Application Note

CONFIDENTIAL

ADSP Driver for Android
RCG3AHPDA8101ZDO

Application Note - ALSA -

RCG3AHPDA8101ZDOE_AN_ALSAIF

Rev. 1.00 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 Figures

List of Tables

1. Overview

1.1. ALSA Driver Architecture

This material describes detailed information of ALSA APIs implemented in ADSP ALSA Driver:

- APIs for PCM data control (**PCM interface**).
- APIs for hardware control (**Control interface**).

  Above interfaces belong to ALSA middle layer. They need to define **PCM callbacks** and **Control callbacks** in ADSP ALSA driver.
- APIs for ALSA sound card register.

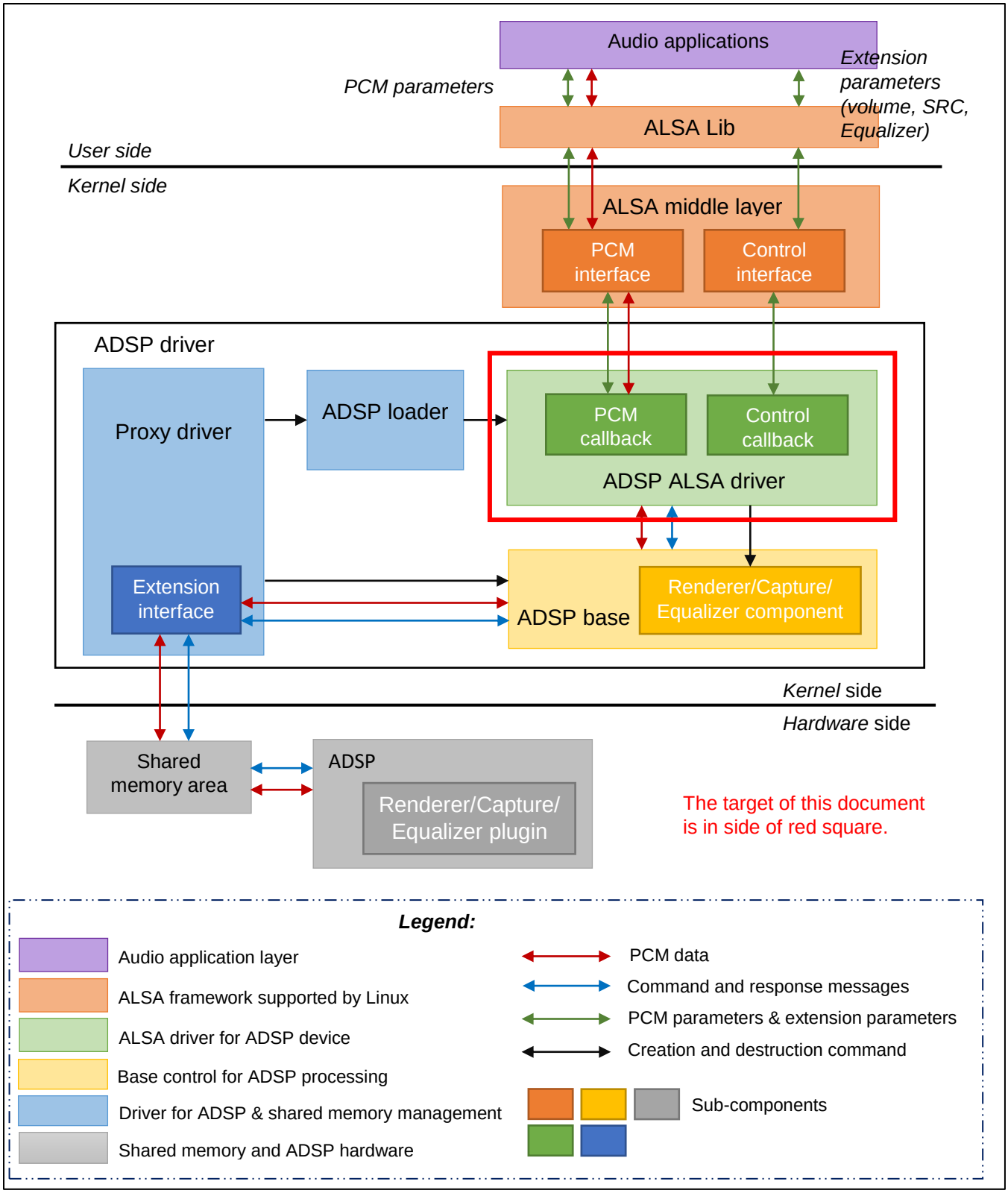

Figure 1-1 Overview architecture of ADSP driver.

- **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.

1.2. Environment

1.2.1. Development Environment

Below table shows development environment for ALSA Interface.

Table 1-1 Development 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

1.2.2. Build Environment

Below table shows build environment for ALSA Interface.

Table 1-2 Build environment

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

2. Terminologies

ALSA	Advanced Linux Sound Architecture (ALSA) is a software framework and part of the Linux kernel that provides an application programming interface (API) for sound card device drivers.
Sound card	The term “sound card” in this material is external audio interfaces used for audio applications to communicate with audio hardware.
Stream	A PCM interface consists of PCM playback and capture streams and each pcm stream consists of one or more pcm substreams.
Substream	A substream correspond to a PCM file opened or recorded.

3. API Specification

3.1. APIs for PCM Interface

Below table presents APIs used for PCM interface. Some callback functions which's name contains "playback" only are used for playback streams and "capture" only are used for capture streams. Other callback functions are common callbacks for both renderer and capture streams.

Table 3-1: List of API functions for PCM interface

Number	API function	Description
1	snd_adsp_playback_pcm_open	In playback case, register a Renderer plugin, get range of hardware parameter into substream.
2	snd_adsp_playback_pcm_close	In playback case, unregister Renderer plugin.
3	snd_adsp_capture_pcm_open	In capture case, register a Capture plugin, get range of hardware parameter into substream.
4	snd_adsp_capture_pcm_close	In capture case, unregister Capture plugin.
5	snd_adsp_pcm_hw_params	This callback is a dummy function. It is added follow the flow of ALSA PCM interface.
6	snd_adsp_pcm_hw_free	This callback is a dummy function. It is added follow the flow of ALSA PCM interface.
7	snd_adsp_playback_pcm_prepare	In playback case, initialize hardware data pointer. Renderer will be set up to go to running state.
8	snd_adsp_capture_pcm_prepare	In capture case, initialize hardware data pointer. Capture will be set up to go to running state.
9	snd_adsp_pcm_trigger	Start, stop pcm substream.
10	snd_adsp_playback_pcm_copy	In playback case, copy data from user contains stream data to HW buffer.
11	snd_adsp_capture_pcm_copy	In capture case, copy data from HW buffer to user.
12	snd_adsp_playback_pcm_silence	In playback case, it's function is same with snd_adsp_playback_pcm_copy but it only copy data with value 0 to HW buffer.
13	snd_adsp_capture_pcm_silence	In capture case, it's function is same with snd_adsp_capture_pcm_copy but it only transfer data with value 0 to user.

