SDK6 API: Audio

Version 2.1

June 22, 2015

Ambarella

Confidentiality Notice:

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
amb_performance_t ***amb_operating_mode_t*** **amb_set_operating_mode()**	API details (e.g., functions, structures, and type definitions)
`/usr/local/bin` `success = amb_set_operating_mode (amb_hal_base_address, & operating_mode)`	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 specifies the Ambarella Audio (AmbaAudio) application programming interface (API) for Ambarella digital processing products. The AmbaAudio API is included in the AmbaSYS library.

The overview chapter is organized as follows:

1.2 Overview: Audio Framework

The AmbaAudio API framework includes the following five modules:

1. Audio Decoder
   - This module decodes the bit-stream from the demuxer.
2. Audio Encoder
   - This module encodes the data into a bit-stream and sends it to the muxer.
3. Audio Output
   - This module outputs data to I2S via DMA.
4. Audio Input
   - This module retrieves input PCM (pulse-code modulation) audio raw data from I2S via DMA.
5. Audio Buffer Unit (ABU)
   - This module serves as the PCM buffer between two audio tasks.

These five modules can be divided into three groups, based on function:

1. Source audio processing
2. Destination audio processing
3. Audio buffering

Table 1-1 lists each module according to the functional group.

AmbaAudio Module	Functional Group		
	Source Audio Processing	Destination Audio Processing	Audio Buffering
Audio Decoder	√		
Audio Encoder		√	
Audio Output		√	
Audio Input	√		
Audio Buffer Unit (ABU)			√

Table 1-1. Overview: AmbaAudio Modules According to Functional Group.

- The **Audio Decoder** and **Audio Input** modules are involved in **Source Audio Processing** tasks. These modules are responsible for feeding data to the **Audio Buffer Units** (**ABUs**). Each module can link its output to multiple **ABUs**.
- The **Audio Output** and **Audio Encoder** modules are involved in **Destination Audio Processing** tasks. These modules are responsible for receiving data from **Audio Buffer Units** (**ABUs**). Each module can link its input to multiple **ABUs** and then mix them.
- **Audio Buffer Units** (**ABUs**) are responsible for transferring pulse-code modulation (PCM) data between Source and Destination Audio Processing modules. Each **ABU** can only be associated with a single Source and Destination Audio Processing module.
- Together, these modules can be used to build audio systems that include audio encoding, decoding, preview, and trans-coding operations. Figure 1-1 below provides an example of audio system data flow.

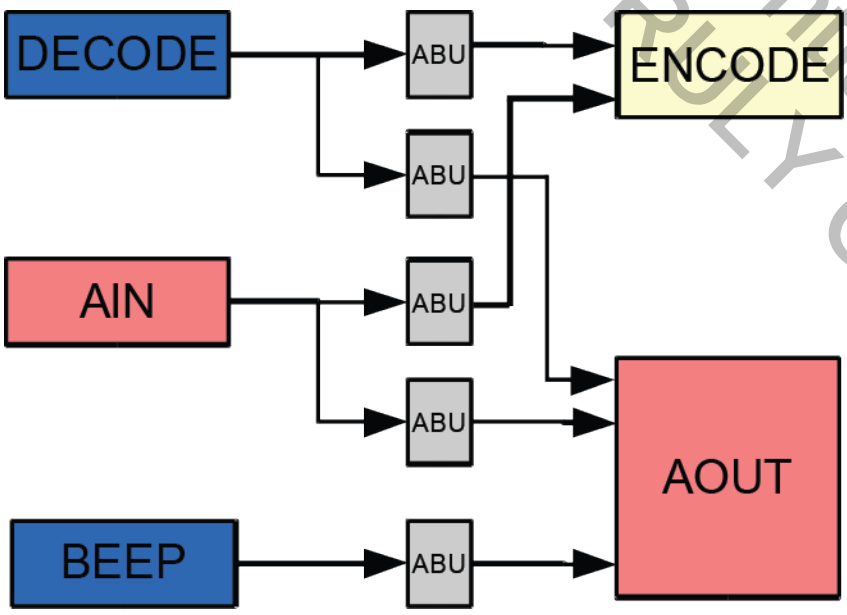

Figure 1-1. Overview: Audio Data Flow Example.

- Beep is one type of decode task.

1.3 Overview: Audio Module Event Handler

Each audio processing task provides event call-back handlers to increase the flexibility of audio control. The events include IDs which can be used for configuration and registry.

AmbaAudio Module	Event ID	Description	Parameters
Audio Encoder	**AMBA_AUDIO_EVENT_ID_ENCODE_ONE_FRAME**	Provides notification when one audio frame is encoded.	Encoded bit-stream information (AMBA_AUDIO_DESC_s *pBitsBufInfo)
	AMBA_AUDIO_EVENT_ID_ENCODE_FADE_OUT_DONE	Provides notification when fade-out processing is finished.	Task handler (UINT32 *pHdlr)
	AMBA_AUDIO_EVENT_ID_ENCODE_STOP	Provides notification when the last frame's bit-stream is sent.	Task handler (UINT32 *pHdlr)
Audio Input	**AMBA_AUDIO_EVENT_ID_INPUT_DMA_STOP**	Provides notification when DMA has stopped.	Task handler (UINT32 *pHdlr)
	AMBA_AUDIO_EVENT_ID_INPUT_CALIB_STATUS	Provides notification when calibration process is finished.	Calibration result (AMBA_AUDIO_CALIB_INFO_s *pCalibInfo). Please refer to Section 1.3.2.
Audio Decoder	**AMBA_AUDIO_EVENT_ID_DECODE_DATA_READY**	Provides notification when the output ABU of the decoder is full.	Event information (AMBA_AUDIO_EVENT_INFO_s *pInfo). Please refer to Section 1.3.1.
	AMBA_AUDIO_EVENT_ID_DECODE_FADE_OUT_DONE	Provides notification when fade-out processing is complete.	Task handler (UINT32 *pHdlr)
	AMBA_AUDIO_EVENT_ID_DECODE_STOP	Provides notification when decoding has stopped.	Task handler (UINT32 *pHdlr)
	AMBA_AUDIO_EVENT_ID_DECODE_EOS_STOP	Provides notification when decoding has stopped and the last frame is retrieved.	Task handler (UINT32 *pHdlr)
	AMBA_AUDIO_EVENT_ID_DECODE_RUN_TIME_ERROR	Provides notification when the error bit-stream is decoded.	Task handler (UINT32 *pHdlr)
	AMBA_AUDIO_EVENT_ID_DECODE_USE_ONE_FRAME	Provides notification when one frame from the demuxer is used.	Task handler (UINT32 *pHdlr)
Audio Output	**AMBA_AUDIO_EVENT_ID_OUTPUT_DMA_STOP**	Provides notification when DMA has stopped.	Task handler (UINT32 *pHdlr)
	AMBA_AUDIO_EVENT_ID_OUTPUT_ABU_GET_LOF	Provides notification when the last frame is retrieved from the ABU.	Event information (AMBA_AUDIO_EVENT_INFO_s *pInfo). Please refer to Section 1.3.1.
	AMBA_AUDIO_EVENT_ID_OUTPUT_NO_ABU_OP	Provides notification when there is no ABU in operating mode.	Task handler (UINT32 *pHdlr)

Table 1-2. Audio Module Event Handlers.

1.3.1 AMBA_AUDIO_EVENT_ID_OUTPUT_ABU_GET_LOF > AMBA_AUDIO_EVENT_INFO_s

Type	Field	Description
UINT32	***pHandle**	Handle of the task
UINT32	***pAbu**	Handle of the audio buffer

*Table 1-3. Definition of **AMBA_AUDIO_EVENT_INFO_s** for Audio API **AMBA_AUDIO_EVENT_ID_OUTPUT_ABU_GET_LOF()**.*

1.3.2 AMBA_AUDIO_EVENT_ID_INPUT_CALIB_STATUS > AMBA_AUDIO_CALIB_INFO_s

Type	Field	Description
UINT32	***pHandle**	Handle of the audio input task
UINT32	**Status**	If the audio calibration is done, it will be 1; otherwise it will be 2 (audio calibration fail).

*Table 1-4. Definition of **MBA_AUDIO_CALIB_INFO_s** for Audio API **AMBA_AUDIO_EVENT_ID_INPUT_CALIB_STATUS()**.*

Here are the example codes to register audio calibration call back event.

```
static AMBA_AUDIO_EVENT_HANDLER_f AmbaAudioInputCalibEntries[1];
static int AudioInputCalibCB(void *pEventData)
{
    AMBA_AUDIO_CALIB_INFO_s *pCalibInfo = pEventData;
    int Rtval = 0;
    INT32 *pTargetdBFS;
    INT32 *pTargetTHD_N;

    if (pCalibInfo->Status == 1) {

      pTargetdBFS = AmbaAudio_InputCalibGet_dBFS(pAudioInputCtrl);
      pTargetTHD_N= AmbaAudio_InputCalibGetTHD_N(pAudioInputCtrl);
      AmbaPrint("dBFS: %d %d", *pTargetdBFS, *(pTargetdBFS+1));
      AmbaPrint("THD+N: %d %d", *pTargetTHD_N, *(pTargetTHD_N+1));
    }
    AmbaPrint("%s: %d", __func__, pCalibInfo->Status);
    return Rtval;
}

AmbaAudio_EventHandlerCtrlConfig(pAudioInputCtrl,
            AMBA_AUDIO_EVENT_ID_INPUT_CALIB_STATUS, 1,
            AmbaAudioInputCalibEntries);
AmbaAudio_RegisterEventHandler(pAudioInputCtrl,
            AMBA_AUDIO_EVENT_ID_INPUT_CALIB_STATUS,
            AudioInputCalibCB);
```

1.4 Overview: Audio Processing Control Examples

This section provides simple programming examples for various audio processing tasks. Note that for each example, the user is required to open the source and destination I/O nodes. Refer to Figure 1-2 below.

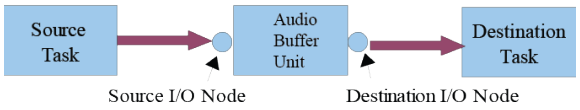

Figure 1-2. Overview: I/O Nodes Between Two Tasks.

Example 1: Decoder Processing

1. Create decoder-required resources.
   - Related APIs:
     - **AmbaAudio_DecSizeQuery()**
     - **AmbaAudio_DecCreate()**
2. Configure and register audio events for decoder flow.
   - Related APIs:
     - **AmbaAudio_EventHandlerCtrlConfig()**
     - **AmbaAudio_RegisterEventHandler()**
3. Create and setup the decoder task.
   - Related APIs:
     - **AmbaAudio_DecTaskCreate()**
     - **AmbaAudio_DecTaskSetUp()**
4. Start the decoder task and open the audio buffer source I/O node to feed the data to the audio buffer.
   - Related APIs:
     - **AmbaAudio_DecSetVpts()**
     - **AmbaAudio_BufferOpenSrcIoNode()**
     - **AmbaAudio_DecTaskStart()**
5. Wait for the **AMBA_AUDIO_EVENT_ID_DECODE_DATA_READY** event and then open the audio buffer destination I/O node. Start the output task if it has not started already.
   - Related APIs:
     - **AmbaAudio_BufferOpenDstIoNode()**
6. Stop the decoder task.
   - Related APIs:
     - **AmbaAudio_OutputTaskStop()**
7. Wait for the **AMBA_AUDIO_EVENT_ID_OUTPUT_ABU_GET_LOF** event to verify that the last de-

coded frame has been sent to the output task. Stop the output task if unused.

Example 2: Encoder Processing

1. Create encoder-required resources.
   - Related APIs:
     - **AmbaAudio_EnSizeQuery()**
     - **AmbaAudio_EncCreate()**
2. Configure and register audio events for the encoder flow.
   - Related APIs:
     - **AmbaAudio_EventHandlerCtrlConfig()**
     - **AmbaAudio_RegisterEventHandler()**
3. Create and setup the encoder task.
   - Related APIs:
     - **AmbaAudio_EncTaskCreate()**
     - **AmbaAudio_EncTaskSetUp()**
4. Start the encoder task and open the audio buffer destination I/O node waiting for data from the audio buffer.
   - Related APIs:
     - **AmbaAudio_BufferOpenDstIoNode()**
     - **AmbaAudio_EncTaskStart()**
5. Open the audio buffer source I/O node and start the input task if it has not started already.
   - Related APIs:
     - **AmbaAudio_BufferOpenSrcIoNode()**
6. Stop the audio encoder and the muxer to wait for the last encoded frame.
   - Related APIs:
     - **AmbaAudio_EncTaskStop()**

Audio output and input tasks are considered real-time tasks as they are triggered by DMA interrupts. Note that audio output tasks will output a mute frame if unable to retrieve data from the audio buffer, while audio input tasks will lose data if unable to feed data into the audio buffer.

Example 3: Output Processing

1. Create output-required resources.
   - Related APIs:
     - **AmbaAudio_OutputCachedSizeQuery()**
     - **AmbaAudio_OutputNonCachedSizeQuery()**
     - **AmbaAudio_OutputCreate()**
2. Configure and register audio events for output flow.
   - Related APIs:
     - **AmbaAudio_EventHandlerCtrlConfig()**
     - **AmbaAudio_RegisterEventHandler()**
3. Create the output task.
   - Related APIs:
     - **AmbaAudio_OutputTaskCreate()**
4. Start the output task.
   - Related APIs:
     - **AmbaAudio_OutputTaskStart()**
5. Check the **AMBA_AUDIO_EVENT_ID_OUTPUT_NO_ABU_OPS** event and stop the output task if it is unused.
   - Related APIs:
     - **AmbaAudio_OutputTaskStop()**
6. Verify that an **AMBA_AUDIO_EVENT_ID_OUTPUT_DMA_STOP** event has been issued and received, and then another cycle can begin.

