1 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_ID_e

Type	Description
AMBA_DSP_EVENT_CFA_3A_DATA_READY	CFA 3A data ready. **AMBA_DSP_EVENT_CFA_3A_DATA_s**. Please refer to Section 3.2.3.3 for more details.
AMBA_DSP_EVENT_RGB_3A_DATA_READY	RGB 3A data ready. **AMBA_DSP_EVENT_RGB_3A_DATA_s**. Please refer to Section 3.2.3.4 for more details.
AMBA_DSP_EVENT_RAW_CFA_3A_DATA_READY	RAW CFA 3A data ready. **AMBA_DSP_EVENT_CFA_3A_DATA_s.**
AMBA_DSP_EVENT_RAW_3A_DATA_READY	RAW RGB 3A data ready. **AMBA_DSP_EVENT_RGB_3A_DATA_s**.
AMBA_DSP_EVENT_VIDEO_DATA_READY	Video data ready. **AMBA_DSP_EVENT_ENC_PIC_READY_s**. Please refer to Section 3.2.3.21 for more details.
AMBA_DSP_EVENT_JPEG_DATA_READY	JPEG data ready. **AMBA_DSP_EVENT_ENC_PIC_READY_s**. Please refer to Section 3.2.3.21 for more details.
AMBA_DSP_EVENT_LIVEVIEW_RAW_DATA_READY	Liveview RAW data is ready. **AMBA_DSP_RAW_BUF_s**. Please refer to Section 7.2.3.3 for more details.
AMBA_DSP_EVENT_LIVEVIEW_LCD_YUV_DATA_READY	Liveview LCD YUV data is ready. **AMBA_DSP_YUV_IMG_BUF_s**.
AMBA_DSP_EVENT_LIVEVIEW_TV_YUV_DATA_READY	Liveview TV YUV data is ready. **AMBA_DSP_YUV_IMG_BUF_s**.
AMBA_DSP_EVENT_MAIN_YUV_DATA_READY	Main video YUV data is ready. **AMBA_DSP_YUV_IMG_BUF_s**.
AMBA_DSP_EVENT_2ND_YUV_DATA_READY	Secondary video YUV data is ready. **AMBA_DSP_YUV_IMG_BUF_s**.
AMBA_DSP_EVENT_SCAP_RAW_DATA_READY	RAW data ready for still capture. **AMBA_DSP_EVENT_SCAP_RAW_DATA_READY_s**. Please refer to Section 3.2.3.24 for more details.
AMBA_DSP_EVENT_SCAP_RAW_BUFFER_FULL	RAW buffer full for still capture
AMBA_DSP_EVENT_SCAP_CLEAN_RAW_DATA_READY	RAW data ready to be cleaned for still capture.
AMBA_DSP_EVENT_SCAP_YUV_DATA_READY	YUV data ready for still capture. **AMBA_DSP_EVENT_SCAP_YUV_DATA_READY**_s. Please refer to Section 3.2.3.26 for more details.
AMBA_DSP_EVENT_SCAP_HDR_DATA_READY	The argument of the event handler will be the HDR capture count.
AMBA_DSP_EVENT_JPEG_DEC_STATUS_REPORT	JPEG decode status report. **AMBA_DSP_EVENT_JPEG_DEC_STATUS_REPORT_s**. Please refer to Section 3.2.3.28 for more details.
AMBA_DSP_EVENT_JPEG_DEC_YUV_DISP_REPORT	YUV report for still decoder. **AMBA_DSP_EVENT_JPEG_DEC_YUV_DISP_REPORT_S**. Please refer to Section 3.2.3.29 for more details.
AMBA_DSP_EVENT_H264_DEC_STATUS_REPORT	Video decode status report. **AMBA_DSP_EVENT_H264_DEC_STATUS_UPDATE_s**. Please refer to Section 3.2.3.30 for more details.
AMBA_DSP_EVENT_VIDEO_ENC_START	Video encode start. Please refer to Section 3.2.3.33 for more details. The high word of parameter is view ID and low word of parameter is stream ID.
AMBA_DSP_EVENT_VIDEO_ENC_PAUSE	Video encode pause. To signal service encoder is paused.

Type	Description
AMBA_DSP_EVENT_2FOV_VCAP_RAW_DATA_READY	RAW data ready for 2 FOV mode. **AMBA_DSP_RAW_BUF_s**. Please refer to Section 7.2.3.3 for more details.
AMBA_NUM_DSP_EVENT	Number of events

*Table 3-6. Definition of **AMBA_DSP_EVENT_ID_e** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig()**.*

3.2.3.2 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_ AMBA_DSP_ EVENT_ HANDLER_f

Type	Description
AMBA_DSP_ EVENT_HANDLER_f	typedef int (*AMBA_DSP_EVENT_HANDLER_f)(void *pEventData)

*Table 3-7. Definition of **AMBA_DSP_ EVENT_ HANDLER_f** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig()**.*

3.2.3.3 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_CFA_3A_DATA_s

Type	Field	Description
AMBA_DSP_3A_HEADER_s	**HEADER**	Header of AAA data. Please refer to Section 3.2.3.4 for more details.
UINT16	**FRAMEID**	Frame ID
AMBA_DSP_CFA_AWB_s	**AWB[ ]**	AMBA_DSP_3A_AWB_TILE_ROW_COUNT * AMBA_DSP_3A_AWB_TILE_COL_COUNT entries. AWB information. Please refer to Section 3.2.3.5 for more details.
AMBA_DSP_CFA_AE_s	**AE[ ]**	AMBA_DSP_3A_AE_TILE_ROW_COUNT * AMBA_DSP_3A_AE_TILE_COL_COUNT entries. AE information. Please refer to Section 3.2.3.6 for more details.
AMBA_DSP_CFA_AF_s	**AF[ ]**	AMBA_DSP_3A_AF_TILE_ROW_COUNT * AMBA_DSP_3A_AF_TILE_COL_COUNT entries. AF information. Please refer to Section 3.2.3.7 for more details.
AMBA_DSP_CFA_HISTO_s	**HISTO**	Histogram. Please refer to Section 3.2.3.8 for more details.
AMBA_DSP_CFA_FLOAT_AWB_s	**FLOATAWB[ ]**	AMBA_DSP_3A_FLOAT_TILE_COUNT entries. AWB information of floating tile. Please refer to Section 3.2.3.9 for more details.
AMBA_DSP_CFA_FLOAT_AE_s	**FLOATAE[ ]**	AMBA_DSP_3A_FLOAT_TILE_COUNT entries. AE information of floating tile. Please refer to Section 3.2.3.10 for more details.
AMBA_DSP_CFA_FLOAT_AF_s	**FLOATAF[ ]**	AE of tile. AMBA_DSP_3A_FLOAT_TILE_COUNT entries. AF information of floating tile. Please refer to Section 3.2.3.11 for more details.

*Table 3-8. Definition of **AMBA_DSP_EVENT_CFA_3A_DATA_s** for Event Handler API **AmbaDSP_GiveEvent()**.*

3.2.3.4 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_3A_HEADER_s

Type	Field	Description
UINT16	**AWBTILECOLSTART**	The horizontal starting pixel of first AWB tile. Unit is pixels.
UINT16	**AWBTILEROWSTART**	The vertical starting pixel of 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.

Type	Field	Description
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 << AwbYShift.
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] << AwbMinMaxShift.
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.
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**	Number of tiles in x direction
UINT8	**AWBTILENUMROW**	Number of tiles in y direction
UINT8	**AETILENUMCOL**	Number of tiles in x direction
UINT8	**AETILENUMROW**	Number of tiles in y direction
UINT8	**AFTILENUMCOL**	Number of tiles in x direction
UINT8	**AFTILENUMROW**	Number of tiles in y direction
UINT8	**TOTALSLICESX**	Total slice number in x direction
UINT8	**TOTALSLICESY**	Total slice number in y direction
UINT8	**SLICEINDEXX**	Index of slice in x direction
UINT8	**SLICEINDEXY**	Index of slice in 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.
UINT8	**RESERVED[32]**	Padding to 128x bytes

*Table 3-9. Definition of **AMBA_DSP_3A_HEADER_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.5 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_CFA_AWB_s

Type	Field	Description
UINT16	**SUMR**	Sum of R pixels
UINT16	**SUMG**	Sum of G pixels

Type	Field	Description
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 3-10. Definition of **AMBA_DSP_CFA_AWB_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.6 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_CFA_AE_s

Type	Field	Description
UINT16	**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 3-11. Definition of **AMBA_DSP_CFA_AE_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.7 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_CFA_AF_s

Type	Field	Description
UINT16	**SUMY**	Sum of luma component
UINT16	**SUMFV1**	Sum of first gradient
UINT16	**SumFV2**	Sum of second gradient

*Table 3-12. Definition of **AMBA_DSP_CFA_AF_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.8 AmbaDSP_EventHandlerCtrlConfig > 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]**	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	**HisBinB[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	**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.

*Table 3-13. Definition of **AMBA_DSP_CFA_HISTO_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.9 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_CFA_FLOAT_AWB_s

Type	Field	Description
UINT32	**SUMR**	Sum of R pixels
UINT32	**SUMG**	Sum of G pixels
UINT32	**SumB**	Sum of B pixels
UINT32	**COUNTMIN**	Number of pixels below the minimum threshold
UINT32	**COUNTMAX**	Number of pixels below the maximum threshold

*Table 3-14. Definition of **AMBA_DSP_CFA_FLOAT_AWB_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.10 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_CFA_FLOAT_AE_s

Type	Field	Description
UINT32	**LINY**	Sum of pseudo Y values
UINT32	**COUNTMIN**	Number of pixels below the minimum threshold
UINT32	**CountMax**	Number of pixels below the maximum threshold

*Table 3-15. Definition of **AMBA_DSP_CFA_FLOAT_AE_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.11 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_CFA_FLOAT_AF_s

Type	Field	Description
UINT32	**SUMFV1**	Sum of first gradient
UINT32	**SUMFV2**	Sum of second gradient

*Table 3-16. Definition of **AMBA_DSP_CFA_FLOAT_AF_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.12 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_RGB_3A_DATA_s

Type	Field	Description
AMBA_DSP_3A_HEADER_s	**HEADER**	Header of AAA data. Please refer to Section 3.2.3.4 for more details.
UINT16	**FRAMEID**	Frame ID
AMBA_DSP_RGB_AF_s	**AF[ ]**	AMBA_DSP_3A_AF_TILE_ROW_COUNT * AMBA_DSP_3A_AF_TILE_COL_COUNT entries. AF information. Please refer to Section 3.2.3.13 for more details.
AMBA_DSP_RGB_AE_s	**AE[ ]**	AMBA_DSP_3A_AE_TILE_ROW_COUNT * AMBA_DSP_3A_AE_TILE_COL_COUNT entries. AE information. Please refer to Section 3.2.3.14 for more details.
AMBA_DSP_RGB_HISTO_s	**HISTO**	Histogram. Please refer to Section 3.2.3.15 for more details.
AMBA_DSP_RGB_FLOAT_AF_s	**FLOATAF[]**	AMBA_DSP_3A_FLOAT_TILE_COUNT entries. AF information of floating tile. Please refer to Section 3.2.3.16 for more details.
AMBA_DSP_RGB_FLOAT_AE_s	**FLOATAE[ ]**	AMBA_DSP_3A_FLOAT_TILE_COUNT entries. AWB information of floating tile Please refer to Section 3.2.3.17 for more details.

*Table 3-17. Definition of **AMBA_DSP_EVENT_RGB_3A_DATA_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.13 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_RGB_AF_s

Type	Field	Description
UINT32	**SUMFY**	Sum of luma component
UINT32	**SUMFV2**	Sum of first gradient
UINT32	**SUMFV2**	Sum of second gradient

*Table 3-18. Definition of **AMBA_DSP_RGB_AF_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.14 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_RGB_AE_s

Type	Field	Description
UINT16	**SUMY**	Sum of luma component regardless of whether the pixel is saturated and right-shifted by AeYShift.

*Table 3-19. Definition of **AMBA_DSP_RGB_AE_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.15 AmbaDSP_EventHandlerCtrlConfig > 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]**	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]**	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	**HISBINB[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.

*Table 3-20. Definition of **AMBA_DSP_RGB_HISTO_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.16 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_RGB_FLOAT_AF_s

Type	Field	Description
UINT32	**SUMFV1H**	Sum of first horizontal gradient
UINT32	**SUMFV2H**	Sum of second horizontal gradient
UINT32	**SUMFV1V**	Sum of first vertical gradient
UINT32	**SUMFV2V**	Sum of second vertical gradient

*Table 3-21. Definition of **AMBA_DSP_RGB_FLOAT_AF_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.17 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_RGB_FLOAT_AE_s

Type	Field	Description
UINT32	**SUMY**	Sum of luma component regardless of whether the pixel is saturated and right-shifted by AeYShift.

*Table 3-22. Definition of **AMBA_DSP_RGB_FLOAT_AE_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.18 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_VIN_CTRL_EFFECTIVE_s

Type	Field	Description
UINT8	**VinID**	VIN ID

*Table 3-23. Definition of **AMBA_DSP_EVENT_VIN_CTRL_EFFECTIVE_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.19 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_VIN_CAPTURE_START_s

Type	Field	Description
UINT8	**VinID**	VIN ID
UINT32	**USERDATA**	Sequence number of the captured frame

*Table 3-24. Definition of **AMBA_DSP_EVENT_VIN_CAPTURE_START_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.20 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_VIN_CAPTURE_END_s

Type	Field	Description
UINT8	**VinID**	VIN ID
UINT32	**USERDATA**	Sequence number of the captured frame

*Table 3-25. Definition of **AMBA_DSP_EVENT_VIN_CAPTURE_END_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.21 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_ENC_PIC_READY_s

Type	Field	Description
UINT32	**CHANNELID**	The channel encoder of the specified picture
UINT32	**STREAMID**	The stream of the specified picture
UINT32	**FRMNO**	Counter of picture
UINT32	**PTS**	Presentation time stamp (PTS) of picture
UINT32	**STARTADDR**	Start address of picture buffer
UINT32	**FRAMETYPE**	Types of picture. 1: IDR picture (I picture implied) 2: Non-IDR I picture 3: P picture 4: B picture (0: Do not care)

Type	Field	Description
UINT32	**PICSTRUCT**	0: Frame picture 1: Field picture
UINT32	**PICSIZE**	Size of the picture

*Table 3-26. Definition of **AMBA_DSP_EVENT_ENC_PIC_READY_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.22 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_VCAP_RAW_DATA_READY_s

Type	Field	Description
UINT32	**CHANNELID**	Channel ID
UINT32	**STREAMID**	Stream ID
UINT32	**CAPSEQNUM**	Sequence number of capture
AMBA_DSP_EVENT_RAW_BUF_s	**RAWBUF**	Raw buffer

*Table 3-27. Definition of **AMBA_DSP_EVENT_VCAP_RAW_DATA_READY_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.23 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_VCAP_YUV_DATA_READY_s

Type	Field	Description
UINT32	**CHANNELID**	Channel ID
UINT32	**STREAMID**	Stream ID
UINT32	**YUVTYPE**	YUV data format
UINT32	**CAPSEQNUM**	Sequence number of capture
AMBA_DSP_EVENT_YUV_BUF_s	**YUVBUF**	YUV buffer

*Table 3-28. Definition of **AMBA_DSP_EVENT_VCAP_YUV_DATA_READY_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.24 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_SCAP_RAW_DATA_READY_s

Type	Field	Description
UINT8	**Compressed**	0: Uncompressed raw data 1: Compressed raw data
UINT8	***pBaseAddr**	Pointer to raw buffer
UINT16	**Pitch**	Raw buffer pitch

Type	Field	Description
AMBA_DSP_WINDOW_s	**Window.OffsetX**	Horizontal offset of the window
AMBA_DSP_WINDOW_s	**Window.OffsetY**	Vertical offset of the window
AMBA_DSP_WINDOW_s	**Window.Width**	Number of pixels per line in the window
AMBA_DSP_WINDOW_s	**Window.Height**	Number of lines in the window

*Table 3-29. Definition of **AMBA_DSP_EVENT_SCAP_RAW_DATA_READY_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.25 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_RAW_BUF_s

Type	Field	Description
UINT8	**ISCOMPACT**	1: Compact RAW data
UINT32	**ADDR**	Address
UINT16	**PITCH**	Pitch
UINT16	**SIZEX**	Width
UINT16	**SIZEY**	Height

*Table 3-30. Definition of **AMBA_DSP_EVENT_RAW_BUF_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.26 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_SCAP_YUV_DATA_READY_s

Type	Field	Description
AMBA_DSP_YUV_FORMAT_e	**DataFmt**	YUV Data format: 420 or 422 AMBA_DSP_YUV420: YUV 420 format AMBA_DSP_YUV422: YUV 422 format
UINT8	***pBaseAddrY**	Pointer to Luma (Y) data area
UINT8	***pBaseAddrUV**	Pointer to Chroma (UV) data area
UINT32	**Pitch**	YUV data buffer pitch
AMBA_DSP_WINDOW_s	**Window**	Window position and size

*Table 3-31. Definition of **AMBA_DSP_EVENT_SCAP_YUV_DATA_READY_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.27 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_YUV_BUF_s

Type	Field	Description
UINT8	**IS422**	1: YUV422 format
UINT32	**ADDRY**	Y Address
UINT32	**ADDRUV**	UV address

Type	Field	Description
UINT16	**PITCH**	Pitch
UINT16	**SIZEX**	Width
UINT16	**SIZEY**	Height

*Table 3-32. Definition of **AMBA_DSP_EVENT_YUV_BUF_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.28 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_JPEG_DEC_STATUS_REPORT_s

Type	Field	Description
UINT16	**Width**	Width of the image
UINT16	**Height**	Height of the image
UINT16	**YuvPitch**	Pitch of YUV data
UINT16	**YuvFormat**	Format of YUV data
UINT32	**STATUS**	Error status. Please refer to Chapter 8.2.3.2 for more details.

*Table 3-33. Definition of **AMBA_DSP_EVENT_JPEG_DEC_STATUS_REPORT_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.29 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_JPEG_DEC_YUV_DISP_REPORT

Type	Field	Description
UINT32	**VoutUpdated**	Indicates whether the VOUT is updated or not
UINT32	**VoutYAddr**	Address of current displayed buffer
UINT32	**VoutUVAddr**	Address of current displayed buffer

*Table 3-34. Definition of **AMBA_DSP_EVENT_JPEG_DEC_YUV_DISP_REPORT** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.30 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_H264_DEC_STATUS_UPDATE_s

Type	Field	Description
UINT32	**DECODERID**	Decoder ID
UINT32	**USERDATA**	User data
AMBA_DSP_DEC_STATE_e	**DECODESTATE**	Decode state

Type	Field	Description
UINT32	**ERRORSTATUS**	Error status. The ErrorStatus[23:16] indicates the error level. The error level: 0 - HDEC_ERR_LVL_NONE. Error Free. 1 - HDEC_ERR_LVL_WARNING. Warning Error Level. 2 - HDEC_ERR_LVL_RECOVERABLE. Recoverable Error Level. 3 - HDEC_ERR_LVL_FATAL. Fatal Error Level.
UINT32	**NUMOFDECODEDPIC**	Number of decoded frame
UINT32	**BITSNEXTREADADDR**	The latest buffer read address of the decoder
UINT32	**DISPLAYFRAMEPTS**	PTS of current display frame

*Table 3-35. Definition of **AMBA_DSP_EVENT_H264_DEC_STATUS_UPDATE_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.31 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_DEC_STATE_e

Type	Description
AMBA_DSP_DEC_STATE_INVALID	Decoder is in an invalid state.
AMBA_DSP_DEC_STATE_IDLE	Decoder is in an idle state.
AMBA_DSP_DEC_STATE_RUN	Decoder is in a running state.
AMBA_DSP_DEC_STATE_TRAN_FROMIDLE_WITH_LAST_PIC	Decoder is in an idle state with the last picture shown on-screen.

*Table 3-36. Definition of **AMBA_DSP_DEC_STATE_e** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.32 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_H264_DEC_YUV_REPORT_s

Type	Field	Description
UINT32	**DECODERID**	Indicates which channel decoder is starting
UINT32	**USERDATA**	User data

*Table 3-37. Definition of **AMBA_DSP_EVENT_H264_DEC_YUV_REPORT_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.3.33 AmbaDSP_EventHandlerCtrlConfig > AMBA_DSP_EVENT_H264_ENC_START_s

Type	Field	Description
UINT32	**CHANNELID**	Indicates which channel encoder is starting
UINT32	**STREAMID**	Indicates which stream is starting
UINT32	**USERDATA**	User data
UINT64	**FIRSTFRAMEPTS**	Indicates the PTS of first encoded frame
AMBA_DSP_EVENT_YUV_BUF_s	**FIRSTIPIC**	Indicates the YUV buffer of first I picture

*Table 3-38. Definition of **AMBA_DSP_EVENT_H264_ENC_START_s** for Event Handler API **AmbaDSP_EventHandlerCtrlConfig**().*

3.2.4 AmbaDSP_RegisterEventHandler

API Syntax:

AmbaDSP_RegisterEventHandler (AMBA_DSP_EVENT_ID_e EventID, AMBA_DSP_EVENT_HANDLER_f EventHandler)

Function Description:

- This function is used to register an event handler. Before calling the function, the user must call **AmbaDSP_EventHandlerCtrlConfig()** first.

Parameters:

Type	Parameter	Description
AMBA_DSP_EVENT_ID_e	**EventID**	The Event ID. Please refer to Section 3.2.3.1 for more details.
AMBA_DSP_EVENT_HANDLER_f	**EventHandler**	Pointer to the Event Handler. Please refer to Section 3.2.3.2 for more details.

*Table 3-39. Parameters for Event Handler API **AmbaDSP_RegisterEvent Handler()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 3-40. Returns for Event Handler API **AmbaDSP_RegisterEvent Handler()**.*

Example:

An example of registering one handler to one event is given below:

```
AmbaDSP_RegisterEventHandler(AMBA_DSP_EVENT_ID_ENC_CH0_MAIN_PIC_READY,
MovieRecordVideoEventHandler);
```

See Also:

None

3.2.5 AmbaDSP_UnRegisterEventHandler

API Syntax:

AmbaDSP_UnRegisterEventHandler (AMBA_DSP_EVENT_ID_e EventID, AMBA_DSP_EVENT_HANDLER_f EventHandler)

Function Description:

- This function is used to de-register an event handler.

Parameters:

Type	Parameter	Description
AMBA_DSP_EVENT_ID_e	**EventID**	The Event ID. Please refer to Section 3.2.3.1 for more details.
AMBA_DSP_EVENT_HANDLER_f	**EventHandler**	Pointer to the Event Handler. Please refer to Section 3.2.3.2 for more details.

*Table 3-41. Parameters for Event Handler API **AmbaDSP_UnRegisterEventHandler()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 3-42. Returns for Event Handler API **AmbaDSP_UnRegisterEventHandler()**.*

Example:

An example of de-registering one handler of one event is given below

```
AmbaDSP_UnRegisterEventHandler(AMBA_DSP_EVENT_ID_ENC_CH0_MAIN_PIC_READY,
MovieRecordVideoEventHandler);
```

See Also:

None

4 Video Output

4.1 Video Output: Overview

This chapter details the functions used to configure Video Output (VOUT) hardware.

4.2 Video Output: List of Functions

4.2.1 AmbaDSP_VoutReset

API Syntax:

AmbaDSP_VoutReset (AMBA_DSP_VOUT_IDX_e VoutIdx)

Function Description:

- This function is used to reset the VOUT module and to stop a specific VOUT instance. This function cannot be called if the DSP has not been initialized.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index to reset. Please refer to Section 5.2.1.1 for more details.

*Table 4-1. Parameters for VOUT API **AmbaDSP_VoutReset()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-2. Returns for VOUT **APIAmbaDSP_VoutReset()**.*

Example:

```
When the user plugs out TV, this API is called to save power.

/* Must set VOUT video source to background first */
AmbaDSP_VoutVideoSourceSel(AMBA_DSP_VOUT_TV, AMBA_DSP_VOUT_VIDEO_SRC_BACK_
COLOR);

/* Disable VOUT video layer */
AmbaDSP_VoutVideoCtrl(AMBA_DSP_VOUT_TV, 0);

AmbaDSP_VoutReset(AMBA_DSP_VOUT_TV);
```

See Also:

AmbaDSP_VoutVideoSourceSel
AmbaDSP_VoutVideoCtrl

4.2.1.1 AmbaDSP_VoutReset > AMBA_DSP_VOUT_IDX_e

Type	Description
AMBA_DSP_VOUT_LCD	LCD channel
AMBA_DSP_VOUT_TV	TV channel
AMBA_NUM_DSP_VOUT_IDX	The VOUT channel number

*Table 4-3. Definition of **AMBA_DSP_VOUT_IDX_e** for VOUT API **AmbaDSP_VoutReset()**.*

4.2.2 AmbaDSP_VoutMixerSetup

API Syntax:

AmbaDSP_VoutMixerSetup (AMBA_DSP_VOUT_IDX_e VoutIdx, AMBA_DSP_VOUT_MIXER_CONFIG_s *pConfig)

Function Description:

- This function is used to configure the Mixer parameters that cannot be changed during run-time. Note that the **AmbaDSP_VoutReset()** function must be called first.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
AMBA_DSP_VOUT_MIXER_CONFIG_s	***pConfig**	Mixer configuration. Please refer to Section 5.2.2.1 for more details.

*Table 4-4. Parameters for VOUT API **AmbaDSP_VoutMixerSetup()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-5. Returns for VOUT API **AmbaDSP_VoutMixerSetup()**.*

Example:

```
AMBA_DSP_VOUT_MIXER_CONFIG_s  VoutMixerConfigLCD;

    VoutMixerConfigLCD.ActiveWidth  = AMBA_LCD_WIDTH;
    VoutMixerConfigLCD.ActiveHeight = AMBA_LCD_HEIGHT;
    VoutMixerConfigLCD.VideoHorFlipEnable = 0;
    VoutMixerConfigLCD.Interlace = 0;
    VoutMixerConfigLCD.FrameRate = VOUT_FRAME_RATE_59_94;
    VoutMixerConfigLCD.MixerColorFormat = MIXER_IN_YUV_444_RGB;

    AmbaDSP_VoutMixerSetup(AMBA_DSP_VOUT_LCD, &VoutMixerConfigLCD);
```

See Also:

None

4.2.2.1 AmbaDSP_CoutMixerSetup > AMBA_DSP_VOUT_MIXER_CONFIG_s

Type	Parameter	Description
UINT16	**ActiveWidth**	Number of pixels per line
UINT16	**ActiveHeight**	Number of lines
UINT8	**VideoHorFlipEnable**	1: Video Data is horizontally flipped
UINT8	**Interlace**	0: Progressive 1: Interlace
AMBA_DSP_FRAME_RATE_s	**FrameRate**	Frame rate. Please refer to Section 2.2.1.4 for more details.
AMBA_DSP_VOUT_MIXER_COLOR_FORMAT_e	**MixerColorFormat**	Mixer color format. Please refer to Section 2.2.1.5 for more details.

*Table 4-6. Definition of **AMBA_DSP_VOUT_MIXER_CONFIG_s** for VOUT API **AmbaDSP_VoutMixerSetup()**.*

4.2.3 AmbaDSP_VoutMixerSetBackColor

API Syntax:

AmbaDSP_VoutMixerSetBackColor (AMBA_DSP_VOUT_IDX_e VoutIdx, UINT32 BackColorYUV)

Function Description:

- This function is used to specify the Mixer background color. This function must be called prior to **AmbaDSP_VoutMixerSetup()**.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
UINT32	**BackColorYUV**	Background color bit[0-7]: Y bit[8-15]: U bit[16-23]: V

*Table 4-7. Parameters for VOUT API **AmbaDSP_VoutMixerSetBackColor()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-8. Returns for VOUT API **AmbaDSP_VoutMixerSetBackColor()**.*

Example:

None

See Also:

AmbaDSP_VoutVideoSourceSel

4.2.4 AmbaDSP_VoutMixerSetHighlight

API Syntax:

AmbaDSP_VoutMixerSetHighlight (AMBA_DSP_VOUT_IDX_e VoutIdx, UINT8 LumaThreshold, UINT32 HighlightColorYUV)

Function Description:

- This function is used to specify the Mixer highlight settings. This function must be called prior to **AmbaDSP_VoutMixerSetup()**.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
UINT8	**LumaThreshold**	Highlight luma threshold
UINT32	**HighlightColorYUV**	Highlight color bit[0-7]: Y bit[8-15]: U bit[16-23]: V

*Table 4-9. Parameters for VOUT API **AmbaDSP_VoutMixerSetHighlight()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-10. Returns for VOUT API **AmbaDSP_VoutMixerSetHighlight()**.*

Example:

None

See Also:

None

4.2.5 AmbaDSP_VoutMixerSetCscMatrix

API Syntax:

AmbaDSP_VoutMixerSetCscMatrix (AMBA_DSP_VOUT_IDX_e VoutIdx, AMBA_DSP_VOUT_MIXER_CSC_CTRL_e CscCtrl, AMBA_DSP_VOUT_CSC_CONFIG_s *pCscConfig)

Function Description:

- This function is used to configure the Mixer color space conversion (CSC) settings. This function must be called prior to **AmbaDSP_VoutMixerSetup()**.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
AMBA_DSP_VOUT_MIXER_CSC_CTRL_e	**CscCtrl**	CSC control. Please refer to Section 5.2.5.2 for more details.
AMBA_DSP_VOUT_CSC_CONFIG_s	***pCscConfig**	CSC settings. Please refer to Section 5.2.5.1 for more details.

*Table 4-11. Parameters for VOUT API **AmbaDSP_VoutMixerSetCscMatrix()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-12. Returns for VOUT API **AmbaDSP_VoutMixerSetCscMatrix()**.*

Example:

None

See Also:

None

4.2.5.1 AmbaDSP_VoutMixerSetCsCMatrix > AMBA_DSP_VOUT_CSC_CONFIG_s

Type	Parameter	Description
float	**CscMatrix[3 * 3]**	The 3x3 matrix
INT16	**Constant[3]**	Constants
UINT16	**Clamp[6]**	Clamping values

*Table 4-13. Definition of **AMBA_DSP_VOUT_CSC_CONFIG_s** for VOUT API **AmbaDSP_VoutMixerSetCscMatrix()**.*

4.2.5.2 AmbaDSP_VoutMixerSetCsCMatrix > AMBA_DSP_VOUT_MIXER_CSC_CTRL_e

Type	Description
MIXER_CSC_ DISABLE	Mixer CSC Disabled
MIXER_CSC_ FOR_VIDEO	Mixer CSC Enabled for Video layer
MIXER_CSC_ FOR_OSD	Mixer CSC Enabled for OSD

*Table 4-14. Definition of **AMBA_DSP_VOUT_MIXER_CSC_CTRL_e** for VOUT API **AmbaDSP_VoutMixerSetCscMatrix()**.*

4.2.6 AmbaDSP_VoutVideoCtrl

API Syntax:

AmbaDSP_VoutVideoCtrl (AMBA_DSP_VOUT_IDX_e VoutIdx, int Enable)

Function Description:

- This function is used to control the VOUT video layer.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
int	**Enable**	0: Turn off VOUT video layer 1: Turn on VOUT video layer

*Table 4-15. Parameters for VOUT API **AmbaDSP_VoutVideoCtrl()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-16. Returns for VOUT API **AmbaDSP_VoutVideoCtrl()**.*

Example:

None

See Also:

None

4.2.7 AmbaDSP_VoutVideoSourceSel

API Syntax:

AmbaDSP_VoutVideoSourceSel (AMBA_DSP_VOUT_IDX_e VoutIdx, int VideoSource)

Function Description:

- This function is used to configure settings for the video source.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
int	**Source**	VideoSource: 0: AMBA_DSP_VOUT_VIDEO_SRC_DEFAULT_IMG 1: AMBA_DSP_VOUT_VIDEO_SRC_BACK_COLOR

*Table 4-17. Parameters for VOUT API **AmbaDSP_VoutVideoSourceSel()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-18. Returns for VOUT API **AmbaDSP_VoutVideoSourceSel()**.*

Example:

If the user wants to show a YUV image, the DSP default image can be set and called as shown below.

```
AmbaDSP_VoutVideoSourceSel(AMBA_DSP_VOUT_LCD, AMBA_DSP_VOUT_VIDEO_SRC_DE-
FAULT_IMG);
```

If the user wants to show a background color, set the Mixer background color call as shown below.

```
AmbaDSP_VoutVideoSourceSel(AMBA_DSP_VOUT_LCD, AMBA_DSP_VOUT_VIDEO_SRC_BACK_
COLOR);
```

See Also:

AmbaDSP_VoutDefaultImgSetup
AmbaDSP_VoutMixerSetBackColor

4.2.8 AmbaDSP_VoutVideoConfig

API Syntax:

AmbaDSP_VoutVideoConfig (AMBA_DSP_VOUT_IDX_e VoutIdx, AMBA_DSP_ROTATE_FLIP_e RotateFlip)

Function Description:

- This function is used to configure Video flip and rotation settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
AMBA_DSP_VOUT_ROTATE_e	**RotateFlip**	Rotation and flip setting. Please refer to Section 5.2.8.1 below for more details.

*Table 4-19. Parameters for VOUT API **AmbaDSP_VoutVideoCtrl()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-20. Returns for VOUT API **AmbaDSP_VoutVideoCtrl()**.*

Example:

None

See Also:

None

4.2.8.1 AmbaDSP_VoutVideoConfig > AMBA_DSP_ROTATE_FLIP_e

Type	Description
AMBA_DSP_ROTATE_0	No rotation and no flip
AMBA_DSP_ROTATE_0_HORZ_FLIP	Horizontally flip
AMBA_DSP_ROTATE_90	Rotate 90 degree.
AMBA_DSP_ROTATE_90_VERT_FLIP	Rotate 90 degree and vertically flip.
AMBA_DSP_ROTATE_180	Rotate 180 degree.
AMBA_DSP_ROTATE_180_HORZ_FLIP	Rotate 180 degree and horizontally flip.
AMBA_DSP_ROTATE_270	Rotate 270 degree.
AMBA_DSP_ROTATE_270_VERT_FLIP	Rotate 270 degree and vertically flip.

*Table 4-21. Definition of **AMBA_DSP_ROTATE_FLIP_e** for VOUT API **AmbaDSP_VoutVideoConfig()**.*

4.2.9 AmbaDSP_VoutVideoWindowSetup

API Syntax:

AmbaDSP_VoutVideoWindowSetup (AMBA_DSP_VOUT_IDX_e VoutIdx, AMBA_DSP_VOUT_WINDOW_s *pWindow)

Function Description:

- This function is used to configure settings for the Video window.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
AMBA_DSP_VOUT_WINDOW_s	***pWindow**	VOUT video window. Please refer to Section 5.2.9.1 for more details.