14	snd_adsp_playback_pcm_pointer	In playback case, return hardware data position on the buffer in frames.
15	snd_adsp_capture_pcm_pointer	In capture case, return hardware data position on the buffer in frames.

3.2. APIs for hardware control

Below table presents APIs used for Control interface. The interface includes Volume control, Sample Rate control, Equalizer control.

Table 3-2: List of API functions for Volume control and Sample Rate control

Number	API function	Description
1	snd_adsp_control_volume_info	Get detail information on volume control
2	snd_adsp_control_sample_rate_info	Get detail information on sample rate control
3	snd_adsp_control_output1_info	Get detail information on the first output device control
4	snd_adsp_control_output2_info	Get detail information on the second output device control
5	snd_adsp_control_input1_info	Get detail information on the first input device control
6	snd_adsp_control_input2_info	Get detail information on the second input device control
7	snd_adsp_control_volume_get	Get volume setting value
8	snd_adsp_control_sample_rate_get	Get sample rate output setting value
9	snd_adsp_control_output1_get	For playback case, get the first output device value
10	snd_adsp_control_output2_get	For playback case, get the second output device value
11	snd_adsp_control_input1_get	For capture case, get the first input device value
12	snd_adsp_control_input2_get	For capture case, get the second input device value
13	snd_adsp_control_volume_put	Set volume value
14	snd_adsp_control_sample_rate_put	Set sample rate output value
15	snd_adsp_control_output1_put	For playback case, set the first output device
16	snd_adsp_control_output2_put	For playback case, set the second output device
17	snd_adsp_control_input1_put	For capture case, set the first input device
18	snd_adsp_control_input2_put	For capture case, set the second input device

Table 3-3 List of API functions for Equalizer control

Number	API function	Description
1	snd_adsp_control_eqz_switch_info	Get detail info on the equalizer switch control.
2	snd_adsp_control_eqz_switch_get	Get information about equalizer activation.
3	snd_adsp_control_eqz_switch_put	Enable or disable equalizer control.
4	snd_adsp_control_eqz_info	Get detailed info on the equalizer control.
5	snd_adsp_control_eqz_get	Get equalizer parameters.
6	snd_adsp_control_eqz_put	Set equalizer parameters.

3.3. APIs for ALSA sound card register

Below table shows APIs used for ALSA sound card register and unregister.

Table 3-4 List of API functions for ALSA sound card

Number	API function	Description
1	snd_adsp_alsa_register	Register an ALSA soundcard for ADSP device.
2	snd_adsp_alsa_unregister	Unregister an ALSA soundcard for ADSP device.
3	snd_adsp_alsa_get_ipc	Gets the proxy IPC data that was stored in ADSP ALSA driver instance

3.4. Detail of APIs for PCM Interface

3.4.1. snd_adsp_playback_pcm_open

snd_adsp_playback_pcm_open		
Synopsis	Register a Renderer plugin. It also registers Equalizer plugin if Equalizer switch is set to 1. Get range of hardware parameter into substream. This callback is used for playback streams.	
Syntax	int snd_adsp_playback_pcm_open(struct snd_pcm_substream *substream)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream
Return value	0	Success
	-EINVAL	Renderer or Equalizer plugin register fails.

3.4.2. snd_adsp_capture_pcm_open

snd_adsp_capture_pcm_open		
Synopsis	Register a Capture plugin. It also registers Equalizer plugin if Equalizer switch is set to 1. Get range of hardware parameter into substream. This callback is used for capture streams.	
Syntax	int snd_adsp_capture_pcm_open(struct snd_pcm_substream *substream)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream
Return value	0	Success
	-EINVAL	Capture or Equalizer plugin register fails.

3.4.3. snd_adsp_playback_pcm_close

snd_adsp_playback_pcm_close		
Synopsis	Unregister Renderer plugin. It also unregisters Equalizer plugin if Equalizer switch is set to 1. This callback is used for playback streams.	
Syntax	int snd_adsp_playback_pcm_close(struct snd_pcm_substream *substream)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream
Return value	0	Success
	-EINVAL	Renderer or Equalizer plugin unregister fails.

3.4.4. snd_adsp_capture_pcm_close

snd_adsp_capture_pcm_close		
Synopsis	Unregister Capture plugin. It also unregisters Equalizer plugin if Equalizer switch is set to 1. This callback is used for capture streams.	
Syntax	int snd_adsp_capture_pcm_close(struct snd_pcm_substream *substream)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream
Return value	0	Success
	-EINVAL	Capture or Equalizer plugin unregister fails.

3.4.5. snd_adsp_pcm_hw_params

snd_adsp_pcm_hw_params		
Synopsis	This callback is a dummy function. It is added follow the flow of ALSA PCM interface. It always returns 0.	
Syntax	int snd_adsp_pcm_hw_params(struct snd_pcm_substream *substream, struct snd_pcm_hw_params *hw_params)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream
	hw_params	hardware parameter is set up by the application
Return value	0	Always return 0

3.4.6. snd_adsp_pcm_hw_free

snd_adsp_pcm_hw_free		
Synopsis	This callback is a dummy function. It is added follow the flow of ALSA PCM interface. It always returns 0.	
Syntax	int snd_adsp_pcm_hw_free(struct snd_pcm_substream *substream)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream
Return value	0	Always return 0

3.4.7. snd_adsp_playback_pcm_prepare

snd_adsp_playback_pcm_prepare		
Synopsis	In playback case, initialize hardware data position with value 0 then set default volume value is 100% unless user set volume before. In this function Renderer plugin is set up to go to running state. If Equalizer switch is enabled, It is also set up. Then set route between Renderer and Equalizer. This callback is also called when data overrun or underrun error occurs.	
Syntax	int snd_adsp_pcm_prepare(struct snd_pcm_substream *substream)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream
Return value	0	Success
	-EINVAL	Cannot set parameters to Renderer or Equalizer plugin. Cannot allocate memory pool. Cannot route Equalizer and Renderer.

[Note] Unless setting a volume value ADSP plugin will run with default value. Then it will not enable DVC module. So volume setting in runtime will not affect.

