- Table of Contents -

[1. Overview 3](#_Toc527033950)

[1.1 Specifications Outline 3](#_Toc527033951)

[1.2 Configuration 6](#_Toc527033952)

[2. Software Specifications 9](#_Toc527033953)

[2.1 API specifications 9](#_Toc527033954)

[2.2 Command 10](#_Toc527033955)

[2.2.1 Command list 11](#_Toc527033956)

[2.2.2 Detail of Command Specifications 21](#_Toc527033957)

[2.3 Structures 82](#_Toc527033958)

[2.3.1 XARelTDMrdr type structure 83](#_Toc527033959)

[2.3.2 XARelTDMcap type structure 84](#_Toc527033960)

[2.4 Memory Specifications 85](#_Toc527033961)

[2.4.1 Persistent Area 85](#_Toc527033962)

[2.4.2 Stack Area 85](#_Toc527033963)

[2.4.3 Heap Area 85](#_Toc527033964)

[2.4.4 Input Buffer 86](#_Toc527033965)

[2.4.5 Output Buffer 86](#_Toc527033966)

[2.5 Error Processing 89](#_Toc527033967)

[2.5.1 Error codes 90](#_Toc527033968)

[3. Processing Flow 94](#_Toc527033969)

[4. Appendix 96](#_Toc527033970)

- List of Figures -

[Figure 1‑1 Example of the ADSP System Configuration for TDM renderer function 6](#_Toc517343829)

[Figure 1‑2 Example of the ADSP System Configuration for capture function 7](#_Toc517343830)

[Figure 2‑1 API command sequence overview 10](file:///D:\02_RSS2_Middleware1\03_Working\02_repositories\s492\documents\40_FD\60_TDM\en\04_Body.docx#_Toc517343831)

[Figure 2‑2 PCM 16-bit Data Access (Little Endian Mode) 87](#_Toc517343832)

[Figure 2‑3 PCM 24-bit Data Access (Little Endian Mode) 87](#_Toc517343833)

[Figure 2‑4 Output Formats 88](#_Toc517343834)

[Figure 3‑1 Example of the Application Processing Flow 95](#_Toc517343835)

- List of Tables -

[Table 1‑1 Basic Specification 3](#_Toc517343836)

[Table 1‑2 Supported TDM Renderer function Specifications 3](#_Toc517343837)

[Table 1‑3 Support TDM Capture function Specification 4](#_Toc517343838)

[Table 1‑4 Memory Size Requirements 5](#_Toc517343839)

[Table 1‑5 Version Information 5](#_Toc517343840)

[Table 2‑1 API Functions of TDM Renderer 9](#_Toc517343841)

[Table 2‑2 API Functions of TDM Capture 9](#_Toc517343842)

[Table 2‑3 List of supported none supported command, subcommand 11](#_Toc517343843)

[Table 2‑4 List of Initialization Commands 13](#_Toc517343844)

[Table 2‑5 List of Set Commands for renderer 14](#_Toc517343845)

[Table 2‑6 List of Set Commands for capture 15](#_Toc517343846)

[Table 2‑7 List of Memory allocation Commands 16](#_Toc517343847)

[Table 2‑8 List of initialize commands 17](#_Toc517343848)

[Table 2‑9 List of Get commands for renderer 18](#_Toc517343849)

[Table 2‑10 List of Get commands for capture 19](#_Toc517343850)

[Table 2‑11 List of execution commands 20](#_Toc517343851)

[Table 2‑12 Structures 82](#_Toc517343852)

[Table 2‑13 XARelTDMrdr type structure information 83](#_Toc517343853)

[Table 2‑14 XARelTDMcap type structure information 84](#_Toc517343854)

[Table 2‑15 Persistent Area Description 85](#_Toc517343855)

[Table 2‑16 Input Buffer Description 86](#_Toc517343856)

[Table 2‑17 Output Buffer Description 86](#_Toc517343857)

[Table 2‑18 Error Codes for TDM Renderer 90](#_Toc517343858)

[Table 2‑19 Error Codes for TDM Capture 92](#_Toc517343859)

[Table 4‑1 Matrix table for sampling rate setting of TDM Renderer 96](#_Toc517343860)

[Table 4‑2 Matrix table for sampling rate setting of TDM Capture 96](#_Toc517343861)

# Overview

This section provides an overview of the Time-Division Multiplexing (TDM) Renderer plugin. It contains TDM renderer and capture function.

## Specifications Outline

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

TDM Capture function capture/record the multiplexing 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 TDM Renderer function Specifications

| Item | Description |
| --- | --- |
| Input data format | |  |  |  |  | | --- | --- | --- | --- | | Channel number | | PCM bit-width (fix-point) | | | 16-bit | 24-bit | | 6ch | 3 \* 2ch | ○ | ○ | | 1 \* 6ch | ○ | ○ | | 8ch | 4 \* 2ch | ○ | ○ | | 1 \* 8ch | ○ | ○ | |
| Output data format | Time-division Multiplexing 16-bit/24-bit linear PCM (fixed point) |
| Input Sampling frequency (Hz) supported | 48000 / 44100 / 32000 |
| Output Sampling frequency (Hz) supported | 48000 / 44100 |
| Number of channels supported | TDM format channel (6 / 8) |
| Reentrant | Supported |
| Other | - |
| Restrictions | - |

Table 1‑3 Support TDM Capture function Specification

| Item | Description |
| --- | --- |
| Output data format | |  |  |  |  | | --- | --- | --- | --- | | Channel number | | PCM bit-width (fix-point) | | | 16-bit | 24-bit | | 6ch | 3 \* 2ch | ○ | ○ | | 1 \* 6ch | ○ | ○ | | 8ch | 4 \* 2ch | ○ | ○ | | 1 \* 8ch | ○ | ○ | |
| Input data format | Time-division Multiplexing 16-bit/24-bit linear PCM (fixed point) |
| Output Sampling frequency (Hz) supported | 48000 / 44100 / 32000 |
| Input Sampling frequency (Hz) supported | 48000 / 44100 |
| Number of channels supported | TDM format channel (6 / 8) |
| Reentrant | Supported |
| Other | - |
| Restrictions | - |

Table 1‑4 Memory Size Requirements

| Memory type | Location | Memory area name | | Size (in bytes) | |
| --- | --- | --- | --- | --- | --- |
| Instruction | ROM | Instruction area | | 53747 | |
| Data | Constant table area | |
| Other area(Depended on the compiler) | |
| RAM  (TDM Capture) | Software work area | | 198284 | |
| Area breakdown | Persistent area | Size breakdown | 67208 |
| Scratch area | 65536 |
| DTCM area | 65536 |
| Built-in descriptor area | 4 |
| User work area | | 34208 | |
| Area breakdown | Output buffer | Size breakdown | 32768 |
| Structure | 1440 |
| Stack area | | 944 | |
| Other area(Depended on the compiler) | | 0 | |
| RAM  (TDM Renderer) | Software work area | | 165516 | |
| Area breakdown | Persistent area | Size breakdown | 67208 |
| Scratch area | 32768 |
| DTCM area | 65536 |
| Built-in descriptor area | 4 |
| User work area | | 34224 | |
| Area breakdown | Input buffer | Size breakdown | 32768 |
| Structure | 1456 |
| Stack area | | 896 | |
| 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 1.0.0 |
| API Version information | Version 1.0.0 |

## Configuration

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

ADSP TDM Plugin

ADSP Framework

HiFi2 TDM Renderer

TDM PCM data

Speaker

4xPCM data

DAC

SRC

DVC

SCU

SSI

Figure 1‑1 Example of the ADSP System Configuration for TDM renderer function

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

ADSP TDM Plugin

ADSP Framework

HiFi2 TDM Capture

TDM PCM data

Microphone

4xPCM data

ADC

DVC

SRC

SCU

SSI

Figure 1‑2 Example of the ADSP System Configuration for capture function

1. ADSP Framework

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

1. HiFi2 TDM Renderer (ADSP TDM Plugin)

It performs merge multiple input PCM data and output to other audio device. It is this software set up as ADSP TDM Plugin.

1. HiFi2 TDM Capture (ADSP TDM Plugin)

It performs split multiple output PCM data from TDM input received from other audio device. It is this software set up as ADSP TDM Plugin.

1. PCM data

16-bit / 24-bit linear PCM data which is a processing by this software.

1. SCU

It performs sampling rate converters (SRC) and volume control (DVC).

1. SSI (\*)

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

1. DAC/ADC

The DAC/ADC converts a digital 16-bit/24-bit linear PCM data into analog signal and vice versa.

