**Document Type: Detail Design**

**Document Name:**

**ADSP EQUALIZER PLUGIN**

Renesas Design Vietnam Co. Ltd

R-Car Software Solution 2 Group

Middleware 1 Team

Renesas Electronics Corporation

Table of Contents

[1. Overview 4](#_Toc528676269)

[1.1 Configuration 4](#_Toc528676270)

[2. Function list 5](#_Toc528676271)

[2.1 API specifications 5](#_Toc528676272)

[2.2 Internal functions 6](#_Toc528676273)

[3. Detail information 7](#_Toc528676274)

[3.1 Data type, macro definition and variable 7](#_Toc528676275)

[3.2 Function definition 9](#_Toc528676276)

[3.2.1 Supported functions 9](#_Toc528676277)

[3.2.1.1 xa\_rel\_eqz\_reset\_bqbuff 9](#_Toc528676278)

[3.2.1.2 xa\_rel\_eqz\_init\_params 10](#_Toc528676279)

[3.2.1.3 xa\_rel\_eqz\_set\_coef 11](#_Toc528676280)

[3.2.1.4 xa\_rel\_eqz\_set\_gcoef 12](#_Toc528676281)

[3.2.1.5 xa\_rel\_eqz\_do\_execute 13](#_Toc528676282)

[3.2.2 Command processing 14](#_Toc528676283)

[3.2.2.1 xa\_releqz\_get\_api\_lib\_id\_string 14](#_Toc528676284)

[3.2.2.2 xa\_releqz\_get\_api\_size 15](#_Toc528676285)

[3.2.2.3 xa\_releqz\_init 16](#_Toc528676286)

[3.2.2.4 xa\_releqz\_get\_memtabs\_size 21](#_Toc528676287)

[3.2.2.5 xa\_releqz\_set\_memtabs\_ptr 22](#_Toc528676288)

[3.2.2.6 xa\_releqz\_get\_n\_memtabs 24](#_Toc528676289)

[3.2.2.7 xa\_releqz\_get\_mem\_info\_size 25](#_Toc528676290)

[3.2.2.8 xa\_releqz\_get\_mem\_info\_alignment 27](#_Toc528676291)

[3.2.2.9 xa\_releqz\_get\_mem\_info\_type 29](#_Toc528676292)

[3.2.2.10 xa\_releqz\_set\_mem\_ptr 31](#_Toc528676293)

[3.2.2.11 xa\_releqz\_input\_over 33](#_Toc528676294)

[3.2.2.12 xa\_releqz\_set\_input\_bytes 34](#_Toc528676295)

[3.2.2.13 xa\_releqz\_get\_curidx\_input\_buf 35](#_Toc528676296)

[3.2.2.14 xa\_releqz\_execute 36](#_Toc528676297)

[3.2.2.15 xa\_releqz\_get\_output\_bytes 37](#_Toc528676298)

[3.2.2.16 xa\_releqz\_set\_config\_param 38](#_Toc528676299)

[3.2.2.17 xa\_releqz\_get\_config\_param 45](#_Toc528676300)

[4. Revision history 47](#_Toc528676301)

List of Figures

[Figure 1‑1 The software architecture 4](#_Toc528676302)

[Figure 2‑1 xa\_rel\_eqz flowchart 5](#_Toc528676303)

[Figure 3‑1 xa\_rel\_eqz\_reset\_bqbuff flowchart 9](#_Toc528676304)

[Figure 3‑2 xa\_rel\_eqz\_init\_params flowchart 10](#_Toc528676305)

[Figure 3‑3 xa\_rel\_eqz\_set\_coef flowchart 11](#_Toc528676306)

[Figure 3‑4 xa\_rel\_eqz\_set\_gcoef flowchart 12](#_Toc528676307)

[Figure 3‑5 xa\_rel\_eqz\_do\_execute flowchart 13](#_Toc528676308)

[Figure 3‑6 xa\_releqz\_get\_api\_lib\_id\_string flowchart 14](#_Toc528676309)

[Figure 3‑7 xa\_releqz\_get\_api\_size flowchart 15](#_Toc528676310)

[Figure 3‑8 xa\_releqz\_init flowchart 17](#_Toc528676311)

[Figure 3‑9 xa\_releqz\_init (A) flowchart 18](#_Toc528676312)

[Figure 3‑10 xa\_releqz\_init (B) flowchart 18](#_Toc528676313)

[Figure 3‑11 xa\_releqz\_init (C) flowchart 19](#_Toc528676314)

[Figure 3‑12 xa\_releqz\_init (D) flowchart 20](#_Toc528676315)

[Figure 3‑13 xa\_releqz\_get\_memtabs\_size flowchart 21](#_Toc528676316)

[Figure 3‑14 xa\_releqz\_set\_memtabs\_ptr flowchart 23](#_Toc528676317)

[Figure 3‑15 xa\_releqz\_get\_n\_memtabs flowchart 24](#_Toc528676318)

[Figure 3‑16 xa\_releqz\_get\_mem\_info\_size flowchart 26](#_Toc528676319)

[Figure 3‑17 xa\_releqz\_get\_mem\_info\_alignment flowchart 28](#_Toc528676320)

[Figure 3‑18 xa\_releqz\_get\_mem\_info\_type flowchart 30](#_Toc528676321)

[Figure 3‑19 xa\_releqz\_set\_mem\_ptr flowchart 32](#_Toc528676322)

[Figure 3‑23 xa\_releqz\_input\_over flowchart 33](#_Toc528676323)

[Figure 3‑24 xa\_releqz\_set\_input\_bytes flowchart 34](#_Toc528676324)

[Figure 3‑25 xa\_releqz\_get\_curidx\_input\_buf flowchart 35](#_Toc528676325)

[Figure 3‑26 xa\_releqz\_execute flowchart 36](#_Toc528676326)

[Figure 3‑27 xa\_releqz\_get\_output\_bytes flowchart 37](#_Toc528676327)

[Figure 3‑28 xa\_releqz\_set\_config\_param flowchart 39](#_Toc528676328)

[Figure 3‑29 xa\_releqz\_set\_config\_param (A) flowchart 40](#_Toc528676329)

[Figure 3‑30 xa\_releqz\_set\_config\_param (B) flowchart 40](#_Toc528676330)

[Figure 3‑31 xa\_releqz\_set\_config\_param (C) flowchart 41](#_Toc528676331)

[Figure 3‑32 xa\_releqz\_set\_config\_param (D) flowchart 41](#_Toc528676332)

[Figure 3‑33 xa\_releqz\_set\_config\_param (E) flowchart 42](#_Toc528676333)

[Figure 3‑34 xa\_releqz\_set\_config\_param (F) flowchart 42](#_Toc528676334)

[Figure 3‑35 xa\_releqz\_set\_config\_param (G) flowchart 43](#_Toc528676335)

[Figure 3‑36 xa\_releqz\_set\_config\_param (H) flowchart 43](#_Toc528676336)

[Figure 3‑37 xa\_releqz\_set\_config\_param (I) flowchart 44](#_Toc528676337)

[Figure 3‑38 xa\_releqz\_set\_config\_param (J) flowchart 44](#_Toc528676338)

[Figure 3‑42 xa\_releqz\_get\_config\_param flowchart 46](#_Toc528676339)

List of Table

[Table 2‑1 API Functions of Equalizer 6](#_Toc528676340)

[Table 2‑2 Function list 7](#_Toc528676341)

[Table 3‑1 releqz\_Memtabs type structure information 8](#_Toc528676342)

[Table 3‑2 Macro definitions 8](#_Toc528676343)

[Table 3‑3 Global variable xa\_releqz\_api 9](#_Toc528676344)

# Overview

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

## Configuration

User Application

ADSP Interface

Kernel Space

User Space

ADSP Driver

ARM

Audio HW

**ADSP Framework**

TDM class

Equalizer class

Capture class

Renderer class

ADSP

Plugin

Equalizer Plugin\*

TDM Plugin

Capture Plugin

Renderer Plugin

SRC/ DMA/ SSI/

DMA

This document’s target is in side of red square

\* Not connect to SCU/SSI DMA block

DAC/

ADC

Figure 1‑1 ****The software architecture****

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

# Function list

## API specifications

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 plugin. |
| Syntax | XA\_ERRORCODE xa\_rel\_eqz(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).  i\_idx: Command subtype or index. (defined in the supplied header files).  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). |

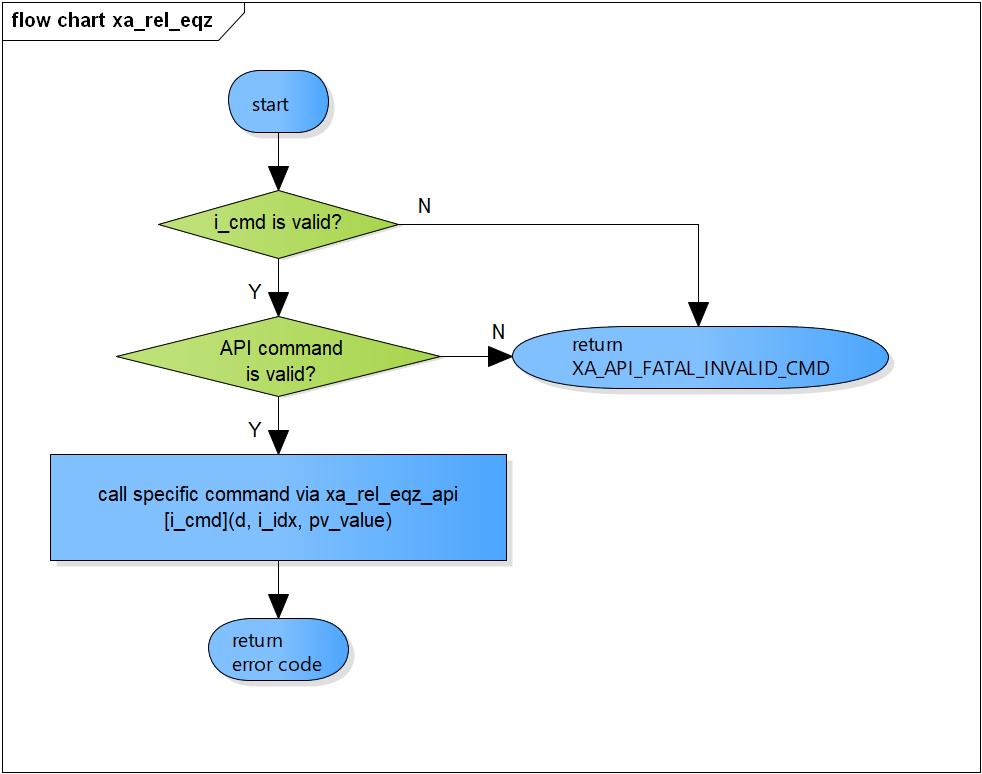


Figure 2‑1 xa\_rel\_eqz flowchart

## Internal functions

The following is list of functions:

Table 2‑2 Function list

|  |  |  |
| --- | --- | --- |
| **No.** | **Function Name** | **Outline** |
| 1 | xa\_rel\_eqz\_reset\_bqbuff | Initial parameters for equalizer, reset Biquad Filters. |
| 2 | xa\_rel\_eqz\_init\_params | Set default parameters for equalizer. |
| 3 | xa\_rel\_eqz\_set\_coef | Set coefficients for parametric equalizer. |
| 4 | xa\_rel\_eqz\_set\_gcoef | Set coefficients for graphic equalizer. |
| 5 | xa\_rel\_eqz\_do\_execute | Execute processing of equalizer: transfer data from input buffer to output buffer. |
| 6 | xa\_releqz\_get\_api\_lib\_id\_string | Get the version of the API, library, it is called by XA\_API\_CMD\_GET\_LIB\_ID\_STRINGS command. |
| 7 | xa\_releqz\_get\_api\_size | Get the size of the API structure; it is called by XA\_API\_CMD\_GET\_API\_SIZE command. |
| 8 | xa\_releqz\_init | Equalizer initialization, it is called by XA\_API\_CMD\_INIT command. |
| 9 | xa\_releqz\_set\_mem\_ptr | Set memory pointer, XA\_API\_CMD\_SET\_MEM\_PTR command. |
| 10 | xa\_releqz\_set\_memtabs\_ptr | Set memory tables pointer, it is called by XA\_API\_CMD\_SET\_MEMTABS\_PTR command. |
| 11 | xa\_releqz\_get\_memtabs\_size | Get the size of the memory structures to be allocated for the memory tables, it is called by XA\_API\_CMD\_GET\_MEMTABS\_SIZE command. |
| 12 | xa\_releqz\_get\_n\_memtabs | Get total amount of memory buffers, it is called by XA\_API\_CMD\_GET\_N\_MEMTABS command. |
| 13 | xa\_releqz\_get\_mem\_info\_size | Get memory buffer size, it is called by XA\_API\_CMD\_GET\_MEM\_INFO\_SIZE command. |
| 14 | xa\_releqz\_get\_mem\_info\_alignment | Get alignment information of the memory-type, it is called by XA\_API\_CMD\_GET\_MEM\_INFO\_ALIGNMENT command. |
| 15 | xa\_releqz\_get\_mem\_info\_type | Get the type of memory being referred to by the index, it is called by XA\_API\_CMD\_GET\_MEM\_INFO\_TYPE command. |
| 16 | xa\_releqz\_set\_input\_bytes | Set number of input bytes, it is called by XA\_API\_CMD\_SET\_INPUT\_BYTES command. |
| 17 | xa\_releqz\_get\_curidx\_input\_buf | Get the number of input buffer bytes consumed, it is called by XA\_API\_CMD\_GET\_CURIDX\_INPUT\_BUF command. |
| 18 | xa\_releqz\_input\_over | Stop equalizer, it is called by XA\_API\_CMD\_INPUT\_OVER command. |
| 19 | xa\_releqz\_get\_output\_bytes | Get the number of bytes output, it is called by XA\_API\_CMD\_GET\_OUTPUT\_BYTES command. |
| 20 | xa\_releqz\_set\_config\_param | Set equalizer configuration parameters, it is called by XA\_API\_CMD\_SET\_CONFIG\_PARAM command. |
| 21 | xa\_releqz\_get\_config\_param | Get equalizer configuration parameters, it is called by XA\_API\_CMD\_GET\_CONFIG\_PARAM command. |
| 22 | xa\_releqz\_execute | Execute Equalizer plugin; it is called by XA\_API\_CMD\_EXECUTE command. |

# Detail information

This section describes detail information of data types, macro definitions, implemented APIs, and internal function units, global variable.

## Data type, macro definition and variable

Refer to 2.6.1 API structure, 2.6.2 Persistent structure into function design of equalizer document [ADSP Reference Equalizer Plugin]

Table 3‑1 releqz\_Memtabs type structure information

|  |  |
| --- | --- |
| Member name | Outline |
| WORD32 iPersist\_size | Persistent buffer size. |
| pVOID pPersistent | Persistent buffer pointer. |
| WORD32 iInput\_size | Input buffer size |
| pVOID pInput | Input buffer pointer. |
| WORD32 iOutput\_size | Output buffer size. |
| pVOID pOutput | Output buffer pointer. |
| pVOID pScratch | Scratch memory use to temporary store information when transfer data from input to output buffer. |

Table 3‑2 Macro definitions

|  |  |  |
| --- | --- | --- |
| Macro | Value | Outline |
| XA\_REL\_REN\_LIB\_VERSION | "1.0.0" | Equalizer library version. |
| XA\_REL\_REN\_API\_VERSION | "1.0.0" | Equalizer api version. |
| XA\_REL\_EQZ\_FLAG\_START | 0 | Equalizer start flag. |
| XA\_REL\_EQZ\_FLAG\_PREINIT\_DONE | 1 << 0 | Pre-initialize done flag. |
| XA\_REL\_EQZ\_FLAG\_MEMTABS\_DONE | 1 << 1 | Set memory table pointer done flag. |
| XA\_REL\_EQZ\_FLAG\_POSTINIT\_DONE | 1 << 2 | Post-initialize done flag. |
| XA\_REL\_EQZ\_FLAG\_RUNNING | 1 << 3 | Equalizer running flag. |
| XA\_REL\_EQZ\_FLAG\_EOS | 1 << 4 | Input end of stream flag. |
| XA\_REL\_EQZ\_FLAG\_COMPLETE | 1 << 5 | Equalizer execute done flag. |
| XA\_REL\_EQZ\_ALIGN\_4BYTE | 4 | 4-byte alignment. |
| XA\_REL\_EQZ\_BUFFER\_SIZE | 1024 | Equalizer buffer size. |
| XA\_CHK\_ALIGN(d, n) | (((UWORD32)(d)) & ((n)-1)) | Check d is alignment with n bytes. |
| GET\_IDX(IDX) | ((IDX) & 0x0F) | Get index of sub-command. |
| XA\_REL\_EQZ\_API\_COMMANDS\_NUM | (sizeof(xa\_releqz\_api) / sizeof(xa\_releqz\_api[0])) | Total number of commands supported. |

Table 3‑3 Global variable xa\_releqz\_api

|  |  |
| --- | --- |
| static XA\_ERRORCODE (\* const xa\_releqz\_api[])(XARelEqz\*, WORD32, pVOID) | |
| Description: variable stores function pointers according to API command index to run in runtime operation. | |
| Array API command index | Value (function pointer) |
| XA\_API\_CMD\_GET\_LIB\_ID\_STRINGS | xa\_releqz\_get\_api\_lib\_id\_string |
| XA\_API\_CMD\_GET\_API\_SIZE | xa\_releqz\_get\_api\_size |
| XA\_API\_CMD\_INIT | xa\_releqz\_init |
| XA\_API\_CMD\_GET\_MEMTABS\_SIZE | xa\_releqz\_get\_memtabs\_size |
| XA\_API\_CMD\_SET\_MEMTABS\_PTR | xa\_releqz\_set\_memtabs\_ptr |
| XA\_API\_CMD\_GET\_N\_MEMTABS | xa\_releqz\_get\_n\_memtabs |
| XA\_API\_CMD\_GET\_MEM\_INFO\_SIZE | xa\_releqz\_get\_mem\_info\_size |
| XA\_API\_CMD\_GET\_MEM\_INFO\_ALIGNMENT | xa\_releqz\_get\_mem\_info\_alignment |
| XA\_API\_CMD\_GET\_MEM\_INFO\_TYPE | xa\_releqz\_get\_mem\_info\_type |
| XA\_API\_CMD\_SET\_MEM\_PTR | xa\_releqz\_set\_mem\_ptr |
| XA\_API\_CMD\_INPUT\_OVER | xa\_releqz\_input\_over |
| XA\_API\_CMD\_SET\_INPUT\_BYTES | xa\_releqz\_set\_input\_bytes |
| XA\_API\_CMD\_GET\_CURIDX\_INPUT\_BUF | xa\_releqz\_get\_curidx\_input\_buf |
| XA\_API\_CMD\_GET\_OUTPUT\_BYTES | xa\_releqz\_get\_output\_bytes |
| XA\_API\_CMD\_EXECUTE | xa\_releqz\_execute |
| XA\_API\_CMD\_SET\_CONFIG\_PARAM | xa\_releqz\_set\_config\_param |
| XA\_API\_CMD\_GET\_CONFIG\_PARAM | xa\_releqz\_get\_config\_param |

Note: All API command index macro is definded in xa\_apicmd\_standards.h.

