**Table of Contents**

[1. Overview 5](#_Toc6909280)

[2. ADSP Base 9](#_Toc6909281)

[2.1 Structure 10](#_Toc6909282)

[2.1.1 xf\_callback\_func structure 10](#_Toc6909283)

[2.1.2 xf\_adsp\_base\_cmd structure 10](#_Toc6909284)

[2.1.3 xf\_handle structure 11](#_Toc6909285)

[2.1.4 xf\_adsp\_base structure 11](#_Toc6909286)

[2.1.5 xf\_adsp\_renderer\_params structure 11](#_Toc6909287)

[2.1.6 xf\_adsp\_renderer structure 12](#_Toc6909288)

[2.1.7 xf\_adsp\_capture\_params structure 12](#_Toc6909289)

[2.1.8 xf\_adsp\_capture structure 12](#_Toc6909290)

[2.1.9 xf\_equalizer\_parametric\_coef structure 12](#_Toc6909291)

[2.1.10 xf\_equalizer\_graphic\_coef structure 13](#_Toc6909292)

[2.1.11 xf\_adsp\_equalizer\_params structure 13](#_Toc6909293)

[2.1.12 xf\_adsp\_equalizer structure 13](#_Toc6909294)

[2.1.13 xf\_adsp\_tdm\_renderer\_params structure 13](#_Toc6909295)

[2.1.14 xf\_adsp\_tdm\_renderer structure 14](#_Toc6909296)

[2.1.15 xf\_adsp\_tdm\_capture\_params structure 14](#_Toc6909297)

[2.1.16 xf\_adsp\_tdm\_capture structure 14](#_Toc6909298)

[2.2 Interface APIs 15](#_Toc6909299)

[2.2.1 xf\_adsp\_base\_create 15](#_Toc6909300)

[2.2.2 xf\_adsp\_base\_destroy 15](#_Toc6909301)

[2.2.3 xf\_adsp\_empty\_this\_buffer 15](#_Toc6909302)

[2.2.4 xf\_adsp\_fill\_this\_buffer 16](#_Toc6909303)

[2.2.5 xf\_adsp\_mmap\_this\_buffer 16](#_Toc6909304)

[2.2.6 xf\_adsp\_allocate\_mem\_pool 16](#_Toc6909305)

[2.2.7 xf\_adsp\_free\_mem\_pool 16](#_Toc6909306)

[2.2.8 xf\_adsp\_get\_data\_from\_pool 18](#_Toc6909307)

[2.2.9 xf\_adsp\_set\_param 18](#_Toc6909308)

[2.2.10 xf\_adsp\_get\_param 18](#_Toc6909309)

[2.2.11 xf\_adsp\_route 19](#_Toc6909310)

[2.2.12 xf\_adsp\_renderer\_create 19](#_Toc6909311)

[2.2.13 xf\_adsp\_renderer\_destroy 20](#_Toc6909312)

[2.2.14 xf\_adsp\_renderer\_set\_params 20](#_Toc6909313)

[2.2.15 xf\_adsp\_renderer\_get\_params 20](#_Toc6909314)

[2.2.16 xf\_adsp\_capture\_create 21](#_Toc6909315)

[2.2.17 xf\_adsp\_capture\_destroy 21](#_Toc6909316)

[2.2.18 xf\_adsp\_capture\_set\_params 22](#_Toc6909317)

[2.2.19 xf\_adsp\_capture\_get\_params 22](#_Toc6909318)

[2.2.20 xf\_adsp\_equalizer\_create 23](#_Toc6909319)

[2.2.21 xf\_adsp\_equalizer\_destroy 23](#_Toc6909320)

[2.2.22 xf\_adsp\_equalizer\_set\_params 24](#_Toc6909321)

[2.2.23 xf\_adsp\_equalizer\_get\_params 24](#_Toc6909322)

[2.2.24 xf\_adsp\_tdm\_renderer\_create 25](#_Toc6909323)

[2.2.25 xf\_adsp\_tdm\_renderer\_destroy 25](#_Toc6909324)

[2.2.26 xf\_adsp\_tdm\_renderer\_set\_params 26](#_Toc6909325)

[2.2.27 xf\_adsp\_tdm\_renderer\_get\_params 26](#_Toc6909326)

[2.2.28 xf\_adsp\_tdm\_capture\_create 27](#_Toc6909327)

[2.2.29 xf\_adsp\_tdm\_capture\_destroy 27](#_Toc6909328)

[2.2.30 xf\_adsp\_tdm\_capture\_set\_params 28](#_Toc6909329)

[2.2.31 xf\_adsp\_tdm\_capture\_get\_params 28](#_Toc6909330)

[3. Proxy Driver Interface - Extension APIs for ADSP Base 29](#_Toc6909331)

[3.1 xf\_adsp\_base\_client\_register 29](#_Toc6909332)

[3.2 xf\_adsp\_base\_client\_unregister 29](#_Toc6909333)

[3.3 xf\_adsp\_base\_send 29](#_Toc6909334)

[3.4 xf\_adsp\_base\_recv 30](#_Toc6909335)

[3.5 xf\_adsp\_base\_poll 30](#_Toc6909336)

[4. Processing Flow 31](#_Toc6909337)

[4.1 ADSP Base Flow 31](#_Toc6909338)

[4.1.1 ADSP Base Creation 31](#_Toc6909339)

[4.1.2 ADSP Base Destruction 32](#_Toc6909340)

[4.2 Renderer Flow 33](#_Toc6909341)

[4.2.1 Renderer Creation 33](#_Toc6909342)

[4.2.2 Renderer Execution 34](#_Toc6909343)

[4.2.3 Renderer Destruction 35](#_Toc6909344)

[4.3 Capture Flow 36](#_Toc6909345)

[4.3.1 Capture Creation 36](#_Toc6909346)

[4.3.2 Capture Execution 37](#_Toc6909347)

[4.3.3 Capture Destruction 38](#_Toc6909348)

[4.4 Equalizer Flow 39](#_Toc6909349)

[4.4.1 Equalizer Creation 39](#_Toc6909350)

[4.4.2 Equalizer Execution 40](#_Toc6909351)

[4.4.3 Equalizer Destruction 41](#_Toc6909352)

[4.5 TDM Renderer Flow 42](#_Toc6909353)

[4.5.1 TDM Renderer Creation 42](#_Toc6909354)

[4.5.2 TDM Renderer Execution 43](#_Toc6909355)

[4.5.3 TDM Renderer Destruction 44](#_Toc6909356)

[4.6 TDM Capture Flow 45](#_Toc6909357)

[4.6.1 TDM Capture Creation 45](#_Toc6909358)

[4.6.2 TDM Capture Execution 46](#_Toc6909359)

[4.6.3 TDM Capture Destruction 47](#_Toc6909360)

[5. Appendix 48](#_Toc6909361)

[5.1 Error Code 48](#_Toc6909362)

[5.2 Structures 48](#_Toc6909363)

**List of Tables**

[Table 1‑1 Development environment 5](#_Toc520184183)

[Table 1‑2 Build environment 5](#_Toc520184184)

[Table 2‑1 List of structure in ADSP base 9](#_Toc520184185)

[Table 2‑2 List of APIs in ADSP base 10](#_Toc520184186)

[Table 3‑1 List of extension APIs for proxy driver 27](#_Toc520184187)

[Table 5‑1 Generic error code definition 46](#_Toc520184188)

[Table 5‑2 External structure definition 46](#_Toc520184189)

**List of Figures**

[Figure 1‑1 Overview configuration of ADSP driver 6](#_Toc520184190)

[Figure 4‑1 ADSP base creation flow chart 29](#_Toc520184191)

[Figure 4‑2 ADSP base destruction flow chart 30](#_Toc520184192)

[Figure 4‑3 Renderer creation flow chart 31](#_Toc520184193)

[Figure 4‑4 Renderer execution flow chart 32](#_Toc520184194)

[Figure 4‑5 Renderer destruction flow chart 33](#_Toc520184195)

[Figure 4‑6 Capture creation flow chart 34](#_Toc520184196)

[Figure 4‑7 Capture execution flow chart 35](#_Toc520184197)

[Figure 4‑8 Capture destruction flow chart 36](#_Toc520184198)

[Figure 4‑9 Equalizer creation flow chart 37](#_Toc520184199)

[Figure 4‑10 Equalizer execution flow chart 38](#_Toc520184200)

[Figure 4‑11 Equalizer destruction flow chart 39](#_Toc520184201)

[Figure 4‑12 TDM Renderer creation flow chart 40](#_Toc520184202)

[Figure 4‑13 TDM Renderer execution flow chart 41](#_Toc520184203)

[Figure 4‑14 TDM Renderer destruction flow chart 42](#_Toc520184204)

[Figure 4‑15 TDM Capture creation flow chart 43](#_Toc520184205)

[Figure 4‑16 TDM Capture execution flow chart 44](#_Toc520184206)

[Figure 4‑17 TDM Capture destruction flow chart 45](#_Toc520184207)

# Overview

The below figure shows the basic configuration of ADSP driver and the connection with the ALSA framework when performs playback and record to ADSP device.

ALSA SoC core

Shared memory area

ADSP

Audio applications

*User side*

*Kernel side*

*Hardware* side

ALSA Lib

Renderer/Capture/ Equalizer/TDM Renderer/TDM Capture plugin

*PCM parameters*

*Extension parameters (volume, SRC, Equalizer)*

PCM interface

Control interface

*Kernel* side

ALSA framework supported by Linux

ALSA SoC core

Shared memory, ADSP hardware, codec

ADSP Driver Extension

PCM data

Communication

Platform driver, Machine driver, Codec driver

Audio application layer

***Legend:***

Plugins (Capture, Renderer, Equalizer, TDM Capture, TDM Renderer)

The target of this document is in side of red square.