3.4.8. snd_adsp_capture_pcm_prepare

snd_adsp_pcm_prepare		
Synopsis	In capture case, initialize hardware data position with value 0 then set default volume value is 100% unless user set volume before. In this function Capture plugin is set up to go to running state. If Equalizer switch is enabled, it is also set up. Then set route between Capture and Equalizer. This callback is also called when data overrun or underrun error occurs.	
Syntax	int snd_adsp_pcm_prepare(struct snd_pcm_substream *substream)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream
Return value	0	Success
	-EINVAL	Cannot set parameters to Capture or Equalizer plugin. Cannot allocate memory pool. Cannot route Capture and Equalizer. Sending fill this buffer message to ADSP get failed

[Note] Unless setting a volume value ADSP plugin will run with default value. Then it will not enable DVC module. So volume setting in runtime will not affect.

3.4.9. snd_adsp_pcm_trigger

snd_adsp_pcm_trigger		
Synopsis	This is called to start, stop pcm substream.	
Syntax	int snd_adsp_pcm_trigger(struct snd_pcm_substream *substream, int idx)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream
	int idx	Start, stop command
Return value	0	Success
	-EINVAL	Invalid command index

3.4.10. snd_adsp_playback_pcm_copy

snd_adsp_playback_pcm_copy		
Synopsis	Copy PCM data from user to HW buffer. This callback is used for playback streams.	
Syntax	static int snd_adsp_playback_pcm_copy(struct snd_pcm_substream *substream, int channel, unsigned long pos, void __user *src, unsigned long bytes)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream.
	int channel	Channel number
	unsigned long pos	Current position offset in bytes on hardware buffer.
	void __user *src	Pointer to PCM data in source.
	unsigned long bytes	Give amount of data in bytes at the specified pointer (src).
Return value	0	Success
	-EINVAL	Can't send request data transfer to ADSP

3.4.11. snd_adsp_capture_pcm_copy

snd_adsp_capture_pcm_copy		
Synopsis	Copy PCM data from HW buffer to user. This callback is used for capture streams.	
Syntax	static int snd_adsp_capture_pcm_copy(struct snd_pcm_substream *substream, int channel, unsigned long pos, void __user *dst, unsigned long bytes)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream.
	int channel	Channel number
	unsigned long pos	Current position offset in bytes on hardware buffer.
	void __user *dst	Pointer to PCM data in destination.
	unsigned long bytes	Give amount of data in bytes at the specified offset (pos) on hardware buffer.
Return value	0	Success
	-EINVAL	Can't send request data transfer to ADSP

3.4.12. snd_adsp_playback_pcm_silence

snd_adsp_playback_pcm_silence		
Synopsis	Copy PCM data with value 0 to HW buffer. This callback is used for playback streams.	
Syntax	static int snd_adsp_playback_pcm_silence(struct snd_pcm_substream *substream, int channel, unsigned long pos, unsigned long bytes)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream.
	int channel	Channel number
	unsigned long pos	Current position offset in frames on hardware buffer.
	unsigned long bytes	Give amount of silence data in frames.
Return value	0	Success
	-EINVAL	Can't send request data transfer to ADSP

3.4.13. snd_adsp_capture_pcm_silence

snd_adsp_capture_pcm_silence		
Synopsis	Copy PCM data with value 0 to user. This callback is used for capture streams.	
Syntax	static int snd_adsp_capture_pcm_silence(struct snd_pcm_substream *substream, int channel, unsigned long pos, unsigned long bytes)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream.
	int channel	Channel number
	unsigned long pos	Current position offset in frames on hardware buffer.
	unsigned long bytes	Give amount of silence data in frames.
Return value	0	Success
	-EINVAL	Can't send request data transfer to ADSP

3.4.14. snd_adsp_playback_pcm_pointer

snd_adsp_playback_pcm_trigger		
Synopsis	In playback case, return data position on the buffer in frames.	
Syntax	snd_pcm_uframes_t snd_adsp_playback_pcm_pointer(struct snd_pcm_substream *substream)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream
Return value	pointer	Position offset in frames on hardware buffer.

3.4.15. snd_adsp_capture_pcm_pointer

snd_adsp_capture_pcm_pointer		
Synopsis	In capture case, return data position on the buffer in frames.	
Syntax	snd_pcm_uframes_t snd_adsp_capture_pcm_pointer(struct snd_pcm_substream *substream)	
Parameter	struct snd_pcm_substream *substream	Pointer to a pcm substream
Return value	pointer	Position offset in frames on hardware buffer.

3.5. Detail of APIs for hardware control

❖ APIs get detailed information from the control

3.5.1. snd_adsp_control_volume_info

snd_adsp_control_volume_info		
Synopsis	Get detailed information on volume control.	
Syntax	int snd_adsp_control_volume_info(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_info *uinfo)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_info *uinfo	Pointer to info structure of volume control
Return value	0	Always return 0

3.5.2. snd_adsp_control_sample_rate_info

snd_adsp_control_sample_rate_info		
Synopsis	Get detailed information on sample rate control.	
Syntax	int snd_adsp_control_sample_rate_info(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_info *uinfo)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_info *uinfo	Pointer to info structure of sample rate control
Return value	0	Always return 0

3.5.3. snd_adsp_control_output1_info

snd_adsp_control_output1_info		
Synopsis	Get detailed information on the first output device control.	
Syntax	int snd_adsp_control_output1_info(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_info *uinfo)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_info *uinfo	Pointer to info structure of the first output device control
Return value	0	Always return 0

3.5.4. snd_adsp_control_output2_info

snd_adsp_control_output2_info		
Synopsis	Get detailed information on the second output device control.	
Syntax	int snd_adsp_control_output2_info(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_info *uinfo)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_info *uinfo	Pointer to info structure of the second output device control
Return value	0	Always return 0

3.5.5. snd_adsp_control_input1_info

snd_adsp_control_input1_info		
Synopsis	Get detailed information on the first input device control.	
Syntax	int snd_adsp_control_input1_info(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_info *uinfo)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_info *uinfo	Pointer to info structure of the first input device control
Return value	0	Always return 0

3.5.6. snd_adsp_control_input2_info

snd_adsp_control_input2_info		
Synopsis	Get detailed information on the second input device control.	
Syntax	int snd_adsp_control_input2_info(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_info *uinfo)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_info *uinfo	Pointer to info structure of the second input device control
Return value	0	Always return 0

3.5.7. snd_adsp_control_eqz_switch_info

snd_adsp_control_eqz_switch_info		
Synopsis	Get detailed info on the equalizer switch control.	
Syntax	int snd_adsp_control_eqz_switch_info(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_info *uinfo)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_info *uinfo	Pointer to info structure of the equalizer switch control
Return value	0	Always return 0

3.5.8. snd_adsp_control_eqz_info

snd_adsp_control_eqz_info		
Synopsis	Get detailed info on the equalizer control.	
Syntax	int snd_adsp_control_eqz_info(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_info *uinfo)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_info *uinfo	Pointer to info structure of the equalizer control
Return value	0	Always return 0

❖ APIs get value from hardware and return to user-space

3.5.9. snd_adsp_control_volume_get

snd_adsp_control_volume_get		
Synopsis	Get the PCM volume rate setting value.	
Syntax	int snd_adsp_control_volume_get(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to volume value
Return value	0	Success
	-EINVAL	Can't get parameter information from ADSP.