## Function definition

### Supported functions

#### xa\_rel\_eqz\_reset\_bqbuff

DD\_PLG\_EQZ\_01\_001

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static void xa\_rel\_eqz\_reset\_bqbuff(XARelEqz \*d) | | | |
| **Function** | Initial parameters for equalizer, reset Biquad Filters. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure  (struct XAReleqz). |
| **Return value** | None. | | | |
| **Description** | * xa\_rel\_eqz\_reset\_bqbuff command processing:   - Reset biquad filters for all channels. | | | |

[Covers: FD\_PLG\_EQZ\_005]

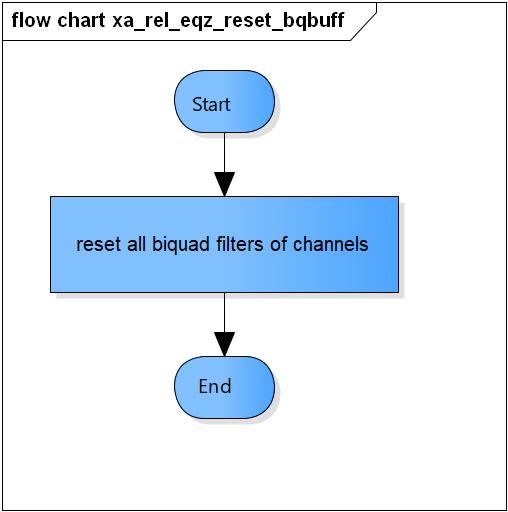


Figure 3‑1 xa\_rel\_eqz\_reset\_bqbuff flowchart

#### xa\_rel\_eqz\_init\_params

DD\_PLG\_EQZ\_01\_002

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static void xa\_rel\_eqz\_init\_params(XARelEqz \*d) | | | |
| **Function** | Set default parameters for equalizer. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure  (struct XAReleqz). |
| **Return value** | None. | | | |
| **Description** | * xa\_rel\_eqz\_init\_params command processing:   - Set default equalizer type is parametric.  - Set default gain for each graphic equalizer band.  - Set default value for buffer size control. | | | |

[Covers: FD\_PLG\_EQZ\_003]

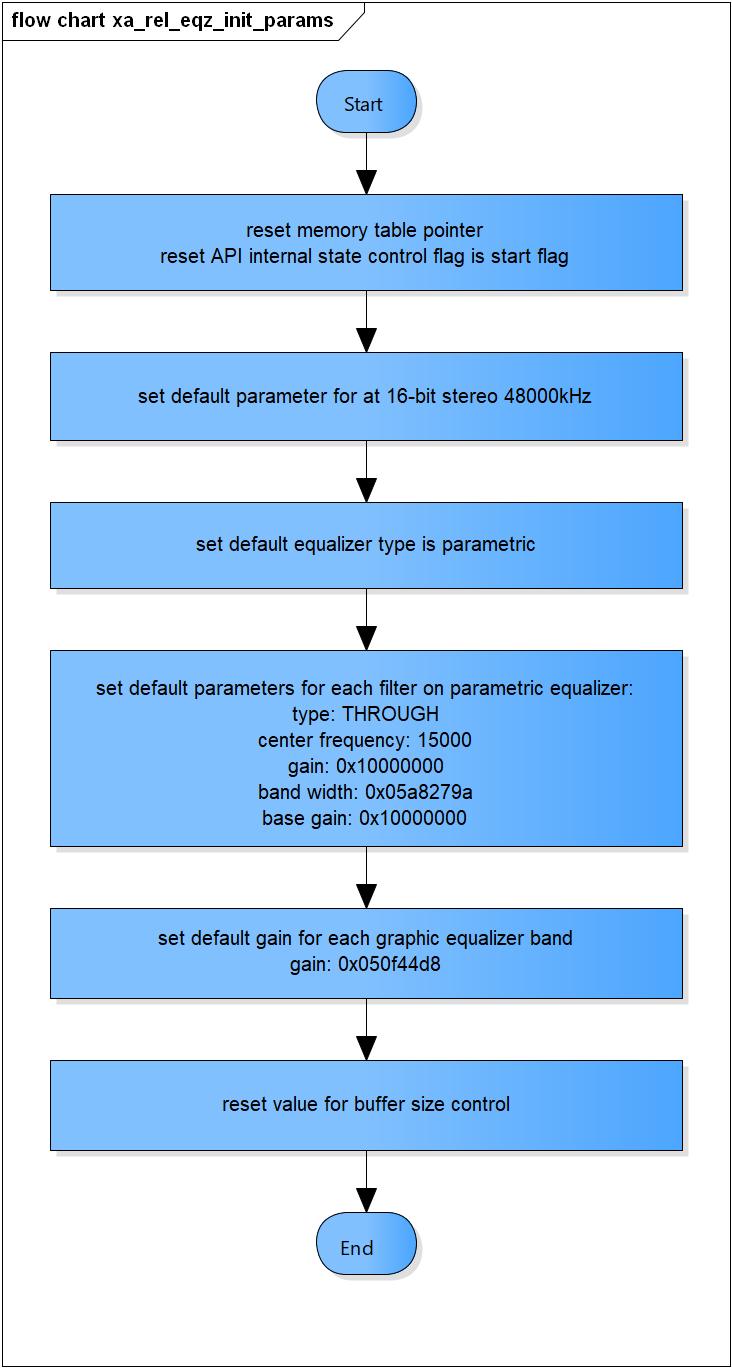


Figure 3‑2 xa\_rel\_eqz\_init\_params flowchart

#### xa\_rel\_eqz\_set\_coef

DD\_PLG\_EQZ\_01\_003

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static void xa\_rel\_eqz\_set\_coef(XARelEqz \*d) | | | |
| **Function** | Set coefficients for parametric equalizer. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure  (struct XAReleqz). |
| **Return value** | None. | | | |
| **Description** | * xa\_rel\_eqz\_set\_coef command processing:   - Set coefficients of parametric equalizer for all channels. | | | |

[Covers: FD\_PLG\_EQZ\_005]

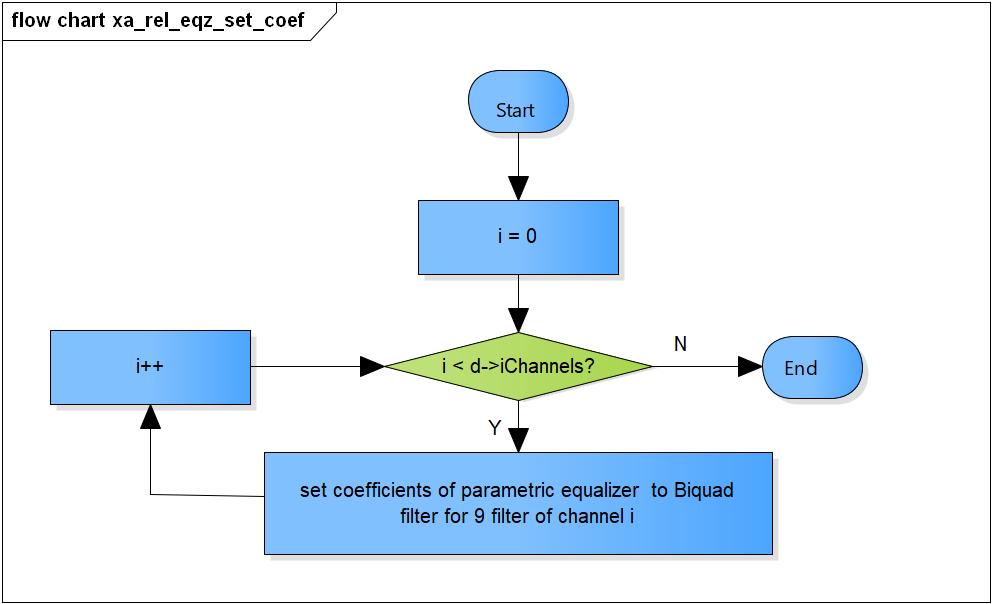


Figure 3‑3 xa\_rel\_eqz\_set\_coef flowchart

#### xa\_rel\_eqz\_set\_gcoef

DD\_PLG\_EQZ\_01\_004

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static void xa\_rel\_eqz\_set\_gcoef(XARelEqz \*d) | | | |
| **Function** | Set coefficients for graphic equalizer. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure  (struct XAReleqz). |
| **Return value** | None. | | | |
| **Description** | * xa\_rel\_eqz\_set\_gcoef command processing:   - Set coefficients of graphic equalizer for all channels. | | | |

[Covers: FD\_PLG\_EQZ\_005]

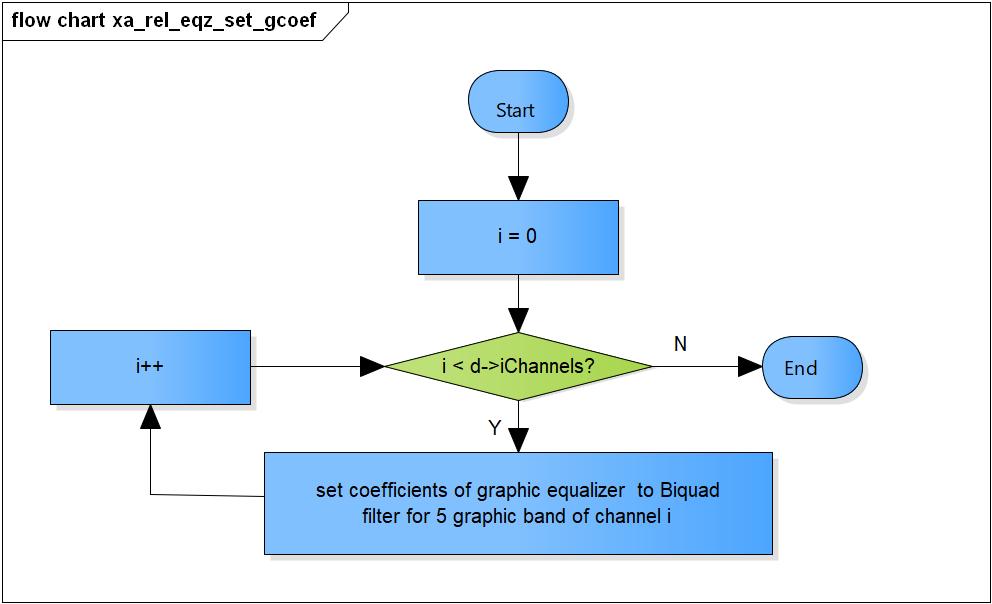


Figure 3‑4 xa\_rel\_eqz\_set\_gcoef flowchart

#### xa\_rel\_eqz\_do\_execute

DD\_PLG\_EQZ\_01\_005

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static void xa\_rel\_eqz\_do\_execute(XARelEqz \*d) | | | |
| **Function** | Transfer data from input buffer to output buffer. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure  (struct XAReleqz). |
| **Return value** | None. | | | |
| **Description** | * xa\_rel\_eqz\_do\_execute command processing:   - Reset all value in scratch memory.  - Get the number of filter need to process based on equalizer type.  - Transfer data from input memory to output memory for each channel.  - Calculate number of byte consumed, number of output byte. | | | |

[Covers: FD\_PLG\_EQZ\_036]

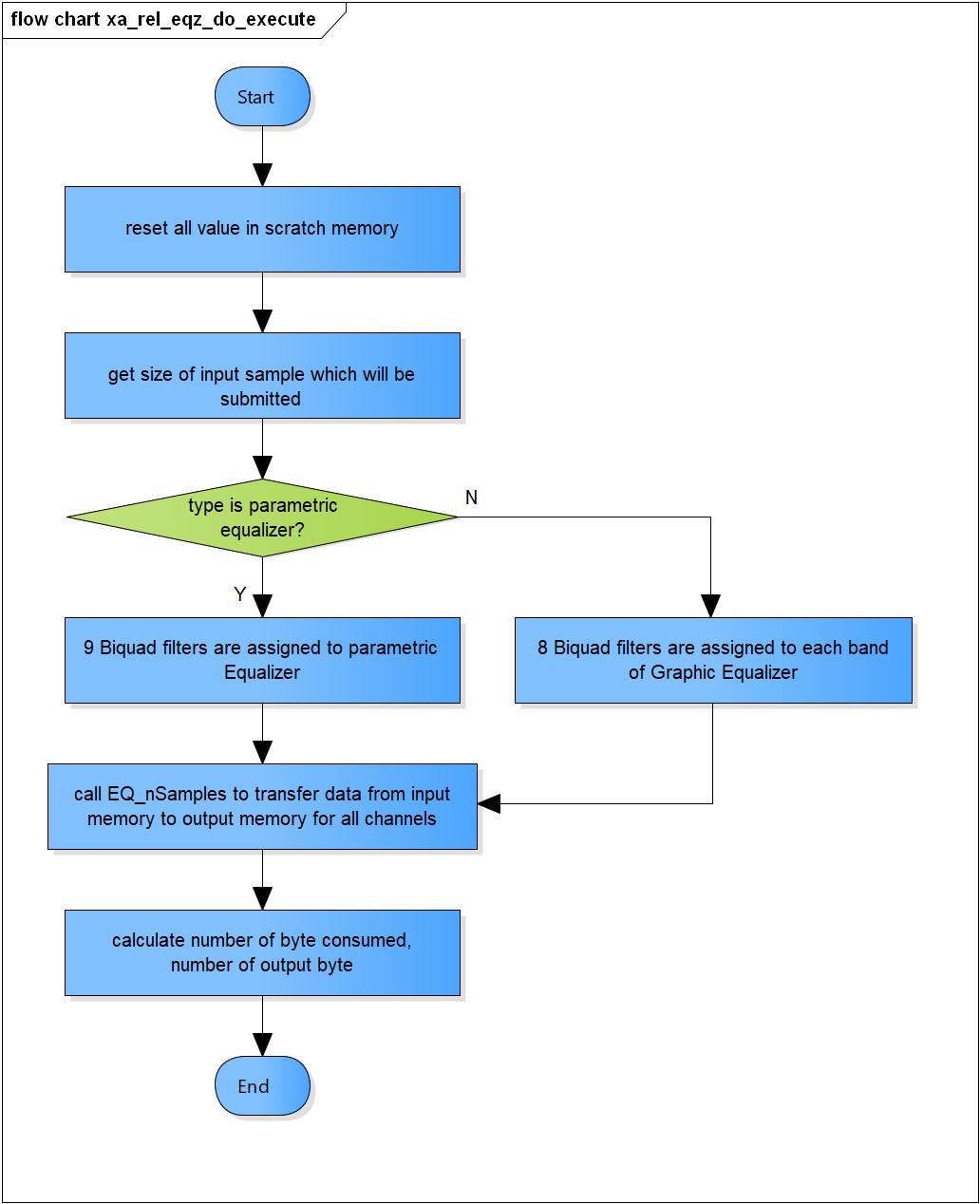


Figure 3‑5 xa\_rel\_eqz\_do\_execute flowchart

### Command processing

#### xa\_releqz\_get\_api\_lib\_id\_string

DD\_PLG\_EQZ\_01\_006

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_get\_api\_lib\_id\_string(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This function 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. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure  (struct XAReleqz). |
| WORD32 | i\_idx | I | Index of sub-command.  Valid values:  XA\_CMD\_TYPE\_LIB\_VERSION  XA\_CMD\_TYPE\_API\_VERSION |
| pVOID | pv\_value | O | Pointer to API version or library version. |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | pv\_value is NULL. | |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE | | i\_idx is invalid. | |
| **Description** | * xa\_releqz\_get\_api\_lib\_id\_string command processing:   - Check pv\_value is valid.  - Get the version of the API, library. | | | |

