RENESAS

**CONFIDENTIAL**

# ADSP Reference Renderer/Capture Plugin
# RCG3AHPLN0202ZDO

User's Manual

RCG3AHPLN0202ZDOE

Rev. 0.10   May, 2018

Renesas Electronics
www.renesas.com

**CONFIDENTIAL**

Trademarks
· Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.
· Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
· Windows and Windows Media are registered trademarks of Microsoft Corporation in the United States and other countries.
· Android is a trademark of Google Inc. Use of this trademark is subject to Google permissions.
· All other company names and product names mentioned in this manual are registered trademarks or trademarks of their respective companies.
· The registered trademark symbol (®) and trademark symbol (™) are omitted in this manual.

# How to Use This Manual

## 1. Purpose and Target Reader

 This manual is designed to provide the user with an understanding of the interface specifications of the Software product. It is intended for users designing application systems incorporating the Software product. Please refer to the related documents with this product.

Use this Software after carefully reading the precautions. The precautions are stated in the main text of each section, at the end of each section, and in the usage precaution section.

The revision history summarizes major corrections and additions to the previous version. It does not cover all the changes. For details, refer to this manual.

## 2. Restrictions on the Use of this Software

This software is MIT license. The certificates from the licensor do not provide any assurances to users that the product performs reliably, intellectual property rights are protected, disputes are resolved by contract, and specifications are not subject to major changes. The user should use this software at his or her own risk.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## 3. Related Manuals

## 4. Technical Terms and Abbreviation

# - Table of Contents -

# - List of Figures -

# - List of Tables -

# 1. Overview

This section provides an overview of the Renderer plugin. It contains renderer and capture function.

## 1.1 Specifications Outline

Renderer function plays the audio signal based on the parameter that was set.

Capture function capture/record the audio signal based on the parameter that was set.

Table 1-1      Basic Specification

| Item | Description |
|------|-------------|
| DSP | Cadence Design Systems, Inc. HiFi2 |
| Compiler | Xtensa C and C++ Compiler (version 12.0.4) |
| Endian | Little Endian |

Table 1-2      Supported Specifications for capture function

| Item | Description |
|------|-------------|
| Input data format | 16-bit/24-bit linear PCM (fixed point) |
| Output data format | 16-bit/24-bit linear PCM (fixed point) |
| Sampling frequency (Hz) supported | 48000 / 44100 / 32000 |
| Number of channels supported | Max 2 channels |
| Reentrant | Supported |
| Volume control | Support volume update during plugin execution |
| Other | |
| Restrictions | |

Table 1-3      Supported Specifications for renderer function

| Item | Description |
|------|-------------|
| Input data format | 16-bit/24-bit linear PCM (fixed point) |
| Output data format | 16-bit/24-bit linear PCM (fixed point) |
| Sampling frequency (Hz) supported | 48000 / 44100 / 32000 |
| Input data channel | Monaural, stereo, 4 channels, 6 channels, 8 channels |
| Output data channel | Max 2 channels |
| Reentrant | Supported |
| Volume control | Support volume update during plugin execution. |
| Other | |
| Restrictions | |

[Note] In case of channel transfer from input monaural to output stereo, the sound will output to both two channels.

Table 1-4          Memory Size Requirements

| Memory type | Location | Memory area name | | | Size (in bytes) | |
|---|---|---|---|---|---|---|
| Instruction | ROM | Instruction area | | | 59041 | |
| | | Constant table area | | | | |
| | | Other area(Depended on the compiler) | | | | |
| Data | RAM Capture | Software work area | | | 30388 | |
| | | Area breakdown | Persistent area | | Size breakdown | 13972 |
| | | | Scratch area | | | 0 |
| | | | D-TCM area | | | 16384 |
| | | | Built-in descriptor area | | | 32 |
| | | User work area | | | 7536 | |
| | | Area breakdown | Input buffer | | Size breakdown | 0 |
| | | | Output buffer | | | 6144 |
| | | | Structure | | | 1392 |
| | | Stack area | | | 464 | |
| | | Other area(Depended on the compiler) | | | 0 | |
| | RAM Renderer | Software work area | | | 30376 | |
| | | Area breakdown | Persistent area | | Size breakdown | 13960 |
| | | | Scratch area | | | 0 |
| | | | D-TCM area | | | 16384 |
| | | | Built-in descriptor area | | | 32 |
| | | User work area | | | 7552 | |
| | | Area breakdown | Input buffer | | Size breakdown | 6144 |
| | | | Output buffer | | | 0 |
| | | | Structure | | | 1408 |
| | | Stack area | | | 480 | |
| | | Other area(Depended on the compiler) | | | 0 | |

[Note] Area whose location is shown as ROM in the location column can be included in RAM or ROM.
[Note] Area whose location is shown as RAM in the location column can be included in RAM only.
[Note] Built-in is a memory area to allocate descriptor memory, which need in the DMAC transfer type of plugin.

Table 1-5          Version Information

| Item | Description |
|---|---|
| Library Version information | Version 1.0.4 |
| API Version information | Version 1.0.0 |

## 1.2    Configuration

Figure 1-1 shows an example of the ADSP system configuration which uses renderer function.
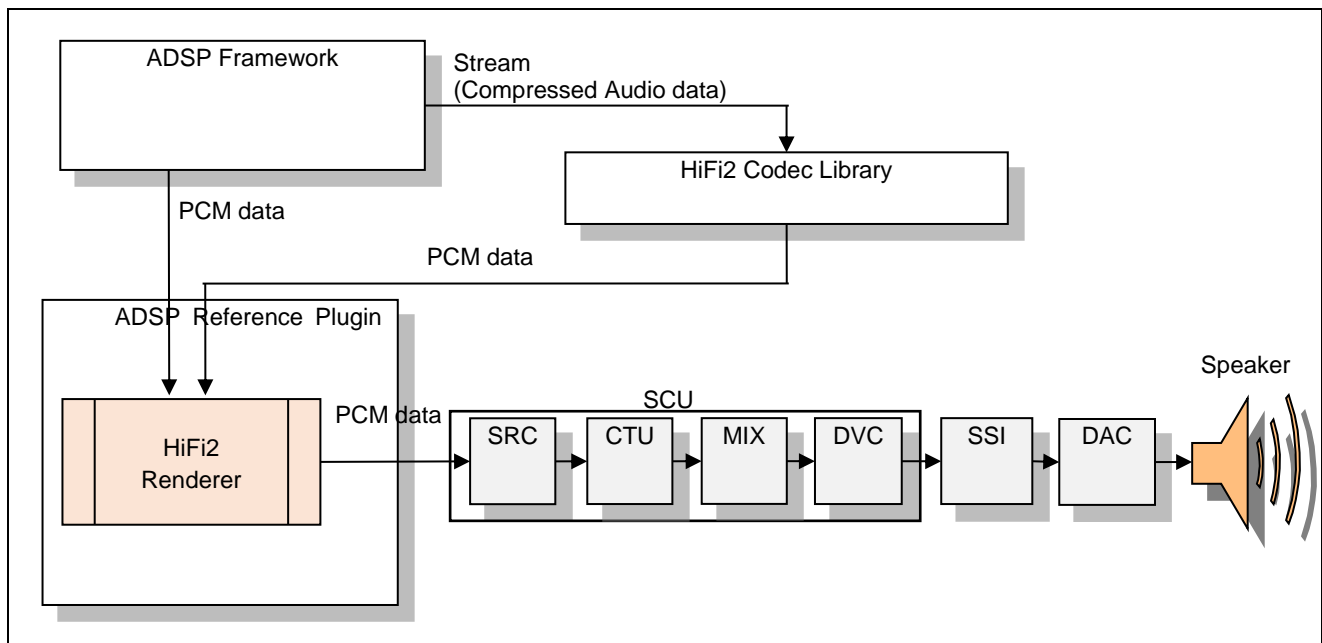


Figure 1-1          Example of the ADSP System Configuration for renderer function

Figure 1-2 **エラー! 参照元が見つかりません。** show an example of using channel converter and mixer function to mix 4 streams from 4 Renderer plugins.
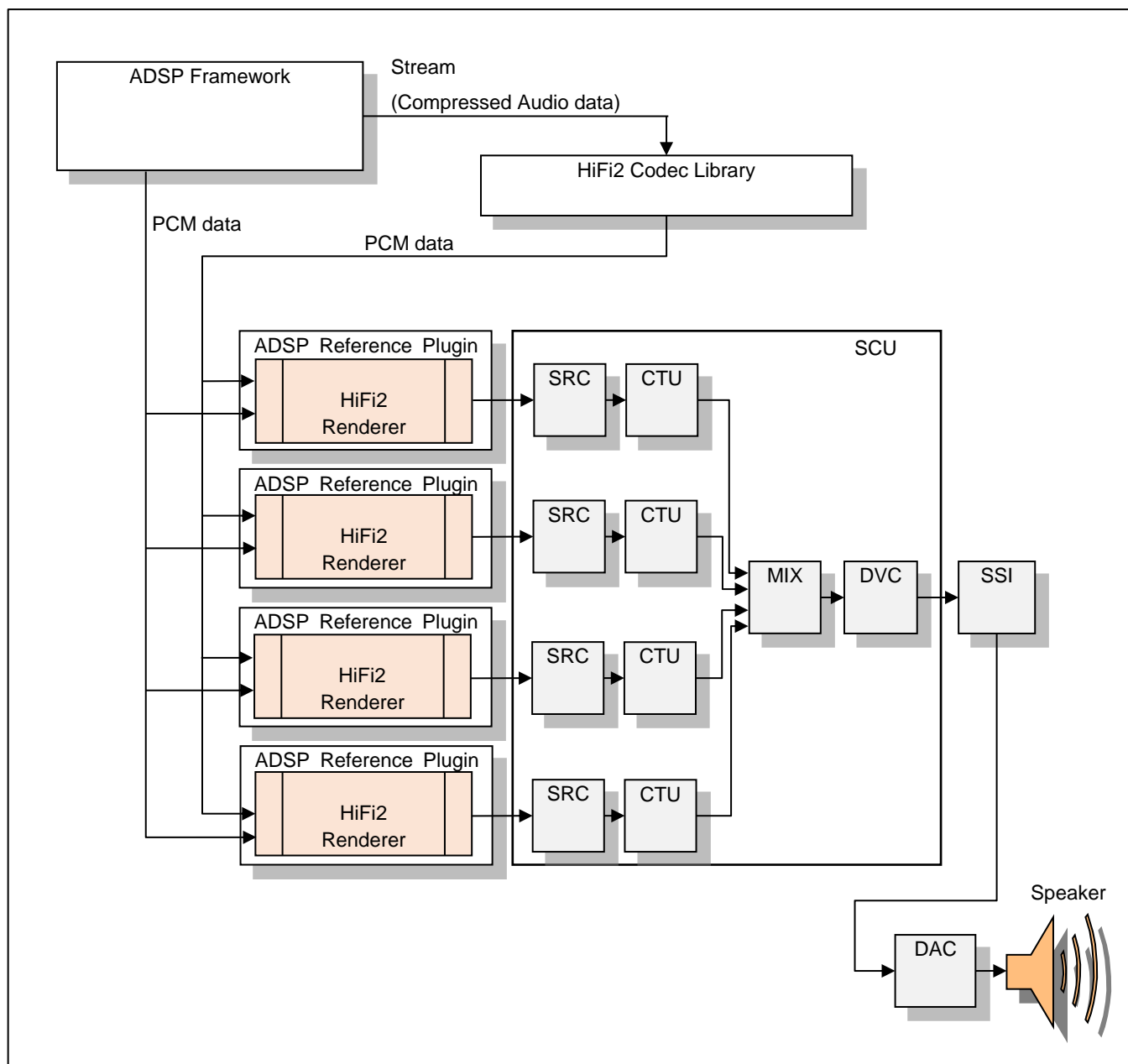


Figure 1-2　　　Example of the ADSP System Configuration uses CTU/MIX for renderer function

Figure 1-3 shows an example of the ADSP system configuration which uses capture function.
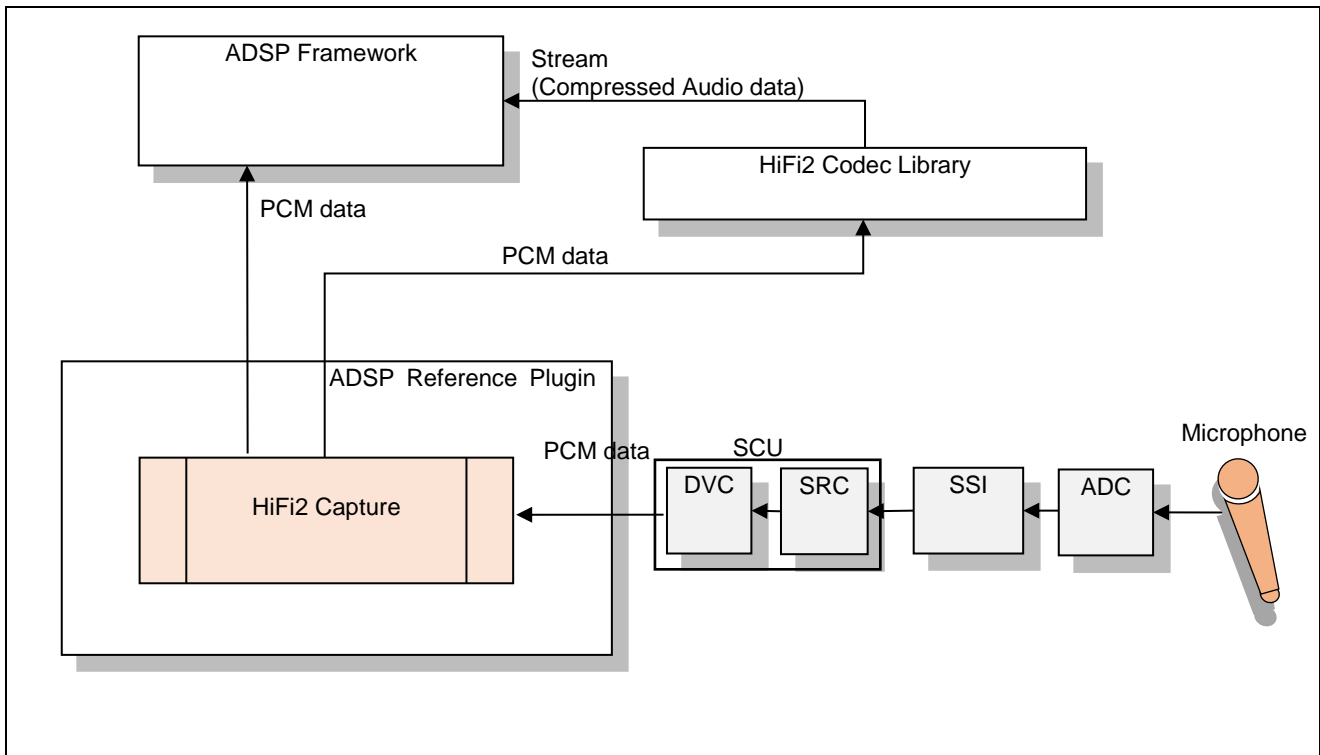


Figure 1-3　　　　Example of the ADSP System Configuration for capture function

1. Stream (Compressed Audio data)

   Compressed Audio data is a linear PCM data sample compressed according to the compression Audio specifications. For these specifications, depends on HiFi2 Codec Library.

2. HiFi2 Codec Library
   It is Codec Library for HiFi2. It decodes the compression Audio in renderer case and encodes in capture case. The user should procure to suit the target system.

3. ADSP Framework

   It controls ADSP Plugin. It is software provided separately as Framework.

4. HiFi2 Renderer (ADSP Reference Plugin)
   It performs output handling of PCM data to other Audio device. It is this software set up as ADSP Reference Plugin.

5. HiFi2 Capture (ADSP Reference Plugin)
   It performs input handling of PCM data from other Audio device. It is this software set up as ADSP Reference Plugin.

6. PCM data

   16-bit linear PCM data which is a processing by this software.

7. SCU

   It performs sampling rate converters (SRC), channel transfer (CTU), mixing (MIX), and volume control (DVC).

8. SRC

   It performs sampling rate conversion function.

9. CTU

   It performs channel transfer unit function such as down-mixing and splitter functions.

