

**CONFIDENTIAL**

# ADSP Reference Equalizer Plugin RCG3AHPLN0201ZDO

User's Manual

RCG3AHPLN0201ZDOE

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (<http://www.renesas.com>).

Rev. 2.00 Dec, 2018

# CONFIDENTIAL

## Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
  2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
  3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
  4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
  5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.  
"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.  
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.  
Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
  6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
  7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
  8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
  9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
  10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
  11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
  12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.
- (Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
- (Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

## CONFIDENTIAL

### Trademarks

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

# How to Use This Manual

## 1. Purpose and Target Reader

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

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

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

## 2. Restrictions on the Use of this Software

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

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

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

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

### 3. Related Manuals

### 4. Technical Terms and Abbreviation

## - Table of Contents -

|  |           |
|--|-----------|
| <b>1. OVERVIEW</b>   | <b>3</b>  |
| 1.1. Overview of this document.                                  | 3         |
| 1.2. The architecture of the Software and scope of this document | 3         |
| 1.3. Specification overview                                      | 3         |
| 1.4. Memory specification  | 4         |
| 1.5. Related documents   | 4         |
| <b>2. SOFTWARE SPECIFICATIONS</b>                                | <b>5</b>  |
| 2.1. API sepecifications   | 5         |
| 2.2. Command list  | 6         |
| 2.2.1. Startup API   | 7         |
| 2.2.2. Parameters Setting  | 7         |
| 2.2.3. Memory Allocation   | 8         |
| 2.2.4. Initialization  | 8         |
| 2.2.5. Parameters Getting  | 9         |
| 2.2.6. Equalizer executing                                       | 9         |
| 2.3. The list of structures                                      | 10        |
| 2.4. Command function specifications                             | 11        |
| 2.4.1. XA_API_CMD_GET_LIB_ID_STRINGS command                     | 11        |
| 2.4.2. XA_API_CMD_GET_API_SIZE command                           | 13        |
| 2.4.3. XA_API_CMD_INIT command                                   | 14        |
| 2.4.4. XA_API_CMD_SET_CONFIG_PARAM command                       | 18        |
| 2.4.5. XA_API_CMD_GET_MEMTABS_SIZE command                       | 28        |
| 2.4.6. XA_API_CMD_SET_MEMTABS_PTR command                        | 29        |
| 2.4.7. XA_API_CMD_GET_N_MEMTABS command                          | 30        |
| 2.4.8. XA_API_CMD_GET_MEM_INFO_SIZE command                      | 31        |
| 2.4.9. XA_API_CMD_GET_MEM_INFO_TYPE command                      | 32        |
| 2.4.10. XA_API_CMD_GET_MEM_INFO_ALIGNMENT command                | 33        |
| 2.4.11. XA_API_CMD_SET_MEM_PTR command                           | 34        |
| 2.4.12. XA_API_CMD_GET_CONFIG_PARAM command                      | 35        |
| 2.4.13. XA_API_CMD_SET_INPUT_BYTES command                       | 45        |
| 2.4.14. XA_API_CMD_INPUT_OVER command                            | 46        |
| 2.4.15. XA_API_CMD_EXECUTE command                               | 47        |
| 2.4.16. XA_API_CMD_GET_OUTPUT_BYTES command                      | 49        |
| 2.4.17. XA_API_CMD_GET_CURIDX_INPUT_BUF command                  | 50        |
| 2.5. Memory Specifications                                       | 51        |
| 2.5.1. Persistent Area   | 51        |
| 2.5.2. Input Buffer  | 52        |
| 2.5.3. Output Buffer   | 53        |
| 2.6. Structures specification                                    | 56        |
| 2.6.1. API Structure   | 56        |
| 2.6.2. Persistent Structure                                      | 56        |
| 2.6.3. Equalizer settings structure                              | 57        |
| 2.6.4. Biquad filter structure                                   | 58        |
| 2.7. Error processing  | 59        |
| <b>3. PROCESSING FLOW</b>  | <b>62</b> |
| <b>4. NOTES</b>  | <b>64</b> |
| 4.1. Function Call   | 64        |
| 4.2. Other notes   | 64        |
| 4.2.1. Allocation of memory                                      | 64        |
| 4.2.2. Out of range memory access                                | 64        |
| 4.2.3. Combination with other applications                       | 64        |

|  |    |
|--|----|
| 4.2.4. Monitoring on Performance ..... | 64 |
|--|----|

## - List of Figures -

|  |    |
|--|----|
| Figure 1-1 The software architecture .....                   | 3  |
| Figure 2-1 API Command Sequence Overview .....               | 6  |
| Figure 2-2 PCM 16-bit Data Access (Little Endian Mode) ..... | 54 |
| Figure 2-3 PCM 24-bit Data Access (Little Endian Mode) ..... | 54 |
| Figure 2-4 Input (Output) Formats .....                      | 55 |
| Figure 3-1 Application Processing Flow .....                 | 63 |

## - List of Tables -

|  |    |
|--|----|
| Table 1-1 Basic Specifications .....                         | 3  |
| Table 1-2 Supported Specifications .....                     | 4  |
| Table 1-3 Memory Size Requirements .....                     | 4  |
| Table 1-4 Related documents .....                            | 4  |
| Table 2-1 API Functions of Equalizer .....                   | 5  |
| Table 2-2 Commands for Initialization .....                  | 7  |
| Table 2-3 Commands for Parameters Setting .....              | 7  |
| Table 2-4 Commands for Initial Memory Table Allocation ..... | 8  |
| Table 2-5 Commands for Memory Allocation .....               | 8  |
| Table 2-6 Commands for Equalizer Initialization .....        | 8  |
| Table 2-7 Commands for Getting Parameters .....              | 9  |
| Table 2-8 Commands for Equalizer Execution .....             | 9  |
| Table 2-9 Structures .....                                   | 10 |
| Table 2-10 Input Buffer Description .....                    | 52 |
| Table 2-11 Output Buffer Description .....                   | 53 |

## 1. Overview

### 1.1. Overview of this document.

In this chapter, overview of ADSP Equalizer plugin is explained.

### 1.2. The architecture of the Software and scope of this document

The architecture of ADSP Equalizer is shown in Figure 1-1. ADSP Equalizer is an ADSP plugin which is controlled by ADSP Framework.