ALSA Middleware Layer

Platform driver (rcar\_adsp\_sound)

ADSP ALSA

ADSP Driver Extension

CPU DAI

Codec driver

Codec DAI

Codec DAI

Codec DAI

ADSP ALSA

CPU DAI and Codec DAI

PCM Interface, Control Interface

Proxy Extension Interface

Proxy Extension Interface

ADSP Driver

Codec

ADSP Driver

Codec

Codec

CPU DAI

CPU DAI

Machine driver

Machine driver

Machine driver

Figure 1‑1 Overview configuration of ADSP driver

* **Audio applications (aplay, arecord, amixer, etc)**:

The user applications that support to play or record sound by using ALSA library.

* **ALSA Lib**:

The ALSA library APIs are the interface to the ALSA drivers.

* **ALSA Middle Layer**:

It is a set of libraries which APIs gives applications access to the sound card drivers. And it can be broken down into the major interfaces such as control interface, PCM interface, raw MIDI interface, timer interface, sequencer interface and mixer interface.

* **ALSA SoC core:**

It is part of ALSA Framework and does processing of PCM data

* **ADSP ALSA**:

It is an ALSA device driver, implements to register a sound card for ADSP device. It provides callback functions for the native supports from ALSA framework to perform both playback and record. For playback/TDM playback, it receives PCM data from user app and transfers to ADSP Renderer plugin/ADSP TDM Renderer plugin. For record, it receives PCM data from ADSP Capture plugin/ADSP TDM Capture plugin and transfers to user app. The equalization function can be integrated into playback and record by routing between Equalizer and Renderer plugin, and between Equalizer and Capture plugins.

* **CPU DAI**:

DAI stands for Digital Audio Interface. CPU DAI is the interface for the platform driver to communicate with other drivers.

* **Platform driver**:

This is used to register ADSP sound card into ASoC framework. It holds ADSP ALSA driver, ADSP Driver Extension and ADSP sound card.

* **Codec driver**:

It represents interface for codecs.

* **Codec DAI**:

The DAI for codecs to communicate with other drivers

* **Machine driver**:

The ASoC machine (or board) driver is the code that glues together the platform driver and codec driver.

* **Proxy Extension Interface**:

APIs of methods through which ADSP Driver Extension communicates with shared memory area in Hardware side.

* **Shared memory area**:

Shared memory is a memory area which can be read and written by both CPU and ADSP.

* **ADSP**:

It is an audio DSP hardware unit. It provides ADSP framework which has the capability to control and execute multiple plugins (Renderer/Capture/Equalizer/TDM Renderer/TDM Capture) for playback, record, TDM and equalization. The communication between ADSP side and CPU side is performed by the interrupt, and the shared memory area.

# ADSP Base

The below table shows list of structures in the ADSP base.

|  |  |  |
| --- | --- | --- |
| No. | Structures | Descriptions |
| 1 | struct xf\_callback\_func | The callback functions for ADSP ALSA driver |
| 2 | struct xf\_callback\_base\_cmd | The callback functions for ADSP base |
| 3 | struct xf\_handle | The registered handle’s data |
| 4 | struct xf\_adsp\_base | ADSP base’s processing data |
| 5 | struct xf\_adsp\_renderer\_params | Renderer plugin’s parameters |
| 6 | struct xf\_adsp\_renderer | Renderer component’s data |
| 7 | struct xf\_adsp\_capture\_params | Capture plugin’s parameters |
| 8 | struct xf\_adsp\_capture | Capture component’s data |
| 9 | struct xf\_equalizer\_parametric\_coef | Parametric coefficients of Equalizer plugin |
| 10 | struct xf\_equalizer\_graphic\_coef | Graphic coefficients of Equalizer plugin |
| 11 | struct xf\_adsp\_equalizer\_params | Equalizer plugin’s parameters |
| 12 | struct xf\_adsp\_equalizer | Equalizer component’s data |
| 13 | struct xf\_adsp\_tdm\_renderer\_params | TDM Renderer plugin’s parameters |
| 14 | struct xf\_adsp\_tdm\_renderer | TDM Renderer component’s data |
| 15 | struct xf\_adsp\_tdm\_capture\_params | TDM Capture plugin’s parameters |
| 16 | struct xf\_adsp\_tdm\_capture | TDM Capture component’s data |

Table 2‑1 List of structure in ADSP base

The below table shows list of APIs in the ADSP base.

|  |  |  |
| --- | --- | --- |
| No. | APIs | Descriptions |
| 1 | xf\_adsp\_base\_create | Create ADSP base control |
| 2 | xf\_adsp\_base\_destroy | Destroy ADSP base control |
| 3 | xf\_adsp\_empty\_this\_buffer | Send data to ADSP plugin |
| 4 | xf\_adsp\_fill\_this\_buffer | Get data from ADSP plugin |
| 5 | xf\_adsp\_allocate\_mem\_pool | Allocate memory pool from ADSP |
| 6 | xf\_adsp\_free\_mem\_pool | Free memory pool to ADSP |
| 7 | xf\_adsp\_get\_data\_from\_pool | Get data buffer from the given pool |
| 8 | xf\_adsp\_set\_param | Set a single parameter to a registered plugin |
| 9 | xf\_adsp\_get\_param | Get a single parameter to a registered plugin |
| 10 | xf\_adsp\_route | Route two registered plugins together |
| 11 | xf\_adsp\_renderer\_create | Register Renderer component |
| 12 | xf\_adsp\_renderer\_destroy | Unregister Renderer component |
| 13 | xf\_adsp\_renderer\_set\_params | Set parameters to Renderer plugin |
| 14 | xf\_adsp\_renderer\_get\_params | Get parameters from Renderer plugin |
| 15 | xf\_adsp\_capture\_create | Register Capture component |
| 16 | xf\_adsp\_capture\_destroy | Unregister Capture component |
| 17 | xf\_adsp\_capture\_set\_params | Set parameters to Capture plugin |
| 18 | xf\_adsp\_capture\_get\_params | Get parameters from Capture plugin |
| 19 | xf\_adsp\_equalizer\_create | Register Equalizer component |
| 20 | xf\_adsp\_equalizer\_destroy | Unregister Equalizer component |
| 21 | xf\_adsp\_equalizer\_set\_params | Set parameters to Equalizer plugin |
| 22 | xf\_adsp\_equalizer\_get\_params | Get parameters from Equalizer plugin |
| 23 | xf\_adsp\_tdm\_renderer\_create | Register TDM Renderer component |
| 24 | xf\_adsp\_tdm\_renderer\_destroy | Unregister TDM Renderer component |
| 25 | xf\_adsp\_tdm\_renderer\_set\_params | Set parameters to TDM Renderer plugin |
| 26 | xf\_adsp\_tdm\_renderer\_get\_params | Get parameters to TDM Renderer plugin |
| 27 | xf\_adsp\_tdm\_capture\_create | Register TDM Capture component |
| 28 | xf\_adsp\_tdm\_capture\_destroy | Unregister TDM Capture component |
| 29 | xf\_adsp\_tdm\_capture\_set\_params | Set parameters to TDM Capture plugin |
| 30 | xf\_adsp\_tdm\_capture\_get\_params | Get parameters to TDM Capture plugin |

Table 2‑2 List of APIs in ADSP base

## Structure

### xf\_callback\_func structure

|  |  |
| --- | --- |
| struct xf\_callback\_func | |
| int (\*empty\_buf\_done)(void \*data, int opcode, int length, char \*buffer) | Callback function for response message of XF\_EMPTY\_THIS\_BUFFER command |
| int (\*fill\_buf\_done)(void \*data, int opcode, int length, char \*buffer) | Callback function for response message of XF\_FILL\_THIS\_BUFFER command |
| int (\*event\_handler)(void \*data) | Callback for event handler |

### xf\_adsp\_base\_cmd structure

|  |  |
| --- | --- |
| struct xf\_adsp\_base\_cmd | |
| int (\*client\_register)(void \*\*private\_data) | Callback function for registering a new client from proxy driver for ADSP base control |
| int (\*client\_unregister)(void \*private\_data) | Callback function for freeing the registered client to proxy driver |
| int (\*recv)(void \*private\_data, void \*buf) | Callback function for getting the message from client message queue |
| int (\*send)(void \*private\_data, void \*buf) | Callback function for sending command to from ADSP base for ADSP transfer process to proxy driver |
| int (\*poll)(void \*private\_data, int \*condition) | Callback function to sleep and wait until there is a response message available from the client message queue or the condition flag becomes true. |

### xf\_handle structure

|  |  |
| --- | --- |
| struct xf\_handle | |
| int comp\_id | Component ID of handler that registered in ADSP |
| struct xf\_callback\_func \*cb | Callback functions to ADSP ALSA driver |
| void \*private\_data | A data in ADSP ALSA driver that is used as a parameter in the callback functions |

### xf\_adsp\_base structure

|  |  |
| --- | --- |
| struct xf\_adsp\_base | |
| struct xf\_adsp\_base\_cmd cmd | Struct of extension APIs for ADSP base |
| void \*client | Client data which registered on proxy driver |
| struct xf\_pool \*aux\_pool | Auxiliary pool data which registered to ADSP [[\*]](#_Structure) |
| struct xf\_handle \*handle[256] | Handle data which registered by ADSP ALSA driver |
| struct task\_struct \*rsp\_thread | Response thread of ADSP base [[\*]](#_Structure) |
| wait\_queue\_head\_t base\_wait | Wait queue for the response messages [[\*]](#_Structure) |
| struct xf\_message base\_msg | The message for stored the response message [[\*]](#_Structure) |
| int base\_flag | The flag for ADSP base waiting |
| int err\_flag | The flag to indicate an error from plugins |
| int wait\_flag | The flag for ADSP base polling waiting |
| spinlock\_t lock | Spinlock data |

### xf\_adsp\_renderer\_params structure

|  |  |
| --- | --- |
| struct xf\_adsp\_renderer\_params | |
| int channel | PCM channels |
| int pcm\_width | PCM width |
| int frame\_size | Size of a frame in sample |
| int in\_rate | Input sampling rate |
| int out\_rate | Output sampling rate |
| int vol\_rate | Volume rate |
| int dev1 | Index of first output device |
| int dev2 | Index of second output device |
| int dma1 | Index of first DMA channel |
| int dma2 | Index of second DMA channel |
| int out\_channel | Output PCM channels |
| int mix\_ctrl | Mix control flag |
| int state | Operation state control |
| int ring\_num | Total number of ring buffers |