10. MIX

    It is used for mixing (adding) streams from two to four audio stream sources into a single stream. It also support the volume control (gain level) for each input stream.

11. DVC

    It performs mute and volume control functions.

12. SSI

    Send or receive audio data interfacing with a variety devices of offering I2C format.

13. ADC

    The ADC converts an analog signal into 16-bit linear PCM data.

# 2. Software Specifications

## 2.1 API specifications

Because one interface function accesses the procedure that was appointed by a command in renderer plugin, it is used.
In renderer case

Table 2-1　　　　API Functions of renderer

| xa_rel_rdr | |
|---|---|
| Description | This API is the only access function to the renderer. |
| Syntax | XA_ERRORCODE xa_rel_rdr( <br> xa_codec_handle_t  p_xa_module_obj, <br> WORD32  i_cmd, <br> WORD32  i_idx, <br> pVOID      pv_value); |
| Parameters | p_xa_module_obj : Pointer to opaque API structure. <br><br> i_cmd : Command. (defined in the supplied header files as) <br><br> i_idx : Command subtype or index. (defined in the supplied header files as) <br><br> pv_value : Pointer to the variable used to pass in, or get out properties, from state structure. |
| Returns | Error Code based on the success or failure of API command (defined in the supplied header files as) |

In capture case

Table 2-2　　　　API Functions of capture

| xa_rel_cap | |
|---|---|
| Description | This API is the only access function to the capture. |
| Syntax | XA_ERRORCODE xa_rel_cap( <br> xa_codec_handle_t  p_xa_module_obj, <br> WORD32  i_cmd, <br> WORD32  i_idx, <br> pVOID      pv_value); |
| Parameters | p_xa_module_obj : Pointer to opaque API structure. <br><br> i_cmd : Command. (defined in the supplied header files as) <br><br> i_idx : Command subtype or index. (defined in the supplied header files as) <br><br> pv_value : Pointer to the variable used to pass in, or get out properties, from state structure. |
| Returns | Error Code based on the success or failure of API command (defined in the supplied header files as) |

## 2.2    Command

Using API functions of the Table 2-1 and Table 2-2, it performs each processing by a combination of Command/Subcommand.
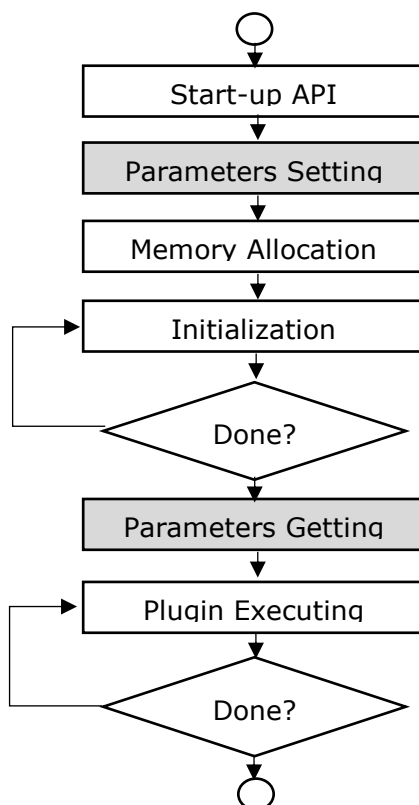


Figure 2-1        Command sequence overview

## 2.2.1    Command list

Below table presents commands used in renderer and capture case.

Table 2-3          List of supported none supported command, subcommand

| Command | Sub command | Ren | Cap |
|---|---|---|---|
| XA_API_CMD_GET_LIB_ID_STRINGS | XA_CMD_TYPE_LIB_VERSION | Y | Y |
| | XA_CMD_TYPE_API_VERSION | Y | Y |
| XA_API_CMD_GET_API_SIZE | - | Y | Y |
| XA_API_CMD_INIT | XA_CMD_TYPE_INIT_API_PRE_CONFIG_PARAMS | Y | Y |
| | XA_CMD_TYPE_INIT_API_POST_CONFIG_PARAMS | Y | Y |
| | XA_CMD_TYPE_INIT_PROCESS | Y | Y |
| | XA_CMD_TYPE_INIT_DONE_QUERY | Y | Y |
| XA_API_CMD_SET_CONFIG_PARAM | XA_CAP_CONFIG_PARAM_PCM_WIDTH | N | Y |
| | XA_CAP_CONFIG_PARAM_CHANNELS | N | Y |
| | XA_CAP_CONFIG_PARAM_SAMPLE_RATE | N | Y |
| | XA_CAP_CONFIG_PARAM_FRAME_SIZE | N | Y |
| | XA_CAP_CONFIG_PARAM_INPUT1 | N | Y |
| | XA_CAP_CONFIG_PARAM_DMACHANNEL1 | N | Y |
| | XA_CAP_CONFIG_PARAM_INPUT2 | N | Y |
| | XA_CAP_CONFIG_PARAM_DMACHANNEL2 | N | Y |
| | XA_CAP_CONFIG_PARAM_OUT_SAMPLE_RATE | N | Y |
| | XA_CAP_CONFIG_PARAM_VOLUME_RATE | N | Y |
| | XA_CAP_CONFIG_PARAM_STATE | N | Y |
| | XA_RDR_CONFIG_PARAM_PCM_WIDTH | Y | N |
| | XA_RDR_CONFIG_PARAM_CHANNELS | Y | N |
| | XA_RDR_CONFIG_PARAM_SAMPLE_RATE | Y | N |
| | XA_RDR_CONFIG_PARAM_FRAME_SIZE | Y | N |
| | XA_RDR_CONFIG_PARAM_OUTPUT1 | Y | N |
| | XA_RDR_CONFIG_PARAM_DMACHANNEL1 | Y | N |
| | XA_RDR_CONFIG_PARAM_OUTPUT2 | Y | N |
| | XA_RDR_CONFIG_PARAM_DMACHANNEL2 | Y | N |
| | XA_RDR_CONFIG_PARAM_OUT_SAMPLE_RATE | Y | N |
| | XA_RDR_CONFIG_PARAM_VOLUME_RATE | Y | N |
| | XA_RDR_CONFIG_PARAM_OUT_CHANNELS | Y | N |
| | XA_RDR_CONFIG_PARAM_MIX_CONTROL | Y | N |
| | XA_RDR_CONFIG_PARAM_STATE | Y | N |
| XA_API_CMD_GET_CONFIG_PARAM | XA_CAP_CONFIG_PARAM_PCM_WIDTH | N | Y |
| | XA_CAP_CONFIG_PARAM_CHANNELS | N | Y |
| | XA_CAP_CONFIG_PARAM_SAMPLE_RATE | N | Y |
| | XA_CAP_CONFIG_PARAM_FRAME_SIZE | N | Y |
| | XA_CAP_CONFIG_PARAM_INPUT1 | N | Y |
| | XA_CAP_CONFIG_PARAM_DMACHANNEL1 | N | Y |
| | XA_CAP_CONFIG_PARAM_INPUT2 | N | Y |
| | XA_CAP_CONFIG_PARAM_DMACHANNEL2 | N | Y |
| | XA_CAP_CONFIG_PARAM_OUT_SAMPLE_RATE | N | Y |
| | XA_CAP_CONFIG_PARAM_VOLUME_RATE | N | Y |
| | XA_CAP_CONFIG_PARAM_STATE | N | Y |
| | XA_RDR_CONFIG_PARAM_PCM_WIDTH | Y | N |
| | XA_RDR_CONFIG_PARAM_CHANNELS | Y | N |
| | XA_RDR_CONFIG_PARAM_SAMPLE_RATE | Y | N |
| | XA_RDR_CONFIG_PARAM_FRAME_SIZE | Y | N |
| | XA_RDR_CONFIG_PARAM_OUTPUT1 | Y | N |
| | XA_RDR_CONFIG_PARAM_DMACHANNEL1 | Y | N |
| | XA_RDR_CONFIG_PARAM_OUTPUT2 | Y | N |
| | XA_RDR_CONFIG_PARAM_DMACHANNEL2 | Y | N |
| | XA_RDR_CONFIG_PARAM_OUT_SAMPLE_RATE | Y | N |
| | XA_RDR_CONFIG_PARAM_VOLUME_RATE | Y | N |

| | | | Ren | Cap |
|---|---|---|---|---|
| | XA_RDR_CONFIG_PARAM_OUT_CHANNELS | | Y | N |
| | XA_RDR_CONFIG_PARAM_MIX_CONTROL | | Y | N |
| | XA_RDR_CONFIG_PARAM_STATE | | Y | N |
| XA_API_CMD_GET_MEMTABS_SIZE | - | | Y | Y |
| XA_API_CMD_SET_MEMTABS_PTR | - | | Y | Y |
| XA_API_CMD_GET_N_MEMTABS | - | | Y | Y |
| XA_API_CMD_GET_MEM_INFO_SIZE | - | | Y | Y |
| XA_API_CMD_GET_MEM_INFO_ALIGNMENT | - | | Y | Y |
| XA_API_CMD_GET_MEM_INFO_TYPE | - | | Y | Y |
| XA_API_CMD_SET_MEM_PTR | - | | Y | Y |
| XA_API_CMD_SET_INPUT_BYTES | - | | Y | N |
| XA_API_CMD_INPUT_OVER | - | | N | Y |
| XA_API_CMD_GET_CURIDX_INPUT_BUF | - | | Y | N |
| XA_API_CMD_EXECUTE | XA_CMD_TYPE_DO_EXECUTE | | N | Y |
| | XA_CMD_TYPE_DONE_QUERY | | Y | N |
| XA_API_CMD_GET_OUTPUT_BYTES | | | N | Y |

Ren: Renderer   Cap: Capture

Y: command or sub command is used by renderer or capture.

N: command or sub command is not used by renderer or capture. They are presented in detailed command part to refer. Actually, they are not implemented.

-: None sub command

2.2.1.1　　　Start-up API

Table 2-4　　　List of Initialization Commands

| upper stage : Command / lower step : Subcommand | | Description |
|---|---|---|
| 1 | XA_API_CMD_GET_LIB_ID_STRINGS | Get the version of the library. |
| | XA_CMD_TYPE_LIB_VERSION | |
| 2 | XA_API_CMD_GET_LIB_ID_STRINGS | Get the version of the API. |
| | XA_CMD_TYPE_API_VERSION | |
| 3 | XA_API_CMD_GET_API_SIZE | Get the size of the API structure |
| | (NULL) | |
| 4 | XA_API_CMD_INIT | Set the default values of all the configuration parameters |
| | XA_CMD_TYPE_INIT_API_PRE_CONFIG_PARAMS | |

### 2.2.1.2 Setting parameters

Table 2-5 List of Set Commands for renderer

| | upper stage : Command / lower step : Subcommand | Description |
|---|---|---|
| 1 | XA_API_CMD_SET_CONFIG_PARAM | Set the input PCM sample bit width to 16 or 24 bits |
| | XA_RDR_CONFIG_PARAM_PCM_WIDTH | |
| 2 | XA_API_CMD_SET_CONFIG_PARAM | Set the input PCM channels (support monaural, stereo, 4 channels, 6 channels, 8 channels) |
| | XA_RDR_CONFIG_PARAM_CHANNELS | |
| 3 | XA_API_CMD_SET_CONFIG_PARAM | Set the input PCM sampling frequency (supported 32000/44100/48000 kHz) |
| | XA_RDR_CONFIG_PARAM_SAMPLE_RATE | |
| 4 | XA_API_CMD_SET_CONFIG_PARAM | Set the input/output frame size |
| | XA_RDR_CONFIG_PARAM_FRAME_SIZE | |
| 5 | XA_API_CMD_SET_CONFIG_PARAM | Set the output destination Audio device 1st for Renderer |
| | XA_RDR_CONFIG_PARAM_OUTPUT1 | |
| 6 | XA_API_CMD_SET_CONFIG_PARAM | Set ADMA channel number usage for Audio device 1st (supported Audio-DMAC, Audio-DMAC-pp) |
| | XA_RDR_CONFIG_PARAM_DMACHANNEL1 | |
| 7 | XA_API_CMD_SET_CONFIG_PARAM | Set the output destination Audio device 2nd for Renderer |
| | XA_RDR_CONFIG_PARAM_OUTPUT2 | |
| 8 | XA_API_CMD_SET_CONFIG_PARAM | Set ADMA channel number usage for Audio device 2nd (supported Audio-DMAC, Audio-DMAC-pp) |
| | XA_RDR_CONFIG_PARAM_DMACHANNEL2 | |
| 9 | XA_API_CMD_SET_CONFIG_PARAM | Set the output PCM sampling frequency (supported 32000/44100/48000 kHz) |
| | XA_RDR_CONFIG_PARAM_OUT_SAMPLE_RATE | |
| 10 | XA_API_CMD_SET_CONFIG_PARAM | Set the output PCM volume rate compare with input PCM (supported from 0 – 8 times) |
| | XA_RDR_CONFIG_PARAM_VOLUME_RATE | |
| 11 | XA_API_CMD_SET_CONFIG_PARAM | Set the output PCM channels (support maximum 2 channels) |
| | XA_RDR_CONFIG_PARAM_OUT_CHANNELS | |
| 12 | XA_API_CMD_SET_CONFIG_PARAM | Set the MIX module usage for Renderer (value 1/0 to used/unused mix function) |
| | XA_RDR_CONFIG_PARAM_MIX_CONTROL | |
| 13 | XA_API_CMD_SET_CONFIG_PARAM | Set the operation state of Renderer plugin (RUN/PAUSE/IDLE) |
| | XA_RDR_CONFIG_PARAM_STATE | |

Table 2-6          List of Set Commands for capture

| upper stage : Command / lower step : Subcommand | | Description |
|---|---|---|
| 1 | XA_API_CMD_SET_CONFIG_PARAM | Set the input PCM sample bit width to 16 or 24 bits |
| | XA_CAP_CONFIG_PARAM_PCM_WIDTH | |
| 2 | XA_API_CMD_SET_CONFIG_PARAM | Set the input PCM channels (Maximum is 2 channels) |
| | XA_CAP_CONFIG_PARAM_CHANNELS | |
| 3 | XA_API_CMD_SET_CONFIG_PARAM | Set the input PCM sampling frequency (supported 32000/44100/48000 kHz) |
| | XA_CAP_CONFIG_PARAM_SAMPLE_RATE | |
| 4 | XA_API_CMD_SET_CONFIG_PARAM | Set the input/output frame size |
| | XA_CAP_CONFIG_PARAM_FRAME_SIZE | |
| 5 | XA_API_CMD_SET_CONFIG_PARAM | Set the input source Audio device 1st for Capture |
| | XA_CAP_CONFIG_PARAM_INPUT1 | |
| 6 | XA_API_CMD_SET_CONFIG_PARAM | Set ADMA channel number usage for Audio device 1st (supported Audio-DMAC, Audio-DMAC-pp) |
| | XA_CAP_CONFIG_PARAM_DMACHANNEL1 | |
| 7 | XA_API_CMD_SET_CONFIG_PARAM | Set the input source Audio device 2nd for Capture |
| | XA_CAP_CONFIG_PARAM_INPUT2 | |
| 8 | XA_API_CMD_SET_CONFIG_PARAM | Set ADMA channel number usage for Audio device 2nd (supported Audio-DMAC, Audio-DMAC-pp) |
| | XA_CAP_CONFIG_PARAM_DMACHANNEL2 | |
| 9 | XA_API_CMD_SET_CONFIG_PARAM | Set the output PCM sampling frequency (supported 32000/44100/48000 kHz) |
| | XA_CAP_CONFIG_PARAM_OUT_SAMPLE_RATE | |
| 10 | XA_API_CMD_SET_CONFIG_PARAM | Set the output PCM volume rate compare with input PCM (supported from 0 – 8 times) |
| | XA_CAP_CONFIG_PARAM_VOLUME_RATE | |
| 11 | XA_API_CMD_SET_CONFIG_PARAM | Set the operation state of Capture plugin (RUN/PAUSE/IDLE) |
| | XA_CAP_CONFIG_PARAM_STATE | |

ADSP Reference Renderer/Capture Plugin User's Manual**2 Software** Specifications