3.5.10. snd_adsp_control_sample_rate_get

snd_adsp_control_sample_rate_get		
Synopsis	Get sample rate value of hardware output in playback case and sample rate of hardware input in capture case.	
Syntax	int snd_adsp_control_sample_rate_get(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to sample rate value
Return value	0	Success
	-EINVAL	Can't get parameter information from ADSP.

3.5.11. snd_adsp_control_output1_get

snd_adsp_control_output1_get		
Synopsis	Get the 1st output destination device for Renderer info	
Syntax	int snd_adsp_control_output1_get(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to output device 1 information
Return value	0	Success
	-EINVAL	Can't get parameter information from ADSP.

3.5.12. snd_adsp_control_output2_get

snd_adsp_control_output2_get		
Synopsis	Get the 2nd output destination device for Renderer info	
Syntax	int snd_adsp_control_output2_get(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to output device 2 information
Return value	0	Success
	-EINVAL	Can't get parameter information from ADSP.

3.5.13. snd_adsp_control_input1_get

snd_adsp_control_input1_get		
Synopsis	Get the 1st input source device for Capture info	
Syntax	int snd_adsp_control_input1_get(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to input device 1 information
Return value	0	Success
	-EINVAL	Can't get parameter information from ADSP.

3.5.14. snd_adsp_control_input2_get

snd_adsp_control_input2_get		
Synopsis	Get the 2nd input source device for Capture info	
Syntax	int snd_adsp_control_input2_get(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to input device 2 information
Return value	0	Success
	-EINVAL	Can't get parameter information from ADSP.

3.5.15. snd_adsp_control_eqz_get

snd_adsp_control_eqz_get		
Synopsis	Get parameters info of equalizer control.	
Syntax	int snd_adsp_control_eqz_get(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to equalizer parameters.
Return value	0	Success
	-EINVAL	Can't get parameter information from ADSP.

3.5.16. snd_adsp_control_eqz_switch_get

snd_adsp_control_eqz_switch_get		
Synopsis	Get status of Equalizer control.	
Syntax	int snd_adsp_control_eqz_switch_get(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to equalizer switch.
Return value	0	Success

- ❖ APIs write value to control from user-space

3.5.17. snd_adsp_control_volume_put

snd_adsp_control_volume_put		
Synopsis	Set the PCM volume rate value	
Syntax	int snd_adsp_control_volume_put(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to volume setting value
Return value	0	Hardware parameter is still not changed.
	1	Hardware parameter is changed.
	-EINVAL	Can't set parameter to ADSP.

3.5.18. snd_adsp_control_sample_rate_put

snd_adsp_control_sample_rate_put		
Synopsis	Set sample rate for hardware output/input in the playback/capture case	
Syntax	int snd_adsp_control_sample_rate_put(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to sample rate value
Return value	0	Hardware parameter is still not changed.
	1	Hardware parameter is changed.
	-EINVAL	Can't set parameter to ADSP.

3.5.19. snd_adsp_control_output1_put

snd_adsp_control_output1_put		
Synopsis	Set 1st output destination device for Renderer	
Syntax	int snd_adsp_control_output1_put(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to output device 1 information
Return value	0	Hardware parameter is still not changed.
	1	Hardware parameter is changed.
	-EINVAL	Can't set parameter to ADSP.

3.5.20. snd_adsp_control_output2_put

snd_adsp_control_output2_put		
Synopsis	Set 2nd output destination device for Renderer	
Syntax	int snd_adsp_control_output2_put(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to output device 2 information
Return value	0	Hardware parameter is still not changed.
	1	Hardware parameter is changed.
	-EINVAL	Can't set parameter to ADSP.

3.5.21. snd_adsp_control_input1_put

snd_adsp_control_input1_put		
Synopsis	Set the 1st input source device for Capture	
Syntax	int snd_adsp_control_output1_put(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to input device 1 information
Return value	0	Hardware parameter is still not changed.
	1	Hardware parameter is changed.
	-EINVAL	Can't set parameter to ADSP.

3.5.22. snd_adsp_control_input2_put

snd_adsp_control_input2_put		
Synopsis	Set the 2nd input source device for Capture	
Syntax	int snd_adsp_control_input2_put(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to input device 2 information
Return value	0	Hardware parameter is still not changed.
	1	Hardware parameter is changed.
	-EINVAL	Can't set parameter to ADSP.

3.5.23. snd_adsp_control_eqz_put

snd_adsp_control_eqz_put		
Synopsis	Set equalizer parameter.	
Syntax	int snd_adsp_control_eqz_put(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to equalizer parameters
Return value	0	Hardware parameter is still not changed.
	1	Hardware parameter is changed.
	-EINVAL	Can’t set parameter to ADSP.

3.5.24. snd_adsp_control_eqz_switch_put

snd_adsp_control_eqz__switch_put		
Synopsis	Enable or disable Equalizer control	
Syntax	int snd_adsp_control_eqz_switch_put(struct snd_kcontrol *kcontrol, struct snd_ctl_elem_value *ucontrol)	
Parameter	struct snd_kcontrol *kcontrol	Pointer to control instance
	struct snd_ctl_elem_value *ucontrol	Pointer to equalizer switch
Return value	0	Parameter after setting is still not changed.
	1	Parameter after setting is changed.
	-EINVAL	Can’t enable EQZ when Renderer/Capture plugin is running.

3.6. Detail of APIs for ALSA sound card

3.6.1. snd_adsp_alsa_register

snd_adsp_alsa_register		
Synopsis	This API registers an ALSA soundcard for ADSP device. The card supports both playback and record based on the registered card index. The ADSP ALSA driver's instance is stored in the given platform device data.	
Syntax	int snd_adsp_alsa_register(struct platform_device *pdev, void *ipc)	
Parameter	struct platform_device *pdev	Pointer to platform device structure
	void *ipc	Pointer to proxy IPC data that initialized in ADSP loader
Return value	0	Success
	-ENOMEM	Cannot allocate memory for the sound card
	-EINVAL	Cannot create a sound card Cannot a PCM device Cannot register a control or the soundcard into ALSA framework

3.6.2. snd_adsp_alsa_unregister

snd_adsp_alsa_unregister		
Synopsis	This API frees the registered card and the ADSP ALSA driver instance from the given platform device data.	
Syntax	int snd_adsp_alsa_unregister(struct platform_device *pdev)	
Parameter	struct platform_device *pdev	Pointer to platform device structure
Return value	0	Success
	-ENODEV	Invalid driver data in the platform device.