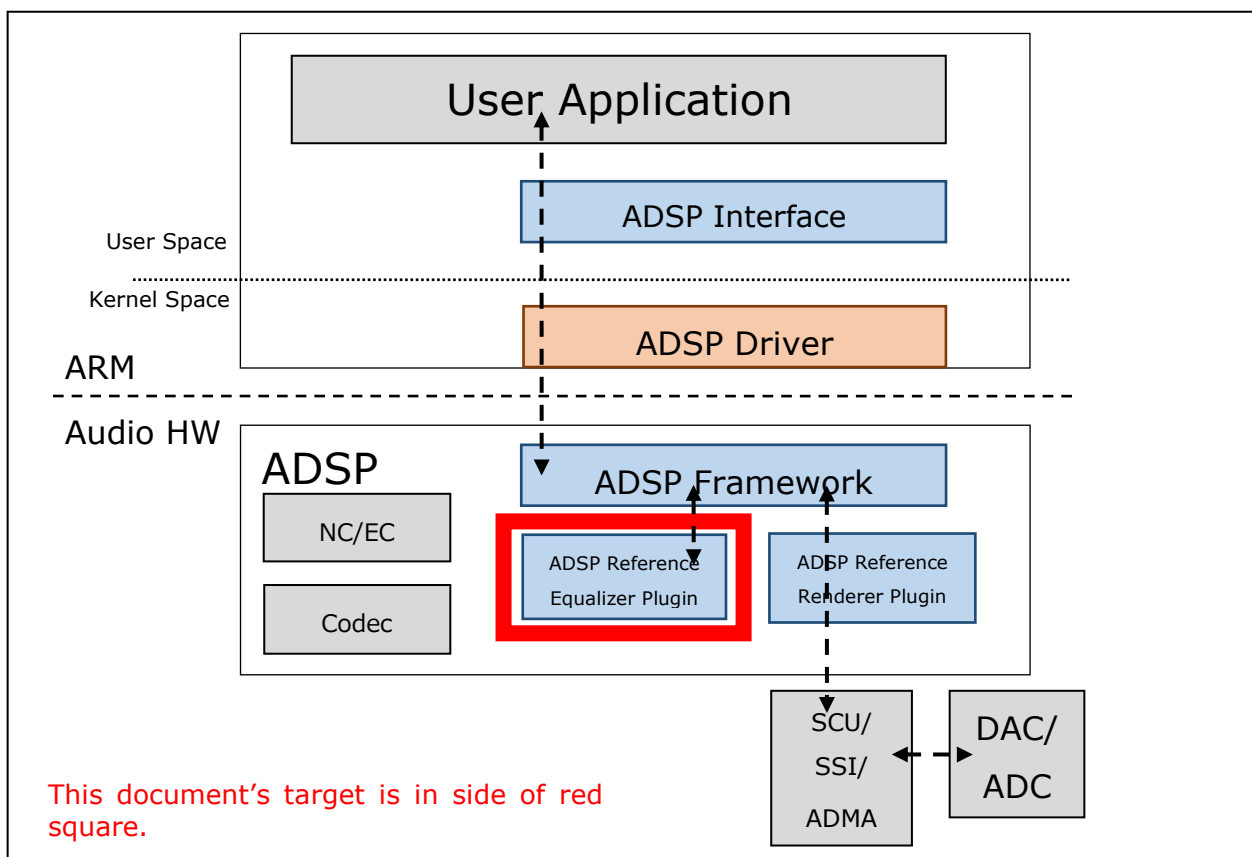


Figure 1-1 The software architecture

### 1.3. Specification overview

Equalizer changes frequency characteristic of the audio signal based on the parameter that was set and performs sound quality correction. Filter processing is performed for the PCM data which were input, and coordinates ingredient of the specific frequency band.

Table 1-1 shows the basic specification and Table 1-2 shows the support specification of Equalizer.

Table 1-1 Basic Specifications

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



|        |               |
|--------|---------------|
| Endian | Little Endian |
|--------|---------------|

Table 1-2 Supported Specifications

| Item                              | Description  |
|-----------------------------------|--|
| Input data format                 | 16-bit / 24-bit linear PCM (fixed point)               |
| Output data format                | 16-bit / 24-bit linear PCM (fixed point)               |
| Sampling frequency (Hz) supported | 48000 / 44100 / 32000                                  |
| Number of channels supported      | Max 2 channels   |
| Filter                            | Direct form II(second-order IIR digital biquad filter) |
| Band number of partitions         | 9 Band   |
| Reentrant                         | Supported  |
| Other                             | Coefficient change function                            |

## 1.4. Memory specification

Table 1-3 Memory Size Requirements

| Memory type | Location | Memory area name                     | Size (in bytes) |
|-------------|----------|--------------------------------------|-----------------|
| Instruction | ROM      | Instruction area                     | 16273           |
|             |          | Constant table area                  |                 |
|             |          | Other area( Depend on the compiler)  |                 |
|             | RAM      | Persistent area                      | 596             |
|             |          | Input buffer                         | 6144            |
|             |          | Output buffer                        | 6144            |
|             |          | Scratch area                         | 6144            |
|             |          | Structure                            | 240             |
|             |          | Stack                                | 336             |
|             |          | 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.

## 1.5. Related documents

Table 1-4 Related documents

| No. | Name                         | Published by                    |
|-----|------------------------------|---------------------------------|
| [1] | ADSP Framework User's Manual | Renesas Electronics Corporation |

## 2. Software Specifications

### 2.1. API sepecifications

Because one interface function accesses the procedure that was appointed by a command in equalizer, it is used.

Table 2-1 API Functions of Equalizer

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

## 2.2. Command list

Using API function of the Table 2-1, it performs each processing by a combination of Command/Subcommand with overview flow chart below

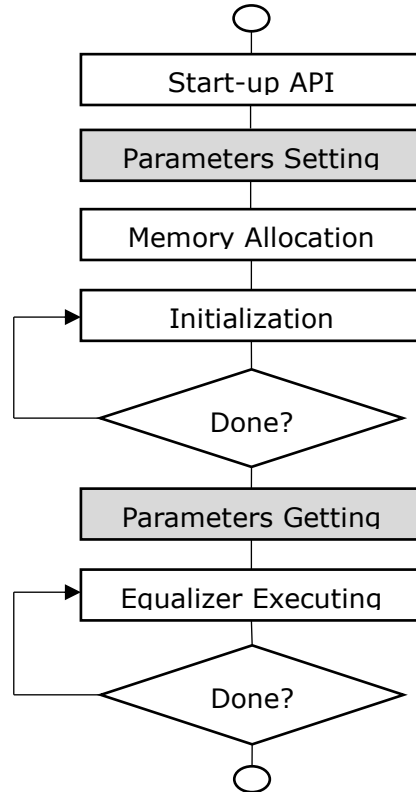


Figure 2-1 API Command Sequence Overview

### 2.2.1. Startup API

Table 2-2 Commands for Initialization

| Upper stage : Command / lower stage :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 structure.                       |
|   | 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_PARA MS |   |

### 2.2.2. Parameters Setting

Table 2-3 Commands for Parameters Setting

| Upper stage : Command / lower stage :Subcommand |  | Description  |
|---|--|--|
| 1   | XA_API_CMD_SET_CONFIG_PARAM              | Set the center frequency of a peaking filter or transition frequency of a Bass/Treble filter for filter n. |
|   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_FC   |  |
| 2   | XA_API_CMD_SET_CONFIG_PARAM              | Set type (Peaking, Bass, Treble) for filter n.   |
|   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_TYPE |  |
| 3   | XA_API_CMD_SET_CONFIG_PARAM              | Set bandwidth for filter n.  |
|   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BW   |  |
| 4   | XA_API_CMD_SET_CONFIG_PARAM              | Set gain for filter n.   |
|   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_GA   |  |
| 5   | XA_API_CMD_SET_CONFIG_PARAM              | Set base gain for filter n.  |
|   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BA   |  |
| 6   | XA_API_CMD_SET_CONFIG_PARAM              | Set graphic equalizer gain.  |
|   | XA_EQZ_CONFIG_PARAM_BAND_<n>_GCOEF_GA    |  |
| 7   | XA_API_CMD_SET_CONFIG_PARAM              | Set PCM data width.  |
|   | XA_EQZ_CONFIG_PARAM_PCM_WIDTH            |  |
| 8   | XA_API_CMD_SET_CONFIG_PARAM              | Set channel numbers.   |
|   | XA_EQZ_CONFIG_PARAM_CH                   |  |
| 9   | XA_API_CMD_SET_CONFIG_PARAM              | Set sampling frequency.  |
|   | XA_EQZ_CONFIG_PARAM_FS                   |  |
| 10  | XA_API_CMD_SET_CONFIG_PARAM              | Select equalizer type.   |
|   | XA_EQZ_CONFIG_PARAM_SELECT_EQZ_TYPE      |  |

[Note] <n> is index filter 0 to 8 for Parametric Equalizer and 0 to 4 for Graphic Equalizer. It will be specified in detailed subcommands.