2.2.1.3 Memory allocation

Table 2-7 List of Memory allocation Commands

| | upper stage : Command / lower step : Subcommand | Description |
|---|---|---|
| 1 | XA_API_CMD_GET_MEMTABS_SIZE | Get the size of the memory structures to be allocated for the plugin tables. |
| | (NULL) | |
| 2 | XA_API_CMD_SET_MEMTABS_PTR | Pass the memory structure pointer allocated for the tables. |
| | (NULL) | |
| 3 | XA_API_CMD_INIT | Calculate the required sizes for all the memory blocks based on the setting specific parameters. |
| | XA_CMD_TYPE_INIT_API_POST_CONFIG_PARAMS | |
| 4 | XA_API_CMD_GET_N_MEMTABS | Obtain the number of memory blocks required by plugin. |
| | (NULL) | |
| 5 | XA_API_CMD_GET_MEM_INFO_SIZE | Get the size of the memory type being referred to by the index. |
| | (NULL) | |
| 6 | XA_API_CMD_GET_MEM_INFO_ALIGNMENT | Get the alignment information of the memory type being referred to by the index. |
| | (NULL) | |
| 7 | XA_API_CMD_GET_MEM_INFO_TYPE | Get the type of memory being referred to by the index. |
| | (NULL) | |
| 8 | XA_API_CMD_SET_MEM_PTR | Set the pointer to the memory allocated for the referred index to the input value. |
| | (NULL) | |

### 2.2.1.4 Initialize plugin

Table 2-8 List of initialize commands

| upper stage : Command / lower step : Subcommand | | Description |
|---|---|---|
| 1 | XA_API_CMD_SET_INPUT_BYTES | Set the number of bytes available in the input buffer. |
| | (NULL) | |
| 2 | XA_API_CMD_INPUT_OVER | Signal to the plugin the end of the bit stream in renderer case. |
| | (NULL) | |
| 3 | XA_API_CMD_INIT | Setup for the HW operation, and initialize state and configuration structure. |
| | XA_CMD_TYPE_INIT_PROCESS | |
| 4 | XA_API_CMD_INIT | Check if the initialization process has completed. |
| | XA_CMD_TYPE_INIT_DONE_QUERY | |
| 5 | XA_API_CMD_GET_CURIDX_INPUT_BUF | Get the number of input buffer bytes consumed. |
| | (NULL) | |

## 2.2.1.5 Getting parameters

Table 2-9    List of Get commands for renderer

| upper stage : Command / lower step : Subcommand | | Description |
|---|---|---|
| 1 | XA_API_CMD_GET_CONFIG_PARAM | Get the output PCM sample bit width to 16 or 24 bits |
| | XA_RDR_CONFIG_PARAM_PCM_WIDTH | |
| 2 | XA_API_CMD_GET_CONFIG_PARAM | Get the input PCM channels (support monaural, stereo, 4 channels, 6 channels, 8 channels) |
| | XA_RDR_CONFIG_PARAM_CHANNELS | |
| 3 | XA_API_CMD_GET_CONFIG_PARAM | Get the output PCM sampling frequency (supported 32000/44100/48000 kHz) |
| | XA_RDR_CONFIG_PARAM_SAMPLE_RATE | |
| 4 | XA_API_CMD_GET_CONFIG_PARAM | Get the output PCM frame size |
| | XA_RDR_CONFIG_PARAM_FRAME_SIZE | |
| 5 | XA_API_CMD_GET_CONFIG_PARAM | Get Renderer output destination Audio device $1^{st}$ |
| | XA_RDR_CONFIG_PARAM_OUTPUT1 | |
| 6 | XA_API_CMD_GET_CONFIG_PARAM | Get ADMA channel number usage for Audio device $1^{st}$ |
| | XA_RDR_CONFIG_PARAM_DMACHANNEL1 | |
| 7 | XA_API_CMD_GET_CONFIG_PARAM | Get Renderer output destination Audio device $2^{nd}$ |
| | XA_RDR_CONFIG_PARAM_OUTPUT2 | |
| 8 | XA_API_CMD_GET_CONFIG_PARAM | Get ADMA channel number usage for Audio device $2^{nd}$ |
| | XA_RDR_CONFIG_PARAM_DMACHANNEL2 | |
| 9 | XA_API_CMD_GET_CONFIG_PARAM | Get the output PCM sampling frequency |
| | XA_RDR_CONFIG_PARAM_OUT_SAMPLE_RATE | |
| 10 | XA_API_CMD_GET_CONFIG_PARAM | Get the output PCM volume rate compare with input PCM |
| | XA_RDR_CONFIG_PARAM_VOLUME_RATE | |
| 11 | XA_API_CMD_GET_CONFIG_PARAM | Get the output PCM channels (support maximum 2 channels) |
| | XA_RDR_CONFIG_PARAM_OUT_CHANNELS | |
| 12 | XA_API_CMD_GET_CONFIG_PARAM | Get the MIX module usage of Renderer (value 1/0 to used/unused mix function) |
| | XA_RDR_CONFIG_PARAM_MIX_CONTROL | |
| 13 | XA_API_CMD_GET_CONFIG_PARAM | Get the operation state of Renderer plugin (RUN/PAUSE/IDLE) |
| | XA_RDR_CONFIG_PARAM_STATE | |

Table 2-10        List of Get commands for capture

| | upper stage : Command / lower step : Subcommand | Description |
|---|---|---|
| 1 | XA_API_CMD_GET_CONFIG_PARAM | Get the output PCM sample bit width to 16 or 24 bits |
| | XA_CAP_CONFIG_PARAM_PCM_WIDTH | |
| 2 | XA_API_CMD_GET_CONFIG_PARAM | Get the output PCM channels (Maximum is 2 channels) |
| | XA_CAP_CONFIG_PARAM_CHANNELS | |
| 3 | XA_API_CMD_GET_CONFIG_PARAM | Get the output PCM sampling frequency (supported 32000/44100/48000 kHz) |
| | XA_CAP_CONFIG_PARAM_SAMPLE_RATE | |
| 4 | XA_API_CMD_GET_CONFIG_PARAM | Get the output PCM frame size |
| | XA_CAP_CONFIG_PARAM_FRAME_SIZE | |
| 5 | XA_API_CMD_GET_CONFIG_PARAM | Get Capture input source Audio device 1st |
| | XA_CAP_CONFIG_PARAM_INPUT1 | |
| 6 | XA_API_CMD_GET_CONFIG_PARAM | Get ADMA channel number usage for Audio device 1st |
| | XA_CAP_CONFIG_PARAM_DMACHANNEL1 | |
| 7 | XA_API_CMD_GET_CONFIG_PARAM | Get Capture input destination Audio device 2nd |
| | XA_CAP_CONFIG_PARAM_INPUT2 | |
| 8 | XA_API_CMD_GET_CONFIG_PARAM | Get ADMA channel number usage for Audio device 2nd |
| | XA_CAP_CONFIG_PARAM_DMACHANNEL2 | |
| 9 | XA_API_CMD_GET_CONFIG_PARAM | Get the output PCM sampling frequency |
| | XA_CAP_CONFIG_PARAM_OUT_SAMPLE_RATE | |
| 10 | XA_API_CMD_GET_CONFIG_PARAM | Get the output PCM volume rate compare with input PCM |
| | XA_CAP_CONFIG_PARAM_VOLUME_RATE | |
| 11 | XA_API_CMD_GET_CONFIG_PARAM | Get the operation state of Capture plugin (RUN/PAUSE/IDLE) |
| | XA_CAP_CONFIG_PARAM_STATE | |

2.2.1.6　　　　Execution

Table 2-11　　　List of execution commands

| upper stage : Command / lower step : Subcommand | | Description |
|---|---|---|
| 1 | XA_API_CMD_INPUT_OVER | Stop capture in capture case. |
| | (NULL) | |
| 2 | XA_API_CMD_SET_INPUT_BYTES | Set the number of bytes available in the input buffer. |
| | (NULL) | |
| 3 | XA_API_CMD_EXECUTE | Execute the capture |
| | XA_CMD_TYPE_DO_EXECUTE | |
| 4 | XA_API_CMD_EXECUTE | Check if the execution process has completed. |
| | XA_CMD_TYPE_DONE_QUERY | |
| 5 | XA_API_CMD_GET_OUTPUT_BYTES | Get the number of bytes output by the plugin in the last frame. |
| | (NULL) | |
| 6 | XA_API_CMD_GET_CURIDX_INPUT_BUF | Get the number of input buffer bytes consumed. |
| | (NULL) | |

## 2.2.2      Detail of Command Specifications

The next sections describe this library command functions by using the description format below.

| | |
|---|---|
| Subcommand | Name of subcommand |
| Description | Outlines the function. |
| Arguments | Describes the arguments for the function. |
| Return value | Return values of function. |
| Restrictions | Provides information such as precautions in using the function. |

[Note] This syntax format complies with ANSI-C.

### 2.2.2.1 XA_API_CMD_GET_LIB_ID_STRINGS command

| Subcommand | XA_CMD_TYPE_LIB_VERSION |
| --- | --- |
| Description | This command obtains the version of the library in the form of a string. The maximum length of the string that the library will provide is 30 bytes. Therefore the application shall pass a pointer to a buffer of a minimum size of 30 bytes. This command is optional |
| Arguments | p_xa_module_obj |
| | NULL |
| | i_cmd |
| | XA_API_CMD_GET_LIB_ID_STRINGS |
| | i_idx |
| | XA_CMD_TYPE_LIB_VERSION |
| | pv_value |
| | Pointer to a character buffer in which the version of the library is returned. |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | pv_value is NULL. |
| Restrictions | - |

Example
char lib_version[30];
res = (*api_func)(NULL,
                  XA_API_CMD_GET_LIB_ID_STRINGS,
                  XA_CMD_TYPE_LIB_VERSION,
                  (pVOID) lib_version);

| Subcommand | XA_CMD_TYPE_API_VERSION | |
|---|---|---|
| Description | This command obtains the version of the API in the form of a string. The maximum length of the string that the library will provide is 30 bytes. Therefore the application shall pass a pointer to a buffer of a minimum size of 30 bytes. This command is optional. | |
| Arguments | p_xa_module_obj | |
| | NULL | |
| | i_cmd | |
| | XA_API_CMD_GET_LIB_ID_STRINGS | |
| | i_idx | |
| | XA_CMD_TYPE_API_VERSION | |
| | pv_value | |
| | Pointer to a character buffer in which the version of the API is returned. | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | pv_value is NULL. |
| Restrictions | - | |

Example
```
char api_version[30];
res = (*api_func)(NULL,
                XA_API_CMD_GET_LIB_ID_STRINGS,
                XA_CMD_TYPE_API_VERSION,
                (pVOID) api_version);
```

### 2.2.2.2 XA_API_CMD_GET_API_SIZE command

| Subcommand | (None) | |
|---|---|---|
| Description | This command is used to obtain the size of the API structure, in order to allocate memory for the API structure. | |
| Arguments | p_xa_module_obj | |
| | NULL | |
| | i_cmd | |
| | XA_API_CMD_GET_API_SIZE | |
| | i_idx | |
| | NULL | |
| | pv_value | |
| | Pointer to API size variable. | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | pv_value is NULL. |
| Restrictions | The application shall allocate memory with an alignment of 4 bytes. | |

Example
WORD32 api_size;
res = (*api_func)(api_obj,
                XA_CMD_TYPE_API_SIZE,
                0,
                &api_size);

### 2.2.2.3 XA_API_CMD_INIT command

| Subcommand | XA_CMD_TYPE_INIT_API_PRE_CONFIG_PARAMS | |
|---|---|---|
| Description | This command is used to set the default value of the configuration parameters. | |
| Arguments | p_xa_module_obj<br><br>Pointer to API Structure. | |
| | i_cmd<br><br>XA_API_CMD_INIT | |
| | i_idx<br><br>XA_CMD_TYPE_INIT_API_PRE_CONFIG_PARAMS | |
| | pv_value<br><br>NULL | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| Restrictions | - | |

Example
```
res = (*api_func)(api_obj,
              XA_API_CMD_INIT,
              XA_CMD_TYPE_INIT_API_PRE_CONFIG_PARAMS,
              NULL);
```

| Subcommand | XA_CMD_TYPE_INIT_API_POST_CONFIG_PARAMS | |
|---|---|---|
| Description | This command is used to calculate the sizes of all the memory blocks required by the application. It should occur after the plugin specific parameters have been set. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_INIT | |
| | i_idx | |
| | XA_CMD_TYPE_INIT_API_POST_CONFIG_PARAMS | |
| | pv_value | |
| | NULL | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE (in Capture case) XA_RDR_CONFIG_FATAL_STATE (in Renderer case) | Incorrect sequence call (i.e. call before set memory table step) |
| | XA_CAP_EXEC_FATAL_INTERNAL (in Capture case) XA_RDR_EXEC_FATAL_INTERNAL (in Renderer case) | Invalid connection device setting path (I.e. setting SRC or SSI module for both device1 and device2). |
| Restrictions | - | |

Example
res = (*api_func)(api_obj,
                XA_API_CMD_INIT,
                XA_CMD_TYPE_INIT_API_POST_CONFIG_PARAMS,
                NULL);

ADSP Reference Renderer/Capture Plugin User's Manual**2 Software** Specifications

| Subcommand | XA_CMD_TYPE_INIT_PROCESS | |
|---|---|---|
| Description | Setup for the HW operation, and initialize state and configuration structure. No output data is created during initialization. In this state, plugin will check all hardware modules. If a module is busy, plugin will try to establish connection with next available one. If all module are busy, this command will return error code and stop immediately.<br>In case of using mix function with a different plugin (\*), if user set different data format (PCM width, output sampling rate, output channel), this command will return error and stop immediately. | |
| Arguments | p_xa_module_obj<br><br>Pointer to API Structure.<br><br> | |
| | i_cmd<br><br>XA_API_CMD_INIT<br><br> | |
| | i_idx<br><br>XA_CMD_TYPE_INIT_PROCESS<br><br> | |
| | pv_value<br><br>NULL<br><br> | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_EXEC_FATAL_STATE<br>(in Capture case)<br>(XA_RDR_EXEC_FATAL_STATE<br>(in Renderer case) | Incorrect sequence call (i.e. call before post-configuration step or run pos-configuration without persistent buffer allocation, or without DTCM memory, built-in descriptor memory allocation (in case of ADMAC used)) |
| | XA_CAP_EXEC_FATAL_HW<br>(in Capture case)<br>XA_RDR_EXEC_FATAL_HW<br>(in Renderer case) | All hardware resource are not available. |
| | XA_RDR_EXEC_FATAL_FORMAT_MISMATCH<br>(only Renderer) | The configured output format is different between this plugin and the plugin it is intended to mix with. |
| Restrictions | - | |

Example
res = (*api_func)(api_obj,
                XA_API_CMD_INIT,
                XA_CMD_TYPE_INIT_PROCESS,
                NULL);

| Subcommand | XA_CMD_TYPE_INIT_DONE_QUERY | |
|---|---|---|
| Description | This command checks to see if the initialization process has completed. If it has, the flag value is set to 1; else, it is set to zero. A pointer to the flag variable is passed as an argument. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_INIT | |
| | i_idx | |
| | XA_CMD_TYPE_INIT_DONE_QUERY | |
| | pv_value | |
| | Pointer to flag that indicates the completion of initialization process | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj or pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_EXEC_FATAL_STATE (in Capture case) (XA_RDR_EXEC_FATAL_STATE (in Renderer case) | Incorrect sequence call (i.e. call before post-configuration step) |
| Restrictions | - | |

Example
WORD32 done;
res = (*api_func)(api_obj,
　　　　　　　　XA_API_CMD_INIT,
　　　　　　　　XA_CMD_TYPE_INIT_DONE_QUERY,
　　　　　　　　&done);

### 2.2.2.4 XA_API_CMD_GET_MEMTABS_SIZE command

| Subcommand | None | |
|---|---|---|
| Description | This command is used to obtain the size of the table used to hold the memory blocks required for the plugin operation. The API returns the total size of the required table. A pointer to the size variable is sent with this API command and the plugin writes the value to the variable. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_MEMTABS_SIZE | |
| | i_idx | |
| | NULL | |
| | pv_value | |
| | Pointer to memory size variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj or pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE (in Capture case) XA_RDR_CONFIG_FATAL_STATE (in Renderer case) | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 memtab_size;
res = (*api_func)(api_obj,
            XA_API_CMD_GET_MEMTABS_SIZE,
            0,
            &memtab_size);