[Covers: FD\_PLG\_EQZ\_001]

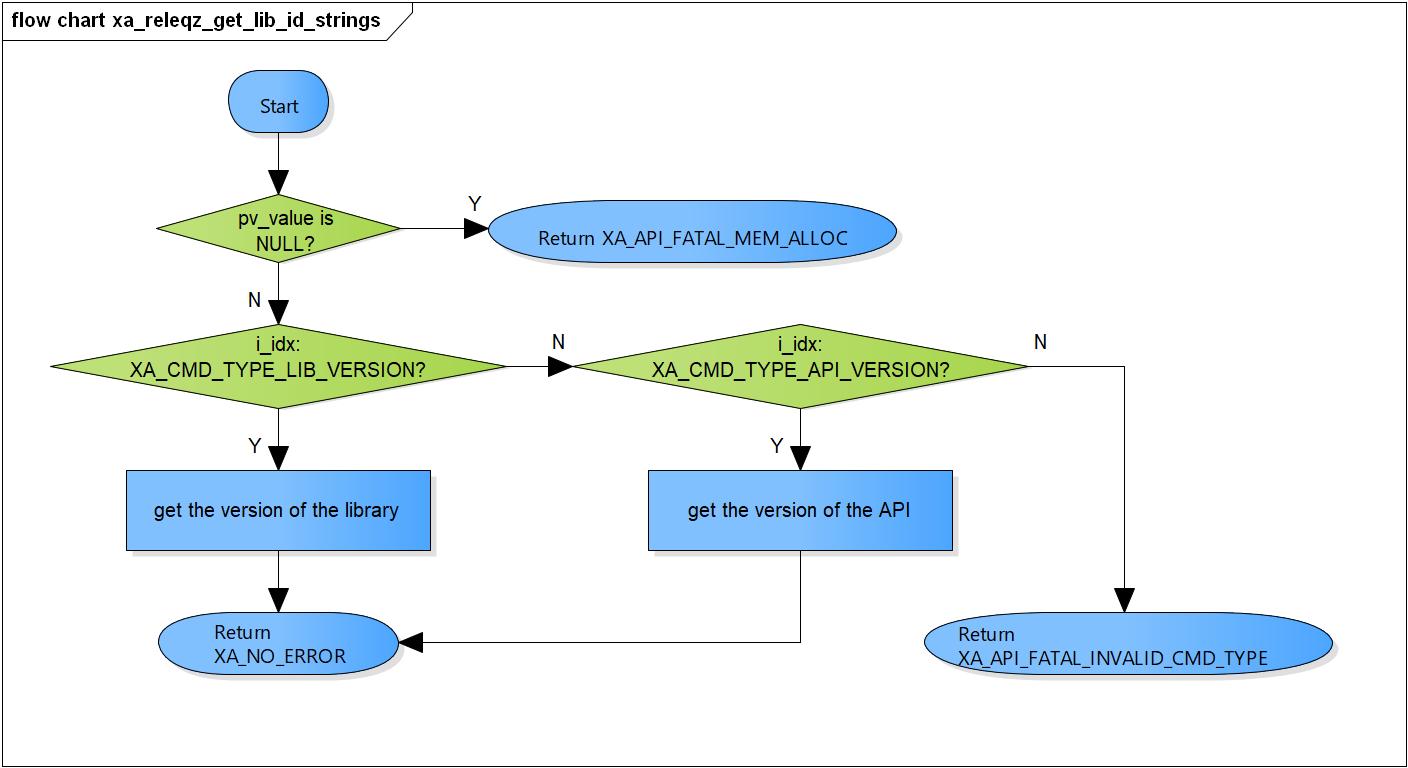


Figure 3‑6 xa\_releqz\_get\_api\_lib\_id\_string flowchart

#### xa\_releqz\_get\_api\_size

DD\_PLG\_EQZ\_01\_007

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_get\_api\_size(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This function is used to obtain the size of the API structure, in order to allocate memory for the API structure. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure  (struct XAReleqz). |
| WORD32 | i\_idx | X | Index of sub-command. |
| pVOID | pv\_value | O | Pointer to plugin API structure size variable (struct XAReleqz). |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | pv\_value is NULL. | |
| **Description** | * xa\_releqz\_get\_api\_size command processing:   - Check pv\_value is valid.  - Get API structure size. | | | |

[Covers: FD\_PLG\_EQZ\_002]

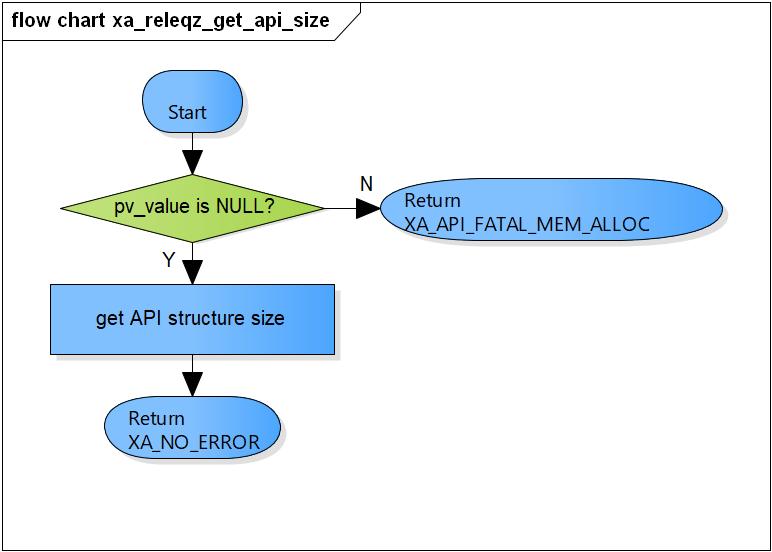


Figure 3‑7 xa\_releqz\_get\_api\_size flowchart

#### xa\_releqz\_init

DD\_PLG\_EQZ\_01\_008

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_init(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This function is used to set the default value of the configuration parameters in pre-configuration process, calculate all memory in post-configuration process, calculate coefficients of equalizer base on equalizer type in initialization process and initialization done query process. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I/O | Pointer to plugin API structure (struct XAReleqz). |
| WORD32 | i\_idx | I | Index of sub-command.  Valid values:  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 |
| pVOID | pv\_value | x | NULL. |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object is NULL. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object is not aligned to 4 bytes. | |
| XA\_EQZ\_CONFIG\_FATAL\_STATE | | Incorrect sequence call.  Memory table is invalid in post init processing. | |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE | | Invalid index. | |
| XA\_EQZ\_EXEC\_FATAL\_STATE | | Scratch, persistent, input, output memory are invalid in init process. | |
| **Description** | * xa\_releqz\_init command processing: * Check API structure is valid. * Check API structure is aligned to 4 bytes. * Process particular type (pre config params, init API post config params, init process, init done query). | | | |

[Covers: FD\_PLG\_EQZ\_003, FD\_PLG\_EQZ\_004, FD\_PLG\_EQZ\_005, FD\_PLG\_EQZ\_006]