### xf\_adsp\_renderer structure

|  |  |
| --- | --- |
| struct xf\_adsp\_renderer | |
| struct xf\_adsp\_renderer\_params params | Store the parameters of Renderer plugin |
| struct xf\_pool \*buf\_pool | Buffer pool data that used for PCM transfer |
| int handle\_id | Handle ID that Renderer has registered in ADSP base |

### xf\_adsp\_capture\_params structure

|  |  |
| --- | --- |
| struct xf\_adsp\_capture\_params | |
| int channel | PCM channels |
| int pcm\_width | PCM width |
| int frame\_size | Size of a frame in sample |
| int in\_rate | Input sampling rate |
| int out\_rate | Output sampling rate |
| int vol\_rate | Volume rate |
| int dev1 | Index of first input device |
| int dev2 | Index of second input device |
| int dma1 | Index of first DMA channel |
| int dma2 | Index of second DMA channel |
| int state | Operation state control |
| int ring\_num | Total number of ring buffers |

### xf\_adsp\_capture structure

|  |  |
| --- | --- |
| struct xf\_adsp\_capture | |
| struct xf\_adsp\_capture\_params params | Store the parameters of Capture plugin |
| struct xf\_pool \*buf\_pool | Buffer pool data that used for PCM transfer |
| int handle\_id | Handle ID that Capture has registered in ADSP base |

### xf\_equalizer\_parametric\_coef structure

|  |  |
| --- | --- |
| struct xf\_equalizer\_parametric\_coef | |
| int type[9] | Parametric filter type |
| int fc[9] | Parametric filter center frequency |
| int gain[9] | Parametric filter gain |
| int band\_width[9] | Parametric filter band width |
| int gain\_base[9] | Parametric filter gain base |

### xf\_equalizer\_graphic\_coef structure

|  |  |
| --- | --- |
| struct xf\_equalizer\_graphic\_coef | |
| int gain\_g[5] | Graphic filter gain |

* Note:

There are 9 filters for parametric equalizer type and 5 filters for graphic equalizer type.

### xf\_adsp\_equalizer\_params structure

|  |  |
| --- | --- |
| struct xf\_adsp\_equalizer\_params | |
| int channel | PCM channel |
| int pcm\_width | PCM width |
| int rate | Sampling rate |
| int eqz\_type | Equalizer type |
| struct xf\_equalizer\_parametric\_coef p\_coef | Parametric coefficients |
| struct xf\_equalizer\_graphic g\_coef | Graphic coefficient |

### xf\_adsp\_equalizer structure

|  |  |
| --- | --- |
| struct xf\_adsp\_equalizer | |
| struct xf\_adsp\_equalizer\_params params | Store the parameters of Equalizer plugin |
| struct xf\_pool \*buf\_pool | Buffer pool data that used for PCM transfer |
| int handle\_id | Handle ID that Equalizer has registered |

### xf\_adsp\_tdm\_renderer\_params structure

|  |  |
| --- | --- |
| struct xf\_adsp\_tdm\_renderer\_params | |
| int ch\_mode | Channel mode |
| int pcm\_width | PCM width |
| int frame\_size | Size of a frame in sample |
| int in\_rate | Input sampling rate |
| int out\_rate | Output sampling rate |
| int vol\_rate | Volume rate |
| int dev1 | Index of first output device |
| int dev2 | Index of second output device |
| int dma1 | Index of first DMA channel |
| int dma2 | Index of second DMA channel |

### xf\_adsp\_tdm\_renderer structure

|  |  |
| --- | --- |
| struct xf\_adsp\_tdm\_renderer | |
| struct xf\_adsp\_tdm\_renderer\_params params | Store the parameters of TDM Renderer plugin |
| struct xf\_pool \*buf\_pool | Buffer pool data that used for PCM transfer |
| int handle\_id | Handle ID that TDM Renderer has registered in ADSP base |

### xf\_adsp\_tdm\_capture\_params structure

|  |  |
| --- | --- |
| struct xf\_adsp\_tdm\_capture\_params | |
| int ch\_mode | Channel mode |
| int pcm\_width | PCM width |
| int frame\_size | Size of a frame in sample |
| int in\_rate | Input sampling rate |
| int out\_rate | Output sampling rate |
| int vol\_rate | Volume rate |
| int dev1 | Index of first input device |
| int dev2 | Index of second input device |
| int dma1 | Index of first DMA channel |
| int dma2 | Index of second DMA channel |

### xf\_adsp\_tdm\_capture structure

|  |  |
| --- | --- |
| struct xf\_adsp\_tdm\_capture | |
| struct xf\_adsp\_tdm\_capture\_params params | Store the parameters of TDM Capture plugin |
| struct xf\_pool \*buf\_pool | Buffer pool data that used for PCM transfer |
| int handle\_id | Handle ID that TDM Capture has registered in ADSP base |

## Interface APIs

### xf\_adsp\_base\_create

FD\_DRV\_CMN\_001

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_base\_create | | |
| Synopsis | This API creates and initializes ADSP base data that ADSP ALSA driver can use to control ADSP plugins (Renderer/Capture/Equalizer/TDM Capture/TDM Renderer). | |
| Syntax | int xf\_adsp\_base\_create(struct xf\_adsp\_base\_cmd \*cmd) | |
| Parameter | struct xf\_adsp\_base\_cmd \*cmd | Pointer to struct of commands for extension interface |
| Return value | 0 | ADSP base registers successful. |
| -EINVAL | ADSP base has been alive.  Parameter cmd is invalid. |
| -ENOMEM | Cannot allocate memory for ADSP base usage. |
| -EBUSY | ADSP base cannot register to proxy driver due to number of client has been exceeded. |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

### xf\_adsp\_base\_destroy

FD\_DRV\_CMN\_002

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_base\_destroy | | |
| Synopsis | This API frees all resources that ADSP base has registered. | |
| Syntax | int xf\_adsp\_base\_destroy (void) | |
| Parameter | - | - |
| Return value | 0 | ADSP base destroy successful. |
| -EINVAL | ADSP base has not been registered yet. |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

### xf\_adsp\_empty\_this\_buffer

FD\_DRV\_CMN\_003

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_empty\_this\_buffer | | |
| Synopsis | This API sends a XF\_EMPTY\_THIS\_BUFFER command to ADSP framework. | |
| Syntax | int xf\_adsp\_empty\_this\_buffer(int handle\_id, char \*buffer, int length) | |
| Parameter | int handle\_id | Handle ID that registered in ADSP base |
| char \*buffer | Pointer to PCM buffer |
| int length | Size of buffer in byte |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or the handle ID has not registered to ADSP base yet. |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

### xf\_adsp\_fill\_this\_buffer

FD\_DRV\_CMN\_004

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_fill\_this\_buffer | | |
| Synopsis | This API sends a XF\_FILL\_THIS\_BUFFER command to ADSP framework. | |
| Syntax | int xf\_adsp\_fill\_this\_buffer(int handle\_id, char \*buffer, int length) | |
| Parameter | int handle\_id | Handle ID that registered in ADSP base |
| char \*buffer | Pointer to PCM buffer |
| int length | Size of buffer in byte |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or the handle ID has not registered to ADSP base yet. |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

### xf\_adsp\_mmap\_this\_buffer

FD\_DRV\_CMN\_036

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_mmap\_this\_buffer | | |
| Synopsis | This API sends a XF\_MMAP\_THIS\_BUFFER command to ADSP framework. | |
| Syntax | int xf\_adsp\_mmap\_this\_buffer(int handle\_id, char \*buffer, int length) | |
| Parameter | int handle\_id | Handle ID that registered in ADSP base |
| char \*buffer | Pointer to PCM buffer |
| int length | Size of buffer in byte |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or the handle ID has not registered to ADSP base yet.  Mapping buffer in ADSP is failed |

[Covers: RD\_015]

### xf\_adsp\_allocate\_mem\_pool

FD\_DRV\_CMN\_005

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_allocate\_mem\_pool | | |
| Synopsis | This API sends a XF\_ALLOC to ADSP to request a memory pool with desired pool size and buffer length. | |
| Syntax | struct xf\_pool \*xf\_adsp\_allocate\_mem\_pool(int pool\_size, int buf\_length) | |
| Parameter | int pool\_size | Number of buffer need to allocate from ADSP |
| int buf\_length | Size of buffer need to allocate from ADSP |
| Return value | Pointer | Pointer to registered memory pool |
| -EINVAL | ADSP base has not registered yet. |
| -ENOMEM | Cannot allocate memory for pool  Cannot allocate buffers in ADSP |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

### xf\_adsp\_free\_mem\_pool

FD\_DRV\_CMN\_006

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_free\_mem\_pool | | |
| Synopsis | This API frees memory pool and sends a XF\_FREE to ADSP to return the previous registered buffers. | |
| Syntax | int xf\_adsp\_free\_mem\_pool(struct xf\_pool \*pool) | |
| Parameter | struct xf\_pool \*pool | Pointer to allocated memory pool |
| Return value | 0 | Pool has been free successfully. |
| -EINVAL | ADSP base has not registered yet, or memory pool is invalid. |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

### xf\_adsp\_get\_data\_from\_pool

FD\_DRV\_CMN\_007

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_get\_data\_from\_pool | | |
| Synopsis | This API gets a data buffer from pool, which registered before. | |
| Syntax | char \*xf\_adsp\_get\_data\_from\_pool(struct xf\_pool \*pool, int index) | |
| Parameter | struct xf\_pool \*pool | Pointer to allocated memory pool |
| int index | The index of buffer in pool |
| Return value | Pointer | Buffer address of pool |
| -EINVAL | Memory pool is invalid or, the index is over than number of buffer in pool. |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