### 2.2.3. Memory Allocation

Table 2-4 Commands for Initial Memory Table Allocation

| Upper stage : Command / lower stage :Subcommand |   | Description  |
|---|---|--|
| 1   | XA_API_CMD_GET_MEMTABS_SIZE             | Get the size of the memory structures to be allocated for the memory 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 equalizer 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 equalizer.  |
|   | (NULL)                                  |  |

Table 2-5 Commands for Memory Allocation

| Upper stage : Command / lower stage :Subcommand |                                   | Description  |
|---|-----------------------------------|--|
| 1   | XA_API_CMD_GET_MEM_INFO_SIZE      | Get the size of the each memory.   |
|   | (NULL)                            |  |
| 2   | XA_API_CMD_GET_MEM_INFO_ALIGNMENT | Get the alignment information of the memory-type being referred to by the index.   |
|   | (NULL)                            |  |
| 3   | XA_API_CMD_GET_MEM_INFO_TYPE      | Get the type of memory being referred to by the index.                             |
|   | (NULL)                            |  |
| 4   | XA_API_CMD_SET_MEM_PTR            | Set the pointer to the memory allocated for the referred index to the input value. |
|   | (NULL)                            |  |

### 2.2.4. Initialization

Table 2-6 Commands for Equalizer Initialization

| Upper stage : Command / lower stage :Subcommand |                                 | Description   |
|---|---------------------------------|---|
| 1   | XA_API_CMD_INPUT_OVER           | Signal the end of bit stream to the library.                              |
|   | (NULL)                          |   |
| 2   | XA_API_CMD_SET_INPUT_BYTES      | Set the number of bytes available in the input buffer for initialization. |
|   | (NULL)                          |   |
| 3   | XA_API_CMD_INIT                 | Initialize state and start run-time data process                          |
|   | 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 by the last initialization. |
|   | (NULL)                          |   |

### 2.2.5. Parameters Getting

Table 2-7 Commands for Getting Parameters

| Upper stage : Command / lower stage :Subcommand |  | Description  |
|---|--|--|
| 1   | XA_API_CMD_GET_CONFIG_PARAM              | Get the center frequency of a peaking filter or transition frequency of a Bass/Treble filter for filter n. |
|   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_FC   |  |
| 2   | XA_API_CMD_GET_CONFIG_PARAM              | Get type (Peaking, Bass, Treble) for filter n.   |
|   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_TYPE |  |
| 3   | XA_API_CMD_GET_CONFIG_PARAM              | Get bandwidth for filter n.  |
|   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BW   |  |
| 4   | XA_API_CMD_GET_CONFIG_PARAM              | Get gain for filter n.   |
|   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_GA   |  |
| 5   | XA_API_CMD_GET_CONFIG_PARAM              | Get base gain for filter n.  |
|   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BA   |  |
| 6   | XA_API_CMD_GET_CONFIG_PARAM              | Get graphic equalizer gain.  |
|   | XA_EQZ_CONFIG_PARAM_BAND_<n>_GCOEF_GA    |  |
| 7   | XA_API_CMD_GET_CONFIG_PARAM              | Get PCM data width.  |
|   | XA_EQZ_CONFIG_PARAM_PCM_WIDTH            |  |
| 8   | XA_API_CMD_GET_CONFIG_PARAM              | Get channel numbers.   |
|   | XA_EQZ_CONFIG_PARAM_CH                   |  |
| 9   | XA_API_CMD_GET_CONFIG_PARAM              | Get the sampling frequency of input signal.  |
|   | XA_EQZ_CONFIG_PARAM_FS                   |  |
| 10  | XA_API_CMD_SET_CONFIG_PARAM              | Get equalizer type.  |
|   | XA_EQZ_CONFIG_PARAM_SELECT_EQZ_TYPE      |  |

### 2.2.6. Equalizer executing

Table 2-8 Commands for Equalizer Execution

| Upper stage : Command / lower stage :Subcommand |                                 | Description  |
|---|---------------------------------|--|
| 1   | XA_API_CMD_INPUT_OVER           | Signal the end of bit stream to the library.                                 |
|   | (NULL)                          |  |
| 2   | XA_API_CMD_SET_INPUT_BYTES      | Set the number of bytes available in the input buffer for the execution.     |
|   | (NULL)                          |  |
| 3   | XA_API_CMD_EXECUTE              | This command executes the equalizer.   |
|   | XA_CMD_TYPE_DO_EXECUTE          |  |
| 4   | XA_API_CMD_EXECUTE              | Check if the end of processing has been reached.                             |
|   | XA_CMD_TYPE_DONE_QUERY          |  |
| 5   | XA_API_CMD_GET_OUTPUT_BYTES     | Get the number of bytes output by the equalizer in the last frame.           |
|   | (NULL)                          |  |
| 6   | XA_API_CMD_GET_CURIDX_INPUT_BUF | Get the number of input buffer bytes consumed by the last call to equalizer. |
|   | (NULL)                          |  |

### 2.3. The list of structures

Table 2-9 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 2.5

Table 2-9 Structures

| Structure name                          | Outline  |
|---|--|
| API structure                           | Store the information of API.  |
| Parametric Equalizer settings structure | Store a parameter to calculate a necessary filter coefficient as Parametric Equalizer. |
| Graphic Equalizer settings structure    | Store a parameter to calculate a necessary filter coefficient as Graphic Equalizer.    |
| Equalizer settings structure            | Store the parameters necessary for equalizer.  |
| Biquad filter structure                 | Store the parameters necessary for Biquad filter.                                      |

## 2.4. Command function specifications

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

|              |  |  |
|--------------|--|--|
| Subcommand   | XA_CMD_TYPE_LIB_VERSION  |  |
| Description  | This command obtains the version of the library in the form of a string. The maximum length of the string that the library will provide is 30 bytes. Therefore the application shall pass a pointer to a buffer of a minimum size of 30 bytes. |  |
| Parameter    | 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 | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - pv_value is NULL                                      |

#### 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. |  |
| Parameter    | 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 | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - pv_value is NULL                                  |

## Example

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

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

|              |  |   |
|--------------|--|---|
| Subcommand   | (None)   |   |
| Description  | This command is used to obtain the size of the API structure, in order to allocate memory for the API structure. |   |
| Parameter    | 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 | Normal   | XA_NO_ERROR                               |
|              | Error  | XA_API_FATAL_MEM_ALLOC - pv_value is NULL |
| Restrictions | The application shall allocate memory with an alignment of 4 bytes.  |   |

Example

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

### 2.4.3. XA\_API\_CMD\_INIT command

|              |  |  |
|--------------|--|--|
| Subcommand   | XA_CMD_TYPE_INIT_API_PRE_CONFIG_PARAMS   |  |
| Description  | This command is used to set the default value of the configuration parameters. |  |
| Parameter    | 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 | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes. |

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 equalizer specific parameters have been set. |  |
| Parameter    | 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 | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy. |

**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  | This command initializes the equalizer. It kicks run-time initialization process. |   |
| Parameter    | 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 | Normal  | XA_NO_ERROR   |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy.<br>XA_EQZ_EXEC_FATAL_STATE - if input, output and scratch memory are not allocated |

## Example

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

|              |  |  |
|--------------|--|--|
| Subcommand   | XA_CMD_TYPE_INIT_DONE_QUERY  |  |
| Description  | This command checks to see if the initialization process has completed. If it has, the flag value is set to 1; else, it is set to zero. A pointer to the flag variable is passed as an argument. |  |
| Parameter    | 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 | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj, pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy. |

**Example**

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

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

[Note] <n> is index filter 0, 1, 2... 8. It will be specified in detailed subcommands.

|              |  |  |
|--------------|--|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_FC   |  |
| Description  | This command is used to set the center frequency of a peaking filter or transition frequency of a Bass/Treble filter for filter n. |  |
| Parameter    | p_xa_module_obj  | Pointer to API Structure.  |
|              | i_cmd  | XA_API_CMD_SET_CONFIG_PARAM  |
|              | i_idx  | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_FC   |
|              | pv_value   | Address that stored center/ transition frequency.<br>Value range:<br><ul style="list-style-type: none"> <li>- Peaking filter: 20-20kHz (or less than Nyquist frequency)</li> <li>- Bass filter: 50-500Hz</li> <li>- Treble filter: 5k - 11kHz</li> </ul>               |
| Return value | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy.<br>XA_EQZ_CONFIG_NONFATAL_ERR_FC- Outside range for center/ transition frequency. |

Example

```
WORD32 coef_fc;
res = (*api_func)(api_obj,
                  XA_API_CMD_SET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_FC,
                  &coef_fc);
```

|              |   |  |
|--------------|---|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_TYPE  |  |
| Description  | This command is used to set type (Peaking, Bass, Treble, Through) for filter n. |  |
| Parameter    | p_xa_module_obj   | Pointer to API Structure.  |
|              | i_cmd   | XA_API_CMD_SET_CONFIG_PARAM  |
|              | i_idx   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_TYPE   |
|              | pv_value  | Address that stored filter type.<br>Value range: Peaking, Bass, Treble, Through filter<br>Through is default filter type.  |
| Return value | Normal  | XA_NO_ERROR  |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy.<br>XA_EQZ_CONFIG_NONFATAL_ERR_TYPE - Outside range for filter type. |