*Table 4-22. Parameters for VOUT API **AmbaDSP_VoutVideoWindowSetup()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-23. Returns for VOUT API **AmbaDSP_VoutVideoWindowSetup()**.*

Example:

```
AMBA_DSP_WINDOW_s VideoWin;

memset(&VideoWin, 0, sizeof(VideoWin));
VideoWin.Width = 960;
VideoWin.Height = 480;
VideoWin.OffsetX = 0;
VideoWin.OffsetY = 0;
if (AmbaDSP_VoutVideoWindowSetup(AMBA_DSP_VOUT_LCD, &VideoWin) == NG)
return NG;
```

See Also:

None

4.2.9.1 AmbaDSP_VoutVideoWindowSetup > AMBA_DSP_WINDOW_s

Type	Field	Description
UINT16	**OffsetX**	Horizontal offset of the window
UINT16	**OffsetY**	Vertical offset of the window
UINT16	**Width**	Number of pixels per line in the window
UINT16	**Height**	Number of lines in the window

*Table 4-24. Dedinition of **AMBA_DSP_WINDOW_s** for VOUT API **AmbaDSP_VoutVideoWindowSetup()**.*

4.2.10 AmbaDSP_VoutVideoGetState

API Syntax:

AmbaDSP_VoutVideoGetState (AMBA_DSP_VOUT_IDX_e VoutIdx)

Function Description:

- This function is used to check whether VOUT video is active or inactive.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.

*Table 4-25. Parameters for VOUT API **AmbaDSP_VoutVideoGetState()**.*

Returns:

Return	Description
0	VOUT video layer is off
-1	VOUT video layer is on

*Table 4-26. Returns for VOUT API **AmbaDSP_VoutVideoGetState()**.*

Example:

None

See Also:

None

4.2.11 AmbaDSP_VoutDefaultImgSetup

API Syntax:

AmbaDSP_VoutDefaultImgSetup (AMBA_DSP_VOUT_IDX_e VoutIdxD, AMBA_DSP_VOUT_DEFAULT_IMG_CONFIG_s *pDefaultImgConfig)

Function Description:

- This function is used to set the default image.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
AMBA_DSP_VOUT_DEFAULT_IMG_CONFIG_s	***pDefaultImgConfig**	Default image configuration. Please refer to Section 2.2.1.7 for more details.

*Table 4-27. Parameters for VOUT API **AmbaDSP_VoutDefaultImgSetup()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-28. Returns for VOUT API **AmbaDSP_VoutDefaultImgSetup()**.*

Example:

```
AMBA_DSP_VOUT_DEFAULT_ IMG_CONFIG_s VoutDefaultImgConfig;

VoutDefaultImgConfig.FieldRepeat = 0;
VoutDefaultImgConfig.Pitch = AMBA_LCD_WIDTH;
VoutDefaultImgConfig.pBaseAddrY = AmbaDSP_DefaultImgLumaLCD;
VoutDefaultImgConfig.pBaseAddrUV = AmbaDSP_DefaultImgChromaLCD;

AmbaDSP_VoutDefaultImgSetup(AMBA_DSP_VOUT_LCD, &VoutDefaultImgConfig);
```

See Also:

AmbaDSP_VoutVideoSourceSel

4.2.12 AmbaDSP_VoutOsdCtrl

API Syntax:

AmbaDSP_VoutOsdCtrl (AMBA_DSP_VOUT_IDX_e VoutIdx, int Enable)

Function Description:

- This function is used to control the VOUT OSD layer.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
int	**Enable**	0: Turn off VOUT OSD layer 1: Turn on VOUT OSD layer

*Table 4-29. Parameters for VOUT API **AmbaDSP_VoutOsdCtrl()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 4-30. Returns for VOUT API **AmbaDSP_VoutOsdCtrl()**.*

Example:

None

See Also:

None

4.2.13 AmbaDSP_VoutOsdBufSetup

API Syntax:

AmbaDSP_VoutOsdBufSetup (AMBA_DSP_VOUT_IDX_e VoutIdx, AMBA_DSP_VOUT_OSD_BUF_CONFIG_s *pBufConfig)

Function Description:

- This function is used to set OSD buffer information.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
AMBA_DSP_VOUT_OSD_BUF_CONFIG_s	***pBufConfig**	OSD base address pointer. Please refer to Section 2.2.1.6 for more details.

*Table 4-31. Parameters for VOUT API **AmbaDSP_VoutOsdBufSetup()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 4-32. Returns for VOUT API **AmbaDSP_VoutOsdBufSetup()**.*

Example:

```
UINT32 AmbaOSD_Buf[AMBA_LCD_WIDTH * AMBA_LCD_HEIGHT];
AMBA_DSP_VOUT_OSD_BUF_CONFIG_s VoutOsdBufConfigLCD;
/* Please refer to example of Section 2.2.1.1 */

VoutOsdBufConfigLCD.FieldRepeat = 0;
VoutOsdBufConfigLCD.Pitch = AMBA_LCD_WIDTH << 2; /* 32-bit OSD */
VoutOsdBufConfigLCD.pBaseAddr = (void *) AmbaOSD_Buf;
AmbaDSP_VoutOsdBufSetup(AMBA_DSP_VOUT_LCD, &VoutOsdBufConfigLCD);
```

See Also:

AmbaDSP_Init

4.2.14 AmbaDSP_VoutOsdWindowSetup

API Syntax:

AmbaDSP_VoutOsdWindowSetup (AMBA_DSP_VOUT_IDX_e VoutIdx, AMBA_DSP_VOUT_WINDOW_s *pWindow)

Function Description:

- This function is used to set up the OSD window with the scaler disabled.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
AMBA_DSP_VOUT_WINDOW_s	***pOsdWindow**	OSD window information. Please refer to Section 5.2.9.1 for more details.

*Table 4-33. Parameters for VOUT API **AmbaDSP_VoutOsdWindowSetup()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-34. Returns for VOUT API **AmbaDSP_VoutOsdWindowSetup()**.*

Example:

```
AMBA_DSP_WINDOW_s OsdWin;

memset(&OsdWin, 0, sizeof(AMBA_DSP_WINDOW_s);
OsdWin.OffsetX = OsdWn.OffsetY = 0;
OsdWin.Wisth = AMBA_LCD_WIDTH;
OsdWin.Height = AMBA_LCD_HEIGHT;
AmbaDSP_VoutOsdWindowSetup(AMBA_DSP_VOUT_LCD, &OsdWin);
```

See Also:

None

4.2.15 AmbaDSP_VoutOsdScalerSetup

API Syntax:

AmbaDSP_VoutOsdScalerSetup (AMBA_DSP_VOUT_IDX_e VoutIdx, AMBA_DSP_VOUT_OSD_SCALER_s *pScalerCfg)

Function Description:

- This function is used to set up the OSD window with the scaler enabled.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
AMBA_DSP_VOUT_OSD_SCALER_s	***pScalerCfg**	OSD input/output window information. Please refer to Section 5.2.15.1 for more details.

*Table 4-35. Parameters for VOUT API **AmbaDSP_VoutOsdScalerSetup()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-36. Returns for VOUT API **AmbaDSP_VoutOsdScalerSetup ()**.*

Example:

```
AMBA_DSP_VOUT_OSD_SCALER_s OsdScaler;

OsdScaler.OffsetX = 0;
OsdScaler.OffsetY = (AMBA_MAX_TV_HEIGHT - (AMBA_MAX_TV_WIDTH * AMBA_LCD_
HEIGHT) / AMBA_LCD_WIDTH) / 2;
OsdScaler.Width = AMBA_MAX_TV_WIDTH;
OsdScaler.Height = (AMBA_MAX_TV_WIDTH * AMBA_LCD_HEIGHT) / AMBA_LCD_WIDTH;  /*
AMBA_MAX_TV_HEIGHT */
OsdScaler.ScalerInputWidth = AMBA_LCD_WIDTH;
OsdScaler.ScalerInputHeight = AMBA_LCD_HEIGHT;

AmbaDSP_VoutOsdScalerSetup(AMBA_DSP_VOUT_TV,  &OsdScaler)
```

See Also:

None

4.2.15.1 AmbaDSP_VoutOsdScalerSetup > AMBA_DSP_VOUT_OSD_SCALER_s

Type	Field	Description
UINT16	**OffsetX**	Horizontal offset of output OSD window related to the active window
UINT16	**OffsetY**	Vertical offset of output OSD window related to the active window
UINT16	**Width**	Number of pixels per line in the output OSD window
UINT16	**Height**	Number of lines in the output OSD window
UINT16	**ScalerInputWidth**	OSD Scaler input width
UINT16	**ScalerInputHeight**	OSD Scaler input height

*Table 4-37. Definition of **AMBA_DSP_VOUT_OSD_SCALER_s** for VOUT API **AmbaDSP_VoutOsdScalerSetup()**.*

4.2.16 AmbaDSP_VoutOsdSetDataFormat

API Syntax:

AmbaDSP_VoutOsdSetDataFormat (AMBA_DSP_VOUT_IDX_e VoutIdx, AMBA_DSP_VOUT_OSD_DATA_FORMAT_e DataFormat)

Function Description:

- This function is used to specify the OSD data format.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
AMBA_DSP_VOUT_OSD_DATA_FORMAT_e	**DataFormat**	OSD data format. Please refer to Section 5.2.16.1 for more details.

*Table 4-38. Parameters for VOUT API **AmbaDSP_VoutOsdSetDataFormat()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-39. Returns for VOUT API **AmbaDSP_VoutOsdSetDataFormat()**.*

Example:

None

See Also:

None

4.2.16.1 AmbaDSP_VoutOsdSetDataFormat > AMBA_DSP_VOUT_OSD_DATA_FORMAT_e

Type	Description
OSD_8BIT_CLUT_MODE	8-bit color look-up table mode
OSD_16BIT_VYU_RGB_565	VYU_RGB_565
OSD_16BIT_UYV_BGR_565	UYV_BGR_565

Type	Description
OSD_16BIT_ AYUV_4444	AYUV_4444
OSD_16BIT_ RGBA_4444	RGBA_4444
OSD_16BIT_ BGRA_4444	BGRA_4444
OSD_16BIT_ ABGR_4444	ABGR_4444
OSD_16BIT_ ARGB_4444	ARGB_4444
OSD_16BIT_ AYUV_1555	AYUV_1555
OSD_16BIT_ YUV_1555	YUV_1555, MSB ignored
OSD_16BIT_ RGBA_5551	RGBA_5551
OSD_16BIT_ BGRA_5551	BGRA_5551
OSD_16BIT_ ABGR_1555	ABGR_1555
OSD_16BIT_ ARGB_1555	ARGB_1555
OSD_32BIT_ AYUV_8888	AYUV_8888. Set the value as 27
OSD_32BIT_ RGBA_8888	RGBA_8888
OSD_32BIT_ BGRA_8888	BGRA_8888
OSD_32BIT_ ABGR_8888	ABGR_8888
OSD_32BIT_ ARGB_8888	ARGB_8888

*Table 4-40. Definition of **AMBA_DSP_VOUT_OSD_DATA_FORMAT_e** for VOUT API **AmbaDSP_VoutOsdSetDataFormat()**.*

4.2.17 AmbaDSP_VoutOsdSetTransparent

API Syntax:

AmbaDSP_VoutOsdSetTransparent (AMBA_DSP_VOUT_IDX_e VoutIdx, int Enable, UINT16 Color)

Function Description:

- This function is used to specify the OSD transparent color (only for 16-bit direct mode).

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
int	**Enable**	0: Disable 1: Enable
UINT16	**Color**	Color that will be treated as transparent. Please refer to Section 5.2.16.1 for details on the 16-bit direct mode.

*Table 4-41. Parameters for VOUT API **AmbaDSP_VoutOsdSetTransparent()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-42. Returns for VOUT API **AmbaDSP_VoutOsdSetTransparent()**.*

Example:

None

See Also:

None

4.2.18 AmbaDSP_VoutOsdSetBlend

API Syntax:

AmbaDSP_VoutOsdSetBlend (AMBA_DSP_VOUT_IDX_e VoutIdx, UINT8 GlobalBlend)

Function Description:

- This function is used to specify the OSD global blending value.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
UINT8	**GlobalBlend**	OSD global blending value: Valid value 0 – 255.

*Table 4-43. Parameters for VOUT API **AmbaDSP_VoutOsdSetBlend()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-44. Returns for VOUT API **AmbaDSP_VoutOsdSetBlend()**.*

Example:

None

See Also:

None

4.2.19 AmbaDSP_VoutOsdSetCLUT

API Syntax:

AmbaDSP_VoutOsdSetCLUT (AMBA_DSP_VOUT_IDX_e VoutIdx, UINT32 *pCLUT)

Function Description:

- This function is used to set the user OSD color lookup table (CLUT).

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
UINT32	***pCLUT**	OSD color lookup table

*Table 4-45. Parameters for VOUT API **AmbaDSP_VoutOsdSetCLUT()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-46. Returns for VOUT API **AmbaDSP_VoutOsdSetCLUT()**.*

Example:

None

See Also:

None

4.2.20 AmbaDSP_VoutOsdSwapByteCtrl

API Syntax:

AmbaDSP_VoutOsdSwapByteCtrl (AMBA_DSP_VOUT_IDX_e VoutIdx, int Enable)

Function Description:

- This function is used to swap the OSD byte order to be compatible with OpenGL.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**Voutidx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
int	**Enable**	Enable swap byte: 0: Disable 1: Enable

*Table 4-47. Parameters for VOUT API **AmbaDSP_VoutOsdSwapByteCtrl()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-48. Returns for VOUT API **AmbaDSP_VoutOsdSwapByteCtrl()**.*

Example:

None

See Also:

None

4.2.21 AmbaDSP_VoutOsdGetState

API Syntax:

AmbaDSP_VoutOsdGetState (AMBA_DSP_VOUT_IDX_e VoutIdx)

Function Description:

- This function is used to check whether the OSD is active or inactive.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**Voutidx**	VOUT index. Please refer to Section 5.2.1.1 for more details.

*Table 4-49. Parameters for VOUT API **AmbaDSP_VoutOsdGetState()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-50. Returns for VOUT API **AmbaDSP_VoutOsdGetState()**.*

Example:

None

See Also:

None

4.2.22 AmbaDSP_VoutDisplayDigitalSetColorSequence

API Syntax:

AmbaDSP_VoutDisplayDigitalSetColorSequence (AMBA_DSP_VOUT_LCD_COLOR_ORDER_e EvenLineColor, AMBA_DSP_VOUT_LCD_COLOR_ORDER_e OddLineColor)

Function Description:

- This function is used to vertically flip the LCD display (program LCD driver) during the LCD VOUT color sequence setup.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_LCD_COLOR_ORDER_e	**EvenLineColor**	Even line color sequence. Please refer to Section 5.2.1.1 for more details.
AMBA_DSP_VOUT_LCD_COLOR_ORDER_e	**OddLineColor**	Odd line color sequence. Please refer to Section 5.2.22.1 for more details.

*Table 4-51. Parameters for VOUT API **AmbaDSP_VoutDisplayDigitalSetColorSequence()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-52. Returns for VOUT API **AmbaDSP_VoutDisplayDigitalSetColorSequence()**.*

Example:

None

See Also:

None

4.2.22.1 AmbaDSP_VoutDisplayDigitalSetColorSequence > AMBA_DSP_VOUT_LCD_COLOR_ORDER_e

Type	Description
AMBA_DSP_VOUT_LCD_COLOR_RGB	R-G-B
AMBA_DSP_VOUT_LCD_COLOR_BGR	B-G-R (reversed R-G-B)
AMBA_DSP_VOUT_LCD_COLOR_GBR	G-B-R
AMBA_DSP_VOUT_LCD_COLOR_RBG	R-B-G (reversed G-B-R)
AMBA_DSP_VOUT_LCD_COLOR_BRG	B-R-G
AMBA_DSP_VOUT_LCD_COLOR_GRB	G-R-B (reversed B-R-G)

*Table 4-53. Definition of **AMBA_DSP_VOUT_LCD_COLOR_ORDER_e** for VOUT API **AmbaDSP_VoutDisplayDigitalSetColorSequence()**.*

4.2.23 AmbaDSP_VoutDisplayDigitalSetGamma

API Syntax:

AmbaDSP_VoutDisplayDigitalSetGamma (void *pGammaTable)

Function Description:

- This function is used to set the digital gamma value for the display.

Parameters:

Type	Parameter	Description
void	***pGammaTable**	3K bytes gamma table pointer

*Table 4-54. Parameters for VOUT API **AmbaDSP_VoutDisplayDigitalSetGamma()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 4-55. Returns for VOUT API **AmbaDSP_VoutDisplayDigitalSetGamma()**.*

Example:

None

See Also:

None

4.2.24 AmbaDSP_VoutDisplayDigitalGammaCtrl

API Syntax:

AmbaDSP_VoutDisplayDigitalGammaCtrl (int Enable)

Function Description:

- This function is used to control the Display digital gamma. This function can only be called after **AmbaDSP_VoutDisplayDigitalSetup()**.

Parameters:

Type	Parameter	Description
int	**Enable**	0: Turn off VOUT digital display gamma 1: Turn on VOUT digital display gamma

*Table 4-56. Parameters for VOUT API **AmbaDSP_VoutDisplayDigitalGammaCtrl()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 4-57. Returns for VOUT API **AmbaDSP_VoutDisplayDigitalGammaCtrl()**.*

Example:

None

See Also:

None

4.2.25 AmbaDSP_VoutDisplayTimingSetup

API Syntax:

AmbaDSP_VoutDisplayTimingSetup (AMBA_DSP_VOUT_IDX_e VoutIdx, AMBA_DSP_VOUT_DISPLAY_TIMING_CONFIG_s *pDisplayTimingConfig)

Function Description:

- This function is used to configure the VOUT Display timing settings.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index. Please refer to Section 5.2.1.1 for more details.
AMBA_DSP_VOUT_DISPLAY_TIMING_CONFIG_s	***pDisplayTimingConfig**	Display timing configuration. Please refer to Section 5.2.25.1 for more details.

*Table 4-58. Parameters for VOUT API **AmbaDSP_VoutDisplayTimingSetup()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-59. Returns for VOUT API **AmbaDSP_VoutDisplayTimingSetup()**.*

Example:

None

See Also:

None

4.2.25.1 AmbaDSP_VoutTiming_setup > AMBA_DSP_VOUT_DISPLAY_TIMING_CONFIG_s

Type	Field	Description
UINT32	**PixelClock**	Pixel clock (Hz)
UINT16	**FrameWidth**	Total Number of clock cycles per line
UINT16	**FrameHeight**	Total Number of lines per frame
UINT16	**FrameActiveColStart**	Start column of active region
UINT16	**FrameActiveColWidth**	Column width of active region
UINT16	**FrameActiveRowStart**	Start row of active region

Type	Field	Description
UINT16	**FrameActiveRowHeight**	Row height of active region
UINT8	**Rotation**	Clockwise rotation (degree)
UINT8	**Interlace**	0: Progressive scan 1: Interlace scan

*Table 4-60. Definition of **AMBA_DSP_VOUT_DISPLAY_TIMING_CONFIG_s** for VOUT API **AmbaDSP_VoutDisplay-TimingSetup()**.*

4.2.26 AmbaDSP_VoutDisplaySetCscMatrix

API Syntax:

AmbaDSP_VoutDisplaySetCscMatrix (AMBA_DSP_VOUT_TYPE_e VoutType, AMBA_DSP_VOUT_CSC_CONFIG_s *pCscConfig)

Function Description:

- This function is used to configure the Display color space conversion (CSC) matrix. This function can only be called after either **AmbaDSP_VoutDisplayDigitalSetup()**, **AmbaDSP_VoutDisplayHdmiSetup()** or **AmbaDSP_VoutDisplayCvbsSetup()** functions are called.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_TYPE_e	**VoutType**	VOUT type. Please refer to Section 5.2.26.1 for more details.
AMBA_DSP_VOUT_CSC_CONFIG_s	**pCscConfig***	Display CSC configuration. Please refer to Section 5.2.5.1 for more details.

*Table 4-61. Parameters for VOUT API **AmbaDSP_VoutDisplaySetCscMatrix()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-62. Returns for VOUT API **AmbaDSP_VoutDisplaySetCscMatrix()**.*

Example:

None

See Also:

None

4.2.26.1 AmbaDSP_VoutDisplaySetCscMatrix > AMBA_DSP_VOUT_TYPE_e

Type	Description
AMBA_DSP_VOUT_DIGITAL	Digital VOUT
AMBA_DSP_VOUT_CVBS	Analog CVBS VOUT
AMBA_DSP_VOUT_HDMI	HDMI VOUT

*Table 4-63. Definition of **AMBA_DSP_VOUT_TYPE_e** for VOUT API **AmbaDSP_VoutDisplaySetCscMatrix()**.*

4.2.27 AmbaDSP_VoutDisplayDigitalSetup

API Syntax:

AmbaDSP_VoutDisplayDigitalSetup (AMBA_DSP_VOUT_DISPLAY_DIGITAL_CONFIG_s *pDisplayDigitalConfig)

Function Description:

- This function is used to configure the VOUT digital display registers that cannot be changed during run-time. Note that the **AmbaDSP_VoutReset()** function must be called first.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_DISPLAY_DIGITAL_CONFIG_s	***pDisplayDigitalConfig**	Digital display configuration. Please refer to Section 5.2.27.1 for more details.

*Table 4-64. Parameters for VOUT API **AmbaDSP_VoutDisplayDigitalSetup()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 4-65. Returns for VOUT API **AmbaDSP_VoutDisplayDigitalSetup ()**.*

Example:

None

See Also:

None

4.2.27.1 AmbaDSP_VoutDisplayDigitalSetup > AMBA_DSP_VOUT_DISPLAY_DIGITAL_CONFIG_s

Type	Field	Description
AMBA_DSP_VOUT_DISPLAY_SYNC_CTRL_s	**SyncCtrl**	Display sync control. Please refer to Section 5.2.27.2 for more details.
AMBA_DSP_VOUT_DIGITAL_OUTPUT_MODE_e	**OutputMode**	Digital Output Mode. Please refer to Section 5.2.27.3 for more details.
AMBA_DSP_VOUT_PIXEL_CLOCK_OUTPUT_e	**DeviceClock**	Source clock frequency of the LCD device. Please refer to Section 5.2.27.4 for more details.
AMBA_DSP_VOUT_LCD_COLOR_ORDER_e	**EvenLineColor**	RGB color sequence of even lines. Please refer to Section 5.2.22.1 for more details.
AMBA_DSP_VOUT_LCD_COLOR_ORDER_e	**OddLineColor**	RGB color sequence of odd lines. Please refer to Section 5.2.22.1 for more details.
AMBA_DSP_VOUT_SIGNAL_POLARITY_e	**ClkSampleEdge**	Sample clock edge. Please refer to Section 5.2.27.5 for more details.
AMBA_DSP_VOUT_SIGNAL_POLARITY_e	**LineSyncPolarity**	Line sync signal polarity. Please refer to Section 5.2.27.5 for more details.
AMBA_DSP_VOUT_SIGNAL_POLARITY_e	**FrameSyncPolarity**	Frame sync signal polarity. Please refer to Section 5.2.27.5 for more details.
AMBA_DSP_VOUT_SIGNAL_POLARITY_e	**LineValidPolarity**	Line valid signal polarity. Please refer to Section 5.2.27.5 for more details.
AMBA_DSP_VOUT_SIGNAL_POLARITY_e	**FrameValidPolarity**	Frame sync signal polarity. Please refer to Section 5.2.27.5 for more details.
UINT8	**Interlace**	0: Progressive scan 1: Interlaced scan
UINT16	**FrameWidth**	Horizontal total pixels
UINT16	**FrameHeight**	Vertical total lines
UINT16	**RowTime**	Row time in clock cycle
UINT16	**Bt656VBitStart**	Define Start Row of V-Bit Assertion
UINT16	**Bt656VBitEnd**	Define End Row of V-Bit Assertion
UINT16	**Bt656SavStart**	Define Start Location of SAV Code

*Table 4-66. Definition of **AMBA_DSP_VOUT_DISPLAY_DIGITAL_CONFIG_s** for VOUT API **AmbaDSP_VoutDisplayDigitalSetup()**.*

4.2.27.2 AmbaDSP_VoutDisplayDigitalSetup > AMBA_DSP_VOUT_DISPLAY_SYNC_CTRL_s

Type	Field	Description
UINT16	**HSyncColStart**	Start column of HSync pulse
UINT16	**HSyncColEnd**	End column of HSync pulse
UINT16	**VSyncColStart**	Start column of VSync pulse
UINT16	**VSyncColEnd**	End column of VSync pulse
UINT16	**VSyncRowStart**	Start row of VSync pulse
UINT16	**VSyncRowEnd**	End row of VSync pulse

*Table 4-67. Definition of **AMBA_DSP_VOUT_DISPLAY_SYNC_CTRL_s** for VOUT API **AmbaDSP_VoutDisplayDigitalSetup()**.*

4.2.27.3 AmbaDSP_VoutDisplayDigitalSetup > AMBA_DSP_VOUT_DIGITAL_OUTPUT_MODE_e

Type	Description
AMBA_DSP_VOUT_LCD_1COLOR_PER_DOT	LCD - Single color per pixel
AMBA_DSP_VOUT_LCD_3COLORS_PER_DOT	LCD - 3 colors per pixel, no dummy clock
AMBA_DSP_VOUT_LCD_3COLORS_DUMMY_PER_DOT	LCD - 3 colors per pixel, dummy clock
AMBA_DSP_VOUT_LCD_RGB565	LCD - RGB 5:6:5
AMBA_DSP_VOUT_BT656	BT.656
AMBA_DSP_VOUT_BT601_16B	BT.601 (16-bit, YCbCr 4:2:2)
AMBA_DSP_VOUT_BT601_24B	BT.601 (24-bit, YCbCr 4:4:4 or RGB)
AMBA_DSP_VOUT_BT601_8B	BT.601 (8-bit, in Cb-Y-Cr-Y order)
AMBA_DSP_VOUT_CFA_BAYER	Bayer CFA

*Table 4-68. Definition of **AMBA_DSP_VOUT_DIGITAL_OUTPUT_MODE_e** for VOUT API **AmbaDSP_VoutDisplayDigitalSetup()**.*

4.2.27.4 AmbaDSP_VoutDisplayDigitalSetup > AMBA_DSP_VOUT_PIXEL_CLOCK_OUTPUT_e

Type	Description
AMBA_DSP_VOUT_PIXEL_CLOCK_NONE	No pixel clock output
AMBA_DSP_VOUT_PIXEL_CLOCK_FULL_DCLK	Same as data clock frequency
AMBA_DSP_VOUT_PIXEL_CLOCK_HALF_DCLK	Half of data clock frequency
AMBA_DSP_VOUT_PIXEL_CLOCK_QUARTER_DCLK	Quarter of data clock frequency

*Table 4-69. Definition of **AMBA_DSP_VOUT_PIXEL_CLOCK_OUTPUT_e** for VOUT API **AmbaDSP_VoutDisplayDigitalSetup()**.*

4.2.27.5 AmbaDSP_VoutDisplayDigitalSetup > AMBA_DSP_VOUT_SIGNAL_POLARITY_e

Type	Description
AMBA_DSP_VOUT_ACTIVE_HIGH	Positive pulse or rising edge
AMBA_DSP_VOUT_ACTIVE_LOW	Negative pulse or falling edge

*Table 4-70. Definition of **AMBA_DSP_VOUT_SIGNAL_POLARITY_e** for VOUT API **AmbaDSP_VoutDisplayDigitalSetup()**.*

4.2.28 AmbaDSP_VoutDisplayHdmiSetup

API Syntax:

AmbaDSP_VoutDisplayHdmiSetup (AMBA_DSP_VOUT_DISPLAY_HDMI_CONFIG_s *pDisplayHdmi-Config)

Function Description:

- This function is used to configure the VOUT HDMI display registers that cannot be changed during run-time. Note that the **AmbaDSP_VoutReset()** function must be called first.

Parameters:

Type	Parameter	Description
AMBA_DSP_ VOUT_DIS-PLAY_HDMI_ CONFIG_s	***pDisplayHdmiConfig**	Write to Group command buffer. Please refer to Section 5.2.28.1 for more details.

*Table 4-71. Parameters for VOUT API **AmbaDSP_VoutDisplayHdmiSetup()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 4-72. Returns for VOUT API **AmbaDSP_VoutDisplayHdmiSetup()**.*

Example:

None

See Also:

None

4.2.28.1 AmbaDSP_VoutDisplay_Hdmi_Setup > AMBA_DSP_VOUT_DISPLAY_HDMI_CONFIG_s

Type	Field	Description
AMBA_DSP_VOUT_DISPLAY_SYNC_CTRL_s	**SyncCtrl**	Display sync control. Please refer to Section 5.2.27.2 for more details.
AMBA_DSP_VOUT_HDMI_OUTPUT_MODE_e	**OutputMode**	HDMI Output Mode Register. Please refer to Section 5.2.28.2 for more details.
AMBA_DSP_VOUT_SIGNAL_POLARITY_e	**LineSyncPolarity**	Line sync signal polarity. Please refer to Section 5.2.27.5 for more details.
AMBA_DSP_VOUT_SIGNAL_POLARITY_e	**FrameSyncPolarity**	Frame sync signal polarity. Please refer to Section 5.2.27.5 for more details.
UINT8	**Interlace**	0: Progressive scan 1: Interlaced scan
UINT16	**RowTime**	Row time in clock cycles

*Table 4-73. Definition of **AMBA_DSP_VOUT_DISPLAY_HDMI_CONFIG_s** for VOUT API **AmbaDSP_VoutDisplayHdmiSetup()**.*

4.2.28.2 AmbaDSP_VoutDisplay_Hdmi_Setup > AMBA_DSP_VOUT_HDMI_OUTPUT_MODE_e

Type	Description
AMBA_DSP_VOUT_HDMI_YCC444_8B	HDMI - YCbCr 4:4:4 8 bits per channel
AMBA_DSP_VOUT_HDMI_RGB444_8B	HDMI - RGB 4:4:4 8 bits per channel
AMBA_DSP_VOUT_HDMI_YCC422_12B	HDMI - YCbCr 4:2:2 12 bits per channel

*Table 4-74. Definition of **AMBA_DSP_VOUT_HDMI_OUTPUT_MODE_e** for VOUT API **AmbaDSP_VoutDisplayHdmiSetup()**.*

4.2.29 AmbaDSP_VoutDisplayCvbsSetup

API Syntax:

AmbaDSP_VoutDisplayCvbsSetup (AMBA_DSP_VOUT_CVBS_TV_SYSTEM_e CvbsTvSystem)

Function Description:

- This function is used to configure the VOUT CVBS display registers that cannot be changed during run-time. Note that the **AmbaDSP_VoutReset()** function must be called first.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_CVBS_TV_SYSTEM_e	**CvbSTvSystem**	TV System AMBA_DSP_VOUT_CVBS_NTSC: NTSC system AMBA_DSP_VOUT_CVBS_PAL: PAL system

*Table 4-75. Parameters for VOUT API **AmbaDSP_VoutDisplayCvbsSetup()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 4-76. Returns for VOUT API **AmbaDSP_VoutDisplayCvbsSetup()**.*

Example:

None

See Also:

None

5 Video Input

5.1 Video Input: Overview

This chapter details the functions used to configure the Video Input (VIN) hardware.

5.2 Video Input: List of Functions

5.2.1 AmbaDSP_VinConfigSLVS

API Syntax:

AmbaDSP_VinConfigSLVS (int VinID, AMBA_DSP_VIN_SLVS_CONFIG_s *pVinSlvsConfig)

Function Description:

- This function is used to configure the VIN SLVS interface.

Parameters:

Type	Parameter	Description
Int	**VinID**	VIN index
AMBA_DSP_VIN_SLVS_CONFIG_s	***pVinSlvsConfig**	Pointer to the VIN SLVS configuration. Please refer to Section 5.2.1.1 for more details.

*Table 5-1. Parameters for VIN API **AmbaDSP_VinConfigSLVS()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 5-2. Returns for VIN API **AmbaDSP_VinConfigSLVS()**.*

Example:

None

See Also:

None

5.2.1.1 AmbaDSP_VinConfigSLVS > AMBA_DSP_VIN_SLVS_CONFIG_s

Type	Field	Description
AMBA_DSP_VIN_INFO_s	**Info**	Basic information about pixel data. Please refer to Section for more details.
AMBA_DSP_VIN_SLVS_SYNC_CODE_s	**SyncDetectCtrl**	Embedded sync code description. Please refer to Section for more details.
AMBA_DSP_VIN_RX_HV_SYNC_PAIR_e	**RxHvSyncCtrl**	Information about sync signal control. Please refer to Section for more details.
AMBA_DSP_VIN_TRIGGER_PULSE_s	**VinTrigPulse[2]**	VIN trigger pulse setting. Please refer to Section 5.2.1.11 for more details.