Example 4: Input Processing

1. Create input-required resources.
   - Related APIs:
     - **AmbaAudio_InputCachedSizeQuery()**
     - **AmbaAudio_InputNonCachedSizeQuery()**
     - **AmbaAudio_InputCreate()**
2. Configure and register audio events for input flow.
   - Related APIs:
     - **AmbaAudio_EventHandlerCtrlConfig()**
     - **AmbaAudio_RegisterEventHandler()**
3. Create the input task.
   - Related APIs:
     - **AmbaAudio_InputTaskCreate()**

4. Start the input task.
    - Related APIs:
        - **AmbaAudio_InputTaskStart()**
5. Verify that an **AMBA_AUDIO_EVENT_ID_INPUT_DMA_STOP** event has been issued and received, and then another cycle can begin.

1.5 Overview: Bit-stream Transfer Management

The audio encoder and decoder modules use event call-back functions to manage the bit-stream transfer process between the muxer and the demuxer. This section provides further information regarding bit-stream transfer management.

1. Encoder:

   Please refer to the **EncGetBsAddr** field (Section 2.2.16.1), a call-back function which provides the write-point address to the encoder. Sufficient space for this address must be prepared for the encoder writing the bit-stream. Once the encoding process is complete, the encoder will provide the muxer with a notification of the encoder status through the audio event **AMBA_AUDIO_EVENT_ID_ENCODE_ONE_FRAME** with parameter (AMBA_AUDIO_DESC_s *__pBitsBufInfo__). Please refer to Section 2.2.6.2 for the definition.

2. Decoder:

   Please refer to the **DecGetBsDesc** field (Section 2.2.6.1), a call-back function which provides decoded bit-stream information to the decoder. The descriptor **AMBA_AUDIO_DESC_s** must be configured with the appropriate setting information. Please note that every descriptor is subject to a data size limit, derived from information used to create the decoder resource (Section 2.2.1.1, **AMBA_AUDIO_TASK_CREATE_INFO_s**). The data limit size is equal to (MaxFrameSize * MaxChNum * sizeof(INT32)). Once the bit-stream has finished processing, the decoder will notify the demuxer through an **AMBA_AUDIO_EVENT_ID_DECODE_USE_ONE_FRAME** audio event. Once this event has been received, the demuxer can reuse the write pointer address.

1.6 Overview: Scope of Document

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

- The chip datasheet provides hardware pin and package details including a feature list with descriptions of the chip performance, brief interface descriptions, a complete power-on configuration table and electrical characteristics.
- Refer to Appendix 1 for additional resources.

2 Audio API

2.1 Audio API: Overview

This chapter details the API functions involved in audio system building. These functions include tasks related to each of the five AmbaAudio modules described in Chapter 1:

1. Audio Decoder
2. Audio Encoder
3. Audio Output
4. Audio Input
5. Audio Buffer Unit (ABU)
6. Audio Plug-in Effect

Additional functions provide event call-back notifications used for audio system management.

2.2 Audio API: List of Functions

- AmbaAudio_DecSizeQuery
- AmbaAudio_DecCreate
- AmbaAudio_DecDelete
- AmbaAudio_DecTaskCreate
- AmbaAudio_DecTaskDelete
- AmbaAudio_DecTaskSetUp
- AmbaAudio_DecTaskStart
- AmbaAudio_DecTaskStop
- AmbaAudio_DecSetVpts
- AmbaAudio_DecSetVolume
- AmbaAudio_EncSizeQuery
- AmbaAudio_EncCreate
- AmbaAudio_EncDelete
- AmbaAudio_EncTaskCreate
- AmbaAudio_EncTaskDelete
- AmbaAudio_EncTaskSetUp
- AmbaAudio_EncTaskStart
- AmbaAudio_EncTaskStop

- AmbaAudio_EncSetVolume
- AmbaAudio_OutputCachedSizeQuery
- AmbaAudio_OutputNonCachedSizeQuery
- AmbaAudio_OutputCreate
- AmbaAudio_OutputDelete
- AmbaAudio_OutputTaskCreate
- AmbaAudio_OutputTaskDelete
- AmbaAudio_OutputTaskStart
- AmbaAudio_OutputTaskStop
- AmbaAudio_OutputSetVolume
- AmbaAudio_InputCachedSizeQuery
- AmbaAudio_InputNonCachedSizeQuery
- AmbaAudio_InputCreate
- AmbaAudio_InputDelete
- AmbaAudio_InputTaskCreate
- AmbaAudio_InputTaskDelete
- AmbaAudio_InputTaskStart
- AmbaAudio_InputTaskStop
- AmbaAudio_InputSetVolume
- AmbaAudio_BufferSizeQuery
- AmbaAudio_BufferCreate
- AmbaAudio_BufferDelete
- AmbaAudio_BufferReset
- AmbaAudio_BufferOpenSrcIoNode
- AmbaAudio_BufferOpenDstIoNode
- AmbaAudio_Combine
- AmbaAudio_Detach
- AmbaAudio_EventHandlerCtrlConfig
- AmbaAudio_RegisterEventHandler
- AmbaAudio_UnRegisterEventHandler
- AmbaAudio_EventHandlerCtrlReset
- AmbaAudio_OutputPluginEffectInstall
- AmbaAudio_OutputPluginEffectEnable
- AmbaAudio_OutputPluginEffectDisable
- AmbaAudio_OutputPluginEffectUpdate
- AmbaAudio_InputPluginEffectInstall
- AmbaAudio_InputPluginEffectEnable
- AmbaAudio_InputPluginEffectDisable

- AmbaAudio_InputPluginEffectUpdate
- AmbaAudio_InputPowerMonitorEnable
- AmbaAudio_InputPowerMonitorGetdB
- AmbaAudio_InputPowerMonitorDisable
- AmbaAudio_InputSetUpCalib
- AmbaAudio_InputCalibGetCurve
- AmbaAudio_InputCalibGet_dBFS
- AmbaAudio_InputCalibGetTHD_N
- AmbaAudio_InputCalibGetFreqCurve
- AmbaAudio_InputDisableCalib
- AmbaAudio_EffectCalibBufferSize
- AmbaAudio_EffectUpdownSampleRateConvertSetup

2.2.1 AmbaAudio_DecSizeQuery

API Syntax:

AmbaAudio_DecSizeQuery (AMBA_AUDIO_TASK_CREATE_INFO_s *pInfo)

Function Description:

- This function is used to query the required memory size when creating a decoder resource.

Parameters:

Type	Parameter	Description
AMBA_AUDIO_TASK_CREATE_INFO_s	***pInfo**	Audio decoder task creation information. Please refer to Section 2.2.1.1 below for the definition.

*Table 2-1. Parameters for Audio API **AmbaAudio_DecSizeQuery()**.*

Returns:

Return	Description
> 0	Required memory size

*Table 2-2. Returns for Audio API **AmbaAudio_DecSizeQuery()**.*

Example:

```
AMBA_AUDIO_TASK_CREATE_INFO_s DecInfo;
UINT32 DecSize;

DecInfo.MaxSampleFreq = 48000;
DecInfo.MaxChNum = 2;
DecInfo.MaxFrameSize = 2048;
DecSize = AmbaAudio_DecSizeQuery(&DecInfo);
```

See Also:

AmbaAudio_DecCreate()

2.2.1.1 AmbaAudio_DecSizeQuery > AMBA_AUDIO_TASK_CREATE_INFO_s

Type	Field	Description
UINT32	**MaxSampleFreq**	Maximum supporting sampling frequency of the decoder task.
UINT32	**MaxChNum**	Maximum supporting channel number
UINT32	**MaxFrameSize**	Maximum possible decoded frame size

*Table 2-3. Definition of **AMBA_AUDIO_TASK_CREATE_INFO_s** for Audio API **AmbaAudio_DecSizeQuery()**.*

2.2.2 AmbaAudio_DecCreate

API Syntax:

AmbaAudio_DecCreate (AMBA_AUDIO_TASK_CREATE_INFO_s *pInfo, UINT32 *pAddr, UINT32 BufferSize)

Function Description:

- This function is used to generate resources required by a decoder task.

Parameters:

Type	Parameter	Description
AMBA_AUDIO_TASK_CREATE_INFO_s	***pInfo**	Audio decoder task creation information. Please refer to Section 2.2.1.1 for the definition.
UINT32	***pAddr**	Pointer of the memory prepared for decoder task resource
UINT32	**BufferSize**	Size of the memory prepared for decoder task resource

*Table 2-4. Parameters for Audio API **AmbaAudio_DecCreate**.*

Returns:

Return	Description
Address	Pointer of the decoder resource handler.
0xFFFFFFFF	Failure

*Table 2-5. Returns for Audio API **AmbaAudio_DecCreate**.*

Example:

```
AMBA_AUDIO_TASK_CREATE_INFO_s DecInfo;
UINT32 DecSize;
UINT32 *pAudioDecCtrl;
UINT32 *pDecAddr

DecInfo.MaxSampleFreq = 48000;
DecInfo.MaxChNum = 2;
DecInfo.MaxFrameSize = 2048;
DecSize = AmbaAudio_DecSizeQuery(&DecInfo);

/* Allocate DecSize buffer with *pDecAddr */

pAudioDecCtrl = AmbaAudio_DecCreate(&DecInfo, pDecAddr, DecSize);
```

See Also:

AmbaAudio_DecSizeQuery()

2.2.3 AmbaAudio_DecDelete

API Syntax:

AmbaAudio_DecDelete (UINT32 *pHdlr)

Function Description:

- This function is used to delete a specified decoder resource.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the decoder resource

*Table 2-6. Parameters for Audio API **AmbaAudio_DecDelete()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-7. Returns for Audio API **AmbaAudio_DecDelete()**.*

Example:

```
UINT32 *pAudioDecCtrl;

...
AmbaAudio_DecDelete(pAudioDecCtrl);
```

See Also:

AmbaAudio_DecCreate()

2.2.4 AmbaAudio_DecTaskCreate

API Syntax:

AmbaAudio_DecTaskCreate (UINT32 *pHdlr, INT32 Priority, UINT32 CoreExlusionMap)

Function Description:

- This function is used to generate a decoder task.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the decoder resource
INT32	**Priority**	Task priority of the decoder task
UINT32	**CoreExlusionMap**	The core selection of this task

*Table 2-8. Parameters for Audio API **AmbaAudio_DecTaskCreate()**.*

Returns:

Return	Description
0	Locked
- 1	Not locked yet

*Table 2-9. Returns for Audio API **AmbaAudio_DecTaskCreate()**.*

Example:

```
UINT32 *pAudioDecCtrl;

…..
AmbaAudio_DecTaskCreate(pAudioDecCtrl, 32, 0);
```

See Also:

AmbaAudio_DecTaskDelete()

2.2.5 AmbaAudio_DecTaskDelete

API Syntax:

AmbaAudio_DecTaskDelete (UINT32 *pHdlr)

Function Description:

- This function is used to delete a specified decoder task .

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the decoder resource

*Table 2-10. Parameters for Audio API **AmbaAudio_DecTaskDelete()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-11. Returns for Audio API **AmbaAudio_DecTaskDelete()**.*

Example:

```
UINT32 *pAudioDecCtrl;

…..
AmbaAudio_DecTaskDelete(pAudioDecCtrl);
```

See Also:

AmbaAudio_DecTaskCreate()

2.2.6 AmbaAudio_DecTaskSetUp

API Syntax:

AmbaAudio_DecTaskSetUp (UINT32 *pHdlr, AMBA_AUDIO_DEC_SETUP_INFO_s *pConfig)

Function Description:

- This function is used to configure the decoder task settings.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the decoder resource
AMBA_AUDIO_DEC_SETUP_INFO_s	***pConfig**	Audio Decoder task set up information. Please refer to Section 2.2.6.1 below for the definition.