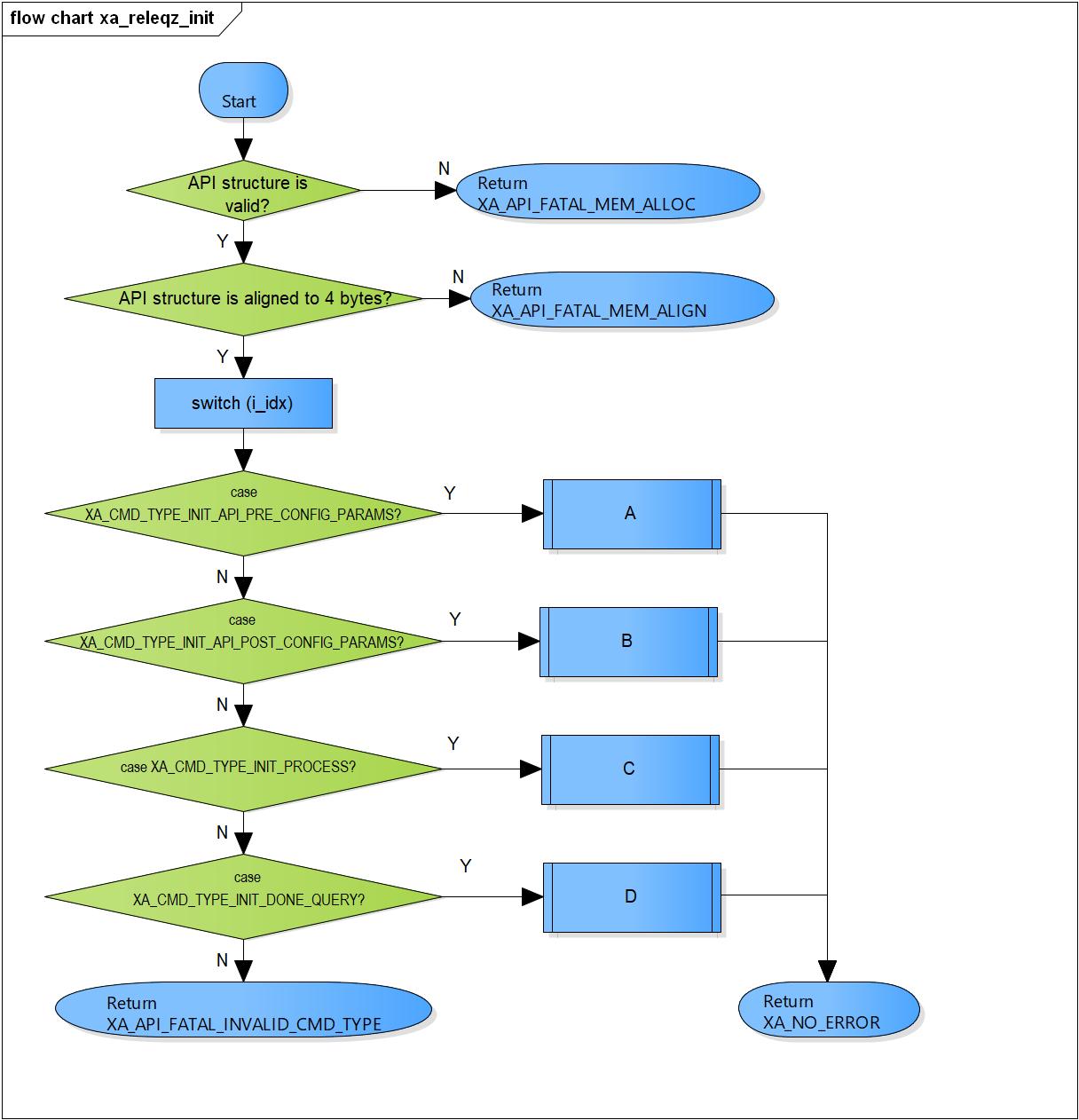


Figure 3‑8 xa\_releqz\_init flowchart

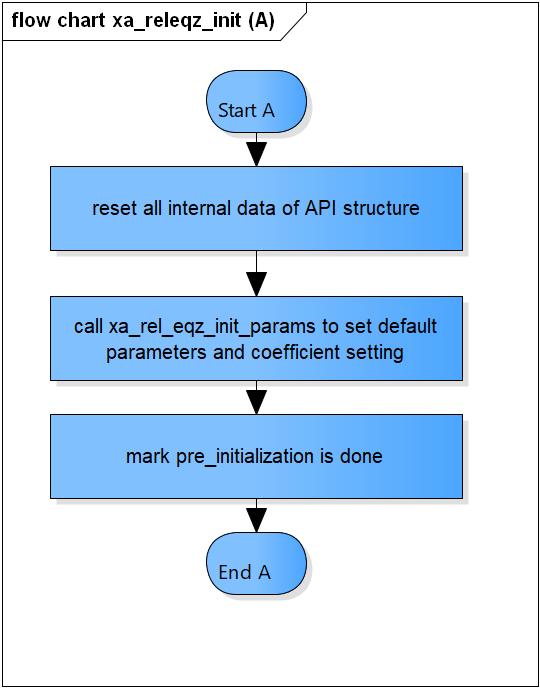


Figure 3‑9 xa\_releqz\_init (A) flowchart

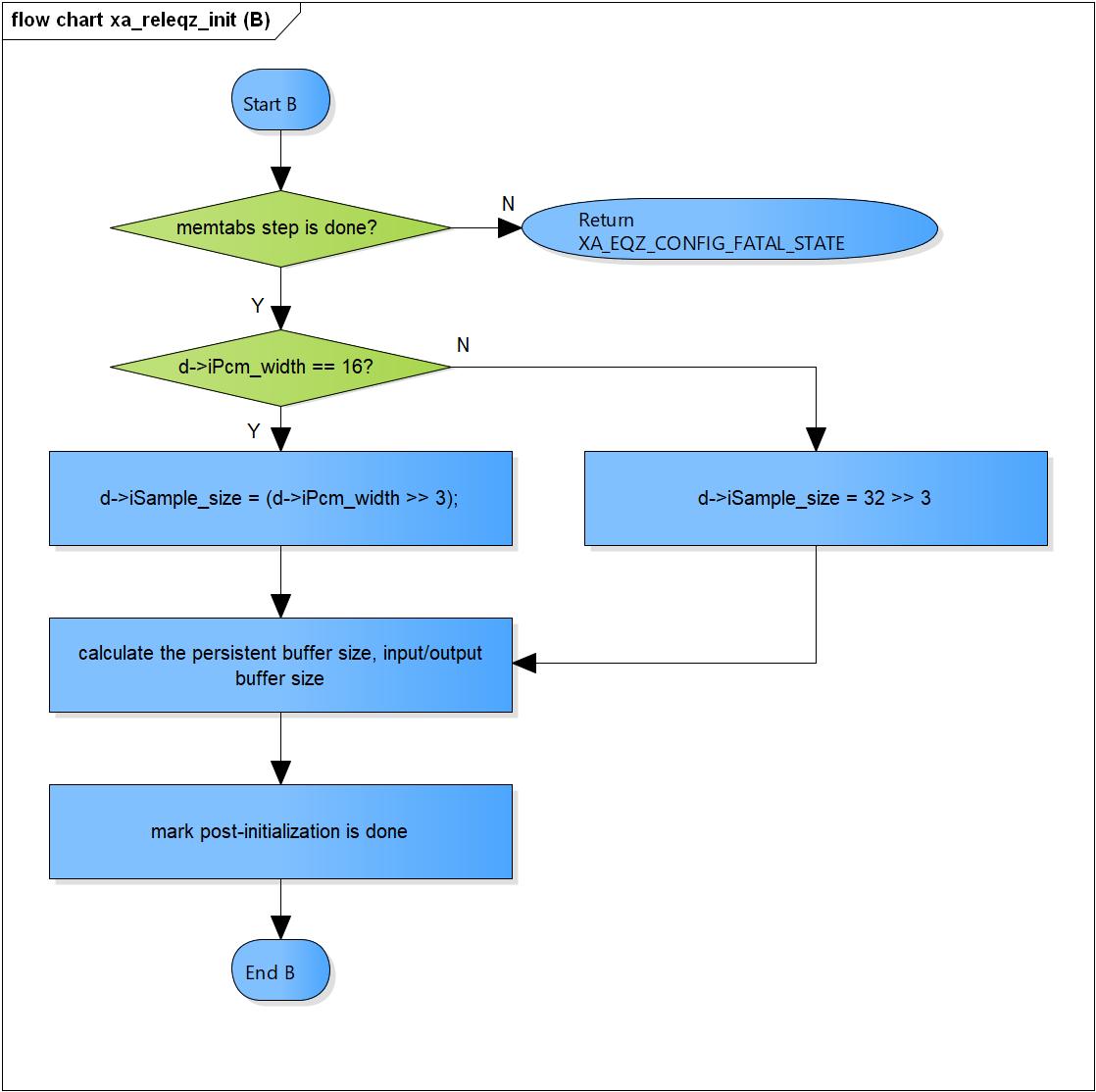


Figure 3‑10 xa\_releqz\_init (B) flowchart

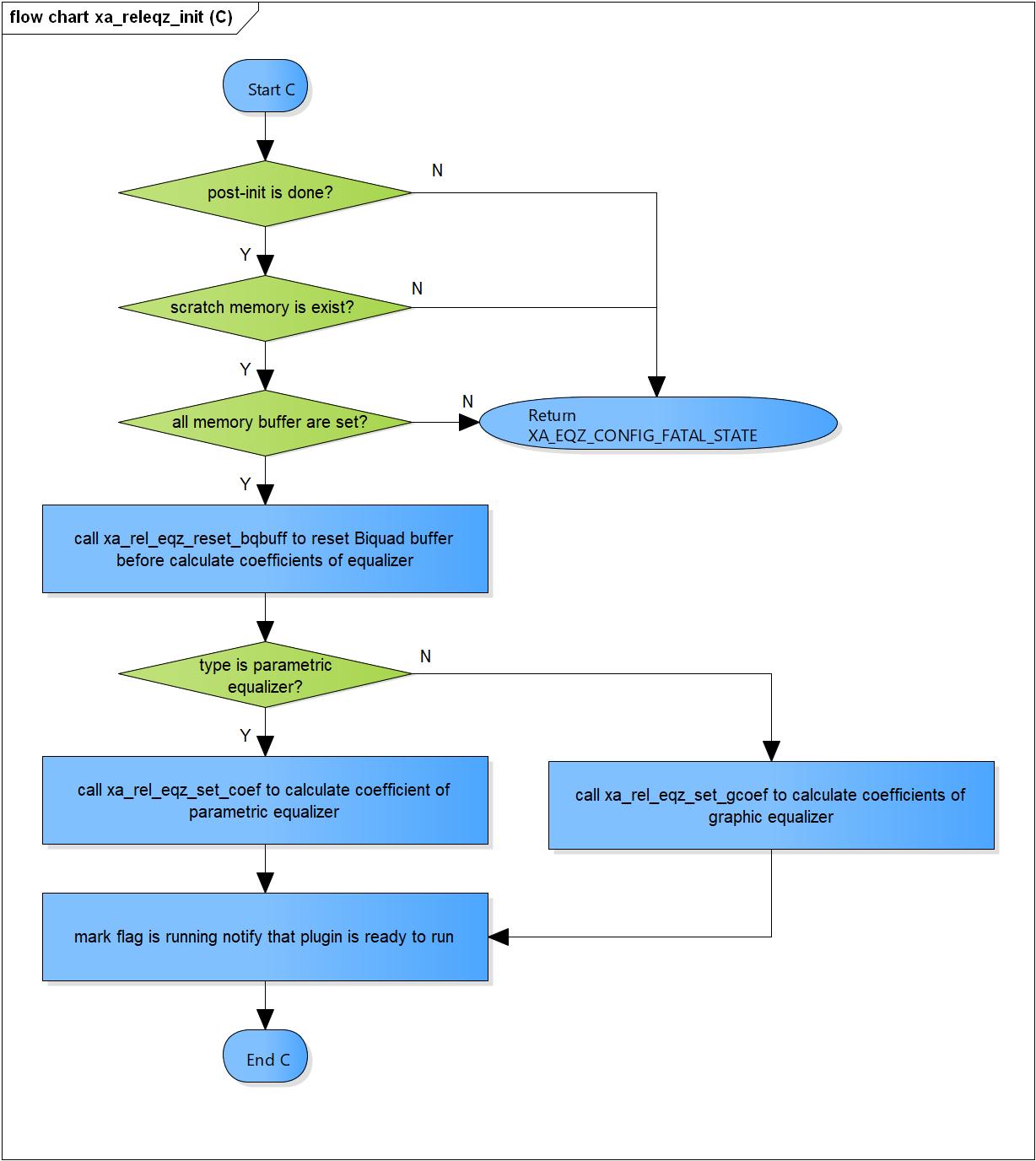


Figure 3‑11 xa\_releqz\_init (C) flowchart

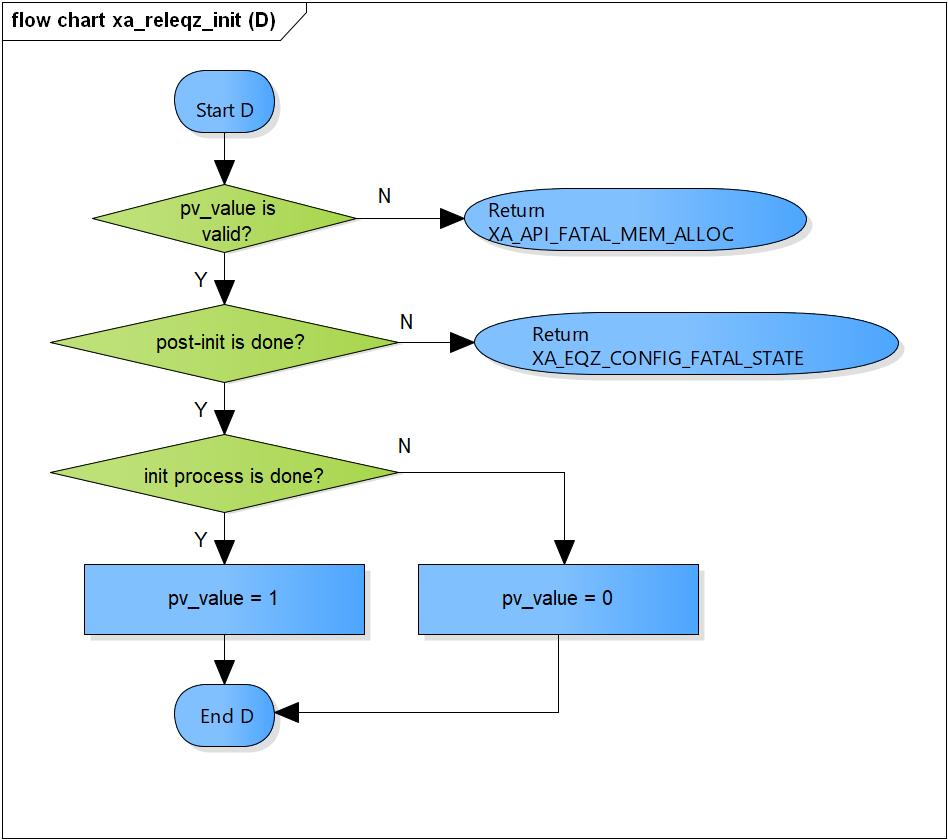


Figure 3‑12 xa\_releqz\_init (D) flowchart

#### xa\_releqz\_get\_memtabs\_size

DD\_PLG\_EQZ\_01\_009

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_get\_memtabs\_size(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This function 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. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure (struct XAReleqz). |
| WORD32 | i\_idx | X | Index of sub-command. |
| pVOID | pv\_value | O | Pointer to memory size variable. |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object is NULL.  pv\_value is NULL. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object is not aligned to 4 bytes. | |
| XA\_EQZ\_CONFIG\_FATAL\_STATE | | Equalizer has not pre-init yet. | |
| **Description** | * xa\_releqz\_get\_memtabs\_size command processing: * Check API structure is valid. * Check API structure is aligned to 4 bytes. * Check equalizer is pre-initialized. * Get size of the memory table of plugin. | | | |

[Covers: FD\_PLG\_EQZ\_017]

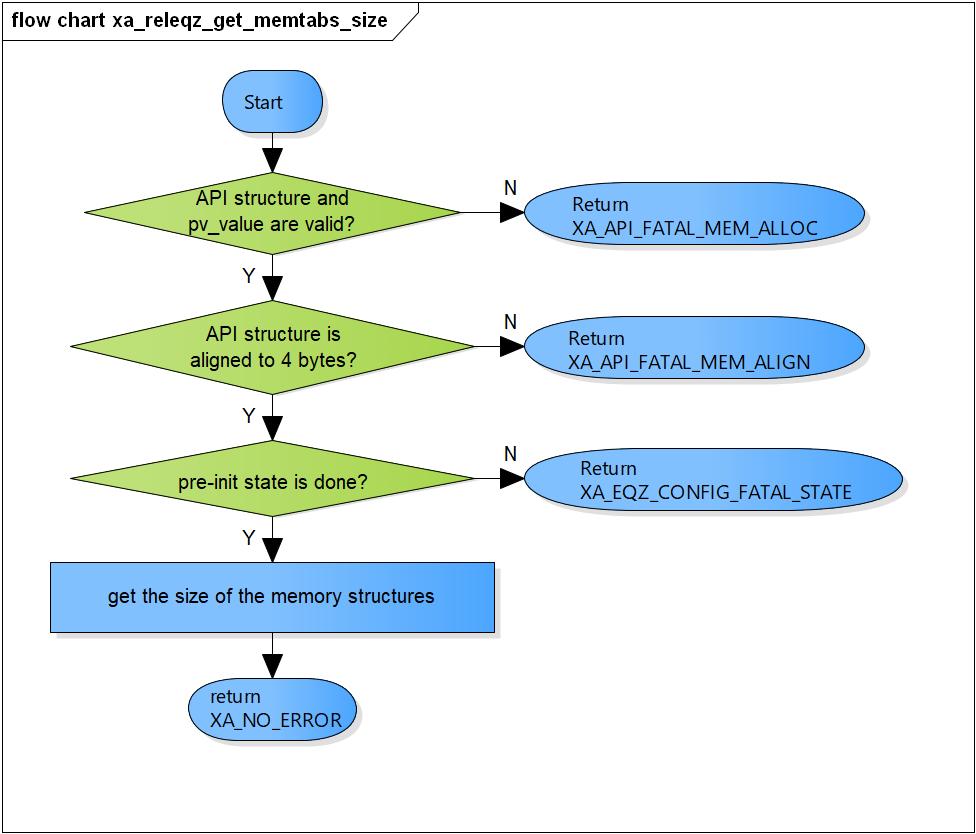


Figure 3‑13 xa\_releqz\_get\_memtabs\_size flowchart

#### xa\_releqz\_set\_memtabs\_ptr

DD\_PLG\_EQZ\_01\_010

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_set\_memtabs\_ptr(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This function is used to set the memory structure pointer to the allocated value. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure (struct XAReleqz). |
| WORD32 | i\_idx | X | Index of sub-command. |
| pVOID | pv\_value | I | Memory table pointer. |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object is NULL.  pv\_value is NULL. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object or pv\_value are not aligned to 4 bytes. | |
| XA\_EQZ\_CONFIG\_FATAL\_STATE | | Equalizer has not pre-init yet. | |
| **Description** | * xa\_releqz\_set\_memtabs\_ptr command processing: * Check API structure and pv\_value are valid. * Check API structure and pv\_value are aligned to 4 bytes. * Check equalizer is pre-initialized. * Set memory tables pointer. * Mark memtabs step is done. | | | |

[Covers: FD\_PLG\_EQZ\_018]

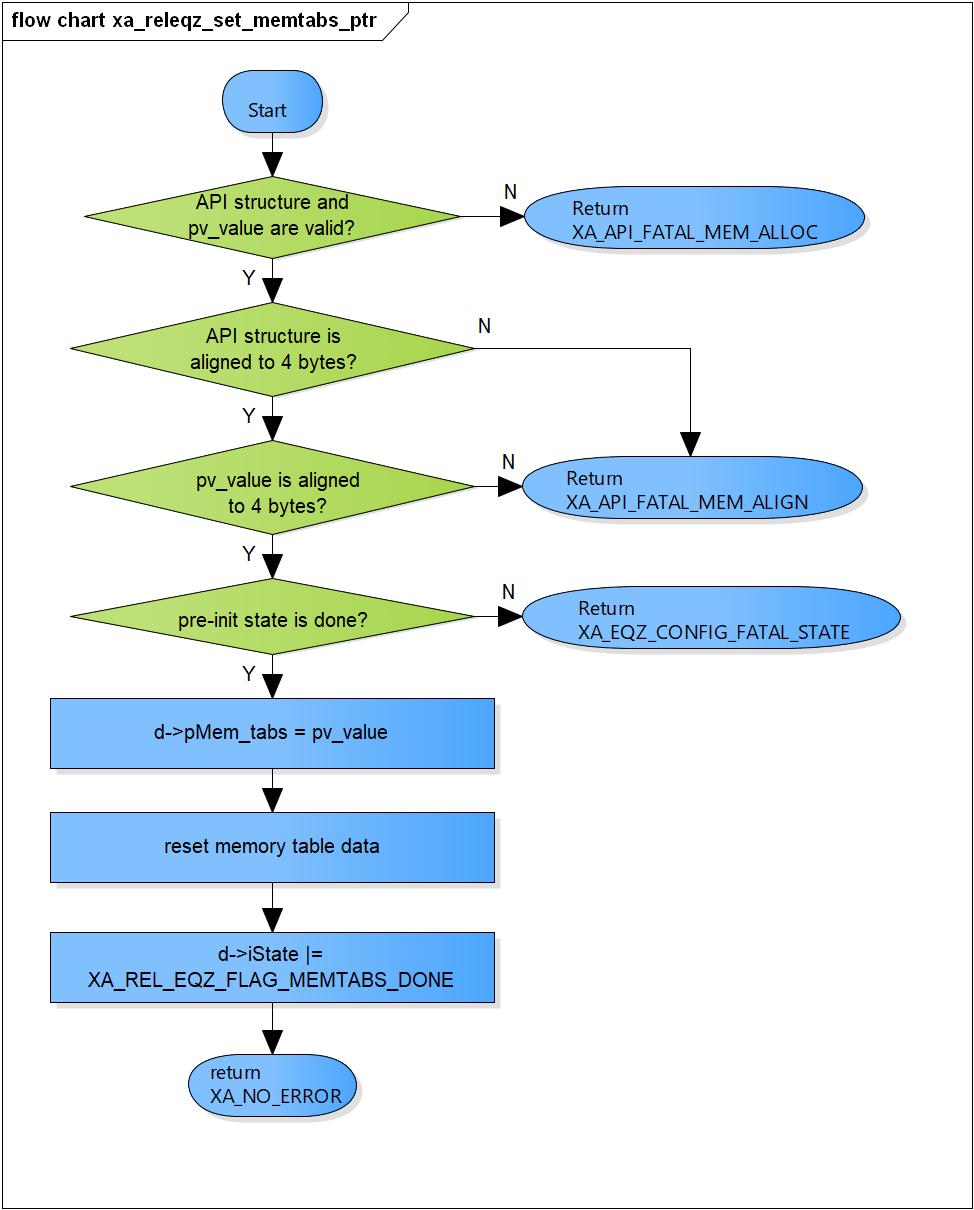


Figure 3‑14 xa\_releqz\_set\_memtabs\_ptr flowchart

#### xa\_releqz\_get\_n\_memtabs

DD\_PLG\_EQZ\_01\_011

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_get\_n\_memtabs(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This function 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. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure (struct XAReleqz). |
| WORD32 | i\_idx | X | Index of sub-command. |
| pVOID | pv\_value | O | Pointer to variable of number of memory buffers required to be allocated. |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object is NULL.  pv\_value is NULL. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object is not aligned to 4 bytes. | |
| **Description** | * xa\_releqz\_get\_n\_memtabs command processing: * Check API structure and pv\_value are valid. * Check API structure is aligned to 4 bytes. * Get number of memory in plugin. | | | |

[Covers: FD\_PLG\_EQZ\_019]

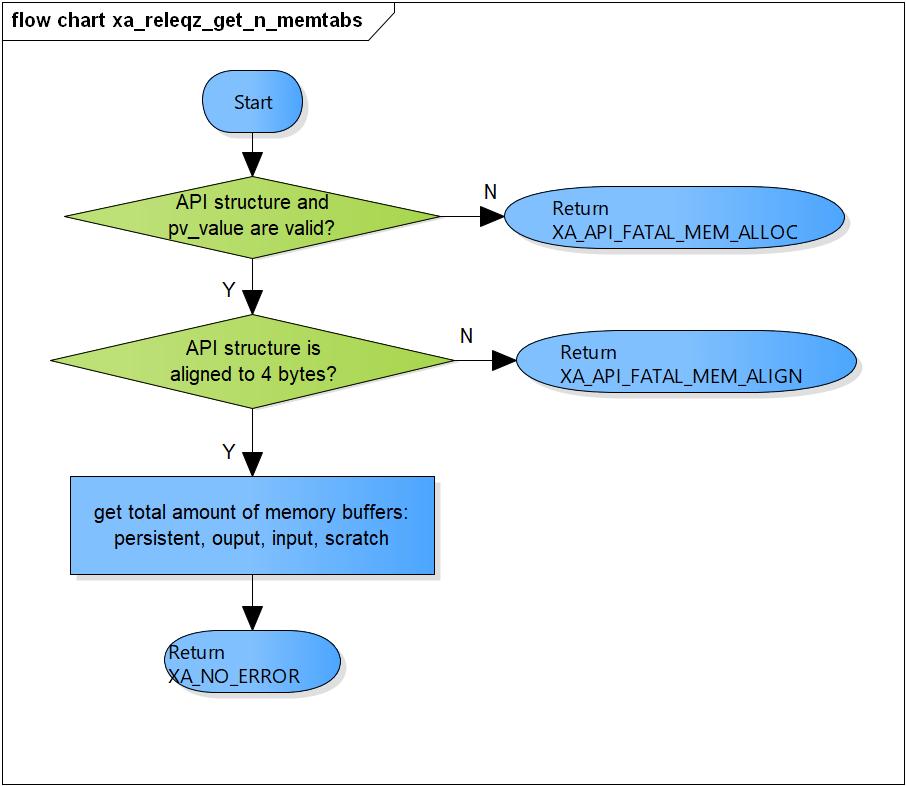


Figure 3‑15 xa\_releqz\_get\_n\_memtabs flowchart

#### xa\_releqz\_get\_mem\_info\_size

DD\_PLG\_EQZ\_01\_012

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_get\_mem\_info\_size(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This function 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** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure (struct XAReleqz). |
| WORD32 | i\_idx | I | Index of the memory  0 - Persistent Area  1 - Input Buffer  2 - Output Buffer  3 – Scratch Area |
| pVOID | pv\_value | O | Pointer to memory size. |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object is NULL.  pv\_value is NULL. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object is not aligned to 4 bytes. | |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE | | Invalid index. | |
| XA\_EQZ\_CONFIG\_FATAL\_STATE | | Equalizer has not post-init yet. | |
| **Description** | * xa\_releqz\_get\_mem\_info\_size command processing: * Check API structure is valid. * Check API structure is aligned to 4 bytes. * Check equalizer is post-initialized. * Get memory buffer size base on i\_idx. | | | |