Type	Field	Description
AMBA_DSP_VIN_VOUT_SYNC_s	**VinVoutSync[2]**	VIN-VOUT synchronization setting. Please refer to Section for more details.
AMBA_DSP_VIN_SLVS_PIN_PAIR_e	***pLaneMapping**	SLVS data lane mapping. Please refer to Section 5.2.1.10 for more details.
UINT8	**NumActiveLanes**	Number of active data lanes. For main VIN, it supports 1-10 lanes. For 2nd VIN, it supports 1-2 lanes.
-	-	-
-	-	-
-	-	-
-	-	-
-	-	-
-	-	-
-	-	-
-	-	-

*Table 5-3. Definition of **AMBA_DSP_VIN_SLVS_CONFIG_s** for VIN API **AmbaDSP_VinConfigSLVS()**.*

5.2.1.2 AmbaDSP_VinConfigSLVS > AMBA_DSP_VIN_INFO_s

Type	Field	Description
AMBA_DSP_FRAME_RATE_s	**FrameRate**	Expected frame rate of the input video signal. Please refer to Section 2.2.1.4 for more details.
AMBA_DSP_PHASE_SHIFT_CTRL_s	**DspPhaseShift**	Sensor readout phase information. Please refer to Section 5.2.1.3 for more details.
AMBA_DSP_COLOR_SPACE_e	**ColorSpace**	Color space of the input data (YUV or RGB)
AMBA_DSP_SENSOR_PATTERN_e	**BayerPattern**	Bayer pattern type. Please refer to Section 5.2.1.4 for more details. This field is only valid for RGB input.
AMBA_DSP_YUV_ORDER_e	**YuvOrder**	YUV pixel order. Please refer to Section 5.2.1.5 for more details. This field is only valid for YUV input.
UINT32	**NumDataBits**	Bit resolution of the input pixel data
UINT8	**NumSkipFrame**	Number of frames to be skipped if VIN configuration changes

*Table 5-4. Definition of **AMBA_DSP_VIN_INFO_s** for VIN API **AmbaDSP_VinConfigSLVS()**.*

5.2.1.3 AmbaDSP_VinConfigSLVS > AMBA_DSP_VIN_SENSOR_READOUT_e

Type	Description
AMBA_DSP_PHASE_SHIFT_e	Phase correction parameter for Horizontal direction.
AMBA_DSP_PHASE_SHIFT_e	Phase correction paramter for Vertical direction.
-	-
-	-
-	-
-	-

*Table 5-5. Definition of **AMBA_DSP_VIN_SENSOR_READOUT_e** for VIN API **AmbaDSP_VinConfigSLVS()**.*

5.2.1.4 AmbaDSP_VinConfigSLVS > AMBA_DSP_SENSOR_PATTERN_e

Type	Description
AMBA_DSP_BAYER_RG	Bayer filter pattern in the order R,G (even line) and G,B (odd line)
AMBA_DSP_BAYER_BG	Bayer filter pattern in the order B,G (even line) and G,R (odd line)
AMBA_DSP_BAYER_GR	Bayer filter pattern in the order G,R (even line) and B,G (odd line)
AMBA_DSP_BAYER_GB	Bayer filter pattern in the order G,B (even line) and R,G (odd line)

*Table 5-6. Definition of **AMBA_DSP_SENSOR_PATTERN_e** for VIN API **AmbaDSP_VinConfigSLVS()**.*

5.2.1.5 AmbaDSP_VinConfigSLVS > AMBA_DSP_YUV_ORDER_e

Type	Description
AMBA_DSP_CR_Y0_CB_Y1	For 1-pixel wide YUV data: Cr, Y0, Cb, Y1 For 2-pixel wide YUV data: {Cr, Y}, {Cb, Y}
AMBA_DSP_CB_Y0_CR_Y1	For 1-pixel wide YUV data: Cb, Y0, Cr, Y1 For 2-pixel wide YUV data: {Cb, Y}, {Cr, Y}
AMBA_DSP_Y0_CR_Y1_CB	For 1-pixel wide YUV data: Y0, Cr, Y1, Cb For 2-pixel wide YUV data: {Y, Cr}, {Y, Cb}
AMBA_DSP_Y0_CB_Y1_CR	For 1-pixel wide YUV data: Y0, Cb, Y1, Cr For 2-pixel wide YUV data: {Y, Cb}, {Y, Cr}

*Table 5-7. Definition of **AMBA_DSP_YUV_ORDER_e** for VIN API **AmbaDSP_VinConfigSLVS()**.*

5.2.1.6 AmbaDSP_VinConfigSLVS > AMBA_DSP_VIN_SLVS_PIN_PAIR_e

Type	Description
AMBA_DSP_VIN_PIN_SD_LVDS_0	LVDS data lane/channel over SD_LVDS_P_0 and SD_LVDS_N_0 pin pair
AMBA_DSP_VIN_PIN_SD_LVDS_1	LVDS data lane/channel over SD_LVDS_P_1 and SD_LVDS_N_1 pin pair

Type	Description
AMBA_DSP_VIN_PIN_SD_LVDS_2	LVDS data lane/channel over SD_LVDS_P_2 and SD_LVDS_N_2 pin pair
AMBA_DSP_VIN_PIN_SD_LVDS_3	LVDS data lane/channel over SD_LVDS_P_3 and SD_LVDS_N_3 pin pair
AMBA_DSP_VIN_PIN_SD_LVDS_4	LVDS data lane/channel over SD_LVDS_P_4 and SD_LVDS_N_4 pin pair
AMBA_DSP_VIN_PIN_SD_LVDS_5	LVDS data lane/channel over SD_LVDS_P_5 and SD_LVDS_N_5 pin pair
AMBA_DSP_VIN_PIN_SD_LVDS_6	LVDS data lane/channel over SD_LVDS_P_6 and SD_LVDS_N_6 pin pair
AMBA_DSP_VIN_PIN_SD_LVDS_7	LVDS data lane/channel over SD_LVDS_P_7 and SD_LVDS_N_7 pin pair
AMBA_DSP_VIN_PIN_SD_LVDS_8	LVDS data lane/channel over SD_LVDS_P_8 and SD_LVDS_N_8 pin pair
AMBA_DSP_VIN_PIN_SD_LVDS_9	LVDS data lane/channel over SD_LVDS_P_9 and SD_LVDS_N_9 pin pair
AMBA_DSP_VIN_PIN_SD_LVDS_10	LVDS data lane/channel over SD_LVDS_P_10 and SD_LVDS_N_10 pin pair
AMBA_DSP_VIN_PIN_SD_LVDS_11	LVDS data lane/channel over SD_LVDS_P_11 and SD_LVDS_N_11 pin pair

*Table 5-8. Definition of **AMBA_DSP_VIN_SLVS_PIN_PAIR_e** for VIN API **AmbaDSP_VinConfigSLVS()**.*

5.2.1.7 AmbaDSP_VinConfigSLVS > AMBA_DSP_VIN_SLVS_SYNC_CODE_e

Type	Field	Description
UINT8	SyncInterleaving	0=none, 1=2-lane interleaving, 2=4-lane interleaving
UINT8	ITU656 Type	1=ITU-656 type, don't care CustomSyncCode structure
	CustomeSyncCode	Customer sync code. Only valid when ITU656Type = 0. Please refer to Section 5.2.1.8 for more details.

*Table 5-9. Definition of **AMBA_DSP_VIN_SLVS_SYNC_CODE_e** for VIN API **AmbaDSP_VinConfigSLVS()**.*

5.2.1.8 AmbaDSP_VinConfigSLVS > CustomerSyncCode

Type	Field	Description
UINT8	PatternAlign	0=LSB aligned, 1=MSB aligned
UINT16	SyncCodeMask	Sync code mask. Data will be identified as sync code when (data & Mask) = Pattern.
	DetectEnable	Indicate which sync code pattern needs to be dealed with.
UINT16	PatternSol	Pattern of customer's SOL sync code
UINT16	PatternEol	Pattern of customer's EOL sync code
UINT16	PatternSof	Pattern of customer's SOF sync code
UINT16	PatternEof	Pattern of customer's EOF sync code
UINT16	PatternSov	Pattern of customer's SOV sync code
UINT16	PatternEov	Pattern of customer's EOV sync code

*Table 5-10. Definition of **CustomerSyncCode** for VIN API **AmbaDSP_VinConfigSLVS()**.*

5.2.1.9 AmbaDSP_VinConfigSLVS > AMBA_DSP_VIN_RX_HV_SYNC_s

Type	Field	Description
UINT32	**NumActivePixels**	Horizontal active region width
UINT32	**NumActiveLines**	Vertical active region height
UINT32	**NumTotalPixels**	Horizontal total width (line_length_pck)
UINT32	**NumTotalLines**	Vertical total height (frame_length_lines)
-	-	-
-	-	-

*Table 5-11. Definition of **AMBA_DSP_VIN_RX_HV_SYNC_s** for VIN API **AmbaDSP_VinConfigSLVS()**.*

5.2.1.10 AmbaDSP_VinConfigSLVS > AMBA_DSP_VIN_TRIGGER_PULSE_s

Type	Field	Description
int	**Enable**	0: Disable trigger pulse control. 1: Enable trigger pulse control.
UINT32	**PulsePolarity**	0: AMBA_DSP_VIN_PULSE_LOW (Active pulse is low) 1: AMBA_DSP_VIN_PULSE_HIGH (Active pulse is high)
UINT32	**PulseStartLine**	Start line of the pulse
UINT32	**PulseEndLine**	End line of the pulse

*Table 5-12. Definition of **AMBA_DSP_VIN_TRIGGER_PULSE_s** for VIN API **AmbaDSP_VinConfigSLVS()**.*

5.2.1.11 AmbaDSP_VinConfigSLVS > AMBA_DSP_VIN_VOUT_SYNC_s

Type	Field	Description
UINT32	**SignalFreq**	0: AMBA_DSP_VIN_VOUT_SYNC_FRAME: Send synchronization signal on every frame. 1: AMBA_DSP_VIN_VOUT_SYNC_FIELD: Send synchronization signal on every field.
UINT32	**SignalLine**	Start active line to send synchronization signal to VOUT.

*Table 5-13. Definition of **AMBA_DSP_VIN_VOUT_SYNC_s** for VIN API **AmbaDSP_VinConfigSLVS()**.*

5.2.2 AmbaDSP_VinConfigMIPI

API Syntax:

AmbaDSP_VinConfigMIPI (int VinID, AMBA_DSP_VIN_MIPI_CONFIG_s *pVinMipiConfig)

Function Description:

- This function is used to configure the VIN MIPI interface.

Parameters:

Type	Parameter	Description
Int	**VinID**	VIN index
AMBA_DSP_VIN_MIPI_CONFIG_s	***pVinMipiConfig**	Pointer to the VIN MIPI configuration. Please refer to Section 5.2.6.1 for more details.

*Table 5-14. Parameters for VIN API **AmbaDSP_VinConfigMIPI()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 5-15. Returns for VIN API **AmbaDSP_VinConfigMIPI()**.*

Example:

None

See Also:

None

5.2.2.1 AmbaDSP_VinConfigMIPI > AMBA_DSP_VIN_MIPI_CONFIG_s

Type	Field	Description
AMBA_DSP_VIN_INFO_s	**Info**	Basic information about pixel data. Please refer to Section 5.2.1.2 for more details.
AMBA_DSP_VIN_MIPI_CSI2_DATA_TYPE_e	**DataType**	Data Type to be captured. Please refer to Section 5.2.2.3 for more details.
UINT8	**NumActiveLanes**	Active Lane Number. For main VIN, it can support 1-4 lanes. For 2nd VIN, it can support 1-2 lanes.
AMBA_DSP_VIN_TRIGGER_PULSE_s	**RxHvSyncCtrl**	Information about sync signal control. Please refer to Section 5.2.1.9 for more details.
AMBA_DSP_VIN_TRIGGER_PULSE_s	**VinTrigPulse[2]**	Information about sync signal control. Please refer to Section 5.2.1.10 for more details.
AMBA_DSP_VIN_VOUT_SYNC_s	**VinVoutSync[2]**	Information about sync signal control. Please refer to Section 5.2.1.8 for more details.
AMBA_DSP_VIN_MPHY_CONFIG_s	**MipiCtrl**	Information about sync signal control. Please refer to Section 5.2.2.2 for more details.

*Table 5-16. Definition of **AMBA_DSP_VIN_MIPI_CONFIG_s** for VIN API **AmbaDSP_VinConfigMIPI()**.*

5.2.2.2 AmbaDSP_VinConfigMIPI > AMBA_DSP_VIN_MPHY_CONFIG_s

Type	Field	Description
-	-	-
-	-	-
UINT32	**HsSettleTime**	Time interval on SoT during which the HS receiver shall ignore any Data Lane HS transitions. For further information, please refer to "*MIPI Alliance Standard for D-PHY*."
UINT32	**HsTermTime**	Specifies the time that the D-PHY waits before enabling data lane termination after detecting that the sensor has driven the data lanes to the LP00 bridge state. For further information, please refer to *"MIPI Alliance Standard for D-PHY".*
UINT32	**ClkSettleTime**	Time interval on SoT during which the HS receiver shall ignore any Clock Lane HS transitions. For further information, please reference *"MIPI Alliance Standard for D-PHY".*
UINT32	**ClkTermTime**	Time for the Clock Lane receiver to enable the HS line termination. For further information, please reference *"MIPI Alliance Standard for D-PHY".*
UINT32	**ClkMissTime**	Detection time that the clock has stopped toggling. For further information, please refer to *"MIPI Alliance Standard for D-PHY".*
UINT32	**RxInitTime**	Initialization period (PHY may calibrate)

*Table 5-17. Definition of **AMBA_DSP_VIN_MPHY_CONFIG_s** for VIN API **AmbaDSP_VinConfigMIPI()**.*

5.2.2.3 AmbaDSP_VinConfigMIPI > AMBA_DSP_VIN_MIPI_CSI2_DATA_TYPE_e

Type	Description
AMBA_DSP_VIN_MIPI_SYNC_FRAME_START	Frame Start Code
AMBA_DSP_VIN_MIPI_SYNC_FRAME_END	Frame End Code
AMBA_DSP_VIN_MIPI_SYNC_LINE_START	Line Start Code
AMBA_DSP_VIN_MIPI_SYNC_LINE_END	Line End Code
AMBA_DSP_VIN_MIPI_SHORT_PACKET1	Generic Short Packet Code 1
AMBA_DSP_VIN_MIPI_SHORT_PACKET2	Generic Short Packet Code 2
AMBA_DSP_VIN_MIPI_SHORT_PACKET3	Generic Short Packet Code 3
AMBA_DSP_VIN_MIPI_SHORT_PACKET4	Generic Short Packet Code 4
AMBA_DSP_VIN_MIPI_SHORT_PACKET5	Generic Short Packet Code 5
AMBA_DSP_VIN_MIPI_SHORT_PACKET6	Generic Short Packet Code 6
AMBA_DSP_VIN_MIPI_SHORT_PACKET7	Generic Short Packet Code 7
AMBA_DSP_VIN_MIPI_SHORT_PACKET8	Generic Short Packet Code 8
AMBA_DSP_VIN_MIPI_LONG_NULL	Generic 8-bit Long Null
AMBA_DSP_VIN_MIPI_LONG_BLANKING	Generic 8-bit Long Blanking Data
AMBA_DSP_VIN_MIPI_LONG_EMBEDDED	Generic 8-bit Long Embedded Information
AMBA_DSP_VIN_MIPI_YUV420_8BIT	YUV420 8-bit
AMBA_DSP_VIN_MIPI_YUV420_10BIT	YUV420 10-bit
AMBA_DSP_VIN_MIPI_YUV422_8BIT	YUV422 8-bit
AMBA_DSP_VIN_MIPI_YUV422_10BIT	YUV42210-bit
AMBA_DSP_VIN_MIPI_RGB444	RGB444
AMBA_DSP_VIN_MIPI_RGB555	RGB555
AMBA_DSP_VIN_MIPI_RGB565	RGB565
AMBA_DSP_VIN_MIPI_RGB666	RGB666
AMBA_DSP_VIN_MIPI_RGB888	RGB888
AMBA_DSP_VIN_MIPI_RAW6	RAW6
AMBA_DSP_VIN_MIPI_RAW7	RAW7
AMBA_DSP_VIN_MIPI_RAW8	RAW8
AMBA_DSP_VIN_MIPI_RAW10	RAW10
AMBA_DSP_VIN_MIPI_RAW12	RAW12
AMBA_DSP_VIN_MIPI_RAW14	RAW14
AMBA_DSP_VIN_MIPI_USER_DEFINED1	User Defined 8-bit Data Type 1
AMBA_DSP_VIN_MIPI_USER_DEFINED2	User Defined 8-bit Data Type 2
AMBA_DSP_VIN_MIPI_USER_DEFINED3	User Defined 8-bit Data Type 3
AMBA_DSP_VIN_MIPI_USER_DEFINED4	User Defined 8-bit Data Type 4
AMBA_DSP_VIN_MIPI_USER_DEFINED5	User Defined 8-bit Data Type 5
AMBA_DSP_VIN_MIPI_USER_DEFINED6	User Defined 8-bit Data Type 6
AMBA_DSP_VIN_MIPI_USER_DEFINED7	User Defined 8-bit Data Type 7
AMBA_DSP_VIN_MIPI_USER_DEFINED8	User Defined 8-bit Data Type 8

*Table 5-18. Definition of **AMBA_DSP_VIN_MIPI_CSI2_DATA_TYPE_e** for VIN API **AmbaDSP_VinConfigMIPI()**.*

5.2.3 AmbaDSP_VinConfigDVP

API Syntax:

AmbaDSP_VinConfigDVP (int VinID, AMBA_DSP_VIN_DVP_CONFIG_s *pVinDvpConfig)

Function Description:

- This function is used to configure the VIN digital video port (DVP) interface.

Parameters:

Type	Parameter	Description
Int	**VinID**	VIN index
AMBA_DSP_VIN_DVP_CONFIG_s	***pVinDvpConfig**	Pointer to the VIN DVP configuration. Please refer to Section 5.2.3.1 for more details.

*Table 5-19. Parameters for VIN API **AmbaDSP_VinConfigDVP()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 5-20. Returns for VIN API **AmbaDSP_VinConfigDVP()**.*

Example:

None

See Also:

None

5.2.3.1 AmbaDSP_VinConfigDVP > AMBA_DSP_VIN_DV_CONFIG_s

Type	Field	Description
AMBA_DSP_VIN_INFO_s	**Info**	Basic information about pixel data. Please refer to Section 5.2.1.2 for more details.
-	-	-
AMBA_DSP_VIN_SIGNAL_EDGE_TYPE_e	**DataClockEdge**	Active signal transition for data latching. Please refer to Section for more details.
AMBA_DSP_DVP_TYPE_s	**DvpType**	DVP data interface. Please refer to Section 5.2.3.2 for more details.
AMBA_DSP_VIN_SYNC_TYPE_e	**SyncType**	Line-sync and frame-sync signal transmission type. Please refer to Section 5.2.3.3 for more details.
AMBA_DSP_VIN_SIGNAL_EDGE_TYPE_e	**HsyncPolarity**	Active signal transition for Hsync. Please refer to Section 5.2.3.4 for more details.
AMBA_DSP_VIN_SIGNAL_EDGE_TYPE_e	**VsyncPolarity**	Active signal transition for Vsync. Please refer to Section 5.2.3.4 for more details.
AMBA_DSP_VIN_RX_HV_SYNC_s	**RxHvSyncCtrl**	Information about sync signal control. Please refer to Section 5.2.1.6 for more details.
AMBA_DSP_VIN_SIGNAL_EDGE_TYPE_e	**FieldPolarity**	Active signal transition for Field. Please refer to Section 5.2.3.4 for more details.
AMBA_DSP_VIN_SYNC_PIN_SELECT_s	**SyncPinSelect**	HSync, VSync and Field pin select. 0-9: pad_p[x]; 10-19: pad_n[x], 20: clk_n[0]; 21: clk_n[1]; 22: clk_p[1]
AMBA_DSP_VIN_VOUT_SYNC_s	**VinVoutSync[2]**	Information about sync signal control. Please refer to Section 5.2.1.8 for more details.

*Table 5-21. Definition of **AMBA_DSP_VIN_DVP_CONFIG_s** for VIN API **AmbaDSP_VinConfigDVP()**.*

5.2.3.2 AmbaDSP_VinConfigDVP > AMBA_DSP_VIN_DVP_TYPE_e

Type	Description
AMBA_DSP_VIN_DATA_DVP_SINGLE_PEL_SDR	One pixel data per pixel clock cycle
AMBA_DSP_VIN_DATA_DVP_SINGLE_PEL_DDR	Two pixel data per pixel clock cycle. (one pixel is sampled at the rising edge while the other pixel is sampled at the falling edge)
AMBA_DSP_VIN_DATA_DVP_DOUBLE_PEL_SDR	Two pixel data per pixel clock cycle. (two pixels are sampled at the same pixel clock edge)
AMBA_DSP_VIN_DATA_DVP_DOUBLE_PEL_DDR	Four pixel data per pixel clock cycle. (two pixels are sampled at the rising edge while the others are sampled at the falling edge)

*Table 5-22. Definition of **AMBA_DSP_VIN_DVP_TYPE_e** for VIN API **AmbaDSP_VinConfigDVP()**.*

5.2.3.3 AmbaDSP_VinConfigDVP > AMBA_DSP_SYNC_TYPE_e

Type	Description
AMBA_DSP_VIN_SYNC_BT601	No embedded sync code
AMBA_DSP_VIN_SYNC_BT656_8BIT_ LOWER_PEL	8-bit embedded sync code carried on lower pixel only
AMBA_DSP_VIN_SYNC_BT656_8BIT_UP_ Per_PEL	8-bit embedded sync code carried on upper pixel only
AMBA_DSP_VIN_SYNC_BT656_8BIT_ BOTH_PEL	8-bit embedded sync code carried on both pixels
AMBA_DSP_VIN_SYNC_BT656_FULL_ LOWER_PEL	Full data range embedded sync code carried on lower pixel only
AMBA_DSP_VIN_SYNC_BT656_FULL_Up_ PER_PEL	Full data range embedded sync code carried on upper pixel only
AMBA_DSP_VIN_SYNC_BT656_FULL_ BOTH_PEL	Full data range embedded sync code carried on both pixels

*Table 5-23. Definition of **AMBA_DSP_SYNC_TYPE_e** for VIN API **AmbaDSP_VinConfigDVP()**.*

5.2.3.4 AmbaDSP_VinConfigDVP > AMBA_DSP_VIN_SIGNAL_EDGE_TYPE_e

Type	Description
AMBA_DSP_SIGNAL_RISING_EDGE	Rising edge. A signal transition from low to high (0 to 1)
AMBA_DSP_SIGNAL_FALLING_EDGE	Falling edge. A signal transition from high to low (1 to 0)

*Table 5-24. Definition of **AMBA_DSP_VIN_SIGNAL_EDGE_TYPE_e** for VIN API **AmbaDSP_VinConfigDVP()**.*

5.2.4 AmbaDSP_VinCaptureConfig

API Syntax:

AmbaDSP_VinCaptureConfig (int VinID, AMBA_DSP_WINDOW_s *pCaptureWindow)

Function Description:

- This function is used to configure the VIN capture window.

Parameters:

Type	Parameter	Description
Int	**VinID**	VIN index
AMBA_DSP_WINDOW_s	***pCaptureWindow**	Pointer to the VIN capture window configuration. Please refer to Section 4.2.9.1 for more details.

*Table 5-25. Parameters for VIN API **AmbaDSP_VinCaptureConfig()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 5-26. Returns for VIN API **AmbaDSP_VinCaptureConfig()**.*

Example:

```
To configure 4Kx2K sensor mode and VIN capture window:

AMBA_DSP_WINDOW_s VinCapture;
AMBA_SENSOR_AREA_INFO_s *pRecordingPixels = &SensorStatus.ModeInfo.OutputInfo.
RecordingPixels;
AMBA_DSP_CHANNEL_ID_u Chan = { .Bits = { .VinID = 0, .SensorID = 0 } };
AMBA_SENSOR_MODE_ID_u SensorMode = { .Data = AMBA_SENSOR_IMX117_TYPE_2_5_
MODE_0 };
UINT32 CapW = 3840, CapH = 2160;

AmbaSensor_Enable(Chan);
AmbaSensor_Config(Chan, SensorMode, AMBA_SENSOR_ESHUTTER_TYPE_ROLLING);
AmbaSensor_GetStatus(Chan, &SensorStatus);
VinCapture.OffsetX = (pRecordingPixels->StartX + ((pRecordingPixels->Width -
CapW) >> 1)) & ~1;
VinCapture.OffsetY = (pRecordingPixels->StartY + ((pRecordingPixels->Height -
CapH) >> 1)) & ~1;
VinCapture.Width = CapW;
VinCapture.Height = CapH;
AmbaDSP_VinCaptureConfig(0, &VinCapture);
```

See Also:

None

5.2.5 AmbaDSP_VinPhySetSLVS

API Syntax:

AmbaDSP_VinPhySetSLVS (int VinID)

Function Description:

- This function is used to configure VIN PHY to VIN SLVS interface.

Parameters:

Type	Parameter	Description
Int	**VinID**	VIN index

*Table 5-27. Parameters for VIN API **AmbaDSP_VinPhySetSLVS()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 5-28. Returns for VIN API **AmbaDSP_VinPhySetSLVS()**.*

Example:

None

See Also:

None

5.2.6 AmbaDSP_VinPhySetMIPI

API Syntax:

AmbaDSP_VinPhySetMIPI (int VinID)

Function Description:

- This function is used to configure VIN PHY to MIPI interface.

Parameters:

Type	Parameter	Description
Int	**VinID**	VIN index

*Table 5-29. Parameters for VIN API **AmbaDSP_VinPhySetMIPI()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 5-30. Returns for VIN API **AmbaDSP_VinPhySetMIPI()**.*

Example:

None

See Also:

None

5.2.7 AmbaDSP_VinPhySetDVP

API Syntax:

AmbaDSP_VinPhySetDVP (int VinID)

Function Description:

- This function is used to configure VIN PHY to DVP interface.

Parameters:

Type	Parameter	Description
Int	**VinID**	VIN index

*Table 5-31. Parameters for VIN API **AmbaDSP_VinPhySetDVP()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 5-32. Returns for VIN API **AmbaDSP_VinPhySetDVP()**.*

Example:

None

See Also:

None

5.2.8 AmbaDSP_VinConfigMasterSync

API Syntax:

AmbaDSP_VinConfigMasterSync (int VinID, AMBA_DSP_VIN_MASTER_SYNC_CONFIG_s *pVinSync-Config)

Function Description:

- This function is used to configure the master HSync/Vsync generator.

Parameters:

Type	Parameter	Description
Int	**VinID**	VIN index
AMBA_DSP_VIN_MAS-TER_SYNC_CONFIG_s	***pVinSyncConfig**	Pointer to the master sync configuration. Please refer to Section for more details.

*Table 5-33. Parameters for VIN API **AmbaDSP_VinConfigMasterSync()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 5-34. Returns for VIN API **AmbaDSP_VinConfigMasterSync()**.*

Example:

None

See Also:

None

5.2.8.1 AmbaDSP_VinConfigMasterSync > AMBA_DSP_VIN_MASTER_SYNC_CONFIG_s

Type	Field	Description
AMBA_DSP_VIN_SYNC_WAVEFORM_s	**HSync**	Hsync waveform configuration. Please refer to Section for more details.

Type	Field	Description
AMBA_DSP_VIN_SYNC_WAVEFORM_CTRL_s	**VSync**	Vsync waveform configuration. Please refer to Section for more details.
UINT16	**VSyncDelayCycles**	Offset between Hsync and Vsync edges
UINT8	**ToggleHsyncInVblank**	1: toggle Hysnc during Vblank

*Table 5-35. Definition of **AMBA_DSP_VIN_MASTER_SYNC_CONFIG_s** for VIN API **AmbaDSP_VinConfigMaster-Sync()**.*

5.2.8.2 AmbaDSP_VinConfigMasterSync > AMBA_DSP_VIN_SYNC_WAVEFORM_s

Type	Field	Description
UINT32	**Period**	HSync/Vsync period. For HSync, the unit is cycle. For Vsync, the unit is Hsync.
UINT32	**PulseWidth**	Sync pulse width (unit: cycle)
UINT32	**Polarity**	0: Active Low, 1: Active High

*Table 5-36. Definition of **AMBA_DSP_VIN_MASTER_SYNC_CONFIG_s** for VIN API **AmbaDSP_VinConfigMaster-Sync()**.*

6 Liveview

6.1 Liveview: Overview

This chapter details the functions used to configure the hardware for Liveview.

6.2 Liveview: List of Functions

6.2.1 AmbaDSP_LiveviewConfig

API Syntax:

AmbaDSP_LiveviewConfig (AMBA_DSP_LIVEVIEW_CTRL_FLAG_u CtrlFlag, int NumChannel, AMBA_DSP_LIVEVIEW_CONFIG_s *pConfig)

Function Description:

- This function is used for Liveview configuration.

Parameters:

Type	Parameter	Description
AMBA_DSP_LIVEVIEW_CTRL_FLAG_u	**CtrlFlag**	Liveview control flags
int	**NumChannel**	Number of channel configurations
AMBA_DSP_Liveview_CONFIG_s	***pConfig**	Pointer to Liveview configuration. Please refer to Section 6.2.1.1 for more details.

*Table 6-1. Parameters for Liveview API **AmbaDSP_LiveviewConfig()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 6-2. Returns for Liveview API **AmbaDSP_LiveviewConfig()**.*

Example:

```
AMBA_DSP_LIVEVIEW_CONFIG_s LiveviewCfg;
AMBA_DSP_LIVEVIEW_CTRL_FLAG_u CtrlFlag;
memset(&LiveviewCfg, 0,  sizeof(LiveviewCfg));
memset(&CtrlFlag, 0,    sizeof(CtrlFlag));
LiveviewCfg.VinID = 0;
LiveviewCfg.ViewZoneID = 0;
LiveviewCfg.FrameRate.TimeScale =  30000;
LiveviewCfg.FrameRate.NumUnitsInTick =  1001;
LiveviewCfg.RawInputWindow.Width =  3840;
LiveviewCfg.RawInputWindow.Height =  2160;
LiveviewCfg.MainYuvWidth =  3840; LiveviewCfg.MainYuvHeight =  2160;
LiveviewCfg.SecYuvWidth =  432;
LiveviewCfg.SecYuvHeight =  240;
CtrlFlag.Bits.BackgroundProc =  1;
CtrlFlag.Bits.MctfCmpr =  1;
AmbaDSP_LiveviewConfig(1, &LiveviewCfg);
```

See Also:

AmbaDSP_LiveviewConfigCtrl

6.2.1.1 AmbaDSP_LiveviewConfig > AMBA_DSP_LIVEVIEW_CONFIG_s

Type	Field	Description
UINT8	**VinID**	Input VIN ID
AMBA_DSP_WINDOW_s	**RawInputWindow**	Input raw window from indicated VIN
UINT8	**ViewZoneID**	Give the raw window a View Zone ID
UINT8	**FrameRate**	Main frame rate.
UINT16	**SecFrameRateDivisor**	Second stream frame
UINT16	**MainYuvWidth**	Main YUV width. This parameter is only for single channel (**NumChannel** = 1). For multiple channels, please refer to Section 9.2.15.1 to configure overall Main YUV size.
UINT16	**MainYuvHeight**	Main YUV height. This parameter is only for single channel (**NumChannel** = 1). For multiple channels, please refer to Section 9.2.15.1 to configure overall Main YUV size.
UINT16	**SecYuvWidth**	Second YUV width. This parameter is for both single channel and multiple channels.
UINT16	**SecYuvHeight**	Second YUV height. This parameter is for both single channel and multiple channels.
-	-	-
AMBA_DSP_LIVEVIEW_EXT_BUF_s	****pExtBuf2FOV**	Pointer to the external image buffer for 2-FOV support. Null: Please refer to Section 6.2.1.4 below for more details .
UINT8	**BlackbarPaddedMBSize**	Black-bar size in micro block. Used when encode rotates. Control black-bar size when encode rotates. Set it to 0 for default value.
AMBA_DSP_RAW_CAP_INFO_s	***pRawCapDramInfo**	Pointer to a user-provided DRAM structure, and the DSP completes the raw capture information. Please refer to Section 6.2.1.8 for more details.

*Table 6-3. Definition of **AMBA_DSP_LIVEVIEW_CONFIG_s** for Liveview API **AmbaDSP_LiveviewConfig()**.*

6.2.1.2 AmbaDSP_LiveviewConfig > AMBA_DSP_LIVEVIEW_CTRL_FLAG_u

Type	Field	Description
UINT16	**Data**	16-bit data.
UINT16: 1	**VideoProcMode**	0: Express pipeline. 1: Hybrid pipeline.
UINT16: 2	**VideoAlgoMode**	Video quality algorithm. 0: Fast mode. 1: LISO mode. 2: HISO, only for Hybrid pipeline.
UINT16: 1	**VideoOSMode**	Oversampling. 0: Disable. Scale down to fit main. 1: Enable.
UINT16: 2	**EnableVideoTuning Mode**	Video tuning mode (feed raw from memory). Hybrid pipeline only. 0: Disable. 1: Enable.
UINT16: 1	**LiveViewOnly**	Disable video encode. 0: Disable. 1: Enable.
UINT16: 1	**MultiChannelProc**	Multi-channel processing mode. 0: Disable. 1: Enable.

*Table 6-4. Definition of **AMBA_DSP_LIVEVIEW_CTRL_FLAG_u** for Liveview API **AmbaDSP_LiveviewConfig()**.*

6.2.1.3 AmbaDSP_LiveviewConfig > AMBA_DSP_CHANNEL_ID_u

Type	Field	Description
UINT8	**Data**	8-bit data.
UINT8 : 2	**VinID**	VIN ID
UINT8 : 6	**SensorID**	Sensor ID/Source ID

*Table 6-5. Definition of **AMBA_DSP_CHANNEL_ID_u** for Liveview API **AmbaDSP_LiveviewConfig()**.*

6.2.1.4 AmbaDSP_LiveviewConfig > AMBA_DSP_LIVEVIEW_EXT_BUF_s

Type	Field	Description
UINT16	**MaxNumPicture**	External buffer capacity
AMBA_DSP_RAW_BUF_s	***pRawBufAddr**	Pointer to Raw buffers. Please refer to Section 7.2.3.3 for more details.
AMBA_DSP_YUV_IMG_BUF_s	***pYuvBufAddr**	Pointer to YUV buffers. Please refer to Section 6.2.1.5 for more details.

Type	Field	Description
UINT8	**RawBayerPattern**	Bayer pattern type of raw image
UINT8	**RawDataBits**	Pixel bit depth of raw image

*Table 6-6. Definition of **AMBA_DSP_LIVEVIEW_EXT_BUF_s** for Liveview API **AmbaDSP_LiveviewConfig()**.*

6.2.1.5 AmbaDSP_LiveviewConfig > AMBA_DSP_YUV_IMG_BUF_s

Type	Field	Description
AMBA_ DSP_YUV_ FORMAT_e	**DataFmt**	YUV Data format. Please refer to Section 6.2.1.6 below for more details.
UINT8	***pBaseAddrY**	Pointer to Luma (Y) data area
UINT8	***pBaseAddrUV**	Pointer to Chroma (UV) data area
UINT16	**Pitch**	YUV buffer pitch
AMBA_DSP_ WINDOW_s	**Window**	Window position and size. Please refer to Section 6.2.1.7 below for more details.

*Table 6-7. Definition of **AMBA_DSP_YUV_IMG_BUF_s** for Liveview API **AmbaDSP_LiveviewConfig()**.*

6.2.1.6 AmbaDSP_LiveviewConfig > AMBA_DSP_YUV_FORMAT_e

Type	Description
AMBA_DSP_YUV420	YUV 420 format
AMBA_DSP_YUV422	YUV 422 format

*Table 6-8. Definition of **AMBA_DSP_YUV_FORMAT_e** for Liveview API **AmbaDSP_LiveviewConfig()**.*

6.2.1.7 AmbaDSP_LiveviewConfig > AMBA_DSP_WINDOW_s

Type	Field	Description
UINT16	**OffsetX**	Horizontal offset of the window
UINT16	**OffsetY**	Vertical offset of the window
UINT16	**Width**	Number of pixels per line in the window
UINT16	**Height**	Number of lines in the window

*Table 6-9. Definition of **AMBA_DSP_WINDOW_s** for Liveview API **AmbaDSP_LiveviewConfig()**.*

6.2.1.8 AmbaDSP_LiveviewConfig > AMBA_DSP_RAW_CAP_INFO_s

Type	Field	Description
UINT32	**CapAddr**	DSP raw capture DRAM address
UINT32	**Pitch**	DRAM pitch
UINT16	**FrmSyncCount**	Frame sync count

Type	Field	Description
UINT16	**CapDreg**	Capture the data register file (DREG)
UINT32	**CapCount**	Capture count

*Table 6-10. Definition of **AMBA_DSP_RAW_CAP_INFO_s** for Liveview API **AmbaDSP_LiveviewConfig()**.*

6.2.2 AmbaDSP_LiveviewDispConfig

API Syntax:

AmbaDSP_LiveviewDispConfig (AMBA_DSP_VOUT_IDX_e VoutIdx, AMBA_DSP_LIVEVIEW_DISP_WINDOW_s *pDispWinConfig)

Function Description:

- This function is used to configure the Liveview channel on the display VOUT.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index
AMBA_DSP_LIVEVIEW_DISP_WINDOW_s	***pDispWinConfig**	Pointer to Liveview channel configuration on display. Please refer to Section 6.2.2.1 below for more details.

