Application Note

RENESAS

CONFIDENTIAL

ADSP Driver for Android
RCG3AHPDA8101ZDO

Application Note - Driver -

RCG3AHPDA8101ZDOE_AN_EXT

Rev. 0.10 May, 2018

Renesas Electronics
www.renesas.com

Notice

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

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Trademarks

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

How to Use This Manual

1. Purpose and Target Reader

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

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

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

2. Restrictions on the Use of this Middleware

Any customer who wishes to use this Software must obtain a software license from Renesas Electronics.

3. Related Manuals

4. Technical Terms and Abbreviation

Table of Contents

List of Tables

List of Figures

1. Overview

1.1 Environment

1.1.1 Development Environment

Below table shows development environment for ADSP driver extension.

Environment	Items	Version
Platform	Linux kernel	4.4.0-31-generic
Evaluation Board	Salvator-X board H3/M3	Gen 3
	R-Car H3 Starter Kit	H3 Ver.1.1
	R-Car M3 Starter Kit	M3 Ver.1.0

Table 1-1 Development environment

1.1.2 Build Environment

Below table shows build environment for ADSP driver extension.

Tool	Description	Version
aarch64-poky-linux-gcc	GNU C compiler for the arm64 architecture	5.2.1 20151005

Table 1-2 Build environment

1.2 Module Configuration

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.

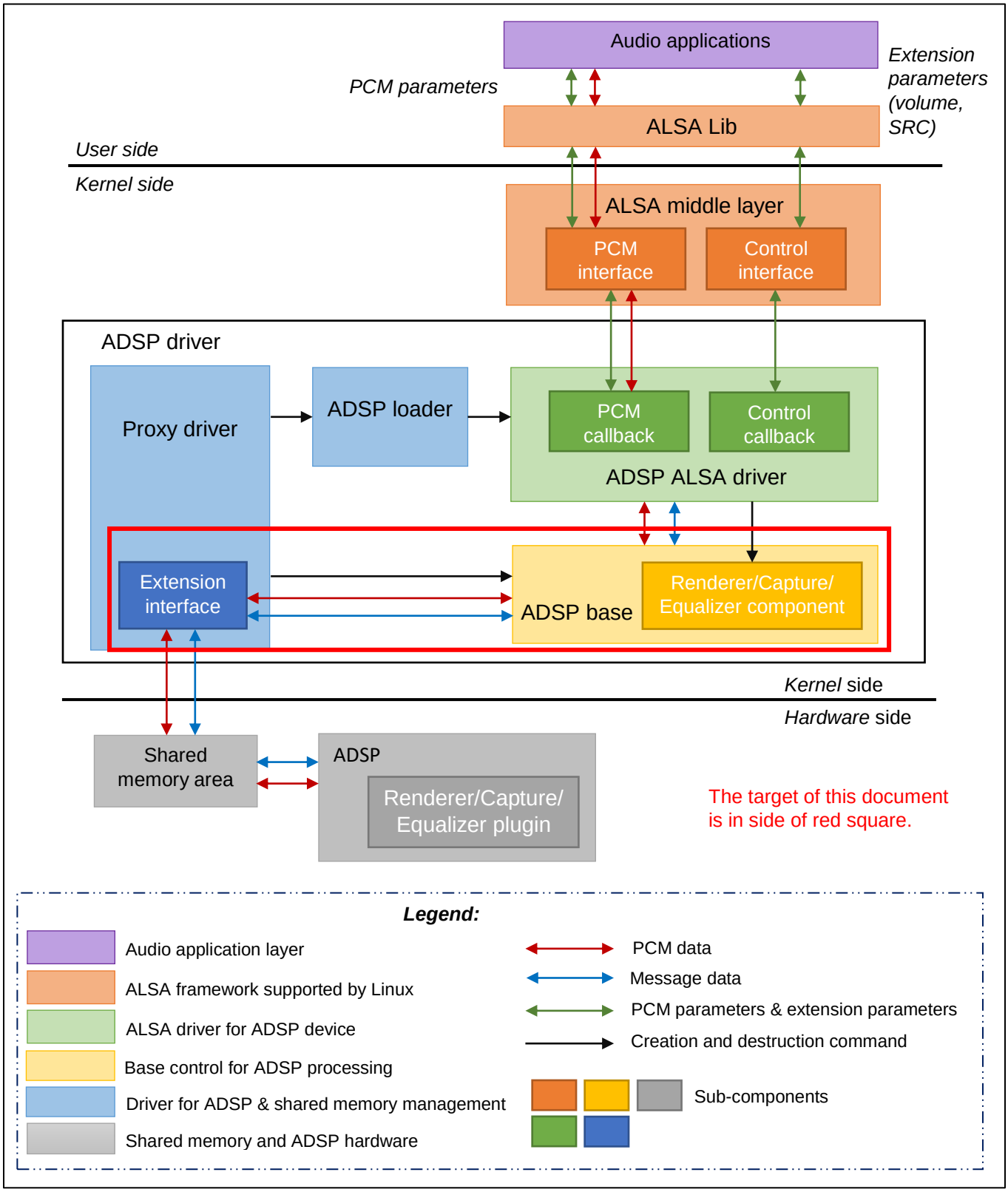

Figure 1-1 Overview configuration of ADSP driver

- **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.

- **ADSP ALSA driver**:
  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, it receives PCM data from user app and transfers to ADSP Renderer plugin. For record, it receives PCM data from ADSP 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.

- **ADSP base**:
  This layer responses to control ADSP plugins. It uses extension interface to sends data from ADSP ALSA driver to proxy driver. To receive the response message from ADSP, this layer provides a thread. This thread is started when ADSP base is created and is stopped when ADSP base is destroyed.

- **ADSP loader**:
  It responds to initialize/de-initialize ADSP ALSA driver and IPC data, load ADSP firmware, hardware management such as ADSP registers initialization, ADSP start/stop, etc.

- **Proxy driver**:
  It responds to initialize/de-initialize ADSP base. It manages sending/receiving messages to/from ADSP by interrupt events and shared memory manipulation, etc. It also implements the extension interface which supports the basic communication with ADSP base such as send/receive message, wait for the response message.

- **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) for playback, record, and equalization. The communication between ADSP side and CPU side is performed by the interrupt, and the shared memory area.