[Covers: FD\_PLG\_EQZ\_020]

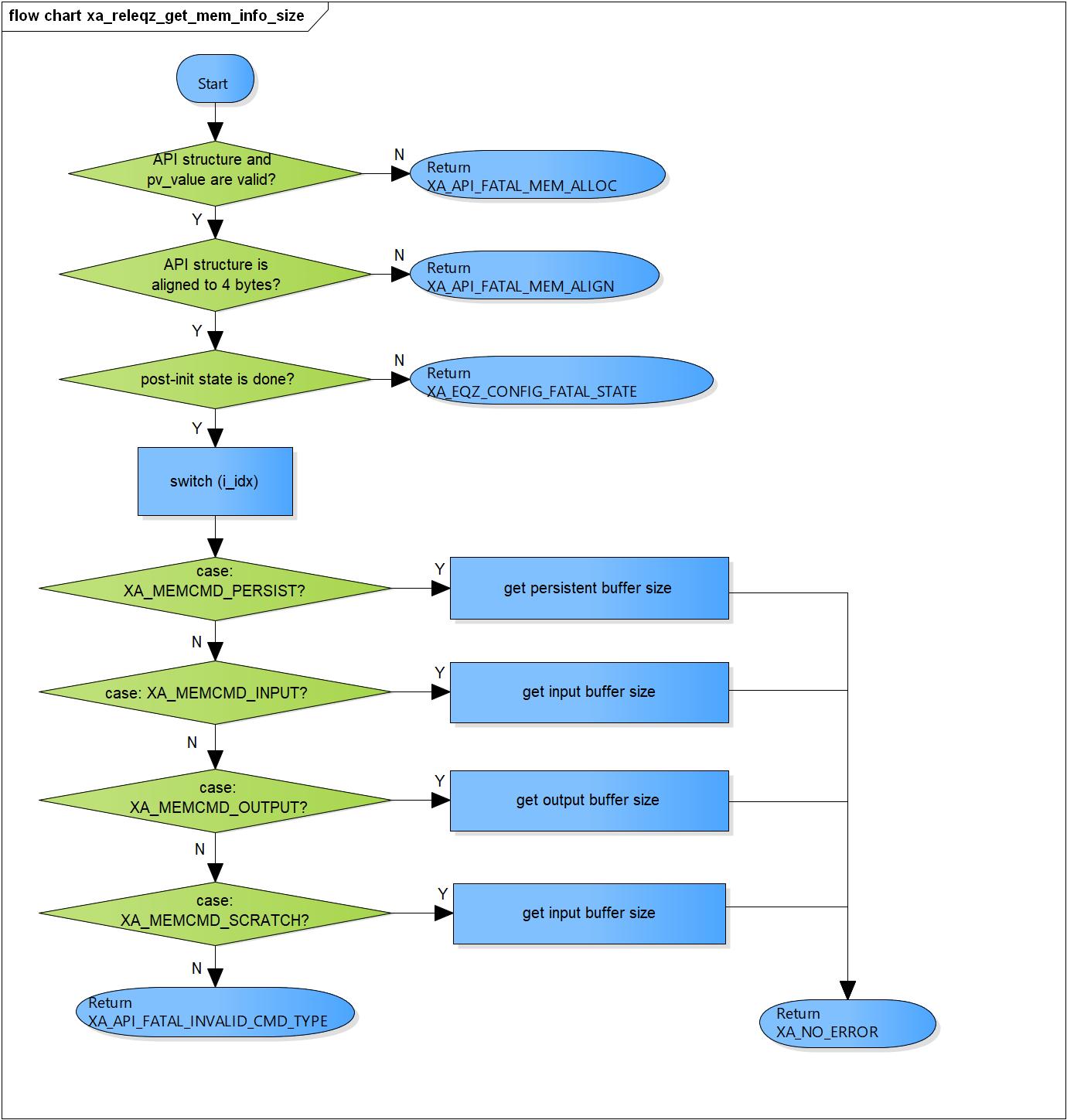


Figure 3‑16 xa\_releqz\_get\_mem\_info\_size flowchart

#### xa\_releqz\_get\_mem\_info\_alignment

DD\_PLG\_EQZ\_01\_013

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_get\_mem\_info\_alignment(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This function gets the alignment information of the memory-type referred to by the index. The alignment required in bytes is returned to the application. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure (struct XAReleqz). |
| WORD32 | i\_idx | I | Index of the memory  0 - Persistent Area  1 - Input Buffer  2 - Output Buffer  3 – Scratch Area |
| pVOID | pv\_value | O | Pointer to the alignment info variable. |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object is NULL.  pv\_value is NULL. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object is not aligned to 4 bytes. | |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE | | Invalid index. | |
| XA\_EQZ\_CONFIG\_FATAL\_STATE | | Equalizer has not post-init yet. | |
| **Description** | * xa\_releqz\_get\_mem\_info\_alignment command processing: * Check API structure is valid. * Check API structure is aligned to 4 bytes. * Check equalizer is post-initialized. * Get alignment information of the memory base on i\_idx. | | | |

[Covers: FD\_PLG\_EQZ\_022]

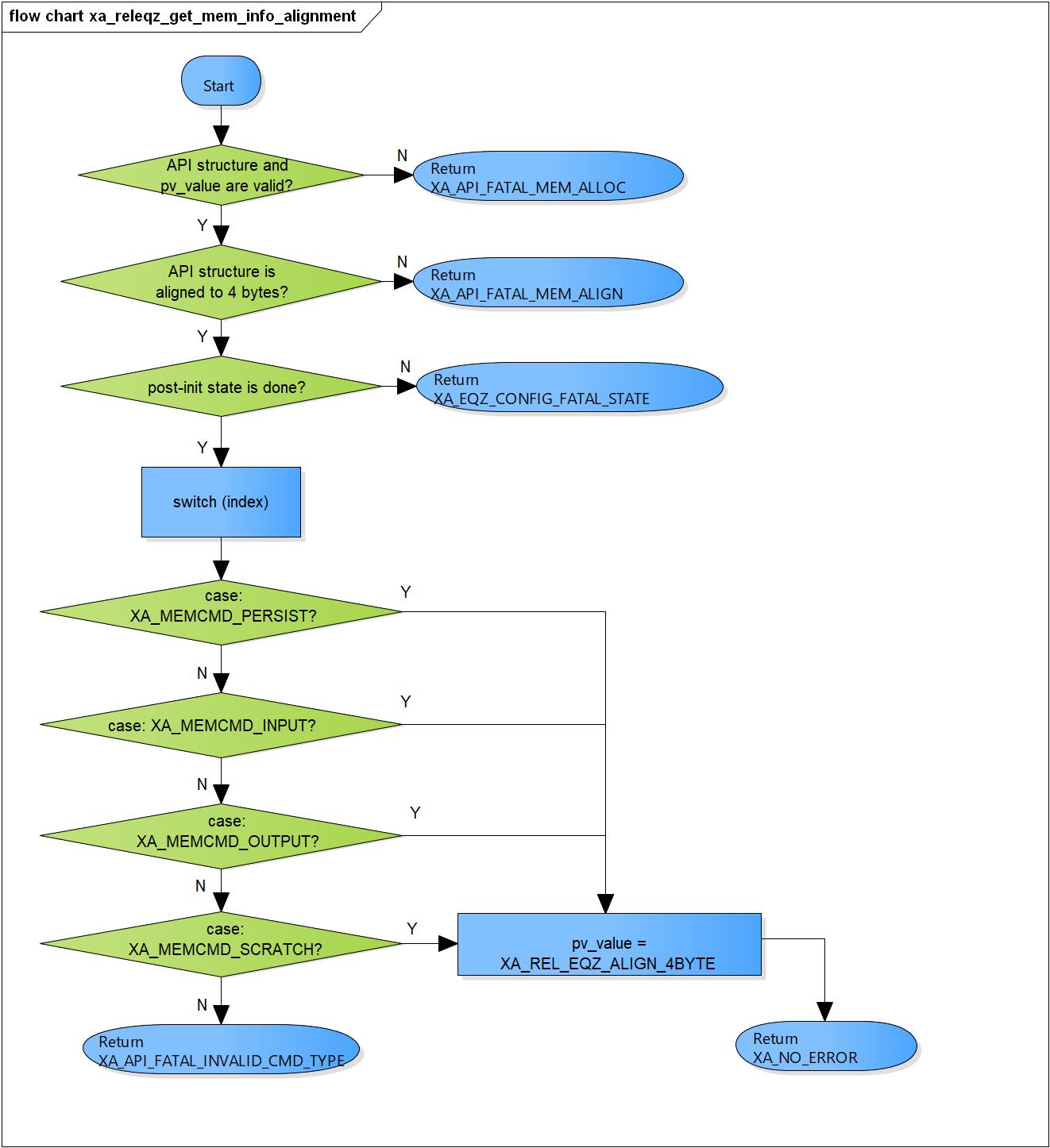


Figure 3‑17 xa\_releqz\_get\_mem\_info\_alignment flowchart

#### xa\_releqz\_get\_mem\_info\_type

DD\_PLG\_EQZ\_01\_014

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_get\_mem\_info\_type(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This function gets the type of memory being referred to by the index. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure (struct XAReleqz). |
| WORD32 | i\_idx | I | Index of the memory  0 - Persistent Area  1 - Input Buffer  2 - Output Buffer  3 – Scratch Area |
| pVOID | pv\_value | O | Pointer to memory type variable. |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object is NULL.  pv\_value is NULL. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object is not aligned to 4 bytes. | |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE | | Invalid index. | |
| **Description** | * xa\_releqz\_get\_mem\_info\_type command processing: * Check API structure is valid. * Check API structure is aligned to 4 bytes. * Check equalizer is post-initialized. * Get type information of the memory base on i\_idx. | | | |

[Covers: FD\_PLG\_EQZ\_021]



Figure 3‑18 xa\_releqz\_get\_mem\_info\_type flowchart

#### xa\_releqz\_set\_mem\_ptr

DD\_PLG\_EQZ\_01\_015

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_set\_mem\_ptr(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This function passes to the equalizer the pointer to the allocated memory. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure (struct XAReleqz). |
| WORD32 | i\_idx | I | Index of the memory  0 - Persistent Area  1 - Input Buffer  2 - Output Buffer  3 – Scratch Area |
| pVOID | pv\_value | I | Pointer to memory buffer allocated. |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object is NULL.  pv\_value is NULL. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object are not aligned to 4 bytes. | |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE | | Invalid index. | |
| XA\_EQZ\_CONFIG\_FATAL\_STATE | | Equalizer has not post-init yet. | |
| **Description** | * xa\_releqz\_set\_mem\_ptr command processing: * Check API structure is valid. * Check API structure is aligned to 4 bytes. * Check Equalizer is post-initialized. * Set memory pointer base on i\_idx. | | | |

[Covers: FD\_PLG\_EQZ\_023]

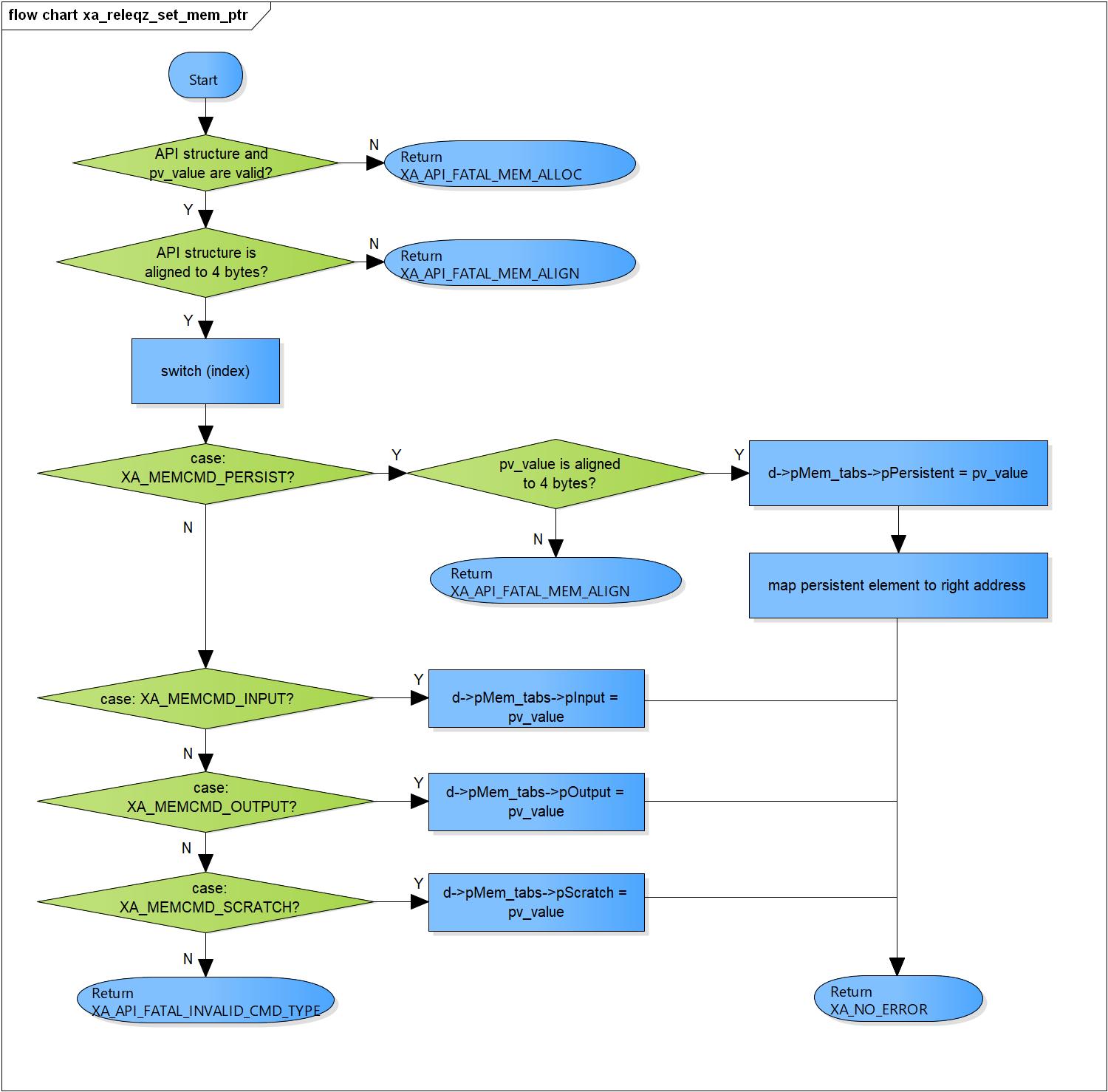


Figure 3‑19 xa\_releqz\_set\_mem\_ptr flowchart

#### xa\_releqz\_input\_over

DD\_PLG\_EQZ\_01\_016

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_input\_over(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This command is used to notify 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. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I/O | Pointer to plugin API structure (struct XAReleqz). |
| WORD32 | i\_idx | x | Not necessary. |
| pVOID | pv\_value | x | Not necessary. |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object is NULL. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object is not aligned to 4 bytes. | |
| **Description** | * xa\_releqz\_input\_over command processing: * Check API structure is valid. * Check API structure is aligned to 4 bytes. * Mark end-of-stream flag. | | | |

[Covers: FD\_PLG\_EQZ\_035]