*Table 6-11. Parameters for Liveview API **AmbaDSP_LiveviewDispConfig()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 6-12. Returns for Liveview API **AmbaDSP_LiveviewDispConfig()**.*

Example:

To configure and enable liveview on LCD VOUT:

```
AMBA_DSP_LIVEVIEW_DISP_WINDOW_s     LiveviewDisplayCfg;
memset(&LiveviewDisplayCfg, 0, sizeof(LiveviewDisplayCfg));
 LiveviewDisplayCfg.ChannelID.Bits.VinID =  0;
LiveviewDisplayCfg.ChannelID.Bits.SensorID =  0; LiveviewDisplayCfg.DispWidth
=  960;
LiveviewDisplayCfg.DispHeight = 480;
LiveviewDisplayCfg.NumChannel = 1;
LiveviewDisplayCfg.FrameRate.TimeScale =  180000;
LiveviewDisplayCfg. FrameRate.NumUnitsInTick =  3003; AmbaDSP_
LiveviewDispConfig(AMBA_DSP_VOUT_LCD, &LiveviewDisplayCfg); AmbaDSP_
LiveviewDispCtrl(AMBA_DSP_VOUT_LCD,1);
```

See Also:

AmbaDSP_LiveviewDispCtrl

6.2.2.1 AmbaDSP_LiveviewDispConfig > AMBA_DSP_LIVEVIEW_DISP_WINDOW_s

Type	Field	Description
AMBA_DSP_FRAME_RATE_s	**FrameRate**	Frame rate. Please refer to Section 4.2.1.9 for more details.
UINT16	**DispWidth**	Overall display width on VOUT
UINT16	**DispHeight**	Overall display height on VOUT
int	**NumChannel**	Number of channels
AMBA_DSP_LIVEVIEW_CHANNEL_WINDOW_s	***pChanCfg**	Pointer to channel configuration. This parameter is needed for **NumChannel** > 1. Please refer to Section 6.2.2.2 below for more details.

*Table 6-13. Definition of **AMBA_DSP_LIVEVIEW_DISP_WINDOW_s** for Liveview API **AmbaDSP_LiveviewDispConfig()**.*

6.2.2.2 AmbaDSP_LiveviewDispConfig > AMBA_DSP_LIVEVIEW_CHANNEL_WINDOW_s

Type	Field	Description
UINT8	**ViewZoneID**	Source View Zone ID
AMBA_DSP_WINDOW_s	**Window**	Channel position and size on VOUT. Please refer to Section 6.2.1.7 for more details.
AMBA_DSP_ROTATE_FLIP_e	**RotateFlip**	Display rotate/flip for this channel. Please refer to Section 4.2.8.1 for more details.
UINT8	***pLumaAlphaTable**	Pointer to luma alpha table for RotateFlip = AMBA_DSP_ROTATE_90, AMBA_DSP_ROTATE_90_VERT_FLIP, AMBA_DSP_ROTATE_270, and AMBA_DSP_ROTATE_270_VERT_FLIP.

*Table 6-14. Definition of **AMBA_DSP_LIVEVIEW_CHANNEL_WINDOW_s** for Liveview API **AmbaDSP_LiveviewDispConfig()**.*

6.2.3 AmbaDSP_LiveviewDispCtrl

API Syntax:

AmbaDSP_LiveviewDispCtrl (AMBA_DSP_VOUT_IDX_e VoutIdx, int Enable)

Function Description:

- This function is used to enable or disable Liveview.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	VOUT index
int	**Enable**	0: Disable Liveview on this VOUT 1: Enable Liveview on this VOUT

*Table 6-15. Parameters for Liveview API **AmbaDSP_LiveviewDispCtrl()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 6-16. Returns for Liveview API **AmbaDSP_LiveviewDispCtrl()**.*

Example:

None

See Also:

AmbaDSP_LiveviewDispConfig

6.2.4 AmbaDSP_LiveviewCtrl

API Syntax:

AmbaDSP_LiveviewCtrl (int Enable, int Show)

Function Description:

- This function is used to configure Liveview timing and display settings. This API is a blocking function and will not return until the request takes effect.

Parameters:

Type	Parameter	Description
int	**Enable**	0: Disable Liveview. 1: Enable Liveview.
int	**Show**	Valid for **Enable** = 1 0: Liveview in background 1: Show Liveview when ready.

*Table 6-17. Parameters for Liveview API **AmbaDSP_LiveviewCtrl()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 6-18. Returns for Liveview API **AmbaDSP_LiveviewCtrl()**.*

Example:

To enable liveview and show:

```
AmbaDSP_LiveviewCtrl(1, 1);
```

To disable liveview:

```
AmbaDSP_LiveviewCtrl(0, 0);
```

To enable liveview but show it later:

```
AmbaDSP_LiveviewCtrl(1, 0);
….
AmbaDSP_LiveviewCtrl(1, 1);
```

See Also:

AmbaDSP_LiveviewConfig

6.2.5 AmbaDSP_LiveviewChannelCtrl

API Syntax:

AmbaDSP_LiveviewChannelCtrl (AMBA_DSP_LIVEVIEW_CHANNEL_CTRL_s *pChannelCtrl)

Function Description:

- This function is used to configure a specified Liveview channel.

Parameters:

Type	Parameter	Description
AMBA_DSP_LIVEVIEW_CHANNEL_CTRL_s	***pChannelCtrl**	Pointer to Liveview channel control. Please refer to Section 6.2.5.1 below for more details.

*Table 6-19. Parameters for Liveview API **AmbaDSP_LiveviewChannelCtrl()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 6-20. Returns for Liveview API **AmbaDSP_LiveviewChannelCtrl()**.*

Example:

None

See Also:

None

6.2.5.1 AmbaDSP_LiveviewChannelCtrl > AMBA_DSP_LIVEVIEW_CHANNEL_CTRL_s

Type	Field	Description
UINT8	**Enable**	0: Disable 1: Enable
UINT8	**ViewZoneID**	View Zone ID.
AMBA_DSP_WINDOW_s	**RawInputWindow**	Input raw window from indicated VIN.
UINT8	**EnableVerticalFlip**	0: Disable 1: Enable vertical flip

*Table 6-21. Definition of **AMBA_DSP_LIVEVIEW_DISP_WINDOW_s** for Liveview API **AmbaDSP_LiveviewChannelCtrl ()**.*

6.2.6 AmbaDSP_LiveviewEisConfig

API Syntax:

AmbaDSP_LiveviewEisConfig (AMBA_EIS_COEFF_INFO_s *pEisCoeffInfo, UINT8 Enable, UINT32 CmdReadDly)

Function Description:

- This function is used to configure the EIS settings.

Parameters:

Type	Parameter	Description
AMBA_EIS_CO-EFF_INFO_s	***pEisCoeffInfo**	Pointer to EIS configuration. Please refer to Section 6.2.6.1 for more details.
UINT8	**Enable**	Enable EIS.
UINT32	**CmdReadDly**	Timing control of EIS

*Table 6-22. Parameters for Liveview API **AmbaDSP_LiveviewEisConfig()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 6-23. Returns for Liveview API **AmbaDSP_LiveviewEisConfig()**.*

Example:

None

See Also:

None

6.2.6.1 AmbaDSP_LiveviewEisConfig > AMBA_EIS_COEFF_INFO_s

Type	Field	Description
UINT32	**ActualLeftTopX**	Horizontal position of top-left-most point
UINT32	**ActualLeftTopY**	Vertical position of top-left-most point
UINT32	**ActualRightBotX**	Horizontal position of bottom-right-most point
UINT32	**ActualRightBotY**	Vertical position of bottom-right-most point
INT32	**HotiSkewPhaseInc**	HotiSkewPhaseInc
UINT32	**ZoomY**	Vertical zoom ratio
UINT16	**DummyWindowXLeft**	Horizontal position of dummy window
UINT16	**DummyWindowYTop**	Vertical position of dummy window
UINT16	**DummyWindowWidth**	Width of dummy window
UINT16	**DummyWindowHeight**	Height of dummy window
UINT32	**ArmSysTime**	System time

*Table 6-24. Definition of **AMBA_EIS_COEFF_INFO_s** for Liveview API **AmbaDSP_LiveviewEisConfig()**.*

6.2.7 AmbaDSP_LiveviewWaitUpdate

API Syntax:

AmbaDSP_LiveviewWaitUpdate (UINT8 FrameCount)

Function Description:

- This function is used to synchronize or wait for the timing of liveview updating.

Parameters:

Type	Parameter	Description
UINT8	**FrameCount**	Number of counts to wait.

*Table 6-25. Parameters for Liveview API **AmbaDSP_LiveviewWaitUpdate()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 6-26. Returns for Liveview API **AmbaDSP_LiveviewWaitUpdate()**.*

Example:

None

See Also:

None

7 Still Capture

7.1 Still Capture: Overview

This chapter specifies functions used to configure settings for still-image capture.

7.2 Still Capture: List of Functions

7.2.1 AmbaDSP_StillRawCaptureConfig

API Syntax:

AmbaDSP_StillRawCaptureConfig (AMBA_DSP_STILL_RAW_CAPTURE_CONFIG_s *pConfig)

Function Description:

- This function is used to pass raw capture information to DSP before starting.

Parameters:

Type	Parameter	Description
AMBA_DSP_STILL_RAW_CAPTURE_CONFIG_s	***pConfig**	Structure Configuration. Please refer to Section 7.2.1.1 for more details.

*Table 7-1. Parameters for Still Capture API **AmbaDSP_StillRawCaptureConfig()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 7-2. Returns for Still Capture **API AmbaDSP_StillRawCaptureConfig()**.*

Example:

See Also:

7.2.1.1 AmbaDSP_StillRawCaptureConfig > AMBA_DSP_STILL_RAW_CAPTURE_CONFIG_s

Type	Field	Description
UINT8	**RawCaptureCntl**	Raw capture control method 0: DSP auto capture until RawCaptureNum reached 1: Manual capture
UINT32	**RawCaptureNum**	Number of frames to capture
UINT16	**YuvDepth**	Number of pictures pYuvBufAddr can store
AMBA_DSP_YUV_IMG_BUF_s	***pYuvBufAddr**	Main YUV buffer configuration. Please refer to Section 6.2.1.5 for more details.
UINT16	**ScrnDepth**	Number of pictures pScrnBufAddr can store

Type	Field	Description
AMBA_DSP_YUV_IMG_BUF_s	***pScrnBufAddr**	Screenail YUV buffer configuration. Please refer to Section 6.2.1.5 for more details.
UINT16	**ThmbDepth**	Number of pictures pThmbBufAddr can store
AMBA_DSP_YUV_IMG_BUF_s	***pThmbBufAddr**	Thumbnail YUV buffer configuration. Please refer to Section 6.2.1.5 for more details.
UINT16	**MainWidth**	Main picture width
UINT16	**MainHeight**	Main picture height
UINT16	**ScrnWidth**	Screennail picture width
UINT16	**ScrnHeight**	Screennail picture height
UINT16	**ScrnActWidth**	Screennail picture active region width
UINT16	**ScrnActHeight**	Screennail picture active region height
UINT16	**ThmbWidth**	Thumbnail picture width
UINT16	**ThmbHeight**	Thumbnail picture height
UINT16	**ThmbActWidth**	Thumbnail picture active region width
UINT16	**ThmbActHeight**	Thumbnail picture active region height

*Table 7-3. Definition of **AMBA_DSP_STILL_RAW_CAPTURE_CONFIG_s** for Still Capture API **AmbaDSP_StillRawCaptureConfig ()**.*

7.2.2 AmbaDSP_StillProcessConfig

API Syntax:

AmbaDSP_StillProcessConfig (AMBA_DSP_STILL_PROCESS_CONFIG_s *pConfig)

Function Description:

- This function is used to pass raw processing information to DSP before starting.

Parameters:

Type	Parameter	Description
AMBA_DSP_STILL_PRO-CESS_CONFIG_s	***pConfig**	Structure Configuration. Please refer to Section 7.2.2.1 for more details.

*Table 7-4. Parameters for Still Capture API **AmbaDSP_StillProcessConfig()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 7-5. Returns for Still Capture **API AmbaDSP_StillProcessConfig()**.*

Example:

See Also:

7.2.2.1 AmbaDSP_StillProcessConfig > AMBA_DSP_STILL_PROCESS_CONFIG_s

Type	Field	Description
AMBA_DSP_STILL_OP_MODE_e	**OpMode**	Still processing mode, please refer to Section 7.2.5.2 for details
UINT8	**InputFormat**	Input format 1: CFA 1: YUV
UINT16	**InputDepth**	Number of frames input buffer can store
Union	**Input**	Input buffer
UINT8	**RawDataBits**	Pixel bit depth when input is CFA
UINT8	**RawBayerPattern**	Bayer pattern type when input is CFA
UINT16	**ScrnActWidth**	Screennail picture active region width
UINT16	**ScrnActHeight**	Screennail picture active region height

Type	Field	Description
UINT16	**ThmbActWidth**	Thumbnail picture active region width
UINT16	**ThmbActHeight**	Thumbnail picture active region height
UINT16	**ScrnDepth**	Number of pictures pScrnBufAddr can store
UINT16	**ThmbDepth**	Number of pictures pThmbBufAddr can store
AMBA_DSP_YUV_IMG_BUF_s	***pScrnBufAddr**	Screennail YUV buffer config. Please refer to Section 6.2.1.5 for more details.
AMBA_DSP_YUV_IMG_BUF_s	***pThmbBufAddr**	Thumbnail YUV buffer configuration. Please refer to Section 6.2.1.5 for more details.

*Table 7-6. Definition of **AMBA_DSP_STILL_RAW_CAPTURE_CONFIG_s** for Still Capture API **AmbaDSP_StillRawCaptureConfig ()**.*

7.2.3 AmbaDSP_JpegEncConfig

API Syntax:

AmbaDSP_JpegEncConfig (AMBA_JPEG_ENC_CONFIG_s *pJpegEncConfig)

Function Description:

- This function is used to configure the JPEG encode settings.

Parameters:

Type	Parameter	Description
AMBA_JPEG_ENC_CONFIG_s	***pJpegEncConfig**	Pointer to the JPEG encode configuration. Please refer to Section 7.2.3.1 for more details.

*Table 7-7. Parameters for Still Capture API **AmbaDSP_JpegEncConfig()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 7-8. Returns for Still Capture **API AmbaDSP_JpegEncConfig()**.*

Example:

```
 #define AMBA_DSP_JPEGENC_BITS_BUF_SIZE     0x1000000       /* 16MB MJPEG/JPEG
Encode bitstream buffer */
    UINT8 AmbaDSP_JpegEncBitsBuffer[AMBA_DSP_JPEGENC_BITS_BUF_SIZE];
    AMBA_JPEG_ENC_CONFIG_s             JpegEncConfig;
    UINT8 AmbaJpegQTable[2][128] = {
    {
        0x10, 0x0B, 0x0C, 0x0E, 0x0C, 0x0A, 0x10, 0x0E,
        0x0D, 0x0E, 0x12, 0x11, 0x10, 0x13, 0x18, 0x28,
        0x1A, 0x18, 0x16, 0x16, 0x18, 0x31, 0x23, 0x25,
        0x1D, 0x28, 0x3A, 0x33, 0x3D, 0x3C, 0x39, 0x33,
        0x38, 0x37, 0x40, 0x48, 0x5C, 0x4E, 0x40, 0x44,
        0x57, 0x45, 0x37, 0x38, 0x50, 0x6D, 0x51, 0x57,
        0x5F, 0x62, 0x67, 0x68, 0x67, 0x3E, 0x4D, 0x71,
        0x79, 0x70, 0x64, 0x78, 0x5C, 0x65, 0x67, 0x63,
        0x11, 0x12, 0x12, 0x18, 0x15, 0x18, 0x2F, 0x1A,
        0x1A, 0x2F, 0x63, 0x42, 0x38, 0x42, 0x63, 0x63,
        0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63,
        0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63,
        0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63,
        0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63,
        0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63
    },
    {
```

```
        0x10, 0x0B, 0x0C, 0x0E, 0x0C, 0x0A, 0x10, 0x0E,
        0x0D, 0x0E, 0x12, 0x11, 0x10, 0x13, 0x18, 0x28,
        0x1A, 0x18, 0x16, 0x16, 0x18, 0x31, 0x23, 0x25,
        0x1D, 0x28, 0x3A, 0x33, 0x3D, 0x3C, 0x39, 0x33,
        0x38, 0x37, 0x40, 0x48, 0x5C, 0x4E, 0x40, 0x44,
        0x57, 0x45, 0x37, 0x38, 0x50, 0x6D, 0x51, 0x57,
        0x5F, 0x62, 0x67, 0x68, 0x67, 0x3E, 0x4D, 0x71,
        0x79, 0x70, 0x64, 0x78, 0x5C, 0x65, 0x67, 0x63,
        0x11, 0x12, 0x12, 0x18, 0x15, 0x18, 0x2F, 0x1A,
        0x1A, 0x2F, 0x63, 0x42, 0x38, 0x42, 0x63, 0x63,
        0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63,
        0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63,
        0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63,
        0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63,
        0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63, 0x63
    },
    };
    JpegEncConfig.MaxNumQTable  =  2;
    JpegEncConfig.pQTable =  AmbaJpegQTable;
    JpegEncConfig.pBitsBufAddr  =  AmbaDSP_JpegEncBitsBuffer;
    JpegEncConfig.BitsBufSize  =  AMBA_DSP_JPEGENC_BITS_BUF_SIZE;
    AmbaDSP_JpegEncConfig(&JpegEncConfig);
```


See Also:

AmbaDSP_JpegEncBitRateCtrl
AmbaDSP_JpegEncode

7.2.3.1 AmbaDSP_JpegEncConfig > AMBA_DSP_JPEG_ENC_CONFIG_s

Type	Field	Description
int	**MaxNumQTable**	Number of Q-tables
UINT8	**(*pQTable)[128]**	Pointer to Q-table array (size of each Q-table is 128 bytes)
UINT8	***pBitsBufAddr**	Pointer to the circular bitstream buffer from which the DSP will generate JPEGs. This pointer must point to a valid DRAM address prior to a first-time JPEG encode. After a first-time encode, the pointer can point to address 0xFFFFFFFF, which directs the DSP to generate the bitstream to the address directly after the last JPEG bitstream in the circular buffer. To reset the address or to assign a new DRAM buffer, this pointer must simply be pointed to a valid address. However, for Video PIV mode, only the first-time setting will take effect during a one-time recording. This is due to the fact that the pointer cannot be set again until video recording terminates.
UINT32	**BitsBufSize**	Bitstream buffer size (bytes)

*Table 7-9. Definition of **AMBA_DSP_JPEG_ENC_CONFIG_s** for Still Capture API **AmbaDSP_JpegEncConfig ()**.*

7.2.4 AmbaDSP_JpegEncBitRateCtrl

API Syntax:

AmbaDSP_JpegEncBitRateCtrl (AMBA_DSP_JPEG_BIT_RATE_CTRL_s *pBitRateCtrl)

Function Description:

- This function is used to control the JPEG encode bit rate.

Parameters:

Type	Parameter	Description
AMBA_DSP_JPEG_BIT_RATE_CTRL_s	***pBitRateCtrl**	Pointer to JPEG bit rate control. Please refer to Section 7.2.4.1 for more details.

*Table 7-10. Parameters for Still Capture API **AmbaDSP_JpegEncBitRateCtrl()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 7-11. Returns for Still Capture API **AmbaDSP_JpegEncBitRateCtrl()**.*

Example:

None

See Also:

AmbaDSP_JpegEncode
AmbaDSP_JpegEncConfig

7.2.4.1 AmbaDSP_JpegEncBitRateCtrl > AMBA_DSP_JPEG_BIT_RATE_CTRL_s

Type	Field	Description
UINT16	**QualityLevel**	Initial quality level
UINT16	**TargetBitRate**	Target bits per pixel
UINT32	**Tolerance**	Bitrate tolerance
UINT32	**RateCurvPoints**	Number of rate curve points
UINT8	***pRateCurv**	Pointer to rate curve points buffer
UINT32	**MaxEncLoop**	Maximum encode loop

*Table 7-12. Definition of **AMBA_DSP_JPEG_BIT_RATE_CTRL_s** for Still Capture API **AmbaDSP_JpegEncBitRateCtrl ()**.*

7.2.5 AmbaDSP_StillCaptureConfig

API Syntax:

AmbaDSP_StillCaptureConfig (AMBA_DSP_STILL_CAPTURE_CONFIG_s *pConfig)

Function Description:

- This function is used to configure settings for still-capture . This API will be only take effect if the background still processing has been enabled in Liveview mode. Please refer to the **AmbaDSP_LiveviewConfig** API for details.

Parameters:

Type	Parameter	Description
AMBA_DSP_STILL_CAPTURE_CONFIG_s	***pConfig**	Pointer to still-capture configuration. Please refer to Section 7.2.5.1 for more details.

*Table 7-13. Parameters for Still Capture API **AmbaDSP_StillCaptureConfig()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 7-14. Returns for Still Capture API **AmbaDSP_StillCaptureConfig()**.*

Example:

```
#define AMBA_STILL_RAW_BUF_SIZE             (4000 * 3008 * 2)    /* Raw buffer
*/
UINT8 AmbaDSP_StillCapRawBuf[AMBA_STILL_RAW_BUF_SIZE];
AMBA_DSP_STILL_CAPTURE_CONFIG_s StillCfg;
AMBA_DSP_RAW_BUF_s RawInfo;

memset(&RawInfo, 0, sizeof(RawInfo));
RawInfo.pBaseAddr = AmbaDSP_StillCapRawBuf;
RawInfo.Compressed = 1;
if (RawInfo.Compressed) {
    RawInfo.Pitch = ALIGN32((4000 * 27 >> 5)); /* (6.75 / 8) */
} else {
    RawInfo.Pitch = ALIGN32(4000) << 1;
}
RawInfo.Window.Width = ALIGN32(4000);
RawInfo.Window.Height = ALIGN16(3000);
```

```
memset(&StillCfg, 0, sizeof(StillCfg));
StillCfg.OpMode = 2;        /* Currently only fast mode verified */
StillCfg.MaxNumPicture = 1;
StillCfg.pRawBufAddr = &RawInfo;
AmbaDSP_StillCaptureConfig(&StillCfg);
```

See Also:

AmbaDSP_StillCaptureRaw

7.2.5.1 AmbaDsp_StillCaptureConfig > AMBA_DSP_STILL_CAPTURE_ CONFIG_s

Type	Field	Description
AMBA_DSP_ STILL_PRO-CESS_MODE_e	**OpMode**	Capture operation mode. Please refer to Section 7.2.5.2 below for more details.
Int	**RawBufDepth**	Raw buffer depth: Buffer capacity
AMBA_DSP_ RAW_BUF_s	* **pRawBufAddr**	Pointer to Raw buffers. Please refer to Section 7.2.5.3 below for more details.

*Table 7-15. Definition of **AMBA_DSP_STILL_CAPTURE_CONFIG_s** for Still Capture API **AmbaDsp_StillCaptureConfig()**.*

7.2.5.2 AmbaDsp_StillCaptureConfig > AMBA_DSP_STILL_OP_MODE_e

Type	Description
AMBA_DSP_STILL_HIGH_ISO	High-ISO
AMBA_DSP_STILL_LOW_ISO	Low-ISO
AMBA_DSP_STILL_FAST_MODE	Fast mode
AMBA_DSP_STILL_MID_ISO_MODE	Middle_ISO mode
AMBA_DSP_STILL_MFHISO_MODE	Multi-frame HISO mode
AMBA_DSP_STILL_NP_MODE	Night portrait mode

*Table 7-16. Definition of **AMBA_DSP_STILL_OP_MODE_e** for Still Capture API **AmbaDsp_StillCaptureConfig()**.*

7.2.5.3 AmbaDsp_StillCaptureConfig > AMBA_DSP_RAW_BUF_s

Type	Field	Description
UINT8	**Compressed**	0: Uncompressed raw data 1: Compressed raw data
UINT8	***pBaseAddr**	Pointer to a DRAM image buffer
UINT16	**Pitch**	Buffer pitch
AMBA_DSP_ WINDOW_s	**Window**	Window position and size. Please refer to Section 6.2.1.7 for more details.

*Table 7-17. Definition of **AMBA_DSP_RAW_BUF_s** for Still Capture API **AmbaDsp_StillCaptureConfig()**.*

7.2.6 AmbaDSP_StillQuickviewConfig

API Syntax:

AmbaDSP_StillQuickviewConfig (AMBA_DSP_ QUICKVIEW_ CONFIG_s *pConfig)

Function Description:

- This API is used to configure quickview settings. The quickview feature allows the most recently recorded still-image to be displayed briefly on the television and LCD device immediately after capture. The quickview image will be allocated in the external buffer. Note that this API must be configured before the **AmbaDSP_StillRaw2Yuv()** API is called. After calling **AmbaDSP_StillRaw2Yuv()**, the system is required to wait for the **AMBA_DSP_EVENT_SCAP_YUV_DATA_READY** event prior to retrieving the quickview image data.

Parameters:

Type	Parameter	Description
AMBA_DSP_ QUICKVIEW_ CONFIG_s	***pConfig**	Pointer to the quickview information. Please refer to Section 7.2.6.1 below for more information.

*Table 7-18. Parameters for Still Capture API **AmbaDSP_StillQuickviewConfig()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 7-19. Returns for Still Capture API **AmbaDSP_StillQuickviewConfig()**.*

Example:

```
UINT8 AmbaDSP_StillCapYuvBufLCD[960 * 480 * 2];
UINT8 AmbaDSP_StillCapYuvBufHDTV[1920 * 1088 * 2];
AMBA_DSP_QUICKVIEW_CONFIG_s QVConfig;
AMBA_DSP_YUV_IMG_BUF_s QV_LCD, QV_TV;

memset(&QVConfig, 0, sizeof(QVConfig));
memset(&QV_LCD, 0, sizeof(QV_LCD));
memset(&QV_TV, 0, sizeof(QV_TV));

QV_LCD.DataFmt     = AMBA_DSP_YUV422;
QV_LCD.pBaseAddrY  = AmbaDSP_StillCapYuvBufLCD;
QV_LCD.Pitch       = ALIGN32(960);
QV_LCD.Window.Width  = ALIGN32(960);
QV_LCD.Window.Height = ALIGN16(480);
```

```
QV_TV.DataFmt     = AMBA_DSP_YUV422;
QV_TV.pBaseAddrY  = AmbaDSP_StillCapYuvBufHDTV;
QV_TV.Pitch       = ALIGN32(1920);
QV_TV.Window.Width  = ALIGN32(1920);
QV_TV.Window.Height = ALIGN16(1080);

QVConfig.pYuvQuickViewLCD = &QV_LCD;
QVConfig.pYuvQuickViewTV = &QV_TV;
QVConfig.Generate = 1;

AmbaDSP_StillQuickviewConfig(&QVConfig);
```

See Also:

AmbaDsp_StillRaw2Yuv

7.2.6.1 AmbaDsp_Quickviewconfig > AMBA_DSP_QUICKVIEW_CONFIG_s

Type	Field	Description
UINT8	**Generate**	0: Do not generate quickview. 1: Generate quickview.
UINT8	**Fastquickview**	Valid for Generate = 1. 0: Generate quickview from YUV data. 1: Generate quickview from downsized RAW data.
AMBA_DSP_QUICKVIEW_YUV_IMG_BUF_s	***pYuvQuickViewLCD**	Pointer to YUV buffer for LCD quickview. Please refer to Section 7.2.6.2 for more details. This pointer must be set to indicate Quickview YUV buffer for LCD or Thumbnail YUV buffer.
AMBA_DSP_QUICKVIEW_YUV_IMG_BUF_s	***pYuvQuickViewTV**	Pointer to YUV buffer for television quickview. Please refer to Section 7.2.6.2 for more details. This pointer is used to specify the Quickview YUV buffer for television or Screennail YUV buffer. If the user does not set this pointer, **pWorkYuvBufAddr** must be set in **AmbaDsp_StillRaw2Yuv()**.
AMBA_DSP_RAW_BUF_s	***pFastQuickRawBufAddr**	Pointer to Raw buffer for Fast Quickview. Please refer to Section 7.2.5.3 for more details.

*Table 7-20. Definition of **AMBA_DSP_QUICKVIEW_CONFIG_s** for Still Capture API **AmbaDSP_StillQuickviewConfig()**.*

7.2.6.2 AmbaDsp_Quickviewconfig > AMBA_DSP_QUICKVIEW_CONFIG_s

Type	Field	Description
AMBA_ DSP_YUV_ FORMAT_e	**DataFmt**	YUV Data format
UINT8	***pBaseAddrY**	Pointer to Luma (Y) data area
UINT8	***pBaseAddrUV**	Pointer to Chroma (UV) data area
UINT32	**Pitch**	YUV data buffer pitch
UINT16	**ActiveWidth**	Active width
UINT16	**ActiveHeight**	Active height
AMBA_DSP_ WINDOW_s	**Window**	Window position and size. Please refer to Section 6.2.1.7 for more details.

*Table 7-21. Definition of **AMBA_DSP_QUICKVIEW_CONFIG_s** for Still Capture API **AmbaDSP_StillQuickviewConfig()**.*

7.2.7 AmbaDSP_StillCaptureRaw

API Syntax:

AmbaDSP_StillCaptureRaw (int NumRawPicture)

Function Description:

- This function is used to configure settings for the capture of raw (RAW) data. This API will only take effect in Liveview mode with the background still processing enabled. The user will receive an **AMBA_DSP_EVENT_SCAP_RAW_DATA_READY** event with the RAW buffer configured in **AmbaDSP_StillCaptureConfig()** after the completion of each RAW frame capture. When performing still capture only, this API must be called after calling **AmbaDSP_LiveviewCtrl**(0, 0). If this API is called while background still-processing is disabled during Photo In Video (PIV) recording, the user will receive an event with invalid RAW buffer address 0xFFFFFFFF.

Parameters:

Type	Parameter	Description
int	**NumRawPicture**	Number of raw frames to capture. 1: Single capture > 1: Burst capture 0: Stop burst capture if it is able to be stopped. Stopping burst capture is a blocking API, and it will return when capture terminates.

*Table 7-22. Parameters for Still Capture API **AmbaDSP_StillCaptureRaw()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 7-23. Returns for Still Capture API **AmbaDSP_StillCaptureRaw()**.*

Example:

None

See Also:

AmbaDSP_StillCaptureConfig

7.2.8 AmbaDSP_StillGenerateRaw3A

API Syntax:

AmbaDSP_StillGenerateRaw3A (AMBA_DSP_RAW_BUF_s *pRawBufAddr, AMBA_DSP_WINDOW_s *pRegionAddr)

Function Description:

- This function is used to generate raw AAA statistics during still-image capture. This API will only take effect in Liveview mode with background still processing enabled. It is not a blocking API. The system cannot call **AmbaDsp_StillRaw2Yuv()**, **AmbaDsp_StillGenerateScrnNails()**, or **AmbaDSP_LiveviewCtrl()** APIs until receiving an **AMBA_DSP_EVENT_RAW_CFA_3A_DATA_READY** event.

Parameters:

Type	Parameter	Description
AMBA_DSP_RAW_BUF_s	* **pRawBufAddr**	Pointer to input Raw Buffer information. Please refer to Section 7.2.5.3 for more details.
AMBA_DSP_WINDOW_s	***pRegionAddr**	Pointer to AAA statistics region window. Please refer to Section 6.2.1.7 for more details.

*Table 7-24. Parameters for Still Capture API **AmbaDSP_StillGenerateRaw3A()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 7-25. Returns for Still Capture API **AmbaDSP_StillGenerateRaw3A()**.*

Example:

None

See Also:

AmbaDsp_StillRaw2Yuv
AmbaDsp_StillGenerateScrnNails
AmbaDSP_LiveviewCtrl

7.2.9 AmbaDsp_StillRaw2Yuv

API Syntax:

AmbaDsp_StillRaw2Yuv (AMBA_DSP_RAW_BUF_s *pRawBufAddr, AMBA_DSP_YUV_IMG_BUF_s *pYuvBufAddr, AMBA_DSP_QUICKVIEW_YUV_IMG_BUF_s *pWorkYuvBufAddr)

Function Description:

- This function is used to configure settings for RAW-to-YUV conversion. This API generates:
  - The main YUV datastream
  - One YUV datastream for TV quickview, screennail, or temporary viewing
  - One YUV datastream for LCD quickview or thumbnail.
- Referring to the figure below, the user will receive three **AMBA_DSP_EVENT_SCAP_YUV_DATA_READY** events with YUV buffers configured in this API and **AmbaDSP_StillQuickviewConfig()** after each YUV datastream is generated. If this API is called while the background still-processing is disabled during Photo In Video (PIV) recording, the user will receive the three events with the invalid YUV buffer address 0xFFFFFFFF.

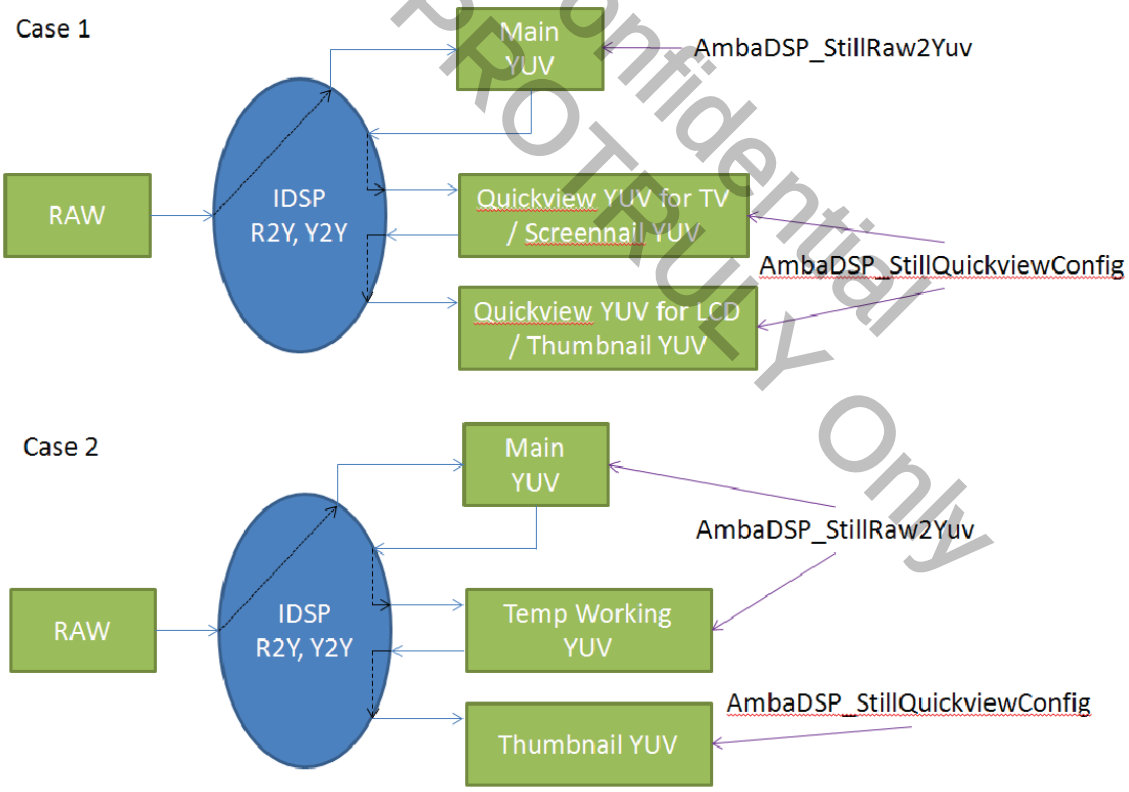

Figure 7-1. RAW-to-YUV conversion.