*Table 2-12. Parameters for Audio API **AmbaAudio_DecTaskSetUp()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-13. Returns for Audio API **AmbaAudio_DecTaskSetUp()**.*

Example:

```
UINT32 *pAudioDecCtrl;

….
AMBA_AUDIO_DEC_SETUP_INFO_s AudioSetupCfg;
AMBA_AUDIO_PCM_CONFIG_s PcmCfg;

memset(&PcmCfg, 0, sizeof(AMBA_AUDIO_PCM_CONFIG_s));
memset(&AudioSetupCfg, 0, sizeof(AMBA_AUDIO_DEC_SETUP_INFO_s));

PcmCfg.BitsPerSample    = 16;
PcmCfg.DataFormat       = 0;
PcmCfg.FrameSize        = 1024;
AudioSetupCfg.PureAudio  = PURE_AUDIO;
AudioSetupCfg.SrcSampleFreq = 48000;
AudioSetupCfg.DstSampleFreq = 48000;
AudioSetupCfg.SrcChMode  = 2;
AudioSetupCfg.DstChMode  = 2;
AudioSetupCfg.DecType = AMBA_AUDIO_DEC_PCM ;
AudioSetupCfg.pDecConfig = &PcmCfg;
AudioSetupCfg.DecGetBsDesc = DecGetReadPointer;
```

```
AmbaAudio_DecTaskSetUp(pAudioDecCtrl, &AudioSetupCfg);
```

See Also:

None

2.2.6.1 AmbaAudio_DecTaskSetUp > AMBA_AUDIO_DEC_SETUP_INFO_s

Type	Field	Description
AMBA_AUDIO_DEC_FLOW_e	**PureAudio**	0: VIDEO_AUDIO /* A/V pre-sync */ 1: PURE_AUDIO /* No pre-sync */
UINT32	**SrcSampleFreq**	Sample frequency of the input
UINT32	**DstSampleFreq**	Sample frequency of the output
UINT32	**SrcChMode**	Channel number of the input
UINT32	**DstChMode**	Channel number of the output
AMBA_AUDIO_DEC_TYPE_e	**DecType**	0x00: AMBA_AUDIO_PCM (Audio RAW data, no compression) 0x10: AMBA_AUDIO_AAC (AAC Plain) 0x11: AMBA_AUDIO_AAC_PLUS (AAC Plus) 0x12: AMBA_AUDIO_PLUS_V2 (AAC Plus V2) 0x20: AMBA_AUDIO_ADPCM (IMA ADPCM) 0x30: AMBA_AUDIO_AC3 0x40: AMBA_AUDIO_MPEG 0x50: AMBA_AUDIO_OPUS 0x60: AMBA_AUDIO_G711A (G711 a-law) 0x61: AMBA_AUDIO_G711U (G711 u-law) 0x62: AMBA_AUDIO_G726A (G726 a-law) 0x63: AMBA_AUDIO_G726U (G726 u-law)
AMBA_AUDIO_DEC_GET_BS_DESC	**DecGetBsDesc**	`typedef AMBA_AUDIO_DESC_s* (*AMBA_AUDIO_DEC_GET_BS_DESC)(UINT32 *pHdlr);` /* Callback function for decoder to retrieve bitstream */. Please refer to Section 2.2.6.2 for the definition of **AMBA_AUDIO_DESC_s**.
void	***pDecConfig**	/* CODEC configuration data pointer. */ PCM and AAC decoder configuration are defined in `AmbaAudio.h`.

*Table 2-14. Definition of **AMBA_AUDIO_DEC_SETUP_INFO_s** for Audio API **AmbaAudio_DecTaskSetUp()**.*

2.2.6.2 AmbaAudio_DecTaskSetUp > AMBA_AUDIO_DESC_s

Type	Field	Description
UINT32	***pHdlr**	Handle of the task
UINT32	**Pts**	Presentation time-stamp (PTS) of this descriptor
UINT32	**PicType**	Set to 1 if this is the final descriptor
UINT32	**DataSize**	Valid data size, from starting address
UINT8	***pBufAddr**	Buffer starting address

*Table 2-15. Definition of **AMBA_AUDIO_DESC_s** for Audio API **AmbaAudio_DecTaskSetUp()**.*

2.2.6.3 AmbaAudio_DecTaskSetUp > pDecConfig

CODEC configuration data point can be assigned to different CODEC data pointers depending on the “DecType.” Audio decoder task provides the following types for user to set the decoder.

Type	Decoder configuration data structures
AMBA_AUDIO_PCM	AMBA_AUDIO_PCM_CONFIG_s
AMBA_AUDIO_AAC / AMBA_AUDIO_AAC_PLUS / AMBA_AUDIO_PLUS_V2	AMBA_AUDIO_AACDEC_CONFIG_s
AMBA_AUDIO_ADPCM	AMBA_AUDIO_ADPCM_CONFIG_s
AMBA_AUDIO_AC3	AMBA_AUDIO_AC3DEC_CONFIG_s
AMBA_AUDIO_OPUS	AMBA_AUDIO_OPUSDEC_CONFIG_s
AMBA_AUDIO_G711A	AMBA_AUDIO_G7XXDEC_CONFIG_s

Table 2-16. Decoder Configuration Data Structures for Different Audio Decoder Types.

2.2.6.4 AmbaAudio_DecTaskSetUp > AMBA_AUDIO_PCM_CONFIG_s

Type	Field	Description
UINT32	**BitsPerSample**	8, 16, 24, or 32-bits per sample
UINT32	**DataFormat**	Endian, Audio_BS_Intel (LSB,MSB), Audio_BS_Motorola (MSB, LSB).
UINT32	**FrameSize**	Not necessary in the decoder mode

Table 2-17. PCM Decoder Configuration Data Structure.

2.2.6.5 AmbaAudio_DecTaskSetUp > AMBA_AUDIO_AACDEC_CONFIG_s

Type	Field	Description
AMBA_AUDIO_AAC_BS_TYPE_e	**BitstreamType**	Bit-stream Type of AAC. AAC_BS_RAW, AAC_BS_ADIF, (Bit-stream with ADIF header) AAC_BS_ADTS, (Bit-stream with ADTS header) AAC_BS_LOAS, (Bit-stream with LOAS header)

Table 2-18. AAC Decoder Configuration Data Structure.

2.2.6.6 AmbaAudio_DecTaskSetUp > AMBA_AUDIO_ADPCM_CONFIG_s

Type	Field	Description
UINT32	**AdpcmFrameSize**	Samples Per Block

Table 2-19. ADPCM Decoder Configuration Data Structure.

2.2.6.7 AmbaAudio_DecTaskSetUp > AMBA_AUDIO_AC3DEC_CONFIG_s

Type	Field	Description
UINT8	**Ac3DecKCapableMode**	Karaoke capable mode
UINT8	**Ac3DecCompMode**	Compression mode
UINT8	**Ac3DecStereoMode**	Stereo downmix mode
UINT8	**Ac3DecDualMono-Mode**	Dual mono reproduction mode
UINT8	**Ac3DecOutputMode**	Output channel configuration
UINT8	**Ac3DecOutLfeOn**	Output subwoofer present flag
UINT16	**Ac3DecOutPair**	Output channel pair
UINT16	**Ac3DecWordSize**	Output word size code
UINT16	**Ac3DecNumChans**	Output number channel
INT32	**Ac3DecDynRngS-caleLow**	Dynamic range scale factor (low): 0x0~0x7FFFFFFF
INT32	**Ac3DecDynRngScaleHi**	Dynamic range scale factor (high): 0x0~0x7FFFFFFF
INT32	**Ac3DecPcmScaleFac**	PCM scale factor: 0x0~0x7FFFFFFF
UINT8	**Ac3DecBsEndian**	Bitstream Endian

Table 2-20. AC3 Decoder Configuration Data Structure.

2.2.6.8 AmbaAudio_DecTaskSetUp > AMBA_AUDIO_OPUSDEC_CONFIG_s

Type	Field	Description
UINT32	**FrameSize**	Sample number per frame per channel
UINT8	**BitstreamType**	Decoded bit-stream type (**OPUS_BS_RTP** or **OPUS_BS_RAW**)

Table 2-21. OPUS Decoder Configuration Data Structure.

2.2.6.9 AmbaAudio_DecTaskSetUp > AMBA_AUDIO_G7XXDEC_CONFIG_s

Type	Field	Description
INT32	**Rate**	The compression rate for G726

Table 2-22. G7XX Decoder Configuration Data Structure.

2.2.7 AmbaAudio_DecTaskStart

API Syntax:

AmbaAudio_DecTaskStart (UINT32 *pHdlr, UINT32 FadeInTime)

Function Description:

- This function is used to initialize the decoder process.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the decoder resource
UINT32	**FadeInTime**	Fade-in time of the decoder (ms)

*Table 2-23. Parameters for Audio API **AmbaAudio_DecTaskStart()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-24. Returns for Audio API **AmbaAudio_DecTaskStart()**.*

Example:

```
UINT32 *pAudioDecCtrl;
….

AmbaAudio_DecTaskStart(pAudioDecCtrl, 10);
```

See Also:

AmbaAudio_DecTaskStop()

2.2.8 AmbaAudio_DecTaskStop

API Syntax:

AmbaAudio_DecTaskStop (UINT32 *pHdlr, UINT32 FadeOutTime)

Function Description:

- This function is used to terminate the decoder process.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the decoder resource
UINT32	**FadeOutTime**	Fade-out time of the decoder (ms)

*Table 2-25. Parameters for Audio API **AmbaAudio_DecTaskStop()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-26. Returns for Audio API **AmbaAudio_DecTaskStop()**.*

Example:

```
UINT32 *pAudioDecCtrl;
….

AmbaAudio_DecTaskStop(pAudioDecCtrl, 10);
```

See Also:

AmbaAudio_DecTaskStart()

2.2.9 AmbaAudio_DecSetVpts

API Syntax:

AmbaAudio_DecSetVpts (UINT32 *pHdlr, UINT32 Vpts)

Function Description:

- This function is used to specify video presentation time-stamp (VPTS) information for the decoder task to ensure A/V synchronization.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the decoder resource
UINT32	**Vpts**	Video PTS of the first output frame

*Table 2-27. Parameters for Audio API **AmbaAudio_DecSetVpts()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-28. Returns for Audio API **AmbaAudio_DecSetVpts()**.*

Example:

```
UINT32 *pAudioDecCtrl;
….

AmbaAudio_DecSetVpts(pAudioDecCtrl, 100);
```

See Also:

None

2.2.10 AmbaAudio_DecSetVolume

API Syntax:

AmbaAudio_DecSetVolume (UINT32 *pHdlr, UINT32 Vpts)

Function Description:

- This function is used to configure volume settings for a specified decoder task.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the decoder resource
UINT32	**Volume**	Volume of the decoder task (min. 0 ~ max. 64)

*Table 2-29. Parameters for Audio API **AmbaAudio_DecSetVolume()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-30. Returns for Audio API **AmbaAudio_DecSetVolume()**.*

Example:

```
UINT32 *pAudioDecCtrl;
….

AmbaAudio_DecSetVolume(pAudioDecCtrl, 64);
```

See Also:

None

2.2.11 AmbaAudio_EncSizeQuery

API Syntax:

AmbaAudio_EncSizeQuery (AMBA_AUDIO_TASK_CREATE_INFO_s *pInfo)

Function Description:

- This function is used to query the memory size required to generate an encoder resource.

Parameters:

Type	Parameter	Description
AMBA_AUDIO_TASK_CREATE_INFO_s	***pInfo**	Audio encoder task creation information. Please refer to Section 2.2.1.1 for the definition.

*Table 2-31. Parameters for Audio API **AmbaAudio_EncSizeQuery()**.*

Returns:

Return	Description
> 0	Required memory size

*Table 2-32. Returns for Audio API **AmbaAudio_EncSizeQuery()**.*

Example:

```
AMBA_AUDIO_TASK_CREATE_INFO_s EncInfo;
UINT32 EncSize;

EncInfo.MaxSampleFreq = 48000;
EncInfo.MaxChNum = 2;
EncInfo.MaxFrameSize = 2048;
EncSize = AmbaAudio_EncSizeQuery(&EncInfo);
```

See Also:

AmbaAudio_EncCreate()

2.2.12 AmbaAudio_EncCreate

API Syntax:

AmbaAudio_EncCreate (AMBA_AUDIO_TASK_CREATE_INFO_s *pInfo, UINT32 *pAddr, UINT32 BufferSize)

Function Description:

- This function is used to generate resources required by an encoder task.

Parameters:

Type	Parameter	Description
AMBA_AUDIO_TASK_CREATE_INFO_s	***pInfo**	Audio encoder task creation information. Please refer to Section 2.2.1.1 for the definition.
UINT32	***pAddr**	Pointer of the memory prepared for encoder task resource
UINT32	**BufferSize**	Size of the memory prepared for encoder task resource

*Table 2-33. Parameters for Audio API **AmbaAudio_EncCreate()**.*

Returns:

Return	Description
Address	Pointer of the encoder resource handler.
0xFFFFFFFF	Failure

*Table 2-34. Returns for Audio API **AmbaAudio_EncCreate()**.*

Example:

```
AMBA_AUDIO_TASK_CREATE_INFO_s EncInfo;
UINT32 EncSize;
UINT32 *pAudioEncCtrl;
UINT32 *pEncAddr

EncInfo.MaxSampleFreq = 48000;
EncInfo.MaxChNum = 2;
EncInfo.MaxFrameSize = 2048;
EncSize = AmbaAudio_EncSizeQuery(&DecInfo);

/* Allocate EncSize buffer with *pEncAddr */

pAudioEncCtrl = AmbaAudio_EncCreate(&EncInfo, pEncAddr, EncSize);
```

See Also:

AmbaAudio_EncSizeQuery()

2.2.13 AmbaAudio_EncDelete

API Syntax:

AmbaAudio_EncDelete (UINT32 *pHdlr)

Function Description:

- This function is used to delete a specified encoder resource.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the Encoder resource

*Table 2-35. Parameters for Audio API **AmbaAudio_EncDelete()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-36. Returns for Audio API **AmbaAudio_EncDelete()**.*

Example:

```
UINT32 *pAudioEncCtrl;

...
AmbaAudio_EncDelete(pAudioEncCtrl);
```

See Also:

AmbaAudio_EncCreate()

2.2.14 AmbaAudio_EncTaskCreate

API Syntax:

AmbaAudio_EncTaskCreate (UINT32 *pHdlr, INT32 Priority, UINT32 CoreExlusionMap)

Function Description:

- This function is used to generate an encoder task.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the Encoder resource
INT32	**Priority**	Task priority of the Encoder task
UINT32	**CoreExlusionMap**	The core selection of this task

*Table 2-37. Parameters for Audio API **AmbaAudio_EncTaskCreate()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-38. Returns for Audio API **AmbaAudio_EncTaskCreate()**.*

Example:

```
UINT32 *pAudioEncCtrl;

…..
AmbaAudio_EncTaskCreate(pAudioEncCtrl, 32, 0);
```

See Also:

AmbaAudio_EncTaskDelete()

2.2.15 AmbaAudio_EncTaskDelete

API Syntax:

AmbaAudio_EncTaskDelete (UINT32 *pHdlr)

Function Description:

- This function is used to delete a specified encoder task.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the Encoder resource

*Table 2-39. Parameters for Audio API **AmbaAudio_EncTaskDelete()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-40. Returns for Audio API **AmbaAudio_EncTaskDelete()**.*

Example:

```
UINT32 *pAudioEncCtrl;

…..
AmbaAudio_EncTaskDelete(pAudioEncCtrl);
```

See Also:

AmbaAudio_EncTaskCreate()

2.2.16 AmbaAudio_EncTaskSetUp

API Syntax:

AmbaAudio_EncTaskSetUp (UINT32 *pHdlr, AMBA_AUDIO_ENC_SETUP_INFO_s *pConfig)

Function Description:

- This function is used to configure encoder task settings.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the Encoder resource
AMBA_AUDIO_ENC_SETUP_INFO_s	***pConfig**	Audio encoder task set up information. Please refer to Section 2.2.16.1 below for the definition.