### 2.2.2.5 XA_API_CMD_SET_MEMTABS_PTR command

| Subcommand | None | |
|---|---|---|
| Description | This command is used to set the memory structure pointer in the library to the allocated value. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_MEMTABS_PTR | |
| | i_idx | |
| | NULL | |
| | pv_value | |
| | Allocated pointer | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj or pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj or pv_value is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE (in Capture case) XA_RDR_CONFIG_FATAL_STATE (in Renderer case) | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
pVOID memtab_ptr;
res = (*api_func)(api_obj,
              XA_API_CMD_SET_MEMTABS_PTR,
              0,
              memtab_ptr);

### 2.2.2.6 XA_API_CMD_GET_N_MEMTABS command

| Subcommand | None | |
|---|---|---|
| Description | This command is used to obtain the number of memory blocks needed by the plugin. This value is used as the iteration counter for the allocation of the memory blocks. A pointer to each memory block will be placed in the previously allocated memory tables. The pointer to the variable is passed to the API and the plugin writes the value to this variable. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_N_MEMTABS | |
| | i_idx | |
| | NULL | |
| | pv_value | |
| | Pointer to variable of number of memory blocks required to be allocated | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj or pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| Restrictions | - | |

Example
WORD32 n_memtab;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_N_MEMTABS,
                0,
                &n_memtab);

### 2.2.2.7 XA_API_CMD_GET_MEM_INFO_SIZE command

| Subcommand | Memory index | |
|---|---|---|
| Description | This command obtains the size of the memory type being referred to by the index. The size in bytes is returned in the variable pointed to by the final argument. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_MEM_INFO_SIZE | |
| | i_idx | |
| | Index of the memory<br>    0 - Persistent Area<br>    1 - Input Buffer (in Renderer case)<br>    1 - Output Buffer (in Capture case)]<br>    2 - D-TCM buffer<br>    3 - Built-in memory | |
| | pv_value | |
| | Pointer to memory size. | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj or pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned 4 bytes |
| | XA_API_FATAL_INVALID_CMD_TYPE | Incorrect index |
| | XA_CAP_CONFIG_FATAL_STATE (in Capture case) XA_RDR_CONFIG_FATAL_STATE (in Renderer case) | Incorrect sequence call (i.e. call before post-configuration step) |
| Restrictions | The index of D-TCM and built-in memory are only valid when 1st DMA device is ADMAC. | |

```
WORD32 mem_size;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_MEM_INFO_SIZE,
                index,
                &mem_size);
```

### 2.2.2.8 XA_API_CMD_GET_MEM_INFO_ALIGNMENT command

| Subcommand | Memory index | |
|---|---|---|
| Description | This command gets the alignment information of the memory-type being referred to by the index. The alignment required in bytes is returned to the application. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_MEM_INFO_ALIGNMENT | |
| | i_idx | |
| | Index of the memory<br>    0 - Persistent Area<br>    1 - Input Buffer (in Renderer case)<br>    1 - Output Buffer (in Capture case)]<br>    2 - D-TCM buffer<br>    3 - Built-in memory | |
| | pv_value | |
| | Pointer to the alignment info variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj or pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned 4 bytes |
| | XA_API_FATAL_INVALID_CMD_TYPE | Incorrect index |
| | XA_CAP_CONFIG_FATAL_STATE<br>(in Capture case)<br>XA_RDR_CONFIG_FATAL_STATE<br>(in Renderer case) | Incorrect sequence call<br>(i.e. call before post-configuration step) |
| Restrictions | The index of D-TCM and built-in memory are only valid when 1st DMA device is ADMAC. | |

Example
WORD32 mem_align;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_MEM_INFO_ALIGNMENT,
                index,
                &mem_align);

### 2.2.2.9 XA_API_CMD_GET_MEM_INFO_TYPE command

| Subcommand | Memory index | |
|---|---|---|
| Description | This command gets the alignment information of the memory-type being referred to by the index. The alignment required in bytes is returned to the application. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_MEM_INFO_TYPE | |
| | i_idx | |
| | Index of the memory<br>0 - Persistent Area<br>1 - Input Buffer (in renderer case)<br>1 - Output Buffer (in capture case)<br>2 - D-TCM buffer<br>3 - Built-in memory | |
| | pv_value | |
| | Pointer to the memory type variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj or pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned 4 bytes |
| | XA_API_FATAL_INVALID_CMD_TYPE | Incorrect index |
| | XA_CAP_CONFIG_FATAL_STATE (in Capture case) XA_RDR_CONFIG_FATAL_STATE (in Renderer case) | Incorrect sequence call (i.e. call before post-configuration step) |
| Restrictions | The index of D-TCM and built-in memory are only valid when 1st DMA device is ADMAC. | |

Example
WORD32 mem_type;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_MEM_INFO_TYPE,
                index,
                &mem_type);

## 2.2.2.10 XA_API_CMD_SET_MEM_PTR command

| Subcommand | Memory index | |
|---|---|---|
| Description | This command passes to the plugin the pointer to the allocated memory. This is then stored in the memory tables structure allocated earlier. For the input and output buffers, it is legitimate to execute this command during the main plugin loop. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_MEM_PTR | |
| | i_idx | |
| | Index of the memory<br>    0 - Persistent Area<br>    1 - Input Buffer ( renderer case)<br>    1 - Output Buffer ( capture case)<br>    2 - D-TCM buffer<br>    3 - Built-in memory | |
| | pv_value | |
| | Pointer to the memory block | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj or pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. pv_value is not aligned to required alignment for the requested memory block. |
| | XA_API_FATAL_INVALID_CMD_TYPE | Incorrect index |
| | XA_CAP_CONFIG_FATAL_STATE (in Capture case) XA_RDR_CONFIG_FATAL_STATE (in Renderer case) | Incorrect sequence call (i.e. call before post-configuration step) |
| Restrictions | The pointer must be correctly aligned to the requirements.<br>The index of D-TCM and built-in memory are only valid when 1st DMA device is ADMAC. | |

Example
pVOID addr;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_MEM_PTR,
                index,
                addr);

### 2.2.2.11    XA_API_CMD_INPUT_OVER command

| Subcommand | None | |
|---|---|---|
| Description | This command is used to tell the plugin that the input signal is over. The execution or initialization step will continue in loop until it all the remaining input data is processed. | |
| Arguments | p_xa_module_obj<br><br>Pointer to API Structure. | |
| | i_cmd<br><br>XA_API_CMD_INPUT_OVER | |
| | i_idx<br><br>  NULL | |
| | pv_value<br><br>  NULL | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_EXEC_FATAL_STATE (only for Capture) | Incorrect sequence call (i.e. call before initialization step – init process) |
| Restrictions | - | |

Example

```
res = (*api_func)(api_obj,
                XA_API_CMD_SET_INPUT_OVER,
                0,
                NULL);
```

### 2.2.2.12 XA_API_CMD_SET_INPUT_BYTES command

| Subcommand | None | |
|---|---|---|
| Description | In capture this command will do nothing. The purpose of this command is filled the full list of standard API. In renderer this command will set number of bytes available in the input buffer. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_INPUT_BYTES | |
| | i_idx | |
| | NULL | |
| | pv_value | |
| | Pointer to the input byte variable (Any value is OK with Capture case) | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj or pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes in renderer case. |
| | XA_RDR_EXEC_FATAL_INPUT (only for Renderer) | Invalid input buffer size (i.e. minus buffer size or buffer size is not align with sample size) |
| | XA_RDR_EXEC_FATAL_STATE (only for Renderer) | Input memory is not allocated before or incorrect sequence call |
| Restrictions | - | |

Example
WORD32 filled;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_INPUT_BYTES,
                0,
                &filled);

### 2.2.2.13    XA_API_CMD_GET_CURIDX_INPUT_BUF command

| Subcommand | None | |
|---|---|---|
| Description | In capture, this command will return value 0 each time it's called<br>In renderer, this command will return number of input buffer bytes consumed | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CURIDX_INPUT_BUF | |
| | i_idx | |
| | NULL | |
| | pv_value | |
| | Pointer to number variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj or pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_EXEC_FATAL_STATE<br>(only for Renderer) | Input memory is not allocated before |
| Restrictions | - | |

Example
WORD32 consumed;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CURIDX_INPUT_BUF,
                0,
                &consumed);

### 2.2.2.14 XA_API_CMD_EXECUTE command

| Subcommand | XA_CMD_TYPE_DO_EXECUTE | |
|---|---|---|
| Description | This command executes the capture. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_EXECUTE | |
| | i_idx | |
| | XA_CMD_TYPE_DO_EXECUTE | |
| | pv_value | |
| | NULL | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_EXEC_FATAL_STATE (only implement in Capture case) | Incorrect sequence call (i.e. call before initialization step) Or output memory is not allocated before |
| Restrictions | - | |

Example
res = (*api_func)(api_obj,
              XA_API_CMD_EXECUTE,
              XA_CMD_TYPE_DO_EXECUTE,
              NULL);

| Subcommand | XA_CMD_TYPE_DONE_QUERY |
|---|---|
| Description | This command checks to see if the end of processing has been reached. If it is, the flag value is set to 1; else, it is set to zero. The pointer to the flag is passed as an argument. Processing by the plugin can continue for several invocations of the DO_EXECUTE command after the last input data has been passed to the plugin, so the application should not assume that the plugin has finished generating all its output until so indicated by this command. |
| Arguments | p_xa_module_obj |
| | Pointer to API Structure. |
| | i_cmd |
| | XA_API_CMD_EXECUTE |
| | i_idx |
| | XA_CMD_TYPE_DONE_QUERY |
| | pv_value |
| | Pointer to the flag variable |

| Return value | XA_NO_ERROR | Normally ends. |
|---|---|---|
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj or pv_value is NULL. |
| | XA_RDR_EXEC_FATAL_STATE (only implement in Renderer) | Incorrect sequence call (i.e. call before initialization step) |
| Restrictions | - | |

Example
WORD32 done;
res = (*api_func)(api_obj,
                XA_API_CMD_EXECUTE,
                XA_CMD_TYPE_DONE_QUERY,
                &done);

RENESAS

### 2.2.2.15 XA_API_CMD_GET_OUTPUT_BYTES command

| Subcommand | None | |
|---|---|---|
| Description | This command obtains the number of bytes output by the plugin during the last execution. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_OUTPUT_BYTES | |
| | i_idx | |
| | NULL | |
| | pv_value | |
| | Pointer to the flag variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj or pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_EXEC_FATAL_STATE (only Capture) | Incorrect sequence call (i.e. call before initialization step) or output memory is not allocated before |
| Restrictions | - | |

Example
WORD32 produced;
res = (*api_func)(api_obj,
                  XA_API_CMD_GET_OUTPUT_BYTES,
                  0,
                  &produced);

### 2.2.2.16 XA_API_CMD_SET_CONFIG_PARAM command

#### 2.2.2.16.1 Set command for renderer

| Subcommand | XA_RDR_CONFIG_PARAM_PCM_WIDTH | |
|---|---|---|
| Description | Set the PCM sample bit width to 16 or 24 bits | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_PCM_WIDTH | |
| | pv_value | |
| | Pointer to the sample bit width variable<br>Valid value: 16 or 24<br>Default value: 16 | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_RDR_CONFIG_NONFATAL_ERR_PCM_WIDTH | PCM sample width size is not valid |
| | XA_RDR_CONFIG_FATAL_ERR_MONO_24BIT | Setting is invalid |
| Restrictions | - | |

Example
WORD32 pcm_width;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_PCM_WIDTH,
                &pcm_width);

| Subcommand | XA_RDR_CONFIG_PARAM_CHANNELS | |
|---|---|---|
| Description | Set the input PCM channels number | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_CHANNELS | |
| | pv_value | |
| | Pointer to the input PCM channels variable<br>Valid value: 1 (monaural), 2 (stereo), 4 (4 channels), 6 (6 channels), 8 (8 channels)<br>Default value: 2 (stereo) | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_RDR_CONFIG_NONFATAL_ERR_CHANNELS | Input PCM channels is invalid |
| | XA_RDR_CONFIG_FATAL_ERR_MONO_24BIT | Setting is invalid |
| Restrictions | | |

Example
WORD32 ch;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_CHANNELS,
                &ch);

| Subcommand | XA_RDR_CONFIG_PARAM_SAMPLE_RATE | |
|---|---|---|
| Description | Set the PCM sampling frequency | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_SAMPLE_RATE | |
| | pv_value | |
| | Pointer to the input sampling frequency variable<br>Valid value: 32,000 / 44,100 / 48,000 Hz<br>Default value: 44,100 Hz | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_RDR_CONFIG_NONFATAL_ERR_SAMPLE_RATE | Input PCM sampling frequency is invalid |
| Restrictions | - | |

Example
WORD32 sample_rate;
res = (*api_func)(api_obj,
                  XA_API_CMD_SET_CONFIG_PARAM,
                  XA_RDR_CONFIG_PARAM_SAMPLE_RATE,
                  &sample_rate);

| Subcommand | XA_RDR_CONFIG_PARAM_FRAME_SIZE | |
|---|---|---|
| Description | Set the PCM frame size | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_FRAME_SIZE | |
| | pv_value | |
| | Pointer to the input frame size variable<br>Valid value: frame size is power of two value<br>Default value: 1024 | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_RDR_CONFIG_NONFATAL_ERR_FRAME_SIZE | PCM frame size value is not the power of two. |
| Restrictions | - | |

Example
WORD32 frame_size;
res = (*api_func)(api_obj,
                  XA_API_CMD_SET_CONFIG_PARAM,
                  XA_RDR_CONFIG_PARAM_FRAME_SIZE,
                  &frame_size);

| Subcommand | XA_RDR_CONFIG_PARAM_OUTPUT1 | |
|---|---|---|
| Description | Set 1st output destination device for Renderer | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_OUTPUT1 | |
| | pv_value | |
| | Pointer to the output device index<br>Default value: 0 (use SSI0) | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_RDR_CONFIG_NONFATAL_ERR_SOURCE | The output source is invalid |
| Restrictions | - | |

[Note] Value range of output source relates to the PDMA and ADMAC source destination enum. It is presented below:

SSI module index: SSI_MODULE_MIN (0) <= output_source <= SSI_MODULE_MAX (97)

SCU SRC module index: SCU_SRC_INPUT_MODULE_MIN (110) <= output_source <= SCU_SRC_INPUT_MODULE_MAX (119)

Other index: reserved, not used.

The output source information is necessary. Default value is 140 means that it is fault if output source is not set by user.
Example
WORD32 output_source;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_OUTPUT1,
                &output_source);

| Subcommand | XA_RDR_CONFIG_PARAM_DMACHANNEL1 | |
|---|---|---|
| Description | Set ADMA channel number usage for 1st Audio device. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_DMACHANNEL1 | |
| | pv_value | |
| | Pointer to the Audio-DMAC / Audio-DMAC-peripheral-peripheral channels number<br><br>Valid value:<br>ADMAC_CH[29-60]     : Use Audio-DMAC to transfer<br>ADMACPP_CH[00-28]   : Use Audio-DMAC-pp extended to transfer<br><br>Default value: 0 - ADMACPP_CH00 | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_RDR_CONFIG_NONFATAL_ERR_DMACHANNEL | DMA channel is invalid |
| Restrictions | - | |

Example
WORD32 dma_channel;
res = (*api_func)(api_obj,
          XA_API_CMD_SET_CONFIG_PARAM,
          XA_RDR_CONFIG_PARAM_DMACHANNEL1,
          &dma_channel);

| Subcommand | XA_RDR_CONFIG_PARAM_OUTPUT2 | |
|---|---|---|
| Description | Set 2nd output destination device for Renderer | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_OUTPUT2 | |
| | pv_value | |
| | Pointer to the output device<br>Default value: 140 (the 2nd output device is not used) | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_RDR_CONFIG_NONFATAL_ERR_SOURCE | Output source is invalid |
| Restrictions | - | |

[Note] Value range of output source relates to the PDMA and ADMAC source destination enum. It is presented below:

SSI module index: SSI_MODULE_MIN (0) <= output_source <= SSI_MODULE_MAX (97)

SCU SRC module index: SCU_SRC_INPUT_MODULE_MIN (110) <= output_source <= SCU_SRC_INPUT_MODULE_MAX (119)

Other index: reserved, not used.