Parameters:

Type	Parameter	Description
AMBA_DSP_RAW_BUF_s	***pRawBufAddr**	Number of raw frames to capture. Please refer to Section 7.2.5.3 for more details.
AMBA_DSP_YUV_IMG_BUF_s	***pYuvBufAddr**	Maximum number of raw frames in the buffer. Please refer to Section 6.2.1.5 for more details.
AMBA_DSP_QUICKVIEW_YUV_IMG_BUF_s	***pWorkYuvBufAddr**	Pointer to the temporary working YUV buffer for **pYuvQuickViewTV** is not assigned in **AmbaDSP_StillQuickviewConfig**(). Please refer to Section 7.2.6.2 for more details.

*Table 7-26. Parameters for Still Capture API **AmbaDsp_StillRaw2Yuv()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 7-27. Returns for Still Capture API **AmbaDsp_StillRaw2Yuv()**.*

Example:

```
#define AMBA_STILL_MAIN_BUF_SIZE            0x2000000       /* 32MB decode main
buffer (Y + UV) */
UINT8 AmbaDSP_StillMainYuvBuf[AMBA_STILL_MAIN_BUF_SIZE];
AMBA_DSP_RAW_BUF_s                  RawInfo;
AMBA_DSP_YUV_IMG_BUF_s          YuvInfo;

memset(&YuvInfo, 0, sizeof(YuvInfo));
YuvInfo.DataFmt     = AMBA_DSP_YUV422;
YuvInfo.pBaseAddrY  = AmbaDSP_StillMainYuvBuf;
YuvInfo.Pitch         = ALIGN32(4000);
YuvInfo.Window.Width  = ALIGN32(4000);
YuvInfo.Window.Height = ALIGN16(3000);

RawInfo.Compressed     = 1;
RawInfo.pBaseAddrY  = AmbaDSP_StillCapRawBuf;
RawInfo.Pitch         = ALIGN32(4000);
RawInfo.Window.Width  = ALIGN32(4000);
RawInfo.Window.Height = ALIGN16(3000);

AmbaDSP_StillRaw2Yuv(&RawInfo, &YuvInfo, NULL, NULL);
```

See Also:

AmbaDSP_StillQuickviewConfig

7.2.9.1 AmbaDsp_StillRaw2Yuv > AMBA_DSP_BUF_s

Type	Field	Description
UINT8	***pBaseAddr**	Pointer to a DRAM image buffer
UINT16	**Pitch**	Buffer pitch
AMBA_DSP_WINDOW_s	**Window**	Window position and size. Please refer to Section 6.2.1.7 for more details.

*Table 7-28. Definition of **AMBA_DSP_BUF_s** for Still Capture API **AmbaDsp_StillRaw2Yuv()**.*

7.2.10 AmbaDsp_StillGenerateScrnNails

API Syntax:

AmbaDsp_StillGenerateScrnNails (AMBA_DSP_YUV_IMG_BUF_s *pYuvBufAddr, int NumScrnNail, AMBA_DSP_STILL_SCRN_NAIL_BUF_s *pScrnNailBufAddr)

Function Description:

- This function generates YUV thumbnails and screennails from the main YUV data. This API is a blocking function, and will return until all thumbnail/screennails are generated. The user will receive **AMBA_DSP_EVENT_SCAP_YUV_DATA_READY** events with thumbnail/screennail buffers configured in this API. This API will only take effect when background still processing is enabled in Liveview mode. During PIV recording mode, the thumbnail and screennail are configured by **AmbaDSP_StillQuickviewConfig()** due to the fact that quickview is disabled.

Parameters:

Type	Parameter	Description
AMBA_DSP_YUV_IMG_BUF_s	***pYuvBufAddr**	Pointer to main YUV buffer information. Please refer to Section 6.2.1.5 for more details.
int	**NumScrnNail**	Number of screennails
AMBA_DSP_STILL_SCRN_NAIL_BUF_s	***pScrnNailBufAddr**	Pointer to screennail buffer information. Please refer to Section 7.2.10.1 below for details.

*Table 7-29. Parameters for Still Capture API **AmbaDsp_StillGenerateScrnNails()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 7-30. Returns for Still Capture API **AmbaDsp_StillGenerateScrnNails()**.*

Example:

```
UINT8 AmbaDSP_StillScrnailBuf[960 * 720 * 2];
AMBA_DSP_YUV_IMG_BUF_s         QV_TV;
AMBA_DSP_STILL_SCRN_NAIL_BUF_s ScrNail;

memset(&QV_TV, 0, sizeof(QV_TV));
QV_TV.DataFmt     = AMBA_DSP_YUV422;
QV_TV.pBaseAddrY  = AmbaDSP_StillCapYuvBufHDTV;
QV_TV.Pitch       = ALIGN32(1920);
QV_TV.Window.Width  = ALIGN32(1920);
QV_TV.Window.Height = ALIGN16(1080);
```

```
memset(&ScrNail, 0, sizeof(ScrNail));
ScrNail.Width = 960;
ScrNail.Height = 720;
ScrNail.ActiveWidth = 960;
ScrNail.ActiveHeight = 720;
ScrNail.pBufAddr = AmbaDSP_StillScrnailBuf;
AmbaDSP_StillGenerateScrnNails(&YuvInfo, 1, &ScrNail);
```

See Also:

None

7.2.10.1 AmbaDsp_StillGenerateScrnNails > AMBA_DSP_STILL_SCRN_NAIL_BUF_s

Type	Field	Description
UINT16	**Width**	Screennail width
UINT16	**Height**	Screennail height
UINT16	**ActiveWidth**	Screennail active width
UINT16	**ActiveHeight**	Screennail active height
UINT8	***pBufAddr**	Pointer to the screennail YUV buffer

*Table 7-31. Definition of **AMBA_DSP_STILL_SCRN_NAIL_BUF_s** for Still Capture API **AmbaDsp_StillGenerateScrn-Nail ()**.*

7.2.11 AmbaDSP_JpegEncode

API Syntax:

AmbaDSP_JpegEncode (int NumImage, AMBA_DSP_JPEG_ENC_CTRL_s *pJpegEncCtrl)

Function Description:

- This function is used to encode a JPEG from YUV data. When background still processing is enabled in the Liveview mode, this is a blocking API, and it will return after all JPEGs are generated. The user will receive one **AMBA_DSP_EVENT_JPEG_DATA_READY** event for each JPEG bitstream.
- When background still processing is disabled during PIV recording, this is not a blocking API, and it will generate the main JPEG bitstream from **pYuvBufAddr** in **AmbaDSP_StillRaw2Yuv()**, and thumbnail/screennail JPEG bitstreams from **AmbaDSP_StillQuickviewConfig()**. The user will receive three **AMBA_DSP_EVENT_JPEG_DATA_READY** events: one for main, screennail, and thumbnail JPEG bitstreams.

Parameters:

Type	Parameter	Description
int	**NumImage**	Number of images to encode
AMBA_DSP_JPEG_ENC_CTRL_s	***pJpegEncCtrl**	Pointer to JPEG encode control array. Please refer to Section 7.2.11.1 for more details.

*Table 7-32. Parameters for Still Capture API **AmbaDSP_JpegEncode()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 7-33. Returns for Still Capture API **AmbaDSP_JpegEncode()**.*

Example:

```
#define ALIGN16(x) ((x+15)&0xfff0)
#define ALIGN32(x) ((x+31)&0xffe0)
#define AMBA_STILL_MAIN_BUF_SIZE            0x2000000       /* 32MB decode main
buffer (Y + UV) */
UINT8 AmbaDSP_StillMainYuvBuf[AMBA_STILL_MAIN_BUF_SIZE];
AMBA_DSP_JPEG_ENC_CTRL_s EncCtrl;
AMBA_DSP_YUV_IMG_BUF_s      YuvInfo;
memset(&YuvInfo, 0, sizeof(YuvInfo));
YuvInfo.DataFmt     = AMBA_DSP_YUV422;
YuvInfo.pBaseAddrY  = AmbaDSP_StillMainYuvBuf;
YuvInfo.Pitch         = ALIGN32(4000);
YuvInfo.Window.Width  = ALIGN32(4000);
YuvInfo.Window.Height = ALIGN16(3000);
EncCtrl.pYuvBuf = &YuvInfo;
```

```
EncCtrl.QTableIdx = 1; /* choose Q-table configured in AmbaDSP_JpegEncConfig()
*/
EncCtrl.EncWidth = 4000;
EncCtrl.EncHeight = 3000;
AmbaDSP_JpegEncode(1, &EncCtrl);
```

See Also:

AmbaDSP_JpegEncBitRateCtrl
AmbaDSP_JpegEncConfig

7.2.11.1 AmbaDSP_JpegEncode > AMBA_DSP_JPEG_ENC_CTRL_s

Type	Field	Description
AMBA_DSP_YUV_IMG_BUF_s	***pYuvBuf**	Pointer to source YUV buffer. Please refer to Section 6.2.1.5 for more details.
UINT8	**QTableIdx**	Q-table index, the Q-tables are assigned by **AmbaDSP_JpegEncConfig()**.
UINT16	**EncWidth**	Main JPEG width
UINT16	**EncHeight**	Main JPEG height
AMBA_DSP_ROTATE_FLIP_e	**RotateFlip**	Rotation and flip setting. Please refer to Section 5.2.8.1 for more details.

*Table 7-34. Definition of **AMBA_DSP_JPEG_ENC_CTRL_s** for Still Capture API **AmbaDSP_JpegEncode ()**.*

7.2.12 AmbaDSP_StillYuvBlend

API Syntax:

AmbaDsp_StillYuvBlend (AMBA_DSP_YUV_IMG_BUF_s *pSrc1Yu- vBuf, AMBA_DSP_YUV_IMG_BUF_s *pSrc2YuvBuf, AMBA_DSP_BUF_s *pAlphaBuf, AMBA_DSP_YUV_IMG_BUF_s *pDestYuvBuf)

Function Description:

- This function is used to blend two YUV buffers with an alpha matrix.

Parameters:

Type	Parameter	Description
AMBA_DSP_YUV_IMG_BUF_s	***pSrc1YuvBuf**	Pointer to source 1 YUV buffer information. Please refer to Section 6.2.1.5 for more details.
AMBA_DSP_YUV_IMG_BUF_s	***pSrc2YuvBuf**	Pointer to source 2 YUV buffer information. Please refer to Section 6.2.1.5 for more details.
AMBA_DSP_BUF_s	***pAlphaBuf**	Pointer to alpha matrix. Please refer to Section 6.2.1.5 for more details.
AMBA_DSP_YUV_IMG_BUF_s	***pDestYuvBuf**	Pointer to destination YUV buffer information. Please refer to Section 6.2.1.5 for more details.

*Table 7-35. Parameters for Still Capture API **AmbaDsp_StillYuvBlend()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 7-36. Returns for Still Capture API **AmbaDsp_StillYuvBlend()**.*

Example:

None

See Also:

None

8 Still Decode

8.1 Still Decode: Overview

This chapter details the functions used to configure the JPEG decoder, decode the JPEG bitstream to the YUV DRAM buffer, execute YUV-related operations, and display YUV images to the VOUT device.

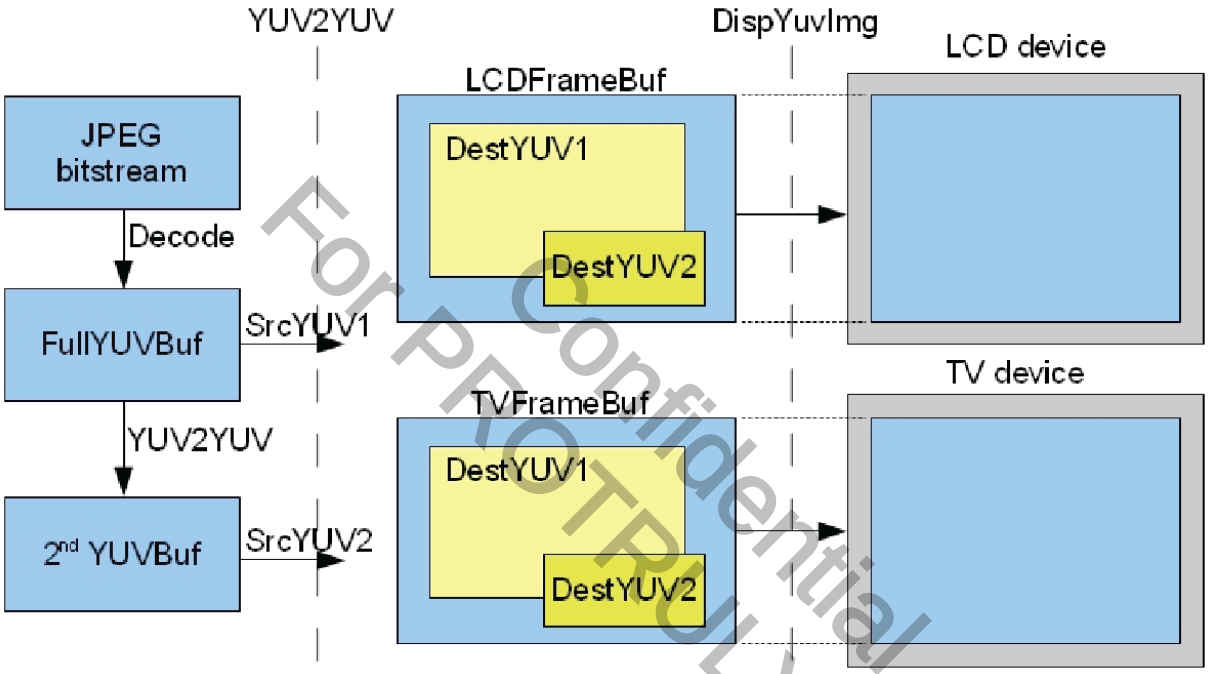

Figure 8-1. YUV operation.

8.2 Still Decode: List of Functions

8.2.1 AmbaDSP_StillDecConfig

API Syntax:

AmbaDSP_StillDecConfig (int ImgDataFmt, UINT32 MaxYuvBufSize, UINT8 *pYuvBufAddr)

Function Description:

- This function is used to configure settings for the block-based decoding process.

Parameters:

Type	Parameter	Description
int	**ImgDataFmt**	Image data format 0: JPEG 1: H.264 I-Frame
UINT32	**MaxYuvBufSize**	Maximum size of the YUV buffer (Byte)
UINT8	***pYuvBufAddr**	Base address of the YUV buffer

*Table 8-1. Parameters for Still Decode API **AmbaDSP_StillDecConfig()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 8-2. Returns for Still Decode API **AmbaDSP_StillDecConfig()**.*

Example:

None

See Also:

AmbaDSP_StillDecSetBitsDataBlk

8.2.2 AmbaDSP_StillDecSetBitsDataBlk

API Syntax:

AmbaDSP_StillDecSetBitsDataBlk (int LastBlkFlag, UINT32 BlkDataSize, UINT8 *pBlkDataBase)

Function Description:

- This function is used to start block-based decoding and to pass bitstream block information to the decoder.
- The API is designed to allow bitstream loading and bitstream decoding to execute in parallel. The user can partition the JPEG bitstream to several different blocks. For example, 100 KB can be allocated for each data block. These data blocks should then be loaded to the bitstream buffer block-by-block. Whenever one data block has been loaded to the bitstream buffer, this API should be called to inform the decoder to begin decoding. The **AMBA_DSP_EVENT_JPEG_DEC_STATUS_REPORT** event handler will be triggered when decoding is complete.

Parameters:

Type	Parameter	Description
int	**LastBlkFlag**	0: Others 1: Last block data
UINT32	**BlkDataSize**	Size of block data (Byte)
UINT32	***pBlkDataBase**	Start address of block data

*Table 8-3. Parameters for Still Decode API **AmbaDSP_StillDecSetBitsDataBlk()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 8-4. Returns for Still Decode API **AmbaDSP_StillDecSetBitsDataBlk()**.*

Example:

```
#define  YUV_BUFFER_SIZE  0x1000000
#define  JPEG_BLOCK_DATA_SIZE (100 * 1024)    // 100 KB
UINT8   YUVBuffer[YUV_BUFFER_SIZE];
UINT8   BitsBuffer[0x800000];

AmbaDSP_StillDecConfig(1, YUV_BUFFER_SIZE, &YUVBuffer);

/* User load 1st bitstream block to BitsBuffer */

/* DSP start to decode bitstream when received the 1st bitstream block */
AmbaDSP_StillDecSetBitsDataBlk(0, block data size, Start address of the
block);
```

```
/* User load 2nd bitstream block to BitsBuffer. In BitsBuffer, the 2nd block
should follow the 1st block. */

AmbaDSP_StillDecSetBitsDataBlk(0, block data size, Start address of the
block);

(Continuously load bitstream and update bitstream information)

/* User load last bitstream block to BitsBuffer */

AmbaDSP_StillDecSetBitsDataBlk(1, block data size, Start address of the
block);

/* DSP put decoded YUV data to YUVBuffer and trigger AMBA_DSP_EVENT_JPEG_DEC_
STATUS_REPORT event handler */
```

See Also:

None

8.2.3 AmbaDSP_StillDecode

API Syntax:

AmbaDSP_StillDecode (int NumImg, AMBA_DSP_STILL_DEC_BUF_CONFIG_s *pDecBufConfig)

Function Description:

- This function is used to configure settings for decoding a JPEG or H264 I-frame to the YUV buffer.
- This is a blocking function.

Parameters:

Type	Parameter	Description
int	**NumImg**	Number of images to be processed
AMBA_DSP_STILL_DEC_BUF_CONFIG_s	* **pDecBufConfig**	Pointer to decoding configuration. Please refer to Section 8.2.3.1 for more details.

*Table 8-5. Parameters for Still Decode API **AmbaDSP_StillDecode()**.*

Returns:

Return	Description
0	Success
-1	Success

*Table 8-6. Returns for Still Decode API **AmbaDSP_StillDecode()**.*

Example:

```
#define YUV_BUFFER_SIZE   0x1000000
UINT8  YUVBuffer[YUV_BUFFER_SIZE];
UINT8  BitsBuffer[0x800000];
AMBA_DSP_STILL_DEC_BUF_CONFIG_s DecConfig;

/* User load whole bitstream to BitsBuffer */

/* User configure DecConfig */
DecConfig.ImgDataFmt = 0;      //JPEG

AmbaDSP_StillDecode(1, &DecConfig);

/* DSP put decoded YUV data to YUVBuffer and report Status via DecConfig.DecIn-
fo */
/
```

See Also:

None

8.2.3.1 AmbaDSP_StillDecode > AMBA_DSP_STILL_DEC_BUF_CONFIG_s

Type	Field	Description
UINT8	**ImgDataFmt**	Image data format 0: JPEG 1: H.264 I-Frame
UINT32	**BitsDataSize**	Size of the bitstream (Byte)
UINT8	***pBitsDataAddr**	Base address of the bitstream
UINT32	**MaxYuvBufSize**	Maximum size of YUV buffer (Byte)
UINT8	***pYuvBufAddr**	Start address of YUV buffer. The decoded Y data will be put at pYuvBufAddr, and UV data will be put at (pYuvBufAddr + ((DecInfo.ImgHeight + 31) & 0XFFFFFFE0) * DecInfo. YuvPitch)
UINT16	**DecInfo.Width**	Output from DSP/uCode: Width of the image
UINT16	**DecInfo.Height**	Output from DSP/uCode: Height of the image
UINT16	**DecInfo.YuvPitch**	Output from DSP/uCode: YUV data Pitch
UINT16	**DecInfo.YuvFormat**	Output from DSP/uCode: YUV format of the image Please refer to Section 6.2.1.5 for more details.
UINT32	**DecInfo.Status**	Output from DSP/uCode: OK/NG/Error Code. Please refer to Section 8.2.3.2 for more details.

*Table 8-7. Definition of **AMBA_DSP_STILL_DEC_BUF_CONFIG_s** for Still Decode API **AmbaDSP_StillDecode()**.*

8.2.3.2 AmbaDSP_StillDecode > DecInfo.Status

Type	Description
AMBA_DSP_STILL_DEC_DONE	Decode complete
AMBA_DSP_STILL_DEC_PENDING_BITS	Decoder is pending on input bitstream
AMBA_DSP_STILL_DEC_JPEG_HEADER_ERR	JPEG header error
AMBA_DSP_STILL_DEC_JPEG_MCU_ERR	JPEG MCU error
AMBA_DSP_STILL_DEC_H264_DEC_ERR	H264 picture error

*Table 8-8. Definition of **DecInfo.Status** for Still Decode API **AmbaDSP_StillDecode()**.*

8.2.4 AmbaDSP_StillDecAbort

API Syntax:

AmbaDSP_StillDecAbort (void)

Function Description:

- This function is used to terminate the decoding process for **AmbaDSP_StillDecSetBitsDataBlk** or **AmbaDSP_StillDecode**.
- This is a blocking function

Parameters:

None

Returns:

Return	Description
0	Success
-1	Failure

*Table 8-9. Returns for Still Decode API **AmbaDSP_StillDecAbort()**.*

Example:

None

See Also:

None

8.2.5 AmbaDSP_StillDecClearYuvBuf

API Syntax:

AmbaDSP_StillDecClearYuvBuf (AMBA_DSP_YUV_IMG_BUF_s *pYuvBufAddr, UINT32 FillYuvColor)

Function Description:

- This function is used to clear the YUV buffer to a specified fill color.
- This is a blocking function

Parameters:

Type	Parameter	Description
AMBA_DSP_YUV_IMG_BUF_s	***pYuvBufAddr**	Destination YUV buffer. The Height must be multiple of 8. Please refer to Section 6.2.1.5 for more details.
UINT32	**FillYuvColor**	Desired YUV color for filling [0:7]: Y [8:15]: U [16:23]: V

*Table 8-10. Parameters for Still Decode API **AmbaDSP_StillDecClearYuvBuf()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 8-11. Returns for Still Decode API **AmbaDSP_StillDecClearYuvBuf()**.*

Example:

```
AMBA_DSP_YUV_IMG_BUF_s  SrcYUV;
UINT32 FillYuvColor = (0x10 << 16) | (0x80 << 8) | (0x80);  /* Black */

/* User configure source YUV buffer */

AmbaDSP_StillDecClearYuvBuf(&SrcYUV, FillYuvColor);
```

See Also:

None

8.2.6 AmbaDSP_StillDecYuv2Yuv

API Syntax:

AmbaDSP_StillDecYuv2Yuv (int NumImg, AMBA_DSP_YUV_IMG_BUF_s *pSrcYuvBufAddr, AMBA_DSP_YUV_IMG_BUF_s *pDestYuvBufAddr, AMBA_DSP_YUV2YUV_OPERATION_s *pOperation)

Function Description:

- This function is used to configure settings for the YUV-to-YUV process. The YUV-to-YUV process includes rescaling, rotation, and flip.
- This is a blocking function

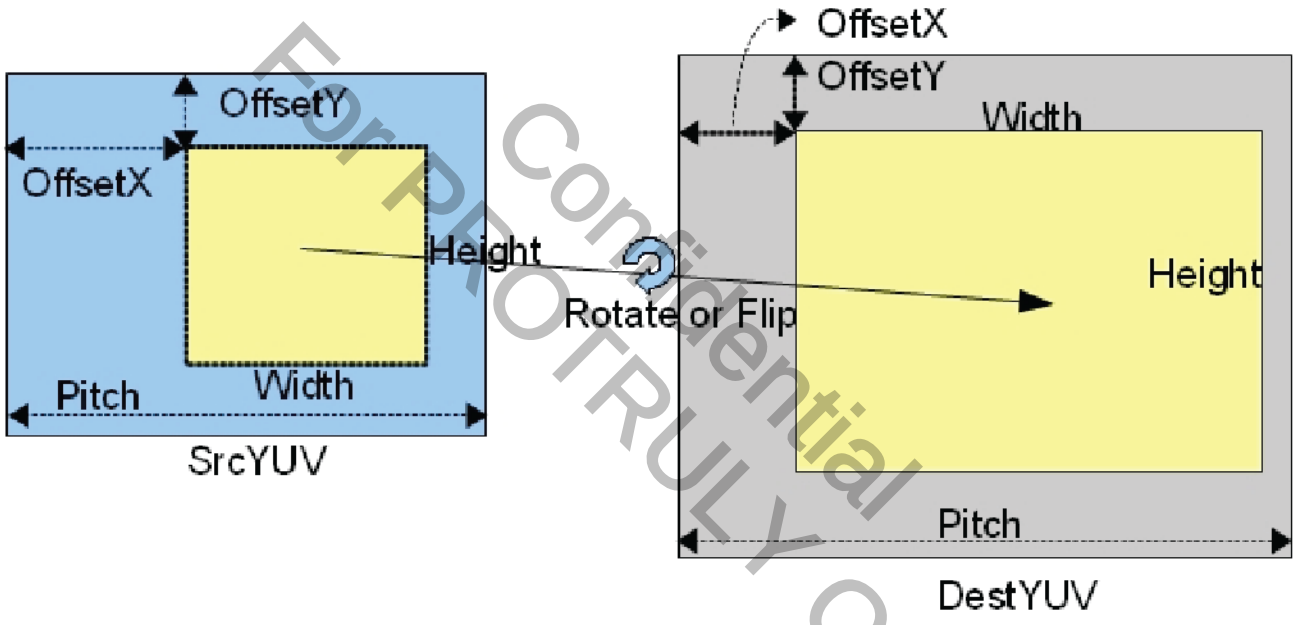

Figure 8-2. YUV to YUV process.

Parameters:

Type	Parameter	Description
UINT32	**NumImg**	Number of images
AMBA_DSP_YUV_IMG_BUF_s	***pSrcYuvBufAddr**	Source YUV buffer. The Width, Height, OffsetX, and OffsetY, must be multiples of 2. Please refer to Section 6.2.1.5 for more details.
AMBA_DSP_YUV_IMG_BUF_s	***pDestYuvBufAddr**	Destination YUV buffer. The Width, Height, OffsetX, and OffsetY, must be multiples of 2. Please refer to Section 6.2.1.5 for more details.
AMBA_DSP_YUV2YUV_OPERATION_s	***pOperation**	YUV operation for Destination YUV buffer. Please refer to Section 8.2.6.1 for more details.

*Table 8-12. Parameters for Still Decode API **AmbaDSP_StillDecYuv2Yuv()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 8-13. Returns for Still Decode API **AmbaDSP_StillDecYuv2Yuv()**.*

Example:

```
AMBA_DSP_YUV_IMG_BUF_s   SrcYUV;
AMBA_DSP_YUV_IMG_BUF_s   DestYUV;
AMBA_DSP_YUV2YUV_OPERATION_s YUVOp;

/* User configure source YUV buffer and destination YUV buffer */
YUVOp.RotateFlip  =  AMBA_DSP_ROTATE_0;
YUVOp.LumaGain    = 128;
AmbaDSP_StillDecYuv2Yuv(1,  &SrcYUV,   &DestYUV,  &YUVOp);
```

See Also:

None

8.2.6.1 AmbaDSP_StillDecYuv2Yuv > AMBA_DSP_YUV2YUV_OPERATION_s

Type	Field	Description
UINT8	**RotateFlip**	Rotation and flip. Please refer to Section 4.2.8.1 for more details.
UINT8	**LumaGain**	Luma gain. The valid value is 1 to 255. 128 is recommended.

*Table 8-14. Definition of **AMBA_DSP_YUV2YUV_OPERATION_s** for Still Decode API **AmbaDSP_StillDecYuv2Yuv()**.*

8.2.7 AmbaDSP_StillDecYuvBlend

API Syntax:

AmbaDSP_StillDecYuvBlend (AMBA_DSP_YUV_IMG_BUF_s *pSrc1YuvBufAddr, AMBA_DSP_YUV_IMG_BUF_s *pSrc2YuvBufAddr, AMBA_DSP_YUV_IMG_BUF_s *pDestYuvBufAddr, AMBA_DSP_BLEND_OPERATION_s *pOperation)

Function Description:

- This function is used to configure settings for the YUV blending process.
- This is a blocking function

Parameters:

Type	Parameter	Description
AMBA_DSP_YUV_IMG_BUF_s	***pSrc1YuvBufAddr**	Source YUV buffer. The Height must be multiple of 16. Please refer to Section 6.2.1.5 for more details.
AMBA_DSP_YUV_IMG_BUF_s	***pSrc2YuvBufAddr**	Source YUV buffer. The Height must be multiple of 16. Please refer to Section 6.2.1.5 for more details.
AMBA_DSP_BLEND_OPERATION_s	***pOperation**	Configuration of blending

*Table 8-15. Parameters for Still Decode API **AmbaDSP_StillDecYuvBlend()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 8-16. Returns for Still Decode API **AmbaDSP_StillDecYuvBlend()**.*

Example:

```
AMBA_DSP_YUV_IMG_BUF_s SrcYUV1, SrcYUV2, DstYUV;
AMBA_DSP_YUV2YUV_OPERATION_s  BlendOp;

/* User configure source YUV buffer and YUV operation */
BlendOp.pAlphaMap   =  0;
BlendOp.GlobalAlpha =  64;
AmbaDSP_StillDecDispYuvImg(&SrcYUV1, &SrcYUV2, &DstYUV, &BlendOp);
```

See Also:

None

8.2.7.1 AmbaDSP_StillDecYuvBlend > AMBA_DSP_BLEND_OPERATION_s

Type	Field	Description
UINT8	***pAlphaMap**	Alpha map
UINT8	**GlobalAlpha**	Use Global alpha value if **pAlphaMap** = NULL. The valid value is 0 to 255.

*Table 8-17. Definition of **AMBA_DSP_BLEND_OPERATION_s** for Still Decode API **AmbaDSP_StillDecYuvBlend()**.*

8.2.8 AmbaDSP_StillDecYuv2RgbConfig

API Syntax:

AmbaDSP_StillDecYuv2RgbConfig (AMBA_DSP_YUV2RGB_CONFIG_s *pYuv2RgbConfig)

Function Description:

- This function is used to configure settings for the YUV to RGB process.

Parameters:

Type	Parameter	Description
AMBA_DSP_YUV2RGB_CONFIG_s	***pYuv2RgbConfig**	Configuration of YUV to RGB process. Please refer to Section 8.2.8.1 for more details.

*Table 8-18. Parameters for Still Decode API **AmbaDSP_StillDecYuv2RgbConfig()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 8-19. Returns for Still Decode API **AmbaDSP_StillDecYuv2RgbConfig()**.*

Example:

None

See Also:

AmbaDSP_StillDecYuv2Rgb()

8.2.8.1 AmbaDSP_StillDecYuv2RgbConfig > AMBA_DSP_YUV2RGB_CONFIG_s

Type	Field	Description
UINT16	**Matrix[9]**	YUV-to-RGB matrix (in row-dominant order). bits [15:13] = Do not care bits [12:0] = sign plus 2.10 bits; 1 sign bit, 2 integer bits and 10 fractional bits.
INT16	**ROffset**	R offset. bits [15:11] = Do not care bits [10:0] = sign plus 10 bits
INT16	**GOffset**	G offset. bits [15:11] = Do not care bits [10:0] = sign plus 10 bits
INT16	**BOffset**	B offset. bits [15:11] = Do not care bits [10:0] = sign plus 10 bits
AMBA_ DSP_STILL_ DEC_RGB_ FORMAT_e	**RgbDataFormat**	RGB data format. Please refer to Section 8.2.8.2 for more details.

*Table 8-20. Definition of **AMBA_DSP_YUV2RGB_CONFIG_s** for Still Decode API **AmbaDSP_StillDecYuv2RgbConfig()**.*

8.2.8.2 AmbaDSP_StillDecYuv2RgbConfig > AMBA_DSP_STILL_DEC_RGB_FORMAT_e

Type	Description
AMBA_DSP_STILL_DEC_ABGR8888	ABGR8888. Alpha is the MSB.
AMBA_DSP_STILL_DEC_ARGB8888	ARGB8888.

*Table 8-21. Definition of **AMBA_DSP_STILL_DEC_RGB_FORMAT_e** for Still Decode API **AmbaDSP_StillDecYuv2RgbConfig()**.*

8.2.9 AmbaDSP_StillDecYuv2Rgb

API Syntax:

AmbaDSP_StillDecYuv2Rgb (AMBA_DSP_YUV_IMG_BUF_s *pSrcYuvBufAddr, AMBA_DSP_BUF_s *pDestRgbBufAddr)

Function Description:

- This function is used to start YUV to RGB process.
- This is a blocking function.

Parameters:

Type	Parameter	Description
AMBA_DSP_YUV_IMG_BUF_s	***pYuvBufAddr**	Source YUV buffer. Please refer to Section 6.2.1.5 for more details.
AMBA_DSP_BUF_s	***pDestRgbBufAddr**	Destination RGB buffer. 4 bytes for each pixel for ARGB8888 and ABGR8888.

*Table 8-22. Parameters for Still Decode API **AmbaDSP_StillDecYuv2Rgb()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 8-23. Returns for Still Decode API **AmbaDSP_StillDecYuv2Rgb()**.*

Example:

```
AMBA_DSP_YUV2RGB_CONFIG_s Y2RConfig;
AMBA_DSP_YUV_IMG_BUF_s    YuvBuf;
AMBA_DSP_BUF_s            RgbBuf;

Y2RConfig.Matrix[0] = 0x400;
Y2RConfig.Matrix[1] = 0xffff;
Y2RConfig.Matrix[2] = 0x59b;
Y2RConfig.Matrix[3] = 0x400;
Y2RConfig.Matrix[4] = 0xfea0;
Y2RConfig.Matrix[5] = 0xfd25;
Y2RConfig.Matrix[6] = 0x400;
Y2RConfig.Matrix[7] = 0x717;
Y2RConfig.Matrix[8] = 0x1;
Y2RConfig.ROffset   = 0xff4d;
Y2RConfig.GOffset   = 0x87;
Y2RConfig.BOffset   = 0xff1d;
Y2RConfig.RgbDataFormat = AMBA_DSP_STILL_DEC_ARGB8888;
AmbaDSP_StillDecYuv2RgbConfig(&Y2RConfig);

/*  User configure source YUV  buffer  and RGB destination buffer */

AmbaDSP_StillDecYuv2Rgb(&YuvBuf, &RgbBuf);
```

See Also:

AmbaDSP_StillDecYuv2RgbConfig()

8.2.10 AmbaDSP_StillDecDispYuvImg

API Syntax:

AmbaDSP_StillDecDispYuvImg (AMBA_DSP_VOUT_IDX_e VoutIdx, AMBA_DSP_YUV_IMG_BUF_s *pYuvBufAddr, AMBA_DSP_VOUT_VIDEO_CONFIG_s *pVoutConfig)

Function Description:

- This function is used to display the contents of a specified YUV buffer to the VOUT device.
- The **AMBA_DSP_EVENT_JPEG_DEC_YUV_DISP_REPORT** event handler will be triggered when the YUV image is displayed.