*Table 2-41. Parameters for Audio API **AmbaAudio_EncTaskSetUp()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-42. Returns for Audio API **AmbaAudio_EncTaskSetUp()**.*

Example:

```
UINT32 *pAudioEncCtrl;

….
AMBA_AUDIO_ENC_SETUP_INFO_s AudioSetupCfg;
AMBA_AUDIO_PCM_CONFIG_s PcmCfg;

memset(&PcmCfg, 0, sizeof(AMBA_AUDIO_PCM_CONFIG_s));
memset(&AudioSetupCfg, 0, sizeof(AMBA_AUDIO_ENC_SETUP_INFO_s));

PcmCfg.BitsPerSample    = 16;
PcmCfg.DataFormat       = 0;
PcmCfg.FrameSize        = 1024;
AudioSetupCfg.SrcSampleFreq = 48000;
AudioSetupCfg.DstSampleFreq = 48000;
AudioSetupCfg.SrcChMode  = 2;
AudioSetupCfg.DstChMode  = 2;
AudioSetupCfg.pEncConfig = &PcmCfg;
AudioSetupCfg.EncType = AMBA_AUDIO_ENC_PCM;
AudioSetupCfg.EncGetBsAddr = EncGetWritePointer;

AmbaAudio_EncTaskSetUp(pAudioEncCtrl, &AudioSetupCfg);
```

See Also:

None

2.2.16.1 AmbaAudio_ EncTaskSetUp > AMBA_AUDIO_ENC_SETUP_INFO_s

Type	Field	Description
UINT32	**SrcSampleFreq**	Sample frequency of the input
UINT32	**DstSampleFreq**	Sample frequency of the output
UINT32	**SrcChMode**	Channel number of the input
UINT32	**DstChMode**	Channel number of the output
AMBA_AUDIO_ENC_TYPE_e	**EncType**	0x00: AMBA_AUDIO_ENC_PCM (Audio RAW data, no compression) 0x10: AMBA_AUDIO_ENC_AAC (AAC Plain) 0x11: AMBA_AUDIO_ENC_AAC_PLUS (AAC Plus) 0x12: AMBA_AUDIO_ENC_PLUS_V2 (AAC Plus V2) 0x20: AMBA_AUDIO_ADPCM (IMA ADPCM) 0x30: AMBA_AUDIO_AC3 0x40: AMBA_AUDIO_MPEG 0x50: AMBA_AUDIO_OPUS 0x60: AMBA_AUDIO_G711A (G711 a-law) 0x61: AMBA_AUDIO_G711U (G711 u-law) 0x62: AMBA_AUDIO_G726A (G726 a-law) 0x63: AMBA_AUDIO_G726U (G726 u-law)
AMBA_AUDIO_ENC_GET_BS_ADDR	**EncGetBsAddr**	typedef UINT8* (*AMBA_AUDIO_ENC_GET_BS_ADDR) (UINT32 *pHdlr); /* Callback function for the encoder to retrieve the write-pointer address */
void	***pEncConfig**	/* CODEC configuration data pointer. */ PCM and AAC encoder configuration are defined in `AmbaAudio.h`.

*Table 2-43. Definition of **AMBA_AUDIO_ENC_SETUP_INFO_s** for Audio API **AmbaAudio_EncTaskSetUp()**.*

2.2.16.2 AmbaAudio_ EncTaskSetUp > AMBA_AUDIO_ENC_SETUP_INFO_s

CODEC configuration data point can be assigned to different CODEC data pointer depend on the “EncType” Audio decoder task provides following types for user setting the decoder.

Type	Encoder configuration data structures
AMBA_AUDIO_PCM	AMBA_AUDIO_PCM_CONFIG_s
AMBA_AUDIO_AAC / AMBA_AUDIO_AAC_PLUS / AMBA_AUDIO_PLUS_V2	AMBA_AUDIO_AACENC_CONFIG_s
AMBA_AUDIO_ADPCM	AMBA_AUDIO_ADPCM_CONFIG_s
AMBA_AUDIO_AC3	AMBA_AUDIO_AC3ENC_CONFIG_s
AMBA_AUDIO_OPUS	AMBA_AUDIO_OPUSENC_CONFIG_s
AMBA_AUDIO_G711A	AMBA_AUDIO_G7XXENC_CONFIG_s

Table 2-44. Encoder Configuration Data Structures for Different Audio Encoder Type.

2.2.16.3 AmbaAudio_ EncTaskSetUp > AMBA_AUDIO_PCM_CONFIG_s

Type	Field	Description
UINT32	**BitsPerSample**	8, 16, 24, or 32-bits per sample
UINT32	**DataFormat**	Endian, Audio_BS_Intel (LSB,MSB), Audio_BS_Motorola (MSB, LSB).
UINT32	**FrameSize**	Samples per frame, eg: 1024 or 2048

Table 2-45. PCM Encoder Configuration Data Structure.

2.2.16.4 AmbaAudio_ EncTaskSetUp > AMBA_AUDIO_AACENC_CONFIG_s

Type	Field	Description
AMBA_AUDIO_AAC_BS_TYPE_e	**BitstreamType**	Bit-stream Type of AAC. AAC_BS_RAW, AAC_BS_ADIF, (Bit-stream with ADIF header) AAC_BS_ADTS, (Bit-stream with ADTS header) AAC_BS_LOAS, (Bit-stream with LOAS header)
UINT32	**Bitrate**	Bitrate of encoded bit-stream (bits/second);

Table 2-46. AAC Encoder Configuration Data Structure.

2.2.16.5 AmbaAudio_ EncTaskSetUp > AMBA_AUDIO_ADPCM_CONFIG_s

Type	Field	Description
UINT32	**AdpcmFrameSize**	Samples Per Block

Table 2-47. ADPCM Encoder Configuration Data Structure.

2.2.16.6 AmbaAudio_ EncTaskSetUp > AMBA_AUDIO_AC3ENC_CONFIG_s

Type	Field	Description
UINT8	**Ac3EncAcmod**	Target bitstream channel mode: AUDIO_CH_MODE_C = 0x01, AUDIO_CH_MODE_L_R = 0x02, AUDIO_CH_MODE_L_C_R = 0x03, AUDIO_CH_MODE_L_R_S = 0x04, AUDIO_CH_MODE_L_C_R_S = 0x05, AUDIO_CH_MODE_L_R_LS_RS = 0x06, AUDIO_CH_MODE_L_C_R_LS_RS = 0x07
UINT32	**Ac3EncBitrate**	Bitrate of encoded bit-stream (bits/second);
UINT8	**Ac3EncAgcEnable**	Auto gain control: 0: Disable 1: Enable
UINT8	**Ac3EncAgcCh2Enable**	TBD

Type	Field	Description
UINT8	**Ac3EncDrcMode**	Dynamic range compression
UINT8	**Ac3EncLfeEnable**	0: No sub woofer 1: Sub woofer exists
UINT8	**Ac3EncLfeFilterEnable**	TBD
UINT8	**Ac3EncTestMode**	TBD
UINT8	**Ac3EncSurroundDelayEnable**	TBD
UINT8	**Ac3EncBsEndian**	Endian of output bitstream

Table 2-48. AC3 Encoder Configuration Data Structure.

2.2.16.7 AmbaAudio_ EncTaskSetUp > AMBA_AUDIO_OPUSENC_CONFIG_s

Type	Field	Description
UINT32	**Bitrate**	Bitrate of encoded bit-stream (bits/second);
UINT32	**FrameSize**	Sample number per frame per channel
UINT8	**BitstreamType**	Decoded bit-stream type (OPUS_BS_RTP or OPUS_BS_RAW)

Table 2-49. OPUS Encoder Configuration Data Structure.

2.2.16.8 AmbaAudio_ EncTaskSetUp > AMBA_AUDIO_G7XXENC_CONFIG_s

Type	Field	Description
INT32	**Rate**	The compression rate for G726

Table 2-50. G7XXENC Encoder Configuration Data Structure.

2.2.17 AmbaAudio_EncTaskStart

API Syntax:

AmbaAudio_EncTaskStart (UINT32 *pHdlr, UINT32 FadeInTime)

Function Description:

- This function is used to initialize the encoder process.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the Encoder resource
UINT32	**FadeInTime**	Fade in time of the encoder (ms)

*Table 2-51. Parameters for Audio API **AmbaAudio_EncTaskStart()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-52. Returns for Audio API **AmbaAudio_EncTaskStart()**.*

Example:

```
UINT32 *pAudioEncCtrl;
….

AmbaAudio_EncTaskStart(pAudioEncCtrl, 10);
```

See Also:

AmbaAudio_EncTaskStop()

2.2.18 AmbaAudio_EncTaskStop

API Syntax:

AmbaAudio_EncTaskStop (UINT32 *pHdlr, UINT32 FadeOutTime)

Function Description:

- This function is used to terminate the encoder process.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the Encoder resource
UINT32	**FadeOutTime**	Fade out time of the encoder (ms)

*Table 2-53. Parameters for Audio API **AmbaAudio_EncTaskStop()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-54. Returns for Audio API **AmbaAudio_EncTaskStop()**.*

Example:

```
UINT32 *pAudioEncCtrl;
….

AmbaAudio_EncTaskStop(pAudioEncCtrl, 10);
```

See Also:

AmbaAudio_EncTaskStart()

2.2.19 AmbaAudio_EncSetVolume

API Syntax:

AmbaAudio_EncSetVolume (UINT32 *pHdlr, UINT32 Volume)

Function Description:

- This function is used to configure volume settings for a specified encoder task.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the Encoder resource
UINT32	**Volume**	Volume of the encoder task (min. 0 ~ max. 64)

*Table 2-55. Parameters for Audio API **AmbaAudio_EncSetVolume()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-56. Returns for Audio API **AmbaAudio_EncSetVolume()**.*

Example:

```
UINT32 *pAudioEncCtrl;
….

AmbaAudio_EncSetVolume(pAudioEncCtrl, 64);
```

See Also:

None

2.2.20 AmbaAudio_OutputCachedSizeQuery

API Syntax:

AmbaAudio_OutputCachedSizeQuery (AMBA_AUDIO_IO_CREATE_INFO_s *pInfo)

Function Description:

- This function is used to query the cached memory size required to generate an audio output resource.

Parameters:

Type	Parameter	Description
AMBA_AUDIO_IO_CREATE_INFO_s	***pInfo**	Audio output task creation information. Please refer to Section 2.2.20.1 below for the definition.

*Table 2-57. Parameters for Audio API **AmbaAudio_OutputCachedSizeQuery()**.*

Returns:

Return	Description
> 0	Required cached memory size

*Table 2-58. Returns for Audio API **AmbaAudio_OutputCachedSizeQuery()**.*

Example:

```
UINT32 *pAudioOutputCtrl;

….....
AMBA_AUDIO_IO_CREATE_INFO_s OutputInfo;
AMBA_AUDIO_BUF_INFO_s OutC, OutNonC;
UINT32 OutputCachedSize, OutputNonCachedSize;
UINT32 *pOutputCachedBase;

OutputInfo.I2sIndex = 0;
OutputInfo.MaxChNum = 2;
OutputInfo.MaxDmaDescNum = 16;
OutputInfo.MaxDmaSize = 256;
OutputInfo.MaxSampleFreq = 48000;

OutputCachedSize = AmbaAudio_OutputCachedSizeQuery(&OutputInfo);
OutputNonCachedSize = AmbaAudio_OutputNonCachedSizeQuery(&OutputInfo);

OutC.MaxSize = OutputCachedSize;
OutC.pHead = pOutputCachedBase;

OutNonC.MaxSize = OutputNonCachedSize;
OutNonC.pHead = pOutputNonCachedBase;

pAudioOutputCtrl = AmbaAudio_OutputCreate(&OutputInfo, &OutC, &OutNonC);
```

See Also:

AmbaAudio_OutputNonCachedSizeQuery()
AmbaAudio_OutputCreate()

2.2.20.1 AmbaAudio_OutputCachedSizeQuery > AMBA_AUDIO_IO_CREATE_INFO_s

Type	Field	Description
UINT32	**MaxSampleFreq**	Maximum supporting sampling frequency of the decoder task
UINT32	**MaxChNum**	Maximum supporting channel number
UINT32	**MaxDmaSize**	Maximum DMA frame size
UINT32	**MaxDmaDescNum**	Max DMA Descriptors number
UINT32	**I2sIndex**	0 or first I2S

*Table 2-59. Definition of **AMBA_AUDIO_IO_CREATE_INFO_s** for Audio API **AmbaAudio_OutputCachedSizeQuery()**.*

2.2.21 AmbaAudio_OutputNonCachedSizeQuery

API Syntax:

AmbaAudio_OutputNonCachedSizeQuery (AMBA_AUDIO_IO_CREATE_INFO_s *pInfo)

Function Description:

- This function is used to query the non-cached memory size required to generate an audio output resource.