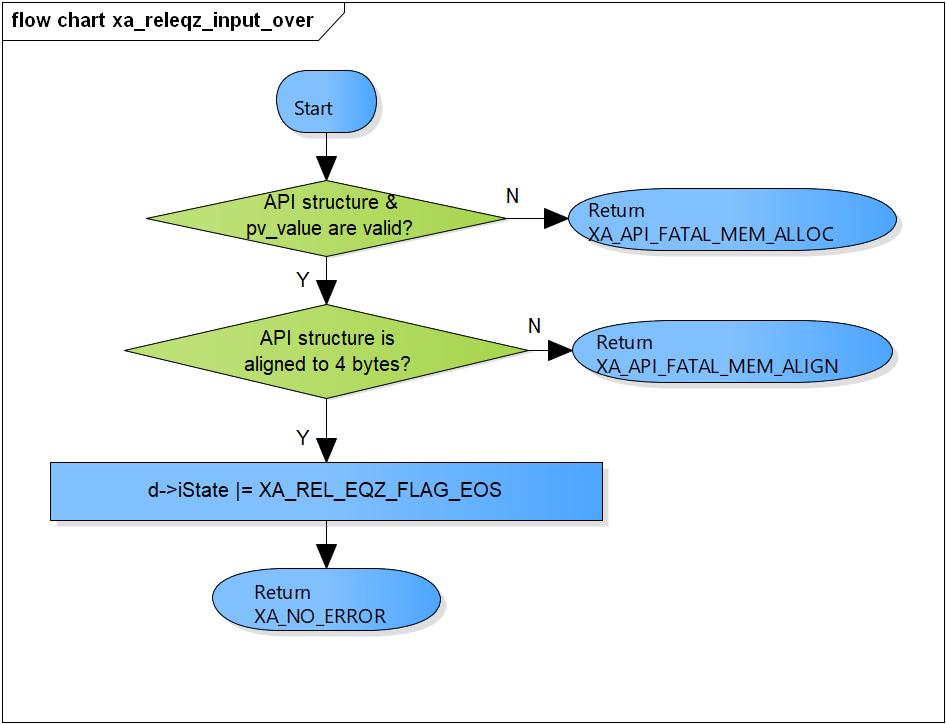


Figure 3‑20 xa\_releqz\_input\_over flowchart

#### xa\_releqz\_set\_input\_bytes

DD\_PLG\_EQZ\_01\_017

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_set\_input\_bytes(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This command is used to set the number of bytes available in the input buffer for the execution. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I/O | Pointer to plugin API structure (struct XAReleqz). |
| WORD32 | i\_idx | x | Not necessary. |
| pVOID | pv\_value | I | Number of input bytes. |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object is NULL. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object is not aligned to 4 bytes. | |
| XA\_EQZ\_EXEC\_FATAL\_STATE | | Equalizer is not post init. | |
| XA\_EQZ\_EXEC\_FATAL\_INPUT | | Input buffer pointer is invalid. | |
| **Description** | * xa\_releqz\_set\_input\_bytes command processing: * Check API structure is valid. * Check API structure is aligned to 4 bytes. * Set input buffer size. | | | |

[Covers: FD\_PLG\_EQZ\_034]

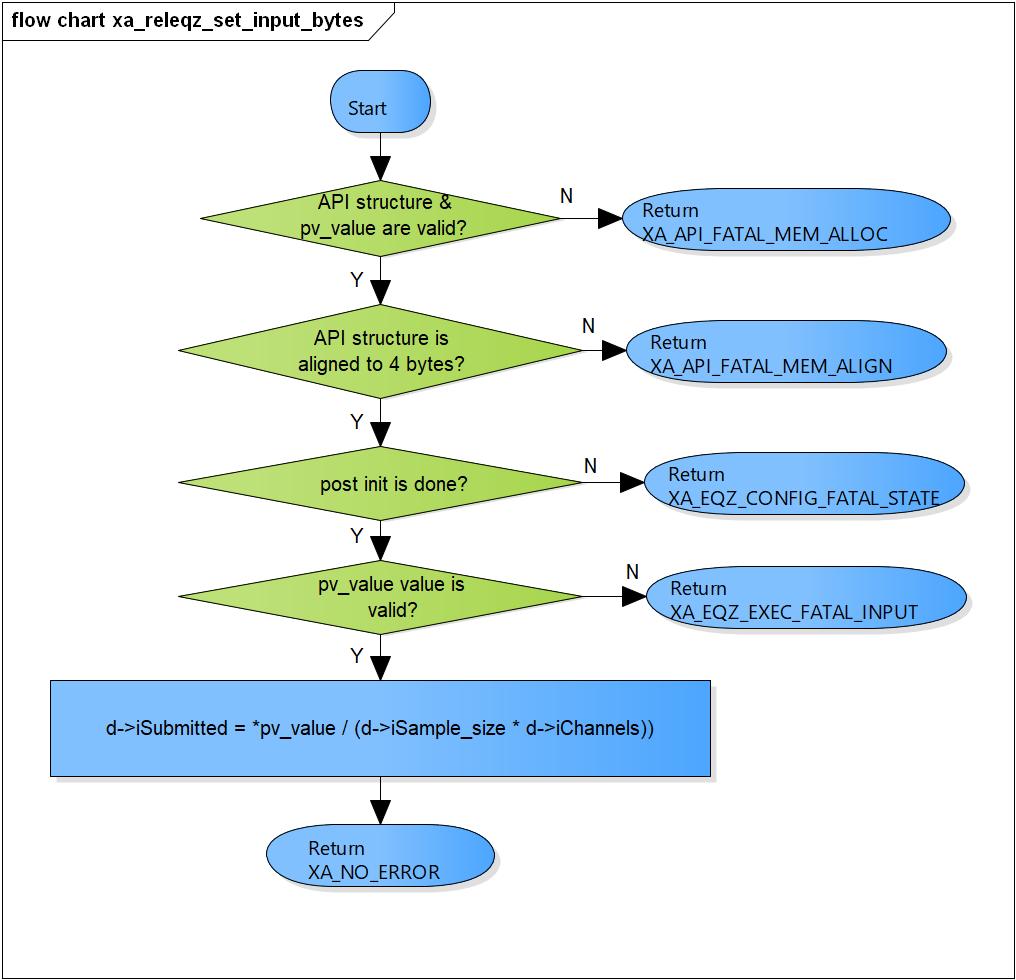


Figure 3‑21 xa\_releqz\_set\_input\_bytes flowchart

#### xa\_releqz\_get\_curidx\_input\_buf

DD\_PLG\_EQZ\_01\_018

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_get\_curidx\_input\_buf(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This function gets the number of input buffer bytes consumed by the equalizer. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure (struct XAReleqz). |
| WORD32 | i\_idx | x | Not necessary. |
| pVOID | pv\_value | x | Pointer to bytes consumed variable. |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object or pv\_value are NULL.  Input buffer is invalid. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object is not aligned to 4 bytes. | |
| **Description** | * xa\_releqz\_get\_curidx\_input\_buf command processing: * Check API structure and pv\_value is valid. * Check API structure is aligned to 4 bytes. * Return number of bytes consumed. | | | |

[Covers: FD\_PLG\_EQZ\_039]

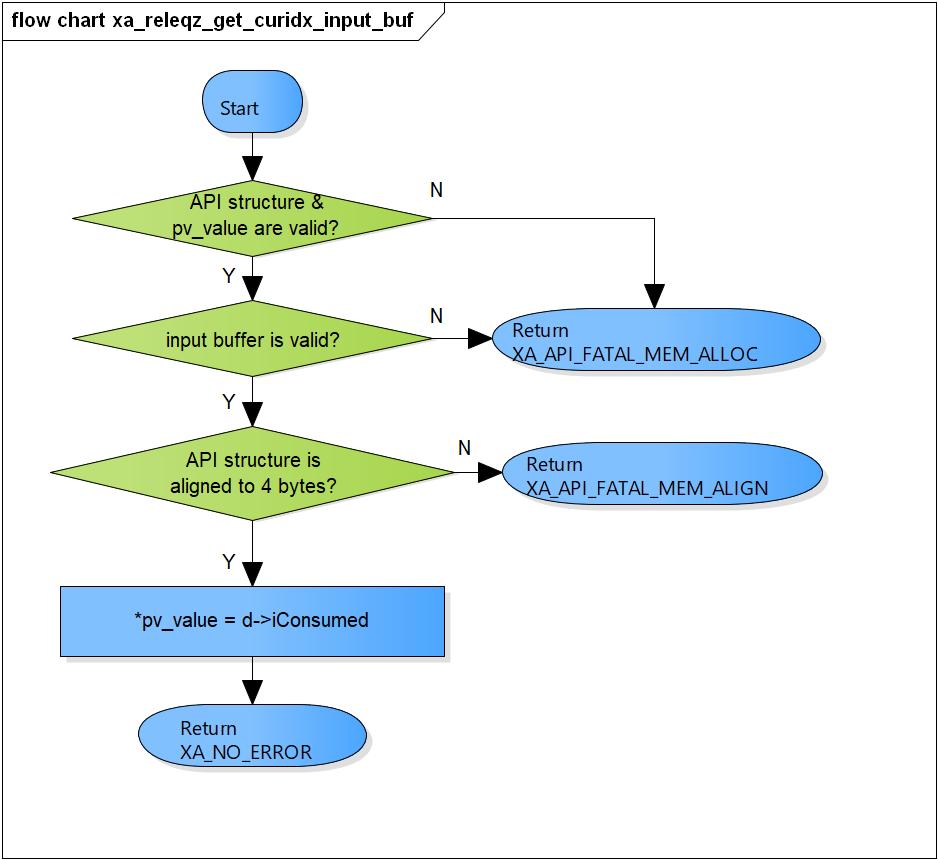


Figure 3‑22 xa\_releqz\_get\_curidx\_input\_buf flowchart

#### xa\_releqz\_execute

DD\_PLG\_EQZ\_01\_019

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_execute(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This function executes the Equalizer plugin in do execute process and completion status of processing in done process. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I/O | Pointer to plugin API structure (struct XAReleqz). |
| WORD32 | i\_idx | I | Index sub commands.  Valid values:  XA\_CMD\_TYPE\_DO\_EXECUTE  XA\_CMD\_TYPE\_DONE\_QUERY  XA\_CMD\_TYPE\_DO\_RUNTIME\_INIT |
| pVOID | pv\_value | I | Depend on every sub commands |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object is NULL. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object is not aligned to 4 bytes. | |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE | | Invalid index. | |
| XA\_EQZ\_EXEC\_FATAL\_STATE | | Equalizer is not running yet | |
| **Description** | * xa\_releqz\_execute command processing: * Check API structure is valid. * Check API structure is aligned to 4 bytes. * Process individual command type base on i\_idx. | | | |

[Covers: FD\_PLG\_EQZ\_036, FD\_PLG\_EQZ\_037]

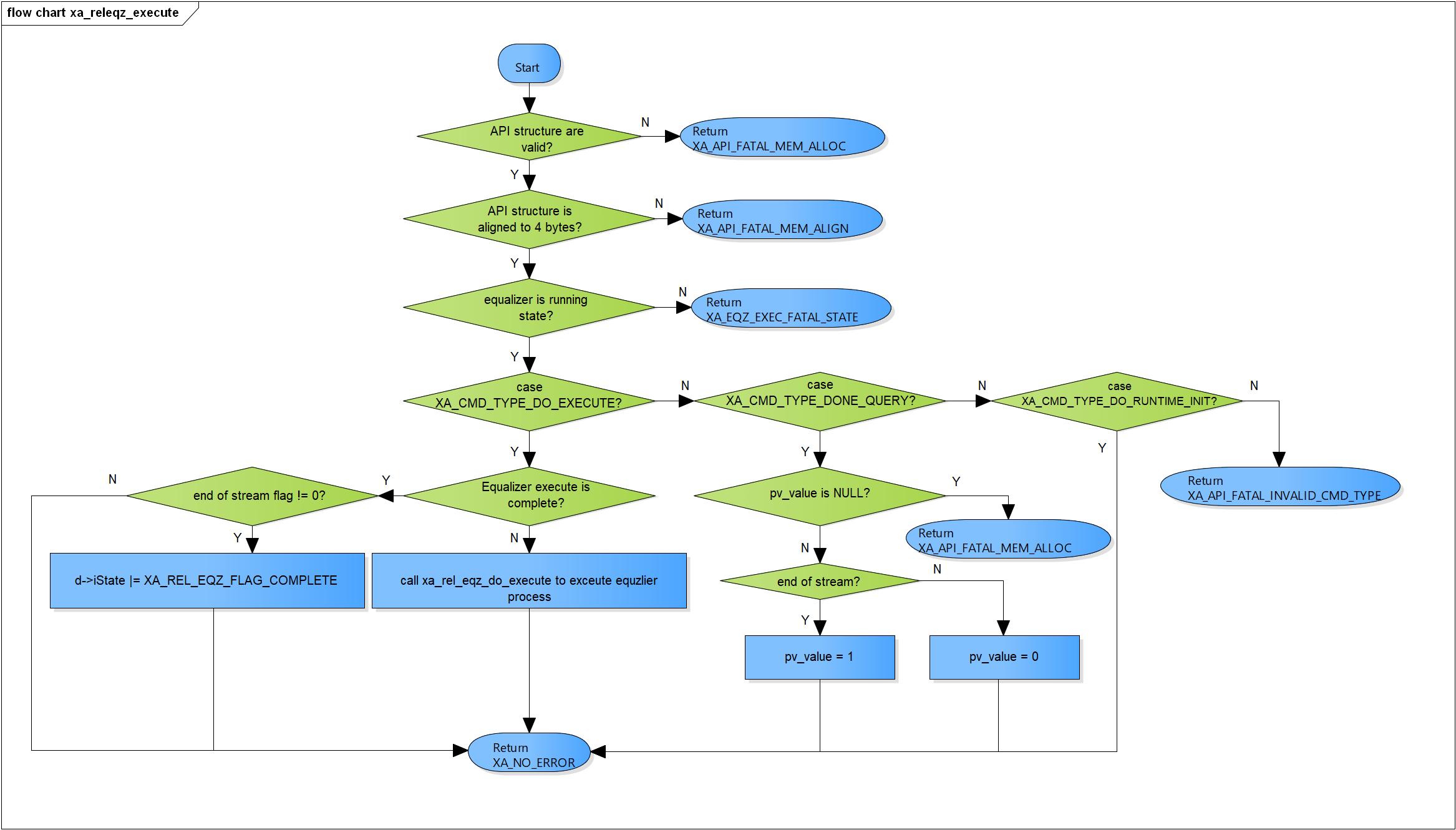


Figure 3‑23 xa\_releqz\_execute flowchart

#### xa\_releqz\_get\_output\_bytes

DD\_PLG\_EQZ\_01\_020

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_get\_output\_bytes(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This function obtains the number of bytes output by the equalizer during the last execution. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure (struct XAReleqz). |
| WORD32 | i\_idx | X | Not necessary. |
| pVOID | pv\_value | I | Pointer to output bytes variable. |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object is NULL.  pv\_value is NULL. | |
| XA\_EQZ\_EXEC\_FATAL\_STATE | | Equalizer plugin is not running. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object is not aligned to 4 bytes. | |
| **Description** | * xa\_releqz\_get\_output\_bytes command processing: * Get output index memory. * Check API structure is valid. * Check API structure is aligned to 4 bytes. * Get produced output samples. | | | |

[Covers: FD\_PLG\_EQZ\_038]