## Example

WORD32 coef\_type;

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

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

XA\_EQZ\_CONFIG\_PARAM\_FILTER\_&lt;n&gt;\_COEF\_TYPE,

&amp;coef\_type);



|              |   |   |
|--------------|---|---|
| Subcommand   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BW              |   |
| Description  | This command is used to set bandwidth for filter n. |   |
| Parameter    | p_xa_module_obj                                     | Pointer to API Structure.   |
|              | i_cmd   | XA_API_CMD_SET_CONFIG_PARAM   |
|              | i_idx   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BW  |
|              | pv_value  | Address that stored filter bandwidth.<br>Value range: $0.2 \times 2^{27}$ to $15 \times 2^{27}$ (fixed point Q5.27)   |
| Return value | Normal  | XA_NO_ERROR   |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy.<br>XA_EQZ_CONFIG_NONFATAL_ERR_BW - Outside range for filter bandwidth. |

Example

```
WORD32 coef_bw;
res = (*api_func)(api_obj,
                  XA_API_CMD_SET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BW,
                  &coef_bw);
```

|              |  |  |
|--------------|--|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_GA         |  |
| Description  | This command is used to set gain for filter n. |  |
| Parameter    | p_xa_module_obj                                | Pointer to API Structure.  |
|              | i_cmd  | XA_API_CMD_SET_CONFIG_PARAM  |
|              | i_idx  | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_GA   |
|              | pv_value                                       | Address that stored filter gain.<br>Value range: $10^{-\frac{15}{20}} \times 2^{28}$ to $10^{\frac{15}{20}} \times 2^{28}$ (fixed point Q4.28)   |
| Return value | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy.<br>XA_EQZ_CONFIG_NONFATAL_ERR_GA - Outside range for filter gain. |

[Note] Value range: -15dB to 15dB (fixed point Q4.28)

-15dB:  $10^{-\frac{15}{20}} \times 2^{28}$

15dB:  $10^{\frac{15}{20}} \times 2^{28}$

Example

WORD32 coef\_ga;

```
res = (*api_func)(api_obj,
                  XA_API_CMD_SET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_GA,
                  &coef_ga);
```

|              |   |   |
|--------------|---|---|
| Subcommand   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BA              |   |
| Description  | This command is used to set base gain for filter n. |   |
| Parameter    | p_xa_module_obj                                     | Pointer to API Structure.   |
|              | i_cmd   | XA_API_CMD_SET_CONFIG_PARAM   |
|              | i_idx   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BA  |
|              | pv_value  | Address that stored filter base gain.<br>Value range: $10^{-\frac{10}{20}} \times 2^{28}$ to $10^{\frac{10}{20}} \times 2^{28}$ (fixed point Q4.28)   |
| Return value | Normal  | XA_NO_ERROR   |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy.<br>XA_EQZ_CONFIG_NONFATAL_ERR_BA - Outside range for filter base gain. |

Example

```
WORD32 coef_ba;
res = (*api_func)(api_obj,
                  XA_API_CMD_SET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BA,
                  &coef_ba);
```

[Note] <m> is index filter 0, 1, 2... 4. It will be specified in detailed subcommands.

|              |   |  |
|--------------|---|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_<m>_GCOEF_GA                    |  |
| Description  | This command is used to set graphic equalizer gain. |  |
| Parameter    | p_xa_module_obj                                     | Pointer to API Structure.  |
|              | i_cmd   | XA_API_CMD_SET_CONFIG_PARAM  |
|              | i_idx   | XA_EQZ_CONFIG_PARAM_BAND_<n>_GCOEF_GA  |
|              | pv_value  | Address that stored graphic equalizer gain.<br>Value range: $10^{-\frac{10}{20}} \times 2^{28}$ to $10^{\frac{10}{20}} \times 2^{28}$ (fixed point Q4.28)  |
| Return value | Normal  | XA_NO_ERROR  |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy.<br>XA_EQZ_CONFIG_NONFATAL_ERR_GA - Outside range for filter gain. |

Example

```
WORD32 gcoef_ga;
res = (*api_func)(api_obj,
                  XA_API_CMD_SET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_FILTER_<n>_GCOEF_GA,
                  &gcoef_ga);
```

|              |   |  |
|--------------|---|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_PCM_WIDTH               |  |
| Description  | This command is used to set PCM data width. |  |
| Parameter    | p_xa_module_obj                             | Pointer to API Structure.  |
|              | i_cmd                                       | XA_API_CMD_SET_CONFIG_PARAM  |
|              | i_idx                                       | XA_EQZ_CONFIG_PARAM_PCM_WIDTH  |
|              | pv_value                                    | Address that stored PCM data width.<br>Value range: 16 bits /24 bits   |
| Return value | Normal                                      | XA_NO_ERROR  |
|              | Error                                       | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy.<br>XA_EQZ_CONFIG_FATAL_ERR_PCM_WIDTH - Outside range for PCM width. |

## Example

```
WORD32 pcm_width;  
res = (*api_func)(api_obj,  
                  XA_API_CMD_SET_CONFIG_PARAM,  
                  XA_EQZ_CONFIG_PARAM_PCM_WIDTH,  
                  &pcm_width);
```

|              |  |   |
|--------------|--|---|
| Subcommand   | XA_EQZ_CONFIG_PARAM_CH                       |   |
| Description  | This command is used to set channel numbers. |   |
| Parameter    | p_xa_module_obj                              | Pointer to API Structure.   |
|              | i_cmd  | XA_API_CMD_SET_CONFIG_PARAM   |
|              | i_idx  | XA_EQZ_CONFIG_PARAM_CH  |
|              | pv_value                                     | Address that stored channel numbers.<br>Value range: 1 or 2   |
| Return value | Normal                                       | XA_NO_ERROR   |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy.<br>XA_EQZ_CONFIG_FATAL_ERR_CH - Outside range for channel numbers. |

Example

```
WORD32 ch;
res = (*api_func)(api_obj,
                  XA_API_CMD_SET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_PARAM_CH,
                  &ch);
```

|              |   |  |
|--------------|---|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_FS  |  |
| Description  | This command is used to set the sampling frequency of input signal. |  |
| Parameter    | p_xa_module_obj   | Pointer to API Structure.  |
|              | i_cmd   | XA_API_CMD_SET_CONFIG_PARAM  |
|              | i_idx   | XA_EQZ_CONFIG_PARAM_FS   |
|              | pv_value  | Address that stored sampling frequency.<br>Value range: 48kHz / 44.1kHz / 32kHz  |
| Return value | Normal  | XA_NO_ERROR  |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_INVALID_CMD_TYPE - i_idx is incorrect.<br>XA_EQZ_CONFIG_FATAL_ERR_FS - Outside range for sampling frequency. |

Example

```
WORD32 param_fs;
res = (*api_func)(api_obj,
                  XA_API_CMD_SET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_PARAM_FS,
                  &param_fs);
```

|              |   |   |
|--------------|---|---|
| Subcommand   | XA_EQZ_CONFIG_PARAM_SELECT_EQZ_TYPE   |   |
| Description  | This command is used to select Equalizer type which is Parametric Equalizer or Graphic Equalizer. |   |
| Parameter    | p_xa_module_obj   | Pointer to API Structure.   |
|              | i_cmd   | XA_API_CMD_SET_CONFIG_PARAM   |
|              | i_idx   | XA_EQZ_CONFIG_PARAM_SELECT_EQZ_TYPE   |
|              | pv_value  | Address that stored index of Equalizer type.<br>Value range:<br>- Parametric Equalizer: 0<br>- Graphic Equalizer: 1   |
| Return value | Normal  | XA_NO_ERROR   |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_INVALID_CMD_TYPE - i_idx is incorrect.<br>XA_EQZ_CONFIG_NONFATAL_ERR_SELECT_EQZ_TYPE - Equalizer type does not support. |

Example

```
WORD32 eqz_type;
res = (*api_func)(api_obj,
                  XA_API_CMD_SET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_EQZ_TYPE,
                  &eqz_type);
```