3.6.3. snd_adsp_alsa_get_ipc

snd_adsp_alsa_get_ipc		
Synopsis	This API gets the proxy IPC data that was stored in ADSP ALSA driver instance.	
Syntax	void * snd_adsp_alsa_get_ipc(struct platform_device *pdev)	
Parameter	struct platform_device *pdev	Pointer to platform device structure
Return value	Pointer	Pointer to stored proxy IPC data
	-ENODEV	Invalid driver data in the platform device.

4. Sequence diagram

4.1. Playback streams flow

4.1.1. Open a playback substream

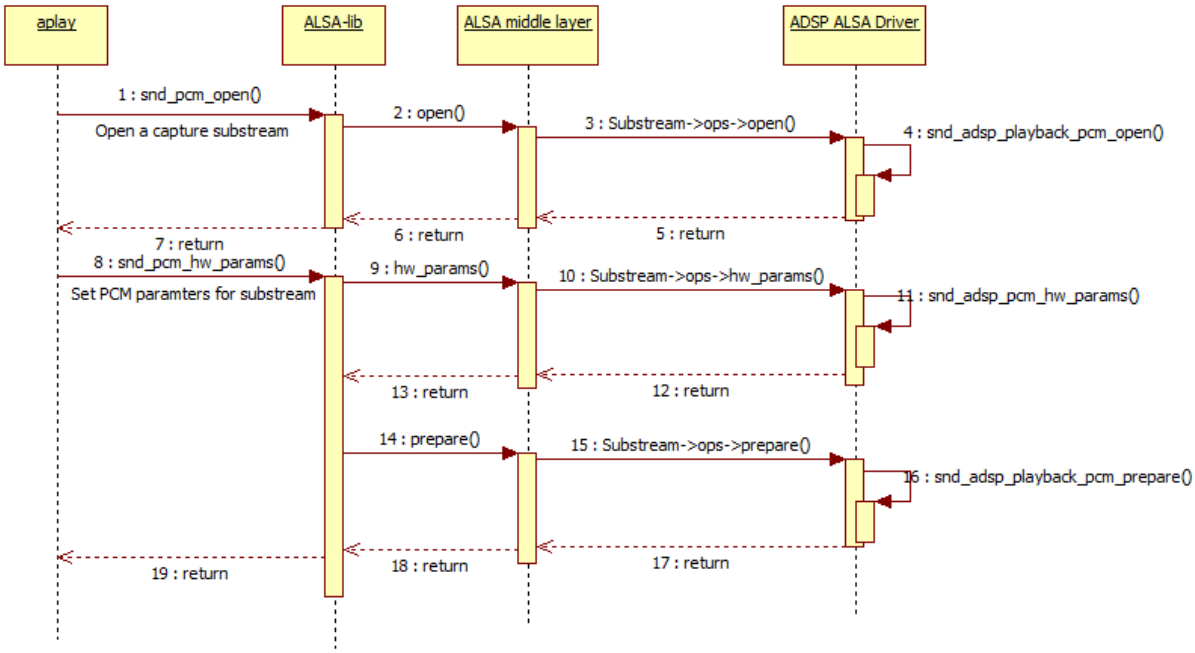

Figure 4-1 Open flow for playback stream

4.1.2. Write data flow

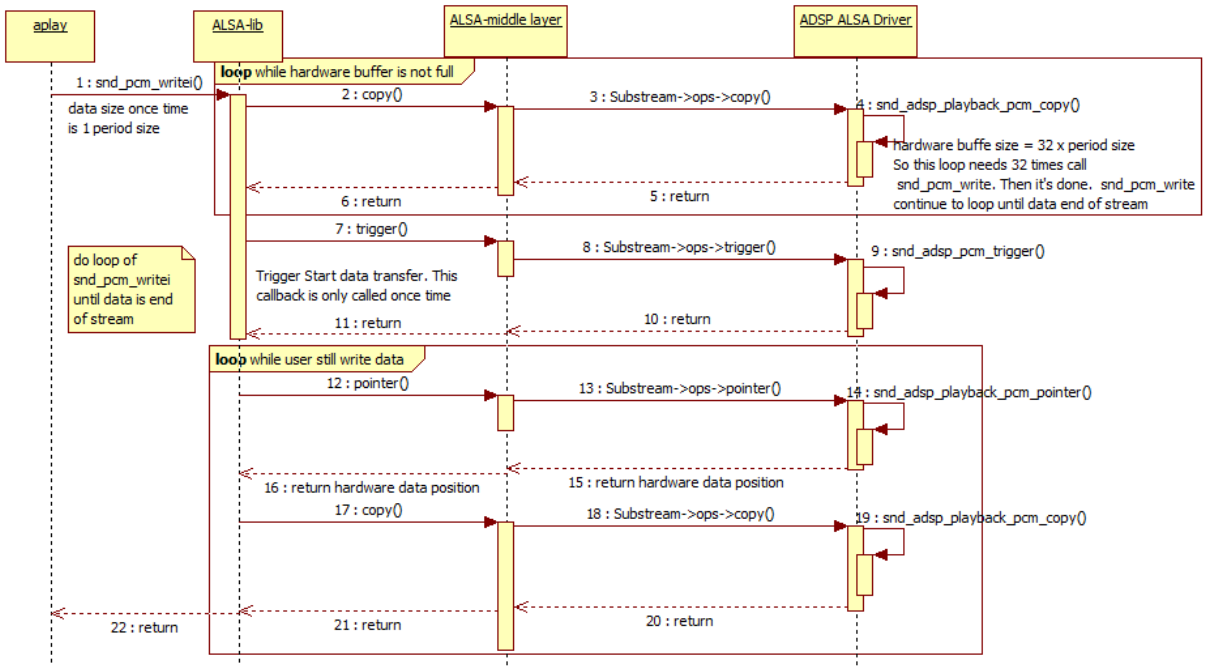

Figure 4-2 Write data flow