### xf\_adsp\_set\_param

FD\_DRV\_CMN\_008

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_set\_param | | |
| Synopsis | This API sets a single parameter to a registered plugin. | |
| Syntax | int xf\_adsp\_set\_param(int handle\_id, int index, int value) | |
| Parameter | int handle\_id | Handle ID that registered in ADSP base |
| int index | The sub-command index of the setting value |
| int value | The setting value |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or the handle ID has not registered to ADSP base yet, or the setting command makes a fatal error from ADSP plugin. |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

### xf\_adsp\_get\_param

FD\_DRV\_CMN\_009

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_get\_param | | |
| Synopsis | This API gets a single parameter to a registered plugin. | |
| Syntax | int xf\_adsp\_get\_param(int handle\_id, int index, int \*value) | |
| Parameter | int handle\_id | Handle ID that registered in ADSP base |
| int index | The sub-command index of the getting value |
| int \*value | Pointer of the stored getting value |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or the handle ID has not registered to ADSP base yet, or the pointer of value is invalid, or the getting command makes a fatal error from ADSP plugin. |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

### xf\_adsp\_route

FD\_DRV\_CMN\_010

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_route | | |
| Synopsis | This API sends a XF\_ROUTE command to ADSP framework to register a tunnel for transfer data between two ADSP plugins. | |
| Syntax | int xf\_adsp\_route(int src\_handle\_id, int dst\_handle\_id, int buf\_cnt, int buf\_size) | |
| Parameter | int src\_handle\_id | Handle ID of source component |
| int dst\_handle\_id | Handle ID of destination component |
| int buf\_cnt | Number of buffer that used for tunnel |
| int buf\_size | Size of a buffer that used for tunnel |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or the src/dst handle ID have not registered to ADSP base yet, or the tunnel request cannot complete from ADSP. |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

### xf\_adsp\_renderer\_create

FD\_DRV\_CMN\_011

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_renderer\_create | | |
| Synopsis | This API initializes a Renderer instance, registers ADSP Renderer plugin. After Renderer has registered successful, the API registers a handler to ADSP base, and get a handle ID which represents for a new handler which has been registered completely. Finally, it initializes all parameters as default values with default values of ADSP Renderer plugin. | |
| Syntax | int xf\_adsp\_renderer\_create(  struct xf\_adsp\_renderer \*\*renderer, struct xf\_callback\_func \*cb,  void \*private\_data) | |
| Parameter | struct xf\_adsp\_renderer \*\*renderer | The pointer to store the created Renderer instance |
| struct xf\_callback\_func \*cb | Pointer to the callback functions |
| void \*private\_data | Pointer to a private data that used as a parameter in callback functions |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or cannot register new handler to ADSP base, or cannot register Renderer plugin to ADSP, or cannot get default value from ADSP Renderer plugin. |
| -ENOMEM | Cannot allocate Renderer instance |

[Covers: RD\_002]

### xf\_adsp\_renderer\_destroy

FD\_DRV\_CMN\_012

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_renderer\_destroy | | |
| Synopsis | This API unregisters ADSP Renderer plugin, frees registered handler and instance. | |
| Syntax | int xf\_adsp\_renderer\_destroy(struct xf\_adsp\_renderer \*renderer) | |
| Parameter | struct xf\_adsp\_renderer \*renderer | Pointer to Renderer instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or Renderer instance is invalid. |

[Covers: RD\_002]

### xf\_adsp\_renderer\_set\_params

FD\_DRV\_CMN\_013

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_renderer\_set\_params | | |
| Synopsis | This API sets all parameters for ADSP Renderer plugin based on the values in params structure of Renderer instance. | |
| Syntax | int xf\_adsp\_renderer\_set\_params(struct xf\_adsp\_renderer \*renderer) | |
| Parameter | struct xf\_adsp\_renderer \*renderer | Pointer to Renderer instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or Renderer instance is invalid, or Renderer instance has not register to ADSP base yet, or the setting command make a fatal error from ADSP Renderer plugin. |

[Covers: RD\_002]

### xf\_adsp\_renderer\_get\_params

FD\_DRV\_CMN\_014

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_renderer\_get\_params | | |
| Synopsis | This API gets all ADSP Renderer’s parameters and stores the returned values in params structure of Renderer instance. | |
| Syntax | int xf\_adsp\_renderer\_get\_params(struct xf\_adsp\_renderer \*renderer) | |
| Parameter | struct xf\_adsp\_renderer \*renderer | Pointer to Renderer instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or Renderer instance is invalid, or Renderer instance has not register to ADSP base yet, or the getting command make a fatal error from ADSP Renderer plugin. |

[Covers: RD\_002]

### xf\_adsp\_capture\_create

FD\_DRV\_CMN\_015

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_capture\_create | | |
| Synopsis | This API initializes a Capture instance, registers ADSP Capture plugin. After Capture has registered successful, the API registers a handler to ADSP base, and get a handle ID which represents for a new handler which has been registered completely. Finally, it initializes all parameters as default values with default values of ADSP Capture plugin. | |
| Syntax | int xf\_adsp\_capture\_create(  struct xf\_adsp\_capture \*\*capture, struct xf\_callback\_func \*cb,  void \*private\_data) | |
| Parameter | struct xf\_adsp\_capture \*\*capture | The pointer to store the created Capture instance |
| struct xf\_callback\_func \*cb | Pointer to the callback functions |
| void \*private\_data | Pointer to a private data that used as a parameter in callback functions |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet or cannot register new handler to ADSP base, or cannot register Capture plugin to ADSP, or cannot get default values from ADSP Capture plugin. |
| -ENOMEM | Cannot allocate Capture instance |

[Covers: RD\_003]

### xf\_adsp\_capture\_destroy

FD\_DRV\_CMN\_016

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_capture\_destroy | | |
| Synopsis | This API unregisters ADSP Capture plugin, frees registered handler and instance. | |
| Syntax | int xf\_adsp\_capture\_destroy(struct xf\_adsp\_capture \*capture) | |
| Parameter | struct xf\_adsp\_capture \*capture | Pointer to Capture instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or Capture instance is invalid. |

[Covers: RD\_003]

### xf\_adsp\_capture\_set\_params

FD\_DRV\_CMN\_017

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_capture\_set\_params | | |
| Synopsis | This API sets all parameters for ADSP Capture plugin based on the values in params structure of Capture instance. | |
| Syntax | int xf\_adsp\_capture\_set\_params(struct xf\_adsp\_capture \*capture) | |
| Parameter | struct xf\_adsp\_capture \*capture | Pointer to Capture instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or Capture instance is invalid, or Capture instance has not register to ADSP base yet, or the setting command make a fatal error from ADSP Capture plugin. |

[Covers: RD\_003]

### xf\_adsp\_capture\_get\_params

FD\_DRV\_CMN\_018

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_capture\_get\_params | | |
| Synopsis | This API gets all ADSP Capture’s parameters and stores the returned values in params structure of Capture instance. | |
| Syntax | int xf\_adsp\_capture\_get\_params(struct xf\_adsp\_capture \*capture) | |
| Parameter | struct xf\_adsp\_capture \*capture | Pointer to Capture instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or Capture instance is invalid, or Capture instance has not register to ADSP base yet, or the getting command make a fatal error from ADSP Capture plugin. |

[Covers: RD\_003]

### xf\_adsp\_equalizer\_create

FD\_DRV\_CMN\_019

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_equalizer\_create | | |
| Synopsis | This API initializes an Equalizer instance, registers ADSP Equalizer plugin. After Equalizer has registered successful, the API registers a handler to ADSP base, and get a handle ID which represents for a new handler which has been registered completely. Finally, it initializes all parameters as default values with defaul values of ADSP Equalizer plugin. | |
| Syntax | int xf\_adsp\_equalizer\_create(  struct xf\_adsp\_equalizer \*\*equalizer, struct xf\_callback\_func \*cb,  void \*private\_data) | |
| Parameter | struct xf\_adsp\_equalizer \*\*equalizer | The pointer to store the created Equalizer instance |
| struct xf\_callback\_func \*cb | Pointer to the callback functions |
| void \*private\_data | Pointer to a private data that used as a parameter in callback functions |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet or cannot register new handler to ADSP base, or cannot register Equalizer plugin to ADSP, or cannot get default values from ADSP Equalizer plugin. |
| -ENOMEM | Cannot allocate Equalizer instance |

[Covers: RD\_005]

### xf\_adsp\_equalizer\_destroy

FD\_DRV\_CMN\_020

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_equalizer\_destroy | | |
| Synopsis | This API unregisters ADSP Equalizer plugin, frees registered handler and instance. | |
| Syntax | int xf\_adsp\_equalizer\_destroy(struct xf\_adsp\_equalizer \*equalizer) | |
| Parameter | struct xf\_adsp\_equalizer \*equalizer | Pointer to Equalizer instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or Equalizer instance is invalid. |

[Covers: RD\_005]

### xf\_adsp\_equalizer\_set\_params

FD\_DRV\_CMN\_021

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_equalizer\_set\_params | | |
| Synopsis | This API sets all parameters for ADSP Equalizer plugin based on the values in params structure of Equalizer instance. | |
| Syntax | int xf\_adsp\_equalizer\_set\_params(struct xf\_adsp\_equalizer \*equalizer) | |
| Parameter | struct xf\_adsp\_equalizer \*equalizer | Pointer to Equalizer instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or Equalizer instance is invalid, or Equalizer instance has not register to ADSP base yet, or the setting command makes a fatal error from ADSP Equalizer plugin. |

[Covers: RD\_005]

### xf\_adsp\_equalizer\_get\_params

FD\_DRV\_CMN\_022

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_equalizer\_get\_params | | |
| Synopsis | This API gets all ADSP Equalizer’s parameters and stores the returned values in params structure of Equalizer instance. | |
| Syntax | int xf\_adsp\_equalizer\_get\_params(struct xf\_adsp\_equalizer \*equalizer) | |
| Parameter | struct xf\_adsp\_equalizer \*equalizer | Pointer to Equalizer instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or Equalizer instance is invalid, or Equalizer instance has not register to ADSP base yet, or the getting command makes a fatal error from ADSP Equalizer plugin. |