The output source information is necessary. Default value is 140 means that it is fault if output source is not set by user.
Example
WORD32 output_source;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_OUTPUT2,
                &output_source);

| Subcommand | XA_RDR_CONFIG_PARAM_DMACHANNEL2 | |
|---|---|---|
| Description | Set ADMA channel number usage for 2nd Audio device. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_DMACHANNEL2 | |
| | pv_value | |
| | Pointer to the Audio-DMAC / Audio-DMAC-peripheral-peripheral channels number<br><br>Valid value:<br>ADMAC_CH[29-60]          : Use Audio-DMAC to transfer<br>ADMACPP_CH[00-28]      : Use Audio-DMAC-pp extended to transfer<br><br>Default value: 1 - ADMACPP_CH01 | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_RDR_CONFIG_NONFATAL_ERR_DMACHANNEL | DMA channel is invalid |
| Restrictions | - | |

Example
WORD32 dma_channel;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_DMACHANNEL2,
                &dma_channel);

RENESAS

| Subcommand | XA_RDR_CONFIG_PARAM_OUT_SAMPLE_RATE | |
|---|---|---|
| Description | Set output sample rate in Sampling Rate Converter (SRC) of Sampling Rate Converter Unit (SCU). If this setting value is different from input sample rate of PCM, SRC connection will be enabled even without setting connection device path. And the connection will automatically use the available Audio-DMAC channel. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_OUT_SAMPLE_RATE | |
| | pv_value | |
| | Pointer to the output sampling frequency variable<br><br>Valid value: 32,000 / 44,100 / 48,000 (Hz)<br>Default value: 44,100 Hz | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_RDR_CONFIG_NONFATAL_ERR_SAMPLE_RATE | Output sample rate is invalid |
| Restrictions | - | |

Example:
WORD32 sample_rate;
res = (*api_func)(api_obj,
        XA_API_CMD_SET_CONFIG_PARAM,
        XA_RDR_CONFIG_PARAM_OUT_SAMPLE_RATE,
        &sample_rate);

# ADSP Reference Renderer/Capture Plugin User's Manual**2 Software** Specifications

| Subcommand | XA_RDR_CONFIG_PARAM_VOLUME_RATE | |
|---|---|---|
| Description | Set the output PCM volume rate in Digital Volume and Mute Function (DVC), and Mixer (MIX) (if mix function is used) of Sampling Rate Converter Unit (SCU). Any set value except for 0xFFFF FFFF (disable DVC) enables DVC of SCU module and the connection will be established even without setting connection path. This command can be set during plugin execution to update the output volume rate. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_VOLUME_RATE | |
| | pv_value | |
| | Pointer to the volume ratio number (using Fix-point Q3.20)<br><br>Valid value:<br>0xFFFF FFFF    : disable DVC module<br>[0, 0x7F FFFF]   : setting volume rate value<br><br>Default value: 0xFFFF FFFF | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| | XA_RDR_CONFIG_NONFATAL_VOLUME_RATE | Volume rate value is invalid |
| Restrictions | To use volume update function, DVC must be enabled in advance (before Renderer goes to post-configuration state). | |

Example:
```
WORD32 vol_rate;
res = (*api_func)(api_obj,
            XA_API_CMD_SET_CONFIG_PARAM,
            XA_RDR_CONFIG_PARAM_VOLUME_RATE,
            &vol_rate);
```

| Subcommand | XA_RDR_CONFIG_PARAM_OUT_CHANNELS | |
|---|---|---|
| Description | Set the output PCM channels number. If this setting is difference between input and output channel, the CTU module will be enabled to perform channel transfer. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_OUT_CHANNELS | |
| | pv_value | |
| | Pointer to the output PCM channels variable<br><br>Valid value: 1 (monaural), 2 (stereo)<br>Default value: 2 (stereo) | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_RDR_CONFIG_NONFATAL_ERR_CHANNELS | Output PCM channels is invalid |
| | XA_RDR_CONFIG_FATAL_ERR_MONO_24BIT | Setting is invalid |
| Restrictions | | |

Example
WORD32 ch;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_OUT_CHANNELS,
                &ch_out);

RENESAS

| Subcommand | XA_RDR_CONFIG_PARAM_MIX_CONTROL | |
|---|---|---|
| Description | Set the MIX module usage for Renderer | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_MIX_CONTROL | |
| | pv_value | |
| | Pointer to the MIX control flag variable<br><br>Valid value:   0 (not use mix function), 1 (use mix function)<br>Default value: 0 | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_RDR_CONFIG_NONFATAL_MIX_CONTROL | The setting value is invalid |
| Restrictions | | |

Example
WORD32 ch;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_MIX_CONTROL,
                &mix_ctl);

| Subcommand | XA_RDR_CONFIG_PARAM_STATE |
|---|---|
| Description | Set the operation state for Renderer plugin. Renderer can change from RUN to PAUSE, from RUN to IDLE, from PAUSE to IDLE, and from PAUSE to RUN during plugin execution. This command only has effect when Renderer is in the execution state. Otherwise, it will do nothing and return no error. |
| Arguments | p_xa_module_obj |
| | Pointer to API Structure. |
| | i_cmd |
| | XA_API_CMD_SET_CONFIG_PARAM |
| | i_idx |
| | XA_RDR_CONFIG_PARAM_STATE |
| | pv_value |
| | Pointer to the expected state of plugin

Valid value:
0 - XA_RDR_STATE_RUN: execute data to output
1 - XA_RDR_STATE_IDLE: stop and destroy plugin
2 - XA_RDR_STATE_PAUSE: pause the executing data to output

Default value: 0 |

| Return value | XA_NO_ERROR | Normally ends. |
|---|---|---|
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| | XA_RDR_CONFIG_NONFATAL_OPERATION | The setting value is invalid |
| Restrictions | |

Example
WORD32 ch;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_STATE,
                &state);

RENESAS

2.2.2.16.2 Set command for capture

| Subcommand | XA_CAP_CONFIG_PARAM_PCM_WIDTH | |
|---|---|---|
| Description | Set the PCM sample bit width to 16 or 24 bits | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_PCM_WIDTH | |
| | pv_value | |
| | Pointer to the sample bit width variable<br><br>Valid value: 16 or 24<br>Default value: 16 | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_CAP_CONFIG_NONFATAL_ERR_PCM_WIDTH | PCM sample width size is not valid |
| | XA_CAP_CONFIG_FATAL_ERR_MONO_24BIT | Setting is invalid |
| Restrictions | - | |

Example
WORD32 pcm_width;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_CONFIG_PARAM,
                XA_CAP_CONFIG_PARAM_PCM_WIDTH,
                &pcm_width);

RENESAS

| Subcommand | XA_CAP_CONFIG_PARAM_CHANNELS | |
|---|---|---|
| Description | Set the PCM channels number | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_CHANNELS | |
| | pv_value | |
| | Pointer to the input channels variable

Valid value: 1 (monaural), 2 (stereo)
Default value: 2 (stereo) | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_CAP_CONFIG_NONFATAL_ERR_CHANNELS | PCM input channels is invalid |
| | XA_CAP_CONFIG_FATAL_ERR_MONO_24BIT | Setting is invalid |
| Restrictions | | |

Example
WORD32 ch;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_CONFIG_PARAM,
                XA_CAP_CONFIG_PARAM_CHANNELS,
                &ch);

| Subcommand | XA_CAP_CONFIG_PARAM_SAMPLE_RATE | |
|---|---|---|
| Description | Set the PCM sampling frequency | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_SAMPLE_RATE | |
| | pv_value | |
| | Pointer to the input sampling frequency variable<br><br>Valid value: 32,000 / 44,100 / 48,000 (Hz)<br>Default value: 44,100 | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_CAP_CONFIG_NONFATAL_ERR_SAMPLE_RATE | PCM input sampling frequency is invalid |
| Restrictions | - | |

Example
WORD32 sample_rate;
res = (*api_func)(api_obj,
            XA_API_CMD_SET_CONFIG_PARAM,
            XA_CAP_CONFIG_PARAM_SAMPLE_RATE,
            &sample_rate);

| Subcommand | XA_CAP_CONFIG_PARAM_FRAME_SIZE | |
|---|---|---|
| Description | Set the PCM frame size | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_FRAME_SIZE | |
| | pv_value | |
| | Pointer to the input frame size variable<br><br>Valid value: frame size is power of two value<br>Default value: 1024 | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_CAP_CONFIG_NONFATAL_ERR_FRAME_SIZE | PCM frame size value is not the power of two. |
| Restrictions | - | |

Example
WORD32 frame_size;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_CONFIG_PARAM,
                XA_CAP_CONFIG_PARAM_FRAME_SIZE,
                &frame_size);

| Subcommand | XA_CAP_CONFIG_PARAM_INPUT1 | |
|---|---|---|
| Description | Set 1st input device for Capture | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_INPUT1 | |
| | pv_value | |
| | Pointer to the input source value<br>Default value: 10 (SSI1) | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_CAP_CONFIG_NONFATAL_ERR_SOURCE | PCM input source is invalid |
| Restrictions | - | |

[Note] Value range of input source relates to the PDMA and ADMAC source destination enum. It is presented below:

SSI module index: SSI_MODULE_MIN (0) <= output_source <= SSI_MODULE_MAX (97)

SCU SRC module index: SCU_SRC_INPUT_MODULE_MIN (110) <= output_source <= SCU_SRC_INPUT_MODULE_MAX (119)

Other index: reserved, not used.

The output source information is necessary. Default value is 140 means that it is fault if output source is not set by user.
Example
WORD32 input_source;
res = (*api_func)(api_obj,
            XA_API_CMD_SET_CONFIG_PARAM,
            XA_CAP_CONFIG_PARAM_INPUT1,
            &input_source);

| Subcommand | XA_CAP_CONFIG_PARAM_DMACHANNEL1 | |
|---|---|---|
| Description | Set ADMA channel number usage for 1st Audio device. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_DMACHANNEL1 | |
| | pv_value | |
| | Pointer to the ADMA channels number<br><br>Valid value:<br>ADMAC_CH[29-60] : Use Audio-DMAC to transfer<br>ADMACPP_CH[0-28] : Use Audio-DMAC-pp extended to transfer<br><br>Default value: ADMACPP_CH10 | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_CAP_CONFIG_NONFATAL_ERR_DMACHANNEL | PCM DMA channel is invalid |
| Restrictions | - | |

Example
WORD32 dma_channel;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_CONFIG_PARAM,
                XA_CAP_CONFIG_PARAM_DMACHANNEL1,
                &dma_channel);

ADSP Reference Renderer/Capture Plugin User's Manual**2 Software** Specifications

| Subcommand | XA_CAP_CONFIG_PARAM_INPUT2 | |
|---|---|---|
| Description | Set 2nd input device for Capture | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_INPUT2 | |
| | pv_value | |
| | Pointer to the input source value<br>Default value: 140 (the 2nd input device is not used) | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_CAP_CONFIG_NONFATAL_ERR_SOURCE | PCM input source is invalid |
| Restrictions | - | |

[Note] Value range of input source relates to the PDMA and ADMAC source destination enum. It is presented below:

SSI module index: SSI_MODULE_MIN (0) <= output_source <= SSI_MODULE_MAX (97)

SCU SRC module index: SCU_SRC_INPUT_MODULE_MIN (110) <= output_source <= SCU_SRC_INPUT_MODULE_MAX (119)

Other index: reserved, not used.

The output source information is necessary. Default value is 140 means that it is fault if output source is not set by user.
Example
WORD32 input_source;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_CONFIG_PARAM,
                XA_CAP_CONFIG_PARAM_INPUT2,
                &input_source);

| Subcommand | XA_CAP_CONFIG_PARAM_DMACHANNEL2 | |
|---|---|---|
| Description | Set ADMA channel number usage for 2$^{nd}$ Audio device. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_DMACHANNEL2 | |
| | pv_value | |
| | Pointer to the ADMA channels number<br><br>Valid value:<br>ADMAC_CH[0-31]          : Audio-DMAC usage channel 0 -31<br>ADMACPP_CH[0-28]       : Audio-DMAC-pp extended usage channel 0-28<br><br>Default value: ADMACPP_CH11 | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_CAP_CONFIG_NONFATAL_ERR_DMACHANNEL | PCM DMA channel is invalid |
| Restrictions | - | |

Example
WORD32 dma_channel;
res = (*api_func)(api_obj,
              XA_API_CMD_SET_CONFIG_PARAM,
              XA_CAP_CONFIG_PARAM_DMACHANNEL2,
              &dma_channel);

# ADSP Reference Renderer/Capture Plugin User's Manual**2 Software** Specifications

| Subcommand | XA_CAP_CONFIG_PARAM_OUT_SAMPLE_RATE | |
|---|---|---|
| Description | Set output sample rate in Sampling Rate Converter (SRC) of Sampling Rate Converter Unit (SCU). If this setting value is different from input sample rate of PCM, SRC connection will be enabled even without setting connection device path. And the connection will automatically use the available Audio-DMAC channel. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_OUT_SAMPLE_RATE | |
| | pv_value | |
| | Pointer to the output sampling frequency variable<br><br>Valid value: 32,000 / 44,100 / 48,000 (Hz)<br>Default value: 44,100 | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step and after post-configuration step) |
| | XA_CAP_CONFIG_NONFATAL_ERR_SAMPLE_RATE | PCM output sample rate is invalid |
| Restrictions | - | |

Example:
WORD32 sample_rate;
res = (*api_func)(api_obj,
              XA_API_CMD_SET_CONFIG_PARAM,
              XA_CAP_CONFIG_PARAM_OUT_SAMPLE_RATE,
              &sample_rate);

| Subcommand | XA_CAP_CONFIG_PARAM_VOLUME_RATE | |
|---|---|---|
| Description | Set the output PCM volume rate in Digital Volume and Mute Function (DVC) of Sampling Rate Converter Unit (SCU). Any set value other than 0xFFFF FFFF (disable DVC) enables DVC of SCU module and the connection will be established even without setting connection path. This command can be set during plugin execution to update the output volume rate. | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_SET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_VOLUME_RATE | |
| | pv_value | |
| | Pointer to the volume ratio number (using Fix-point Q3.20)<br><br>Valid value:<br>0xFFFF FFFF    : disable DVC module<br>[0, 0x7F FFFF]   : setting volume rate value<br><br>Default value: 0xFFFF FFFF | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| | XA_CAP_CONFIG_NONFATAL_VOLUME_RATE | PCM volume rate value is invalid |
| Restrictions | To use volume update function, DVC must be enabled in advance (before Capture goes to post-configuration state). | |

Example:
WORD32 vol_rate;
res = (*api_func)(api_obj,
           XA_API_CMD_SET_CONFIG_PARAM,
           XA_CAP_CONFIG_PARAM_VOLUME_RATE,
           &vol_rate);

| Subcommand | XA_CAP_CONFIG_PARAM_STATE |
|---|---|
| Description | Set the operation state for Capture plugin. Capture can change from RUN to PAUSE, from RUN to IDLE, from PAUSE to IDLE, and from PAUSE to RUN during plugin execution. This command only has effect when Capture is in the execution state. Otherwise, it will do nothing and return no error. |
| Arguments | p_xa_module_obj<br><br>Pointer to API Structure.<br><br><br>i_cmd<br><br>XA_API_CMD_SET_CONFIG_PARAM<br><br><br>i_idx<br><br>XA_CAP_CONFIG_PARAM_STATE<br><br><br>pv_value<br><br>Pointer to the expected state of plugin<br><br>Valid value:<br>0 - XA_CAP_STATE_RUN: execute data to output<br>1 - XA_CAP_STATE_IDLE: stop and destroy plugin<br>2 - XA_CAP_STATE_PAUSE: pause the executing data to output<br><br>Default value: 0 |

| Return value | XA_NO_ERROR | Normally ends. |
|---|---|---|
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| | XA_CAP_CONFIG_NONFATAL_OPERATION | The setting value is invalid |
| Restrictions | | |

Example
WORD32 ch;
res = (*api_func)(api_obj,
                XA_API_CMD_SET_CONFIG_PARAM,
                XA_CAP_CONFIG_PARAM_STATE,
                &state);