Parameters:

Type	Parameter	Description
AMBA_DSP_VOUT_IDX_e	**VoutIdx**	Vout index
AMBA_DSP_YUV_IMG_BUF_s	***pYuvBufAddr**	Source YUV buffer. Please refer to Section 8.2.6.1 for more details.
AMBA_DSP_VOUT_VIDEO_CONFIG_s	***pVoutConfig**	VOUT configuration. If **pVoutConfig** = NULL, the decoder will display the YUV image to VOUT without changing VOUT settings. If **pVoutConfig** != NULL, the decoder will display the YUV image to VOUT as well as change VOUT settings.

*Table 8-24. Parameters for Still Decode API **AmbaDSP_StillDecDispYuvImg()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 8-25. Returns for Still Decode API **AmbaDSP_StillDecDispYuvImg()**.*

Example:

```
AMBA_DSP_YUV_IMG_BUF_s  SrcYUV;
AMBA_DSP_VOUT_VIDEO_CONFIG_s  VoutConfig;
/* User configure source YUV buffer */
If (change VOUT setting) {
/* User configure VoutConfig */

AmbaDSP_StillDecDispYuvImg(AMBA_DSP_VOUT_LCD, &SrcYUV,      &VoutConfig);
} else {
AmbaDSP_StillDecDispYuvImg(AMBA_DSP_VOUT_LCD, &SrcYUV, NULL);
}
```

See Also:

AmbaDSP_VoutVideoConfig, AmbaDSP_VoutVideoWindowSetup

8.2.10.1 AmbaDSP_StillDecDispYuvImg > AMBA_DSP_VOUT_VIDE O_CONFIG_s

Type	Parameter	Description
AMBA_DSP_ ROTATE_FLIP_e	**RotateFlip**	Rotation and flip setting. Please refer to Section 4.2.8.1 for more details.
AMBA_DSP_ WINDOW_s	**Window**	VOUT video window. If RotateFlip = AMBA_DSP_ROTATE_90, AMBA_DSP_ROTATE_90_VERT_FLIP, AMBA_DSP_ROTATE_270, or AMBA_DSP_ROTATE_270_VERT_FLIP, Window. Width must be multiple of 8. Please refer to Section 4.2.9.1 for more details.

*Table 8-26. Definition of **AMBA_DSP_VOUT_VIDEO_CONFIG_s** for Still Decode API **AmbaDSP_StillDecDispYuvImg()**.*

9 Video Encode

9.1 Video Encode: Overview

This chapter details the functions used to configure A12 video encoder settings during the compression of H.264 or MJPEG video bitstreams. The following figures illustrate the typical video record process, including the APIs involved during a specific record event.

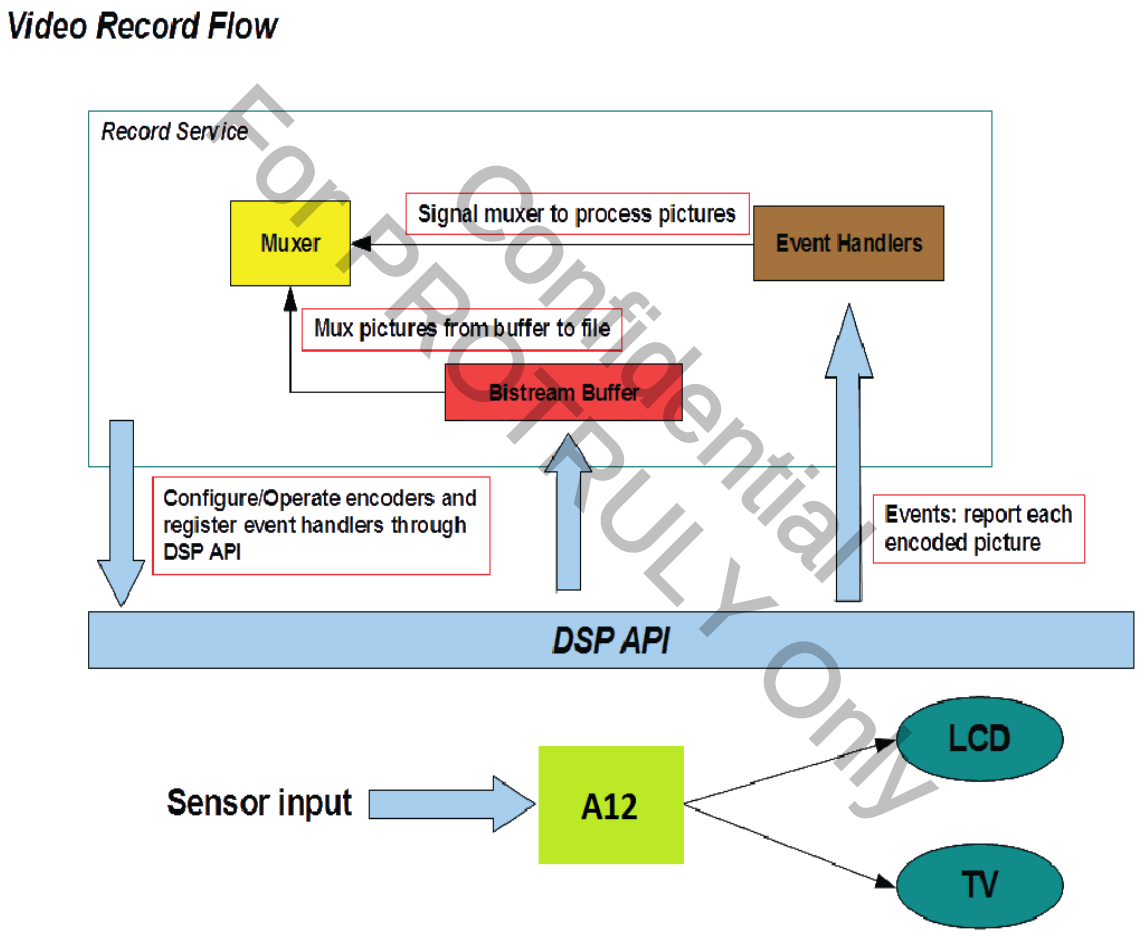

Figure 9-1. Video Record Flow.

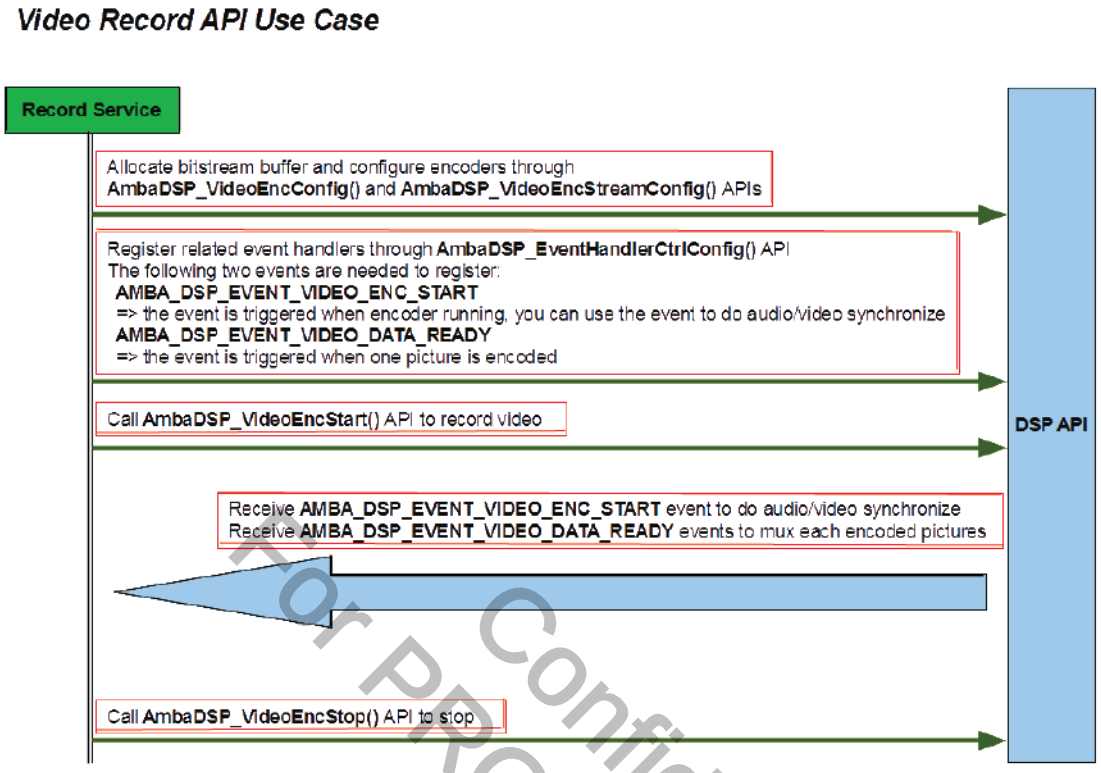

Figure 9-2. Cases of APIs used in video record.

9.2 Video Encode: List of functions

9.2.1 AmbaDSP_SetVideoEncResourceLimitation

API Syntax:

AmbaDSP_SetVideoEncResourceLimitation (AMBA_DSP_VIDEO_ENC_RESOURCE_CONFIG_s * pConfig)

Function Description:

- This function is used to as resource saving flags for DSP so that it can consume less memory.

Parameters:

Type	Parameter	Description
UINT8:1	**DisablePiv**	Disable PIV
UINT8:1	**DisableH264**	Disable H264 encoding
UINT8:1	**DisableMJPEG**	Disable MJPEG encoding
UINT8:1	**Disable2ndStream**	Disable 2nd stream encoding
UINT8:1	**DisableHDPreview**	Disable 2nd stream encoding
UINT8:3	**EnableSqueezeYuvBuffer**	Enable YUV buffer usage reduction
UINT16	**MaxPivWidth**	Max PIV width
UINT16	**MaxPivHeight**	Max PIV height
UINT16	**MaxPivThumbWidth**	Max PIV thumbnail width
UINT16	**MaxPivThumbHeight**	Max PIV thumbnail height

*Table 9-1. Parameters for Video Encode API **AmbaDSP_SetVideoEncResourceLimitation()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 9-2. Returns for Video Encode API **AmbaDSP_SetVideoEncResourceLimitation()**.*

Example:

See Also:

None

9.2.2 AmbaDSP_VideoEncConfig

API Syntax:

AmbaDSP_VideoEncConfig (int MaxNumStream, AMBA_DSP_VIDEO_ENC_STREAM_CONFIG_s *pStreamConfig)

Function Description:

- This function is used to configure settings for video stream encoding. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**MaxNumStream**	Maximum number of video streams to encode
AMBA_DSP_VIDEO_ENC_STREAM_CONFIG_s	***pStreamConfig**	Stream configurations table

*Table 9-3. Parameters for Video Encode API **AmbaDSP_VideoEncConfig()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 9-4. Returns for Video Encode API **AmbaDSP_VideoEncConfig()**.*

Example:

```
/* Declare global variables for encode video streams */
static AMBA_DSP_VIDEO_ENC_STREAM_CONFIG_s  AmbaEncStreamCfg[2];

{
memset(AmbaEncStreamCfg, 0, sizeof(AMBA_DSP_VIDEO_ENC_STREAM_CONFIG_s) * 2);
/* Assign global variables to encoders  */
AmbaDSP_VideoEncConfig(2, AmbaEncStreamCfg);
}
```

See Also:

None

9.2.2.1 AmbaDSP_VideoEncConfig > AMBA_DSP_VIDEO_ENC_ STREAM_CONFIG_s

Type	Field	Description
UINT8	**DataFmt**	0: H.264 1: Motion JPEG
UINT8	**IsIntervalCap**	Set 1 for interval capture.
AMBA_DSP_VIDEO_ STREAM_ID_s	**StreamID**	Video Stream ID. Please refer to Section 9.2.2.2 for more details.
AMBA_DSP_WINDOW_s	**Window**	Window position and size. Please refer to Section 9.2.2.3 for more details.
UINT32	**FrameRateDivisor**	Encode frame rate = VIN frame rate / divisor
AMBA_DSP_VIDEO_ENC_ CONFIG_u	**EncConfig**	Configurations for H.264/MJPEG encoder. Please refer to Section 9.2.2.4 for more details.
AMBA_DSP_ROTATE_FLIP_e	**RotateConfig**	Please refer to Section 4.2.8.1 for more details.
AMBA_DSP_FRAME_RATE_s	**FrameRate**	Encode frame rate. Please refer to Section 2.2.1.4 for more details.

*Table 9-5. Definition of **AMBA_DSP_VIDEO_ENC_ STREAM_CONFIG_s** for Video Encode API **AmbaDSP_VideoEnc-Config()**.*

9.2.2.2 AmbaDSP_VideoEncConfig > AMBA_DSP_VIDEO_STREAM_ID_s

Type	Field	Description
UINT8	**StreamID**	DSP Stream Type
UINT_8	**ViewZoneID**	Source View Zone ID

*Table 9-6. Definition of **AMBA_DSP_VIDEO_STREAM_ID_s** for Video Encode API **AmbaDSP_VideoEncConfig()**.*

9.2.2.3 AmbaDSP_VideoEncConfig > AMBA_DSP_WINDOW_s

Type	Field	Description
UINT16	**OffsetX**	Horizontal offset of the window
UINT16	**OffsetY**	Vertical offset of the window
UINT16	**Width**	Number of pixels per line in the window
UINT16	**Height**	Number of lines in the window

*Table 9-7. Definition of **AMBA_DSP_WINDOW_s** for Video Encode API **AmbaDSP_VideoEncConfig()**.*

9.2.2.4 AmbaDSP_VideoEncConfig > AMBA_DSP_VIDEO_ENC_CONFIG_u

Type	Field	Description
AMBA_DSP_H264ENC_ CONFIG_s	**H264Config**	H.264 encoder configurations

Type	Field	Description
AMBA_DSP_MJPEG_ENC_CONFIG_s	**MJpegConfig**	MJPEG encoder configurations

*Table 9-8. Definition of **AMBA_DSP_VIDEO_ENC_CONFIG_u** for Video Encode API **AmbaDSP_VideoEncConfig()**.*

9.2.2.5 AmbaDSP_VideoEncConfig > AMBA_DSP_H264ENC_CONFIG_s

Type	Field	Description
UINT8	**ProfileIDC**	H.264 profile
UINT8	**LevelIDC**	H.264 level
UINT8	**IsCabac**	H.264 Entropy coding mode 0: CAVLC 1: CABAC
UINT8	**M**	Number of pictures between reference pictures
UINT8	**N**	Number of pictures between I-pictures
UINT8	**GopStruct**	0: SIMPLE_GOP 1: HIERB_GOP (dyadic hierarchical B prediction structure)
UINT8	**NumPRef**	P-reference picture number 0: Default
UINT8	**NumBRef**	B-reference picture number 0: Default
UINT32	**QualityLevel**	H.264 quality level
AMBA_DSP_BITRATE_CTRL_e	**BitRateSetting**	Please refer to Section 9.2.2.7 for more details.
UINT8	**VbrComplexLevel**	Complexity level. The desired picture quality level
UINT8	**VbrPercent**	0 ~ 99. The percentage of average rate that will be devoted to VBR.
UINT16	**VbrMinRatio**	0 ~ 100. The minimum rate that VBR will not dip below.
UINT16	**VbrMaxRatio**	100 ~ . The maximum rate the VBR will not go above.
UINT32	**IdrInterval**	Indicates the number of GOPs after which an IDR picture should be generated.
UINT8	***pBitsBufAddr**	Base address of H.264 output bitstream FIFO
UINT32	**BitsBufSize**	Size of H.264 output bitstream FIFO (Byte)
UINT32	**BitRate**	Encoding bit rate
UINT8	**HierPGop**	Enable hierarchial P-frame mode
UINT8	***pBitsBufStartAddr**	Assign any address to start within bitstream buffer
AMBA_DSP_H264ENC_QP_CONTROL_s	**QPControl**	QP Config, please refer to Section 9.2.6.3 for details
AMBA_DSP_H264ENC_ROI_CONFIG_s	**RoiControl**	New ROI control to config, please refer to Section 9.2.6.5 for details
AMBA_DSP_H264ENC_HQP_CONFIG_s	**HQPControl**	New HQP control to config, please refer to Section 9.2.6.6 for details
UINT32	**ZmvThreshold**	New Zmv threshold to config

*Table 9-9. Definition of **AMBA_DSP_H264ENC_CONFIG_s** for Video Encode API **AmbaDSP_VideoEncConfig()**.*

9.2.2.6 AmbaDSP_VideoEncConfig > AMBA_DSP_MJPEG_ENC_CONFIG_s

Type	Field	Description
UINT8	***pBitsBufAddr**	Size of MJPEG output bitstream FIFO
UINT32	**BitsBufSize**	Size of MJPEG output bitstream FIFO
UINT8	***pQuantMatrixAddr**	The address of quantization table

*Table 9-10. Definition of **AMBA_DSP_MJPEG_ENC_CONFIG_s** for Video Encode API **AmbaDSP_VideoEncConfig()**.*

9.2.2.7 AmbaDSP_VideoEncConfig > AMBA_DSP_BITRATE_CTRL_e

Type	Description
AMBA_DSP_BITRATE_CBR	Constant bitrate (CBR)
AMBA_DSP_BITRATE_VBR	Variable bitrate (VBR)

*Table 9-11. Definition of **AMBA_DSP_BITRATE_CTRL_e** for Video Encode API **AmbaDSP_VideoEncConfig()**.*

9.2.3 AmbaDSP_VideoEncStreamConfig

API Syntax:

AmbaDSP_VideoEncStreamConfig (int StreamIdx)

Function Description:

- This function is used to reconfigure individual video streams. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**Streamidx**	Assign stream index to reconfigure.

*Table 9-12. Parameters for Video Encode API **AmbaDSP_VideoEncStreamConfig()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 9-13. Returns for Video Encode API **AmbaDSP_VideoEncStreamConfig()**.*

Example:

```
/* Edit global variables which is configured to encoder by AmbaDSP_VideoEncCo-
nfig() */
AmbaEncStreamCfg[StreamIdx].DataFmt                          = 0;
/* H.264 */
AmbaEncStreamCfg[StreamIdx].StreamID.ChannelID.Data = 0;          /* Channel 0
*/
AmbaEncStreamCfg[StreamIdx].StreamID.StreamID          = StreamIdx;     /*
Stream ID */
AmbaEncStreamCfg[StreamIdx].Window.Width                 = 1920;      /* En-
code width */
AmbaEncStreamCfg[StreamIdx].Window.Height                = 1080;      /* En-
code height */
AmbaEncStreamCfg[StreamIdx].FrameRateDivisor             = 1;         /* The
same as vin frame rate */
pEnc = &(AmbaEncStreamCfg[StreamIdx].EncConfig);
pEnc->H264Config.pBitsBufAddr  = ???            /* Base address of bitstream
buffer */
pEnc->H264Config.BitsBufSize     = ???               /* Size of bitstream
buffer */
```

```
pEnc->H264Config.M                    = 1;
pEnc->H264Config.N                    = 8;
pEnc->H264Config.IsCabac           = 0x1;
pEnc->H264Config.QualityLevel    = 0x83;
pEnc->H264Config.BitRate            = 16000000;
pEnc->H264Config.BitRateSetting = 1;             /* CBR */
pEnc->H264Config.IdrInterval        = 4;
memcpy(&(pEnc->H264Config.FrameRate), ???, sizeof(AMBA_DSP_FRAME_RATE_s));

      /* Apply configurations to stream */
AmbaDSP_VideoEncStreamConfig(StreamIdx);
```

See Also:

None

9.2.4 AmbaDSP_VideoEncStart

API Syntax:

AmbaDSP_VideoEncStart (int NumStream, int *pStreamIdx, AMBA_DSP_VIDEO_ENC_START_CONFIG_u *pStartConfig)

Function Description:

- This function is used to initialize the specified encoders. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to start
int	***pStreamIdx**	Array of index in stream configurations table
AMBA_DSP_VIDEO_ENC_START_CONFIG_u	***pStartConfig**	Array of start configuration. Please refer to Section 9.2.4.1 for more details.

*Table 9-14. Parameters for Video Encode API **AmbaDSP_EncVideoStart()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 9-15. Returns forVideo Encode API **AmbaDSP_EncVideoStart()**.*

Example:

```
int
i, StreamIdx[2];
AMBA_DSP_VIDEO_ENC_START_CONFIG_u  StartCfg[2];

memset(StartCfg, 0, sizeof(AMBA_DSP_VIDEO_ENC_START_CONFIG_u) * 2);
for (i = 0; i < 2; i++) {
StreamIdx[i] = i;

if (0 == AmbaEncStreamCfg[i].DataFmt) {   /* H.264 */
StartCfg[i].H264.EnableLoopFilter  = 1;
/* Configure VUI */
StartCfg[i].H264.Vui.VuiEnable       = 0x1;
.....
} else {    /* MJPEG */
StartCfg[i].MJpeg.NumeratorOfFrameReduction   = 0;
StartCfg[i].MJpeg.DenominatorOfFrameReduction = 0;
}
}

/* Start to encode */
AmbaDSP_VideoEncStart(UserCfgNum, StreamIdx, StartCfg);
```

Related Events:

The following events must be registered prior to encoder initialization.

(1) **AMBA_DSP_EVENT_VIDEO_ENC_START**: This event will be triggered when the encoder starts. The user can use the event to perform AV synchronization.
(2) **AMBA_DSP_EVENT_VIDEO_DATA_READY**: This event will be triggered when the encoded frames begin arriving. The pointer of pEventData will include picture-related information including base address and size. The user can refer to the structure of **AMBA_DSP_EVENT_ENC_PIC_READY_s**.

Example:

```
static AMBA_DSP_EVENT_HANDLER_f  AvSyncEvtEntry[1];
static AMBA_DSP_EVENT_HANDLER_f  VideoDataEvtEntry[2];

/* Register event handlers */
AmbaDSP_EventHandlerCtrlConfig(AMBA_DSP_EVENT_VIDEO_ENC_START, 1, AvSyncEvtEntry);
AmbaDSP_RegisterEventHandler(AMBA_DSP_EVENT_VIDEO_ENC_START, AvSyncEventHandler);
AmbaDSP_EventHandlerCtrlConfig(AMBA_DSP_EVENT_VIDEO_DATA_READY, 2, VideoDataEvtEn-
try);
AmbaDSP_RegisterEventHandler(AMBA_DSP_EVENT_VIDEO_DATA_READY, VideoDataEventHan-
dler);

/* AV sync event handler */
static int AvSyncEventHandler(void *pEventData)
{
……….
/* To start audio input task */
if (OK != AmbaAudio_InputOpenEncIoNode(AmbaAudioInputIndex)) {
DBG_MREC("Fail to open encoder I/O node");
}
if (OK != AmbaAudio_InputTaskStart(AmbaAudioInputIndex)) {
            DBG_MREC("Fail to start audio input task");
}
……….
}

/* Video data ready event handler */
static int VideoDataEventHandler(void *pEventData)
{
……….
      /* Send picture information to queue and signal recorder task to mux picture
*/
Flag = AMBA_MOVIE_RECORD_FLG_VIDEO;
if (OK != AmbaKAL_MsgQueueSend(&MovieRecordVideoQueId, pEventData, TimeOut)) {
Flag |= AMBA_MOVIE_RECORD_FLG_VIDEO_QFULL;
DBG_MREC("Video Queue full");
      }
AmbaMovieRecord_SetFlags(Flag);
……….
return OK;
}
```

See Also:

None

9.2.4.1 AmbaDSP_VideoEncStart > AMBA_DSP_VIDEO_ENC_START_CONFIG_u

Type	Field	Description
UINT8	**EnableSlowShutter**	Enables slow shutter mode (H.264)
UINT8	**FirstGOPStartB**	Indicates that the first M-1 frame of the first GOP should be B-pictures (H.264)
UINT8	**SyncWithPIV**	This one-bit flag instructs the encode module to synchronize PIV with the first frame in the encoding sequence.
UINT8	**EnableLoopFilter**	0: Disable 1: Enable the use of alpha and beta values specified below 2: Use internally computed alpha and beta
UINT8	**LoopFilterAlpha**	Used to set alpha for loop filter. (-6 ~ 6)
UINT8	**LoopFitlerBeta**	Used to set beta for loop filter. (-6 ~ 6)
UINT16	**AuType**	0: No Access Unit Delimiters (AUD), No Supplemental Enhancement Information (SEI) 1: AUD before Sequence Parameter Set (SPS), Picture Parameter Set (PPS) with SEI 2: AUD after SPS, PPS with SEI 3: NO AUD with SEI
UINT16	**FrameCroppingFlag**	See H.264 specification
UINT16	**FrameCropLeftOffset**	See H.264 specification
UINT16	**FrameCropRightOffset**	See H.264 specification
UINT16	**FrameCropTopOffset**	See H.264 specification
UINT16	**FrameCropBoBottomOffset**	See H.264 specification
AMBA_DSP_H264ENC_VUI_s	**Vui**	H.264 VUI parameter (H.264)
UINT8	**NumeratorOfFrameReduction**	Numerator of frame-reduction factor (MJPEG)
UINT8	**DenominatorOfFrameReduction**	Denominator of frame-reduction factor (MJPEG)
UINT32	**EncDuration**	0 is continuous. The unit is frame.
AMBA_DSP_H264ENC_QUALITY_MODEL_s	**QModelCtrl**	Quality control to take effect at the Ist frame. Please refer to Section 9.2.6.4 for details.
UINT32	**ResRateMin**	Bitrate distribution parameter for AQP. Range: 0~ 100.

*Table 9-16. Definition of **AMBA_DSP_VIDEO_ENC_START_CONFIG_u** for Video Encode API **AmbaDSP_EncVideoStart()**.*

9.2.4.2 AmbaDSP_VideoEncStart > AMBA_DSP_H264ENC_VUI_s

Type	Field	Description
UINT8	**VuiEnable**	H.264 Video Usability Information (VUI) parameters

Type	Field	Description
UINT8	**AspectRatioInfoPresentFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**OverscanInfoPresentFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**OverscanAppropriateFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**VideoSignalTypePresentFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**VideoFullRangeFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**ColourDescriptionPresentFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**ChromaLocInfoPresentFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**TimingInfoPresentFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**FixedFrameRateFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**NalHrdParametersPresentFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**VclHrdParametersPresentFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**LowDelayHrdFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**PicStructPresentFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**BitstreamRestrictionFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**MotionVectorsOverPicBoundariesFlag**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**AspectRatioIdc**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT16	**SarWidth**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT16	**SarHeight**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**VideoFormat**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**ColourPrimaries**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**TransferCharacteristics**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**MatrixCoefficients**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**ChromaSampleLocTypeTopField**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT8	**ChromaSampleLocTypeBottomField**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
-	-	-
UINT8	**Log2MaxMvLengthHorizontal:**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.

Type	Field	Description
UINT8	**Log2MaxMvLength-Vertical**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT16	**NumReorderFrames**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT16	**MaxDecFrameBuffer-ing**	Please refer to ISO/IEC 14496-10 standard D.1 VUI syntax.
UINT16	**MaxBytesPerPicDe-nom**	See H.264 specifications
UINT16	**MaxBitsPerMbDeom**	See H.264 specifications

*Table 9-17. Definition of **AMBA_DSP_H264ENC_VUI_s** for Video Encode API **AmbaDSP_EncVideoStart()**.*

9.2.5 AmbaDSP_VideoEncStop

API Syntax:

AmbaDSP_VideoEncStop (int NumStream, int *pStreamIdx, AMBA_DSP_H264ENC_STOP_PAUSE_OPTION_e *pOption)

Function Description:

- This function is used to stop the specified encoders. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to stop
int	***pStreamIdx**	Array of index in stream configurations table
AMBA_DSP_H264ENC_STOP_PAUSE_OPTION_e	***pOption**	Array of stop options. Please refer to Section 9.2.5.1 for more details.

*Table 9-18. Parameters for Video Encode API **AmbaDSP_VideoEncStop()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 9-19. Returns for Video Encode API **AmbaDSP_VideoEncStop()**.*

Example:

```
User must wait to EOS after stopping encoders. The picture size of EOS is
0xFFFFFF.

int                                   StreamIdx[1];
AMBA_DSP_H264ENC_STOP_PAUSE_OPTION_e  Options[1];

StreamIdx[i] = 0;
Options[i]   = AMBA_DSP_H264ENC_STOP_PAUSE_IMMEDIATELY;
/* Stop encoders */
AmbaDSP_VideoEncStop(1, StreamIdx, Options);
/* Wait to EOS */
AmbaMovieRecord_WaitFlags(AMBA_MOVIE_RECORD_FLG_VIDEO_EOS, AMBA_KAL_AND_CLEAR,
AMBA_KAL_WAIT_FOREVER);
……
……

static void MovieRecordTaskEntry(AMBA_MOVIE_RECORD_CTRL_s *pMRec)
{
```

```
    ……..
    while (1) {
        ActualFlags = AmbaMovieRecord_WaitFlags(AMBA_MOVIE_RECORD_FLG_ALL,
                                                AMBA_KAL_OR_CLEAR,
                                                AMBA_KAL_WAIT_FOREVER);
……..
        if (ActualFlags & AMBA_MOVIE_RECORD_FLG_VIDEO) {
            for (i = 0; i < VideoCount; i++) {
                if (OK == AmbaKAL_MsgQueueReceive(&MovieRecordVideoQueId,
                                                  &DescInfo,
                                                  AMBA_KAL_NO_WAIT)) {
                    if (0xFFFFFF == DescInfo.PicSize) {
                        if (Stop) {
                            /* Set flag to signal EOS is received */
                            AmbaMovieRecord_SetFlags(AMBA_MOVIE_RECORD_FLG_
VIDEO_EOS);
                        }
                    } else {
                                        /* Mux incoming encoded pictures to file
*/
}
                    }
                }
            }
        }
}
```

See Also:

None

9.2.5.1 AmbaDSP_VideoEncStop > AMBA_DSP_H264ENC_STOP_PAUSE_OPTION_e

Type	Description
AMBA_DSP_H264ENC_STOP_PAUSE_IMMEDIATELY	Stop immediately.
AMBA_DSP_H264ENC_STOP_PAUSE_NEXT_IP	Stop on next P- or I-picture.
AMBA_DSP_H264ENC_STOP_PAUSE_NEXT_I	Stop on I-picture boundary.
AMBA_DSP_H264ENC_STOP_PAUSE_NEXT_IDR	Stop on IDR picture boundary.
AMBA_DSP_H264ENC_STOP_MBSYNC	Stop on MBSYNC.
AMBA_DSP_H264ENC_STOP_EMERG	Stop emergent, just for H264 encoder.

*Table 9-20. Definition of **AMBA_DSP_H264ENC_STOP_PAUSE_OPTION_e** for Video Encode API **AmbaDSP_VideoEncStop()**.*

9.2.6 AmbaDSP_VideoEncRealTimeQualityCtrl

API Syntax:

AmbaDSP_VideoEncRealTimeQualityCtrl (int MaxNumStream, int *pStreamIdx, AMBA_DSP_H264ENC_REALTIME_QUALITY_s *pRealTimeQ)

Function Description:

- This function is used to control realtime visual qualities for H264 encoding.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to control
int	***pStreamIdx**	Array of index in stream configuration table
AMBA_DSP_H264ENC_REALTIME_QUALITY_s	***pRealTimeQ**	Control Structure. Please refer to Section 9.2.6.1 for details

*Table 9-21. Parameters for Video Encode API **AmbaDSP_VideoEncRealTimeQualityCtrl()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 9-22. Returns for Video Encode API **AmbaDSP_VideoEncRealTimeQualityCtrl()**.*

Example:

See Also:

None

9.2.6.1 AmbaDSP_VideoEncRealTimeQualityCtrl > AMBA_DSP_H264ENC_REALTIME_QUALITY_s

Type	Parameter	Description
UINT32	EnableFlag	Enable flag, bit distribution is as below
UINT32:1		Enable bitrate control
UINT32:1		Enable config GOP
UINT32:1		Enable config QP
UINT32:1		Enable config quality model
UINT32:1		Enable config ROI
UINT32:1		Enable config High Quality P-frame
UINT32:1		Enable config zero motion vector threshold
UINT32	BitRate	New bitrate to config

Type	Parameter	Description
AMBA_DSP_H264ENC_GOP_CONFIG_s	GopConfig	New GOP to config, please refer to Section 9.2.6.2 for details
AMBA_DSP_H264ENC_QP_CONTROL_s	QpConfig	New QP control to config, please refer to Section 9.2.6.3 for details
AMBA_DSP_H264ENC_QUALITY_MODEL_s	QualityModel-Config	New quality model control to config, please refer to Section 9.2.6.4 for details
AMBA_DSP_H264ENC_ROI_CONFIG_s	RoiConfig	New ROI control to config, please refer to Section 9.2.6.5 for details
AMBA_DSP_H264ENC_HQP_CONFIG_s	HqpConfig	New HQP control to config, please refer to Section 9.2.6.6 for details
UINT32	ZmvConfig	New Zmv threshold to config

*Table 9-23. Definition of **AMBA_DSP_H264ENC_REALTIME_QUALITY_s** for Video Encode API **AmbaDSP_VideoEncRealTimeQualityCtrl()**.*

9.2.6.2 AmbaDSP_VideoEncRealTimeQualityCtrl > AMBA_DSP_H264ENC_GOP_CONFIG_s

Type	Parameter	Description
UINT8	M	Distance between P frames
UINT8	N	Distance between I frames
UINT8	IDR	Distance between IDR frames
UINT8	GopStruct	0: SIMPLE GOP 1: HIERB_GOP (hierarchial B prediction structure)
UINT8	HierP	Enable Hierarchial P frame mode (M must be 1)

*Table 9-24. Definition of **AMBA_DSP_H264ENC_GOP_CONFIG_s** for Video Encode API **AmbaDSP_VideoEncRealTimeQualityCtrl()**.*

9.2.6.3 AmbaDSP_VideoEncRealTimeQualityCtrl > AMBA_DSP_H264ENC_QP_CONTROL_s

Type	Parameter	Description
UINT8	QpMinIFrame	Minimum QP allowed for I frames
UINT8	QpMaxIFrame	Maximum QP allowed for I frames
UINT8	QpMinPFrame	Minimum QP allowed for P frames
UINT8	QpMaxPFrame	Maximum QP allowed for P frames
UINT8	QpMinBFrame	Minimum QP allowed for B frame
UINT8	QpMaxBFrame	Maximum QP allowed for B frames

*Table 9-25. Definition of **AMBA_DSP_H264ENC_QP_CONTROL_s** for Video Encode API **AmbaDSP_VideoEncRealTimeQualityCtrl()**.*

9.2.6.4 AmbaDSP_VideoEncRealTimeQualityCtrl > AMBA_DSP_H264ENC_QUALITY_MODEL_s

Type	Parameter	Description
UINT8	IorIDRFrameNeedsRateControlMask	Indicates which picture around I or IDR picture needs special RC treatment. Specifically, 0 means no special handling 1 means M-1 pictures after IDR require special quant reduction 2 means the P picture just before IDR requires special quant reduction 4 means M-1 B pictures before IDR requires special quant reduction 8 means the P picture just before I requires special quant reduction
UINT8	QPReduceNearIDRFrame	Quant reduction amount for pictures having special treatment near IDR
UINT8	ClassNumberLimitofQPReduceNearIDRFrame	The class number up to which the quant reduction can apply for pictures near IDR
UINT8	QPReduceNearIFrame	Quant reduction amount for pictures having special treatment near I
UINT8	ClassNumberLimitOfQPReduceNearIFrame	The class number up to which the quant reduction can apply for pictures near I
INT8	Intra16by16Bias	Make DSP tend to select intra16x16 MB. Range: -64 ~ 64
INT8	Intra4by4Bias	Make DSP tend to select intra4x4 MB. Range: -64 ~ 64
INT8	Inter16by16Bias	Make DSP tend to select inter16x16 MB. Range: -64 ~ 64
INT8	Inter8by8bias	Make DSP tend to select inter4x4 MB. Range: -64 ~ 64
INT8	Direct16by16Bias	Make DSP tend to select Direct16x16 MB. Range: -64 ~ 64
INT8	Direct8by8Bias	Make DSP tend to select Direct8x8 MB. Range: -64 ~ 64
INT8	MeLambdaQPOffset	Make DSP tend to use SKIP MB. Range: 0 ~ 51
INT8	AQPStrength	0: Automatic, 1-81: fixed strength; 1 for no AQP; -1 for inverse AQP
INT8	Alpha	Used to set alpha for loop filter (-6 -6)
INT8	Beta	Used to set beta for loop filter (-6 -6)

*Table 9-26. Definition of **AMBA_DSP_H264ENC_QUALITY_MODEL_s** for Video Encode API **AmbaDSP_VideoEncRealTimeQualityCtrl()**.*

9.2.6.5 AmbaDSP_VideoEncRealTimeQualityCtrl > AMBA_DSP_H264ENC_ROI_CONFIG_s

Type	Parameter	Description
UINT32	RoiBuffer	ROI mapping table of every MB
INT8	RoiDelta[3][4]	ROI delta value -51~51, [3]: 0 is I-Frame, 1 is P-Frame, 2 is B-Frame [4]: Four category (0~3) corresponding to roi_mapping table

*Table 9-27. Definition of **AMBA_DSP_H264ENC_ROI_CONFIG_s** for Video Encode API **AmbaDSP_VideoEncRealTimeQualityCtrl()**.*

9.2.6.6 AmbaDSP_VideoEncRealTimeQualityCtrl > AMBA_DSP_H264ENC_HQP_CONFIG_s

Type	Parameter	Description
UINT8	HighPNumber	Number of high quality P pictures (Q picture) in GOP Set it as N, (2 ^N)-1 high quality P pictures in GOP
UINT8	QpMin	Min QP for P picture, 0~51, default is 14
UINT8	QpMax	Max QP for P picture, 0~51, default is 51
UINT8	QpReduce	How much better to make Q QP relative to P QP, 1~10, default is 6

*Table 9-28. Definition of **AMBA_DSP_H264ENC_HQP_CONFIG_s** for Video Encode API **AmbaDSP_VideoEncRealTimeQualityCtrl()**.*

9.2.7 AmbaDSP_VideoEncPause

API Syntax:

AmbaDSP_VideoEncPause (int NumStream, int *pStreamIdx, AMBA_DSP_H264ENC_STOP_PAUSE_OPTION_e *pOption)

Function Description:

- This function is used to pause the encoders. Note that Liveview will continue even after the encoding function has been paused. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to pause
int	***pStreamIdx**	Array of index in stream configurations table
AMBA_DSP_H264ENC_STOP_PAUSE_OPTION_e	***pOption**	Array of pause options. Please refer to Section 9.2.5.1 for more details.