[Covers: RD\_005]

### xf\_adsp\_tdm\_renderer\_create

FD\_DRV\_CMN\_023

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_tdm\_renderer\_create | | |
| Synopsis | This API initializes a TDM Renderer instance, registers ADSP TDM Renderer plugin. After TDM Renderer has been registered successfully, the API registers a handler to ADSP base, and gets a handle ID which represents for a new handler which has been registered completely. Finally, it initializes all parameters as default values with default values of TDM Renderer plugin. | |
| Syntax | int xf\_adsp\_tdm\_renderer\_create(  struct xf\_adsp\_tdm\_renderer \*\*tdm\_renderer, struct xf\_callback\_func \*cb,  void \*private\_data) | |
| Parameter | struct xf\_adsp\_tdm\_renderer \*\*tdm\_renderer | The pointer to store the created TDM Renderer instance |
| struct xf\_callback\_func \*cb | Pointer to the callback functions |
| void \*private\_data | Pointer to a private data that used as a parameter in callback functions |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or cannot register new handler to ADSP base, or cannot register TDM Renderer plugin to ADSP, or cannot get default values from ADSP TDM Renderer plugin. |
| -ENOMEM | Cannot allocate TDM Renderer instance |

[Covers: RD\_004]

### xf\_adsp\_tdm\_renderer\_destroy

FD\_DRV\_CMN\_024

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_tdm\_renderer\_destroy | | |
| Synopsis | This API unregisters ADSP TDM Renderer plugin, frees registered handler and instance. | |
| Syntax | int xf\_adsp\_tdm\_renderer\_destroy(  struct xf\_adsp\_tdm\_renderer \*tdm\_renderer) | |
| Parameter | struct xf\_adsp\_tdm\_renderer \*tdm\_renderer | Pointer to TDM Renderer instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or TDM Renderer instance is invalid. |

[Covers: RD\_004]

### xf\_adsp\_tdm\_renderer\_set\_params

FD\_DRV\_CMN\_025

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_tdm\_renderer\_set\_params | | |
| Synopsis | This API sets all parameters for ADSP TDM Renderer plugin based on the values in params structure of TDM Renderer instance. | |
| Syntax | int xf\_adsp\_tdm\_renderer\_set\_params(struct xf\_adsp\_tdm\_renderer \*tdm\_renderer) | |
| Parameter | struct xf\_adsp\_tdm\_renderer \*tdm\_renderer | Pointer to TDM Renderer instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or TDM Renderer instance is invalid, or TDM Renderer instance has not register to ADSP base yet, or the setting command make a fatal error from ADSP TDM Renderer plugin. |

[Covers: RD\_004]

### xf\_adsp\_tdm\_renderer\_get\_params

FD\_DRV\_CMN\_026

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_tdm\_renderer\_get\_params | | |
| Synopsis | This API gets all ADSP TDM Renderer’s parameters and stores the returned values in params structure of TDM Renderer instance. | |
| Syntax | int xf\_adsp\_tdm\_renderer\_get\_params(struct xf\_adsp\_tdm\_renderer \*tdm\_renderer) | |
| Parameter | struct xf\_adsp\_tdm\_renderer \*tdm\_renderer | Pointer to TDM Renderer instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or TDM Renderer instance is invalid, or TDM Renderer instance has not register to ADSP base yet, or the getting command make a fatal error from ADSP TDM Renderer plugin. |

[Covers: RD\_004]

### xf\_adsp\_tdm\_capture\_create

FD\_DRV\_CMN\_027

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_tdm\_capture\_create | | |
| Synopsis | This API initializes a TDM Capture instance, registers ADSP TDM Capture plugin. After TDM Capture has been registered successfully, the API registers a handler to ADSP base, and get a handle ID which represents for a new handler has been registered completely. Finally, it initializes all parameters as default values with default values of the plugin. | |
| Syntax | int xf\_adsp\_tdm\_capture\_create(  struct xf\_adsp\_tdm\_capture \*\*tdm\_capture, struct xf\_callback\_func \*cb,  void \*private\_data) | |
| Parameter | struct xf\_adsp\_tdm\_capture \*\*tdm\_capture | The pointer to store the created TDM Capture instance |
| struct xf\_callback\_func \*cb | Pointer to the callback functions |
| void \*private\_data | Pointer to a private data that used as a parameter in callback functions |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet or cannot register new handler to ADSP base, or cannot register TDM Capture plugin to ADSP, or cannot get default values from ADSP TDM Capture plugin. |
| -ENOMEM | Cannot allocate TDM Capture instance |

[Covers: RD\_004]

### xf\_adsp\_tdm\_capture\_destroy

FD\_DRV\_CMN\_028

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_tdm\_capture\_destroy | | |
| Synopsis | This API unregisters ADSP TDM Capture plugin, frees registered handler and instance. | |
| Syntax | int xf\_adsp\_tdm\_capture\_destroy(struct xf\_adsp\_tdm\_capture \*tdm\_capture) | |
| Parameter | struct xf\_adsp\_tdm\_capture \*tdm\_capture | Pointer to TDM Capture instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or TDM Capture instance is invalid. |

[Covers: RD\_004]

### xf\_adsp\_tdm\_capture\_set\_params

FD\_DRV\_CMN\_029

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_tdm\_capture\_set\_params | | |
| Synopsis | This API sets all parameters for ADSP TDM Capture plugin based on the values in params structure of TDM Capture instance. | |
| Syntax | int xf\_adsp\_tdm\_capture\_set\_params(struct xf\_adsp\_tdm\_capture \*tdm\_capture) | |
| Parameter | struct xf\_adsp\_tdm\_capture \*tdm\_capture | Pointer to TDM Capture instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or TDM Capture instance is invalid, or TDM Capture instance has not register to ADSP base yet, or the setting command make a fatal error from ADSP TDM Capture plugin. |

[Covers: RD\_004]

### xf\_adsp\_tdm\_capture\_get\_params

FD\_DRV\_CMN\_030

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_tdm\_capture\_get\_params | | |
| Synopsis | This API gets all ADSP TDM Capture’s parameters and stores the returned values in params structure of TDM Capture instance. | |
| Syntax | int xf\_adsp\_tdm\_capture\_get\_params(struct xf\_adsp\_tdm\_capture \*tdm\_capture) | |
| Parameter | struct xf\_adsp\_tdm\_capture \*tdm\_capture | Pointer to TDM Capture instance |
| Return value | 0 | Success |
| -EINVAL | ADSP base has not registered yet, or TDM Capture instance is invalid, or TDM Capture instance has not register to ADSP base yet, or the getting command make a fatal error from ADSP TDM Capture plugin. |

[Covers: RD\_004]

# Proxy Driver Interface - Extension APIs for ADSP Base

The below table shows list of extension APIs for proxy driver.

|  |  |  |
| --- | --- | --- |
| No. | APIs | Descriptions |
| 1 | xf\_adsp\_base\_client\_register | Register a proxy client for ADSP base |
| 2 | xf\_adsp\_base\_client\_unregister | Unregister the given client of ADSP base |
| 3 | xf\_adsp\_base\_send | Send a command message to proxy driver |
| 4 | xf\_adsp\_base\_recv | Receive a response message from proxy driver |
| 5 | xf\_adsp\_base\_poll | Wait for the response message from proxy driver |

Table 3‑1 List of extension APIs for proxy driver

## xf\_adsp\_base\_client\_register

FD\_DRV\_CMN\_031

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_base\_client\_register | | |
| Synopsis | This API registers a new client from proxy driver for ADSP base control. | |
| Syntax | int xf\_adsp\_base\_client\_register(void \*\*private\_data) | |
| Parameter | void \*\*private\_data | The pointer to store the registered proxy client. |
| Return value | 0 | Success |
| -ENOMEM | Cannot allocate memory for client |
| -EBUSY | Number of client in proxy has exceeded. |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

## xf\_adsp\_base\_client\_unregister

FD\_DRV\_CMN\_032

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_base\_client\_unregister | | |
| Synopsis | This API frees the registered client to proxy driver. | |
| Syntax | int xf\_adsp\_base\_client\_unregister(void \*private\_data) | |
| Parameter | void \*private\_data | Pointer to registered client of ADSP base |
| Return value | 0 | Success |
| -EINVAL | Client data is invalid. |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

## xf\_adsp\_base\_send

FD\_DRV\_CMN\_033

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_base\_send | | |
| Synopsis | This API sends a command message from ADSP base for ADSP transfer process to proxy driver. | |
| Syntax | int xf\_adsp\_base\_send(void \*private\_data, void \*buf) | |
| Parameter | void \*private\_data | Pointer to registered client of ADSP base |
| void \*buf | Pointer to command message data |
| Return value | 0 | Success |
| -EINVAL | Client data is invalid, or it has not registered to proxy driver yet, or cannot send message to proxy |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

## xf\_adsp\_base\_recv

FD\_DRV\_CMN\_034

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_base\_recv | | |
| Synopsis | This API gets the message from client message queue | |
| Syntax | int xf\_adsp\_base\_recv (void \*private\_data, void \*buf) | |
| Parameter | void \*private\_data | Pointer to registered client of ADSP base |
| void \*buf | Pointer to the stored message data |
| Return value | 0 | Success |
| -EINVAL | Client data is invalid, or it has not registered to proxy driver yet, or the response message is invalid. |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

## xf\_adsp\_base\_poll

FD\_DRV\_CMN\_035

|  |  |  |
| --- | --- | --- |
| xf\_adsp\_base\_poll | | |
| Synopsis | This API sleeps and waits until there are a response message is available from client message queue or the condition flag becomes true. | |
| Syntax | int xf\_adsp\_base\_poll (void \*private\_data, int \*condition) | |
| Parameter | void \*private\_data | Pointer to registered client of ADSP base |
| int \*condition | Pointer to a waiting condition flag  Valid value:  0: wait for the valid response message from client queue  1: cancel the waiting |
| Return value | 0 | Responded message is available. |
| 1 | Responded message is not available. |
| -EINVAL | Client or condition data is invalid, or the value of condition is not supported, or the waiting of event is interrupted. |

[Covers: RD\_002, RD\_003, RD\_004, RD\_005]

# Processing Flow

This part shows the processing flow of ADSP base, Renderer, Capture, and Equalizer component.