### 2.2.2.17    XA_API_CMD_GET_CONFIG_PARAM command

2.2.2.17.1 Get command for renderer

| Subcommand | XA_RDR_CONFIG_PARAM_PCM_WIDTH | |
|---|---|---|
| Description | Get the PCM sample bit width settings | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_PCM_WIDTH | |
| | pv_value | |
| | Pointer to the sample bit width variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 pcm_width;
res = (*api_func)(api_obj,
           XA_API_CMD_GET_CONFIG_PARAM,
           XA_RDR_CONFIG_PARAM_PCM_WIDTH,
           &pcm_width);

| Subcommand | XA_RDR_CONFIG_PARAM_CHANNELS | |
|---|---|---|
| Description | Get the input PCM channels number setting | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_CHANNELS | |
| | pv_value | |
| | Pointer to channels variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
```
WORD32 ch;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_CHANNELS,
                &ch);
```

| Subcommand | XA_RDR_CONFIG_PARAM_SAMPLE_RATE | |
|---|---|---|
| Description | Get the PCM sampling frequency setting | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_SAMPLE_RATE | |
| | pv_value | |
| | Pointer to sampling frequency variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call<br>(i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 sample_rate;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_SAMPLE_RATE,
                &sample_rate);

| Subcommand | XA_RDR_CONFIG_PARAM_FRAME_SIZE | |
|---|---|---|
| Description | Get the PCM frame size setting | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_FRAME_SIZE | |
| | pv_value | |
| | Pointer to frame size variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 frame_size;
res = (*api_func)(api_obj,
          XA_API_CMD_GET_CONFIG_PARAM,
          XA_RDR_CONFIG_PARAM_FRAME_SIZE,
          &frame_size);

| Subcommand | XA_RDR_CONFIG_PARAM_OUTPUT1 | |
|---|---|---|
| Description | Get 1st output destination device for Renderer info | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_OUTPUT1 | |
| | pv_value | |
| | Pointer to the 1st output destination device value | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 output_source;
res = (*api_func)(api_obj,
            XA_API_CMD_GET_CONFIG_PARAM,
            XA_RDR_CONFIG_PARAM_OUTPUT1,
            &output_source);

| Subcommand | XA_RDR_CONFIG_PARAM_DMACHANNEL1 |
|---|---|
| Description | Get ADMA channel number usage for 1st Audio device info |
| Arguments | p_xa_module_obj |
| | Pointer to API Structure. |
| | i_cmd |
| | XA_API_CMD_GET_CONFIG_PARAM |
| | i_idx |
| | XA_RDR_CONFIG_PARAM_DMACHANNEL1 |
| | pv_value |
| | Pointer to the 1st Audio-DMAC / Audio-DMAC-peripheral-peripheral channel variable |

| Return value | XA_NO_ERROR | Normally ends. |
|---|---|---|
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - |

Example
WORD32 dma_channel;
res = (*api_func)(api_obj,
            XA_API_CMD_GET_CONFIG_PARAM,
            XA_RDR_CONFIG_PARAM_DMACHANNEL1,
            &dma_channel);

| Subcommand | XA_RDR_CONFIG_PARAM_OUTPUT2 | |
|---|---|---|
| Description | Get 2<sup>nd</sup> output destination device for Renderer info | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_OUTPUT2 | |
| | pv_value | |
| | Pointer to the 2nd output destination device value | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 output_source;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_OUTPUT2,
                &output_source);

| Subcommand | XA_RDR_CONFIG_PARAM_DMACHANNEL2 | |
|---|---|---|
| Description | Get ADMA channel number usage for 2nd Audio device info | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_DMACHANNEL2 | |
| | pv_value | |
| | Pointer to the 2nd Audio-DMAC / Audio-DMAC-peripheral-peripheral channel variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 dma_channel;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_DMACHANNEL2,
                &dma_channel);

| Subcommand | XA_RDR_CONFIG_PARAM_OUT_SAMPLE_RATE | |
|---|---|---|
| Description | Get output sample rate setting | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_OUT_SAMPLE_RATE | |
| | pv_value | |
| | Pointer to the output sampling frequency variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example:
WORD32 sample_rate;
res = (*api_func)(api_obj,
            XA_API_CMD_GET_CONFIG_PARAM,
            XA_RDR_CONFIG_PARAM_OUT_SAMPLE_RATE,
            &sample_rate);

| Subcommand | XA_RDR_CONFIG_PARAM_VOLUME_RATE | |
|---|---|---|
| Description | Get the output PCM volume rate setting value | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_VOLUME_RATE | |
| | pv_value | |
| | Pointer to the volume ratio number (using Fix-point Q3.20) | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example:
WORD32 vol_rate;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_VOLUME_RATE,
                &vol_rate);

| Subcommand | XA_RDR_CONFIG_PARAM_OUT_CHANNELS | |
|---|---|---|
| Description | Get the output PCM channels number setting | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_OUT_CHANNELS | |
| | pv_value | |
| | Pointer to output channels variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 ch;
res = (*api_func)(api_obj,
　　　　　　XA_API_CMD_GET_CONFIG_PARAM,
　　　　　　XA_RDR_CONFIG_PARAM_OUT_CHANNELS,
　　　　　　&ch_out);

| Subcommand | XA_RDR_CONFIG_PARAM_MIX_CONTROL | |
|---|---|---|
| Description | Get the MIX module usage of Renderer | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_MIX_CONTROL | |
| | pv_value | |
| | Pointer to Mix control flag variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 ch;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_MIX_CONTROL,
                &mix_ctl);

| Subcommand | XA_RDR_CONFIG_PARAM_STATE | |
|---|---|---|
| Description | Get the current operation state of Renderer plugin | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_RDR_CONFIG_PARAM_STATE | |
| | pv_value | |
| | Pointer to the current operation state of plugin | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_RDR_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 ch;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CONFIG_PARAM,
                XA_RDR_CONFIG_PARAM_STATE,
                &cur_state);

### 2.2.2.17.2 Get command for capture

| Subcommand | XA_CAP_CONFIG_PARAM_PCM_WIDTH | |
|---|---|---|
| Description | Get the PCM sample bit width settings | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_PCM_WIDTH | |
| | pv_value | |
| | Pointer to the sample bit width variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 pcm_width;
res = (*api_func)(api_obj,
            XA_API_CMD_GET_CONFIG_PARAM,
            XA_CAP_CONFIG_PARAM_PCM_WIDTH,
            &pcm_width);

| Subcommand | XA_CAP_CONFIG_PARAM_CHANNELS | |
|---|---|---|
| Description | Get the PCM channels number setting | |
| Arguments | p_xa_module_obj<br><br>Pointer to API Structure. | |
| | i_cmd<br><br>XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx<br><br>XA_CAP_CONFIG_PARAM_CHANNELS | |
| | pv_value | |
| | Pointer to channels variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call<br>(i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 ch;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CONFIG_PARAM,
                XA_CAP_CONFIG_PARAM_CHANNELS,
                &ch);

| Subcommand | XA_CAP_CONFIG_PARAM_SAMPLE_RATE | |
|---|---|---|
| Description | Get the PCM sampling frequency setting | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_SAMPLE_RATE | |
| | pv_value | |
| | Pointer to sampling frequency variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 sample_rate;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CONFIG_PARAM,
                XA_CAP_CONFIG_PARAM_SAMPLE_RATE,
                &sample_rate);

| Subcommand | XA_CAP_CONFIG_PARAM_FRAME_SIZE | |
|---|---|---|
| Description | Get the PCM frame size setting | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_FRAME_SIZE | |
| | pv_value | |
| | Pointer to frame size variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 frame_size;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CONFIG_PARAM,
                XA_CAP_CONFIG_PARAM_FRAME_SIZE,
                &frame_size);

| Subcommand | XA_CAP_CONFIG_PARAM_INPUT1 | |
|---|---|---|
| Description | Get 1st input source device for Capture info | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_INPUT1 | |
| | pv_value | |
| | Pointer to the 1st input device value | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 input_source;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CONFIG_PARAM,
                XA_CAP_CONFIG_PARAM_INPUT1,
                &input_source);

| Subcommand | XA_CAP_CONFIG_PARAM_DMACHANNEL1 | |
|---|---|---|
| Description | Get ADMA channel number usage for 1st Audio device info | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_DMACHANNEL1 | |
| | pv_value | |
| | Pointer to the 1st Audio-DMAC channel | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 dma_channel;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CONFIG_PARAM,
                XA_CAP_CONFIG_PARAM_DMACHANNEL1,
                &dma_channel);

| Subcommand | XA_CAP_CONFIG_PARAM_INPUT2 | |
|---|---|---|
| Description | Get 2nd input source device for Capture info | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_INPUT2 | |
| | pv_value | |
| | Pointer to the 2nd input source device value | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 input_source;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CONFIG_PARAM,
                XA_CAP_CONFIG_PARAM_INPUT2,
                &input_source);

| Subcommand | XA_CAP_CONFIG_PARAM_DMACHANNEL2 | |
|---|---|---|
| Description | Get ADMA channel number usage for 2<sup>nd</sup> Audio device info | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_DMACHANNEL2 | |
| | pv_value | |
| | Pointer to the 2nd Audio-DMAC channel | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 dma_channel;
res = (*api_func)(api_obj,
                XA_API_CMD_GET_CONFIG_PARAM,
                XA_CAP_CONFIG_PARAM_DMACHANNEL2,
                &dma_channel);

| Subcommand | XA_CAP_CONFIG_PARAM_OUT_SAMPLE_RATE | |
|---|---|---|
| Description | Get output sample rate setting value | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_OUT_SAMPLE_RATE | |
| | pv_value | |
| | Pointer to the output sampling frequency variable | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example:
WORD32 sample_rate;
res = (*api_func)(api_obj,
            XA_API_CMD_GET_CONFIG_PARAM,
            XA_CAP_CONFIG_PARAM_OUT_SAMPLE_RATE,
            &sample_rate);

| Subcommand | XA_CAP_CONFIG_PARAM_VOLUME_RATE | |
|---|---|---|
| Description | Get the output PCM volume rate setting value | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_VOLUME_RATE | |
| | pv_value | |
| | Pointer to the volume ratio number (using Fix-point Q3.20) | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

Example:
WORD32 vol_rate;
res = (*api_func)(api_obj,
            XA_API_CMD_GET_CONFIG_PARAM,
            XA_CAP_CONFIG_PARAM_VOLUME_RATE,
            &vol_rate);

| Subcommand | XA_CAP_CONFIG_PARAM_STATE | |
|---|---|---|
| Description | Get the current operation state of Capture plugin | |
| Arguments | p_xa_module_obj | |
| | Pointer to API Structure. | |
| | i_cmd | |
| | XA_API_CMD_GET_CONFIG_PARAM | |
| | i_idx | |
| | XA_CAP_CONFIG_PARAM_STATE | |
| | pv_value | |
| | Pointer to the current operation state of plugin | |
| Return value | XA_NO_ERROR | Normally ends. |
| | XA_API_FATAL_MEM_ALLOC | p_xa_module_obj / pv_value is NULL. |
| | XA_API_FATAL_MEM_ALIGN | p_xa_module_obj is not aligned to 4 bytes. |
| | XA_CAP_CONFIG_FATAL_STATE | Incorrect sequence call<br>(i.e. call before pre-configuration step) |
| Restrictions | - | |

Example
WORD32 ch;
res = (*api_func)(api_obj,
            XA_API_CMD_GET_CONFIG_PARAM,
            XA_CAP_CONFIG_PARAM_STATE,
            &cur_state);

## 2.3    Structures

Table 2-12 lists the structures for this software. The user should reserve areas required for these structures. For detailed specifications of these input structures, refer to Section 2.3.1.

Table 2-12        Structures

| Structure name | Size (bytes) | Outline |
|---|---|---|
| XARelrdr | 1432 (bytes) | API's structure to stores the information of API |
| XARelcap | 1392 (bytes) | API's structure to stores the information of API |

## 2.3.1 XARelrdr type structure

The XARelrdr type structure is the work area used by the renderer of Renderer plugin. When using this plugin, secure the area with the application program. It's not necessary to refer to this area because it only contains the internal variables and working buffers of the plugin. Make sure not to change the value of this area with the application program.

Table 2-13　　　XARelrdr type structure information

| Member name | Outline |
|---|---|
| pVOID pMem_tabs | Memory table controller |
| UWORD32 ring_num | Number of ring buffer |
| UWORD32 ring_size | Total size of ring-buffer in sample |
| UWORD32 buffer_size | Total size of ring-buffer in bytes |
| UWORD32 sample_size | Size of PCM sample |
| UWORD32 write_idx | Software writing position |
| UWORD32 read_idx | FIFO reading position |
| UWORD32 filled | Number of samples present in the buffer |
| UWORD32 submitted | Total number of submitted samples |
| UWORD32 core | Identifier of the core we are running on |
| UWORD32 state | Component state |
| xa_renderer_cb_t *cdata | Notification callback pointer |
| UWORD32 consumed | Number of samples consumed to framework |
| XosEvent relrdr_event | Renderer polling event |
| XosThread relrdr_thread | Renderer polling thread |
| UWORD32 frame_size | Number of sample in each frame |
| UWORD32 in_channels | Number of input channels |
| UWORD32 out_channels | Number of output channels |
| UWORD32 in_rate | Input sampling rate |
| UWORD32 out_rate | Output sampling rate |
| UWORD32 pcm_width | Sample width |
| UWORD32 stage_flag | Present current stage used in ADMAC |
| UWORD32 stage_size | Stage memory block size when using DMAC |
| UWORD32 stage_num | Stage number when using ADMAC |
| UWORD32 trans_num | Transfer number time when using ADMAC |
| relren_Device dev[2] | Output device info array after setting param |
| relren_Enable_Module enable_module | Module list is used |
| WORD32 volume | Volume value |
| UWORD32 mix_ctrl | Mix control value |
| UWORD32 operation_state | Operation state of plugin |
| SSIU_SSI_MODULE ssi_module | SSI module information |
| WORD32 first_run | Flag indicating whether the current stream is the first stream |

## 2.3.2    XARelcap type structure

The XARelcap type structure is the work area used by the capture of Renderer plugin. When using this plugin, secure the area with the application program. It's not necessary to refer to this area because it only contains the internal variables and working buffers of the plugin. Make sure not to change the value of this area with the application program.

Table 2-14    XARelcap type structure information

| Member name | Outline |
|---|---|
| pVOID pMem_tabs | Memory table controller |
| UWORD32 sample_size | Size of PCM sample |
| UWORD32 core | Identifier of the core we are running on |
| UWORD32 state | Component state |
| UWORD32 transfered | Number of samples have been transferred |
| XosEvent relcap_event | Capture polling event |
| XosThread relcap_thread | Capture polling thread |
| UWORD32 frame_size | Number of sample in each frame |
| UWORD32 channels | Number of channels |
| UWORD32 in_rate | Input sampling rate |
| UWORD32 out_rate | Output sample rate |
| UWORD32 pcm_width | Sample width |
| UWORD32 ring_num | Number of ring buffer |
| UWORD32 SSI0_setting_flag | Mark SSI0 setting |
| UWORD32 stage_flag | Present current stage used in ADMAC |
| UWORD32 stage_size | Stage memory block size when using DMAC |
| UWORD32 stage_num | Stage number when using DMAC |
| UWORD32 trans_num | Transfer number time when using DMAC |
| UWORD32 ring_size | Total size of ring-buffer in samples |
| UWORD32 buffer_size | Total size of ring-buffer in bytes |
| relcap_Device dev[2] | Output device info array after set param |
| relcap_Enable_module enable_module | Module list is used |
| WORD32 volume | Volume value |
| SSIU_SSI_MODULE ssi_module | SSI module information |
| UWORD32 operation_state | Operation state of plugin |
| SRC_START_MODE start_mode | Start mode when using SRC |

## 2.4 Memory Specifications

This section describes the memory areas used by this software.

## 2.4.1 Persistent Area

Table 2-15        Persistent Area Description

| Item | Area which always holds values when this software is used.<br>If the user manipulates this area after initialization, the correct execution of this software is not ensured. |
|---|---|
| Symbol name | - (freely defined by the user) |
| Size | Obtain the actually required size with 2.2.2.7 |
| Area reservation | The user should reserve this area. |
| Allocation | This area is included in RAM. |
| Alignment | Align this area on a 4-byte boundary. |

## 2.4.2 Stack Area

Table 2-16 Stack Area Description

| Item | Stack area used by this software |
|---|---|
| Symbol name | - (freely defined by the user) |
| Size | Obtain the actually required size with 2.2.2.7 |
| Area reservation | The user should reserve this area.<br>To use this software, reserve a software stack area which exceeds the size above. |
| Allocation | This area is included in RAM. |
| Alignment | - |

## 2.4.3 Heap Area

This software does not use a heap area.