4.1.3. Close a playback substream

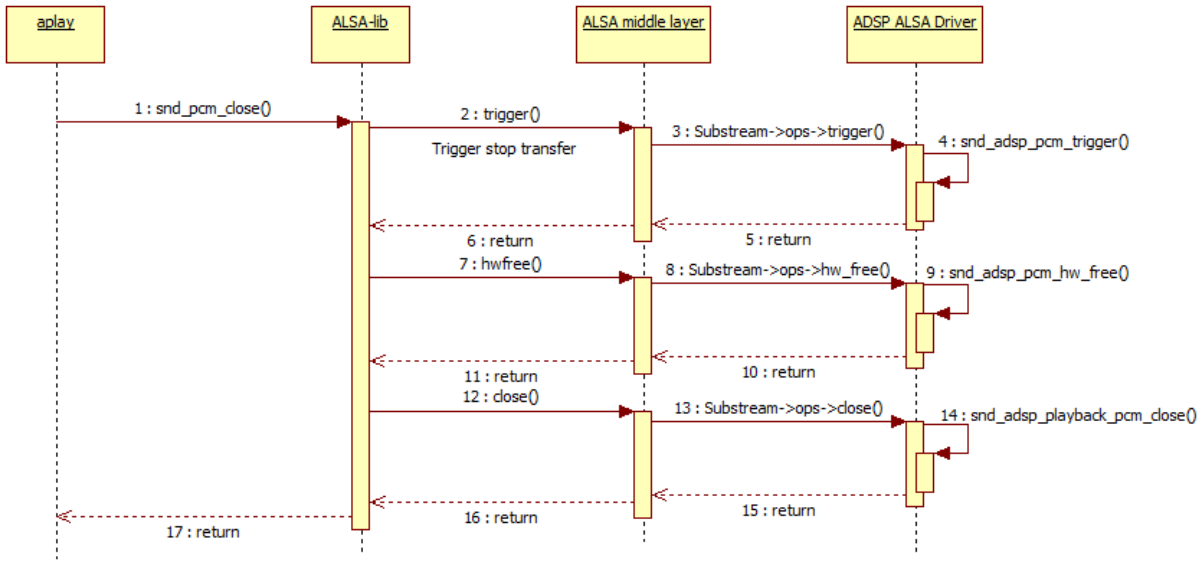

Figure 4-3 Close flow for playback stream

4.2. Capture streams flow

4.2.1. Open a capture substream

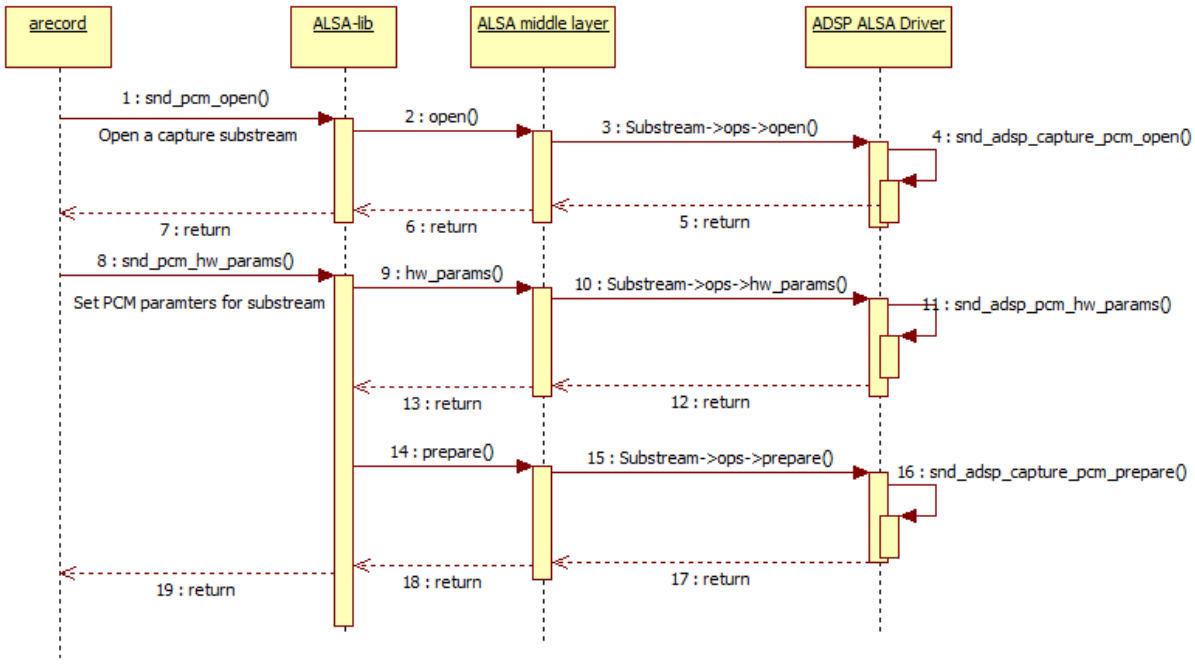

Figure 4-4 Open flow for capture stream

4.2.2. Read data flow

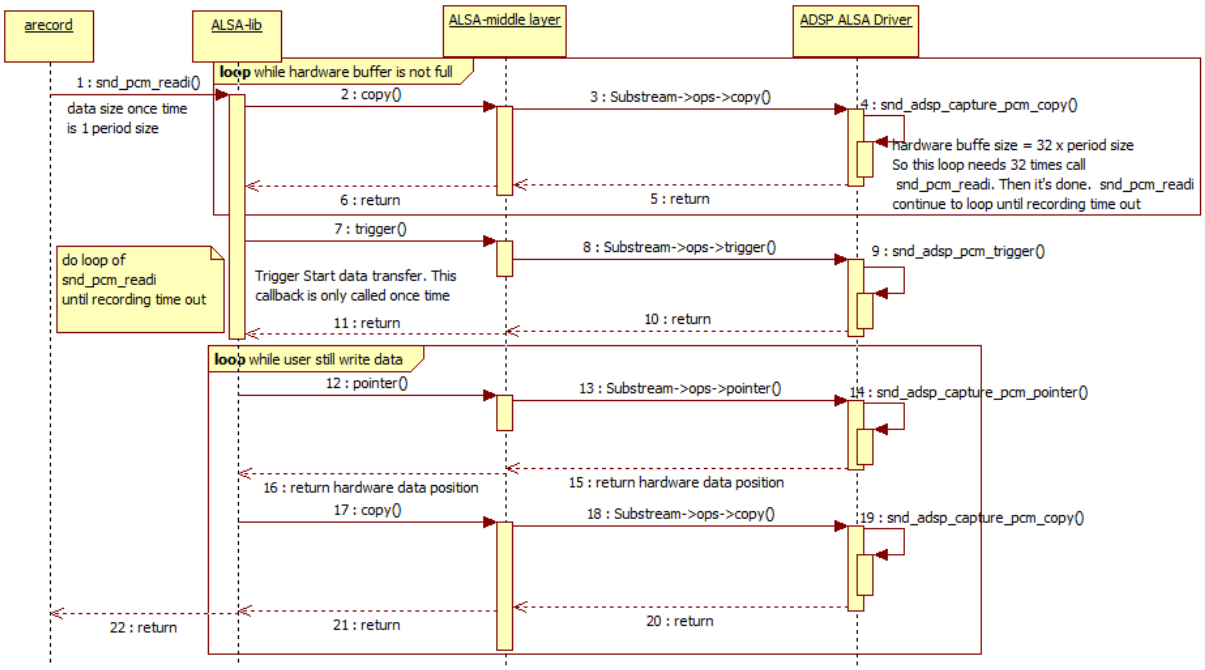

Figure 4-5 Read data flow

4.2.3. Close a capture substream

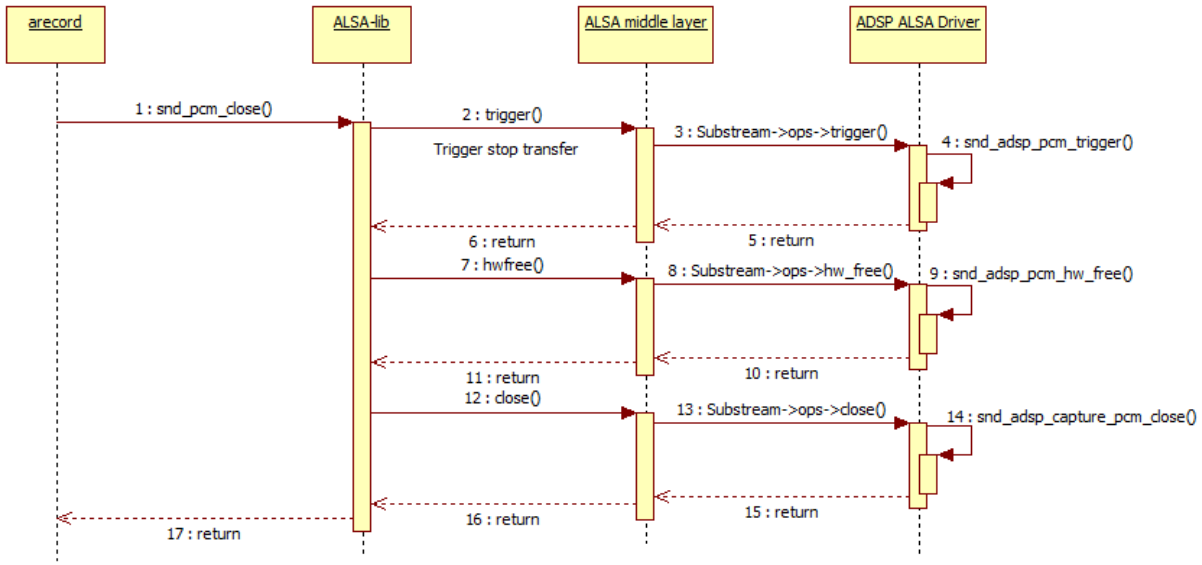

Figure 4-6 Close flow for capture stream

4.3. Volume Control Flow

4.3.1. Get volume value from hardware

In playback case, amixer runs with argument: cget name=**'Playback Volume'**. This string maps with a control defined in ADSP ALSA Driver.