*Table 9-29. Parameters for Video Encode API **AmbaDSP_VideoEncPause()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 9-30. Returns for Video Encode API **AmbaDSP_VideoEncPause()**.*

Example:

```
int
StreamIdx[1];
AMBA_DSP_H264ENC_STOP_PAUSE_OPTION_e  PauseOption[1];

StreamIdx[0]   = 0;
PauseOption[0] = AMBA_DSP_H264ENC_STOP_PAUSE_IMMEDIATELY;
/* Pause encoders */
AmbaDSP_VideoEncPause(1, StreamIdx, PauseOption);
```

See Also:

None

9.2.8 AmbaDSP_VideoEncResume

API Syntax:

AmbaDSP_VideoEncResume (int NumStream, int *pStreamIdx)

Function Description:

- This function is used to resume the functioning of the specified encoders. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to resume
int	***pStreamIdx**	Array of index in stream configurations table

*Table 9-31. Parameters for Video Encode API **AmbaDSP_VideoEncResume()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 9-32. Returns for Video Encode API **AmbaDSP_VideoEncResume()**.*

Example:

```
int  StreamIdx[1];

StreamIdx[0] = 0;
/* Resume encoders */
AmbaDSP_VideoEncResume(1, StreamIdx);
```

See Also:

None

9.2.9 AmbaDSP_VideoEncChangeFrameRate

API Syntax:

AmbaDSP_VideoEncChangeFrameRate (int NumStream, int *pStreamIdx, UINT32 *pDivisor)

Function Description:

- This function is used to configure the encode frame rate and is only valid after encode start. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to change encode frame rate
int	***pStreamIdx**	Array of index in stream configurations table
UINT32	***pDivisor**	Array of encode frame rate divisor

*Table 9-33. Parameters for Video Encode API **AmbaDSP_VideoEncChangeFrameRate()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 9-34. Returns for Video Encode API **AmbaDSP_VideoEncChangeFrameRate()**.*

Example:

```
Assume original encode frame rate is 60p. User can change to 30p by calling as
following.

int          StreamIdx[1];
UINT32  Divisor[1];

StreamIdx[0] = 0;
Divisor[0]      = 2;
/* Change original encode frame rate to "original / Divisor" */
AmbaDSP_VideoEncChangeFrameRate(1, StreamIdx, Divisor);
```

See Also:

None

9.2.10 AmbaDSP_VideoEncChangeBitRate

API Syntax:

AmbaDSP_VideoEncChangeBitRate (int NumStream, int *pStreamIdx, UINT32 *pBitRate)

Function Description:

- This function is used to configure the encode bitrate and is only valid after encode start. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to change encode frame rate
int	***pStreamIdx**	Array of index in stream configurations table
UINT32	***pBitRate**	Array of encode bitrate

*Table 9-35. Parameters for Video Encode API **AmbaDSP_VideoEncChangeBitRate()**.*

Returns:

Return	Description
0	Success
-	Failure

*Table 9-36. Returns for Video Encode API **AmbaDSP_VideoEncChangeBitRate()**.*

Example:

If the user wants to change bitrate to 20Mbps, they can do as follows.

```
int          StreamIdx[1];
UINT32  BitRate[1];

StreamIdx[0] = 0;
BitRate[0]      = 20000000;
/* Change bit rate to 20Mbps */
AmbaDSP_VideoEncChangeBitRate(1, StreamIdx, BitRate);
```

See Also:

None

9.2.11 AmbaDSP_VideoEncIntervalCapCtrl

API Syntax:

AmbaDSP_VideoEncIntervalCapCtrl (UINT8 VinID)

Function Description:

- This function is used to perform interval capture and is only valid after encode start. This is a non-blocking function.
- Note that the **IsIntervalCap** field in the **AMBA_DSP_VIDEO_ENC_STREAM_CONFIG_s** structure must be set prior to encode start. The function will apply to both the main and secondary streams.

Parameters:

Type	Parameter	Description
UINT8	**VinID**	ID of video input (Set it to 0 if single VIN)

*Table 9-37. Parameters for Video Encode API **AmbaDSP_VideoEncIntervalCapCtrl()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 9-38. Returns for Video Encode API **AmbaDSP_VideoEncIntervalCapCtrl()**.*

Example:

To do interval capture every 1 second.

```
/* The   following is  just  pseudo code.  User can use timer  to  do   it */
while (1)  {
/*  Capture one picture */ AmbaDSP_VideoEncIntervalCapCtrl(0);
/* Sleep 1  second */
}
```

See Also:

None

9.2.12 AmbaDSP_VideoEncRepeatDropCtrl

API Syntax:

AmbaDSP_VideoEncRepeatDropCtrl (int NumStream, int *pStreamIdx, AMBA_DSP_VIDEO_REPEAT_DROP_CONFIG_s *pRepeatDropConfig)

Function Description:

- This function is used to repeat or drop encode frames. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to repeat/drop encode frames
int	***pStreamIdx**	Array of index in stream configurations table
AMBA_DSP_VIDEO_REPEAT_DROP_CONFIG_s	***pRepeatDropConfig**	Array of repeat/drop configurations. Please refer to Section 9.2.12.1 below for more details.

*Table 9-39. Parameters for Video Encode API **AmbaDSP_VideoEncRepeatDropCtrl ()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 9-40. Returns for Video Encode API **AmbaDSP_VideoEncRepeatDropCtrl ()**.*

Example:

```
int
StreamIdx[1];
AMBA_DSP_VIDEO_REPEAT_DROP_CONFIG_s  ReptDropCfg[1];

ReptDropCfg.Option          = AMBA_DSP_VIDEO_DROP_AND_REPEAT_FRAME;
ReptDropCfg.FrameCount = 1;
/* To drop and repeat one frame */
AmbaDSP_VideoEncRepeatDropCtrl(1, StreamIdx, & ReptDropCfg)
```

See Also:

None

9.2.12.1 AmbaDSP_VideoEncRepeatDropCtrl > AMBA_DSP_VIDEO_REPEAT_DROP_CONFIG_s

Type	Field	Description
UINT8	**FrameCount**	Frames to repeat/drop
AMBA_DSP_ENC_REPEAT_DROP_OPTION_e	**Option**	Option to repeat/drop encode frames. Please refer to Section 9.2.12.2 for more details.

*Table 9-41. Definition of **AMBA_DSP_VIDEO_REPEAT_DROP_CONFIG_s** for Video Encode API **AmbaDSP_VideoEncRepeatDropCtrl ()**.*

9.2.12.2 AmbaDSP_VideoEncRepeatDropCtrl > AMBA_DSP_ENC_REPEAT_DROP_OPTION_e

Type	Description
AMBA_DSP_VIDEO_DROP_FRAME_ONLY	Drop frame only
AMBA_DSP_VIDEO_DROP_AND_REPEAT_FRAME	Drop the frame currently being captured and replace it with a repeat of a subsequent frame to keep the total video frame rate constant
AMBA_DSP_VIDEO_REPEAT_FRAME	Repeat (one time) the frame currently being captured

*Table 9-42. Definition of **AMBA_DSP_ENC_REPEAT_DROP_OPTION_e** for Video Encode API **AmbaDSP_VideoEncRepeatDropCtrl ()**.*

9.2.13 AmbaDSP_VideoEncSlowShutterCtrl

API Syntax:

AmbaDSP_VideoEncSlowShutterCtrl (int NumStream, int *pStreamIdx, UINT32 *pUpsamplingRate)

Function Description:

- This function is used to maintain a constant encode frame rate when the sensor enters the slow-shutter mode and is only valid after the encode starts. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to repeat/drop encode frames
int	***pStreamIdx**	Array of index in stream configurations table
UINT32	***pUpsamplingRate**	Upsampling rate

*Table 9-43. Parameters for Video Encode API **AmbaDSP_VideoEncSlowShutterCtrl ()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 9-44. Returns for Video Encode API **AmbaDSP_VideoEncSlowShutterCtrl ()**.*

Example:

Assume that the frame rate for the sensor input is 60p. When entering the slow-shutter mode, the frame rate drops to 30p. In this case, the user must set **UpsamplingRate** to 2 in order to maintain the encode frame rate at 60p.

```
int           StreamIdx[1];
UINT32  UpSamplingRate[1];

StreamIdx[0]            = 0;
UpSamplingRate[0] = 2;
AmbaDSP_VideoEncSlowShutterCtrl(1, StreamIdx, UpSampling);
```

See Also:

None

9.2.14 AmbaDSP_VideoEncBlendCtrl

API Syntax:

AmbaDSP_VideoEncBlendCtrl int NumStream, int *pStreamIdx, AMBA_DSP_VIDEO_BLEND_CONFIG_s *pBlendConfig)

Function Description:

- This function is used to configure settings for video encode blending. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to repeat/drop encode frames
int	***pStreamIdx**	Array of index in stream configurations table
AMBA_DSP_VIDEO_BLEND_CONFIG_s	***pBlendConfig**	Configurations of blending. Please refer to Section 9.2.14.1 for more details.

*Table 9-45. Parameters for Video Encode API **AmbaDSP_VideoEncBlendCtrl()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 9-46. Returns for Video Encode API **AmbaDSP_VideoEncBlendCtrl()**.*

Example:

```
/* Prepare buffer for blend area. The format is 420 */
UINT8 BlendY[32 * 40];
UINT8 BlendUV[32 * 40];
UINT8 BlendAlpha[32 * 40];
int StreamIdx[1];
AMBA_DSP_VIDEO_BLEND_CONFIG_s BlendCfg[1];

/* Fill data to buffer and clean data from cache to memory */
memset(&BlendAlpha, 0, sizeof(BlendAlpha));
memset(&BlendUV, 0x80, sizeof(BlendUV));
AmbaCache_Clean(&BlendY, sizeof(BlendAlpha));
AmbaCache_Clean(&BlendUV, sizeof(BlendUV));
AmbaCache_Clean(&BlendAlpha, sizeof(BlendAlpha));
```

```
memset(&BlendCfg, 0, sizeof(BlendCfg));
BlendCfg[0].Enable = 1;
/* Configure Y, UV and alpha buffer */
BlendCfg[0].BlendYuvBuf.pBaseAddrY   = BlendY;
BlendCfg[0].BlendYuvBuf.pBaseAddrUV  = BlendrUV;
BlendCfg[0].BlendYuvBuf.Window.Width = 32;
BlendCfg[0].BlendYuvBuf.Pitch        = ALIGN32(BlendCfg[0].BlendYuvBuf.Window.
Width);
BlendCfg[0].BlendYuvBuf.Window.Height = 40;
BlendCfg[0].AlphaBuf.pBaseAddr = BlendAlpha;
/* Fill id of blend area */
BlendCfg[0].BlendArea = 0;
/* Apply blend configurations */
AmbaDSP_VideoEncBlendCtrl(1, StreamIdx, &BlendCfg[0]);
```

See Also:

None

9.2.14.1 AmbaDSP_VideoEncBlendCtrl > AMBA_DSP_VIDEO_BLEND_CONFIG_s

Type	Field	Description
UINT8	**BlendArea**	0 ~ 31, user-specific blending area
UINT8	**Enable**	0: Disable 1: Enable
AMBA_DSP_YUV_IMG_BUF_s	**BlendYuvBuf**	YUV 420 blending buffer information
AMBA_DSP_BUF_s	**AlphaBuf**	Alpha matrix information

*Table 9-47. Definition of **AMBA_DSP_VIDEO_BLEND_CONFIG_s** for Video Encode API **AmbaDSP_VideoEncBlendCtrl()**.*

9.2.15 AmbaDSP_VideoEncMainStreamConfig

API Syntax:

AmbaDSP_VideoEncMainStreamConfig (int NumStream, iAMBA_DSP_LIVEVIEW_STREAM_CONFIG_s *pStreamConfig)

Function Description:

- This function is used to configure the Main YUV stream size for video encode. This is a non-blocking function. This API must be called before **AmbaDSP_LiveviewConfig()**. This function is used only for multiple-channel configurations.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to repeat/drop encode frames
AMBA_DSP_LIVEVIEW_STREAM_CONFIG_s	***pStreamConfig**	Configurations of Main YUV stream size. Please refer to Section 9.2.15.1 for more details.

*Table 9-48. Parameters for Video Encode API **AmbaDSP_VideoEncMainStreamConfig()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 9-49. Returns for Video Encode API **AmbaDSP_VideoEncMainStreamConfig()**.*

Example:

```
AMBA_DSP_LIVEVIEW_STREAM_CONFIG_s StreamConfig;
       memset(&StreamConfig, 0, sizeof(StreamConfig));
       StreamConfig.StreamID = 0;
       StreamConfig.Width = 2560;
       StreamConfig.Height = 1440;
       AmbaDSP_VideoEncMainStreamConfig(1, &StreamConfig);
```

See Also:

AmbaDSP_LiveviewConfig()

9.2.15.1 AmbaDSP_VideoEncMainStreamConfig > AMBA_DSP_LIVEVIEW_STREAM _CONFIG_s

Type	Field	Description
UINT16	**StreamID**	Stream ID
UINT16	**Width**	Overall stream width
UINT16	**Height**	Overall stream height
int	**NumChan**	Number of channels
AMBA_DSP_LIVEVIEW_ CHANNEL_WINDOW_s	***pChanCfg**	Pointer to channel configuration

*Table 9-50. Definition of **AMBA_DSP_LIVEVIEW_STREAM _CONFIG_s** for Video Encode API **AmbaDSP_VideoEnc-MainStreamConfig()**.*

9.2.16 AmbaDSP_VideoEncSecStreamConfig

API Syntax:

AmbaDSP_VideoEncSecStreamConfig (AMBA_DSP_LIVEVIEW_STREAM_CONFIG_s)

Function Description:

- This function is used to select the source of the secondary YUV stream. This is a non-blocking function. This API must be called before **AmbaDSP_LiveviewConfig()**. This function is only for multiple-channel configurations.

Parameters:

Type	Parameter	Description
AMBA_DSP_LIVEVIEW_STREAM_CONFIG_s	***pStreamConfig**	Configurations of Main YUV stream size. Please refer to Section 9.2.15.1 for more details.

*Table 9-51. Parameters for Video Encode API **AmbaDSP_VideoEncSecStreamConfig()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 9-52. Returns for Video Encode API **AmbaDSP_VideoEncSecStreamConfig()**.*

Example:

None

See Also:

AmbaDSP_LiveviewConfig()

10 Video Decode

10.1 Video Decode: Overview

This chapter details the functions used to configure the H.264 decoder and decompress the H.264 video bit-stream. The following figures illustrate the typical video playback process, including the APIs involved during a specific playback event.

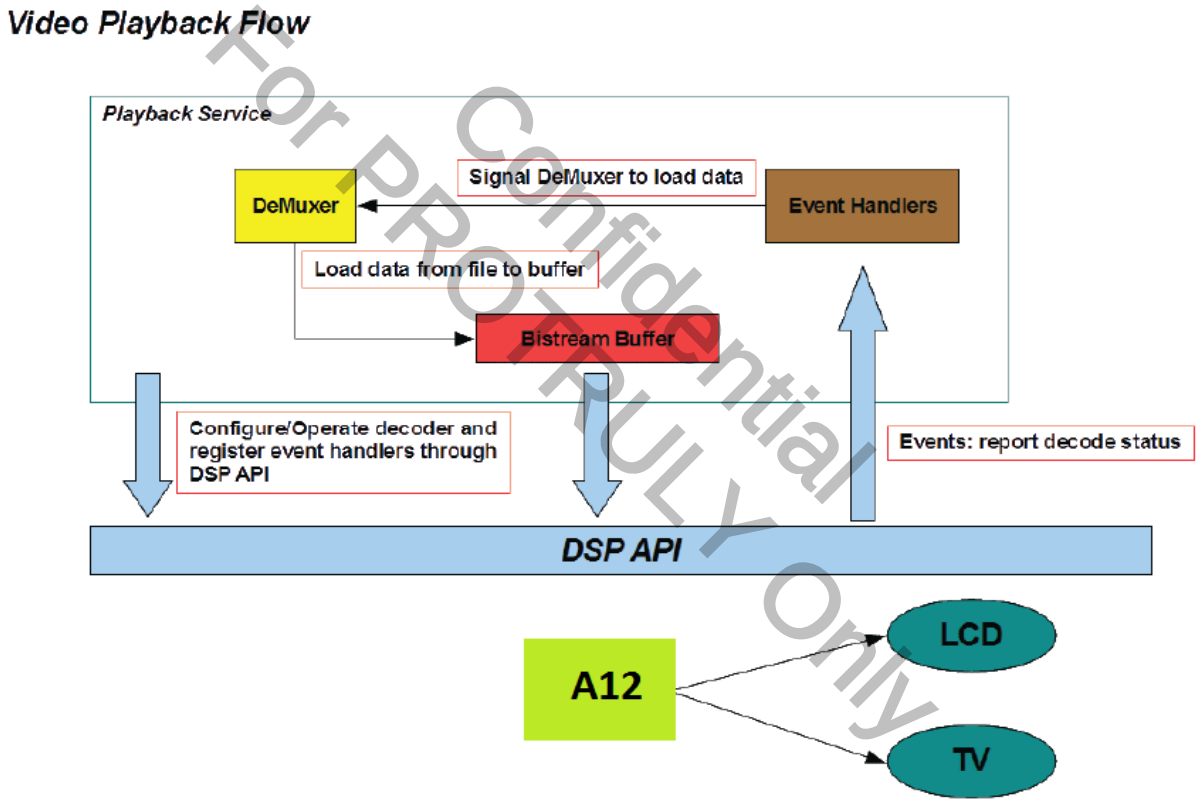

Figure 10-1. Video playback flow.

Video Playback API Use Case

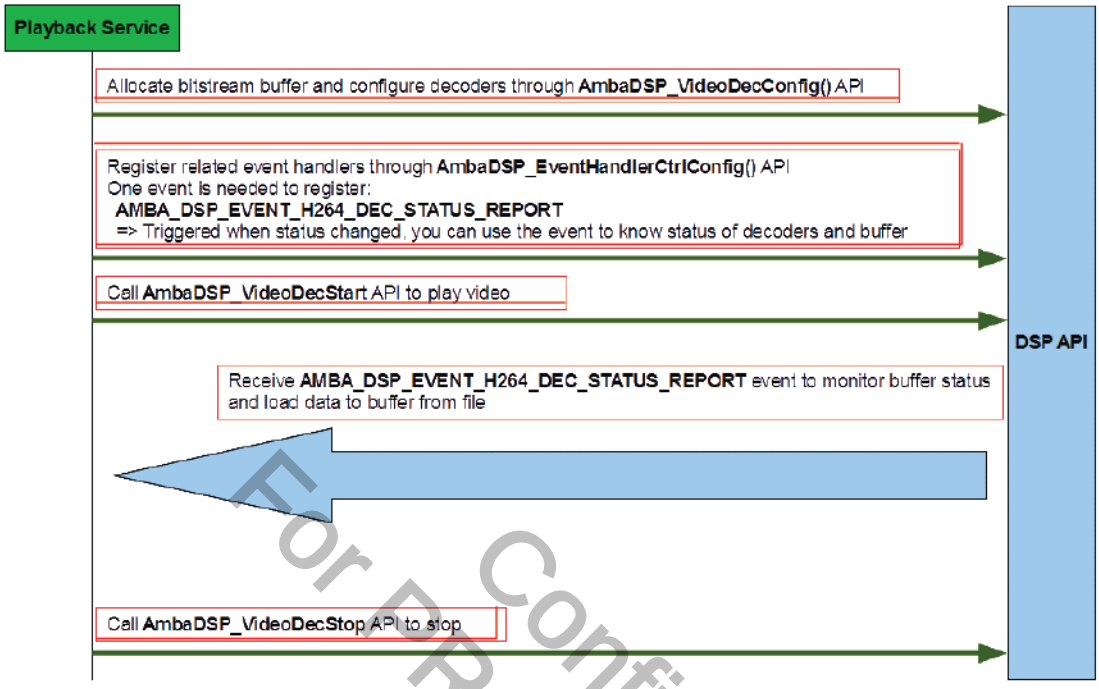

Figure 10-2. Cases of APIs used in video playback.

10.2 Video Decode: List of Functions

- AmbaDSP_VideoDecConfig
- AmbaDSP_VideoDecStart
- AmbaDSP_VideoDecStop
- AmbaDSP_VideoDecSpeedDirSet
- AmbaDSP_VideoDecTrickPlay
- AmbaDSP_VideoDecBitsFifoUpdate
- AmbaDSP_VideoDecPostCtrl
- AmbaDSP_VideoDecFadeEffect
- AmbaDSP_VideoDecVideo2Yuv

10.2.1 AmbaDSP_VideoDecConfig

API Syntax:

AmbaDSP_VideoDecConfig (int MaxNumStream, AMBA_DSP_H264DEC_STREAM_CONFIG_s *pStreamConfig)

Function Description:

- This function is used to configure settings for video stream decoding. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**MaxNumStream**	Maximum video streams to decode
AMBA_DSP_H264DEC_STREAM_CONFIG_s	***pStreamConfig**	Stream configurations table. Please refer to Section 10.2.1.1 for more details.

*Table 10-1. Parameters for Video Decode API **AmbaDSP_VideoDecConfig()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 10-2. Returns for Video Decode API **AmbaDSP_VideoDecConfig()**.*

Example:

```
AMBA_DSP_H264DEC_STREAM_CONFIG_s  DecStreamCfg;

DecStreamCfg.pBitsBufAddr       = ???;/* Base address of bitstream buffer */
DecStreamCfg.BitsBufSize          = ???;/* Size of bitstream buffer */
DecStreamCfg.VideoDecType    = AMBA_DSP_DEC_TYPE_NORMAL;
DecStreamCfg.ErrConcealMode = 1;
DecStreamCfg.MaxFrameWidth  = 4096; /* Set to 4096x2160 to cover 4K clip */
DecStreamCfg.MaxFrameHeight = 2160;

AmbaDSP_VideoDecConfig(1, &DecStreamCfg);
```

See Also:

None

10.2.1.1 AmbaDSP_VideoDecConfig > AMBA_DSP_H264DEC_STREAM_CONFIG_s

Type	Parameter	Description
UINT8	**StreamID**	ID of decode video stream
AMBA_DSP_ DEC_TYPE_e	**VideoDecType**	0: AMBA_DSP_DEC_TYPE_NORMAL 1: AMBA_DSP_DEC_TYPE_LOW_DELAY 2: AMBA_DSP_DEC_TYPE_COMPLIANT
UINT8	**ErrConcealMode**	Skip decode error
UINT8	***pBitsBufAddr**	Base address of decode bitstream FIFO
UINT32	**BitsBufSize**	Size of decode bitstream FIFO (Byte)
UINT16	**MaxFrameWidth**	Maximum frame width
UINT16	**MaxFrameHeight**	Maximum frame height
UINT8	**MaxRatioOfopNM**	Maximum M/N ratio (set 0 to use default value)
UINT8	**IPFrameOnly**	Set to 1 if only I, P frames are in the clip.
UINT8	**BackwardTrickPlay**	If backward playback is required, set the field to 1
The following three fields are only required when performing transcode.		
UINT8	**EnableEncode**	Set this field to enable transcode.
UINT16	**EncodeWidth**	Encode width when performing transcode.
UINT16	**EncodeHeight**	Encode height when performing transcode.
UINT32	**FrameRateDivisor**	Encode Frame rate divisor when performing transcoding.

*Table 10-3. Definition of **AMBA_DSP_H264DEC_STREAM_CONFIG_s** for Video Decode API **AmbaDSP_VideoDecConfig()**.*

10.2.2 AmbaDSP_VideoDecStart

API Syntax:

AmbaDSP_VideoDecStart (int NumStream, int *pStreamIdx, AMBA_DSP_H264DEC_START_CONFIG_s *pStartConfig)

Function Description:

- This function is used to configure settings for the decoder. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to start
int	***pStreamIdx**	Array of index in stream configurations table
AMBA_DSP_H264DEC_START_CONFIG_s	***pStartConfig**	Array of start configurations. Please refer to Section 10.2.2.1 for more details.

*Table 10-4. Parameters for Video Decode API **AmbaDSP_VideoDecStart()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 10-5. Returns for Video Decode API **AmbaDSP_VideoDecStart()**.*

Example:

```
int
StreamIdx[1];
AMBA_DSP_H264DEC_START_CONFIG_s  StartCfg[1];

StreamIdx[0] = 0;
memset(StartCfg, 0, sizeof(StartCfg));
      StartCfg[0].PreloadDataSize = PreloadSize;      /* Size of data is load-
ed to bitstream buffer */
StartCfg[0].FirstDisplayPTS = 0;               /* Set to 0 if playback is
from beginning */

AmbaDSP_VideoDecStart(1, StreamIdx, StartCfg);
```

Related Events:

(1) **AMBA_DSP_EVENT_H264_DEC_STATUS_REPORT**: The decoder will trigger the event to update decode status, including information regarding the buffer, error details, and the decoder. Refer to the structure of **AMBA_DSP_EVENT_H264_DEC_STATUS_UPDATE_*s***.

DecodeState indicates the state of decoder.
ErrorStatus indicates whether decode errors exist.
BitsNextReadAddr indicates the size of the data that has been decoded and calculates the free space available for loading new data.

See Also:

None

10.2.2.1 AmbaDSP_VideoDecStart > AMBA_DSP_H264DEC_START_CONFIG_s

Type	Field	Description
UINT32	**PreloadDataSize**	Size of preload data in decode bitstream FIFO
UINT64	**FirstDisplayPTS**	The presentation timestamp (PTS) from the first frame to display. For forward playback, all frames with PTS values greater than or equal to this value will be displayed. For reverse playback; all frames with PTS values lower than or equal to this value will be displayed.

*Table 10-6. Definition of **AMBA_DSP_H264DEC_START_CONFIG_s** for Video Decode API **AmbaDSP_VideoDecStart()**.*

10.2.3 AmbaDSP_VideoDecStop

API Syntax:

AmbaDSP_VideoDecStop (int NumStream, int *pStreamIdx, AMBA_DSP_H264DEC_STOP_OPTION_e *pStopOption)

Function Description:

- This function is used to stop the decoding of specified video streams. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to stop
int	***pStreamIdx**	Array of index in stream configurations table
AMBA_DSP_H264DEC_STOP_OPTION_e	***pStopOption**	Array of stop options. Please refer to Section 10.2.3.1 for more details.

*Table 10-7. Parameters for Video Decode API **AmbaDSP_VideoDecStop()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 10-8. Returns for Video Decode API **AmbaDSP_VideoDecStop()**.*

Example:

```
int StreamIdx[1];
AMBA_DSP_H264DEC_STOP_OPTION_e  StopOption[1];

StreamIdx[0] = 0;
StopOption[0] = AMBA_DSP_H264DEC_STOP_OPTION_NORMAL;

AmbaDSP_VideoDecStop(1, StreamIdx, StopOption);
```

See Also:

None

10.2.3.1 AmbaDSP_VideoDecStop > AMBA_DSP_H264DEC_STOP_OPTION_e

Type	Description
AMBA_DSP_H264DEC_STOP_OPTION_NORMAL	Stop immediately; no picture is displayed.
AMBA_DSP_H264DEC_STOP_OPTION_KEEP_LAST_DISP	Stop after last picture is displayed; last picture continues displaying.

*Table 10-9. Definition of **AMBA_DSP_H264DEC_STOP_OPTION_e** for Video Decode API **AmbaDSP_VideoDecStop()**.*

10.2.4 AmbaDSP_VideoDecSpeedDirSet

API Syntax:

AmbaDSP_VideoDecSpeedDirSet (int NumStream, int *pStreamIdx, UINT32 *pSpeed, AMBA_DSP_H264DEC_DIR_e *pDir)

Function Description:

- This function is used to configure the speed and direction of the specified decoders. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to configure
int	***pStreamIdx**	Array of index in stream configurations table
UINT16	***pSpeed**	Array of speed configurations. The speed configuration is 8 bits integer and 8 bits fraction. For 1x speed, the value is 0x100.
AMBA_DSP_H264DEC_DIR_e	***pDir**	Array of direction configurations. Please refer to Section 10.2.4.1 for more details.

*Table 10-10. Parameters for Video Decode API **AmbaDSP_VideoDecSpeedDirSet()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 10-11. Returns for Video Decode API **AmbaDSP_VideoDecSpeedDirSet()**.*

Example:

```
int StreamIdx[1];
UINT16 SpeedCfg[1];
AMBA_DSP_H264DEC_DIR_e  DirCfg[1];

StreamIdx[0] = 0;
SpeedCfg[0] = 0x100;          /* Forward 1x */
DirCfg[0] = AMBA_DSP_H264DEC_DIR_FORWARD;

AmbaDSP_VideoDecSpeedDirSet(1, StreamIdx, SpeedCfg, DirCfg);
```

See Also:

None

10.2.4.1 AmbaDSP_VideoDecSpeedDirSet > AMBA_DSP_H264DEC_DIR_e

Type	Description
AMBA_DSP_H264DEC_DIR_FORWARD	Forward playback
AMBA_DSP_H264DEC_DIR_BACKWARD	Backward playback

*Table 10-12. Definition of **AmbaDSP_VideoDecSpeedDirSet** for Video Decode API **AmbaDSP_VideoDecSpeedDirSet()**.*

10.2.5 AmbaDSP_VideoDecTrickPlay

API Syntax:

AmbaDSP_VideoDecTrickPlay (int NumStream, int *pStreamIdx, AMBA_DSP_H264DEC_TRICK_PLAY_e *pTrickPlay)

Function Description:

- This function is used to pause / resume / step the decoding of video streams. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to pause/resume/step
int	***pStreamIdx**	Array of index in stream configurations table
AMBA_DSP_H264DEC_TRICK_PLAY_e	***pTrickPlay**	Array of trickplay options. Please refer to Section 10.2.5.1 for more details.

*Table 10-13. Parameters for Video Decode API **AmbaDSP_VideoDecTrickPlay()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 10-14. Returns for Video Decode API **AmbaDSP_VideoDecTrickPlay()**.*

Example:

```
int StreamIdx[1];
AMBA_DSP_H264DEC_TRICK_PLAY_e  TkCfg[1];

StreamIdx[0] = 0;       /* Pause to decode */
TkCfg[0] = AMBA_DSP_H264DEC_TRICK_PLAY_PAUSE;

AmbaDSP_VideoDecTrickPlay(1, StreamIdx, TkCfg);
```

See Also:

None

10.2.5.1 AmbaDSP_VideoDecTrickPlay > AMBA_DSP_H264DEC_TRICK_PLAY_e

Type	Description
AMBA_DSP_H264DEC_TRICK_PLAY_PAUSE	Pause playback
AMBA_DSP_H264DEC_TRICK_PLAY_RESUME	Resume playback
AMBA_DSP_H264DEC_TRICK_PLAY_STEP	Step playback

*Table 10-15. Definition of **AMBA_DSP_H264DEC_TRICK_PLAY_e** for Video Decode API **AmbaDSP_VideoDecTrick-Play()**.*

10.2.6 AmbaDSP_VideoDecBitsFifoUpdate

API Syntax:

AmbaDSP_VideoDecBitsFifoUpdate (int NumStream, int *pStreamIdx, AMBA_DSP_H264DEC_BITS_FIFO_s *pBitsFIFO)

Function Description:

- This function is used to update the bitstream buffer. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**NumStream**	Number of streams to update bitstream
int	***pStreamIdx**	Array of index in stream configurations table
AMBA_DSP_H264DEC_BITS_FIFO_s	***pBitsFIFO**	Array of update descriptions. Please refer to Section 10.2.6.1 for more details.