# Software Specifications

## API specifications

A single interface function is used to access the plugin, with operation specified by command codes. Each library has a single C API call. The parameter definition for every library are same and is specified as below:

In TDM renderer case

Table 2‑1 API Functions of TDM Renderer

|  |  |
| --- | --- |
| xa\_rel\_tdm\_rdr | |
| Description | This API is the only access function to the TDM renderer. |
| Syntax | XA\_ERRORCODE xa\_rel\_tdm\_rdr(  xa\_codec\_handle\_t p\_xa\_module\_obj,  WORD32 i\_cmd,  WORD32 i\_idx,  pVOID pv\_value); |
| Parameters | p\_xa\_module\_obj : Pointer to opaque API structure.  i\_cmd : Command. (defined in the supplied header files as)  i\_idx : Command subtype or index. (defined in the supplied header files as)  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) |

In TDM capture case

Table 2‑2 API Functions of TDM Capture

|  |  |
| --- | --- |
| xa\_rel\_tdm\_cap | |
| Description | This API is the only access function to the capture. |
| Syntax | XA\_ERRORCODE xa\_rel\_tdm\_cap(  xa\_codec\_handle\_t p\_xa\_module\_obj,  WORD32 i\_cmd,  WORD32 i\_idx,  pVOID pv\_value); |
| Parameters | p\_xa\_module\_obj : Pointer to opaque API structure.  i\_cmd : Command. (defined in the supplied header files as)  i\_idx : Command subtype or index. (defined in the supplied header files as)  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) |

## 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 API command sequence overview

Start-up API

Parameters Setting

Memory Allocation

Initialize plugin

Done?

Parameters Getting

Execution

Done?

### 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 | R | C |
| XA\_API\_CMD\_GET\_LIB\_ID\_STRINGS | XA\_CMD\_TYPE\_LIB\_VERSION | ○ | ○ |
| XA\_CMD\_TYPE\_API\_VERSION | ○ | ○ |
| XA\_API\_CMD\_GET\_API\_SIZE | - | ○ | ○ |
| XA\_API\_CMD\_INIT | XA\_CMD\_TYPE\_INIT\_API\_PRE\_CONFIG\_PARAMS | ○ | ○ |
| XA\_CMD\_TYPE\_INIT\_API\_POST\_CONFIG\_PARAMS | ○ | ○ |
| XA\_CMD\_TYPE\_INIT\_PROCESS | ○ | ○ |
| XA\_CMD\_TYPE\_INIT\_DONE\_QUERY | ○ | ○ |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | XA\_TDM\_RDR\_CONFIG\_PARAM\_PCM\_WIDTH | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_CHANNEL\_MODE | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_FRAME\_SIZE | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT1 | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL1 | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT2 | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL2 | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT3\* | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL3\* | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUT\_SAMPLE\_RATE | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_VOLUME\_RATE | ○ | - |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_PCM\_WIDTH | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_CHANNEL\_MODE | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_FRAME\_SIZE | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT1 | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_DMACHANNEL1 | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT2 | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_DMACHANNEL2 | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT3\* | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_DMACHANNEL3\* | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_OUT\_SAMPLE\_RATE | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_VOLUME\_RATE | - | ○ |
| XA\_API\_CMD\_GET\_CONFIG\_PARAM | XA\_TDM\_RDR\_CONFIG\_PARAM\_PCM\_WIDTH | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_CHANNEL\_MODE | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_FRAME\_SIZE | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT1 | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL1 | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT2 | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL2 | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT3\* | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL3\* | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUT\_SAMPLE\_RATE | ○ | - |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_VOLUME\_RATE | ○ | - |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_PCM\_WIDTH | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_CHANNEL\_MODE | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_FRAME\_SIZE | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT1 | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_DMACHANNEL1 | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT2 | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_DMACHANNEL2 | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT3\* | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_DMACHANNEL3\* | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_OUT\_SAMPLE\_RATE | - | ○ |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_VOLUME\_RATE | - | ○ |
| XA\_API\_CMD\_GET\_MEMTABS\_SIZE | - | ○ | ○ |
| XA\_API\_CMD\_SET\_MEMTABS\_PTR | - | ○ | ○ |
| XA\_API\_CMD\_GET\_N\_MEMTABS | - | ○ | ○ |
| XA\_API\_CMD\_GET\_MEM\_INFO\_SIZE | - | ○ | ○ |
| XA\_API\_CMD\_GET\_MEM\_INFO\_ALIGNMENT | - | ○ | ○ |
| XA\_API\_CMD\_GET\_MEM\_INFO\_TYPE | - | ○ | ○ |
| XA\_API\_CMD\_SET\_MEM\_PTR | - | ○ | ○ |
| XA\_API\_CMD\_SET\_INPUT\_BYTES | - | ○ | ○ |
| XA\_API\_CMD\_INPUT\_OVER | - | ○ | ○ |
| XA\_API\_CMD\_GET\_CURIDX\_INPUT\_BUF | - | ○ | - |
| XA\_API\_CMD\_EXECUTE | XA\_CMD\_TYPE\_DO\_EXECUTE | ○ | ○ |
| XA\_CMD\_TYPE\_DONE\_QUERY | ○ | ○ |
| XA\_API\_CMD\_GET\_OUTPUT\_BYTES | - | - | ○ |

R: TDM Renderer C: TDM Capture

○ : Available

- : Omitted

\* : Not applicable in current library version

#### 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 |

#### Parameters setting

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 TDM PCM sample bit width to 16 or 24 |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_PCM\_WIDTH |
| 2 | XA\_API\_CMD\_SET\_CONFIG\_PARAM | Set the input TDM PCM channel mode |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_CHANNEL\_MODE |
| 3 | XA\_API\_CMD\_SET\_CONFIG\_PARAM | Set the input TDM PCM sampling frequency (supported 32000/44100/48000 Hz) |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE |
| 4 | XA\_API\_CMD\_SET\_CONFIG\_PARAM | Set the input/output frame size |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_FRAME\_SIZE |
| 5 | XA\_API\_CMD\_SET\_CONFIG\_PARAM | Set the output destination Audio device 1st for TDM Renderer |
| XA\_TDM\_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\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL1 |
| 7 | XA\_API\_CMD\_SET\_CONFIG\_PARAM | Set the output destination Audio device 2nd for TDM Renderer |
| XA\_TDM\_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\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL2 |
| 9 | XA\_API\_CMD\_SET\_CONFIG\_PARAM | Set the output PCM sampling frequency  (supported 48000/44100 Hz) |
| XA\_TDM\_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\_TDM\_RDR\_CONFIG\_PARAM\_VOLUME\_RATE |

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 TDM PCM sample bit width to 16 or 24 |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_PCM\_WIDTH |
| 2 | XA\_API\_CMD\_SET\_CONFIG\_PARAM | Set the input TDM PCM channel mode |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_CHANNEL\_MODE |
| 3 | XA\_API\_CMD\_SET\_CONFIG\_PARAM | Set the input TDM PCM sampling frequency  (supported 48000/44100 Hz) |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE |
| 4 | XA\_API\_CMD\_SET\_CONFIG\_PARAM | Set the input/output frame size |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_FRAME\_SIZE |
| 5 | XA\_API\_CMD\_SET\_CONFIG\_PARAM | Set the input source Audio device 1st for TDM Capture |
| XA\_TDM\_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\_TDM\_CAP\_CONFIG\_PARAM\_DMACHANNEL1 |
| 7 | XA\_API\_CMD\_SET\_CONFIG\_PARAM | Set the input source Audio device 2nd for TDM Capture |
|  | XA\_TDM\_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\_TDM\_CAP\_CONFIG\_PARAM\_DMACHANNEL2 |
| 9 | XA\_API\_CMD\_SET\_CONFIG\_PARAM | Set the output PCM sampling frequency (supported 32000/44100/48000 Hz) |
|  | XA\_TDM\_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\_TDM\_CAP\_CONFIG\_PARAM\_VOLUME\_RATE |

#### 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) |

#### 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) |

#### Parameters getting

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 input TDM PCM sample bit width |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_PCM\_WIDTH |
| 2 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get the input TDM PCM channel mode |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_CHANNEL\_MODE |
| 3 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get the input TDM PCM sampling frequency |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE |
| 4 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get the input/output frame size |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_FRAME\_SIZE |
| 5 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get TDM Renderer output destination Audio device 1st |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT1 |
| 6 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get ADMA channel number usage for Audio device 1st |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL1 |
| 7 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get TDM Renderer output destination Audio device 2nd |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT2 |
| 8 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get ADMA channel number usage for Audio device 2nd |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL2 |
| 9 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get the output PCM sampling frequency |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUT\_SAMPLE\_RATE |
| 10 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get the output PCM volume rate compare with input PCM |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_VOLUME\_RATE |