2. ADSP Base

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

No.	Structures	Descriptions
1	xf_callback_func_t	The callback functions for ADSP ALSA driver
2	xf_handle_t	The registered handle's data
3	xf_adsp_base_t	ADSP base's processing data
4	xf_adsp_renderer_params_t	Renderer plugin's parameters
5	xf_adsp_renderer_t	Renderer component's data
6	xf_adsp_capture_params_t	Capture plugin's parameters
7	xf_adsp_capture_t	Capture component's data
8	xf_equalizer_parametric_coef_t	Parametric coefficients of Equalizer plugin
9	xf_equalizer_graphic_coef_t	Graphic coefficients of Equalizer plugin
10	xf_adsp_equalizer_params_t	Equalizer plugin's parameters
11	xf_adsp_equalizer_t	Equalizer 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

Table 2-2 List of APIs in ADSP base

2.1 Structure

2.1.1 xf_callback_func_t structure

xf_callback_func_t	
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

2.1.2 xf_handle_t structure

xf_handle_t		
int	comp_id	Component ID of handler that registered in ADSP
xf_callback_func_t	*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

2.1.3 xf_adsp_base_t structure

xf_adsp_base_t		
void	*client	Client data which registered on proxy driver
xf_pool_t	*aux_pool	Auxiliary pool data which registered to ADSP [*]
xf_handle_t	*handle[256]	Handle data which registered by ADSP ALSA driver
struct task_struct	*rsp_thread	Response thread of ADSP base [*]
wait_queue_head_t	base_wait	Wait queue for the response messages [*]
xf_message_t	base_msg	The message for stored the response message [*]
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

2.1.4 xf_adsp_renderer_params_t structure

xf_adsp_renderer_params_t		
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

2.1.5 xf_adsp_renderer_t structure

xf_adsp_renderer_t		
xf_adsp_renderer_params_t	params	Store the parameters of Renderer plugin
xf_pool_t	*buf_pool	Buffer pool data that used for PCM transfer
int	handle_id	Handle ID that Renderer has registered in ADSP base

2.1.6 xf_adsp_capture_params_t structure

xf_adsp_capture_params_t		
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

2.1.7 xf_adsp_capture_t structure

xf_adsp_capture_t	
xf_adsp_capture_params_t params	Store the parameters of Capture plugin
xf_pool_t *buf_pool	Buffer pool data that used for PCM transfer
int handle_id	Handle ID that Capture has registered in ADSP base

2.1.8 xf_equalizer_parametric_coef_t structure

xf_equalizer_parametric_coef_t	
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

2.1.9 xf_equalizer_graphic_coef_t structure

xf_equalizer_graphic_coef_t	
int gain_g[5]	Graphic filter gain

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

2.1.10 xf_adsp_equalizer_params_t structure

xf_adsp_equalizer_params_t	
int channel	PCM channel
int pcm_width	PCM width
int rate	Sampling rate
int eqz_type	Equalizer type
xf_equalizer_parametric_coef_t p_coef	Parametric coefficients
xf_equalizer_graphic_t g_coef	Graphic coefficient

2.1.11 xf_adsp_equalizer_t structure

xf_adsp_equalizer_t	
xf_adsp_equalizer_params_t params	Store the parameters of Equalizer plugin
xf_pool_t *buf_pool	Buffer pool data that used for PCM transfer
int handle_id	Handle ID that Equalizer has registered

2.2 Interface APIs

2.2.1 xf_adsp_base_create

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).	
Syntax	int xf_adsp_base_create(void)	
Parameter	-	-
Return value	0	ADSP base registers successful.
	-EINVAL	ADSP base has been alive.
	-ENOMEM	Cannot allocate memory for ADSP base usage.
	-EBUSY	ADSP base cannot register to proxy driver due to number of client has been exceeded.

2.2.2 xf_adsp_base_destroy

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 registered yet.

2.2.3 xf_adsp_empty_this_buffer

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.

2.2.4 xf_adsp_fill_this_buffer

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.

2.2.5 xf_adsp_allocate_mem_pool

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	xf_pool_t *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

2.2.6 xf_adsp_free_mem_pool

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(xf_pool_t *pool)	
Parameter	xf_pool_t *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.

2.2.7 xf_adsp_get_data_from_pool

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(xf_pool_t *pool, int index)	
Parameter	xf_pool_t *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.

2.2.8 xf_adsp_set_param

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.

2.2.9 xf_adsp_get_param

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.

2.2.10 xf_adsp_route

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	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.

2.2.11 xf_adsp_renderer_create

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 has been registered completely. Finally, it initializes all parameters as default values from plugin.	
Syntax	int xf_adsp_renderer_create(xf_adsp_renderer_t **renderer, xf_callback_func_t *cb, void *private_data)	
Parameter	xf_adsp_renderer_t **renderer	The pointer to store the created Renderer instance
	xf_callback_func_t *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 plugin.
	-ENOMEM	Cannot allocate Renderer instance

2.2.12 xf_adsp_renderer_destroy

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

2.2.13 xf_adsp_renderer_set_params

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(xf_adsp_renderer_t *renderer)	
Parameter	xf_adsp_renderer_t *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 plugin.

2.2.14 xf_adsp_renderer_get_params

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(xf_adsp_renderer_t *renderer)	
Parameter	xf_adsp_renderer_t *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 plugin.

2.2.15 xf_adsp_capture_create

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 has been registered completely. Finally, it initializes all parameters as default values from plugin.	
Syntax	int xf_adsp_capture_create(xf_adsp_capture_t **capture, xf_callback_func_t *cb, void *private_data)	
Parameter	xf_adsp_capture_t **capture	The pointer to store the created Capture instance
	xf_callback_func_t *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 plugin.
	-ENOMEM	Cannot allocate Capture instance

2.2.16 xf_adsp_capture_destroy

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

2.2.17 xf_adsp_capture_set_params

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(xf_adsp_capture_t *capture)	
Parameter	xf_adsp_capture_t *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 plugin.

2.2.18 xf_adsp_capture_get_params

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(xf_adsp_capture_t *capture)	
Parameter	xf_adsp_capture_t *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 plugin.

2.2.19 xf_adsp_equalizer_create

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 has been registered completely. Finally, it initializes all parameters as default values from plugin.	
Syntax	int xf_adsp_equalizer_create(xf_adsp_equalizer_t **equalizer, xf_callback_func_t *cb, void *private_data)	
Parameter	xf_adsp_equalizer_t **equalizer	The pointer to store the created Equalizer instance
	xf_callback_func_t *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 plugin.
	-ENOMEM	Cannot allocate Equalizer instance

2.2.20 xf_adsp_equalizer_destroy

xf_adsp_equalizer_destroy		
Synopsis	This API unregisters ADSP Equalizer plugin, frees registered handler and instance.	
Syntax	int xf_adsp_equalizer_destroy(xf_adsp_equalizer_t *equalizer)	
Parameter	xf_adsp_equalizer_t *equalizer	Pointer to Equalizer instance
Return value	0	Success

	-EINVAL	ADSP base has not registered yet, or Equalizer instance is invalid.

2.2.21 xf_adsp_equalizer_set_params

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(xf_adsp_equalizer_t *equalizer)	
Parameter	xf_adsp_equalizer_t *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 plugin.

2.2.22 xf_adsp_equalizer_get_params

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(xf_adsp_equalizer_t *equalizer)	
Parameter	xf_adsp_equalizer_t *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 plugin.

3. 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

3.1 xf_adsp_base_client_register

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.

3.2 xf_adsp_base_client_unregister

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.

3.3 xf_adsp_base_send

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.