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

|              |   |   |
|--------------|---|---|
| Subcommand   | None  |   |
| Description  | This command is used to obtain the size of the table used to hold memory blocks. These blocks required for the equalizer operation. The API returns the total size of the required table. |   |
| Parameter    | 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 | Normal  | XA_NO_ERROR   |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes<br>XA_EQZ_CONFIG_FATAL_STATE - Precondition is incorrect. |

##### Example

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

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

|              |  |   |
|--------------|--|---|
| Subcommand   | None   |   |
| Description  | This command is used to set the memory structure pointer to the allocated value. |   |
| Parameter    | 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 | Normal   | XA_NO_ERROR   |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d or pv_value is not aligned to 4 bytes<br>XA_EQZ_CONFIG_FATAL_STATE - Precondition is incorrect. |

##### Example

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

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

|              |   |  |
|--------------|---|--|
| Subcommand   | None  |  |
| Description  | This command is used to obtain the number of memory blocks the equalizer needs. 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. |  |
| Parameter    | p_xa_module_obj   | Pointer to API Structure.  |
|              | i_cmd   | XA_API_CMD_GET_N_MEMTABS   |
|              | i_idx   | NULL   |
|              | pv_value  | Number of memory blocks required to be allocated.  |
| Return value | Normal  | XA_NO_ERROR  |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes. |

##### Example

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

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

|              |  |  |
|--------------|--|--|
| Subcommand   | Memory index   |  |
| Description  | This command obtains the size of the memory type being referred to by the index. The size in bytes is returned in the variable pointed to by the final argument. |  |
| Parameter    | p_xa_module_obj  | Pointer to API Structure.  |
|              | i_cmd  | XA_API_CMD_GET_MEM_INFO_SIZE   |
|              | i_idx  | Index of the memory<br>0 - Persistent Area<br>1 - Input Buffer<br>2 - Output Buffer<br>3 - Scratch Area  |
|              | pv_value   | Pointer to memory size.  |
| Return value | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy.<br>XA_API_FATAL_INVALID_CMD_TYPE - i_idx is an invalid memory block number. |

##### Example

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

**2.4.9. XA\_API\_CMD\_GET\_MEM\_INFO\_TYPE command**

|              |  |  |
|--------------|--|--|
| Subcommand   | Memory index   |  |
| Description  | This command gets the type of memory being referred to by the index. |  |
| Parameter    | p_xa_module_obj  | Pointer to API Structure.  |
|              | i_cmd  | XA_API_CMD_GET_MEM_INFO_TYPE   |
|              | i_idx  | Index of the memory<br>0 - Persistent Area<br>1 - Input Buffer<br>2 - Output Buffer<br>3 - Scratch Area  |
|              | pv_value   | Pointer to memory type variable.   |
| Return value | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_API_FATAL_INVALID_CMD_TYPE - i_idx is an invalid memory block number. |

**Example**

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

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

|              |  |   |
|--------------|--|---|
| Subcommand   | Memory index   |   |
| Description  | This command gets the alignment information of the memory-type referred to by the index. The alignment required in bytes is returned to the application. |   |
| Parameter    | p_xa_module_obj  | Pointer to API Structure.   |
|              | i_cmd  | XA_API_CMD_GET_MEM_INFO_ALIGNMENT   |
|              | i_idx  | Index of the memory<br>0 - Persistent Area<br>1 - Input Buffer<br>2 - Output Buffer<br>3 - Scratch Area   |
|              | pv_value   | Pointer to the alignment info variable.   |
| Return value | Normal   | XA_NO_ERROR   |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_API_FATAL_INVALID_CMD_TYPE - i_idx is an invalid memory index. |

Example

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

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

|              |   |  |
|--------------|---|--|
| Subcommand   | Memory index  |  |
| Description  | This command passes to the equalizer the pointer to the allocated memory. |  |
| Parameter    | p_xa_module_obj   | Pointer to API Structure.  |
|              | i_cmd   | XA_API_CMD_SET_MEM_PTR   |
|              | i_idx   | Index of the memory<br>0 - Persistent Area<br>1 - Input Buffer<br>2 - Output Buffer<br>3 - Scratch Area  |
|              | pv_value  | Pointer to memory buffer allocated.  |
| Return value | Normal  | XA_NO_ERROR  |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_INVALID_CMD_TYPE - i_idx is an invalid memory block number.<br>XA_API_FATAL_MEM_ALIGN - d or pv_value is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy. |

**Example**

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

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

|              |  |  |
|--------------|--|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_PCM_WIDTH                    |  |
| Description  | This command is used to get PCM width parameter. |  |
| Parameter    | p_xa_module_obj                                  | Pointer to API Structure.  |
|              | i_cmd  | XA_API_CMD_GET_CONFIG_PARAM  |
|              | i_idx  | XA_EQZ_CONFIG_PARAM_PCM_WIDTH  |
|              | pv_value   | Pointer to PCM width.  |
| Return value | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy. |

Example

```
WORD32 pcm_width;
res = (*api_func)(api_obj,
                  XA_API_CMD_GET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_PCM_WIDTH,
                  &pcm_width);
```



|              |  |  |
|--------------|--|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_CH                           |  |
| Description  | This command is used to get PCM width parameter. |  |
| Parameter    | p_xa_module_obj                                  | Pointer to API Structure.  |
|              | i_cmd  | XA_API_CMD_GET_CONFIG_PARAM  |
|              | i_idx  | XA_EQZ_CONFIG_PARAM_CH   |
|              | pv_value   | Pointer to channel numbers.  |
| Return value | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy. |

Example

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

|              |  |  |
|--------------|--|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_FC   |  |
| Description  | This command is used to get the center frequency of a peaking filter or transition frequency of a Bass/Treble filter for filter n. |  |
| Parameter    | p_xa_module_obj  | Pointer to API Structure.  |
|              | i_cmd  | XA_API_CMD_GET_CONFIG_PARAM  |
|              | i_idx  | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_FC   |
|              | pv_value   | Pointer to center / transition frequency   |
| Return value | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy. |

**Example**

```
WORD32 coef_fc;
res = (*api_func)(api_obj,
                  XA_API_CMD_GET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_FC,
                  &coef_fc);
```

|              |   |  |
|--------------|---|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_TYPE  |  |
| Description  | This command is used to get type (Peaking, Bass, Treble, Through) for filter n. |  |
| Parameter    | p_xa_module_obj   | Pointer to API Structure.  |
|              | i_cmd   | XA_API_CMD_GET_CONFIG_PARAM  |
|              | i_idx   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_TYPE   |
|              | pv_value  | Pointer to filter type.  |
| Return value | Normal  | XA_NO_ERROR  |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy. |

**Example**

```
WORD32 coef_type;
res = (*api_func)(api_obj,
                  XA_API_CMD_GET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_TYPE,
                  &coef_type);
```

|              |   |  |
|--------------|---|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BW              |  |
| Description  | This command is used to get bandwidth for filter n. |  |
| Parameter    | p_xa_module_obj                                     | Pointer to API Structure.  |
|              | i_cmd   | XA_API_CMD_GET_CONFIG_PARAM  |
|              | i_idx   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BW   |
|              | pv_value  | Pointer to filter bandwidth.   |
| Return value | Normal  | XA_NO_ERROR  |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy. |

Example

```
WORD32 coef_bw;
res = (*api_func)(api_obj,
                  XA_API_CMD_GET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BW,
                  &coef_bw);
```

|              |  |  |
|--------------|--|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_GA         |  |
| Description  | This command is used to get gain for filter n. |  |
| Parameter    | p_xa_module_obj                                | Pointer to API Structure.  |
|              | i_cmd  | XA_API_CMD_GET_CONFIG_PARAM  |
|              | i_idx  | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_GA   |
|              | pv_value                                       | Pointer to filter gain.  |
| Return value | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy. |

Example

```
WORD32 coef_ga;
res = (*api_func)(api_obj,
                  XA_API_CMD_GET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_GA,
                  &coef_ga);
```

|              |   |  |
|--------------|---|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BA              |  |
| Description  | This command is used to get base gain for filter n. |  |
| Parameter    | p_xa_module_obj                                     | Pointer to API Structure.  |
|              | i_cmd   | XA_API_CMD_GET_CONFIG_PARAM  |
|              | i_idx   | XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BA   |
|              | pv_value  | Pointer to filter base gain.   |
| Return value | Normal  | XA_NO_ERROR  |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy. |