In capture case, amixer runs with argument: cget name=**'Capture Volume'**. This string maps with a control defined in ADSP ALSA Driver.

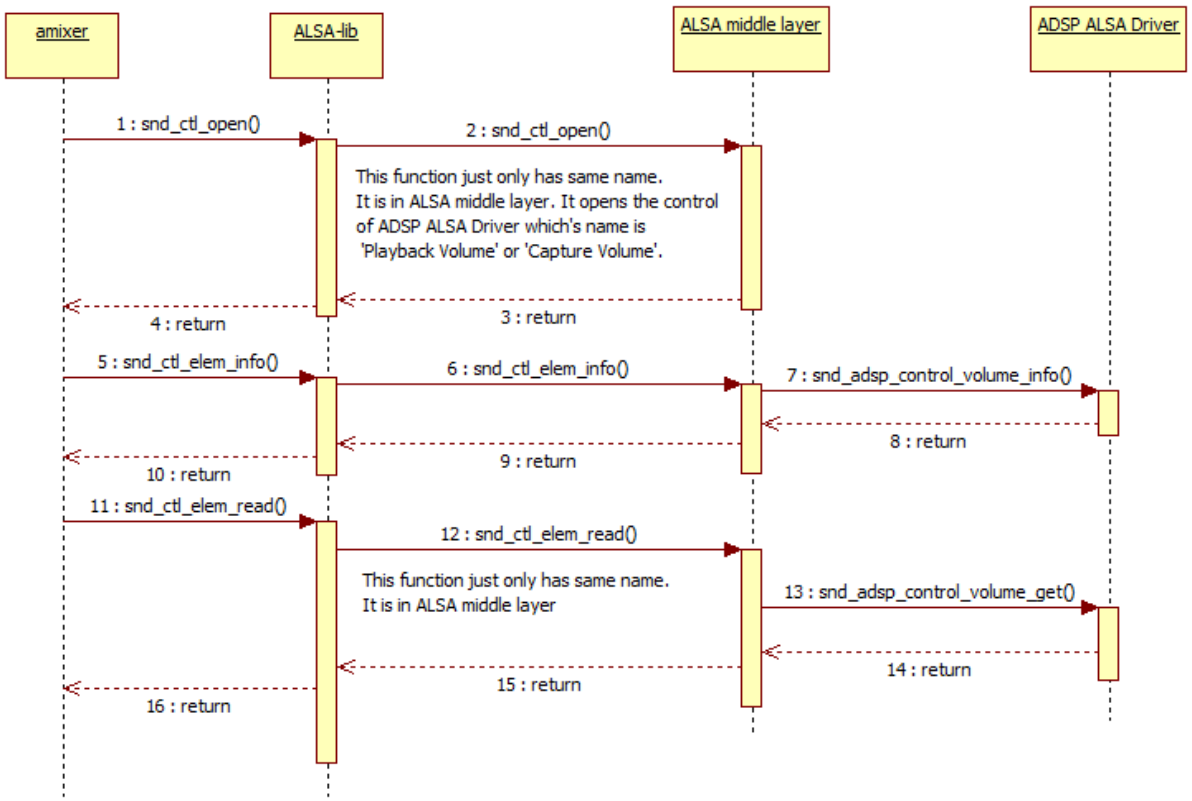

Figure 4-7 Flow of getting volume information from hardware

4.3.2. Set volume value to hardware

In playback case, amixer runs with argument: cset name=**'Playback Volume'** 200 to increase volume to 200% **'Playback Volume'** string maps with a control defined in ADSP ALSA Driver.

In capture case, amixer runs with argument: cset name=**'Capture Volume'** 50 to down volume to 50%. **'Capture Volume'** string maps with a control defined in ADSP ALSA Driver.