3.4 xf_adsp_base_recv

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.

3.5 xf_adsp_base_poll

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.

4. Processing Flow

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

4.1 ADSP Base Flow

4.1.1 ADSP Base Creation

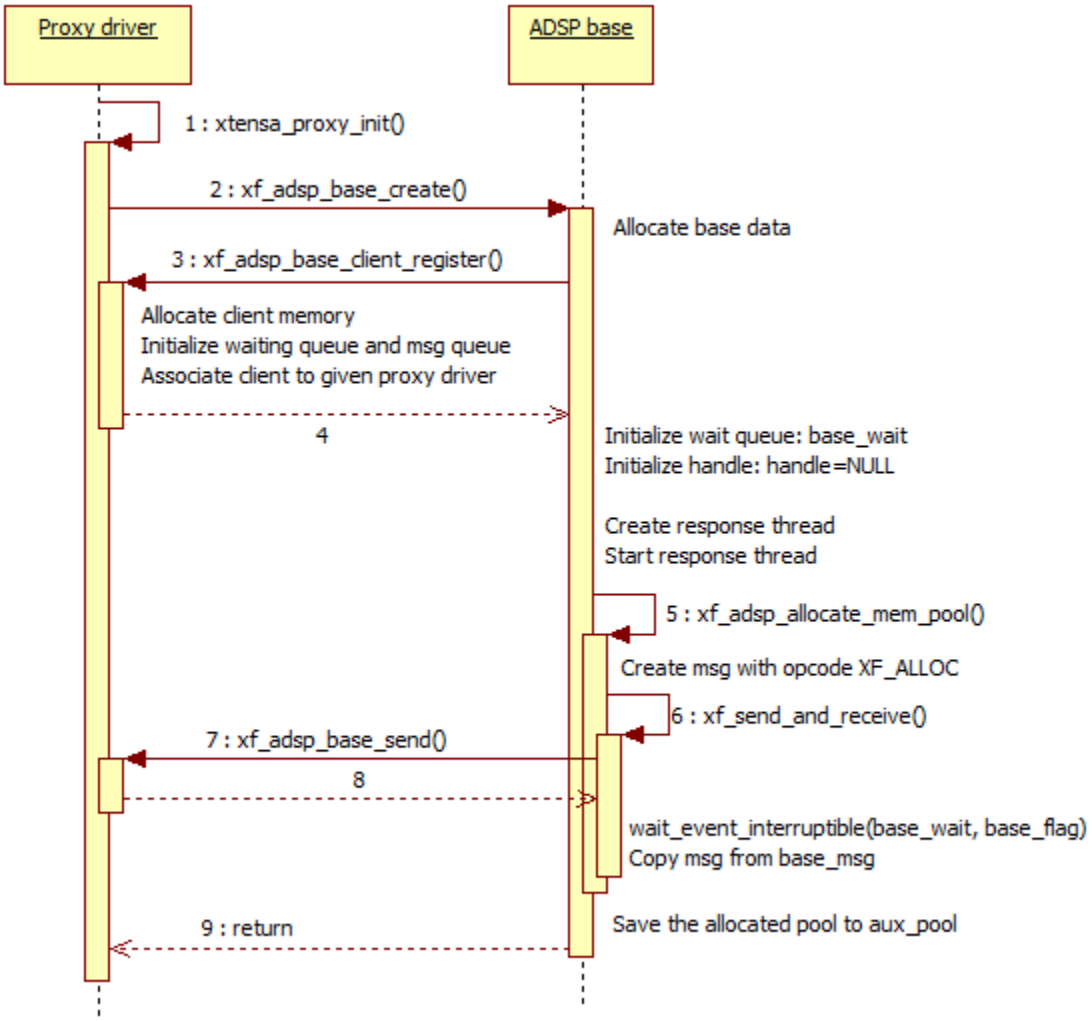

Figure 4-1 ADSP base creation flow chart

4.1.2 ADSP Base Destruction

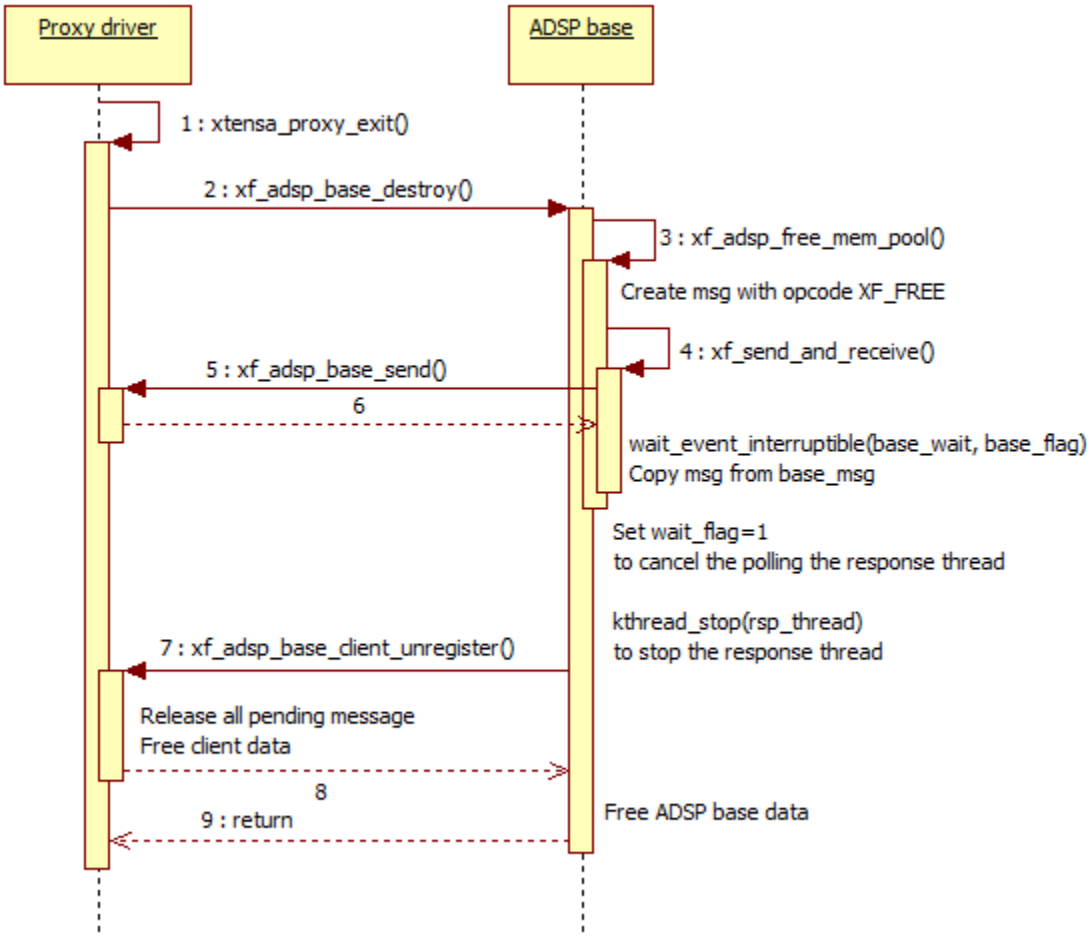