Parameters:

Type	Parameter	Description
AMBA_AUDIO_IO_CREATE_INFO_s	***pInfo**	Audio output task creation information. Please refer to Section 2.2.20.1 for the definition.

*Table 2-60. Parameters for Audio API **AmbaAudio_OutputNonCachedSizeQuery()**.*

Returns:

Return	Description
> 0	Required cached memory size

*Table 2-61. Returns for Audio API **AmbaAudio_OutputNonCachedSizeQuery()**.*

Example:

```
UINT32 *pAudioOutputCtrl;

….....
AMBA_AUDIO_IO_CREATE_INFO_s OutputInfo;
AMBA_AUDIO_BUF_INFO_s OutC, OutNonC;
UINT32 OutputCachedSize, OutputNonCachedSize;
UINT32 *pOutputCachedBase;

OutputInfo.I2sIndex = 0;
OutputInfo.MaxChNum = 2;
OutputInfo.MaxDmaDescNum = 16;
OutputInfo.MaxDmaSize = 256;
OutputInfo.MaxSampleFreq = 48000;

OutputCachedSize = AmbaAudio_OutputCachedSizeQuery(&OutputInfo);
OutputNonCachedSize = AmbaAudio_OutputNonCachedSizeQuery(&OutputInfo);

OutC.MaxSize = OutputCachedSize;
OutC.pHead = pOutputCachedBase;

OutNonC.MaxSize = OutputNonCachedSize;
OutNonC.pHead = pOutputNonCachedBase;

pAudioOutputCtrl = AmbaAudio_OutputCreate(&OutputInfo, &OutC, &OutNonC);
```

See Also:

AmbaAudio_OutputCachedSizeQuery()
AmbaAudio_OutputCreate()

2.2.22 AmbaAudio_OutputCreate

API Syntax:

AmbaAudio_OutputCreate (AMBA_AUDIO_IO_CREATE_INFO_s *pInfo, AMBA_AUDIO_BUF_INFO_s *pCachedInfo, AMBA_AUDIO_BUF_INFO_s *pNonCachedInfo)

Function Description:

- This function is used to generate resources required by an audio output task.

Parameters:

Type	Parameter	Description
AMBA_AUDIO_IO_CREATE_INFO_s	***pInfo**	Audio output task creation information. Please refer to Section 2.2.20.1 below for the definition.
AMBA_AUDIO_BUF_INFO_s	***pCachedInfo**	Cached memory information. Please refer to Section 2.2.22.1 below for the definition.
AMBA_AUDIO_BUF_INFO_s	***pNonCachedInfo**	Non-cached memory information. Please refer to Section 2.2.22.1 below for the definition.

*Table 2-62. Parameters for Audio API **AmbaAudio_OutputCreate()**.*

Returns:

Return	Description
Address	Pointer of the audio output resource handler
0xFFFFFFFF	Failure

*Table 2-63. Returns for Audio API **AmbaAudio_OutputCreate()**.*

Example:

```
UINT32 *pAudioOutputCtrl;

…....
AMBA_AUDIO_IO_CREATE_INFO_s OutputInfo;
AMBA_AUDIO_BUF_INFO_s OutC, OutNonC;
UINT32 OutputCachedSize, OutputNonCachedSize;
UINT32 *pOutputCachedBase;

OutputInfo.I2sIndex = 0;
OutputInfo.MaxChNum = 2;
OutputInfo.MaxDmaDescNum = 16;
OutputInfo.MaxDmaSize = 256;
OutputInfo.MaxSampleFreq = 48000;

OutputCachedSize = AmbaAudio_OutputCachedSizeQuery(&OutputInfo);
OutputNonCachedSize = AmbaAudio_OutputNonCachedSizeQuery(&OutputInfo);

OutC.MaxSize = OutputCachedSize;
OutC.pHead = pOutputCachedBase;
```

```
OutNonC.MaxSize = OutputNonCachedSize;
OutNonC.pHead = pOutputNonCachedBase;

pAudioOutputCtrl = AmbaAudio_OutputCreate(&OutputInfo, &OutC, &OutNonC);
```

See Also:

AmbaAudio_OutputCachedSizeQuery()
AmbaAudio_OutputNonCachedSizeQuery()

2.2.22.1 AmbaAudio_OutputCreate > AMBA_AUDIO_BUF_INFO_s

Type	Field	Description
UINT32	***pHead**	Memory head
UINT32	**MaxSize**	Maximum memory size

*Table 2-64. Definition of **AMBA_AUDIO_BUF_INFO_s** for Audio API **AmbaAudio_OutputCreate()**.*

2.2.23 AmbaAudio_OutputDelete

API Syntax:

AmbaAudio_OutputDelete (UINT32 *pHdlr)

Function Description:

- This function is used to delete a specified audio output resource.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the audio output resource

*Table 2-65. Parameters for Audio API **AmbaAudio_OutputDelete()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-66. Returns for Audio API **AmbaAudio_OutputDelete()**.*

Example:

```
UINT32 *pAudioOutputCtrl;

...
AmbaAudio_OutputDelete(pAudioOutputCtrl);
```

See Also:

AmbaAudio_OutputCreate()

2.2.24 AmbaAudio_OutputTaskCreate

API Syntax:

AmbaAudio_OutputTaskCreate (UINT32 *pHdlr, INT32 Priority, UINT32 CoreExlusionMap)

Function Description:

- This function is used to generate an audio output task.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the audio output resource
UINT32	**Priority**	Task priority of the audio output task
UINT32	**CoreExlusionMap**	The core selection of this task

*Table 2-67. Parameters for Audio API **AmbaAudio_OutputTaskCreate()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-68. Returns for Audio API **AmbaAudio_OutputTaskCreate()**.*

Example:

```
UINT32 *pAudioOutputCtrl;

…..
AmbaAudio_OutputTaskCreate(pAudioOutputCtrl, 32, 0);
```

See Also:

AmbaAudio_OutputTaskDelete()

2.2.25 AmbaAudio_OutputTaskDelete

API Syntax:

AmbaAudio_OutputTaskDelete (UINT32 *pHdlr)

Function Description:

- This function is used to delete a specified audio output task.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the audio output resource

*Table 2-69. Parameters for Audio API **AmbaAudio_OutputTaskDelete()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-70. Returns for Audio API **AmbaAudio_OutputTaskDelete()**.*

Example:

```
UINT32 *pAudioOutputCtrl;

…..
AmbaAudio_OutputTaskDelete(pAudioOutputCtrl)
```

See Also:

AmbaAudio_OutputTaskCreate()

2.2.26 AmbaAudio_OutputTaskStart

API Syntax:

AmbaAudio_OutputTaskStart (UINT32 *pHdlr)

Function Description:

- This function is used to initialize the audio output process.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the audio output resource

*Table 2-71. Parameters for Audio API **AmbaAudio_OutputTaskStart()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-72. Returns for Audio API **AmbaAudio_OutputTaskStart()**.*

Example:

```
UINT32 *pAudioOutputCtrl;
….

AmbaAudio_OutputTaskStart(pAudioOutputCtrl);
```

See Also:

AmbaAudio_OutputTaskStop()

2.2.27 AmbaAudio_OutputTaskStop

API Syntax:

AmbaAudio_OutputTaskStop (UINT32 *pHdlr)

Function Description:

- This function is used to terminate the audio output process.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the audio output resource

*Table 2-73. Parameters for Audio API **AmbaAudio_OutputTaskStop()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-74. Returns for Audio API **AmbaAudio_OutputTaskStop()**.*

Example:

```
UINT32 *pAudioOutputCtrl;
….

AmbaAudio_OutputTaskStop(pAudioOutputCtrl);
```

See Also:

AmbaAudio_OutputTaskStart()

2.2.28 AmbaAudio_OutputSetVolume

API Syntax:

AmbaAudio_OutputSetVolume (UINT32 *pHdlr, UINT32 Volume)

Function Description:

- This function is used to configure volume settings for a specified output task.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the audio output resource
UINT32	**Volume**	Volume of the audio output task (min. 0 ~ max. 64)

*Table 2-75. Parameters for Audio API **AmbaAudio_OutputSetVolume()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-76. Returns for Audio API **AmbaAudio_OutputSetVolume()**.*

Example:

```
UINT32 *pAudioOutputCtrl;
….

AmbaAudio_OutputSetVolume(pAudioOutputCtrl, 64);
```

See Also:

None

2.2.29 AmbaAudio_InputCachedSizeQuery

API Syntax:

AmbaAudio_InputCachedSizeQuery (AMBA_AUDIO_IO_CREATE_INFO_s *pInfo)

Function Description:

- This function is used to query the cached memory size required to generate an audio input resource.

Parameters:

Type	Parameter	Description
AMBA_AUDIO_IO_CREATE_INFO_s	***pInfo**	Audio input task creation information. Please refer to Section 2.2.20.1 for the definition.

*Table 2-77. Parameters for Audio API **AmbaAudio_InputCachedSizeQuery()**.*

Returns:

Return	Description
> 0	Required cached memory size

*Table 2-78. Returns for Audio API **AmbaAudio_InputCachedSizeQuery()**.*

Example:

```
UINT32 *pAudioInputCtrl;

…....
AMBA_AUDIO_IO_CREATE_INFO_s InputInfo;
AMBA_AUDIO_BUF_INFO_s InC, InNonC;
UINT32 InputCachedSize, InputNonCachedSize;
UINT32 *pInputCachedBase;

InputInfo.I2sIndex = 0;
InputInfo.MaxChNum = 2;
InputInfo.MaxDmaDescNum = 16;
InputInfo.MaxDmaSize = 1024;
InputInfo.MaxSampleFreq = 48000;

InputCachedSize = AmbaAudio_InputCachedSizeQuery(&InputInfo);
InputNonCachedSize = AmbaAudio_InputNonCachedSizeQuery(&InputInfo);

InC.MaxSize = InputCachedSize;
InC.pHead = pInputCachedBase;

InNonC.MaxSize = InputNonCachedSize;
InNonC.pHead = pInputNonCachedBase;

pAudioInputCtrl = AmbaAudio_InputCreate(&InputInfo, &InC, &InNonC);
```

See Also:

AmbaAudio_InputNonCachedSizeQuery()
AmbaAudio_InputCreate()

2.2.30 AmbaAudio_InputNonCachedSizeQuery

API Syntax:

AmbaAudio_InputNonCachedSizeQuery (AMBA_AUDIO_IO_CREATE_INFO_s *pInfo)

Function Description:

- This function is used to query the non-cached memory size required to generate an audio input resource.

Parameters:

Type	Parameter	Description
AMBA_AUDIO_IO_CREATE_INFO_s	***pInfo**	Audio input task creation information. Please refer to Section 2.2.20.1 for the definition.

*Table 2-79. Parameters for Audio API **AmbaAudio_InputNonCachedSizeQuery()**.*

Returns:

Return	Description
> 0	Required cached memory size

*Table 2-80. Returns for Audio API **AmbaAudio_InputNonCachedSizeQuery()**.*

Example:

```
UINT32 *pAudioInputCtrl;

….....
AMBA_AUDIO_IO_CREATE_INFO_s InputInfo;
AMBA_AUDIO_BUF_INFO_s InC, InNonC;
UINT32 InputCachedSize, InputNonCachedSize;
UINT32 *pInputCachedBase;

InputInfo.I2sIndex = 0;
InputInfo.MaxChNum = 2;
InputInfo.MaxDmaDescNum = 16;
InputInfo.MaxDmaSize = 256;
InputInfo.MaxSampleFreq = 48000;

InputCachedSize = AmbaAudio_InputCachedSizeQuery(&InputInfo);
InputNonCachedSize = AmbaAudio_InputNonCachedSizeQuery(&InputInfo);

InC.MaxSize = InputCachedSize;
InC.pHead = pInputCachedBase;

InNonC.MaxSize = InputNonCachedSize;
InNonC.pHead = pInputNonCachedBase;

pAudioInputCtrl = AmbaAudio_InputCreate(&InputInfo, &InC, &InNonC);
```

See Also:

AmbaAudio_InputCachedSizeQuery()
AmbaAudio_InputCreate()

2.2.31 AmbaAudio_InputCreate

API Syntax:

AmbaAudio_InputCreate (AMBA_AUDIO_IO_CREATE_INFO_s *pInfo, AMBA_AUDIO_BUF_INFO_s *pCachedInfo, AMBA_AUDIO_BUF_INFO_s *pNonCachedInfo)

Function Description:

- This function is used to generate audio resources required for a specified input task.

Parameters:

Type	Parameter	Description
AMBA_AUDIO_IO_CREATE_INFO_s	***pInfo**	Audio input task creation information. Please refer to Section 2.2.20.1 for the definition.
AMBA_AUDIO_BUF_INFO_s	***pCachedInfo**	Cached memory information. Please refer to Section 2.2.22.1 for the definition.
AMBA_AUDIO_BUF_INFO_s	***pNonCachedInfo**	Non-cached memory information. Please refer to Section 2.2.22.1 for the definition.