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 input TDM PCM sample bit width |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_PCM\_WIDTH |
| 2 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get the input TDM PCM channel mode |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_CHANNEL\_MODE |
| 3 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get the input TDM PCM sampling frequency |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE |
| 4 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get the input/output frame size |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_FRAME\_SIZE |
| 5 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get TDM Capture input source Audio device 1st |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT1 |
| 6 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get ADMA channel number usage for Audio device 1st |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_DMACHANNEL1 |
| 7 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get TDM Capture input destination Audio device 2nd |
|  | XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT2 |
| 8 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get ADMA channel number usage for Audio device 2nd |
|  | XA\_TDM\_CAP\_CONFIG\_PARAM\_DMACHANNEL2 |
| 9 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get the output PCM sampling frequency |
|  | XA\_TDM\_CAP\_CONFIG\_PARAM\_OUT\_SAMPLE\_RATE |
| 10 | XA\_API\_CMD\_GET\_CONFIG\_PARAM | Get the output PCM volume rate compare with input PCM |
|  | XA\_TDM\_CAP\_CONFIG\_PARAM\_VOLUME\_RATE |

#### Execution

Table 2‑11 List of execution commands

| upper stage : Command / lower step : Subcommand | | Description |
| --- | --- | --- |
| 1 | XA\_API\_CMD\_INPUT\_OVER | Signal TDM Renderer/Capture the input data is over |
| (NULL) |
| 2 | XA\_API\_CMD\_SET\_INPUT\_BYTES | Set the number of bytes available in the input buffer (only available in TDM Renderer) |
| (NULL) |
| 3 | XA\_API\_CMD\_EXECUTE | Execute TDM Renderer/Capture plugin |
| 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 (only available in TDM Capture) |
| (NULL) |
| 6 | XA\_API\_CMD\_GET\_CURIDX\_INPUT\_BUF | Get the number of input buffer bytes consumed  (only available in TDM Renderer) |
| (NULL) |

### Detail of Command Specifications

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

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

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

#### XA\_API\_CMD\_GET\_LIB\_ID\_STRINGS command

FD\_PLG\_TDM\_001

|  |  |  |
| --- | --- | --- |
| 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 | - | |

[Covers: RD\_014]

Example:

char api\_version[30];

res = (\*api\_func)(NULL,

XA\_API\_CMD\_GET\_LIB\_ID\_STRINGS,

XA\_CMD\_TYPE\_API\_VERSION,

(pVOID) api\_version);

#### XA\_API\_CMD\_GET\_API\_SIZE command

FD\_PLG\_TDM\_002

|  |  |  |
| --- | --- | --- |
| 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. | |

[Covers: RD\_014]

Example:

WORD32 api\_size;

res = (\*api\_func)(api\_obj,

XA\_CMD\_TYPE\_API\_SIZE,

0,

&api\_size);

#### 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 | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_INIT | |
| i\_idx | |
| XA\_CMD\_TYPE\_INIT\_API\_PRE\_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. |
| Restrictions | - | |

FD\_PLG\_TDM\_003

[Covers: RD\_014]

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.  If there are any parameters cannot be applied. Plugin returns a fatal error, or performs the change of these parameters automatically based on defined cases (i.e. enable SRC module if input sample rate sets to 32000 Hz, ...) | |
| 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE (in TDM Capture)  Or  XA\_TDM\_RDR\_CONFIG\_FATAL\_STATE (in TDM Renderer) | Incorrect sequence call (i.e. call before pre-configuration step or call before set memory table step) |
| XA\_TDM\_CAP\_EXEC\_FATAL\_INTERNAL (in TDM Capture)  XA\_TDM\_RDR\_EXEC\_FATAL\_INTERNAL  (in TDM Renderer) | Invalid connection device setting path (i.e. setting SRC module for both device1 and device2), or lack of memory resource. |
| Restrictions | - | |

FD\_PLG\_TDM\_004

[Covers: RD\_014]

Example:

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_INIT,

XA\_CMD\_TYPE\_INIT\_API\_POST\_CONFIG\_PARAMS,

NULL);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_CMD\_TYPE\_INIT\_PROCESS | |
| Description | Setup and start 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, plugin will return error code. | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_INIT | |
| i\_idx | |
| XA\_CMD\_TYPE\_INIT\_PROCESS | |
| 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\_TDM\_CAP\_EXEC\_FATAL\_STATE (in TDM Capture)  (XA\_TDM\_RDR\_EXEC\_FATAL\_STATE (in TDM Renderer) | Incorrect sequence call (i.e. call before post-configuration step or without persistent/scratch buffer allocation, or without DTCM/Built-in descriptor memory allocation (in case of DMAC used)). |
| XA\_TDM\_CAP\_EXEC\_FATAL\_INTERNAL (in TDM Capture)  XA\_TDM\_RDR\_EXEC\_FATAL\_INTERNAL  (in TDM Renderer) | Plugin has some abnormal cases happened from hardware modules (i.e. all hardware resource is busy). |
| Restrictions | - | |

FD\_PLG\_TDM\_005

[Covers: RD\_014]

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 one; 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\_TDM\_CAP\_EXEC\_FATAL\_STATE (in TDM Capture)  (XA\_TDM\_RDR\_EXEC\_FATAL\_STATE (in TDM Renderer) | Incorrect sequence call (i.e. call before post-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_006

[Covers: RD\_014]

Example:

WORD32 done;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_INIT,

XA\_CMD\_TYPE\_INIT\_DONE\_QUERY,

&done);

#### XA\_API\_CMD\_GET\_MEMTABS\_SIZE command

FD\_PLG\_TDM\_007

|  |  |  |
| --- | --- | --- |
| 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE (in TDM Capture)  Or  XA\_TDM\_RDR\_CONFIG\_FATAL\_STATE (in TDM Renderer) | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

[Covers: RD\_014]

Example:

WORD32 memtab\_size;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_MEMTABS\_SIZE,

0,

&memtab\_size);

#### XA\_API\_CMD\_SET\_MEMTABS\_PTR command

FD\_PLG\_TDM\_008

|  |  |  |
| --- | --- | --- |
| 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE (in TDM Capture)  Or  XA\_TDM\_RDR\_CONFIG\_FATAL\_STATE (in TDM Renderer) | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

[Covers: RD\_014]

Example:

pVOID memtab\_ptr;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_MEMTABS\_PTR,

0,

memtab\_ptr);

#### XA\_API\_CMD\_GET\_N\_MEMTABS command

FD\_PLG\_TDM\_009

|  |  |  |
| --- | --- | --- |
| 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. |
| XA\_TDM\_CAP\_CONFIG\_FATAL\_STATE (in TDM Capture)  Or  XA\_TDM\_RDR\_CONFIG\_FATAL\_STATE (in TDM Renderer) | Incorrect sequence call (i.e. call before post-configuration step) |
| Restrictions | pv\_value will be changed depend on channel mode and DMAC transfer type (using ADMAC or DMACPP) | |

[Covers: RD\_014]

Example:

WORD32 n\_memtab;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_N\_MEMTABS,

0,

&n\_memtab);

#### XA\_API\_CMD\_GET\_MEM\_INFO\_SIZE command

FD\_PLG\_TDM\_010

|  |  |  |
| --- | --- | --- |
| 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  0 - 1st Input Buffer (TDM Renderer) / 1st Output Buffer (TDM Capture)  1 - 2nd Input Buffer (TDM Renderer) / 2nd Output Buffer (TDM Capture)  2 - 3rd Input Buffer (TDM Renderer) / 3rd Output Buffer (TDM Capture)  3 - 4th Input Buffer (TDM Renderer) / 4th Output Buffer (TDM Capture)  4 - Persistent Area  5 - Scratch Area  6 - DTMC Area  7 - Built-in Area | |
| 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE (in TDM Capture)  Or  XA\_TDM\_RDR\_CONFIG\_FATAL\_STATE (in TDM Renderer) | Incorrect sequence call  (i.e. call before post-configuration step) |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE | Incorrect index |
| Restrictions | The index of DTCM and built-in area are only used in case of using ADMAC module to transfer data. And the index of input buffer will be affected by channel mode. So it may also affect to the other index memory. | |

[Covers: RD\_014]

Example:

WORD32 mem\_size;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_MEM\_INFO\_SIZE,

index,

&mem\_size);