Figure 4-2 ADSP base destruction flow chart

4.2 Renderer Flow

4.2.1 Renderer Creation

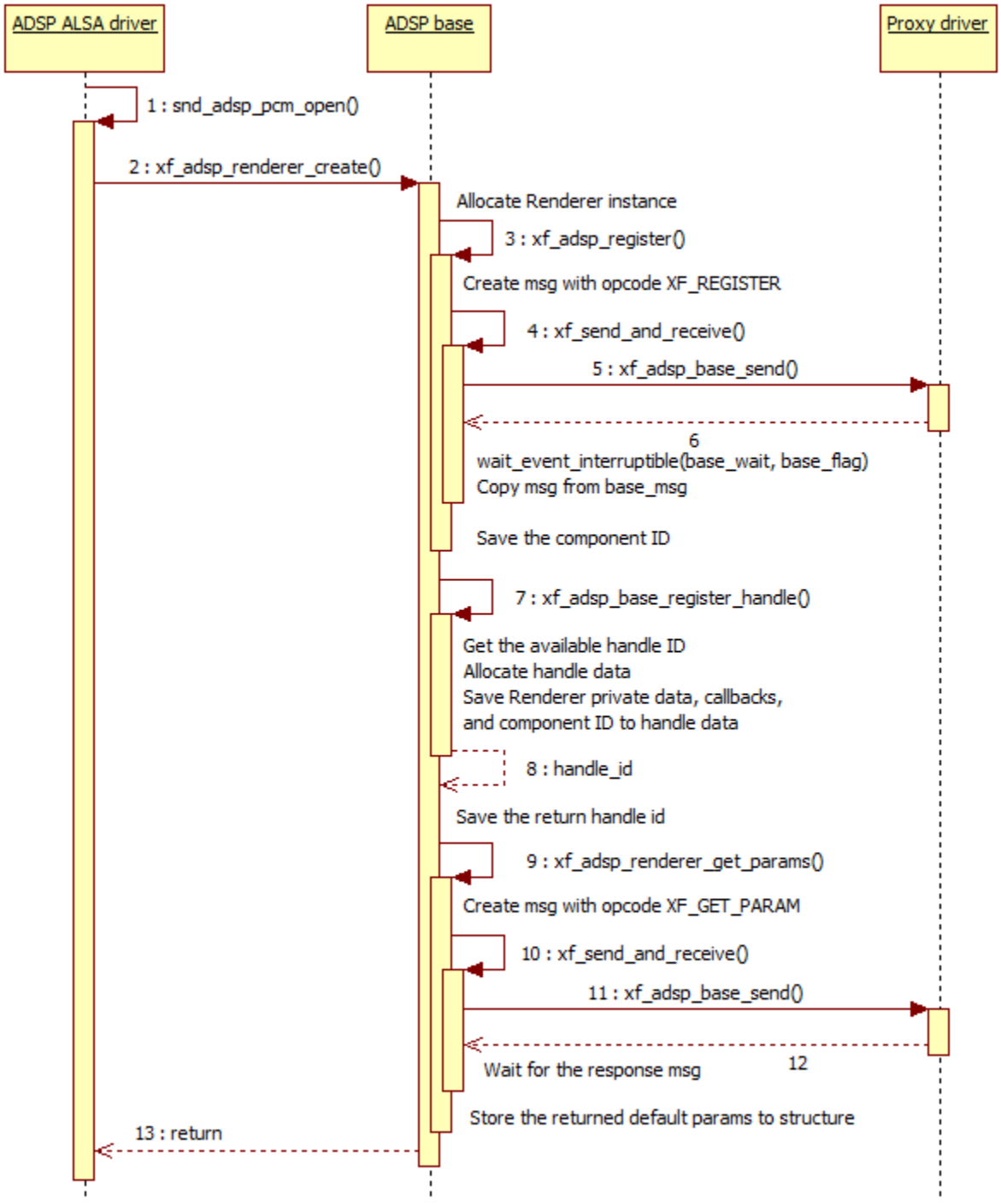

Figure 4-3 Renderer creation flow chart

4.2.2 Renderer Execution

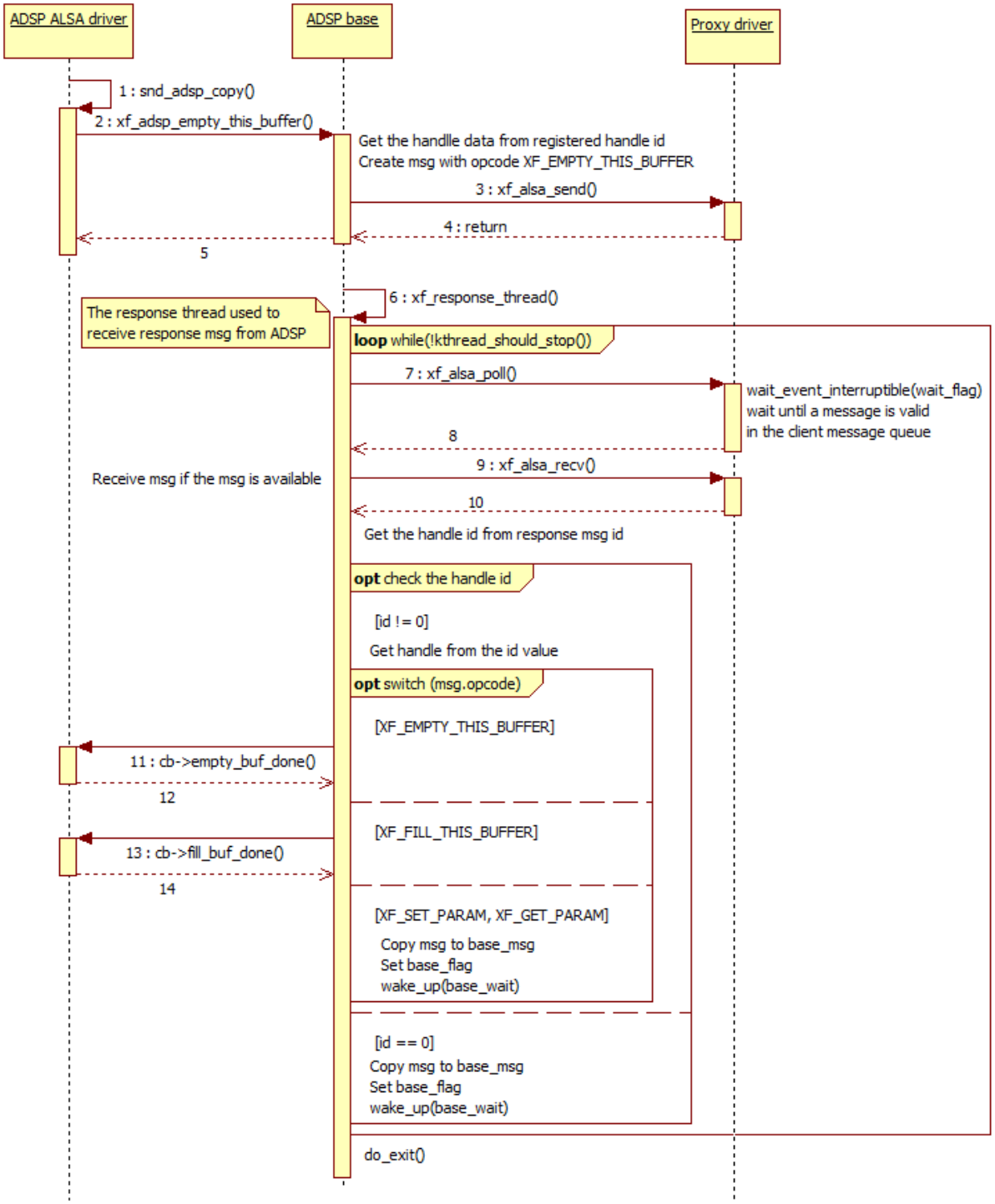

Figure 4-4 Renderer execution flow chart

4.2.3 Renderer Destruction

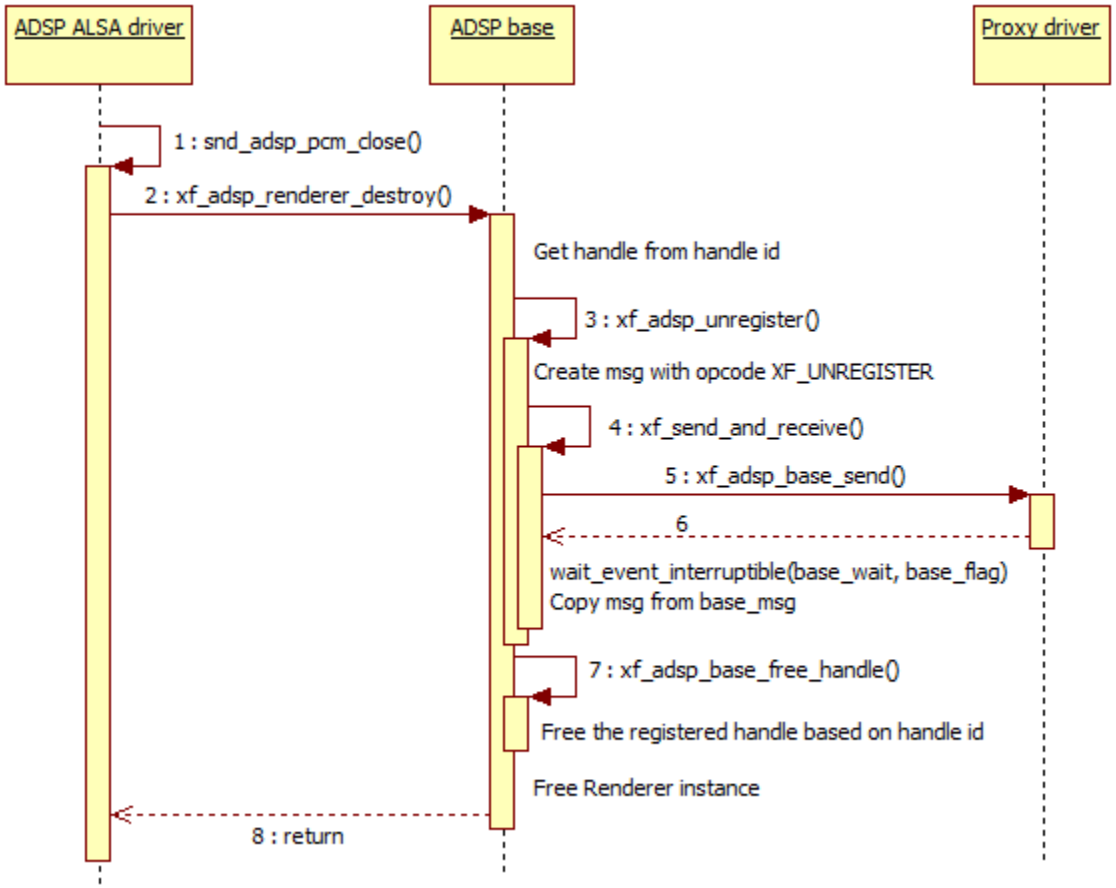

Figure 4-5 Renderer destruction flow chart

4.3 Capture Flow