## ADSP Base Flow

### ADSP Base Creation

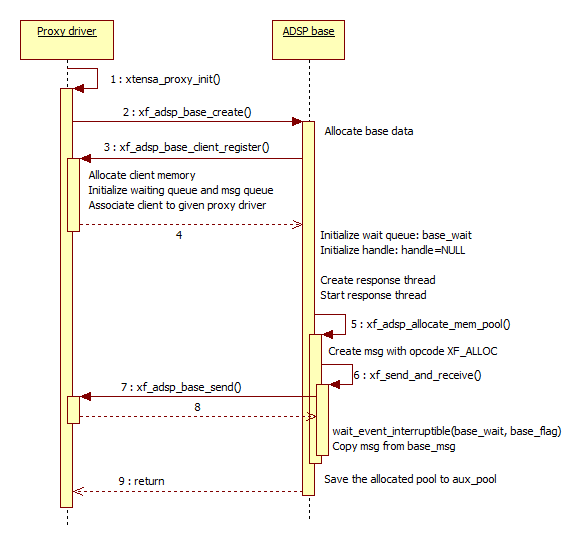


Figure 4‑1 ADSP base creation flow chart

### ADSP Base Destruction

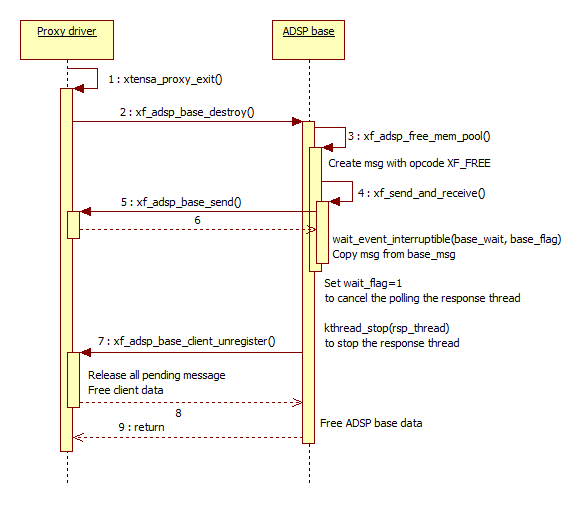


Figure 4‑2 ADSP base destruction flow chart

## Renderer Flow

### Renderer Creation

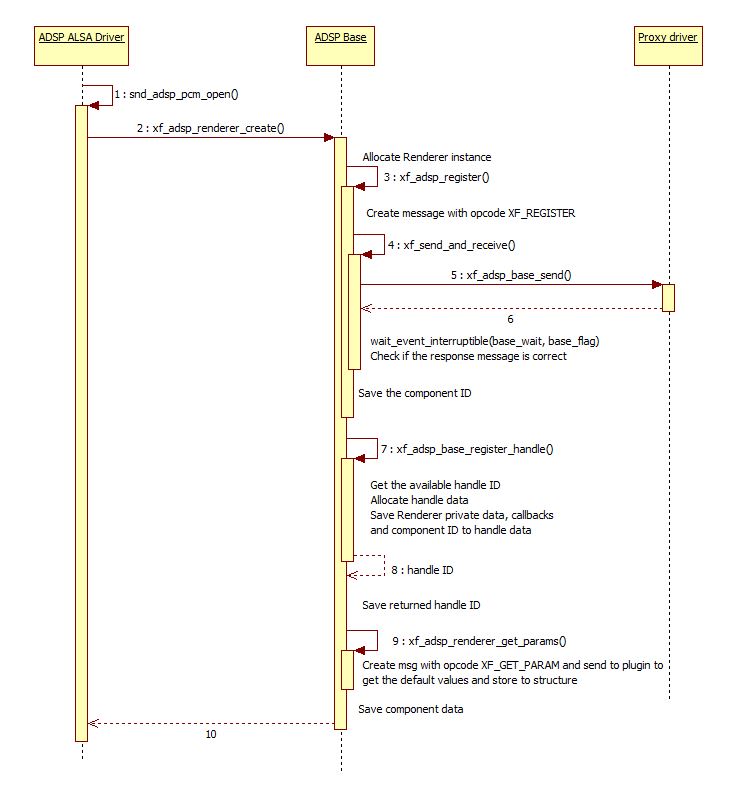


Figure 4‑3 Renderer creation flow chart

### Renderer Execution

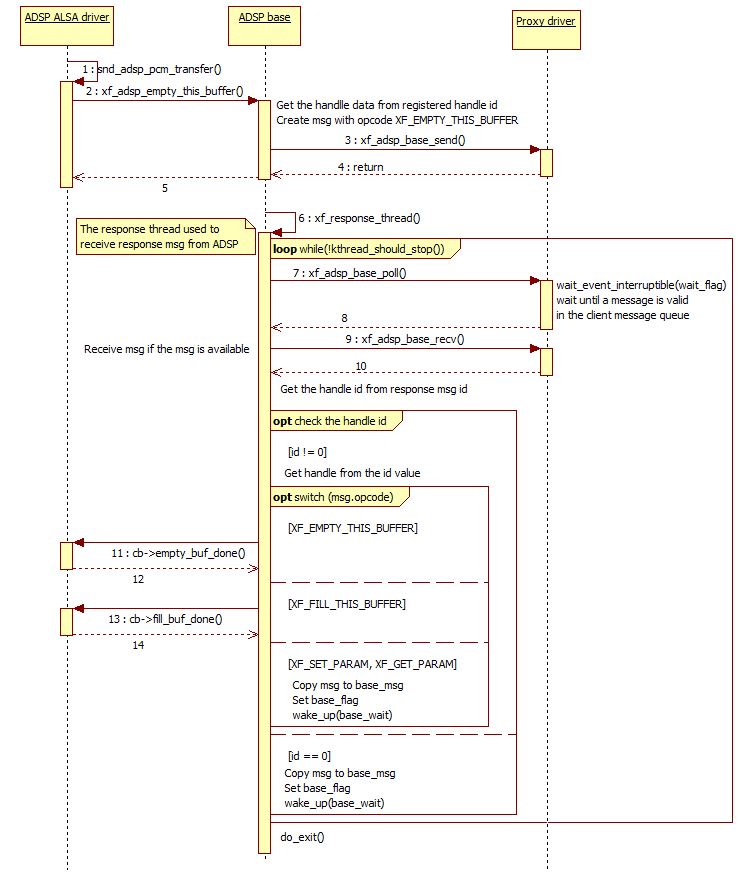


Figure 4‑4 Renderer execution flow chart

### Renderer Destruction

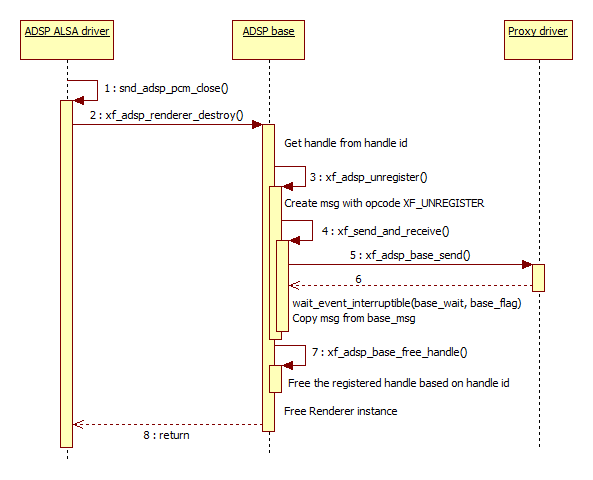


Figure 4‑5 Renderer destruction flow chart

## Capture Flow

### Capture Creation

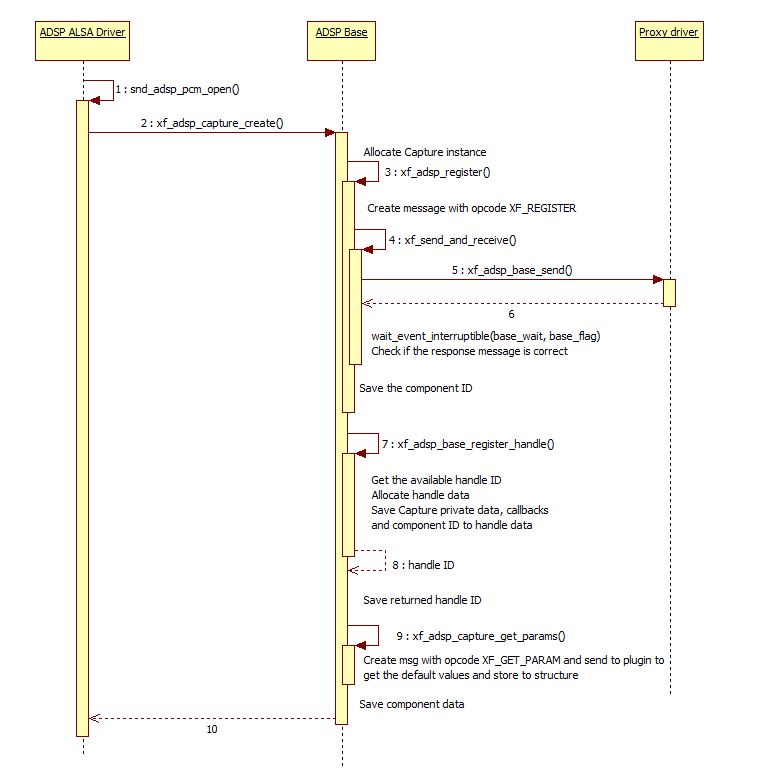


Figure 4‑6 Capture creation flow chart

### Capture Execution

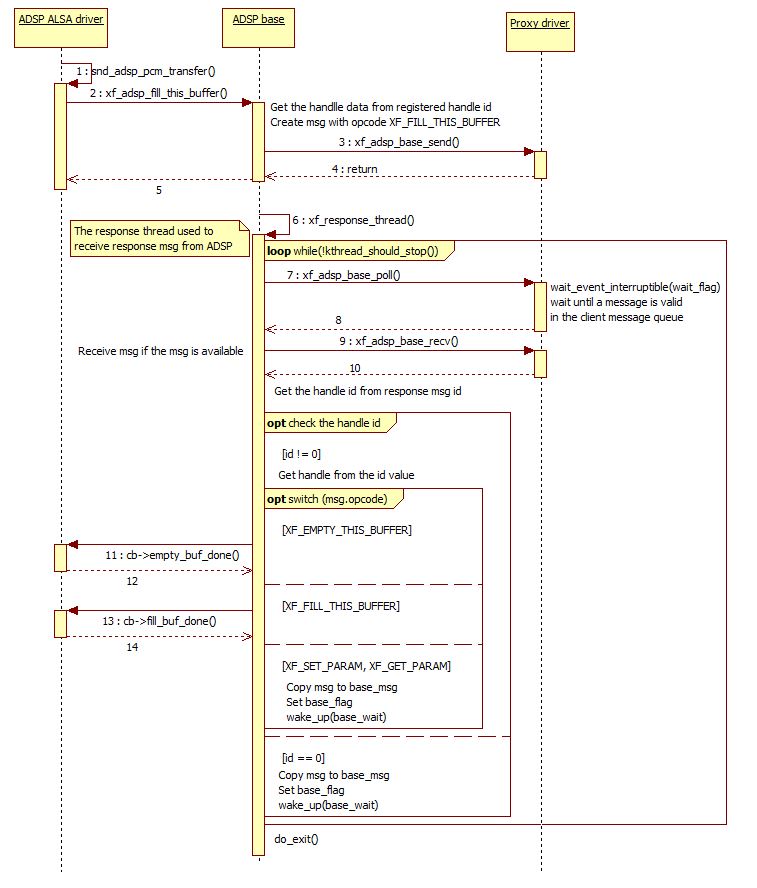


Figure 4‑7 Capture execution flow chart

### Capture Destruction

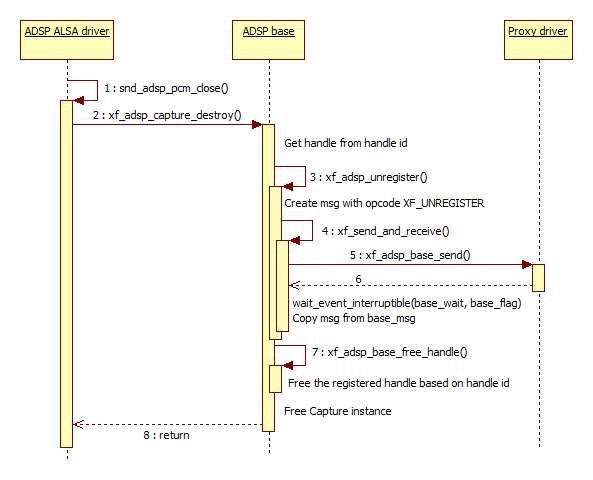


Figure 4‑8 Capture destruction flow chart

## Equalizer Flow

### Equalizer Creation

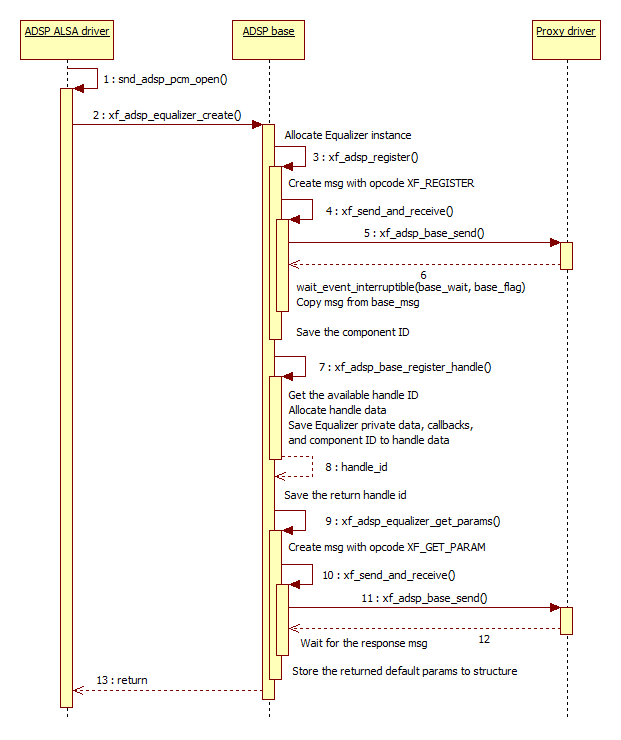