#### XA\_API\_CMD\_GET\_MEM\_INFO\_ALIGNMENT command

FD\_PLG\_TDM\_011

|  |  |  |
| --- | --- | --- |
| 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  0 - 1st Input Buffer (TDM Renderer) / 1st Output Buffer (TDM Capture)  1 - 2nd Input Buffer (TDM Renderer) / 2nd Output Buffer (TDM Capture)  2 - 3rd Input Buffer (TDM Renderer) / 3rd Output Buffer (TDM Capture)  3 - 4th Input Buffer (TDM Renderer) / 4th Output Buffer (TDM Capture)  4 - Persistent Area  5 - Scratch Area  6 - DTMC Area  7 - Built-in Area | |
| 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE (in TDM Capture)  Or  XA\_TDM\_RDR\_CONFIG\_FATAL\_STATE (in TDM Renderer) | Incorrect sequence call  (i.e. call before post-configuration step) |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE | Incorrect index |
| Restrictions | The index of DTCM and built-in area are only used in case of using ADMAC module to transfer data. And the index of input buffer will be affected by channel mode. So it may also affect to the other index memory. | |

[Covers: RD\_014]

Example:

WORD32 mem\_align;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_MEM\_INFO\_ALIGNMENT,

index,

&mem\_align);

#### XA\_API\_CMD\_GET\_MEM\_INFO\_TYPE command

FD\_PLG\_TDM\_012

|  |  |  |
| --- | --- | --- |
| 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  0 - 1st Input Buffer (TDM Renderer) / 1st Output Buffer (TDM Capture)  1 - 2nd Input Buffer (TDM Renderer) / 2nd Output Buffer (TDM Capture)  2 - 3rd Input Buffer (TDM Renderer) / 3rd Output Buffer (TDM Capture)  3 - 4th Input Buffer (TDM Renderer) / 4th Output Buffer (TDM Capture)  4 - Persistent Area  5 - Scratch Area  6 - DTMC Area  7 - Built-in Area | |
| 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE (in TDM Capture)  Or  XA\_TDM\_RDR\_CONFIG\_FATAL\_STATE (in TDM Renderer) | Incorrect sequence call (i.e. call before post-configuration step) |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE | Incorrect index |
| Restrictions | The index of DTCM and built-in area are only used in case of using ADMAC module to transfer data. And the index of input buffer will be affected by channel mode. So it may also affect to the other index memory. | |

[Covers: RD\_014]

Example:

WORD32 mem\_type;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_MEM\_INFO\_TYPE,

index,

&mem\_type);

#### XA\_API\_CMD\_SET\_MEM\_PTR command

FD\_PLG\_TDM\_013

|  |  |  |
| --- | --- | --- |
| 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  0 - 1st Input Buffer (TDM Renderer) / 1st Output Buffer (TDM Capture)  1 - 2nd Input Buffer (TDM Renderer) / 2nd Output Buffer (TDM Capture)  2 - 3rd Input Buffer (TDM Renderer) / 3rd Output Buffer (TDM Capture)  3 - 4th Input Buffer (TDM Renderer) / 4th Output Buffer (TDM Capture)  4 - Persistent Area  5 - Scratch Area  6 - DTMC Area  7 - Built-in Area | |
| 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE (in TDM Capture)  Or  XA\_TDM\_RDR\_CONFIG\_FATAL\_STATE (in TDM Renderer) | Incorrect sequence call  (i.e. call before post-configuration step) |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE | Incorrect index |
| Restrictions | The index of DTCM and built-in area are only used in case of using ADMAC module to transfer data. And the index of input buffer will be affected by channel mode. So it may also affect to the other index memory. | |

[Covers: RD\_014]

Example:

pVOID addr;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_MEM\_PTR,

index,

addr);

#### XA\_API\_CMD\_INPUT\_OVER command

FD\_PLG\_TDM\_014

|  |  |  |
| --- | --- | --- |
| 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 | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_INPUT\_OVER | |
| i\_idx | |
| NULL | |
| pv\_value | |
| NULL | |
| 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\_TDM\_CAP\_EXEC\_FATAL\_STATE  (in TDM Capture)  Or  XA\_TDM\_RDR\_EXEC\_FATAL\_STATE  (in TDM Renderer) | Incorrect sequence call  (i.e. call before initialization step – init process) |
| Restrictions | - | |

[Covers: RD\_014]

Example:

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_INPUT\_OVER,

0,

NULL);

#### XA\_API\_CMD\_SET\_INPUT\_BYTES command

FD\_PLG\_TDM\_015

|  |  |  |
| --- | --- | --- |
| Subcommand | None | |
| Description | In TDM Capture this command will do nothing. The purpose of this command is filled the full list of standard API.  In TDM 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 | |
| The index of input buffer (only for TDM Renderer) | |
| pv\_value | |
| Pointer to the input byte variable (Any value is OK with TDM Capture) | |
| 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\_TDM\_RDR\_EXEC\_FATAL\_STATE  (only for TDM Renderer) | Input buffer is not ready, and have not init done |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE  (only for TDM Renderer) | Incorrect index of input buffer |
| XA\_TDM\_RDR\_EXEC\_FATAL\_INPUT  (only for TDM Renderer) | Invalid input buffer size (i.e. minus buffer size or buffer size is not align with sample size) |
| Restrictions | - | |

[Covers: RD\_014]

Example:

WORD32 filled;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_INPUT\_BYTES,

index,

&filled);

#### XA\_API\_CMD\_GET\_CURIDX\_INPUT\_BUF command

FD\_PLG\_TDM\_016

|  |  |  |
| --- | --- | --- |
| Subcommand | None | |
| Description | In TDM Capture, this command will return value 0 each time it’s called  In TDM 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 | |
| The index of input buffer (only for TDM Renderer) | |
| 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\_TDM\_RDR\_EXEC\_FATAL\_STATE  (only for TDM Renderer) | Input buffer is not ready |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE  (only for TDM Renderer) | Invalid index of input buffer |
| Restrictions | - | |

[Covers: RD\_014]

Example:

WORD32 consumed;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CURIDX\_INPUT\_BUF,

index,

&consumed);

#### XA\_API\_CMD\_EXECUTE command

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_CMD\_TYPE\_DO\_EXECUTE | |
| Description | This command execute the TDM Renderer/Capture plugin. | |
| 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\_TDM\_CAP\_EXEC\_FATAL\_STATE  (in TDM Capture)  Or  XA\_TDM\_RDR\_EXEC\_FATAL\_STATE  (in TDM Renderer) | Incorrect sequence call  (i.e. call before initialization step)  Or input / output buffer is not ready |
| XA\_TDM\_CAP\_EXEC\_FATAL\_INTERNAL (in TDM Capture)  Or  XA\_TDM\_RDR\_EXEC\_FATAL\_INTERNAL (in TDM Renderer) | Hardware does not stop successfully |
| Restrictions | - | |

FD\_PLG\_TDM\_017

[Covers: RD\_014]

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\_API\_FATAL\_MEM\_ALIGN | p\_xa\_module\_obj is not aligned to 4 bytes. |
|  | XA\_TDM\_CAP\_EXEC\_FATAL\_STATE (in TDM Capture)  Or  XA\_TDM\_RDR\_EXEC\_FATAL\_STATE (in TDM Renderer) | Incorrect sequence call (i.e. call before initialization step) |
| Restrictions | - | |

FD\_PLG\_TDM\_018

[Covers: RD\_014]

Example:

WORD32 done;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_EXECUTE,

XA\_CMD\_TYPE\_DONE\_QUERY,

&done);

#### XA\_API\_CMD\_GET\_OUTPUT\_BYTES command

FD\_PLG\_TDM\_019

|  |  |  |
| --- | --- | --- |
| Subcommand | None | |
| Description | In TDM Renderer, this command will do nothing. The purpose of this command is fulfilled the standard APIs list.  In TDM Capture, 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 | |
| The index of output buffer (only for TDM Capture) | |
| 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\_TDM\_CAP\_EXEC\_FATAL\_STATE (only for TDM Capture) | Incorrect sequence call (i.e. call before initialization step) Or output buffer is not ready |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE  (only for TDM Capture) | Invalid index of output buffer |
| Restrictions | - | |

[Covers: RD\_014]

Example

WORD32 produced;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_OUTPUT\_BYTES,

index,

&produced);

#### XA\_API\_CMD\_SET\_CONFIG\_PARAM command

2.2.2.16.1 Set configuration command for TDM Renderer

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_RDR\_CONFIG\_PARAM\_PCM\_WIDTH | |
| Description | Set the TDM 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\_TDM\_RDR\_CONFIG\_PARAM\_PCM\_WIDTH | |
| pv\_value | |
| Pointer to the sample bit width variable (valid value: 16 or 24) | |
| 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_RDR\_CONFIG\_FATAL\_PCM\_WIDTH | TDM PCM sample bit width is invalid |
| Restrictions | - | |