4.3.1 Capture Creation

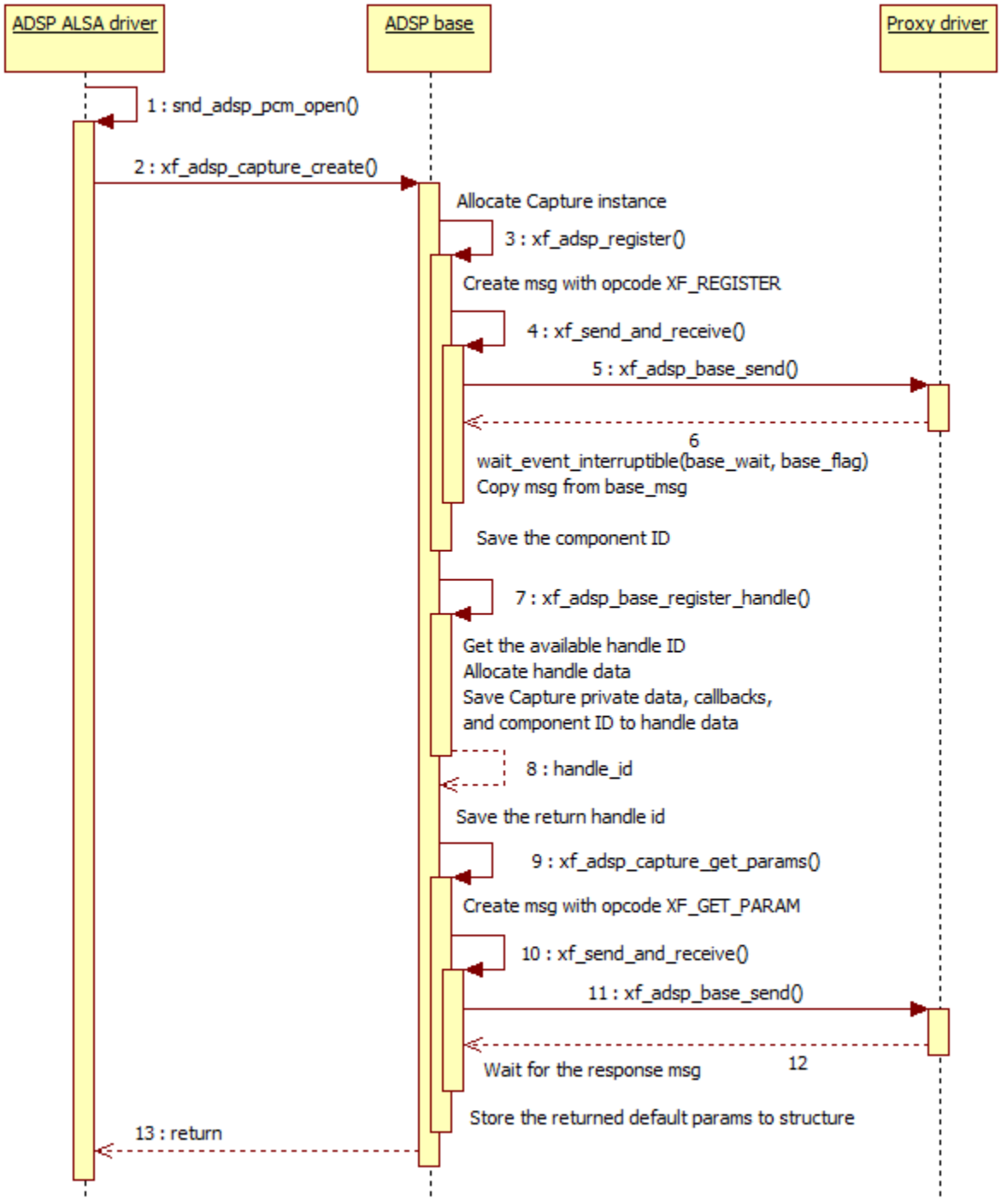

Figure 4-6 Capture creation flow chart

4.3.2 Capture Execution

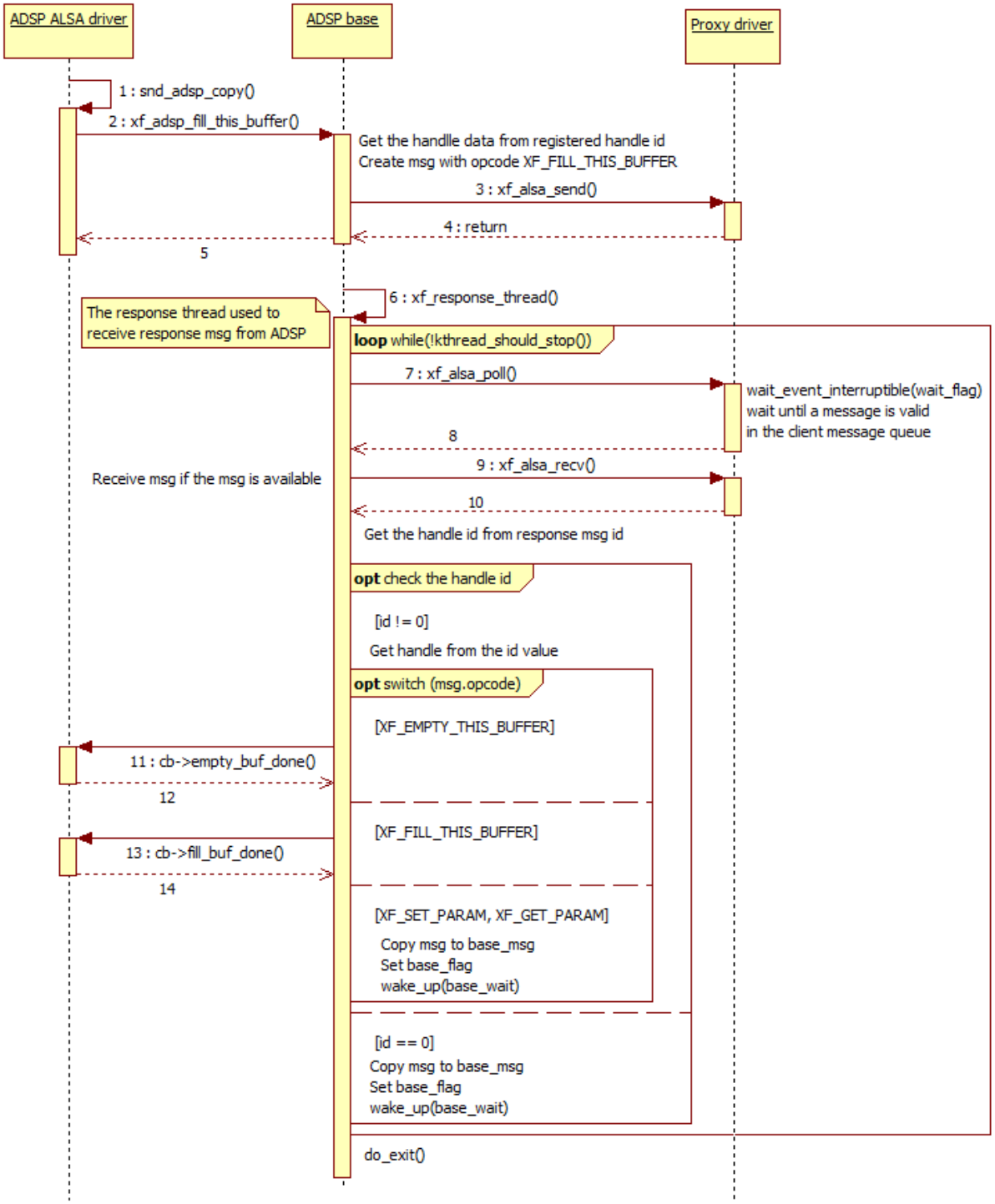

Figure 4-7 Capture execution flow chart

4.3.3 Capture Destruction

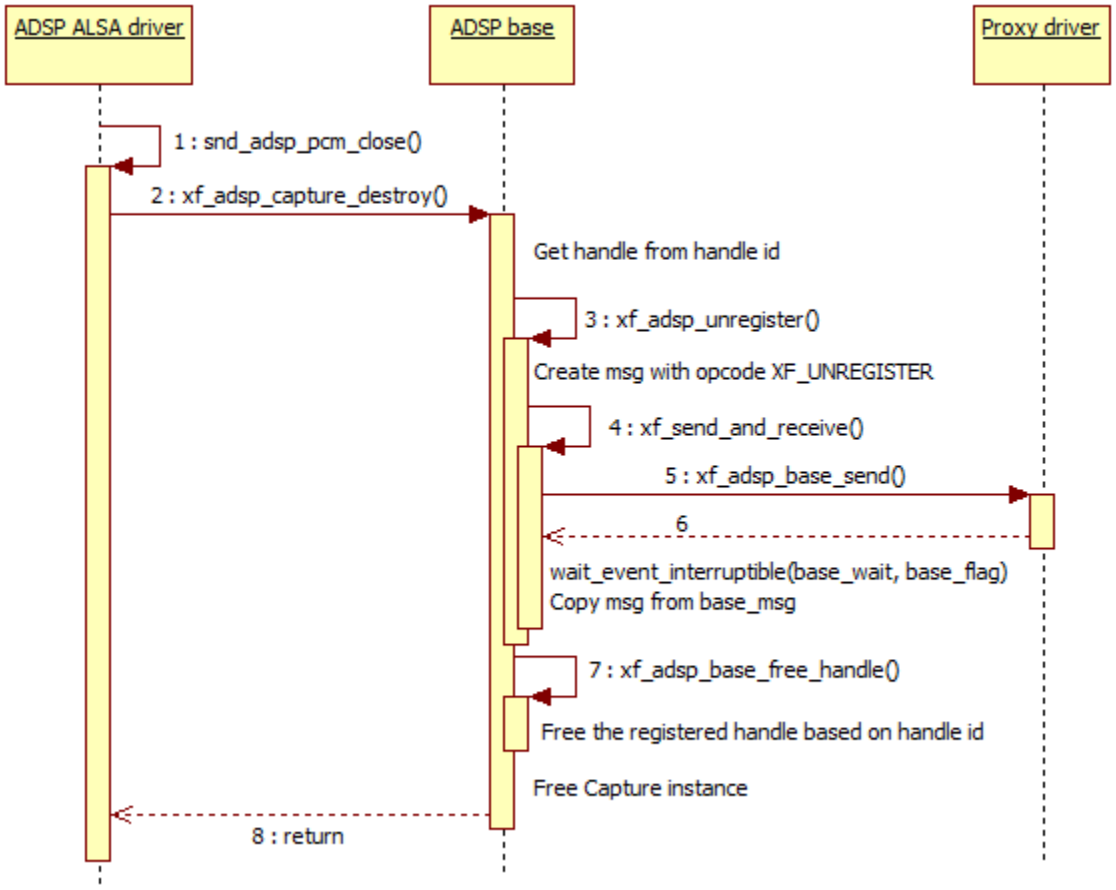

Figure 4-8 Capture destruction flow chart

4.4 Equalizer Flow

4.4.1 Equalizer Creation

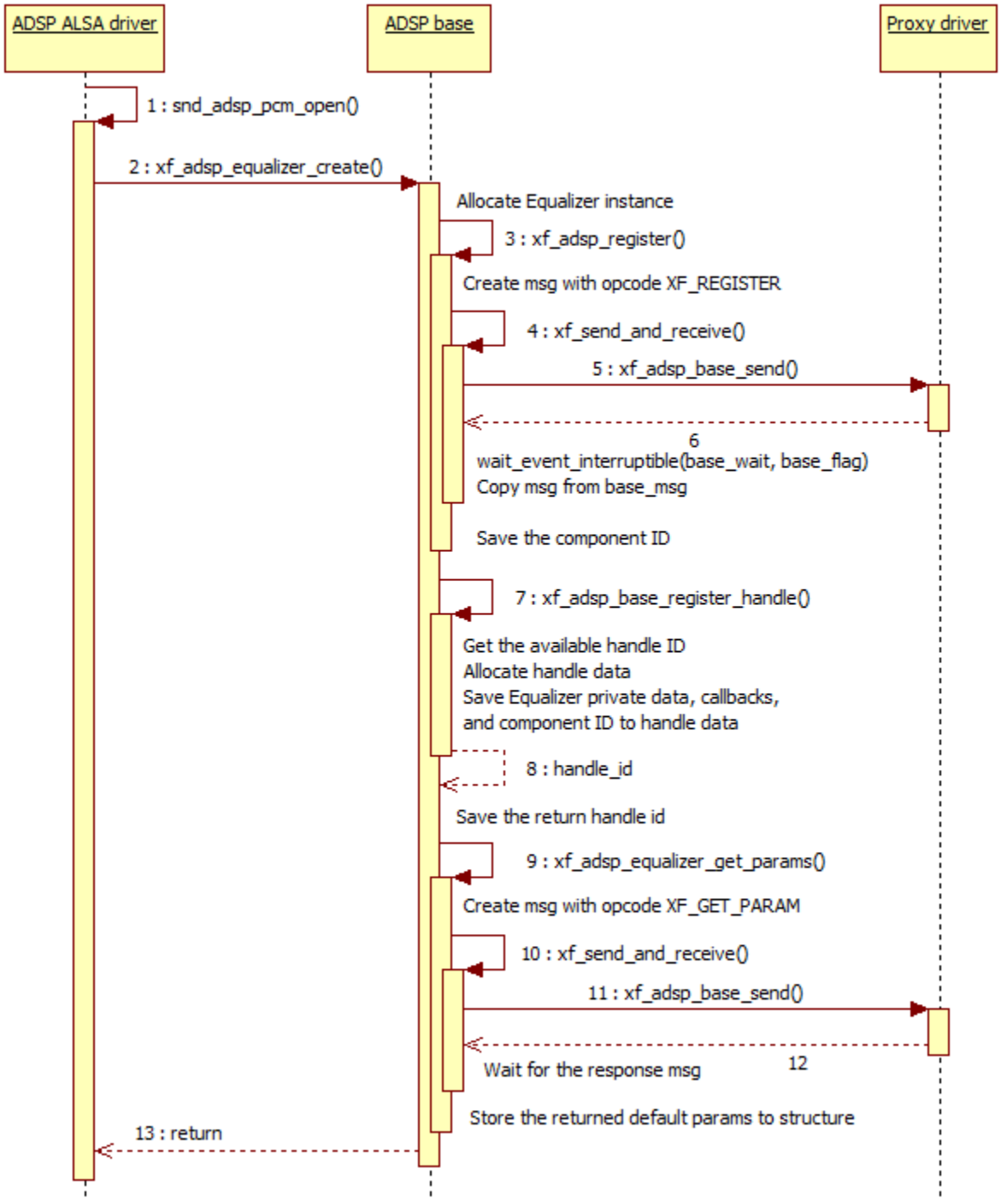

Figure 4-9 Equalizer creation flow chart