## Example

```
WORD32 coef_ba;
res = (*api_func)(api_obj,
                  XA_API_CMD_GET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_FILTER_<n>_COEF_BA,
                  &coef_ba);
```

|              |   |  |
|--------------|---|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_BAND_<n>_GCOEF_GA               |  |
| Description  | This command is used to get graphic equalizer gain. |  |
| Parameter    | p_xa_module_obj                                     | Pointer to API Structure.  |
|              | i_cmd   | XA_API_CMD_GET_CONFIG_PARAM  |
|              | i_idx   | XA_EQZ_CONFIG_PARAM_BAND_<n>_GCOEF_GA  |
|              | pv_value  | Pointer to graphic equalizer gain.   |
| Return value | Normal  | XA_NO_ERROR  |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy. |

**Example**

```
WORD32 gcoef_ga;
res = (*api_func)(api_obj,
                  XA_API_CMD_GET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_FILTER_<n>_GCOEF_GA,
                  &gcoef_ga);
```

|              |   |  |
|--------------|---|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_FS                              |  |
| Description  | This command is used to get the sampling frequency. |  |
| Parameter    | p_xa_module_obj                                     | Pointer to API Structure.  |
|              | i_cmd   | XA_API_CMD_GET_CONFIG_PARAM  |
|              | i_idx   | XA_EQZ_CONFIG_PARAM_FS   |
|              | pv_value  | Pointer to sampling frequency.   |
| Return value | Normal  | XA_NO_ERROR  |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy. |

**Example**

```
WORD32 param_fs;  
res = (*api_func)(api_obj,  
                  XA_API_CMD_GET_CONFIG_PARAM,  
                  XA_EQZ_CONFIG_PARAM_PARAM_FS,  
                  &param_fs);
```



|              |   |  |
|--------------|---|--|
| Subcommand   | XA_EQZ_CONFIG_PARAM_SELECT_EQZ_TYPE   |  |
| Description  | This command is used to select Equalizer type which is Parametric Equalizer or Graphic Equalizer. |  |
| Parameter    | p_xa_module_obj   | Pointer to API Structure.  |
|              | i_cmd   | XA_API_CMD_SET_CONFIG_PARAM  |
|              | i_idx   | XA_EQZ_CONFIG_PARAM_SELECT_EQZ_TYPE  |
|              | pv_value  | Address that stored index of Equalizer type.<br>Value range:<br>- Parametric Equalizer: 0<br>- Graphic Equalizer: 1  |
| Return value | Normal  | XA_NO_ERROR  |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_INVALID_CMD_TYPE - i_idx is incorrect.<br>XA_EQZ_CONFIG_NONFATAL_ERR_SELECT_EQZ_TYPE - Equalizer type does not support.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_CONFIG_FATAL_STATE - precondition does not satisfy. |

Example

```
WORD32 eqz_type;
res = (*api_func)(api_obj,
                  XA_API_CMD_GET_CONFIG_PARAM,
                  XA_EQZ_CONFIG_PARAM_EQZ_TYPE,
                  &eqz_type);
```

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

|              |  |  |
|--------------|--|--|
| Subcommand   | None   |  |
| Description  | This command is used to set the number of bytes available in the input buffer for the execution. |  |
| Parameter    | p_xa_module_obj  | Pointer to API Structure.  |
|              | i_cmd  | XA_API_CMD_SET_INPUT_BYTES   |
|              | i_idx  | NULL   |
|              | pv_value   | Pointer to the input byte variable.  |
| Return value | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - The pointer is incorrectly aligned to the requirements.<br>XA_EQZ_EXEC_FATAL_STATE - Precondition is incorrect.<br>XA_EQZ_EXEC_FATAL_INPUT - pv_value is invalid |

#### Example

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

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

|              |  |   |
|--------------|--|---|
| Subcommand   | None   |   |
| Description  | This command is used to tell the equalizer that the end of the input data has been reached. This situation can arise both in the initialization loop and the execute loop. |   |
| Parameter    | p_xa_module_obj  | Pointer to API Structure.   |
|              | i_cmd  | XA_API_CMD_INPUT_OVER   |
|              | i_idx  | NULL  |
|              | pv_value   | NULL  |
| Return value | Normal   | XA_NO_ERROR   |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj is NULL.<br>XA_API_FATAL_MEM_ALIGN - The pointer is incorrectly aligned to the requirements. |

## Example

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

## 2.4.15. XA\_API\_CMD\_EXECUTE command

|              |                                      |  |
|--------------|--------------------------------------|--|
| Subcommand   | XA_CMD_TYPE_DO_EXECUTE               |  |
| Description  | This command executes the equalizer. |  |
| Parameter    | 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 | Normal                               | XA_NO_ERROR  |
|              | Error                                | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_EXEC_FATAL_STATE - Precondition is incorrect. |

## 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 equalizer can continue for several invocations of the DO_EXECUTE command after the last input data has been passed to the equalizer, so the application should not assume that the equalizer has finished generating all its output until so indicated by this command. |  |
| Parameter    | 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 | Normal   | XA_NO_ERROR  |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - d is not aligned to 4 bytes.<br>XA_EQZ_EXEC_FATAL_STATE - Precondition is incorrect. |

Example

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

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

|              |   |   |
|--------------|---|---|
| Subcommand   | None  |   |
| Description  | This command obtains the number of bytes output by the equalizer during the last execution. |   |
| Parameter    | p_xa_module_obj   | Pointer to API Structure.   |
|              | i_cmd   | XA_API_CMD_GET_OUTPUT_BYTES   |
|              | i_idx   | NULL  |
|              | pv_value  | Pointer to output bytes variable.   |
| Return value | Normal  | XA_NO_ERROR   |
|              | Error   | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value is NULL.<br>XA_API_FATAL_MEM_ALIGN - The pointer is incorrectly aligned to the requirements.<br>XA_EQZ_EXEC_FATAL_STATE - Precondition is incorrect. |

## Example

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

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

|              |  |   |
|--------------|--|---|
| Subcommand   | XA_API_CMD_GET_CURIDX_INPUT_BUF  |   |
| Description  | This command gets the number of input buffer bytes consumed by the equalizer. It is used both in the initialization loop and execute loop. |   |
| Parameter    | p_xa_module_obj  | Pointer to API Structure.   |
|              | i_cmd  | XA_API_CMD_GET_CURIDX_INPUT_BUF   |
|              | i_idx  | NULL  |
|              | pv_value   | Pointer to bytes consumed variable.   |
| Return value | Normal   | XA_NO_ERROR   |
|              | Error  | XA_API_FATAL_MEM_ALLOC - p_xa_module_obj or pv_value or d->pMem_tabs->pInput is NULL.<br>XA_API_FATAL_MEM_ALIGN - The pointer is incorrectly aligned to the requirements. |

Example

WORD32 consumed;

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

## 2.5. Memory Specifications

This section describes the memory areas used by this software.