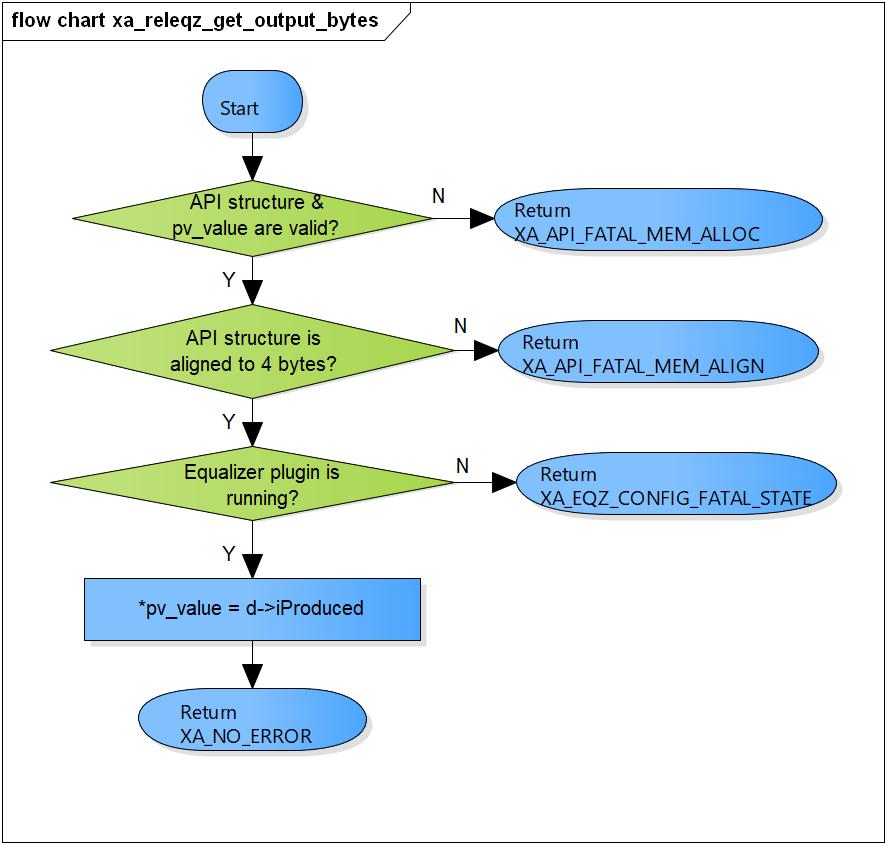


Figure 3‑24 xa\_releqz\_get\_output\_bytes flowchart

#### xa\_releqz\_set\_config\_param

DD\_PLG\_EQZ\_01\_021

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_set\_config\_param(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | This command is used to set the center frequency of a peaking filter or transition frequency of a Bass/Treble filter for filter n. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I/O | Pointer to plugin API structure  (struct XAReleqz). |
| WORD32 | i\_idx | I | Index of sub commands.  Valid values:  XA\_EQZ\_CONFIG\_PARAM\_FILTER\_<n>\_COEF\_FC  XA\_EQZ\_CONFIG\_PARAM\_FILTER\_<n>\_COEF\_TYPE  XA\_EQZ\_CONFIG\_PARAM\_FILTER\_<n>\_COEF\_BW  XA\_EQZ\_CONFIG\_PARAM\_FILTER\_<n>\_COEF\_GA  XA\_EQZ\_CONFIG\_PARAM\_FILTER\_<n>\_COEF\_BA  XA\_EQZ\_CONFIG\_PARAM\_BAND\_<m>\_GCOEF\_GA  XA\_EQZ\_CONFIG\_PARAM\_PCM\_WIDTH  XA\_EQZ\_CONFIG\_PARAM\_CH  XA\_EQZ\_CONFIG\_PARAM\_EQZ\_TYPE  (n = [0: 8], m = [0: 4] |
| pVOID | pv\_value | I | Address that stored center/ transition frequency.  Value range:  - Peaking filter: 20-20kHz (or less than Nyquist frequency)  - Bass filter: 50-500Hz  - Treble filter: 5k - 11kHz |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object or pv\_value is NULL. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object is not aligned to 4 bytes. | |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE | | Invalid index. | |
| XA\_EQZ\_CONFIG\_FATAL\_STATE | | Pre-initialization is not done.  Post-init is done. | |
| XA\_EQZ\_CONFIG\_NONFATAL\_ERR\_FC | | FC’s value is out range (from 20 – 20000). | |
| XA\_EQZ\_CONFIG\_NONFATAL\_ERR\_TYPE | | Type of equalizer is neither through, peak, bass nor treble. | |
| XA\_EQZ\_CONFIG\_FATAL\_ERR\_CH | | Channel is neither stereo nor mono. | |
| XA\_EQZ\_CONFIG\_FATAL\_ERR\_FS | | Sample rate is neither 32kHz, 44,1kHz nor 48kHz. | |
| XA\_EQZ\_CONFIG\_NONFATAL\_ERR\_BW | | BW’s value is out range  (from 26843545 – 2013265920). | |
| XA\_EQZ\_CONFIG\_NONFATAL\_ERR\_GA | | GA’s value for parametric is invalid  (from 47735324 – 1509523500).  GA’s value for graphic is invalid  (from 84886744 – 848867445). | |
| XA\_EQZ\_CONFIG\_NONFATAL\_ERR\_BA | | BA’s value is invalid  (from 84886744 – 848867445). | |
| XA\_EQZ\_CONFIG\_FATAL\_ERR\_PCM\_WIDTH | | PCM width is neither 16 nor 24. | |
| XA\_EQZ\_CONFIG\_NONFATAL\_ERR\_SELECT\_EQZ\_TYPE | | Type of equalizer is neither parametric nor graphic. | |
| **Description** | * xa\_releqz\_set\_config\_param command processing: * Check API structure and pv\_vlue is valid. * Check API structure is aligned to 4 bytes. * Check pre-initialization is done. * Process individual configuration parameter base on i\_idx. | | | |

[Covers: FD\_PLG\_EQZ\_007, FD\_PLG\_EQZ\_008, FD\_PLG\_EQZ\_009, FD\_PLG\_EQZ\_010, FD\_PLG\_EQZ\_011, FD\_PLG\_EQZ\_012, FD\_PLG\_EQZ\_013, FD\_PLG\_EQZ\_014, FD\_PLG\_EQZ\_015, FD\_PLG\_EQZ\_016]