FD\_PLG\_TDM\_020

[Covers: RD\_014]

Example

WORD32 pcm\_width;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_PCM\_WIDTH,

&pcm\_width);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_RDR\_CONFIG\_PARAM\_CHANNEL\_MODE | |
| Description | Set TDM PCM channels mode | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_CHANNEL\_MODE | |
| pv\_value | |
| Pointer to the TDM channels mode variable  XA\_TDM\_RDR\_CHANNEL\_MODE\_2X4 : 4 stereo TDM data  XA\_TDM\_RDR\_CHANNEL\_MODE\_1X8 : 1 eight-channel TDM data  XA\_TDM\_RDR\_CHANNEL\_MODE\_2X3 : 3 stereo TDM data  XA\_TDM\_RDR\_CHANNEL\_MODE\_1X6 : 1 six-channel TDM data | |
| 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_RDR\_CONFIG\_FATAL\_CHANNEL\_MODE | Invalid TDM format |
| Restrictions | - | |

FD\_PLG\_TDM\_021

[Covers: RD\_014]

Example:

WORD32 ch\_mode;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_CHANNEL\_MODE,

&ch\_mode);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_RDR\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE | |
| Description | Set input TDM PCM sampling frequency | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE | |
| pv\_value | |
| Pointer to the input sampling frequency variable  (valid value: 32,000 / 44,100 / 48,000 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_RDR\_CONFIG\_FATAL\_SAMPLE\_RATE | Input TDM PCM sampling frequency is out of range. |
| Restrictions | - | |

FD\_PLG\_TDM\_022

[Covers: RD\_014]

Example

WORD32 sample\_rate;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE,

&sample\_rate);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_RDR\_CONFIG\_PARAM\_FRAME\_SIZE | |
| Description | Set input/output TDM PCM frame size in sample | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_FRAME\_SIZE | |
| pv\_value | |
| Pointer to frame size in sample variable  (valid value: 512 / 1024 / 2048) | |
| 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_RDR\_CONFIG\_FATAL\_FRAME\_SIZE | TDM PCM frame size value is out of range. |
| Restrictions | - | |

FD\_PLG\_TDM\_023

[Covers: RD\_014]

Example

WORD32 frame\_size;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_FRAME\_SIZE,

&frame\_size);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT1 | |
| Description | Set 1st output destination device for TDM Renderer. | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT1 | |
| pv\_value | |
| Pointer to output destination value 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_RDR\_CONFIG\_FATAL\_INVALID\_OUTPUT | TDM PCM output device is out of range. |
| Restrictions | List of supported module:   |  |  | | --- | --- | | Macro | Value | | SSI00 | 0 | | SSI10 | 10 | | SSI20 | 20 | | SSI30 | 30 | | SSI40 | 40 | | SSI90 | 90 | | SCU\_SRCI0 | 110 | | SCU\_SRCI1 | 111 | | SCU\_SRCI3 | 113 | | SCU\_SRCI4 | 114 | | |

FD\_PLG\_TDM\_024

[Covers: RD\_014]

Example:

WORD32 output\_dev;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT1,

&output\_dev);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_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\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL1 | |
| pv\_value | |
| Pointer to the Audio-DMAC / Audio-DMAC-peripheral-peripheral channels number  ADMAC\_CH[0-31] : Audio-DMAC usage  ADMACPP\_CH[0-28] : Audio-DMACpp usage | |
| 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_RDR\_CONFIG\_FATAL\_DMACHANNEL | TDM PCM ADMA channel setting is out of range. |
| Restrictions | - | |

FD\_PLG\_TDM\_025

[Covers: RD\_014]

Example:

WORD32 dma\_channel;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL1,

&dma\_channel);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT2 | |
| Description | Set 2nd output destination device for TDM Renderer. | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT2 | |
| pv\_value | |
| Pointer to output destination value 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_RDR\_CONFIG\_FATAL\_INVALID\_OUTPUT | TDM PCM output device is out of range. |
| Restrictions | List of supported module:   |  |  | | --- | --- | | Macro | Value | | SSI00 | 0 | | SSI10 | 10 | | SSI20 | 20 | | SSI30 | 30 | | SSI40 | 40 | | SSI90 | 90 | | SCU\_SRCI0 | 110 | | SCU\_SRCI1 | 111 | | SCU\_SRCI3 | 113 | | SCU\_SRCI4 | 114 | | |

FD\_PLG\_TDM\_026

[Covers: RD\_014]

Example:

WORD32 output\_dev;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT2,

&output\_dev);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_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\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL2 | |
| pv\_value | |
| Pointer to the Audio-DMAC / Audio-DMAC-peripheral-peripheral channels number  ADMAC\_CH[0-31] : Audio-DMAC usage  ADMACPP\_CH[0-28] : Audio-DMAC-pp usage | |
| 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_RDR\_CONFIG\_FATAL\_DMACHANNEL | TDM PCM ADMA channel setting is out of range. |
| Restrictions | - | |

FD\_PLG\_TDM\_027

[Covers: RD\_014]

Example:

WORD32 dma\_channel;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL2,

&dma\_channel);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_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 is valid and non-zero value, SRC connection will be enabled even without setting connection device path. And the connection will automatically use the available Audio-DMAC channel. If this setting is zero, SRC module will not be used. | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUT\_SAMPLE\_RATE | |
| pv\_value | |
| Pointer to the output sampling frequency variable.  Valid value:  0: disable SRC module  48,000/44,100 Hz: setting output sampling rate for SRC module | |
| 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_RDR\_CONFIG\_FATAL\_SAMPLE\_RATE | TDM PCM output sample rate is out of range. |
| Restrictions | - | |

FD\_PLG\_TDM\_028

[Covers: RD\_014]

Example:

WORD32 sample\_rate;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_OUT\_SAMPLE\_RATE,

&sample\_rate);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_RDR\_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 setting values except 0xFFFF FFFF (disable) will enabled DVC of SCU module and the connection will be established even without setting connection path. | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_VOLUME\_RATE | |
| pv\_value | |
| Pointer to the volume ratio number (using Fix-point Q3.20):  0xFFFF FFFF : disable DVC module  [0, 0x7F FFFF] : setting volume rate 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_RDR\_CONFIG\_FATAL\_VOLUME\_RATE | TDM PCM volume rate value is out of range. |
| Restrictions | - | |

FD\_PLG\_TDM\_029

[Covers: RD\_014]

Example:

WORD32 vol\_rate;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_VOLUME\_RATE,

&vol\_rate);

2.2.2.16.2 Set configuration command for TDM Capture

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_CONFIG\_PARAM\_PCM\_WIDTH | |
| Description | Set TDM 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\_TDM\_CAP\_CONFIG\_PARAM\_PCM\_WIDTH | |
| pv\_value | |
| Pointer to the sample bit width variable  (valid value: 16 or 24) | |
| 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_CAP\_CONFIG\_FATAL\_PCM\_WIDTH | TDM PCM sample width size is out of range. |
| Restrictions | - | |

FD\_PLG\_TDM\_030

[Covers: RD\_014]

Example

WORD32 pcm\_width;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_PCM\_WIDTH,

&pcm\_width);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_CONFIG\_PARAM\_CHANNEL\_MODE | |
| Description | Set TDM PCM channels mode | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_CHANNEL\_MODE | |
| pv\_value | |
| Pointer to the TDM channels mode variable  XA\_TDM\_CAP\_CHANNEL\_MODE\_2X4 : 4 stereo TDM data  XA\_TDM\_CAP\_CHANNEL\_MODE\_1X8 : 1 eight-channel TDM data  XA\_TDM\_CAP\_CHANNEL\_MODE\_2X3 : 3 stereo TDM data  XA\_TDM\_CAP\_CHANNEL\_MODE\_1X6 : 1 six-channel TDM data | |
| 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_CAP\_CONFIG\_FATAL\_CHANNEL\_MODE | Invalid TDM format |
| Restrictions | - | |

FD\_PLG\_TDM\_031

[Covers: RD\_014]

Example

WORD32 ch;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_CHANNEL\_MODE,

&ch);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE | |
| Description | Set input sample rate in Sampling Rate Converter (SRC) of Sampling Rate Converter Unit (SCU). If this setting is valid and non-zero value, SRC connection will be enabled even without setting connection device path. And the connection will automatically use the available Audio-DMAC channel. If this setting is zero, SRC module will not be used. | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE | |
| pv\_value | |
| Pointer to the input sampling frequency variable.  Valid value:  0: disable SRC module  48,000/44,100 Hz: setting input sampling rate for SRC module | |
| 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_CAP\_CONFIG\_FATAL\_SAMPLE\_RATE | Input TDM PCM sampling frequency is out of range. |
| Restrictions | - | |