*Table 2-81. Parameters for Audio API **AmbaAudio_InputCreate()**.*

Returns:

Return	Description
Address	Pointer of the audio input resource handler.
0xFFFFFFFF	Failure

*Table 2-82. Returns for Audio API **AmbaAudio_InputCreate()**.*

Example:

```
UINT32 *pAudioInputCtrl;

…....
AMBA_AUDIO_IO_CREATE_INFO_s InputInfo;
AMBA_AUDIO_BUF_INFO_s InC, InNonC;
UINT32 InputCachedSize, InputNonCachedSize;
UINT32 *pInputCachedBase;

InputInfo.I2sIndex = 0;
InputInfo.MaxChNum = 2;
InputInfo.MaxDmaDescNum = 16;
InputInfo.MaxDmaSize = 256;
InputInfo.MaxSampleFreq = 48000;

InputCachedSize = AmbaAudio_InputCachedSizeQuery(&InputInfo);
InputNonCachedSize = AmbaAudio_InputNonCachedSizeQuery(&InputInfo);
```

```
InC.MaxSize = InputCachedSize;
InC.pHead = pInputCachedBase;

InNonC.MaxSize = InputNonCachedSize;
InNonC.pHead = pInputNonCachedBase;

pAudioInputCtrl = AmbaAudio_InputCreate(&InputInfo, &InC, &InNonC);
```

See Also:

AmbaAudio_InputCachedSizeQuery()
AmbaAudio_InputNonCachedSizeQuery()

2.2.32 AmbaAudio_InputDelete

API Syntax:

AmbaAudio_InputDelete (UINT32 *pHdlr)

Function Description:

- This function is used to delete a specified audio input resource.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the audio input resource

*Table 2-83. Parameters for Audio API **AmbaAudio_InputDelete()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-84. Returns for Audio API **AmbaAudio_InputDelete()**.*

Example:

```
UINT32 *pAudioInputCtrl;

...
AmbaAudio_InputDelete(pAudioInputCtrl);
```

See Also:

AmbaAudio_InputCreate()

2.2.33 AmbaAudio_InputTaskCreate

API Syntax:

AmbaAudio_InputTaskCreate (UINT32 *pHdlr, INT32 Priority, UINT32 CoreExlusionMap)

Function Description:

- This function is used to generate an audio input task.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the audio input resource
INT32	**Priority**	Task priority of the audio input task
UINT32	**CoreExlusionMap**	The core selection of this task

*Table 2-85. Parameters for Audio API **AmbaAudio_InputTaskCreate()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-86. Returns for Audio API **AmbaAudio_InputTaskCreate()**.*

Example:

```
UINT32 *pAudioInputCtrl;

…..
AmbaAudio_InputTaskCreate(pAudioInputCtrl, 32, 0);
```

See Also:

AmbaAudio_InputTaskDelete()

2.2.34 AmbaAudio_InputTaskDelete

API Syntax:

AmbaAudio_InputTaskDelete (UINT32 *pHdlr)

Function Description:

- This function is used to delete a specified audio input task.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the audio input resource

*Table 2-87. Parameters for Audio API **AmbaAudio_InputTaskDelete()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-88. Returns for Audio API **AmbaAudio_InputTaskDelete()**.*

Example:

```
UINT32 *pAudioInputCtrl;

…..
AmbaAudio_InputTaskDelete(pAudioInputCtrl);
```

See Also:

AmbaAudio_InputTaskCreate()

2.2.35 AmbaAudio_InputTaskStart

API Syntax:

AmbaAudio_InputTaskStart (UINT32 *pHdlr)

Function Description:

- This function is used to initialize the audio input process.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the audio input resource

*Table 2-89. Parameters for Audio API **AmbaAudio_InputTaskStart()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-90. Returns for Audio API **AmbaAudio_InputTaskStart()**.*

Example:

```
UINT32 *pAudioInputCtrl;
….

AmbaAudio_InputTaskStart(pAudioInputCtrl);
```

See Also:

AmbaAudio_InputTaskStop()

2.2.36 AmbaAudio_InputTaskStop

API Syntax:

AmbaAudio_InputTaskStop (UINT32 *pHdlr)

Function Description:

- This function is used to terminate the audio input process.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the audio input resource

*Table 2-91. Parameters for Audio API **AmbaAudio_InputTaskStop()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-92. Returns for Audio API **AmbaAudio_InputTaskStop()**.*

Example:

```
UINT32 *pAudioInputCtrl;
….

AmbaAudio_InputTaskStop(pAudioInputCtrl);
```

See Also:

AmbaAudio_InputTaskStart()

2.2.37 AmbaAudio_InputSetVolume

API Syntax:

AmbaAudio_InputSetVolume (UINT32 *pHdlr, UINT32 Volume)

Function Description:

- This function is used to configure volume settings for a specified input task.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the audio input resource
UINT32	**Volume**	Volume of the audio input task (min. 0 ~ max. 64)

*Table 2-93. Parameters for Audio API **AmbaAudio_InputSetVolume()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-94. Returns for Audio API **AmbaAudio_InputSetVolume()**.*

Example:

```
UINT32 *pAudioInputCtrl;
….

AmbaAudio_InputSetVolume(pAudioInputCtrl, 64);
```

See Also:

None

2.2.38 AmbaAudio_BufferSizeQuery

API Syntax:

AmbaAudio_BufferSizeQuery (AMBA_ABU_CREATE_INFO_s *pInfo)

Function Description:

- This function is used to query the memory size required to generate an audio buffer.

Parameters:

Type	Parameter	Description
AMBA_ABU_CREATE_INFO_s	***pInfo**	Audio buffer unit creation information. Please refer to Section 2.2.38.1 below for the definition.

*Table 2-95. Parameters for Audio API **AmbaAudio_BufferSizeQuery()**.*

Returns:

Return	Description
> 0	Required memory size

*Table 2-96. Returns for Audio API **AmbaAudio_BufferSizeQuery()**.*

Example:

```
AMBA_ABU_CREATE_INFO_s AbuInfo;
UINT32 AbuSize;
….

AbuInfo.MaxSampleFreq = 48000;
AbuInfo.MaxChNum = 2;
AbuInfo.MaxChunkNum = 16;
AbuSize = AmbaAudio_BufferSizeQuery(&AbuInfo);
```

See Also:

AmbaAudio_BufferCreate()

2.2.38.1 AmbaAudio_ BufferSizeQuery > AMBA_ABU_CREATE_INFO_s

Type	Field	Description
UINT32	**MaxSampleFreq**	Maximum supporting sampling frequency of the decoder task.
UINT32	**MaxChNum**	Maximum supporting channel number
UINT32	**MaxChunkNum**	Maximum chunk number of ABU

*Table 2-97. Definition of **AMBA_ABU_CREATE_INFO_s** for Audio API **AmbaAudio_BufferSizeQuery()**.*

2.2.39 AmbaAudio_BufferCreate

API Syntax:

AmbaAudio_BufferCreate (AMBA_ABU_CREATE_INFO_s *pInfo, UINT32 *pAddr, UINT32 BufferSize)

Function Description:

- This function is used to generate resources required by the audio buffer.

Parameters:

Type	Parameter	Description
AMBA_ABU_CREATE_INFO_s	***pInfo**	Audio buffer unit creation information. Please refer to Section 2.2.38.1 for the definition.
UINT32	***pAddr**	Pointer of the memory prepared for audio buffer resource
UINT32	**BufferSize**	Size of the memory prepared for audio buffer resource

*Table 2-98. Parameters for Audio API **AmbaAudio_BufferCreate()**.*

Returns:

Return	Description
Address	Pointer of the audio buffer resource handler.
0xFFFFFFFF	Failure

*Table 2-99. Returns for Audio API **AmbaAudio_BufferCreate()**.*

Example:

```
AMBA_ABU_CREATE_INFO_s AbuInfo;
UINT32 AbuSize;
UINT32 *pAbuCtrl;
UINT32 *pAbuAddr

….

AbuInfo.MaxSampleFreq = 48000;
AbuInfo.MaxChNum = 2;
AbuInfo.MaxChunkNum = 16;
AbuSize = AmbaAudio_BufferSizeQuery(&AbuInfo);

/* Allocate AbuSize buffer with *pAbuAddr */

pAbuCtrl = AmbaAudio_BufferCreate(&AbuInfo, pAbuAddr, AbuSize);
```

See Also:

AmbaAudio_BufferSizeQuery()

2.2.40 AmbaAudio_BufferDelete

API Syntax:

AmbaAudio_BufferDelete (UINT32 *pHandle)

Function Description:

- This function is used to delete a specified audio buffer resource.

Parameters:

Type	Parameter	Description
UINT32	***pHandle**	Handle of the audio buffer resource

*Table 2-100. Parameters for Audio API **AmbaAudio_BufferDelete()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-101. Returns for Audio API **AmbaAudio_BufferDelete()**.*

Example:

```
UINT32 *pAbuCtrl;

...
AmbaAudio_BufferDelete(pAbuCtrl);
```

See Also:

AmbaAudio_BufferCreate()

2.2.41 AmbaAudio_BufferReset

API Syntax:

AmbaAudio_BufferReset (UINT32 *pHandle)

Function Description:

- This function is used to reset a specified audio buffer resource.

Parameters:

Type	Parameter	Description
UINT32	***pHandle**	Handle of the audio buffer resource

*Table 2-102. Parameters for Audio API **AmbaAudio_BufferReset()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-103. Returns for Audio API **AmbaAudio_BufferReset()**.*

Example:

```
UINT32 *pAbuCtrl;

...
AmbaAudio_BufferReset(pAbuCtrl);
```

See Also:

AmbaAudio_BufferCreate()

2.2.42 AmbaAudio_BufferOpenSrcIoNode

API Syntax:

AmbaAudio_BufferOpenSrcIoNode (UINT32 *pHandle)

Function Description:

- This function is used to open the source I/O node of a specified audio buffer.

Parameters:

Type	Parameter	Description
UINT32	***pHandle**	Handle of the audio buffer resource

*Table 2-104. Parameters for Audio API **AmbaAudio_BufferOpenSrcIoNode()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-105. Returns for Audio API **AmbaAudio_BufferOpenSrcIoNode()**.*

Example:

```
UINT32 *pAbuCtrl;

...
AmbaAudio_BufferOpenSrcIoNode(pAbuCtrl);
```

See Also:

None

2.2.43 AmbaAudio_BufferOpenDstIoNode

API Syntax:

AmbaAudio_BufferOpenDstIoNode (UINT32 *pHandle)

Function Description:

- This function is used to open the destination I/O node of a specified audio buffer.

Parameters:

Type	Parameter	Description
UINT32	***pHandle**	Handle of the audio buffer resource

*Table 2-106. Parameters for Audio API **AmbaAudio_BufferOpenDstIoNode()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-107. Returns for Audio API **AmbaAudio_BufferOpenDstIoNode()**.*

Example:

```
UINT32 *pAbuCtrl;

...
AmbaAudio_BufferOpenDstIoNode(pAbuCtrl);
```

See Also:

None

2.2.44 AmbaAudio_Combine

API Syntax:

AmbaAudio_Combine (AMBA_AUDIO_COMBINE_INFO_s *pInfo)

Function Description:

- This function is used to combine audio resources, including the source task, the audio buffer unit, and the destination task.

Parameters:

Type	Parameter	Description
AMBA_AUDIO_COMBINE_INFO_s	***pInfo**	Audio resource combination information. Please refer to Section 2.2.44.1 below for the definition.

*Table 2-108. Parameters for Audio API **AmbaAudio_Combine()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-109. Returns for Audio API **AmbaAudio_Combine()**.*

Example:

```
AMBA_AUDIO_COMBINE_INFO_s Combine;


Combine.pAbu = pAbuCtrl;
Combine.pSrcApu = pAudioDecCtrl;
Combine.pDstApu = pAudioOutputCtrl;

AmbaAudio_Combine(&Combine)
```

See Also:

AmbaAudio_Detach()

2.2.44.1 AmbaAudio_Combine > AMBA_AUDIO_COMBINE_INFO_s

Type	Field	Description
UINT32	***pSrcApu**	Source audio resource handler
UINT32	***pDstApu**	Destination audio resource handler
UINT32	***pAbu**	Handle of the audio buffer resource

*Table 2-110. Definition of **AMBA_AUDIO_COMBINE_INFO_s** for Audio API **AmbaAudio_Combine()**.*

2.2.45 AmbaAudio_Detach

API Syntax:

AmbaAudio_Detach (AMBA_AUDIO_COMBINE_INFO_s *pInfo)

Function Description:

- This function is used to separate previously combined audio resources, such as the source task, the audio buffer unit, and the destination task.

Parameters:

Type	Parameter	Description
AMBA_AUDIO_COMBINE_INFO_s	***pInfo**	Audio resource combination information. Please refer to Section 2.2.44.1 for the definition.