## 2.4.4 Input Buffer

Input buffer only is used in the renderer case.

Table 2-17          Input Buffer Description

| Item | Area which stores inputs from this software.<br>The input buffer contains 16-bit linear PCM data (hereinafter called PCM data).<br>If the user manipulates this area during rendering processing, the normal execution of the program cannot be ensured. |
|---|---|
| Symbol name | - (freely defined by the user) |
| Size | Please secure more than size with 2.2.2.7 (a multiple of 2.2.2.7). |
| Area reservation | The user should reserve this area.<br>The user can freely use this area after the rendering of one block. |
| Allocation | This area is included in RAM. |
| Alignment | Align this area on a 4-byte boundary. |

## 2.4.5 Output Buffer

Output buffer only is used in the capture case.

Table 2-18          Output Buffer Description

| Item | Area which stores outputs from this software.<br>The output buffer contains 16-bit linear PCM data (hereinafter called PCM data).<br>If the user manipulates this area during rendering processing, the normal execution of the program cannot be ensured. |
|---|---|
| Symbol name | - (freely defined by the user) |
| Size | Please secure more than size with 2.2.2.7 (a multiple of 2.2.2.7). |
| Area reservation | The user should reserve this area.<br>The user can freely use this area after the rendering of one block. |
| Allocation | This area is included in RAM. |
| Alignment | Align this area on a 4-byte boundary. |

(1)  Input/ Output data storage method

Data is input/ output in the formats as shown in Figure 2-4(consecutive buffers are specified for the channels). The input/output buffer (memory) stores data in 2-byte (16-bit) units. The byte order for accessing the buffer is little endian (see Figure 2-2).
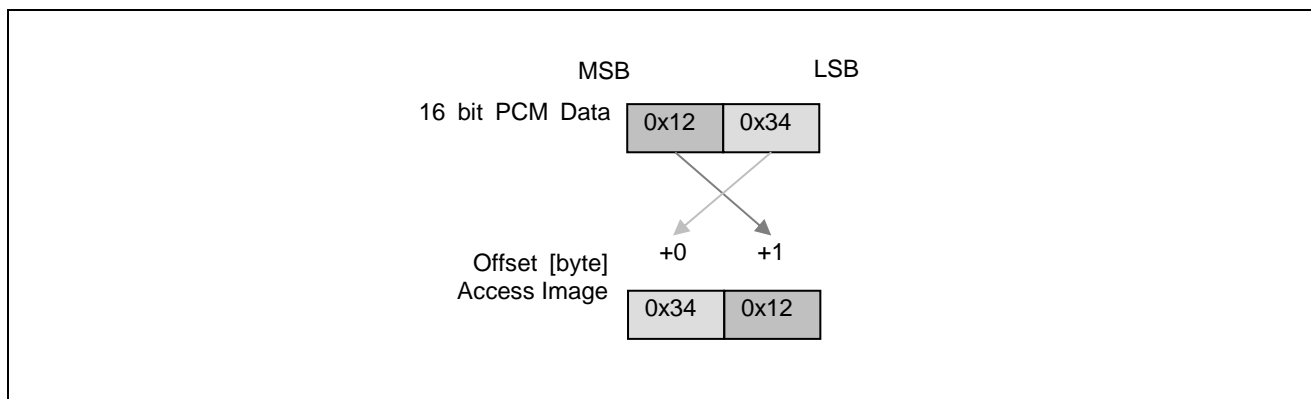


Figure 2-2        PCM 16-bit Data Access (Little Endian Mode)
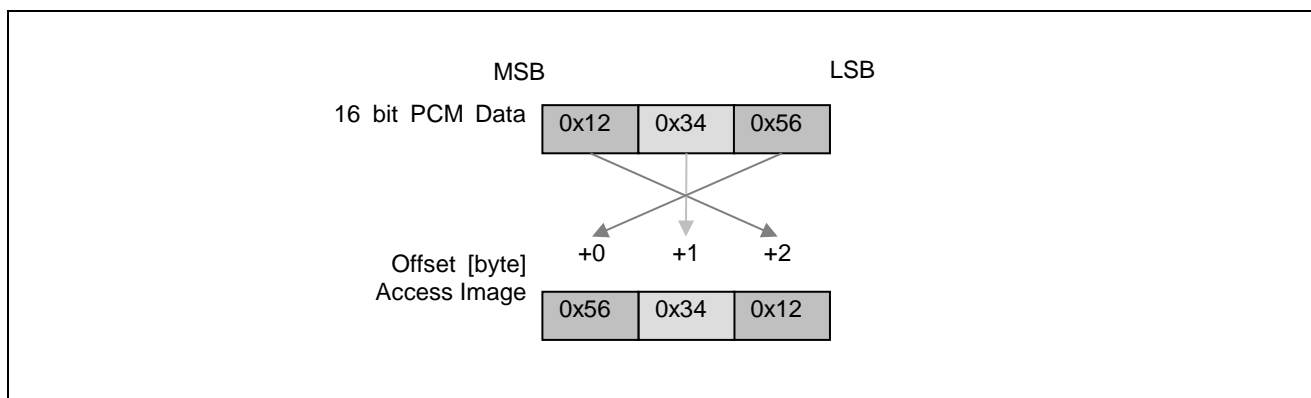


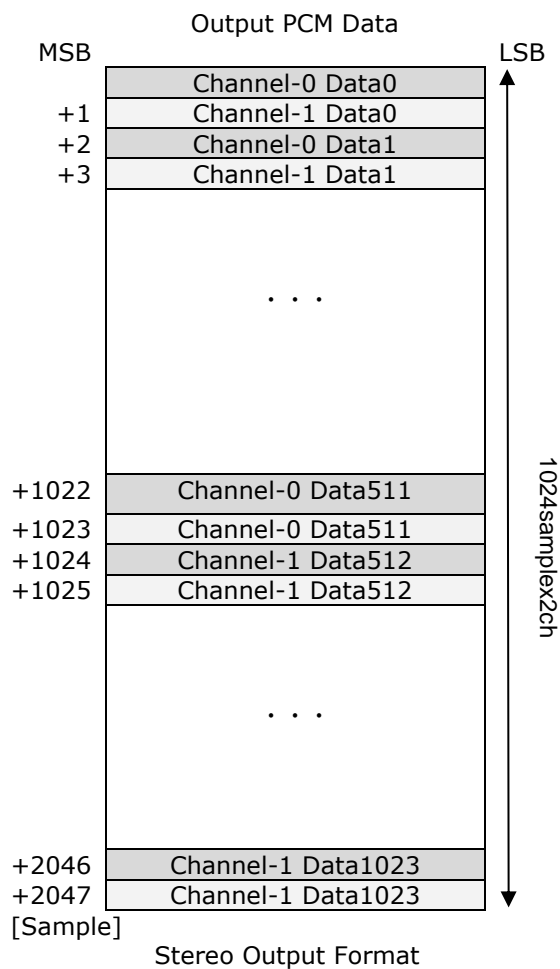Figure 2-3        PCM 24-bit Data Access (Little Endian Mode)

RENESAS

Output PCM Data

MSB                                              LSB

| | |
|---|---|
| | Channel-0 Data0 |
| +1 | Channel-1 Data0 |
| +2 | Channel-0 Data1 |
| +3 | Channel-1 Data1 |
| | . . . |
| +1022 | Channel-0 Data511 |
| +1023 | Channel-0 Data511 |
| +1024 | Channel-1 Data512 |
| +1025 | Channel-1 Data512 |
| | . . . |
| +2046 | Channel-1 Data1023 |
| +2047 | Channel-1 Data1023 |

[Sample]

1024samplex2ch

Stereo Output Format

Figure 2-4        Output Formats

## 2.5    Error Processing

This software's functions return the error codes listed in Table 2-20.

### 2.5.1        Error codes

Below are the error codes for this software.
Table 2-19        Error Codes for Renderer

| Error code (32bit) | Value | Description |
|---|---|---|
| [1]<br>XA_NO_ERROR | 0x00000000 | The processing results are normal.<br>The process has terminated normally. |
| [2]<br>XA_API_FATAL_MEM_ALLOC | 0xFFFF8000 | Abnormality has occurred, which disables process continuation. An address of API structure was specified at the argument is NULL, the program execution is incorrect.<br>Because it becomes the common API error, please check the correct procedure. |
| [3]<br>XA_API_FATAL_MEM_ALIGN | 0xFFFF8001 | Abnormality has occurred, which disables process continuation. An address of API structure was specified at the argument does not 4 byte align. Because it becomes the common API error, please check the correct procedure. |
| [4]<br>XA_API_FATAL_INVALID_CMD | 0xFFFF8002 | Abnormality has occurred, which disables process continuation. The command was specified at the argument does not support. Because it becomes the common API error, please check the correct procedure. |
| [5]<br>XA_API_FATAL_INVALID_CMD_TYPE | 0xFFFF8003 | Abnormality has occurred, which disables process continuation. The subcommand was specified at the argument does not support. Because it becomes the common API error, please check the correct procedure. |
| [6]<br>XA_RDR_CONFIG_FATAL_STATE | 0xFFFF8881 | Abnormality has occurred, which disables process continuation. The subcommand was specified at the argument does not support. Because it becomes the common API error, please check the correct procedure. |
| [7]<br>XA_RDR_CONFIG_FATAL_ERR_MONO_24BIT | 0xFFFF8883 | It is an error for invalid setting, 24 bit mono channel case does not supported. |
| [8]<br>XA_RDR_EXEC_FATAL_STATE | 0xFFFF9081 | Abnormality has occurred, which disables process continuation. The subcommand was specified at the argument does not support. Because it becomes the common API error, please check the correct procedure. |

| [7]<br>XA_RDR_EXEC_FATAL_INPUT | 0xFFFF9082 | Abnormality has occurred, which disables process continuation. The input size is not align with sample size. Because it becomes the common API error, please check the correct size of input buffers. |
|---|---|---|
| [9]<br>XA_RDR_EXEC_FATAL_INTERNAL | 0xFFFF9083 | Abnormality has occurred, which disables process continuation. Some of setting becomes incorrect after combination. Because it becomes the common API error, please check the correct parameters. |
| [10]<br>XA_RDR_EXEC_FATAL_HW | 0xFFFF9080 | Abnormality has occurred, which disables process continuation. Hardware modules are not available... Because it becomes the common API error, please check the correct parameters and make sure the resource is validity. |
| [11]<br>XA_RDR_CONFIG_NONFATAL_ERR_PCM_WIDTH | 0x00000880 | It is an error for renderer specifications out of the range. The PCM width was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [12]<br>XA_RDR_CONFIG_NONFATAL_ERR_CHANNELS | 0x00000881 | It is an error for renderer specifications out of the range. The channel numbers was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [13]<br>XA_RDR_CONFIG_NONFATAL_ERR_SAMPLE_RATE | 0x00000882 | It is an error for renderer specifications out of the range. The sample rate was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [14]<br>XA_RDR_CONFIG_NONFATAL_ERR_FRAME_SIZE | 0x00000883 | It is an error for renderer specifications out of the range. The Input buffer size was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [15]<br>XA_RDR_CONFIG_NONFATAL_ERR_SOURCE | 0x00000884 | It is an error for renderer specifications out of the range. The input buffer size was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16 |
| [16]<br>XA_RDR_CONFIG_NONFATAL_ERR_DMACHANNEL | 0x00000885 | It is an error for renderer specifications out of the range. The input buffer size was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [17]<br>XA_RDR_CONFIG_NONFATAL_VOLUME_RATE | 0x00000886 | It is an error for renderer specification out of range. The volume rate value was specified at the argument does not support. |
| [18]<br>XA_RDR_CONFIG_NONFATAL_MIX_CONTROL | 0x00000887 | It is an error for renderer specification out of range. The mix control value was specified at the argument does not support. |

| [19]<br>XA_RDR_EXEC_FATAL_FORMAT_MI<br>SMATCH | 0xFFFF9084 | It is an error related to the mismatching data format between 2 output streams which are going to mix together. Please make sure these streams have the same format (PCM width, output sampling rate, output channel). |
|---|---|---|
| [20]<br>XA_RDR_CONFIG_NONFATAL_OPE<br>RATION | 0x00000888 | It is an error for renderer specification out of range. The setting state was specified at the argument does not support. |
| [21] | Others | Reserved |

Table 2-20 Error Codes for Capture

| Error code (32bit) | Value | Description |
|---|---|---|
| [1]<br>XA_NO_ERROR | 0x00000000 | The processing results are normal.<br>The process has terminated normally. |
| [2]<br>XA_API_FATAL_MEM_ALLOC | 0xFFFF8000 | Abnormality has occurred, which disables process continuation. An address of API structure was specified at the argument is NULL, the program execution is incorrect.<br>Because it becomes the common API error, please check the correct procedure. |
| [3]<br>XA_API_FATAL_MEM_ALIGN | 0xFFFF8001 | Abnormality has occurred, which disables process continuation. An address of API structure was specified at the argument does not 4 byte align. Because it becomes the common API error, please check the correct procedure. |
| [4]<br>XA_API_FATAL_INVALID_CMD | 0xFFFF8002 | Abnormality has occurred, which disables process continuation. The command was specified at the argument does not support. Because it becomes the common API error, please check the correct procedure. |
| [5]<br>XA_API_FATAL_INVALID_CMD_TYPE | 0xFFFF8003 | Abnormality has occurred, which disables process continuation. The subcommand was specified at the argument does not support. Because it becomes the common API error, please check the correct procedure. |
| [6]<br>XA_CAP_CONFIG_FATAL_STATE | 0xFFFF88C0 | Abnormality has occurred, which disables process continuation. The subcommand was specified at the argument does not support. Because it becomes the common API error, please check the correct procedure. |
| [7]<br>XA_CAP_CONFIG_FATAL_ERR_MONO_24BIT | 0xFFFF88C2 | It is an error for invalid setting, 24 bit mono channel case does not supported. |
| [8]<br>XA_CAP_EXEC_FATAL_STATE | 0xFFFF90C0 | Abnormality has occurred, which disables process continuation. The subcommand was specified at the argument does not support. Because it becomes the common API error, please check the correct procedure. |
| [9]<br>XA_CAP_EXEC_FATAL_INTERNAL | 0xFFFF90C2 | Abnormality has occurred, which disables process continuation. Some of setting becomes incorrect after combination (hardware modules are not available...). Because it becomes the common API error, please check the correct parameters. |

| [10]<br>XA_CAP_EXEC_FATAL_HW | 0xFFFF90C1 | Abnormality has occurred, which disables process continuation. Hardware modules are not available... Because it becomes the common API error, please make sure the resource is validity. |
|---|---|---|
| [11]<br>XA_CAP_CONFIG_NONFATAL_ERR_PCM_WIDTH | 0x000008C0 | It is an error for Capture specifications out of the range. The output buffer size was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [12]<br>XA_CAP_CONFIG_NONFATAL_ERR_CHANNELS | 0x000008C1 | It is an error for Capture specifications out of the range. The output buffer size was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [13]<br>XA_CAP_CONFIG_NONFATAL_ERR_SAMPLE_RATE | 0x000008C2 | It is an error for Capture specifications out of the range. The output buffer size was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [14]<br>XA_CAP_CONFIG_NONFATAL_ERR_FRAME_SIZE | 0x000008C3 | It is an error for Capture specifications out of the range. The output buffer size was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [15]<br>XA_CAP_CONFIG_NONFATAL_ERR_SOURCE | 0x000008C4 | It is an error for Capture specifications out of the range. The output buffer size was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [16]<br>XA_CAP_CONFIG_NONFATAL_ERR_DMACHANNEL | 0x000008C5 | It is an error for Capture specifications out of the range. The output buffer size was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [17]<br>XA_CAP_CONFIG_NONFATAL_VOLUME_RATE | 0x000008C6 | It is an error for Capture specification out of range. The volume rate value was specified at the argument does not support. |
| [18]<br>XA_CAP_CONFIG_NONFATAL_OPERATION | 0x000008C7 | It is an error for Capture specification out of range. The setting state was specified at the argument does not support. |
| [19] | Others | Reserved |