FD\_PLG\_TDM\_032

[Covers: RD\_014]

Example

WORD32 sample\_rate;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE,

&sample\_rate);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_CONFIG\_PARAM\_FRAME\_SIZE | |
| Description | Set input/output TDM PCM frame size in sample | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_FRAME\_SIZE | |
| pv\_value | |
| Pointer to frame size variable  (valid value: 512 / 1024 / 2048) | |
| 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_CAP\_CONFIG\_FATAL\_FRAME\_SIZE | TDM PCM frame size value is out of range. |
| Restrictions | - | |

FD\_PLG\_TDM\_033

[Covers: RD\_014]

Example

WORD32 frame\_size;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_FRAME\_SIZE,

&frame\_size);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT1 | |
| Description | Set 1st input source device for TDM Capture | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT1 | |
| pv\_value | |
| Pointer to the input device value 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_CAP\_CONFIG\_FATAL\_INVALID\_INPUT | TDM PCM input device is out of range. |
| Restrictions | List of supported module:   |  |  | | --- | --- | | Macro | Value | | SSI00 | 0 | | SSI10 | 10 | | SSI20 | 20 | | SSI30 | 30 | | SSI40 | 40 | | SSI90 | 90 | | SCU\_SRCI0 | 110 | | SCU\_SRCI1 | 111 | | SCU\_SRCI3 | 113 | | SCU\_SRCI4 | 114 | | |

FD\_PLG\_TDM\_034

[Covers: RD\_014]

Example

WORD32 input\_source;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT1,

&input\_source);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_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\_TDM\_CAP\_CONFIG\_PARAM\_DMACHANNEL1 | |
| pv\_value | |
| Pointer to the Audio-DMAC / Audio-DMAC-peripheral-peripheral channels number  ADMAC\_CH[0-31] : Audio-DMAC usage  ADMACPP\_CH[0-28] : Audio-DMAC-pp usage | |
| 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 or call after post-configuration step) |
| XA\_CAP\_CONFIG\_FATAL\_DMACHANNEL | TDM PCM ADMA channel setting is out of range. |
| Restrictions | - | |

FD\_PLG\_TDM\_035

[Covers: RD\_014]

Example

WORD32 dma\_channel;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_CAP\_CONFIG\_PARAM\_DMACHANNEL1,

&dma\_channel);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT2 | |
| Description | Set 2nd input source device for TDM Capture | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT2 | |
| pv\_value | |
| Pointer to the input device value 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_CAP\_CONFIG\_FATAL\_INVALID\_INPUT | TDM PCM input device is out of range. |
| Restrictions | List of supported module:   |  |  | | --- | --- | | Macro | Value | | SSI00 | 0 | | SSI10 | 10 | | SSI20 | 20 | | SSI30 | 30 | | SSI40 | 40 | | SSI90 | 90 | | SCU\_SRCI0 | 110 | | SCU\_SRCI1 | 111 | | SCU\_SRCI3 | 113 | | SCU\_SRCI4 | 114 | | |

FD\_PLG\_TDM\_036

[Covers: RD\_014]

Example

WORD32 input\_source;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT2,

&input\_source);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_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\_TDM\_CAP\_CONFIG\_PARAM\_DMACHANNEL2 | |
| pv\_value | |
| Pointer to the Audio-DMAC / Audio-DMAC-peripheral-peripheral channels number  XA\_TDM\_CAP\_ADMAC\_CH[0-31] : Audio-DMAC usage  XA\_TDM\_CAP\_ADMACPP\_CH[0-28] : Audio-DMAC-pp usage | |
| 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 or call after post-configuration step) |
| XA\_CAP\_CONFIG\_NONFATAL\_ERR\_DMACHANNEL | TDM PCM ADMA channel setting is out of range. |
| Restrictions | - | |

FD\_PLG\_TDM\_037

[Covers: RD\_014]

Example

WORD32 dma\_channel;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_CAP\_CONFIG\_PARAM\_DMACHANNEL2,

&dma\_channel);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_CONFIG\_PARAM\_OUT\_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\_TDM\_CAP\_CONFIG\_PARAM\_OUT\_SAMPLE\_RATE | |
| pv\_value | |
| Pointer to the output sampling frequency variable.  Valid value: (32,000 / 44,100 / 48,000 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_CAP\_CONFIG\_FATAL\_SAMPLE\_RATE | TDM PCM output sample rate is out of range. |
| Restrictions | - | |

FD\_PLG\_TDM\_038

[Covers: RD\_014]

Example:

WORD32 sample\_rate;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_OUT\_SAMPLE\_RATE,

&sample\_rate);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_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 setting values except 0xFFFF FFFF (disable) will enabled DVC of SCU module and the connection will be established even without setting connection path. | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_VOLUME\_RATE | |
| pv\_value | |
| Pointer to the volume ratio number (using Fix-point Q3.20):  0xFFFF FFFF : disable DVC module  [0, 0x7F FFFF] : setting volume rate 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step or call after post-configuration step) |
| XA\_TDM\_CAP\_CONFIG\_FATAL\_VOLUME\_RATE | TDM PCM volume rate value is out of range. |
| Restrictions | - | |

FD\_PLG\_TDM\_039

[Covers: RD\_014]

Example:

WORD32 vol\_rate;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_SET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_VOLUME\_RATE,

&vol\_rate);

#### XA\_API\_CMD\_GET\_CONFIG\_PARAM command

2.2.2.16.1 Get configuration command for TDM Renderer

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_RDR\_CONFIG\_PARAM\_PCM\_WIDTH | |
| Description | Get the TDM PCM sample bit width setting | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_GET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_040

[Covers: RD\_014]

Example

WORD32 pcm\_width;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_PCM\_WIDTH,

&pcm\_width);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_RDR\_CONFIG\_PARAM\_CHANNEL\_MODE | |
| Description | Get TDM PCM channels mode setting | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_GET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_CHANNEL\_MODE | |
| pv\_value | |
| Pointer to the TDM channels mode | |
| 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_041

[Covers: RD\_014]

Example:

WORD32 ch\_mode;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_CHANNEL\_MODE,

&ch\_mode);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_RDR\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE | |
| Description | Get input TDM PCM sampling frequency setting | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_GET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE | |
| pv\_value | |
| Pointer to the input 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_042

[Covers: RD\_014]

Example

WORD32 sample\_rate;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE,

&sample\_rate);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_RDR\_CONFIG\_PARAM\_FRAME\_SIZE | |
| Description | Get input/output TDM PCM frame size in sample setting | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_GET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_FRAME\_SIZE | |
| pv\_value | |
| Pointer to frame size in sample 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_043

[Covers: RD\_014]

Example

WORD32 frame\_size;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_FRAME\_SIZE,

&frame\_size);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT1 | |
| Description | Get 1st output destination device for TDM Renderer info | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_GET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT1 | |
| pv\_value | |
| Pointer to output destination value 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_044

[Covers: RD\_014]

Example:

WORD32 output\_dev;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT1,

&output\_dev);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_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\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL1 | |
| pv\_value | |
| Pointer to the Audio-DMAC / Audio-DMAC-peripheral-peripheral 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_045

[Covers: RD\_014]

Example:

WORD32 dma\_channel;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL1,

&dma\_channel);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT2 | |
| Description | Get 2nd output destination device for TDM Renderer info | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_GET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT2 | |
| pv\_value | |
| Pointer to output destination value 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_046

[Covers: RD\_014]

Example:

WORD32 output\_dev;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_OUTPUT2,

&output\_dev);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_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\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL2 | |
| pv\_value | |
| Pointer to the Audio-DMAC / Audio-DMAC-peripheral-peripheral 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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_047

[Covers: RD\_014]

Example:

WORD32 dma\_channel;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_DMACHANNEL2,

&dma\_channel);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_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\_TDM\_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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_048

[Covers: RD\_014]

Example:

WORD32 sample\_rate;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_OUT\_SAMPLE\_RATE,

&sample\_rate);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_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\_TDM\_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\_TDM\_RDR\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_049

[Covers: RD\_014]

Example:

WORD32 vol\_rate;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_RDR\_CONFIG\_PARAM\_VOLUME\_RATE,

&vol\_rate);

2.2.2.16.2 Get configuration command for TDM Capture

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_CONFIG\_PARAM\_PCM\_WIDTH | |
| Description | Get TDM PCM sample bit width setting | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_GET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_050

[Covers: RD\_014]

Example

WORD32 pcm\_width;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_PCM\_WIDTH,