Figure 4‑9 Equalizer creation flow chart

### Equalizer Execution

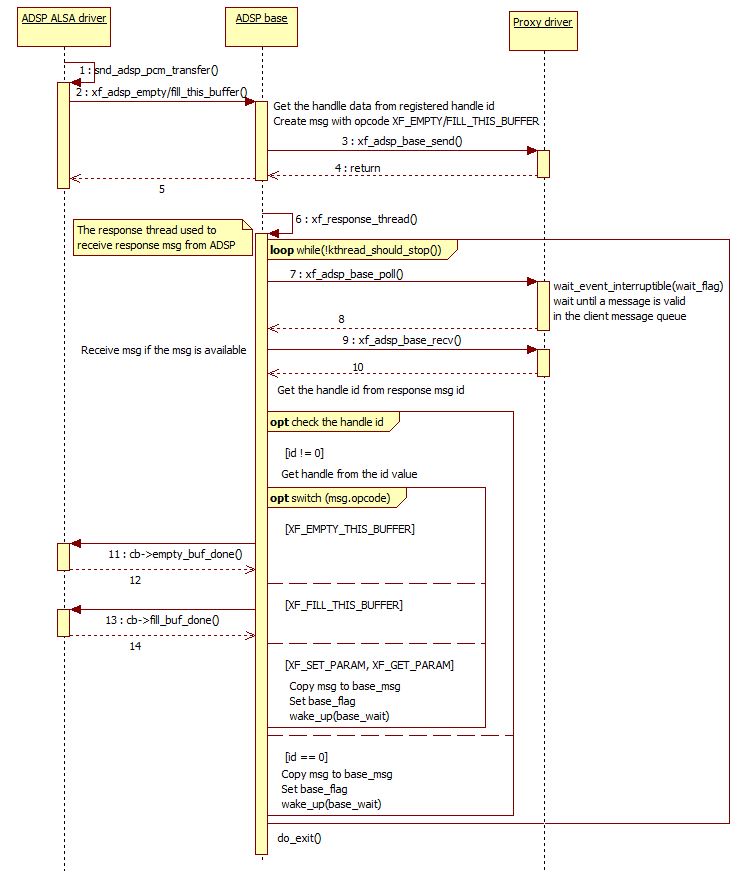


Figure 4‑10 Equalizer execution flow chart

### Equalizer Destruction

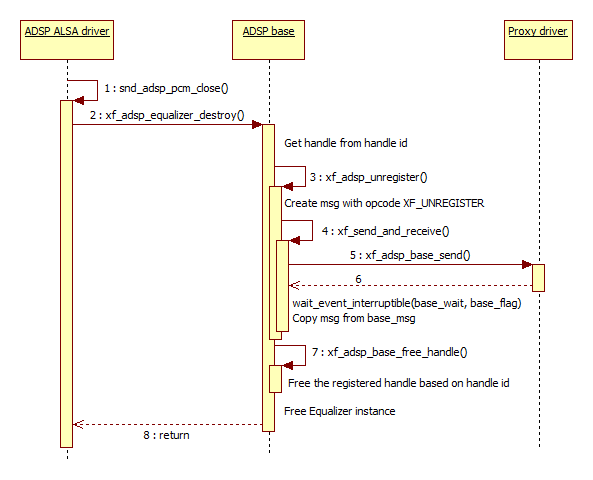


Figure 4‑11 Equalizer destruction flow chart

## TDM Renderer Flow

### TDM Renderer Creation

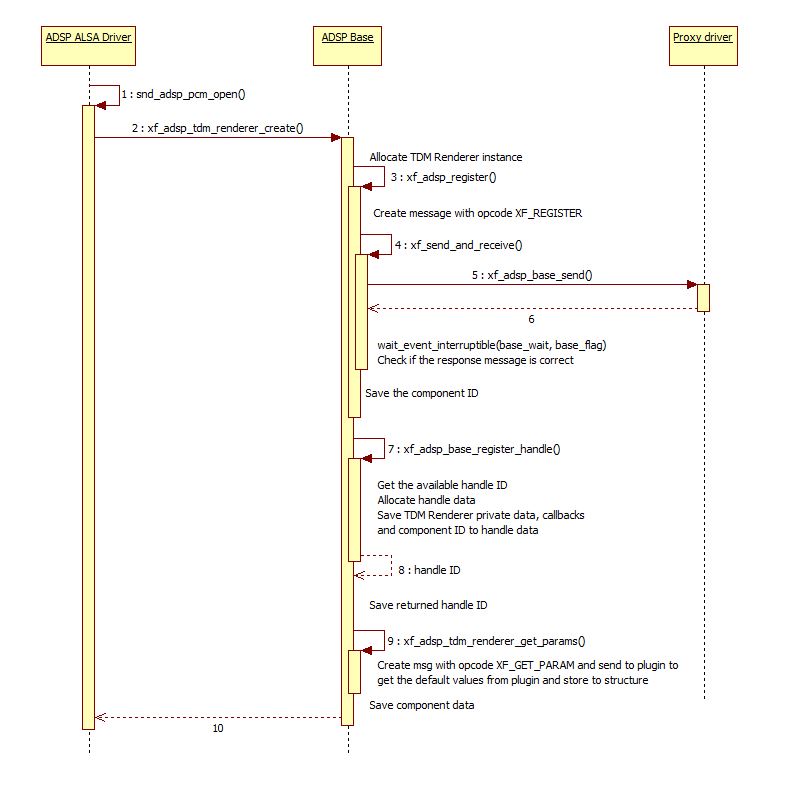


Figure 4‑12 TDM Renderer creation flow chart

### TDM Renderer Execution

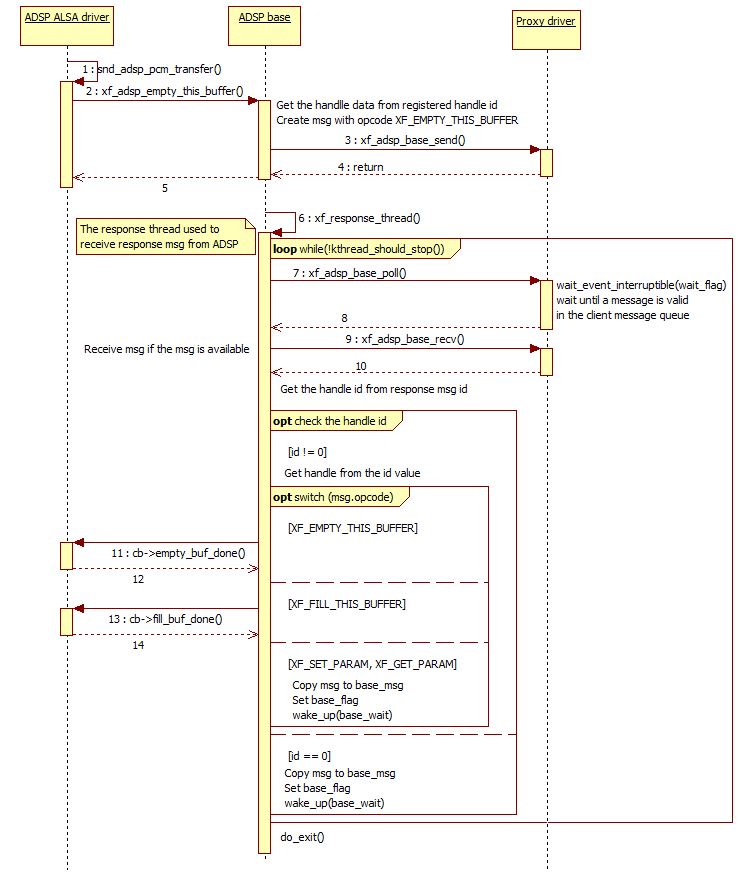


Figure 4‑13 TDM Renderer execution flow chart

### TDM Renderer Destruction

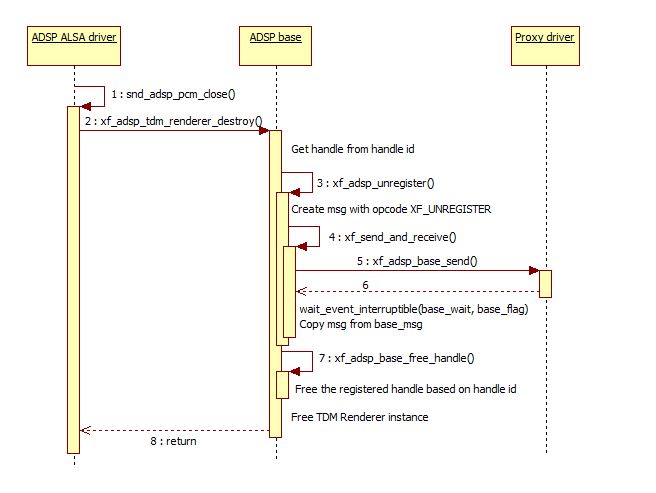


Figure 4‑14 TDM Renderer destruction flow chart

## TDM Capture Flow

### TDM Capture Creation

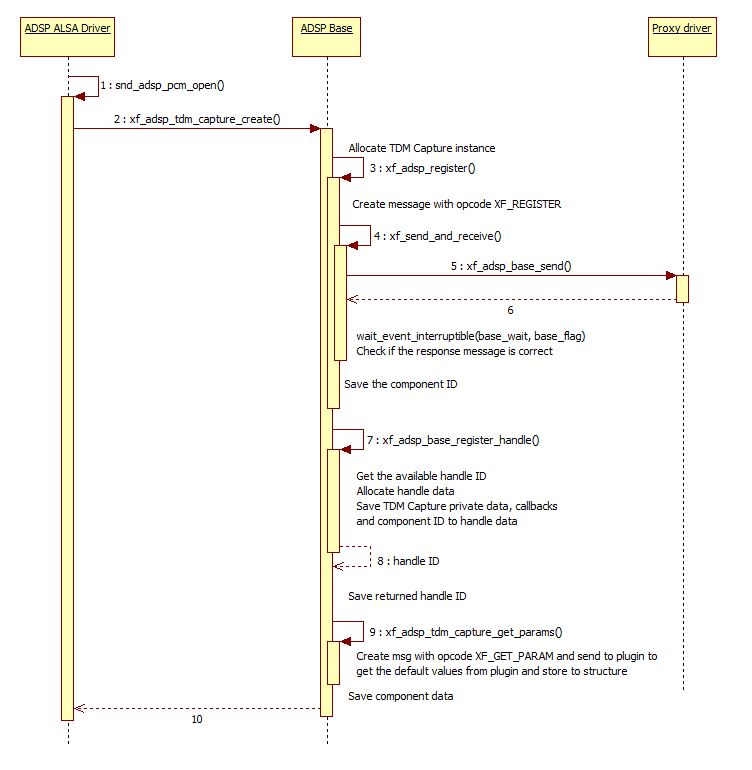


Figure 4‑15 TDM Capture creation flow chart

### TDM Capture Execution

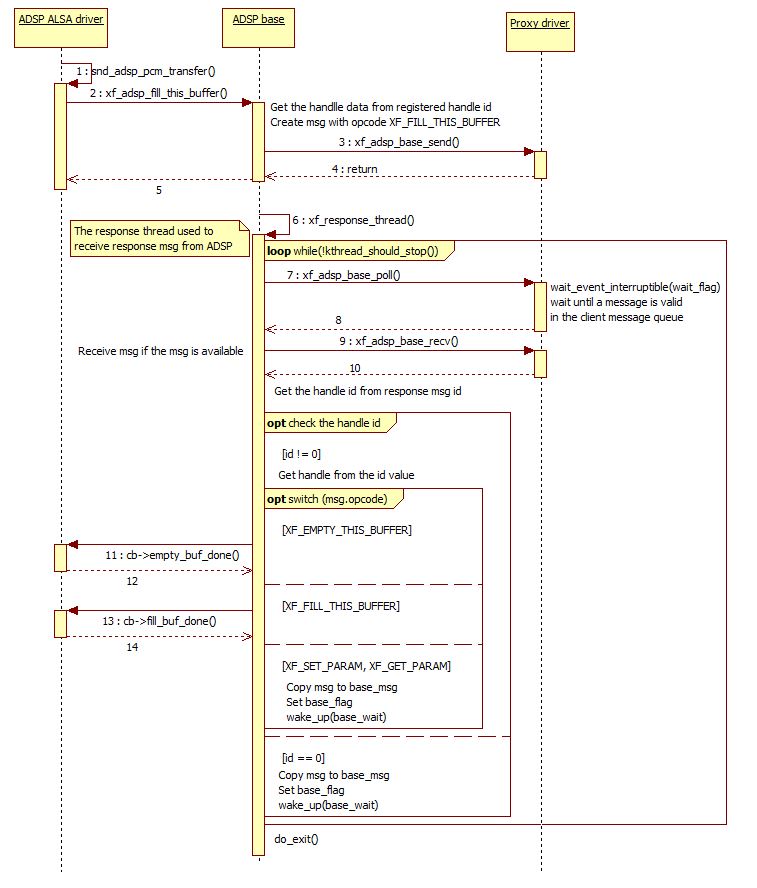


Figure 4‑16 TDM Capture execution flow chart

### TDM Capture Destruction

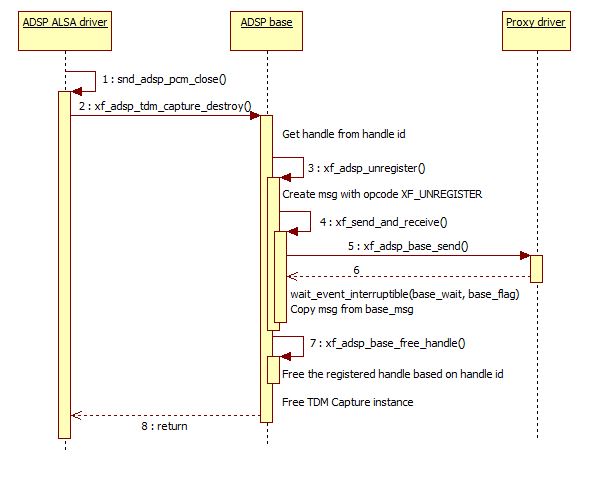


Figure 4‑17 TDM Capture destruction flow chart

# Appendix

## Error Code

|  |  |  |
| --- | --- | --- |
| **Error code** | **Description** | **Reference** |
| ENOMEM | Out of memory | https://elixir.free-electrons.com/linux/v4.0/source/include/uapi/asm-generic/errno-base.h |
| EINVAL | Invalid argument |
| ENODEV | No such device |
| EBUSY | Device or resource busy |

Table 5‑1 Generic error code definition

## Structures

|  |  |  |
| --- | --- | --- |
| **Structure** | **Description** | **Reference** |
| struct xf\_pool | Data pool type | Defined in the ADSP Interface for Linux document |
| struct xf\_message | Message type | Defined in the ADSP framework document |
| struct task\_struct | Task process structure | http://elixir.free-electrons.com/linux/latest/source/include/linux/sched.h |
| wait\_queue\_head\_t | Waiting queue structure | https://elixir.free-electrons.com/linux/latest/source/include/linux/wait.h |

Table 5‑2 External structure definition