### 2.5.1. Persistent Area

|                  |  |
|------------------|--|
| Item             | Area which always holds values when this software is used.<br>If the user manipulates this area after initialization, the correct execution of this software is not ensured. |
| Symbol name      | - (freely defined by the user)   |
| Size             | Obtain the actually required size with 2.4.8.  |
| Area reservation | The user should reserve this area.   |
| Allocation       | This area is included in RAM.  |
| Alignment        | Align this area on a 4-byte boundary.  |



### 2.5.2. Input Buffer

Table 2-10 Input Buffer Description

|                  |   |
|------------------|---|
| Item             | Area which stores inputs to this software.<br>The input buffer contains 16/24 bit linear PCM data.<br>If the user manipulates this area during equalize processing, the normal execution of the program cannot be ensured.<br>[Note] This software does not support an input buffer which is a circular buffer. |
| Symbol name      | - (freely defined by the user)  |
| Size             | Please secure more than size with 2.4.8.  |
| Area reservation | The user should reserve this area.<br>The user can freely use this area after the equalizing of one block.  |
| Allocation       | This area is included in RAM.   |
| Alignment        | Align this area on a 4-byte boundary.   |

### 2.5.3. Output Buffer

Table 2-11 Output Buffer Description

|                  |  |
|------------------|--|
| Item             | Area which stores outputs from this software.<br>The output buffer contains 16/24-bit linear PCM data (hereinafter called PCM data).<br>If the user manipulates this area during equalize processing, the normal execution of the program cannot be ensured. |
| Symbol name      | - (freely defined by the user)   |
| Size             | Size same as input buffer  |
| Area reservation | The user should reserve this area.<br>The user can freely use this area after the equalizing of one block.   |
| Allocation       | This area is included in RAM.  |
| Alignment        | Align this area on a 4-byte boundary.  |

(1) Input and output data storage method

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

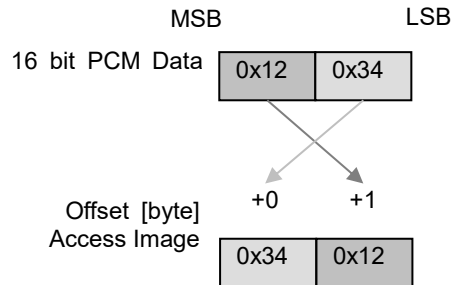


Figure 2-2 PCM 16-bit Data Access (Little Endian Mode)

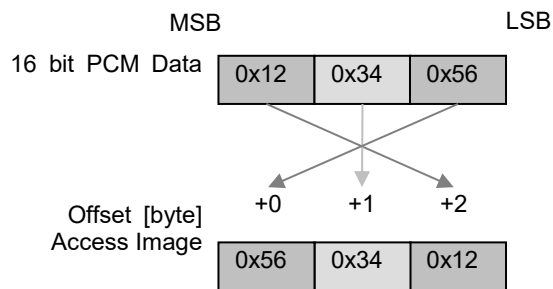
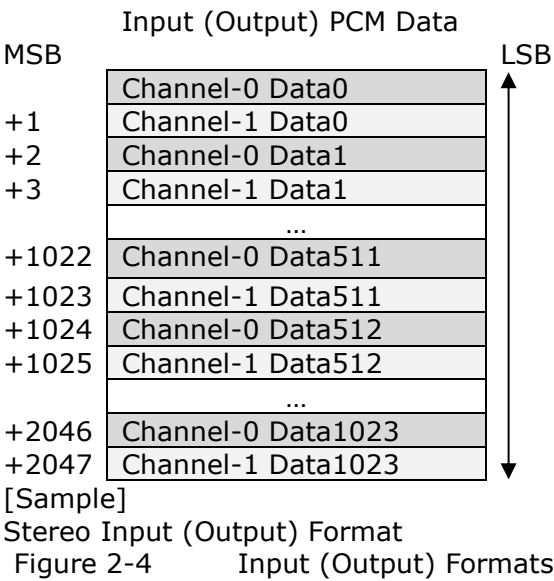


Figure 2-3 PCM 24-bit Data Access (Little Endian Mode)



## 2.6. Structures specification

### 2.6.1. API Structure

Name of structure type: XARElEqz

Structure Members:

| Type                              | Name         | Description                      |
|-----------------------------------|--------------|----------------------------------|
| WORD32                            | iState       | State of API                     |
| pVOID                             | pMem_tabs    | Memory table controller          |
| pVOID                             | iChannels    | Number of input channels         |
| WORD32                            | iPcm_width   | Width of PCM data                |
| WORD32                            | iEqz_type    | Number of input channels         |
| WORD32                            | iFs          | Type of Equalizer                |
| WORD32                            | iSample_size | Sample size                      |
| WORD32                            | iSubmitted   | Total submitted samples          |
| WORD32                            | iConsumed    | Current consumed samples         |
| WORD32                            | iProduced    | Produced output samples          |
| releqz_setParametricEqualizerCoef | stEqCoef     | Parametric equalizer coefficient |
| releqz_setGraphicEqualizerCoef    | stEqGCoef    | Graphic equalizer coefficient    |

### 2.6.2. Persistent Structure

Name of structure type: releqz\_Persistent

Function: It stores equalizer calculation data buffer.

Structure Members:

| Name            | Description                       |
|-----------------|-----------------------------------|
| Equalizer *stEq | Equalizer calculation data buffer |

### 2.6.3. Equalizer settings structure

Structure name: Equalizer

Function: It stores the parameters necessary for Equalizer object.

Structure Members:

|        | Name                         | Description                            |
|--------|------------------------------|--|
| Biquad | biquad[EQ_FILTER_N]          | Biquad filter                          |
| WORD32 | coef[EQ_FILTER_N][BQ_COEF_N] | Coefficients for each Biquad           |
| WORD32 | blockSize                    | Block size of one equalizer processing |
| WORD32 | sampleStep                   | Sample step                            |

#### 2.6.4. Biquad filter structure

Structure name: Biquad

Function: It stores the parameters necessary for Biquad filter object.

Structure Members:

| Name   |          | Description                           |
|--------|----------|---------------------------------------|
| WORD32 | delay[2] | Store data delay time.                |
| WORD32 | *coef    | Pointer to biquad filter coefficient. |

## 2.7. Error processing

| Status code | Error code (32bit)            | Value      | Description   |
|-------------|-------------------------------|------------|---|
| Normal      | XA_NO_ERROR                   | 0x00000000 | The processing results are normal.<br>The process has terminated normally.  |
| Error       | XA_API_FATAL_MEM_ALLOC        | 0xFFFF8000 | Abnormality has occurred, which disables process continuation. An address of API structure was specified at the argument is NULL, the program execution is incorrect.<br>Because it becomes the common API error, please check the correct procedure. |
| Error       | 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.                          |
| Error       | 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.   |
| Error       | 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.  |
| Error       | XA_EQZ_CONFIG_FATAL_ERR_CH    | 0xFFFF0800 | It is an error for equalizer specifications out of the range.<br>The number of the channels was specified at the argument does not support. Please set a right value.   |
| Error       | XA_EQZ_CONFIG_PARAM_ERR_FS    | 0xFFFF0801 | It is an error for equalizer  |



|       |                              |            |   |
|-------|------------------------------|------------|---|
|       |                              |            | <p>specifications out of the range.</p> <p>The number of the sampling frequency was specified at the argument does not support. Please set a right value.</p>   |
| Error | XA_EQZ_CONFIG_PARAM_ERR_SIZE | 0xFFFF0802 | <p>It is an error for equalizer specifications out of the range.</p> <p>The Input or output buffer size was specified at the argument does not support. Please set an appropriate value. Refer 2.5.2 and 2.5.3.</p> |
| Error | XA_EQZ_CONFIG_PARAM_ERR_FC   | 0x00000803 | <p>It is an error for equalizer specifications out of the range.</p> <p>The number of the center/transition frequency was specified at the argument does not support. Please set a right value.</p>                 |
| Error | XA_EQZ_CONFIG_PARAM_ERR_GA   | 0x00000804 | <p>It is an error for equalizer specifications out of the range.</p> <p>The number of the filter gain was specified at the argument does not support. Please set a right value.</p>                                 |
| Error | XA_EQZ_CONFIG_PARAM_ERR_BA   | 0x00000805 | <p>It is an error for equalizer specifications out of the range.</p> <p>The number of the filter base gain was specified at the argument does not support. Please set a right value.</p>                            |
| Error | XA_EQZ_CONFIG_PARAM_ERR_BW   | 0x00000806 | <p>It is an error for equalizer specifications out of the range.</p> <p>The number of the filter bandwidth was specified at the argument does not support. Please set a right value.</p>                            |
| Error | XA_EQZ_CONFIG_PARAM_ERR_TYPE | 0x00000807 | <p>It is an error for equalizer specifications out of the range.</p> <p>The number of the filter type was specified at the argument does not support. Please set a right value.</p>                                 |

|       |   |            |   |
|-------|---|------------|---|
| Error | XA_EQZ_CONFIG_PARAM_ERR_PCM_WIDTH       | 0xFFFF0808 | It is an error when setting PCM width of input data. This value was specified at the argument does not support. Please set a right value. |
| Error | XA_EQZ_CONFIG_PARAM_ERR_OVERWRITE       | 0x00000809 | When setting Parametric and Graphic Equalizer currently. The result will be overwritten.  |
| Error | XA_EQZ_CONFIG_PARAM_ERR_SELECT_EQZ_TYPE | 0x00000810 | It is an error when select equalizer type. This value was specified at the argument does not support. Please set a right value.           |
| Error | XA_EQZ_CONFIG_FATAL_STATE               | 0x00000811 | It is an error when precondition does not satisfy.  |

### 3. Processing Flow

Figure 3.1 shows a flow diagram of processing performed by an application which uses this software. These steps are grouped into 6 stages: Startup API, Parameters Setting, Memory Allocation, Initialization, Parameter Getting and Equalizer Executing.