&pcm\_width);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_CONFIG\_PARAM\_CHANNEL\_MODE | |
| Description | Get TDM PCM channels mode setting | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_GET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_CHANNEL\_MODE | |
| pv\_value | |
| Pointer to the TDM channels mode 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_051

[Covers: RD\_014]

Example

WORD32 ch;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_CHANNEL\_MODE,

&ch);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE | |
| Description | Get the PCM sampling frequency setting value | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_GET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE | |
| pv\_value | |
| Pointer to the input 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_052

[Covers: RD\_014]

Example

WORD32 sample\_rate;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_IN\_SAMPLE\_RATE,

&sample\_rate);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_CONFIG\_PARAM\_FRAME\_SIZE | |
| Description | Get input/output TDM PCM frame size in sample setting | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_GET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_053

[Covers: RD\_014]

Example

WORD32 frame\_size;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_FRAME\_SIZE,

&frame\_size);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT1 | |
| Description | Get 1st input source device for TDM Capture info | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_GET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT1 | |
| pv\_value | |
| Pointer to the input destination value 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_054

[Covers: RD\_014]

Example

WORD32 input\_source;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT1,

&input\_source);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_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\_TDM\_CAP\_CONFIG\_PARAM\_DMACHANNEL1 | |
| pv\_value | |
| Pointer to the Audio-DMAC / Audio-DMAC-peripheral-peripheral 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 (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_055

[Covers: RD\_014]

Example

WORD32 dma\_channel;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_CAP\_CONFIG\_PARAM\_DMACHANNEL1,

&dma\_channel);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT2 | |
| Description | Get 2nd input source device for TDM Capture info | |
| Arguments | p\_xa\_module\_obj | |
| Pointer to API Structure. | |
| i\_cmd | |
| XA\_API\_CMD\_GET\_CONFIG\_PARAM | |
| i\_idx | |
| XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT2 | |
| pv\_value | |
| Pointer to the input destination value 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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_056

[Covers: RD\_014]

Example

WORD32 input\_source;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_INPUT2,

&input\_source);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_CAP\_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\_TDM\_CAP\_CONFIG\_PARAM\_DMACHANNEL2 | |
| pv\_value | |
| Pointer to the Audio-DMAC / Audio-DMAC-peripheral-peripheral 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 (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_057

[Covers: RD\_014]

Example

WORD32 dma\_channel;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_CAP\_CONFIG\_PARAM\_DMACHANNEL2,

&dma\_channel);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_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\_TDM\_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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_058

[Covers: RD\_014]

Example:

WORD32 sample\_rate;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_OUT\_SAMPLE\_RATE,

&sample\_rate);

|  |  |  |
| --- | --- | --- |
| Subcommand | XA\_TDM\_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\_TDM\_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\_TDM\_CAP\_CONFIG\_FATAL\_STATE | Incorrect sequence call (i.e. call before pre-configuration step) |
| Restrictions | - | |

FD\_PLG\_TDM\_059

[Covers: RD\_014]

Example:

WORD32 vol\_rate;

res = (\*api\_func)(api\_obj,

XA\_API\_CMD\_GET\_CONFIG\_PARAM,

XA\_TDM\_CAP\_CONFIG\_PARAM\_VOLUME\_RATE,

&vol\_rate);

## 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 | Outline |
| --- | --- | --- |
| XARelTDMrdr | 1456 bytes | API’s structure to stores the information of API |
| XARelTDMcap | 1440 bytes | API’s structure to stores the information of API |

### XARelTDMrdr type structure

The XARelTDMrdr type structure is the work area used by the TDM Renderer of TDM 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 XARelTDMrdr type structure information

| Member name | Outline |
| --- | --- |
| pVOID pMem\_tabs | Pointer to memory tables |
| WORD32 persist\_size | Size of persistent memory |
| WORD32 descript\_size | Descriptor memory size |
| WORD32 ring\_size | Total size of ring buffer in sample |
| WORD32 sample\_size | Size of PCM sample in byte (respect channels and PCM width) |
| WORD32 input\_total | Number of input port based on channels mode of TDM plugin |
| WORD32 channels | Format channel of input PCM data |
| relTDMrdr\_Parameters parameters | Parameter structure of TDM renderer plugin |
| DMAC\_SETTING dma\_params | ADMAC parameters structure |
| WORD32 output1\_type | 1st audio device type |
| WORD32 output2\_type | 2nd audio device type |
| WORD32 dma1\_type | 1st DMAC connection type |
| WORD32 dma2\_type | 2nd DMAC connection type |
| SSIU\_SSI\_MODULE ssi\_module | SSI module information |
| SRC\_MODULES src\_module | SRC module information |
| CMD\_MODULE cmd\_module | CMD module information |
| Fifo\_modules fifo\_module | FIFO module information |
| WORD32 state | TDM renderer state |
| WORD32 dmac\_stage | ADMAC stage flag |
| WORD32 hw\_module | Store module information used in plugin |
| WORD32 write\_idx | FIFO writing position |
| WORD32 read\_idx | Software reading position |
| WORD32 filled | Number of sample present in the buffer |
| WORD32 merging\_count | Number of bytes plugin has written into scratch area |
| WORD32 port\_filled[4] | Number of bytes port has been submitted from user |
| WORD32 merging\_done | Flag to tell plugin that merging process is done or not yet |
| WORD32 consumed[4] | Number of byte consumed in each port |
| XosEvent relrdr\_event | TDM Renderer polling event |
| XosThread relrdr\_thread | TDM Renderer polling thread |

### XARelTDMcap type structure

The XARelTDMcap type structure is the work area used by the TDM Capture of TDM 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 XARelTDMcap type structure information

| Member name | Outline |
| --- | --- |
| pVOID pMem\_tabs | Pointer to memory tables |
| WORD32 persist\_size | Size of persistent memory |
| WORD32 descript\_size | Descriptor memory size |
| WORD32 ring\_size | Total size of ring-buffer in sample |
| WORD32 sample\_size | Size of PCM sample in byte (respect channels and PCM width) |
| WORD32 output\_total | Number of output port based on channel mode of plugin |
| WORD32 channels | Format channel of input PCM data |
| relTDMcap\_Parameters parameters | Parameter structure of TDM Capturer plugin |
| DMAC\_SETTING dma\_params | ADMAC parameters structure |
| WORD32 input1\_type | 1st audio device type |
| WORD32 input2\_type | 2nd audio device type |
| WORD32 dma1\_type | 1st DMAC connection type |
| WORD32 dma2\_type | 2nd DMAC connection type |
| SSIU\_SSI\_MODULE ssi\_master | SSI module master information |
| SSIU\_SSI\_MODULE ssi\_slave | SSI module slaver information |
| SRC\_MODULES src\_module | SRC module information |
| CMD\_MODULE cmd\_module | CMD module information |
| Fifo\_modules fifo\_module | FIFO module information |
| WORD32 state | TDM Capture plugin current state |
| WORD32 dmac\_stage | ADMAC stage flag |
| WORD32 hw\_module | Store module information used in plugin |
| WORD32 head\_idx | Head index of ring buffer |
| WORD32 tail\_idx | Tail index of ring buffer |
| WORD32 filled | Number of samples present in the buffer |
| WORD32 transfered\_idx | Index of output port has been transferred in the last execution |
| WORD32 transferred[4] | Number of byte transferred by plugin for each port |
| XosEvent relcap\_event | TDM Capture polling event |
| XosThread relcap\_thread | TDM Capture polling thread |

## Memory Specifications

This section describes the memory areas used by this software.

### Persistent Area

Table 2‑15 Persistent Area Description

| Item | Area which always holds values when this software is used.  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. |

### Stack Area

This software does not use a stack area.

### Heap Area

This software does not use a heap area.

### Input Buffer

Input buffer only is used in the TDM Renderer case.

Table 2‑16 Input Buffer Description

| Item | Area which stores inputs from this software.  The input buffer contains 16-bit or 24-bit linear PCM data (hereinafter called PCM data).  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.  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. |

### Output Buffer

Output buffer only is used in the TDM Capture case.

Table 2‑17 Output Buffer Description

| Item | Area which stores outputs from this software.  The output buffer contains 16-bit or 24-bit linear PCM data (hereinafter called PCM data).  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.  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 2‑3(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 2‑1).