*Table 10-16. Parameters for Video Decode API **AmbaDSP_VideoDecBitsFifoUpdate()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 10-17. Returns for Video Decode API **AmbaDSP_VideoDecBitsFifoUpdate()**.*

Example:

```
int StreamIdx[1];
AMBA_DSP_H264DEC_BITS_FIFO_s  BufDesc;

/* load data to buffer and update to decoder */
StreamIdx[0] = 0;
BufDesc.pStartAddr = ???;     /* Fill start address of buffer */
BufDesc.pEndAddr  = ???;      /* Fill end address of buffer */

AmbaDSP_VideoDecBitsFifoUpdate(1, StreamIdx, & BufDesc);
```

See Also:

None

10.2.6.1 AmbaDSP_VideoDecBitsFifoUpdate > AMBA_DSP_H264DEC_BITS_FIFO_s

Type	Field	Description
UINT8	***pStartAddr**	Start address of updated buffer
UINT8	***pEndAddr**	End address of updated buffer

*Table 10-18. Definition of **AMBA_DSP_H264DEC_BITS_FIFO_s** for Video Decode API **AmbaDSP_VideoDecBitsFifoUpdate()**.*

10.2.7 AmbaDSP_VideoDecPostCtrl

API Syntax:

AmbaDSP_VideoDecPostCtrl (int StreamIdx, int NumPostCtrl, AMBA_DSP_DEC_POST_CTRL_s *pPost-Ctrl)

Function Description:

- This function is used to configure post-control settings when decoding, including zoom in/out, flip, and rotate. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**StreamIdx**	Stream index in stream configurations table
int	**NumPostCtrl**	Number of post-control configurations
AMBA_DSP_ DEC_POST_ CTRL_s	***pPostCtrl**	Array of post-control configurations. Please refer to Section 10.2.7.1 for more details.

*Table 10-19. Parameters for Video Decode API **AmbaDSP_VideoDecPostCtrl()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 10-20. Returns for Video Decode API **AmbaDSP_VideoDecPostCtrl()**.*

Example

```
Int StreamIdx[1];
AMBA_DSP_DEC_POST_CTRL_s PPCtrlConfig[1];

StreamIdx[0] = 0;
memset(PPCtrlConfig, 0,  sizeof(PPCtrlConfig));
PPCtrlConfig[0].InputWindow.Off- setX = 0;
PPCtrlConfig[0].InputWindow.OffsetY   = 0;
PPCtrlConfig[0].InputWindow.Width     = 1920;
PPCtrlConfig[0].InputWindow.Height    = 1080;
PPCtrlConfig[0].TargetWindow.OffsetX  = 0;
PPCtrlConfig[0].TargetWindow.OffsetY  = 0;
PPCtrlConfig[0].TargetWindow.Width    = AMBA_LCD_WIDTH;
PPCtrlConfig[0].TargetWindow.Height   = AMBA_LCD_HEIGHT;
PPCtrlConfig[0].OutputSelect          = AMBA_DSP_VOUT_LCD;

PPCtrlConfig[0].RotateFlip            =  AMBA_DSP_ROTATE_0;

AmbaDSP_VideoDecPostCtrl(StreamIdx, 1,  PPCtrlConfig);
```

See Also:

None

10.2.7.1 AmbaDSP_VideoDecPostCtrl > AMBA_DSP_DEC_POST_CTRL_s

Type	Field	Description
AMBA_DSP_WINDOW_s	**InputWindow**	Cropping window of decode YUV
AMBA_DSP_WINDOW_s	**TargetWindow**	Target display size
AMBA_DSP_VOUT_IDX_e	**OutputSelect**	Select LCD or TV VOUT.
AMBA_DSP_ROTATE_FLIP_e	**RotateFlip**	Please refer to Section 4.2.8.1 for more details.
AMBA_DSP_WINDOW_s	**VoutWindow**	VOUT video window

*Table 10-21. Definition of **AMBA_DSP_DEC_POST_CTRL_s** for Video Decode API **AmbaDSP_VideoDecPostCtrl()**.*

10.2.8 AmbaDSP_VideoDecFadeEffect

API Syntax:

AmbaDSP_VideoDecFadeEffect (int StreamIdx, int NumEffect, AMBA_DSP_H264DEC_FADE_EFFECT_s *pFadeEffect)

Function Description:

- This function is used to configure settings for fade-in / fade-out effects. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**StreamIdx**	Stream index in stream configurations table
int	**NumEffect**	Number of fading effect. The maximum is 2.
AMBA_DSP_H264DEC_FADE_EFFECT_s	***pFadeEffect**	Configuration of fading effect. Please refer to Section 10.2.8.1 below for more details.

*Table 10-22. Parameters for Video Decode API **AmbaDSP_VideoDecFadeEffect()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 10-23. Returns for Video Decode API **AmbaDSP_VideoDecFadeEffect()**.*

Example

```
#define FADE_DURATION   3000 /* ms */
UINT32 LastPTS = 720480;   /* This is an example. The last PTS of the video */
AMBA_DSP_H264DEC_FADE_EFFECT_s  FadeEffect[2];

/* Fade in from black to normal */
FadeEffect[0].StartPTS       = 0;
FadeEffect[0].Duration       = FADE_DURATION;
FadeEffect[0].StartMatrix[0] = 0;
FadeEffect[0].StartMatrix[1] = 0;
FadeEffect[0].StartMatrix[2] = 0;
FadeEffect[0].StartMatrix[3] = 0;
FadeEffect[0].StartMatrix[4] = 0;
FadeEffect[0].StartMatrix[5] = 0;
FadeEffect[0].StartMatrix[6] = 0;
FadeEffect[0].StartMatrix[7] = 0;
FadeEffect[0].StartMatrix[8] = 0;
FadeEffect[0].StartYOffset   = 0;
FadeEffect[0].StartUOffset   = 128;
```

```
FadeEffect[0].StartVOffset  = 128;


FadeEffect[0].EndMatrix[0]  = 1024;
FadeEffect[0].EndMatrix[1]  = 0;
FadeEffect[0].EndMatrix[2]  = 0;
FadeEffect[0].EndMatrix[3]  = 0;
FadeEffect[0].EndMatrix[4]  = 1024;
FadeEffect[0].EndMatrix[5]  = 0;
FadeEffect[0].EndMatrix[6]  = 0;
FadeEffect[0].EndMatrix[7]  = 0;
FadeEffect[0].EndMatrix[8]  = 1024;
FadeEffect[0].EndYOffset    = 0;
FadeEffect[0].EndUOffset    = 128;
FadeEffect[0].EndVOffset    = 128;

/* Fade out from normal to black */
FadeEffect[1].StartPTS       = LastPTS - 90K * (FADE_DURATION / 1000) + 1;
FadeEffect[1].Duration       = FADE_DURATION;
FadeEffect[1].StartMatrix[0] = 1024;
FadeEffect[1].StartMatrix[1] = 0;
FadeEffect[1].StartMatrix[2] = 0;
FadeEffect[1].StartMatrix[3] = 0;
FadeEffect[1].StartMatrix[4] = 1024;
FadeEffect[1].StartMatrix[5] = 0;
FadeEffect[1].StartMatrix[6] = 0;
FadeEffect[1].StartMatrix[7] = 0;
FadeEffect[1].StartMatrix[8] = 1024;
FadeEffect[1].StartYOffset   = 0;
FadeEffect[1].StartUOffset   = 128;
FadeEffect[1].StartVOffset   = 128;
FadeEffect[1].EndMatrix[0]  = 0;
FadeEffect[1].EndMatrix[1]  = 0;
FadeEffect[1].EndMatrix[2]  = 0;
FadeEffect[1].EndMatrix[3]  = 0;
FadeEffect[1].EndMatrix[4]  = 0;
FadeEffect[1].EndMatrix[5]  = 0;
FadeEffect[1].EndMatrix[6]  = 0;
FadeEffect[1].EndMatrix[7]  = 0;
FadeEffect[1].EndMatrix[8]  = 0;
FadeEffect[1].EndYOffset    = 0;
FadeEffect[1].EndUOffset    = 128;
FadeEffect[1].EndVOffset    = 128;

AmbaDSP_VideoDecFadeEffect(0, 2, &FadeEffect[0]);
…
AmbaDSP_VideoDecStart(…);
```


See Also:

None

10.2.8.1 AmbaDSP_VideoDecFadeEffect > AMBA_DSP_H264DEC_FADE_EFFECT_s

Type	Field	Description
UINT32	**StartPTS**	Starting presentation time stamp (PTS) of effect.
UINT32	**Duration**	Duration of effect in milliseconds.
UINT16	**StartMatrix[9]**	Starting YUV-to-YUV matrix (in row-dominant order). bits [15:13] = Do not care bits [12:0] = sign plus 2.10 bits; 1 sign bit, 2 integer bits and 10 fractional bits.
INT16	**StartYOffset**	bits [15:11] = Do not care bits [10:0] = sign plus 10 bits
INT16	**StartUOffset**	bits [15:11] = Do not care bits [8:0] = sign plus 8 bits
INT16	**StartVOffset**	bits [15:11] = Do not care bits [8:0] = sign plus 8 bits
UINT16	**EndMatrix[9]**	End YUV-to-YUV matrix (in row-dominant order). bits [15:13] = Do not care bits [12:0] = sign plus 2.10 bits; 1 sign bit, 2 integer bits and 10 fractional bits.
INT16	**EndYOffset**	bits [15:11] = Do not care bits [10:0] = sign plus 10 bits
INT16	**EndUOffset**	bits [15:11] = Do not care bits [8:0] = sign plus 8 bits
INT16	**EndVOffset**	bits [15:11] = Do not care bits [8:0] = sign plus 8 bits

*Table 10-24. Definition of **AMBA_DSP_H264DEC_FADE_EFFECT_s** for Video Decode API **AmbaDSP_VideoDecFadeEffect()**.*

10.2.9 AmbaDSP_VideoDecVideo2Yuv

API Syntax:

AmbaDSP_VideoDecVideo2Yuv (int StreamIdx, AMBA_DSP_YUV_IMG_BUF_s *pYuvBufAddr)

Function Description:

- This function is used to convert the current video picture to YUV data. The user can subsequently use the YUV data to implement photo capture in video playback. This is a non-blocking function.

Parameters:

Type	Parameter	Description
int	**StreamIdx**	Stream index in stream configurations table
AMBA_DSP_YUV_IMG_BUF_s	***pYuvBufAddr**	YUV buffer in which to place picture after playback capture. Please refer to Section 6.2.1.5 for more details.

*Table 10-25. Parameters for Video Decode API **AmbaDSP_VideoDecVideo2Yuv()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 10-26. Returns for Video Decode API **AmbaDSP_VideoDecVideo2Yuv()**.*

Example

```
AMBA_DSP_YUV_IMG_BUF_s VidYuv;

/* Must pause decoder before calling this function */
VidYuv.DataFmt        = AMBA_DSP_YUV422;
VidYuv.pBaseAddrY     = ???;        /* Base address of Y buffer */
VidYuv.pBaseAddrUV    = ???;        /* Base address of UV buffer */
VidYuv.Window.OffsetX = 0;
VidYuv.Window.OffsetY = 0;
VidYuv.Window.Width   = 1920;
VidYuv.Window.Height  = 1080;
VidYuv.Pitch          = 1920;

AmbaDSP_VideoDecVideo2Yuv(0, &VidYuv);
```

See Also:

None

11 Warp Core

11.1 Warp Core: Overview

This chapter details the functions used to configure the A12 video warp.

11.2 Warp Core: List of Functions

11.2.1 AmbaDSP_WarpCore_Init

API Syntax:

AmbaDSP_WarpCore_Init (void)

Function Description:

- This function is used to initialize the warp module.

Parameters:

None

Returns:

Return	Description
0	Success
-1	Failure

*Table 11-1. Returns for Warp Core API **AmbaDSP_WarpCore_Init()**.*

Example:

```
/* Dzoom: 2x */
      AMBA_DSP_IMG_MODE_CFG_s              ImgMode;
      AMBA_DSP_IMG_VIN_SENSOR_GEOMRY_s     VinGeo;
      IMG_SENSOR_FRAME_TIMING_s            ImgFrameTiming;
      IMG_OUT_WIN_INFO_s                   ImgOutputWin;
      IMG_DZOOM_INFO_s                     ImgDzoom;
      double ZoomRatio;
      AMBA_DSP_IMG_CALIB_WARP_INFO_s       ImgCalibWarp;
      AMBA_DSP_IMG_CALIB_CAWARP_INFO_s     ImgCalibCawarp;

      memset(&ImgMode, 0 ,sizeof(AMBA_DSP_IMG_MODE_CFG_s));
      ImgMode.Pipe            = AMBA_DSP_IMG_PIPE_VIDEO;
      ImgMode.BatchId   = AMBA_DSP_VIDEO_FILTER;//AMBA_DSP_VIDEO_FILTER: 0

      memset(&VinGeo, 0, sizeof(AMBA_DSP_IMG_VIN_SENSOR_GEOMRY_s));
      VinGeo.StartX                 = 44;
      VinGeo.StartY                 = 14;
      VinGeo.Width                  = 1312;
      VinGeo.Height                 = 984;
      VinGeo.HSubSample.FactorNum   = 3;
      VinGoo.HSubSample.FactorDen   = 3;
      VinGeo.VSubSample.FactorNum   = 3;
      VinGeo.VSubSample.FactorDen   = 3;
      AmbaDSP_WarpCore_SetVinSensorGeo(ImgMode, &VinGeo);
```

```
        memset(&ImgFrameTiming, 0, sizeof(IMG_SENSOR_FRAME_TIMING_s));
        ImgFrameTiming.TimeScale      = 30000;
        ImgFrameTiming.NumUnitsInTick = 1001;
        ImgFrameTiming.FrameLengthLines     = 2080;
        AmbaDSP_WarpCore_SetSensorFrameTiming(ImgMode, &ImgFrameTiming);

        memset(&ImgOutputWin, 0, sizeof(IMG_OUT_WIN_INFO_s));
        ImgOutputWin.Width            = 960;

    ImgOutputWin.Height               = 480;
        AmbaDSP_WarpCore_SetOutputWin(ImgMode, &ImgOutputWin);
        ……
        ZoomRatio = 2.0 / 100;
        memset(&ImgDzoom, 0, sizeof(IMG_DZOOM_INFO_s));
        ImgDzoom.ZoomX    = (UINT32)(ZoomRatio * 65536);
        ImgDzoom.ZoomY    = (UINT32)(ZoomRatio * 65536);
             AmbaDSP_WarpCore_SetDzoomFactor(ImgMode, &ImgDzoom);

        ImgCalibWarp.Version          = 0x20130101;
        ImgCalibWarp.HorGridNum = 22;
        ImgCalibWarp.VerGridNum = 17;
        ImgCalibWarp.TileWidthExp     = 6;
        ImgCalibWarp.TileHeightExp    = 6;
        ImgCalibWarp.VinSensorGeo     = VinGeo;
        ImgCalibWarp.pWarp            = pCalibWarpData;  // A table with 22*17
    calibration grids.
        AmbaDSP_WarpCore_SetCalibWarpInfo(ImgMode, &ImgCalibWarp);

        ImgCalibCawarp.Version  = 0x20130125;
        ImgCalibCawarp.HorGridNum     = 22;
        ImgCalibCawarp.VerGridNum     = 17;
        ImgCalibCawarp.TileWidthExp   = 6;
        ImgCalibCawarp.TileHeightExp  = 6;
        ImgCalibCawarp.VinSensorGeo   = VinGeo;
        ImgCalibCawarp.RedScaleFactor = 256;
        ImgCalibCawarp.BlueScaleFactor      = 256;
        ImgCalibCawarp.pWarp = pCalibCawarpData;  // A table with 22*17 calibra-
    tion grids.
        AmbaDSP_WarpCore_SetCalibCawarpInfo(ImgMode, &ImgCalibCawarp);

        AmbaDSP_WarpCore_CalcDspWarp(ImgMode, 0);
        AmbaDSP_WarpCore_CalcDspCawarp(ImgMode, 0);
        AmbaDSP_WarpCore_SetDspWarp(ImgMode);
        AmbaDSP_WarpCore_SetDspCawarp(ImgMode);
```

See Also:

None

11.2.2 AmbaDSP_WarpCore_SetVinSensorGeo

API Syntax:

AmbaDSP_WarpCore_SetVinSensorGeo (AMBA_DSP_IMG_MODE_CFG_s Mode, AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s *pVinSensorGeo)

Function Description:

- This function is used to set the VIN geometry.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	**Mode**	Mode configure. Please refer to Section 11.2.2.4 for more details.
AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s	***pVinSensorGeo**	The pointer of sensor VIN geometry. Please refer to Section 11.2.2.6 for more details.

*Table 11-2. Parameters for Warp Core API **AmbaDSP_WarpCore_SetVinSensorGeo()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 11-3. Returns for Warp Core API **AmbaDSP_WarpCore_SetVinSensorGeo()**.*

Example:

Please refer to the example of Section 11.2.1 “AmbaDSP_WarpCore_Init”.

See Also:

AmbaDSP_WarpCore_Init()

11.2.2.1 AmbaDSP_WarpCore_SetVinSensorGeo > AMBA_DSP_IMG_PIPE_e

Type	Description
AMBA_DSP_IMG_PIPE_VIDEO	Video pipe
AMBA_DSP_IMG_PIPE_STILL	Still pipe
AMBA_DSP_IMG_PIPE_DEC	Decode pipe

*Table 11-4. Definition of **AMBA_DSP_IMG_PIPE_e** for Warp Core API **AmbaDSP_WarpCore_SetVinSensorGeo()**.*

11.2.2.2 AmbaDSP_WarpCore_SetVinSensorGeo > AMBA_DSP_IMG_ALGO_MODE_e

Type	Description
AMBA_DSP_IMG_ALGO_MODE_FAST	The fast mode
AMBA_DSP_IMG_ALGO_MODE_LISO	The LISO mode
AMBA_DSP_IMG_ALGO_MODE_HISO	The HISO mode

*Table 11-5. Definition of **AMBA_DSP_IMG_ALGO_MODE_e** for Warp Core API **AmbaDSP_WarpCore_SetVinSensorGeo()**.*

11.2.2.3 AmbaDSP_WarpCore_SetVinSensorGeo > AMBA_DSP_IMG_FUNC_MODE_e

Type	Description
AMBA_DSP_IMG_FUNC_MODE_FV	Full view
AMBA_DSP_IMG_FUNC_MODE_QV	Quick view
AMBA_DSP_IMG_FUNC_MODE_PIV	PIV

*Table 11-6. Definition of **AMBA_DSP_IMG_FUNC_MODE_e** for Warp Core API **AmbaDSP_WarpCore_SetVinSensorGeo()**.*

11.2.2.4 AmbaDSP_WarpCore_SetVinSensorGeo > AMBA_DSP_IMG_MODE_CFG_s

Type	Field	Description
AMBA_DSP_IMG_PIPE_e	**Pipe**	Image Kernel Pipe. Please refer to Section 11.2.2.1 for more details.
AMBA_DSP_IMG_ALGO_MODE_e	**AlgoMode**	Image Kernel algo mode. Please refer to Section 11.2.2.2 for more details.
AMBA_DSP_IMG_FUNC_MODE_e	**FuncMode**	Image Kernel function mode. Please refer to Section 11.2.2.3 for more details.
UINT8	**ContextId**	Image Kernel Context ID
UINT8	**BatchId**	The batch buffer ID. Each batch buffer will be packaged by a different batch command.
UINT8	**ConfigId**	Image Kerenl configures ID.

*Table 11-7. Definition of **AMBA_DSP_IMG_MODE_CFG_s** for Warp Core API **AmbaDSP_WarpCore_SetVinSensorGeo()**.*

11.2.2.5 AmbaDSP_WarpCore_SetVinSensorGeo > AMBA_DSP_IMG_SENSOR_SUBSAMPLING_s

Type	Field	Description
UINT8	**FactorNum**	The sub sampling factor of numerator
UINT8	**FactorDen**	The sub sampling factor of denominator

*Table 11-8. Definition of **AMBA_DSP_IMG_SENSOR_SUBSAMPLING_s** for Warp Core API **AmbaDSP_WarpCore_SetVinSensorGeo()**.*

11.2.2.6 AmbaDSP_WarpCore_SetVinSensorGeo > AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s

Type	Field	Description
UINT32	**StartX**	The start of x-axis. Before down sample.
UINT32	**StartY**	The start of y-axis. Before down sample.
UINT32	**Width**	Width. After down sample.
UINT32	**Height**	Height. After down sample.
AMBA_DSP_IMG_SENSOR_SUBSAMPLING_s	**HSubSample**	The sub sample configure of Horizontal
AMBA_DSP_IMG_SENSOR_SUBSAMPLING_s	**VSubSample**	The sub sample configure of Vertical

*Table 11-9. Definition of **AMBA_DSP_IMG_VIN_SENSOR_GEOMETRY_s** for Warp Core API **AmbaDSP_WarpCore_SetVinSensorGeo()**.*

11.2.3 AmbaDSP_WarpCore_SetOutputWin

API Syntax:

AmbaDSP_WarpCore_SetOutputWin (AMBA_DSP_IMG_MODE_CFG_s Mode, IMG_OUT_WIN_INFO_s *pOutWinInfo)

Function Description:

- This function is used to set the output window for warp result.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	**Mode**	Mode configure. Please refer to Section 11.2.2.4 for more details.
IMG_OUT_WIN_INFO_s	***pOutWinInfo**	The pointer of output window information. Please refer to Section 11.2.3.2 for more details.

*Table 11-10. Parameters for Warp Core API **AmbaDSP_WarpCore_SetOutputWin()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 11-11. Returns for Warp Core API **AmbaDSP_WarpCore_SetOutputWin()**.*

Example:

None

See Also:

None

11.2.3.1 AmbaDSP_WarpCore_SetOutputWin > AMBA_DSP_IMG_WIN_DIMENSION_s

Type	Field	Description
UINT32	**Width**	The window width
UINT32	**Height**	The window height

*Table 11-12. Definition of **AMBA_DSP_IMG_WIN_DIMENSION_s** for Warp Core API **AmbaDSP_WarpCore_SetOutputWin()**.*

11.2.3.2 AmbaDSP_WarpCore_SetOutputWin > IMG_OUT_WIN_INFO_s

Type	Field	Description
AMBA_DSP_IMG_WIN_DIMENSION_s	**MainWinDim**	The main window dimension. Please refer to Section 11.2.3.1 for more details.
AMBA_DSP_IMG_WIN_DIMENSION_s	**PrevWinDim**	The previous window dimenstion. Please refer to Section 11.2.3.1 for more details.
AMBA_DSP_IMG_WIN_DIMENSION_s	**ScreennalDim**	The screennail window dimenstion. Please refer to Section 11.2.3.1 for more details.
AMBA_DSP_IMG_WIN_DIMENSION_s	**ThumbnailDim**	The thumbnail window dimenstion. Please refer to Section 11.2.3.1 for more details.

*Table 11-13. Definition of **IMG_OUT_WIN_INFO_s** for Warp Core API **AmbaDSP_WarpCore_SetOutputWin()**.*

11.2.4 AmbaDSP_WarpCore_SetDzoomFactor

API Syntax:

AmbaDSP_WarpCore_SetDzoomFactor (AMBA_DSP_IMG_MODE_CFG_s Mode, IMG_DZOOM_INFO_s *pDzoomInfo)

Function Description:

- This function is used to set the zoom factor for digital zoom.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	**Mode**	Mode configure. Please refer to Section 11.2.2.4 for more details.
IMG_DZOOM_INFO_s	***pDzoomInfo**	The pointer of zoom factor information. Please refer to Section 11.2.4.1 for more details.

*Table 11-14. Parameters for Warp Core API **AmbaDSP_WarpCore_SetDzoomFactor()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 11-15. Returns for Warp Core API **AmbaDSP_WarpCore_SetDzoomFactor()**.*

Example:

Please refer to the example of (Section 11.2.1) AmbaDSP_WarpCore_Init.

See Also:

None

11.2.4.1 AmbaDSP_WarpCore_SetDzoomFactor > IMG_DZOOM_INFO_s

Type	Field	Description
UINT32	**ZoomX**	The zoom factor of x-axis. It is 16.16 format.
UINT32	**ZoomY**	The zoom factor of y-axis. It is 16.16 format.
Int	**ShiftX**	The shift of x-axis for VIN geometry domain.
int	**ShiftY**	The shift of y-axis for VIN geometry domain.

*Table 11-16. Definition of **IMG_DZOOM_INFO_s** for Warp Core API **AmbaDSP_WarpCore_SetDzoomFactor()**.*

11.2.5 AmbaDSP_WarpCore_SetSensorFrameTiming

API Syntax:

AmbaDSP_WarpCore_SetSensorFrameTiming (AMBA_DSP_IMG_MODE_CFG_s Mode, IMG_SENSOR_FRAME_TIMING_s *pSsrFrmTiming)

Function Description:

- This function is used to set the frame rate for warp.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	**Mode**	Mode configure. Please refer to Section 11.2.2.4 for more details.
IMG_SENSOR_FRAME_TIMING_s	***pSsrFrmTiming**	The pointer of frame rate information. Please refer to Section 11.2.5.1 for more details.

*Table 11-17. Parameters for Warp Core API **AmbaDSP_WarpCore_SetSensorFrameTiming()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 11-18. Returns for Warp Core API **AmbaDSP_WarpCore_SetSensorFrameTiming()**.*

Example:

Please refer to the example of (Section 11.2.1) AmbaDSP_WarpCore_Init.

See Also:

None

11.2.5.1 AmbaDSP_WarpCore_SetSensorFrameTiming > IMG_SENSOR_FRAME_TIMING_s

Type	Field	Description
UINT32	**TimeScale**	The time scale of sensor frame rate
UINT32	**NumUnitsInTick**	The number of units per tick
UINT32	**FrameLengthLines**	The lines of sensor

*Table 11-19. Definition of **IMG_SENSOR_FRAME_TIMING_s** for Warp Core API **AmbaDSP_WarpCore_SetSensorFrameTiming()**.*

11.2.6 AmbaDSP_WarpCore_CalcDspWarp

API Syntax:

AmbaDSP_WarpCore_CalcDspWarp (AMBA_DSP_IMG_MODE_CFG_s Mode, UINT32 Config)

Function Description:

- This function is used to calculate the warp parameters.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	**Mode**	Mode configure. Please refer to Section 11.2.2.4 for more details.
UINT32	**Config**	Configure warp module. **IMG_WARP_CONFIG_FORCE_DISABLE**: Not calculate warp parameters (digital zoom only).

*Table 11-20. Parameters for Warp Core API **AmbaDSP_WarpCore_CalcDspWarp()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 11-21. Returns for Warp Core API **AmbaDSP_WarpCore_CalcDspWarp()**.*

Example:

Please refer to the example of (Section 11.2.1) AmbaDSP_WarpCore_Init.

See Also:

None

11.2.7 AmbaDSP_WarpCore_CalcDspCawarp

API Syntax:

AmbaDSP_WarpCore_CalcDspCawarp (AMBA_DSP_IMG_MODE_CFG_s Mode, UINT32 Config)

Function Description:

- This function is used to calculate the cawarp parameters.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	**Mode**	Mode configure. Please refer to Section 11.2.2.4 for more details.
UINT32	**Config**	Configure warp module. **IMG_WARP_CONFIG_FORCE_DISABLE**: Not calculate cawarp parameters (digital zoom only).

*Table 11-22. Parameters for Warp Core API **AmbaDSP_WarpCore_CalcDspCawarp()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 11-23. Returns for Warp Core API **AmbaDSP_WarpCore_CalcDspCawarp()**.*

Example:

Please refer to the example of (Section 11.2.1) AmbaDSP_WarpCore_Init.

See Also:

None

11.2.8 AmbaDSP_WarpCore_SetDspWarp

API Syntax:

AmbaDSP_WarpCore_SetDspWarp (AMBA_DSP_IMG_MODE_CFG_s Mode)

Function Description:

- This function is used to prepare the warp command to DSP.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	**Mode**	Mode configure. Please refer to Section 11.2.2.4 for more details.

*Table 11-24. Parameters for Warp Core API **AmbaDSP_WarpCore_SetDspWarp()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 11-25. Returns for Warp Core API **AmbaDSP_WarpCore_SetDspWarp()**.*

Example:

Please refer to the example (Section 11.2.1) AmbaDSP_WarpCore_Init.

See Also:

None

11.2.9 AmbaDSP_WarpCore_SetDspCawarp

API Syntax:

AmbaDSP_WarpCore_SetDspCawarp (AMBA_DSP_IMG_MODE_CFG_s Mode)

Function Description:

- This function is used to prepare the cawarp command to DSP.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	**Mode**	Mode configure. Please refer to Section 11.2.2.4 for more details.

*Table 11-26. Parameters for Warp Core API **AmbaDSP_WarpCore_SetDspCawarp()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 11-27. Returns for Warp Core API **AmbaDSP_WarpCore_SetDspCawarp()**.*

Example:

Please refer to the example of (Section 11.2.1) AmbaDSP_WarpCore_Init.

See Also:

None

11.2.10 AmbaDSP_WarpCore_SetCalibWarpInfo

API Syntax:

AmbaDSP_WarpCore_SetCalibWarpInfo (AMBA_DSP_IMG_MODE_CFG_s Mode , AMBA_DSP_IMG_CALIB_WARP_INFO_s *pCalibWarpInfo)

Function Description:

- This function is used to set the warp calibration information for warp.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	**Mode**	Mode configure. Please refer to Section 11.2.2.4 more details.
AMBA_DSP_IMG_CALIB_WARP_INFO_s	***pCalibWarpInfo**	Calibration warp information. This is where the calibrated warp table grid numbers, tile size, and sensor geometry information are stored. Please refer to Section 11.2.10.1 for more details.

*Table 11-28. Parameters for Warp Core API **AmbaDSP_WarpCore_SetCalibWarpInfo()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 11-29. Returns for Warp Core API **AmbaDSP_WarpCore_SetCalibWarpInfo()**.*

Example:

Please refer to the example of (Section 11.2.1) AmbaDSP_WarpCore_Init.

See Also:

None

11.2.10.1 AmbaDSP_WarpCore_SetCalibWarpInfo > AMBA_DSP_IMG_CALIB_WARP_INFO_s

Type	Parameter	Description
UINT32	**Version**	The version number. Current version number is 0x20130101.
Int	**HorGridNum**	Horizontal grid number
Int	**VerGridNum**	Vertical grid number
Int	**TitleWidthExp**	Tile width exponent. Tile width is 2^**TileWidthExp**. Note that (**HorGridNum**-1) << **TileWidthExp** must be greater than or equal to **VinSensorGeo.Width**.
Int	**TitleHeightExp**	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	**VinSensorGro**	VIN sensor geometry when calibrating. Please refer to Section 11.2.2.6 for more details.
UINT32 x 3	**Reserved**	None
AMBA_DSP_IMG_GRID_POINT_s	***pWarp**	Warp grid vector array. Please refer to Section 11.2.10.2 below for details.

*Table 11-30. Definition of **AMBA_DSP_IMG_CALIB_WARP_INFO_s** for Warp Core API **AmbaDSP_WarpCore_SetCalibWarpInfo()**.*

11.2.10.2 AmbaDSP_WarpCore_SetCalibWarpInfo > AMBA_DSP_IMG_GRID_POINT_s

Type	Parameter	Description
INT16	**X**	The value of x-axis
INT16	**Y**	The value of y-axis

*Table 11-31. Definition of **AMBA_DSP_IMG_GRID_POINT_s** for Warp Core API **AmbaDSP_WarpCore_SetCalibWarpInfo()**.*

11.2.11 AmbaDSP_WarpCore_SetCalibCawarpInfo

API Syntax:

AmbaDSP_WarpCore_SetCalibCawarpInfo (AMBA_DSP_IMG_MODE_CFG_s Mode , AMBA_DSP_IMG_CALIB_CAWARP_INFO_s *pCalibCaWarpInfo)

Function Description:

- This function is used to prepare the cawarp command to DSP.

Parameters:

Type	Parameter	Description
AMBA_DSP_IMG_MODE_CFG_s	**Mode**	Mode configure. Please refer to Section 11.2.2.4 for more details.
AMBA_DSP_IMG_CALIB_CAWARP_INFO_s	***pCalibCaWarpInfo**	Calibration cawarp information. This is where the calibrated cawarp table grid numbers, tile size, and sensor geometry information are stored. Please refer to Section 11.2.11.1 for more details.

*Table 11-32. Parameters for Warp Core API **AmbaDSP_WarpCore_SetCalibCawarpInfo()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 11-33. Returns for Warp Core API **AmbaDSP_WarpCore_SetCalibCawarpInfo()**.*

Example:

Please refer to the example of (Section 11.2.1) AmbaDSP_WarpCore_Init.

See Also:

None

11.2.11.1 AmbaDSP_WarpCore_SetCalibCawarpInfo > AMBA_DSP_IMG_CALIB_WARP_INFO_s

Type	Parameter	Description
UINT32	**Version**	The version number. Current version number is 0x20130125.
Int	**HorGridNum**	Horizontal grid number
Int	**VerGridNum**	Vertical grid number
Int	**TitleWidthExp**	Tile width exponent. Tile width is 2^**TileWidthExp**. Note that (**HorGridNum**-1) << **TileWidthExp** must be greater than or equal to **VinSensorGeo.Width**.
Int	**TitleHeightExp**	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	**VinSensorGro**	VIN sensor geometry when calibrating. Please refer to Section 11.2.2.6 for more details
INT16	**RedScaleFactor**	Red scale factor. Red channel uses (RedScaleFactor * CawarpTableVector) / 256.
INT16	**BlueScaleFactor**	Blue scale factor. Blue channel uses (BlueScaleFactor * CawarpTableVector) / 256.
UINT32 x 3	**Reserved**	None
AMBA_DSP_IMG_GRID_POINT_s	***pWarp**	Warp grid vector array. Please refer to Section 11.2.10.2 for more details.

*Table 11-34. Definition of **AMBA_DSP_IMG_CALIB_WARP_INFO_s** for Warp Core API **AmbaDSP_WarpCore_SetCalibCawarpInfo()**.*

Appendix 1 Additional Resources

Related resources include:

- *A12 Datasheet (chip specific title)*
- *A12 Hardware Programming Reference Manual*
- *A12 RM: System Hardware*
- *A12 RM: Firmware and System Boot*
- *A12 RM: Software Design Specification*
- *A12 AN: ADC and IR Input*
- *A12 AN: Audio Codec Driver*
- *A12 AN: Custom Image Sensor Driver*
- *A12 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/chapter numbers for previous drafts may differ from those of the current version.

Version	Date	Comments
1.0	15 October 2014	Formatted for SDK6
1.1	31 October 2014	Updated Sections 3.2.3.28 and 3.2.3.30

Table A3-1. Revision History.