4.4.2 Equalizer Execution

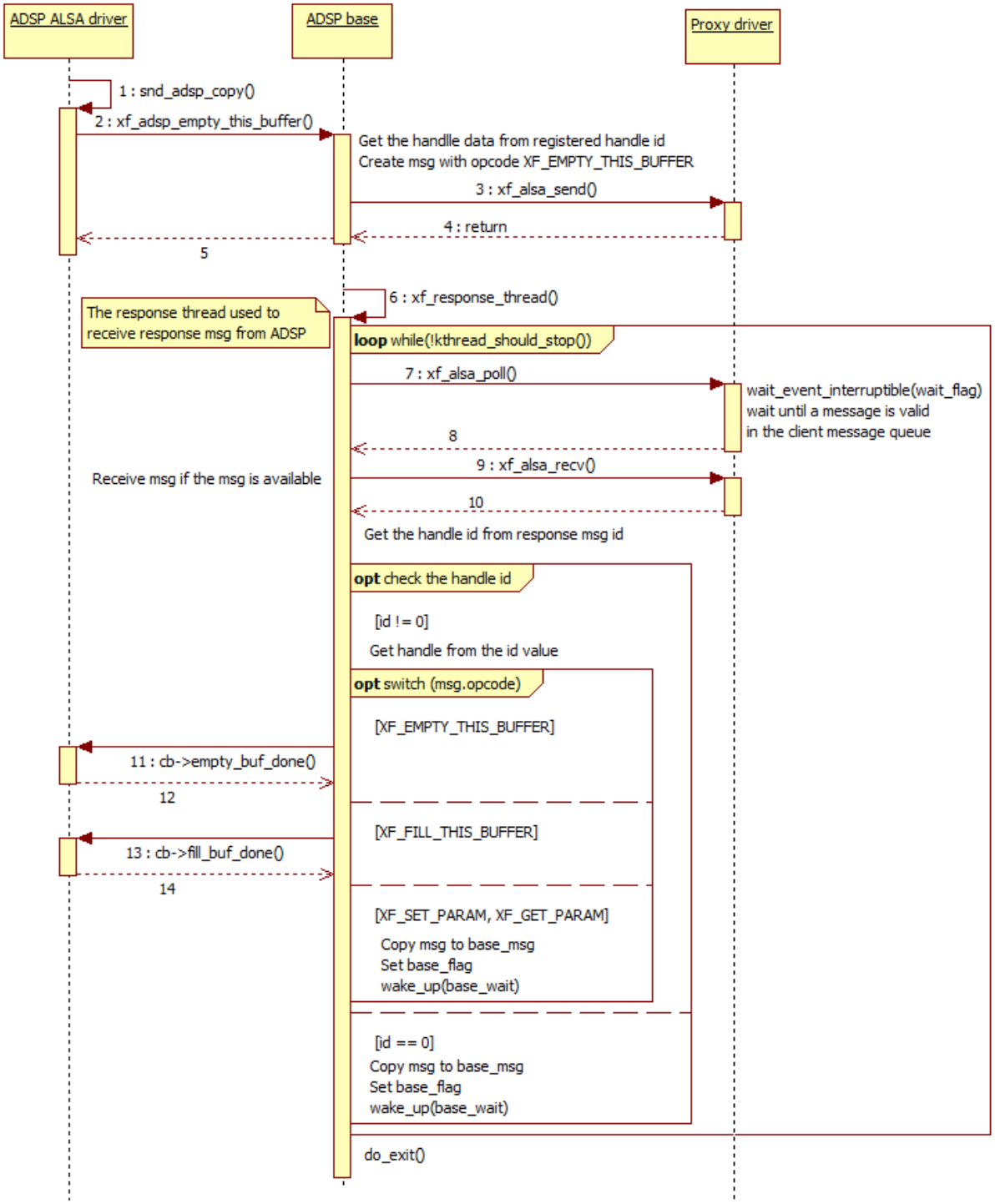

Figure 4-10 Equalizer execution flow chart

4.4.3 Equalizer Destruction

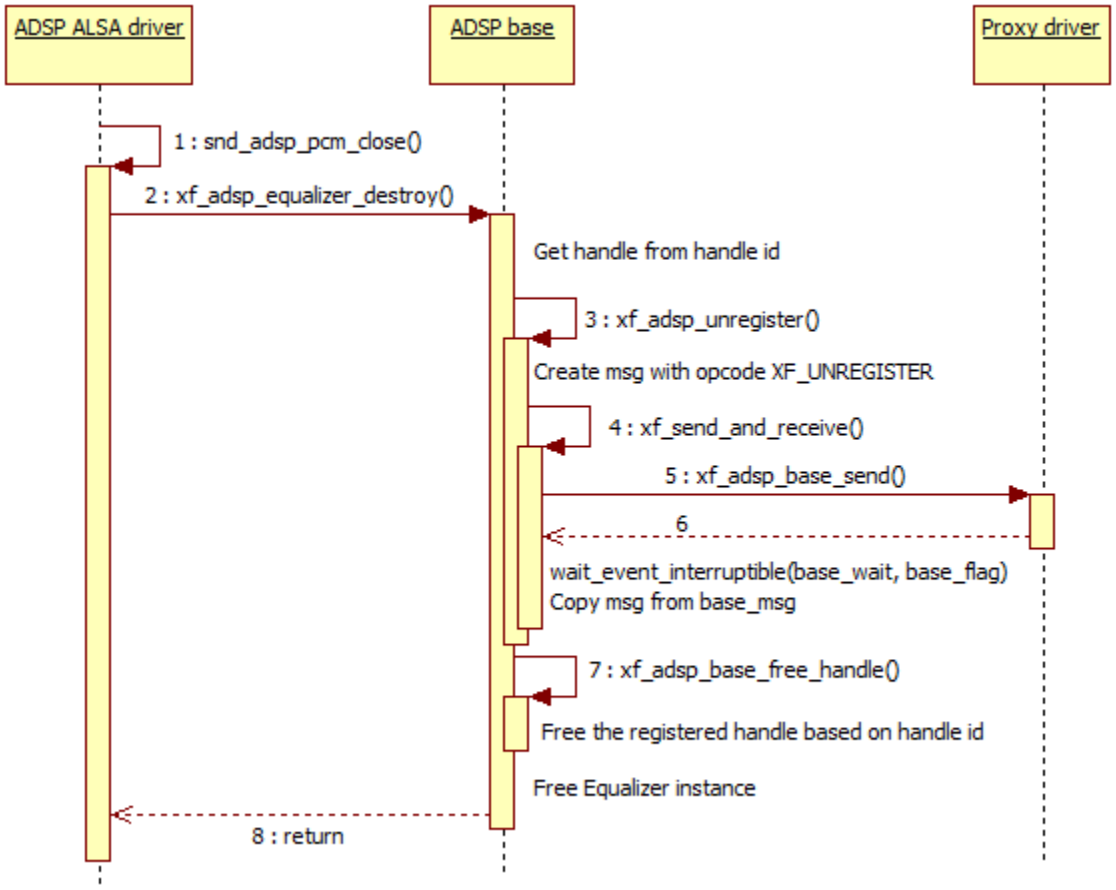

Figure 4-11 Equalizer destruction flow chart

5. Appendix

5.1 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

5.2 Structure

Structure	Description	Reference
xf_pool_t	Data pool type	Defined in the ADSP Interface for Linux document
xf_message_t	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

Revision History	ADSP Interface for Android Application Note - Driver -

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

ADSP Driver for Android Application Note - Driver -

Publication Date: May 30, 2018 Rev. 0.10

Published by: Renesas Electronics Corporation

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

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

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 5.0

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

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

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

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

Colophon 3.1

ADSP Driver for Android
RCG3AHPDA8101ZDO

RCG3AHPDA8101ZDOE_AN_EXT

Renesas Electronics Corporation