*Table 2-111. Parameters for Audio API **AmbaAudio_Detach()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-112. Returns for Audio API **AmbaAudio_Detach()**.*

Example:

```
AMBA_AUDIO_COMBINE_INFO_s Combine;


Combine.pAbu = pAbuCtrl;
Combine.pSrcApu = pAudioDecCtrl;
Combine.pDstApu = pAudioOutputCtrl;

AmbaAudio_Detach(&Combine);
```

See Also:

AmbaAudio_Combine ()

2.2.46 AmbaAudio_EventHandlerCtrlConfig

API Syntax:

AmbaAudio_EventHandlerCtrlConfig (UINT32 *pHdlr, AMBA_AUDIO_EVENT_ID_e EventID, int MaxNumHandler, AMBA_AUDIO_EVENT_HANDLER_f *pEventHandlers)

Function Description:

- This function is used to configure control settings for the audio event handler.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource
AMBA_AUDIO_EVENT_ID_e	**EventID**	Event IDs are defined in `AmbaAudio.h`.
int	**MaxNumHandler**	Maximum number of Handlers
AMBA_AUDIO_EVENT_HANDLER_f	***pEventHandlers**	Pointer to the Event Handlers: `typedef int (*AMBA_AUDIO_EVENT_HANDLER_f) (void *pEventData);`

*Table 2-113. Parameters for Audio API **AmbaAudio_EventHandlerCtrlConfig()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-114. Returns for Audio API **AmbaAudio_EventHandlerCtrlConfig()**.*

Example:

```
AMBA_AUDIO_EVENT_HANDLER_f AmbaAudioDecAbuReadyEntries[1];


….. .
AmbaAudio_EventHandlerCtrlConfig(pAudioDecCtrl,

                                 AMBA_AUDIO_EVENT_ID_DECODE_DATA_READY,
                                 1, AmbaAudioDecAbuReadyEntries);
```

See Also:

AmbaAudio_RegisterEventHandler()

2.2.47 AmbaAudio_RegisterEventHandler

API Syntax:

AmbaAudio_RegisterEventHandler (UINT32 *pHdlr, AMBA_AUDIO_EVENT_ID_e EventID, AMBA_AUDIO_EVENT_HANDLER_f EventHandlers)

Function Description:

- This function is used to register a specified audio event handler.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource
AMBA_AUDIO_EVENT_ID_e	**EventID**	Event IDs are defined in `AmbaAudio.h`.
AMBA_AUDIO_EVENT_HANDLER_f	**EventHandlers**	Event Handlers: `typedef int (*AMBA_AUDIO_EVENT_HANDLER_f)(void *pEventData);`

*Table 2-115. Parameters for Audio API **AmbaAudio_RegisterEventHandler()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-116. Returns for Audio API **AmbaAudio_RegisterEventHandler()**.*

Example:

```
int AmbaAudioDecAbuReadyCB(void *pEventData);


…...
AmbaAudio_RegisterEventHandler(pAudioDecCtrl, AMBA_AUDIO_EVENT_ID_DECODE_DATA_
READY,
                  AmbaAudioDecAbuReadyCB);
```

See Also:

AmbaAudio_UnRegisterEventHandler()

2.2.48 AmbaAudio_UnRegisterEventHandler

API Syntax:

AmbaAudio_UnRegisterEventHandler (UINT32 *pHdlr, AMBA_AUDIO_EVENT_ID_e EventID, AMBA_AUDIO_EVENT_HANDLER_f EventHandlers)

Function Description:

- This function is used to de-register a specified audio event handler.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource
AMBA_AUDIO_EVENT_ID_e	**EventID**	Event IDs are defined in `AmbaAudio.h`.
AMBA_AUDIO_EVENT_HANDLER_f	**EventHandlers**	Event Handlers: `typedef int (*AMBA_AUDIO_EVENT_HANDLER_f)(void *pEventData);`

*Table 2-117. Parameters for Audio API **AmbaAudio_UnRegisterEventHandler()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-118. Returns for Audio API **AmbaAudio_UnRegisterEventHandler()**.*

Example:

```
int AmbaAudioDecAbuReadyCB(void *pEventData);


…...
AmbaAudio_UnRegisterEventHandler(pAudioDecCtrl,
          AMBA_AUDIO_EVENT_ID_DECODE_DATA_READY, AmbaAudioDecAbuReadyCB);
```

See Also:

AmbaAudio_RegisterEventHandler()

2.2.49 AmbaAudio_EventHandlerCtrlReset

API Syntax:

AmbaAudio_EventHandlerCtrlReset (UINT32 *pHdlr)

Function Description:

- This function is used to reset a specified audio event handler.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource

*Table 2-119. Parameters for Audio API **AmbaAudio_EventHandlerCtrlReset()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-120. Returns for Audio API **AmbaAudio_EventHandlerCtrlReset()**.*

Example:

```
AmbaAudio_EventHandlerCtrlReset(pAudioDecCtrl);
```

See Also:

AmbaAudio_EventHandlerCtrlConfig()

2.2.50 AmbaAudio_OutputPluginEffectInstall

API Syntax:

AmbaAudio_OutputPluginEffectInstall (UINT32 *pHdlr, UINT32 Id, AMBA_AUDIO_PLUGIN_EFFECT_CS_s *pCs)

Function Description:

- This function is used to install output plug-in effect.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource
UINT32	**Id**	Effect ID
AMBA_AUDIO_PLUGIN_EFFECT_CS_s	***pCs**	Plug-in effect control structure pointer. Please refer to Section 2.2.50.1 for definition.

*Table 2-121. Parameters for Audio API **AmbaAudio_OutputPluginEffectInstall()**.*

Returns:

None

Example:

```
UINT32 *pAudioOutputCtrl;
UINT32 Id = 5;
AMBA_AUDIO_PLUGIN_EFFECT_CS_s Cs;

Cs.self = &EffectSelf;
Cs.setup = NULL;
Cs.report = EffectReport;
Cs.proc = EffectProc;
…
AmbaAudio_OutputPluginEffectInstall(pAudioOutputCtrl, Id, &Cs);
```

See Also:

AmbaAudio_InputPluginEffectInstall()

2.2.50.1 AmbaAudio_OutputPluginEffectInstall > AMBA_AUDIO_PLUGIN_EFFECT_CS_s

Type	Parameter	Description
int	***src**	**src** is a pointer points to the start address of source pcm buffer.
int	***dest**	**dest** is a pointer points to the start address of destination pcm buffer.
UINT32	**src_ch**	ch is the total channel number of source pcm buffer.
UINT32	**dest_ch**	ch is the total channel number of destination pcm buffer.
UINT32	**src_size**	size is the sample number of one channel of source pcm buffer.
UINT32	**dest_size**	size is the sample number of one channel of destination pcm buffer.
void	**(*setup)(struct _AMBA_AUDIO_PLUGIN_EFFECT_CS_s_ *)**	Start address of “setup” API provided by effect library
void	**(*proc)(struct _AMBA_AUDIO_PLUGIN_EFFECT_CS_s_ *)**	Start address of “proc” API provided by effect library
void	**(*report)(struct _AMBA_AUDIO_PLUGIN_EFFECT_CS_s_ *)**	Start address of “report” callback provided by effect library, to report the effect lib status.
UINT32	**size_of_self**	Size of self-control structure of an effect library.
void	***self**	A pointer to self-control structure of an effect, more information can be contained.
UINT32	**dest_auto_assign**	**dest_auto_assign** is a flag to automatically assign the start address of destination pcm buffer, the start address of destination will be the same as source if set to 1.
UINT32	**dest_ch_auto_assign**	**dest_ch_auto_assign** is a flag to automatically assign the total channel number of source and destination pcm buffer, the channel number of destination will be the same as source if set to 1.
UINT32	**dest_size_auto_assign**	**dest_size_auto_assign** is a flag to automatically assign the sample number of one channel of source and destination pcm buffer, the sample number of one channel of destination will be the same as source if set to 1.

*Table 2-122. Definition of **AMBA_AUDIO_PLUGIN_EFFECT_CS_s** for Audio API **AmbaAudio_OutputPluginEffectInstall()**.*

2.2.51 AmbaAudio_OutputPluginEffectEnable

API Syntax:

AmbaAudio_OutputPluginEffectEnable (UINT32 *pHdlr, UINT32 Id)

Function Description:

- This function is used to enable Output plug-in effect.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource
UINT32	**Id**	Effect ID

*Table 2-123. Parameters for Audio API **AmbaAudio_OutputPluginEffectEnable()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-124. Returns for Audio API **AmbaAudio_OutputPluginEffectEnable()**.*

Example:

```
UINT32 *pAudioOutputCtrl;
UINT32 Id = 5;

…
AmbaAudio_OutputPluginEffectEnable(pAudioOutputCtrl, Id);
```

See Also:

AmbaAudio_InputPluginEffectEnable()

2.2.52 AmbaAudio_OutputPluginEffectDisable

API Syntax:

AmbaAudio_OutputPluginEffectDisable (UINT32 *pHdlr, UINT32 Id)

Function Description:

- This function is used to disable Output plug-in effect.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource
UINT32	**Id**	Effect ID

*Table 2-125. Parameters for Audio API **AmbaAudio_OutputPluginEffectDisable()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-126. Returns for Audio API **AmbaAudio_OutputPluginEffectDisable()**.*

Example:

```
UINT32 *pAudioInputCtrl;
UINT32 Id = 5;

…
AmbaAudio_InputPluginEffectEnable(pAudioOutputCtrl, Id);
```

See Also:

AmbaAudio_InputPluginEffectEnable()

2.2.53 AmbaAudio_OutputPluginEffectUpdate

API Syntax:

AmbaAudio_OutputPluginEffectUpdate (UINT32 *pHdlr, UINT32 Id, AMBA_AUDIO_PLUGIN_EFFECT_CS_s *pCsShadow)

Function Description:

- This function is used to update Output plug-in effect.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource
UINT32	**Id**	Effect ID
AMBA_AUDIO_PLUGIN_EFFECT_CS_s	***pCsShadow**	Plug-in effect control structure pointer. Please refer to Section 2.2.50.1 for definition.

*Table 2-127. Parameters for Audio API **AmbaAudio_OutputPluginEffectUpdate()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-128. Returns for Audio API **AmbaAudio_OutputPluginEffectUpdate()**.*

Example:

```
UINT32 *pAudioOutputCtrl;
UINT32 Id = 5;
AMBA_AUDIO_PLUGIN_EFFECT_CS_s CsShadow;

CsShadow.self = &EffectSelf;
CsShadow.setup = NULL;
CsShadow.report = EffectReport;
CsShadow.proc = EffectProc;
…
AmbaAudio_OutputPluginEffectUpdate(pAudioOutputCtrl, Id, &CsShadow);
```

See Also:

AmbaAudio_InputPluginEffectUpdate()

2.2.54 AmbaAudio_InputPluginEffectInstall

API Syntax:

AmbaAudio_InputPluginEffectInstall (UINT32 *pHdlr, UINT32 Id, AMBA_AUDIO_PLUGIN_EFFECT_CS_s *pCs)

Function Description:

- This function is used to install input plug-in effect.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource
UINT32	**Id**	Effect ID
AMBA_AUDIO_PLUGIN_EFFECT_CS_s	***pCs**	Plug-in effect control structure pointer. Please refer to Section 2.2.50.1 for definition.

*Table 2-129. Parameters for Audio API **AmbaAudio_InputPluginEffectInstall()**.*

Returns:

None

Example:

```
UINT32 *pAudioInputCtrl;
UINT32 Id = 5;
AMBA_AUDIO_PLUGIN_EFFECT_CS_s Cs;

Cs.self = &EffectSelf;
Cs.setup = NULL;
Cs.report = EffectReport;
Cs.proc = EffectProc;
…
AmbaAudio_InputPluginEffectInstall(pAudioInputCtrl, Id, &Cs);
```

See Also:

AmbaAudio_OutputPluginEffectInstall()

2.2.55 AmbaAudio_InputPluginEffectEnable

API Syntax:

AmbaAudio_InputPluginEffectEnable (UINT32 *pHdlr, UINT32 Id)

Function Description:

- This function is used to enable input plug-in effect.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource
UINT32	**Id**	Effect ID

*Table 2-130. Parameters for Audio API **AmbaAudio_InputPluginEffectEnable()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-131. Returns for Audio API **AmbaAudio_InputPluginEffectEnable()**.*

Example:

```
UINT32 *pAudioInputCtrl;
UINT32 Id = 5;

…
AmbaAudio_InputPluginEffectDisable(pAudioOutputCtrl, Id);
```

See Also:

AmbaAudio_OutputPluginEffectDisable()

2.2.56 AmbaAudio_InputPluginEffectDisable

API Syntax:

AmbaAudio_InputPluginEffectDisable (UINT32 *pHdlr, UINT32 Id)

Function Description:

- This function is used to disable input plug-in effect.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource
UINT32	**Id**	Effect ID

*Table 2-132. Parameters for Audio API **AmbaAudio_InputPluginEffectDisable()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-133. Returns for Audio API **AmbaAudio_InputPluginEffectDisable()**.*

Example:

```
UINT32 *pAudioInputCtrl;
UINT32 Id = 5;

…
AmbaAudio_InputPluginEffectDisable(pAudioOutputCtrl, Id);
```

See Also:

AmbaAudio_OutputPluginEffectDisable()

2.2.57 AmbaAudio_InputPluginEffectUpdate

API Syntax:

AmbaAudio_InputPluginEffectUpdate (UINT32 *pHdlr, UINT32 Id, AMBA_AUDIO_PLUGIN_EFFECT_CS_s *pCsShadow)

Function Description:

- This function is used to update input plug-in effect.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource
UINT32	**Id**	Effect ID
AMBA_AUDIO_PLUGIN_EFFECT_CS_s	***pCsShadow**	Plug-in effect control structure pointer. Please refer to Section 2.2.50.1 for the definition.