0x12

16 bit PCM Data

Offset [byte]

Access Image

0x34

0x12

0x34

MSB

LSB

+0

+1

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

0x12

24 bit PCM Data

Offset [byte]

Access Image

0x34

0x12

0x34

MSB

LSB

+1

+2

0x56

0x56

+0

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | Output PCM Data | |  |
| MSB |  |  | LSB |
|  | Channel-0 Data0 | |  |
| +1 | Channel-1 Data0 | |  |
| +2 | Channel-2 Data0 | |  |
| +3 | Channel-3 Data0 | |  |
| +4 | Channel-4 Data0 | |  |
| +5 | Channel-5 Data0 | |  |
| +6 | Channel-6 Data0 | |  |
| +7 | Channel-7 Data0 | |  |
|  | . . . | | 1024sample x 8ch |
|  |  |
|  |  |
|  |  |
| +4088 | Channel-0 Data511 | |  |
| +4089 | Channel-1 Data511 | |  |
| +4090 | Channel-2 Data511 | |  |
| +4091 | Channel-3 Data511 | |  |
| +4092 | Channel-4 Data511 | |  |
| +4093 | Channel-5 Data511 | |  |
| +4094 | Channel-6 Data511 | |  |
| +4095 | Channel-7 Data511 | |  |
|  | ・・・ | |  |
|  |  |
|  |  |
|  |  |
| +8190 | Channel-6 Data1023 | |  |
| +8191 | Channel-7 Data1023 | |  |
| [Sample] | | | |
| Stereo Output Format | | | |

Figure 2‑4 Output Formats

## Error Processing

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

### Error codes

Below are the error codes for this software.

Table 2‑18 Error Codes for TDM Renderer

| Error code (32bit) | Value | Description |
| --- | --- | --- |
| [1]  XA\_NO\_ERROR | 0x00000000 | The processing results are normal.  The process has terminated normally. |
| [2]  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.  Because it becomes the common API error, please check the correct procedure. |
| [3]  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]  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]  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]  XA\_TDM\_RDR\_EXEC\_FATAL\_STATE | 0xFFFF9080 | Abnormality has occurred, which disables process continuation. The command does not follow procedure. Because it becomes the common API error, please check the correct procedure. |
| [7]  XA\_TDM\_RDR\_EXEC\_FATAL\_INPUT | 0xFFFF9081 | 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. |
| [8]  XA\_TDM\_RDR\_EXEC\_FATAL\_INTERNAL | 0xFFFF9082 | Abnormality has occurred, which disables process continuation. Some of setting becomes incorrect after combination (out of memory, hardware modules are not available...). Because it becomes the common API error, please check the correct parameters and make sure the resource is validity. |

| [9]  XA\_TDM\_RDR\_CONFIG\_FATAL\_STATE | 0xFFFF8880 | Abnormality has occurred, which disables process continuation. The command does not follow procedure. Because it becomes the common API error, please check the correct procedure. |
| --- | --- | --- |
| [10]  XA\_TDM\_RDR\_CONFIG\_FATAL\_PCM\_WIDTH | 0xFFFF8881 | It is an error for TDM Renderer specifications out of the range.  The pcm width value was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [11]  XA\_TDM\_RDR\_CONFIG\_FATAL\_CHANNEL\_MODE | 0xFFFF8882 | It is an error for TDM Renderer specifications out of the range.  The channel mode value was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [12]  XA\_TDM\_RDR\_CONFIG\_FATAL\_SAMPLE\_RATE | 0xFFFF8883 | It is an error for TDM Renderer specifications out of the range.  The sample rate value was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [13]  XA\_TDM\_RDR\_CONFIG\_FATAL\_FRAME\_SIZE | 0xFFFF8884 | It is an error for TDM Renderer specifications out of the range.  The frame size was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [14]  XA\_TDM\_RDR\_CONFIG\_FATAL\_INVALID\_OUTPUT | 0xFFFF8885 | It is an error for TDM Renderer specifications out of the range.  The output value was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16 |
| [15]  XA\_TDM\_RDR\_CONFIG\_FATAL\_DMACHANNEL | 0xFFFF8886 | It is an error for TDM Renderer specifications out of the range.  The adma channel value was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [16]  XA\_TDM\_RDR\_CONFIG\_FATAL\_VOLUME\_RATE | 0xFFFF8887 | It is an error for TDM Renderer specifications out of the range.  The volume rate value was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [17] | Others | Reserved |

Table 2‑19 Error Codes for TDM Capture

| Error code (32bit) | Value | Description |
| --- | --- | --- |
| [1]  XA\_NO\_ERROR | 0x00000000 | The processing results are normal.  The process has terminated normally. |
| [2]  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.  Because it becomes the common API error, please check the correct procedure. |
| [3]  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]  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]  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]  XA\_TDM\_CAP\_EXEC\_FATAL\_STATE | 0xFFFF90C0 | Abnormality has occurred, which disables process continuation. The command does not follow procedure. Because it becomes the common API error, please check the correct procedure. |
| [8]  XA\_TDM\_CAP\_EXEC\_FATAL\_INTERNAL | 0xFFFF90C1 | Abnormality has occurred, which disables process continuation. Some of setting becomes incorrect after combination (out of memory, hardware module not available...). Because it becomes the common API error, please check the correct parameters and make sure the resource is validity. |

| [9]  XA\_TDM\_CAP\_CONFIG\_FATAL\_STATE | 0xFFFF88C0 | Abnormality has occurred, which disables process continuation. The command does not follow procedure. Because it becomes the common API error, please check the correct procedure. |
| --- | --- | --- |
| [10]  XA\_TDM\_CAP\_CONFIG\_FATAL\_PCM\_WIDTH | 0xFFFF88C1 | It is an error for TDM Capture specifications out of the range.  The pcm width value was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [11]  XA\_TDM\_CAP\_CONFIG\_FATAL\_CHANNEL\_MODE | 0xFFFF88C2 | It is an error for TDM Capture specifications out of the range.  The channel mode value was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [12]  XA\_TDM\_CAP\_CONFIG\_FATAL\_SAMPLE\_RATE | 0xFFFF88C3 | It is an error for TDM Capture specifications out of the range.  The sample rate value was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [13]  XA\_TDM\_CAP\_CONFIG\_FATAL\_FRAME\_SIZE | 0xFFFF88C4 | It is an error for TDM Capture specifications out of the range.  The frame size was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [12]  XA\_TDM\_CAP\_CONFIG\_FATAL\_INVALID\_INPUT | 0xFFFF88C5 | It is an error for TDM Capture specifications out of the range.  The input value was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [13]  XA\_TDM\_CAP\_CONFIG\_FATAL\_DMACHANNEL | 0xFFFF88C6 | It is an error for TDM Capture specifications out of the range.  The adma channel was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [14]  XA\_TDM\_CAP\_CONFIG\_FATAL\_VOLUME\_RATE | 0xFFFF88C7 | It is an error for TDM Capture specifications out of the range.  The volume rate was specified at the argument does not support. Please set an appropriate value.(Refer to 2.2.2.16) |
| [15] | Others | Reservered |

# Processing Flow

Figure 3‑1 shows a flow diagram of processing performed by an application which uses this software. It applies for both case: TDM renderer and TDM capture.

The basic steps executed by the framework are white. The steps defined by the user framework are shaded. Design the process to suit the target system.

Initialize Processing

END

START

Read from input.

Indicate INPUT\_OVER if end of input.

Process data buffer

Transfer to output

Set the I/O buffer

(Table 2-7 No.8)

Initialize

plug-in

Error

Yes

No

Short of buffer size?

Set memory blocks

Get configuration parameters.

Start-up

API

Set the number of channels and PCM width, sample rate

Invalid parameters?

Yes

Get memory blocks info

Set the default parameters (API).

Get the size of the API structure.

No

Get version info.

Parameters Getting

Memory Allocation

Execution

Parameters Setting

Figure 3‑1 Example of the Application Processing Flow

# Appendix

Below matrix tables show behavior of TDM plugin when user sets different sampling rate (Fs) (Hz) to plugin.

Table 4‑1 Matrix table for sampling rate setting of TDM Renderer

|  |  |  |  |
| --- | --- | --- | --- |
| Input Fs  Output Fs | 32000 | 44100 | 48000 |
| 32000 | - | - | - |
| 44100 | ○ | ○ | ○ |
| 48000 | ○ | ○ | ○ |
| 0 (Non-use SRC) | \* | ○ | ○ |

Table 4‑2 Matrix table for sampling rate setting of TDM Capture

|  |  |  |  |
| --- | --- | --- | --- |
| Output Fs  Input Fs | 32000 | 44100 | 48000 |
| 32000 | - | - | - |
| 44100 | ○ | ○ | ○ |
| 48000 | ○ | ○ | ○ |
| 0 (Non-use SRC) | \* | ○ | ○ |

○ : Plugin runs as normal

- : Plugin returns error due to invalid sample rate setting

\* : Plugin enables SRC module automatically to perform sample rate conversion