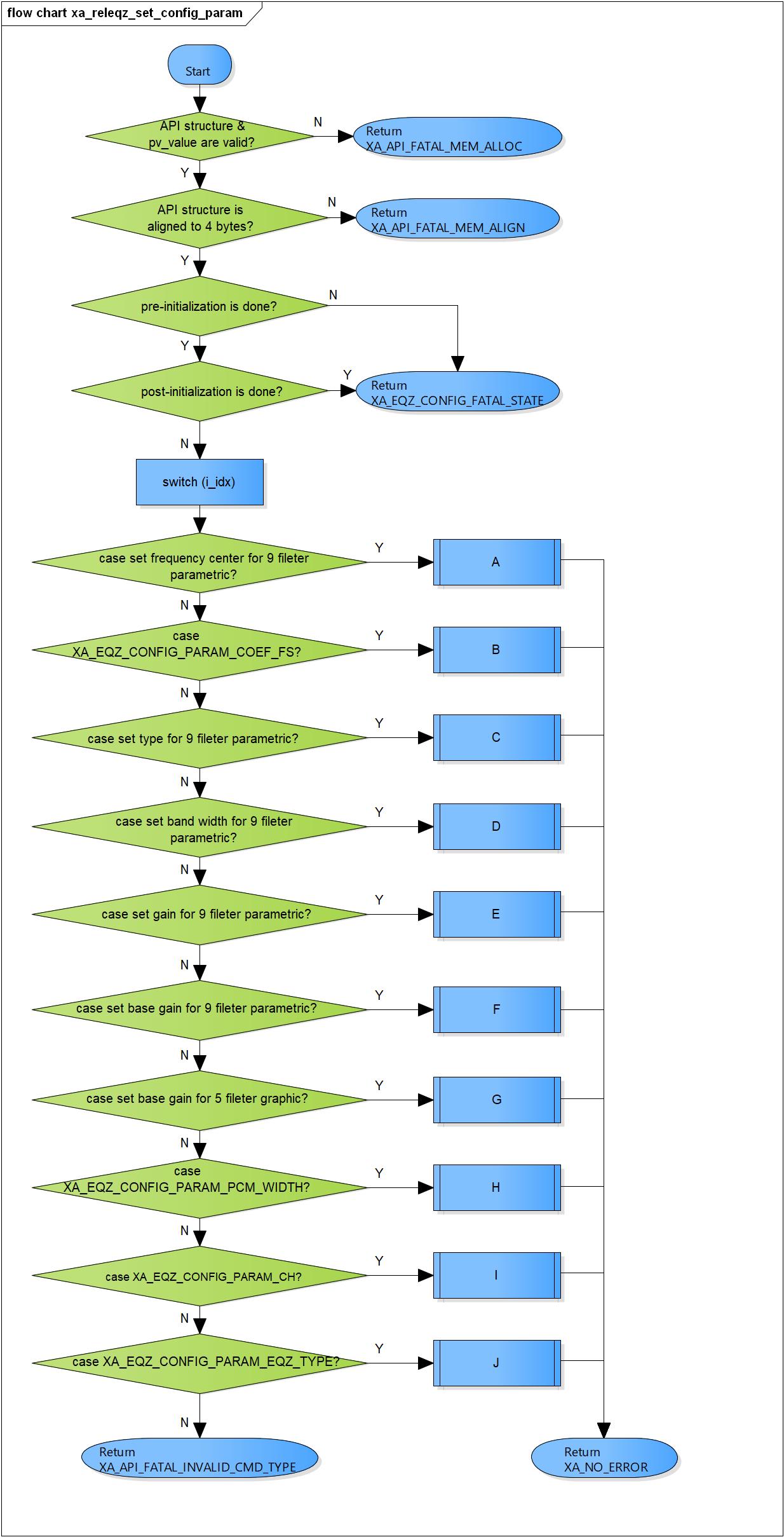


Figure 3‑25 xa\_releqz\_set\_config\_param flowchart

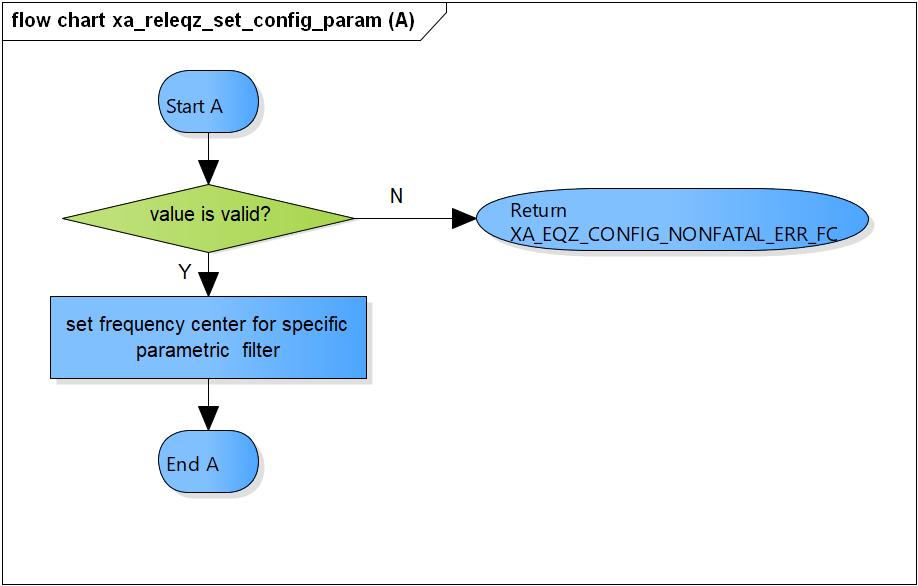


Figure 3‑26 xa\_releqz\_set\_config\_param (A) flowchart

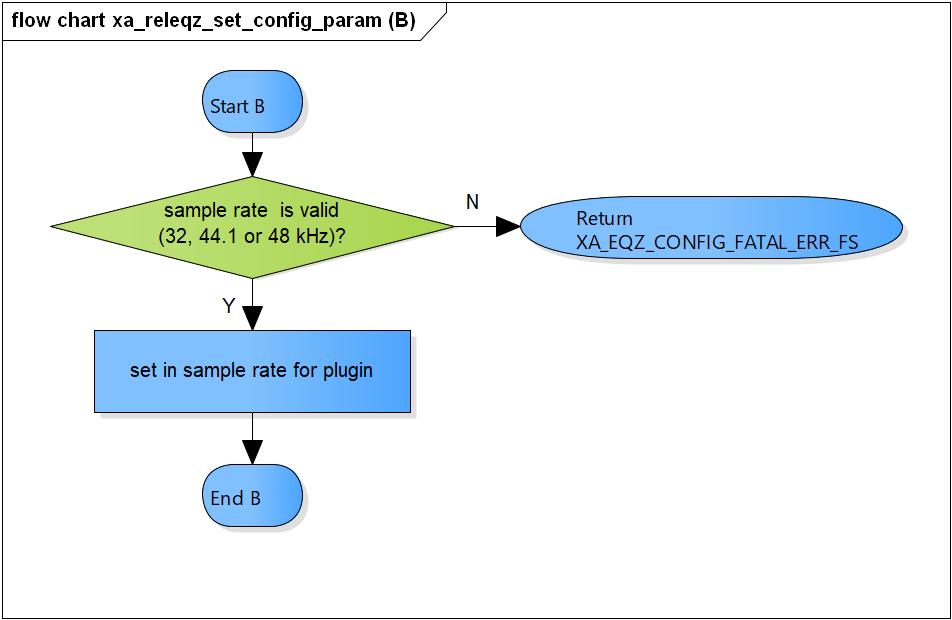


Figure 3‑27 xa\_releqz\_set\_config\_param (B) flowchart

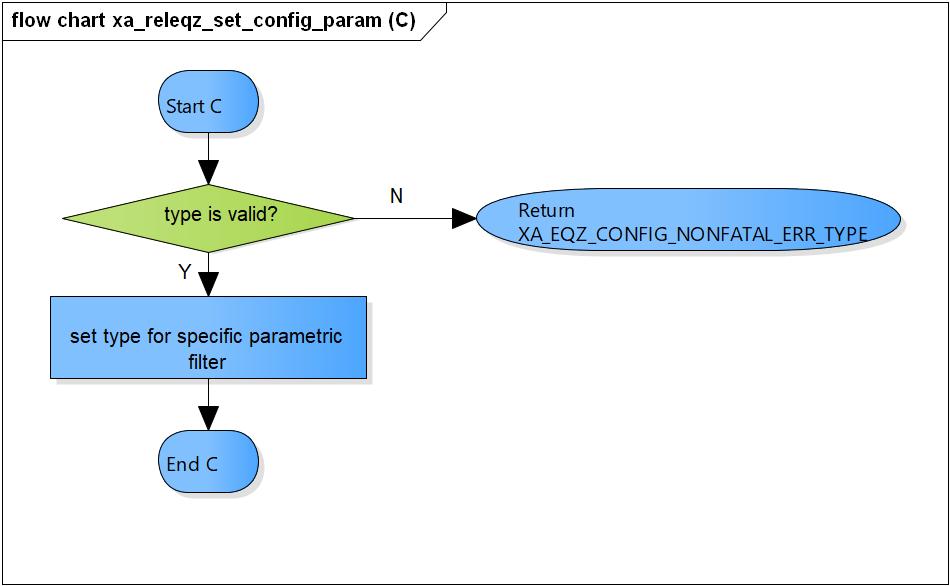


Figure 3‑28 xa\_releqz\_set\_config\_param (C) flowchart

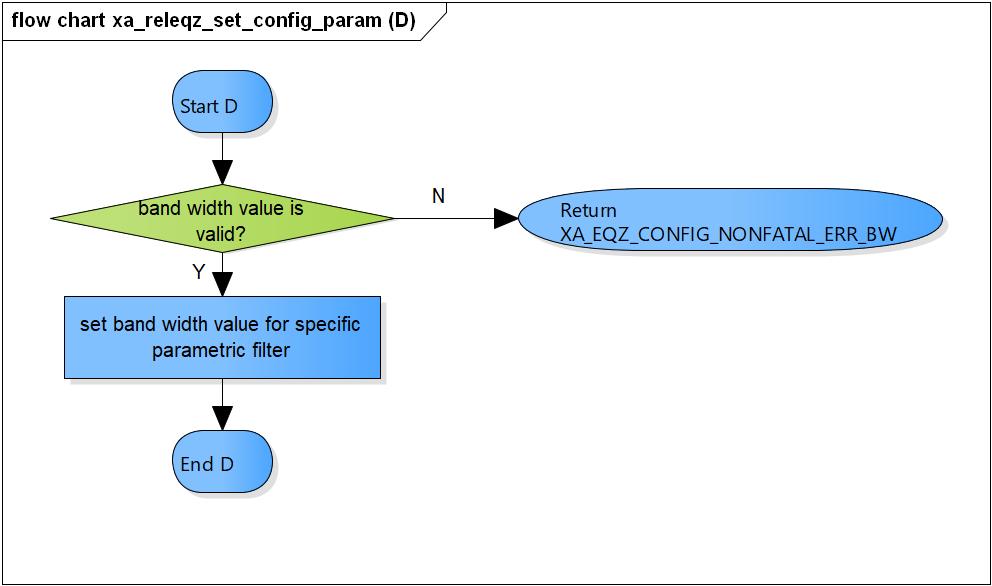


Figure 3‑29 xa\_releqz\_set\_config\_param (D) flowchart

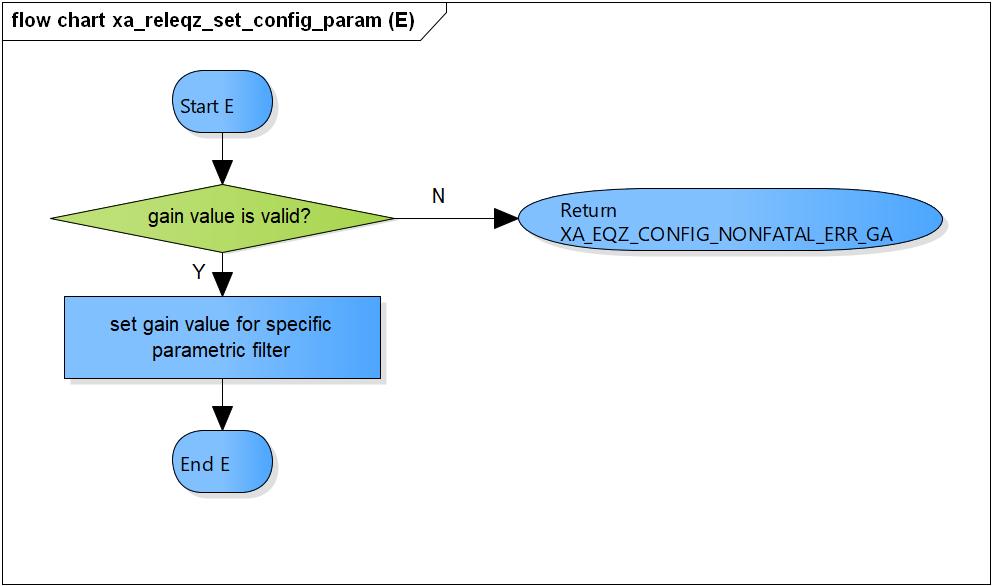


Figure 3‑30 xa\_releqz\_set\_config\_param (E) flowchart

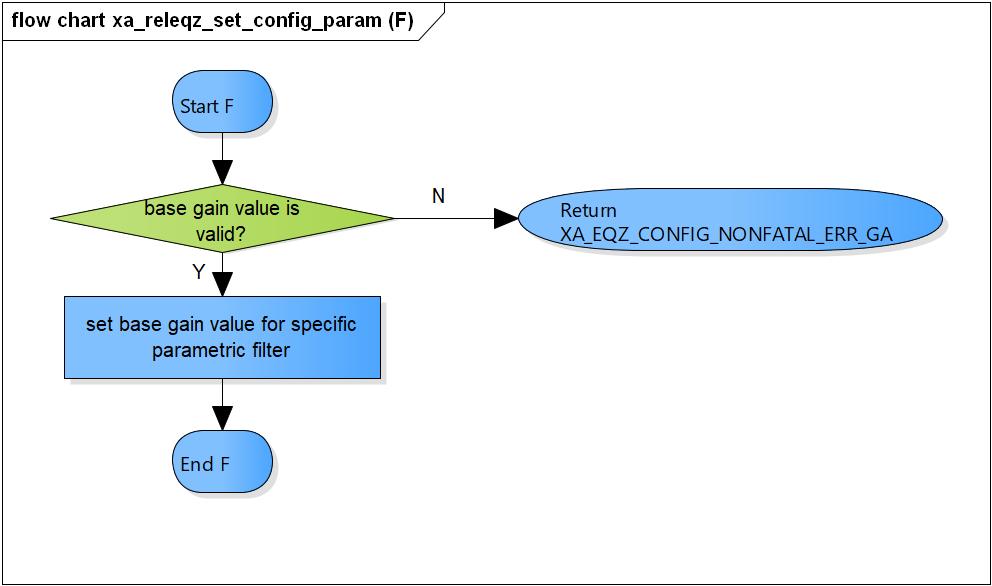


Figure 3‑31 xa\_releqz\_set\_config\_param (F) flowchart

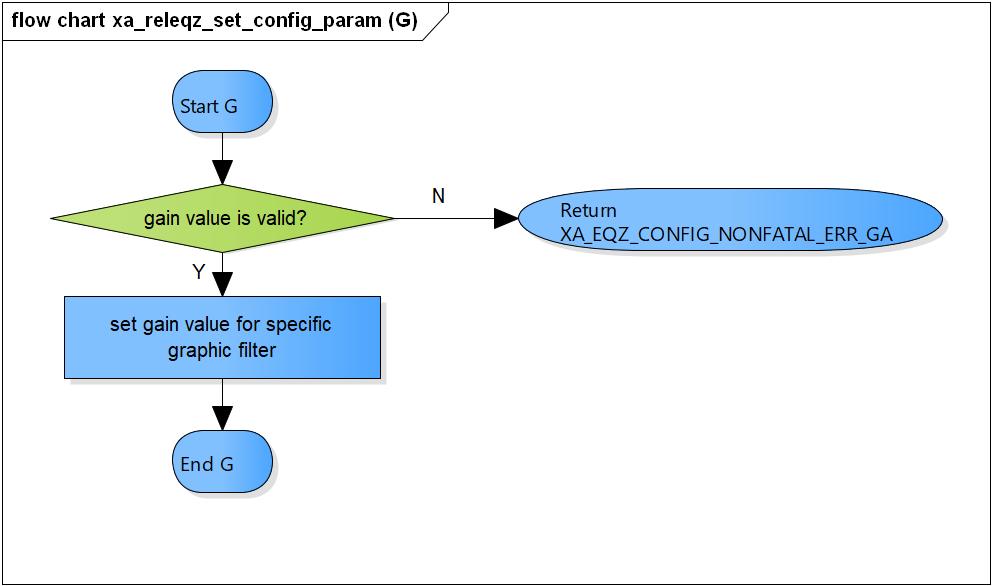


Figure 3‑32 xa\_releqz\_set\_config\_param (G) flowchart

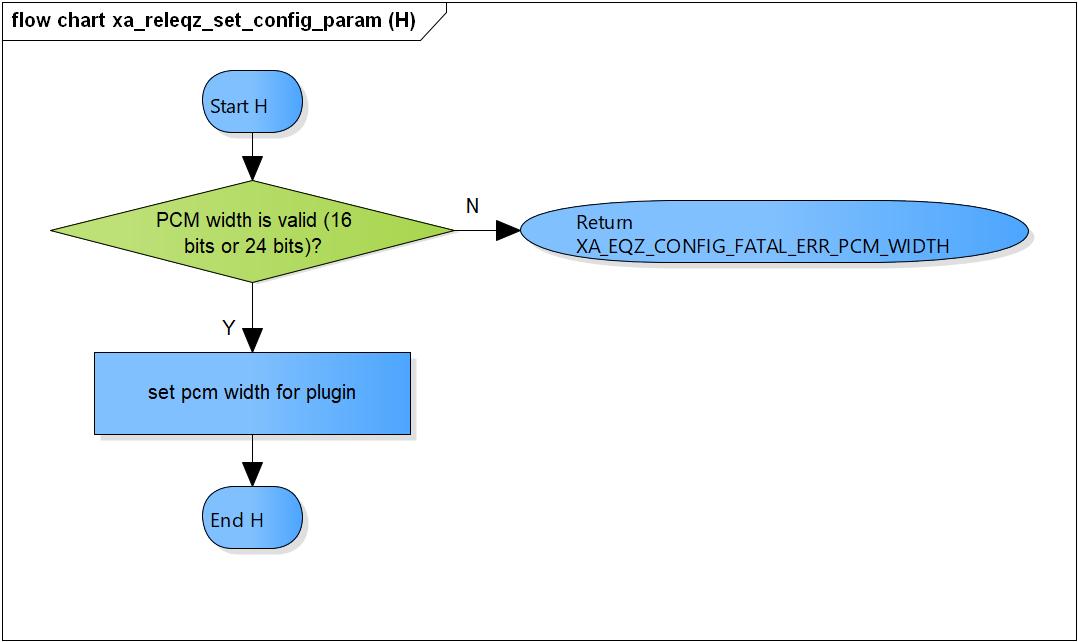


Figure 3‑33 xa\_releqz\_set\_config\_param (H) flowchart

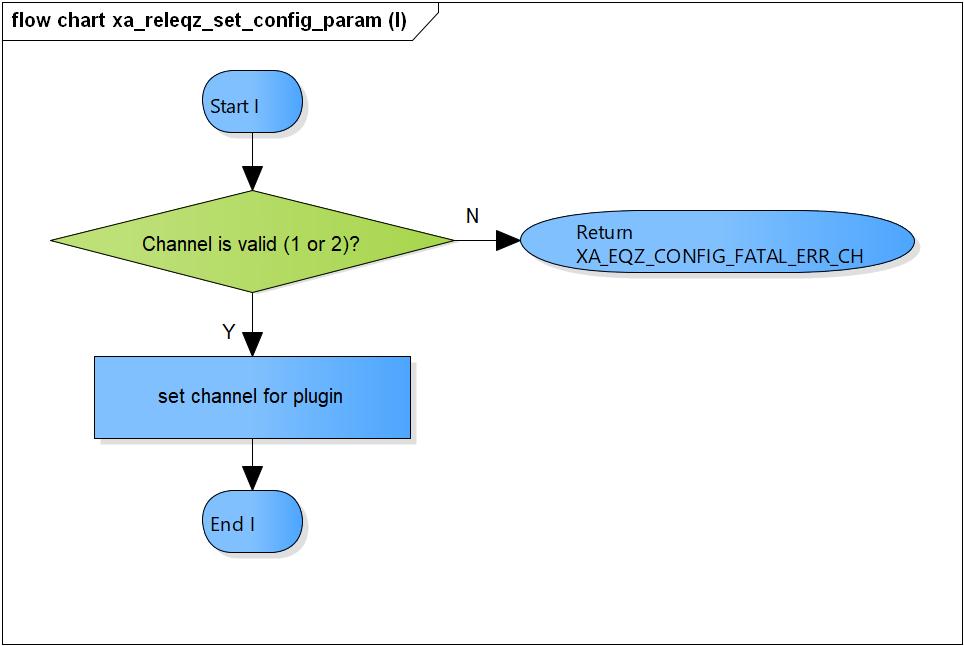


Figure 3‑34 xa\_releqz\_set\_config\_param (I) flowchart

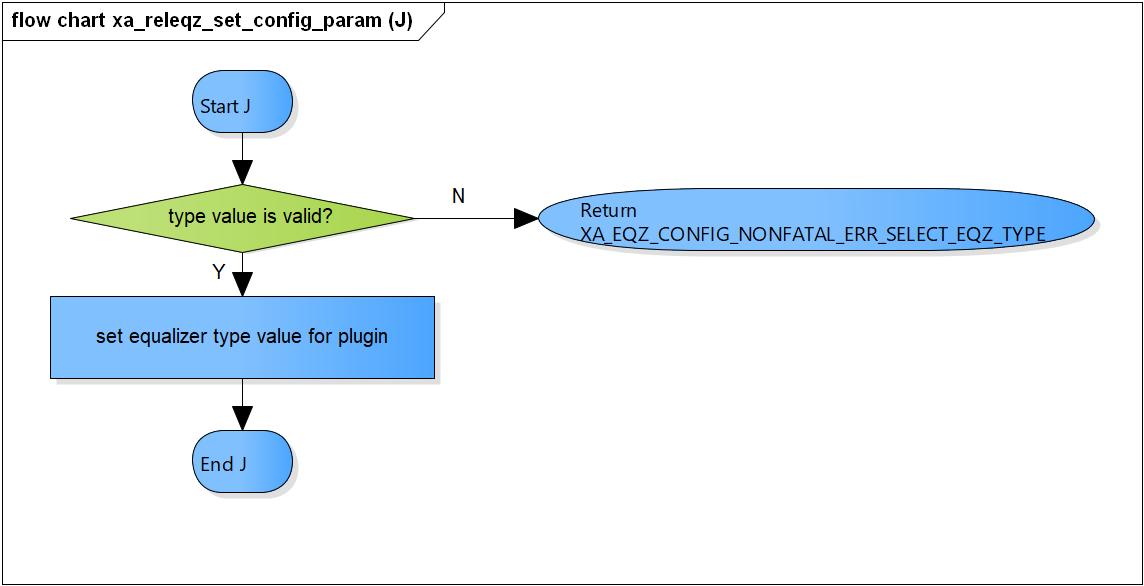


Figure 3‑35 xa\_releqz\_set\_config\_param (J) flowchart

#### xa\_releqz\_get\_config\_param

DD\_PLG\_EQZ\_01\_022

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static XA\_ERRORCODE xa\_releqz\_get\_config\_param(XARelEqz \*d,  WORD32 i\_idx,  pVOID pv\_value) | | | |
| **Function** | Get equalizer configuration parameters. | | | |
| **Arguments** | Type | Name | I/O | Description |
| XARelEqz\* | d | I | Pointer to plugin API structure (struct XAReleqz). |
| WORD32 | i\_idx | I | Index sub commands.  Valid values:  XA\_EQZ\_CONFIG\_PARAM\_FILTER\_<n>\_COEF\_FC  XA\_EQZ\_CONFIG\_PARAM\_FILTER\_<n>\_COEF\_TYPE  XA\_EQZ\_CONFIG\_PARAM\_FILTER\_<n>\_COEF\_BW  XA\_EQZ\_CONFIG\_PARAM\_FILTER\_<n>\_COEF\_GA  XA\_EQZ\_CONFIG\_PARAM\_FILTER\_<n>\_COEF\_BA  XA\_EQZ\_CONFIG\_PARAM\_BAND\_<m>\_GCOEF\_GA  XA\_EQZ\_CONFIG\_PARAM\_PCM\_WIDTH  XA\_EQZ\_CONFIG\_PARAM\_CH  XA\_EQZ\_CONFIG\_PARAM\_EQZ\_TYPE  (n = [0: 8], m = [0: 4] |
| pVOID | pv\_value | O | Pointer to parameters variable. |
| **Return value** | XA\_NO\_ERROR | | Normal return. | |
| XA\_API\_FATAL\_MEM\_ALLOC | | API object or pv\_value is NULL. | |
| XA\_API\_FATAL\_MEM\_ALIGN | | API object is not aligned to 4 bytes. | |
| XA\_API\_FATAL\_INVALID\_CMD\_TYPE | | Invalid index. | |
| XA\_EQZ\_EXEC\_FATAL\_STATE | | Plugin has not pre-init yet. | |
| **Description** | * xa\_releqz\_get\_config\_param command processing: * Check API structure is valid. * Check API structure is aligned to 4 bytes. * Check pre-initialization is completed. * Process individual configuration parameter base on i\_idx. | | | |

[Covers: FD\_PLG\_EQZ\_024, FD\_PLG\_EQZ\_025, FD\_PLG\_EQZ\_026, FD\_PLG\_EQZ\_027, FD\_PLG\_EQZ\_028, FD\_PLG\_EQZ\_029, FD\_PLG\_EQZ\_030, FD\_PLG\_EQZ\_031, FD\_PLG\_EQZ\_032, FD\_PLG\_EQZ\_033]

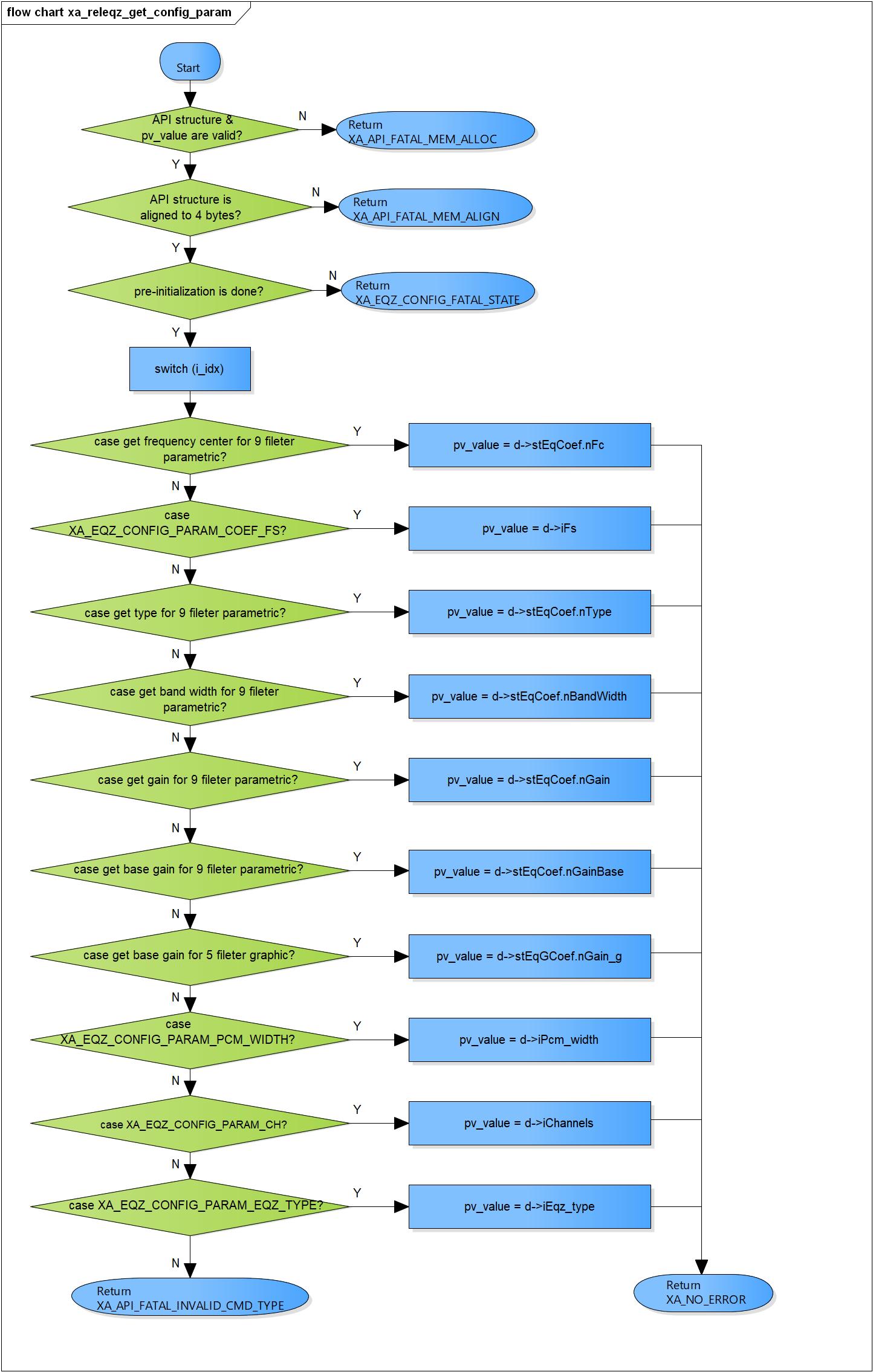


Figure 3‑36 xa\_releqz\_get\_config\_param flowchart

# Revision history

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Date** | **Page** | **Content** | **Approved** | **Changed** |
| 1.0.0 | Nov 14 2018 | - | First Edition issued | Vu Phan | Ngu Pham |
| 1.1.0 | Dec 10 2018 | - | Add traceability ID | Vu Phan | Ngu Pham |
| 1.2.0 | Jan 03, 2019 | - | Add range for input parameters | Vu Phan | Tien Tran |