*Table 2-134. Parameters for Audio API **AmbaAudio_InputPluginEffectUpdate()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-135. Returns for Audio API **AmbaAudio_InputPluginEffectUpdate()**.*

Example:

```
UINT32 *pAudioInputCtrl;
UINT32 Id = 5;
AMBA_AUDIO_PLUGIN_EFFECT_CS_s CsShadow;

CsShadow.self = &EffectSelf;
CsShadow.setup = NULL;
CsShadow.report = EffectReport;
CsShadow.proc = EffectProc;
…
AmbaAudio_InputPluginEffectUpdate(pAudioInputCtrl, Id, &CsShadow);
```

See Also:

AmbaAudio_OutputPluginEffectUpdate()

2.2.58 AmbaAudio_InputPowerMonitorEnable

API Syntax:

AmbaAudio_InputPowerMonitorEnable (UINT32 *pHdlr)

Function Description:

- This function is used to enable the Input task power meter effect.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource

*Table 2-136. Parameters for Audio API **AmbaAudio_InputPowerMonitorEnable()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-137. Returns for Audio API **AmbaAudio_InputPowerMonitorEnable()**.*

Example:

```
UINT32 *pAudioInputCtrl;

…
AmbaAudio_InputPowerMonitorEnable(pAudioInputCtrl);
```

See Also:

AmbaAudio_InputPowerMonitorDisable()

2.2.59 AmbaAudio_InputPowerMonitorGetdB

API Syntax:

AmbaAudio_InputPowerMonitorGetdB (UINT32 *pHdlr, UINT32 *pPower, UINT32 *pPowerPeak)

Function Description:

- This function is used to get the Input task power values.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource
UINT32	***pPower**	Address of power value array. It is an un-signed integer array which size is 2. Each value in this array stands for the power value of left and right channel.
UINT32	***pPowerPeak**	Address of peak power value array. It is an un-signed integer array which size is 2. Each value in this array stands for the peak power value of left and right channel.

*Table 2-138. Parameters for Audio API **AmbaAudio_InputPowerMonitorGetdB()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-139. Returns for Audio API **AmbaAudio_InputPowerMonitorGetdB()**.*

Example:

```
UINT32 *pAudioInputCtrl;
UINT32 Power[2], PeakPower[2];

…
AmbaAudio_InputPowerMonitorGetdB(pAudioInputCtrl, Power, PeakPower);
AmbaPrint("Power L: %d, Power R: %d", Power[0], Power[1]);
AmbaPrint("Peak Power L: %d, Peak Power R: %d", PeakPower[0], PeakPower[1]);
```

See Also:

None

2.2.60 AmbaAudio_InputPowerMonitorDisable

API Syntax:

AmbaAudio_InputPowerMonitorDisable (UINT32 *pHdlr)

Function Description:

- This function is used to disable the Input task power meter effect.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource

*Table 2-140. Parameters for Audio API **AmbaAudio_InputPowerMonitorDisable()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-141. Returns for Audio API **AmbaAudio_InputPowerMonitorDisable()**.*

Example:

```
UINT32 *pAudioInputCtrl;

…
AmbaAudio_InputPowerMonitorDisable(pAudioInputCtrl);
```

See Also:

AmbaAudio_InputPowerMonitorEnable()

2.2.61 AmbaAudio_InputSetUpCalib

API Syntax:

AmbaAudio_InputSetUpCalib (UINT32 *pHdlr, AMBA_AUDIO_CALIB_CTRL_s *pConfig)

Function Description:

- This function is used to set up audio calibration operations.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource
AMBA_AUDIO_CALIB_CTRL_s	***pConfig**	Configuration point of the audio calibration operations setting. Please refer to "Chapter 3 Audio Calibration of *SDK6 UG Calibration"* for more details.

*Table 2-142. Parameters for Audio API **AmbaAudio_InputSetUpCalib()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-143. Returns for Audio API **AmbaAudio_InputSetUpCalib()**.*

Example:

```
UINT32 *pAudioInputCtrl;
AMBA_AUDIO_CALIB_CTRL_s Config;

...
Config.CalibMode = AUDIO_CALIB_PROC;
Config.CalibProcCtrl.CalibOperateMode = 1;
Config.CalibProcCtrl.pCalibNoiseThAddr = NULL;
Config.CalibProcCtrl.pCalibRangeAddr = NULL;
Config.CalibProcCtrl.CalibreFreqIdx = 42;  /* 1kHz freqency idx: 42 */
Config.CalibProcCtrl.pCalibBuffer = AmbaAudio_CalibData.pMemAlignedBase;
AmbaAudio_InputSetUpCalib(pAudioInputCtrl, &Config);
```

See Also:

None

2.2.62 AmbaAudio_InputCalibGetCurve

API Syntax:

AmbaAudio_InputCalibGetCurve (UINT32 *pHdlr)

Function Description:

- This function is used to get calibration data curve. The return value is the address that is an un-signed char array whose size is 2048 (1024 subband *2 channel, non-interleave).

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource

*Table 2-144. Parameters for Audio API **AmbaAudio_InputCalibGetCurve()**.*

Returns:

Return	Description
INT8 *	This API will return 1024 x 2 x INT8 data array.

*Table 2-145. Returns for Audio API **AmbaAudio_InputCalibGetCurve()**.*

Example:

```
UINT32 *pAudioInputCtrl;
INT8 *pCalibCurv;
int i;

...
pCalibCurv = AmbaAudio_InputCalibGetFreqCurve(pAudioInputCtrl);
for (i = 0; i < 1024*2 ; i++) {
    AmbaPrint("Calib %d", *pCalibCurv++);
}
```

See Also:

AmbaAudio_InputCalibGetFreqCurve()

2.2.63 AmbaAudio_InputCalibGet_dBFS

API Syntax:

AmbaAudio_InputCalibGet_dBFS (UINT32 *pHdlr)

Function Description:

- This function is to return the address of an signed integer array whose size is 2. The dBFS values of audio left and right channel store on the array. The range of the dBFS value is from 0 to -30 dBFS. As the power of the testing pattern signal is smaller, the THD+N value will not be calculated correctly. The limitation dBFS value of the testing pattern must be larger than -30 dBFS.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource

*Table 2-146. Parameters for Audio API **AmbaAudio_InputCalibGet_dBFS()**.*

Returns:

Return	Description
INT32 *	This API will return 2 x INT32 data array.

*Table 2-147. Returns for Audio API **AmbaAudio_InputCalibGet_dBFS()**.*

Example:

```
UINT32 *pAudioInputCtrl;
INT32 *pTargetdBFS;

...
pTargetdBFS = AmbaAudio_InputCalibGet_dBFS(pAudioInputCtrl);
AmbaPrint("dBFS => Left Ch:%d, Right Ch:%d", *pTargetdBFS, *(pTargetdBFS+1));
```

See Also:

AmbaAudio_InputCalibGetTHD_N()

2.2.64 AmbaAudio_InputCalibGetTHD_N

API Syntax:

AmbaAudio_InputCalibGetTHD_N (UINT32 *pHdlr)

Function Description:

- This function is to return the address of an signed integer array whose size is 2. The THD+N values of left and right channel store on the array. The THD+N value is a fractional number which uses 17.15 fixed point format to record.
- Example : THD+N = 100 => 100/32768 = 0.305%
- THD+N = 300 => 300/32768 = 0.92%

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource

*Table 2-148. Parameters for Audio API **AmbaAudio_InputCalibGetTHD_N()**.*

Returns:

Return	Description
INT32 *	This API will return 2 x INT32 data array.

*Table 2-149. Returns for Audio API **AmbaAudio_InputCalibGetTHD_N()**.*

Example:

```
UINT32 *pAudioInputCtrl;
INT32 *pTargetTHD_N;

...
pTargetTHD_N= AmbaAudio_InputCalibGetTHD_N(pAudioInputCtrl);
AmbaPrint("dBFS => Left Ch:%d, Right Ch:%d", *pTargetdBFS, *(pTargetdBFS+1));
```

See Also:

AmbaAudio_InputCalibGet_dBFS()

2.2.65 AmbaAudio_InputCalibGetFreqCurve

API Syntax:

AmbaAudio_InputCalibGetFreqCurve (UINT32 *pHdlr)

Function Description:

- This function is used to return the address of a signed char array whose size is 1024(sub-band) *2 (channel). The first 1024 elements of the array is left channel subband and the other 1024 elements of the array is right channel subband. Each value in this array stands for the dBFS value of its band. The dBFS value range is from 0 to -96.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource

*Table 2-150. Parameters for Audio API **AmbaAudio_InputCalibGetFreqCurve()**.*

Returns:

Return	Description
INT8 *	This API will return 1024 x 2 x INT8 data array.

*Table 2-151. Returns for Audio API **AmbaAudio_InputCalibGetFreqCurve()**.*

Example:

```
UINT32 *pAudioInputCtrl;
INT8 *pCalibFreqCurv;
int i;

...
pCalibFreqCurv = AmbaAudio_InputCalibGetFreqCurve(pAudioInputCtrl);
for (i = 0; i < 1024*2 ; i++) {
    AmbaPrint("Freq curve: %d", *pCalibCurv++);
}
```

See Also:

AmbaAudio_InputCalibGetCurve()

2.2.66 AmbaAudio_InputDisableCalib

API Syntax:

AmbaAudio_InputDisableCalib (UINT32 *pHdlr)

Function Description:

- This function is used to disable audio calibration effect. The audio calibration effect will be enabled when users call **AmbaAudio_InputSetUpCalib()**. If users do not want to use the audio calibration effect, they can call this API to disable the audio calibration effect.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the task resource

*Table 2-152. Parameters for Audio API **AmbaAudio_InputDisableCalib()**.*

Returns:

Return	Description
INT8 *	This API will return 1024 x 2 x INT8 data array.

*Table 2-153. Returns for Audio API **AmbaAudio_InputDisableCalib()**.*

Example:

```
UINT32 *pAudioInputCtrl;

...
AmbaAudio_InputDisableCalib(pAudioInputCtrl);
```

See Also:

AmbaAudio_InputCalibGetCurve()

2.2.67 AmbaAudio_EffectCalibBufferSize

API Syntax:

AmbaAudio_EffectCalibBufferSize (void)

Function Description:

- This function is used to get the required buffer size for audio calibration operations.

Parameters:

None

Returns:

Return	Description
>/= 0	Required buffer size

*Table 2-154. Returns for Audio API **AmbaAudio_EffectCalibBufferSize()**.*

Example:

```
UINT32 Size;

...
Size = AmbaAudio_EffectCalibBufferSize();
```

See Also:

AmbaAudio_InputSetUpCalib()

2.2.68 AmbaAudio_EffectUpdownSampleRateConvertSetup

API Syntax:

AmbaAudio_EffectUpdownSampleRateConvertSetup (UINT32 *pHdlr, UINT32 Id, UINT32 SampleRate, UINT32 OutSampleRate)

Function Description:

- This function is used to set the Sample Rate Convert (SRC) effect operations.

Parameters:

Type	Parameter	Description
UINT32	***pHdlr**	Handle of the the task resource
UINT32	**Id**	Effect ID
UINT32	**SampleRate**	Input sample rate
UINT32	**OutSampleRate**	Output sample rate

*Table 2-155. Parameters for Audio API **AmbaAudio_EffectUpdownSampleRateConvertSetup()**.*

Returns:

Return	Description
0	Success
-1	Failure

*Table 2-156. Returns for Audio API **AmbaAudio_EffectUpdownSampleRateConvertSetup()**.*

Example:

```
UINT32 *pDecTest;
UINT32 Id = 2;
UINT32 SrcFreq, DstFreq;


…
AmbaAudio_EffectUpdownSampleRateConvertSetup(pDecTest, Id, SrcFreq,
DstFreq);
```

See Also:

None

Appendix 1 Additional Resources

Related resources include:

- *AMBARELLA SDK6 AN: Custom Audio CODEC Driver*
- *AMBARELLA SDK6 API: System (AmbaSYS)*
- *AMBARELLA SDK6 AN: Audio Plugin Effect*
- *AMBARELLA SDK6 UG: Calibration*

Please contact an Ambarella representative for a full list of related resources.

Appendix 2 Important Notice

All Ambarella design specifications, datasheets, drawings, files, and other documents (together and separately, "materials") are provided on an "*as is*" basis, and Ambarella makes no warranties, expressed, implied, statutory, or otherwise with respect to the materials, and expressly disclaims all implied warranties of noninfringement, merchantability, and fitness for a particular purpose. The information contained herein is believed to be accurate and reliable. However, Ambarella assumes no responsibility for the consequences of use of such information.

Ambarella Incorporated reserves the right to correct, modify, enhance, improve, and otherwise change its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

All products are sold subject to Ambarella's terms and conditions of sale supplied at the time of order acknowledgment. Ambarella warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with its standard warranty. Testing and other quality control techniques are used to the extent Ambarella deems necessary to support this warranty.

Ambarella assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using Ambarella components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

Ambarella does not warrant or represent that any license, either expressed or implied, is granted under any Ambarella patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Ambarella products or services are used. Information published by Ambarella regarding third-party products or services does not constitute a license from Ambarella to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from Ambarella under the patents or other intellectual property of Ambarella.

Reproduction of information from Ambarella documents is not permissible without prior approval from Ambarella.

Ambarella products are not authorized for use in safety-critical applications (such as life support) where a failure of the product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Customers acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of Ambarella products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by Ambarella. Further, Customers must fully indemnify Ambarella and its representatives against any damages arising out of the use of Ambarella products in such safety-critical applications.

Ambarella products are neither designed nor intended for use in military/aerospace applications or environments. Customers acknowledge and agree that any such use of Ambarella products is solely at the Customer's risk, and they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

Appendix 3 Revision History

NOTE: Page numbers for previous drafts may differ from page numbers in the current version.

Version	Date	Comments
1.0	20 August 2013	Formatting
1.5	8 October 2013	Refine descriptions; update formatting
1.6	30 December 2013	Add new APIs in chapter 2. Update in overview.
1.7	17 April 2014	Update in overview and Audio API.
1.8	26 September 2014	Formatted to SDK6
1.9	5 January 2015	Update in Section 2.2.4, 2.2.14, 2.2.24 and 2.2.33.
2.0	12 June 2015	Add Section 2.2.68 AmbaAudio_EffectUpdownSampleRateConvertSetup.
2.1	22 June 2015	Update Sections 2.2.6, 2.2.6.3, 2.2.16.1 and 2.2.16.2. Add Section 2.2.6.9

Table A3-1. Revision History.