# 3.    Processing Flow

Figure 3-1 shows a flow diagram of processing performed by an application which uses this software. It applies for both case: renderer and capture.
The basic steps executed by the framework are shaded. The steps defined by the user framework are white. Design the process to suit the target system.
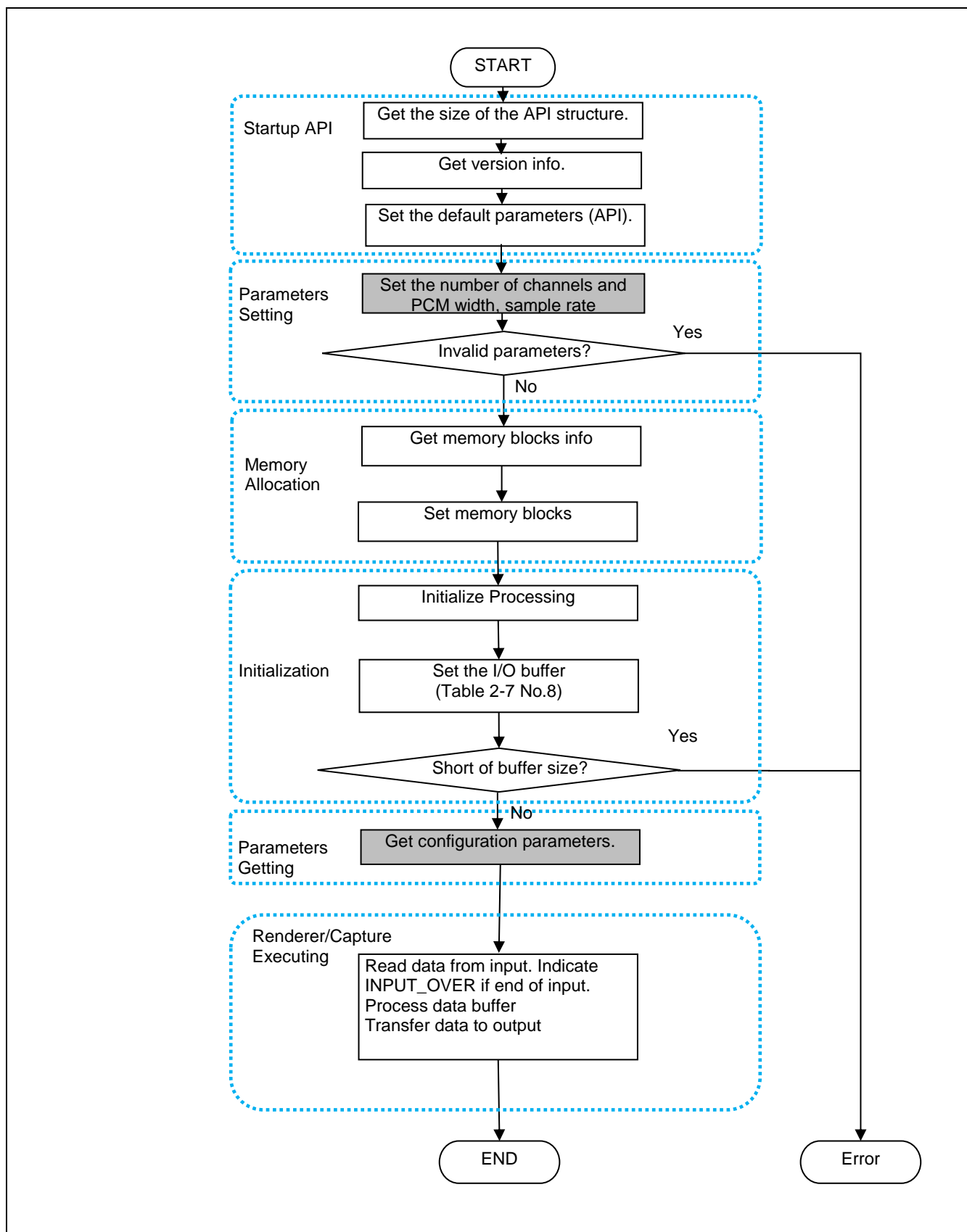
Figure 3-1        Example of the Application Processing Flow

# 4. Appendix

This section will explain more detail about the configuration to using CTU, MIX module to perform channel transfer and mixing functions.

## 4.1 CTU (Channel transfer unit)

CTU implements the channel transfer unit function. It converts PCM format from "x" channel (for input PCM of Renderer plugin) to "y" channel (expected output channel for speaker).
*(x = monaural, stereo, 4 channels, 6 channels, 8 channels; y = monaural, stereo)*

The setting of input, output channel can be referred in XA_API_CMD_SET_CONFIG_PARAM command.

To use CTU functions, the input channel's and output channel's values must be different, if not, it implies that CTU module is not used, except when Renderer is using MIX function. Because CTU and MIX are electrically connected as Figure 4-1.

There are two discrete CTU modules available. And each module includes four sub-modules. The output of these sub-modules internally connects a MIX module (see Figure 4-1).
So, if you play two PCMs (from 2 Renderers) and mix them together, two sub-modules of CTU0 will be enabled, while CTU1 module is still available. But if you use CTU function in playing two PCMs (from 2 Renderers) separately to two different outputs, then both CTU0 and CTU1 are enabled. Note that, the index of CTU will be selected inside Renderer plugins automatically.

Figure 4-1 shows all available of CTU modules.
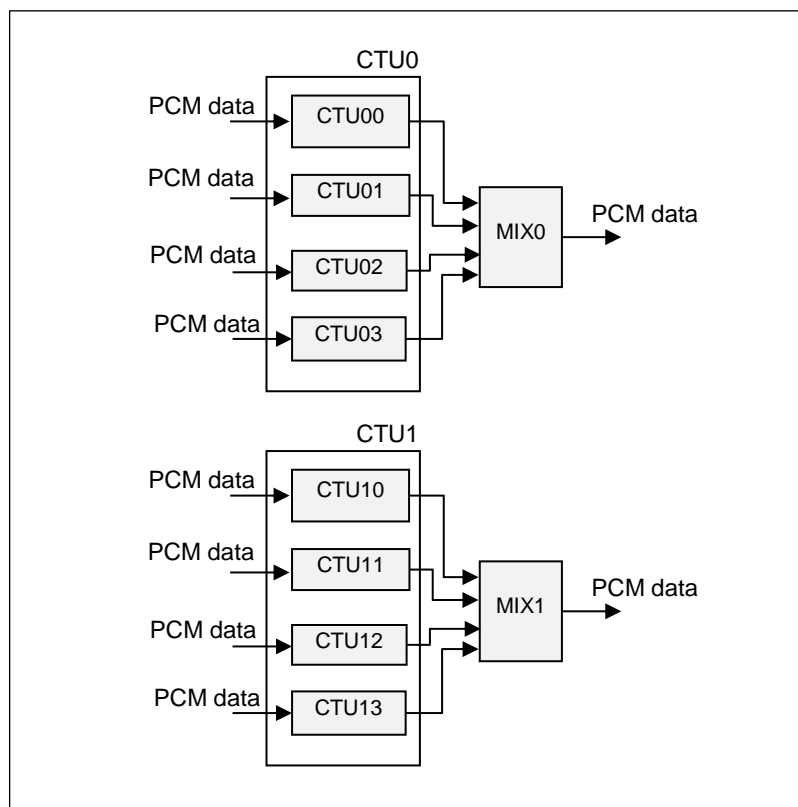


Figure 4-1      Block diagram of CTU

## 4.2    MIX (Mixing)

MIX implements the mixing (adding) two to four streams from Renderer plugins into a single stream, which will be output to a speaker.

The setting of plugin to use MIX functions can be referred to XA_API_CMD_SET_CONFIG_PARAM command.

As Figure 4-1, there are two MIX modules available. And each one supports maximum of 4 inputs.

The below conditions must be assured when mixing PCMs output from Renderers.

(1)    Set mix control flag
 -    Plugin has to set the mix control flag to assure that this stream want to mix with others.

(2)    Same output destination
 -    All plugins have to set the same output SSI module index. Otherwise, the plugin will play independently as PCM 3 in Figure 4-2.

(3)    Input mixing available
 -    A Mix module supports maximum of 4 streams. So when the 5th stream performs mixing (condition (2)), the busy error of hardware will be returned.

(4)    Consistent PCMs format
 -    The output PCM format (output channels, output sampling rate, and PCM width) has to have same configurations. Otherwise, this plugin will return an error and stop immediately.

Below are figures that present some use cases in using CTU/MIX modules in ADSP Renderer plugin.
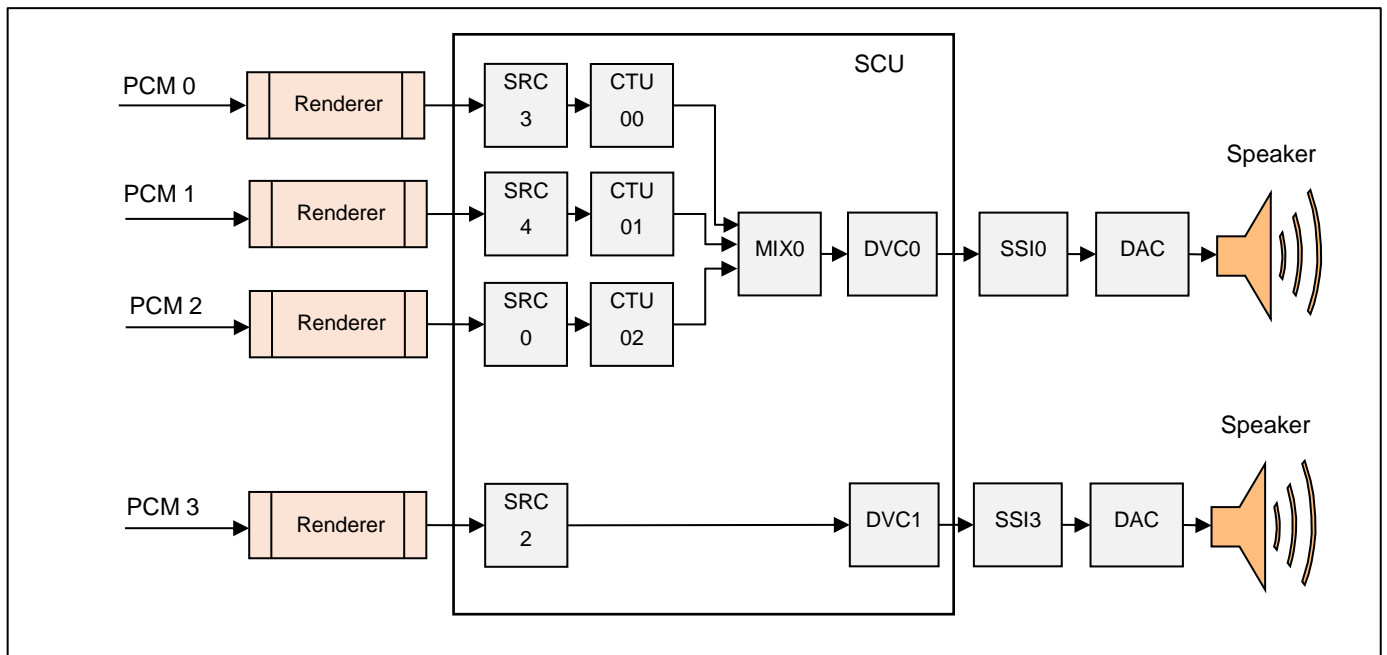


Figure 4-2        Example of using CTU/MIX for 3 streams to SSI0 and another stream to SSI3
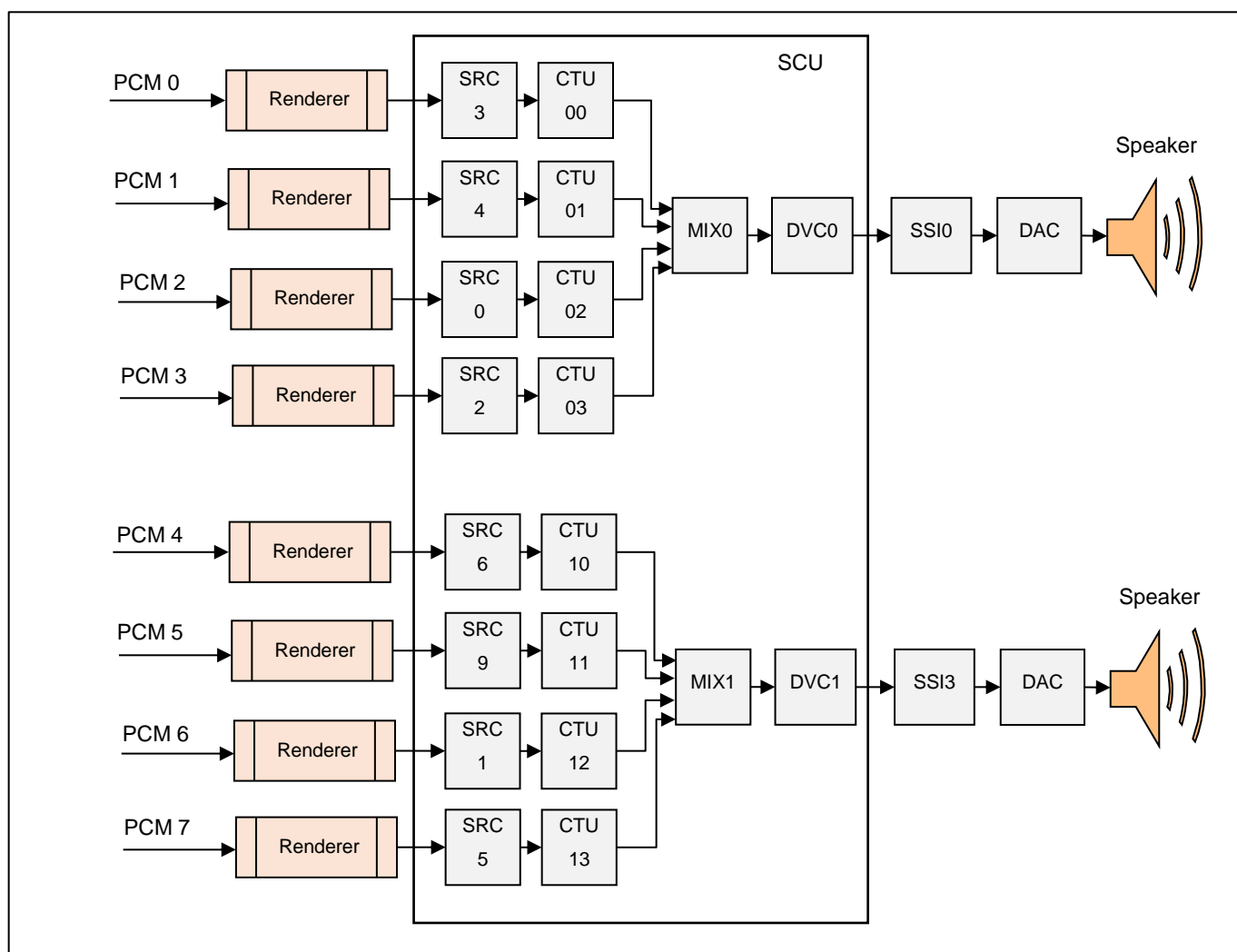
Figure 4-3        Example of using CTU/MIX to play 8 streams to SSI0 and SSI3.

About the volume control feature, MIX supports to control the decibel (gain level) of each input stream. From that, the output volume will be controlled by both DVC and MIX (Figure 4-3). On the other hand, by setting MIX, user can set different volume rates for four Renderer plugins when mixing. The output sound is a combination of the inputs and their expected volumes.

| Revision History | ADSP Reference Renderer/Capture Plugin User's Manual | | |
| --- | --- | --- | --- |

| Rev. | Date | Description | |
| --- | --- | --- | --- |
| | | Page | Summary |
| 0.10 | May. 30, 2018 | - | New Create |

ADSP Reference Renderer/Capture Plugin User's Manual

# RENESAS

# ADSP Reference Renderer/Capture Plugin
# RCG3AHPLN0202ZDO

**RENESAS**

Renesas Electronics Corporation