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

[Note] **Set the equalizer parameter** phase performs for only Parametric Equalizer or Graphic Equalizer. If we set them currently it will return error code: (XA\_EQZ\_CONFIG\_NONFATAL\_ERR\_OVERWRITE).

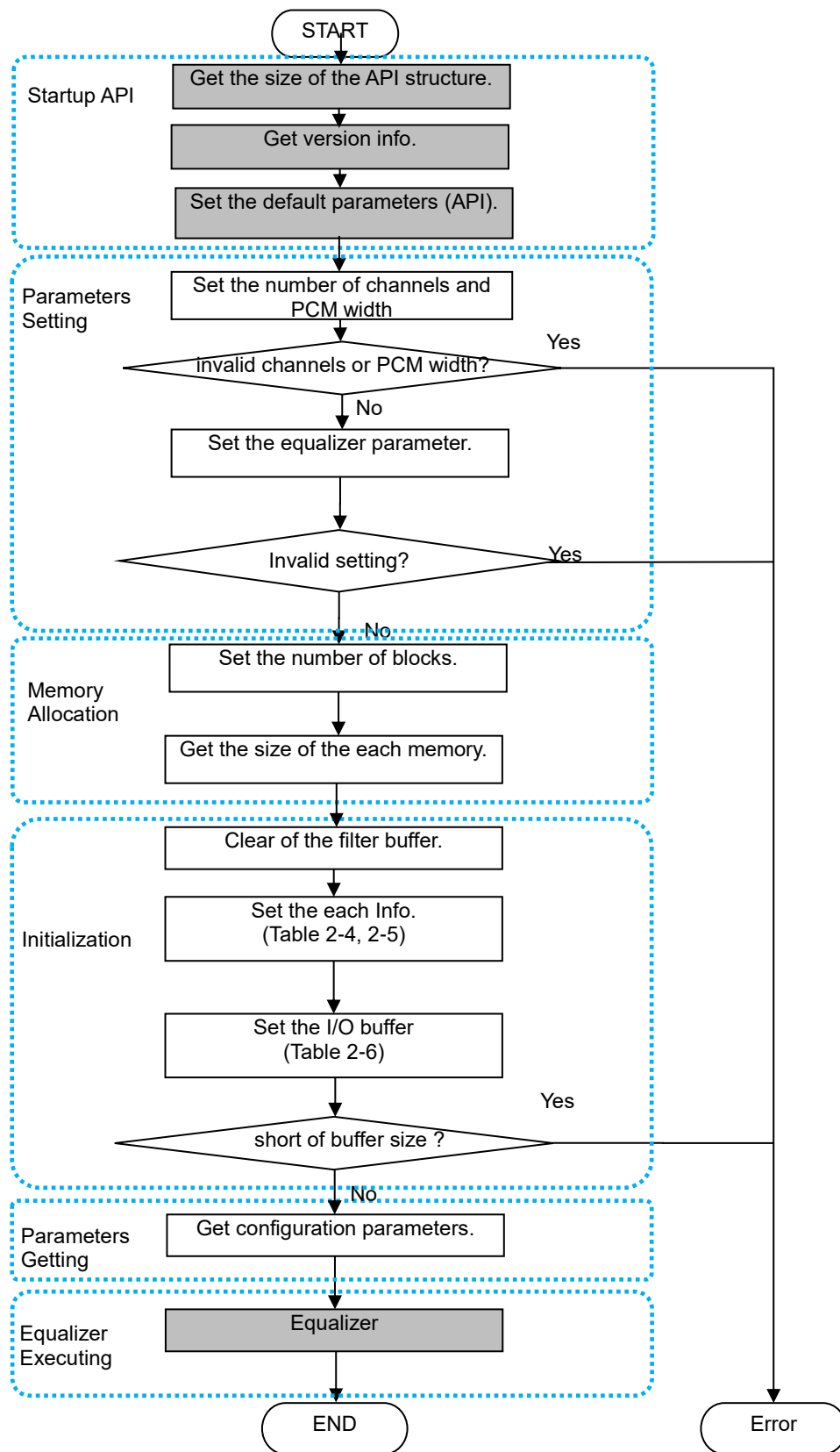


Figure 3-1 Application Processing Flow

## 4. Notes

This section describes the notice of developing user programs.

### 4.1. Function Call

User programs which call the functions in this specification should obey the calling rules of compiler.

### 4.2. Other notes

#### 4.2.1. Allocation of memory

Before calling a function in this software, reserve a persistent area, an input/output buffer area, and areas for structures which should hold the arguments of functions.

#### 4.2.2. Out of range memory access

The functions in this specification never access out of allocated memory or related I/O.

#### 4.2.3. Combination with other applications

Take care not to duplicate symbol names when other applications are combined with other programs.

#### 4.2.4. Monitoring on Performance

The products embedding this Software shall observe performance of the Software periodically with Watch Dog timer or such functions in order not to damage system performance.

# CONFIDENTIAL

|                  |   |
|------------------|---|
| Revision History | ADSP Reference Equalizer Plugin User's Manual |
|------------------|---|

| Rev. | Date          | Description |                   |
|------|---------------|-------------|-------------------|
|      |               | Page        | Summary           |
| 1.00 | Jan. 29, 2017 | -           | New Create        |
| 1.01 | Jun. 28, 2018 | -           | Style Modify      |
| 2.00 | Dec. 25, 2018 | -           | Official Release  |
|      |               | 55          | Update figure 2-4 |

---

ADSP Reference Equalizer Plugin User's Manual

Publication Date: Dec. 25, 2018 Rev. 2.00

Published by: Renesas Electronics Corporation

---



## SALES OFFICES

## Renesas Electronics Corporation

<http://www.renesas.com>

Refer to "<http://www.renesas.com/>" for the latest and detailed information.

### **Renesas Electronics Corporation**

TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

### **Renesas Electronics America Inc.**

1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.

Tel: +1-408-432-8888, Fax: +1-408-434-5351

### **Renesas Electronics Canada Limited**

9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3

Tel: +1-905-237-2004

### **Renesas Electronics Europe Limited**

Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K

Tel: +44-1628-651-700

### **Renesas Electronics Europe GmbH**

Arcadiastrasse 10, 40472 Düsseldorf, Germany

Tel: +49-211-6503-0, Fax: +49-211-6503-1327

### **Renesas Electronics (China) Co., Ltd.**

Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China

Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

### **Renesas Electronics (Shanghai) Co., Ltd.**

Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China

Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

### **Renesas Electronics Hong Kong Limited**

Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong

Tel: +852-2265-6688, Fax: +852 2886-9022

### **Renesas Electronics Taiwan Co., Ltd.**

13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan

Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

### **Renesas Electronics Singapore Pte. Ltd.**

80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949

Tel: +65-6213-0200, Fax: +65-6213-0300

### **Renesas Electronics Malaysia Sdn.Bhd.**

Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia

Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

### **Renesas Electronics India Pvt. Ltd.**

No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India

Tel: +91-80-67208700, Fax: +91-80-67208777

### **Renesas Electronics Korea Co., Ltd.**

17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea

Tel: +82-2-558-3737, Fax: +82-2-558-5338





ルネサスエレクトロニクス株式会社

営業お問い合わせ窓口

<http://www.renesas.com>

営業お問い合わせ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社 〒135-0061 東京都江東区豊洲3-2-24 (豊洲フォレシア)

技術的なお問合せおよび資料のご請求は下記へどうぞ。  
総合お問い合わせ窓口：<https://www.renesas.com/contact/>

# ADSP Reference Equalizer Plugin RCG3AHPLN0201ZDO