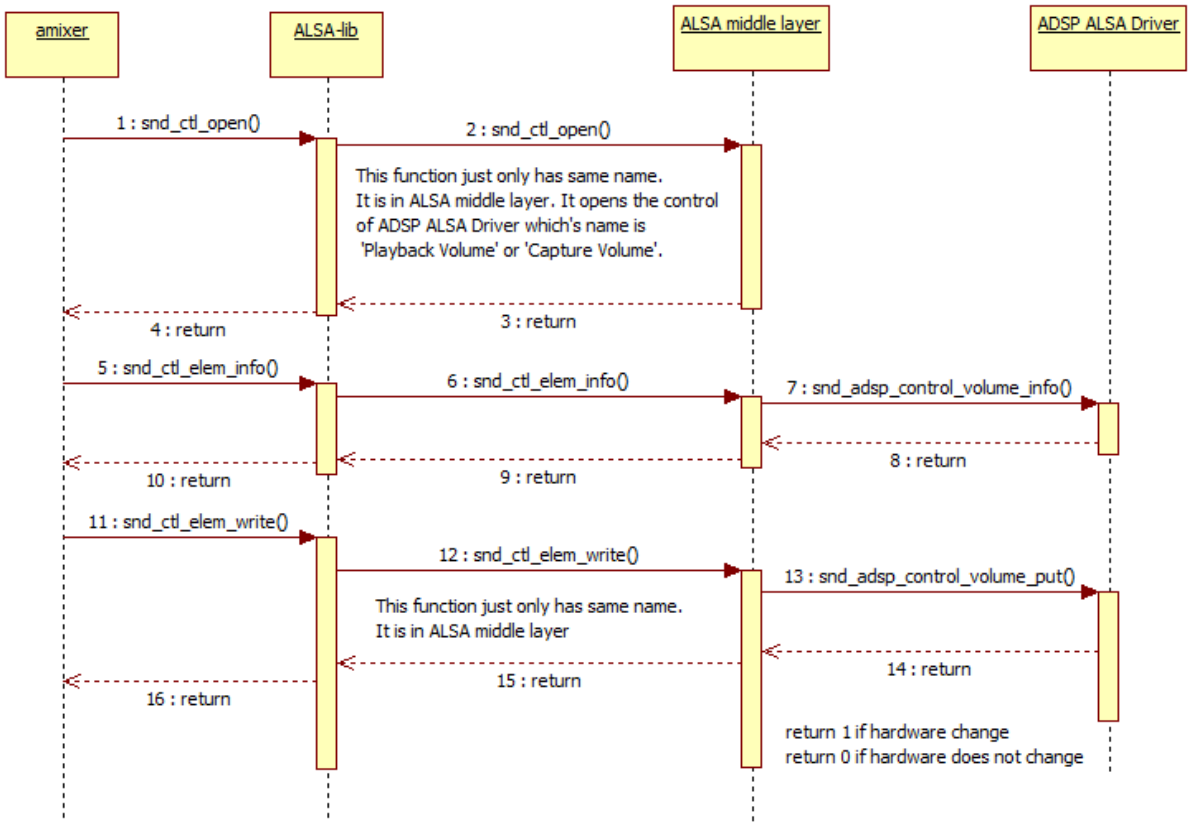

Figure 4-8 Flow of setting volume of hardware

4.4. Sample Rate Converter Control Flow

4.4.1. Get sample rate of hardware

In playback case, amixer runs with argument: cget name=**'Playback Out Rate'**. This string maps with a control defined in ADSP ALSA Driver.

In capture case, amixer runs with argument: cget name=**'Capture In Rate'**. This string maps with a control defined in ADSP ALSA Driver.

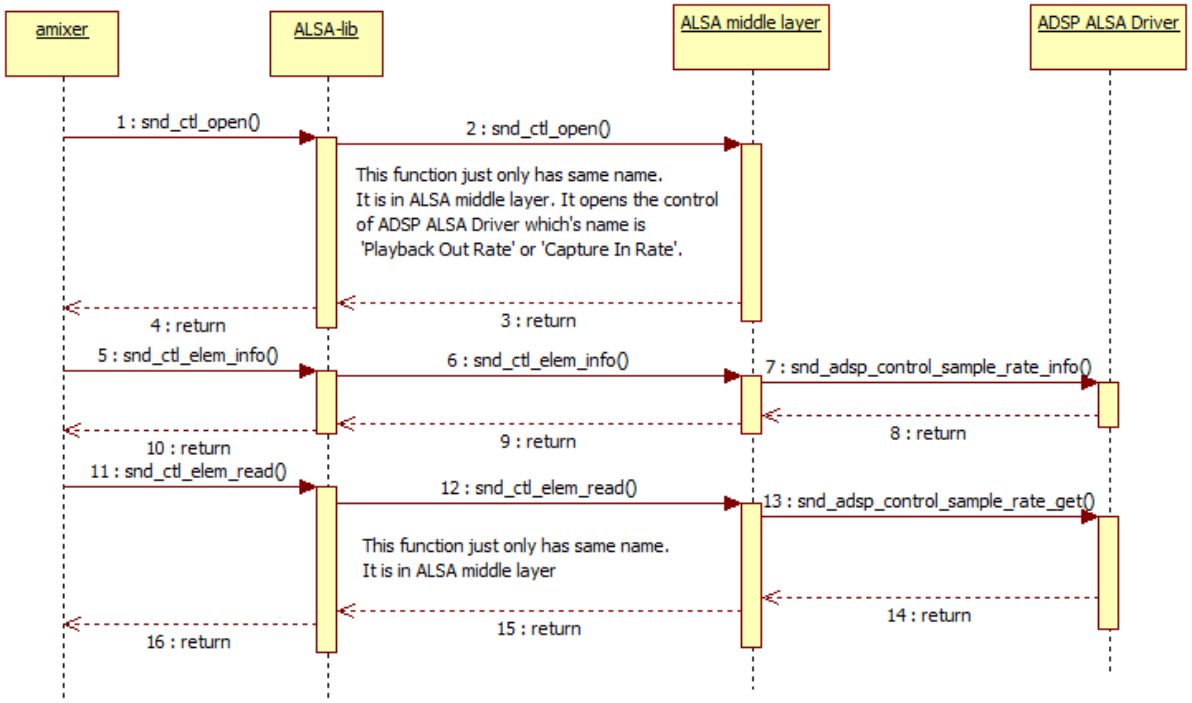

Figure 4-9 Flow of getting sample rate information from hardware

4.4.2. Set sample rate of hardware

In playback case, amixer runs with argument: cset name=**'Playback Out Rate** 44100 to set sample rate of hardware output to 44100 **'Playback Out Rate'** string maps with a control defined in ADSP ALSA Driver.

In capture case, amixer runs with argument: cset name=**'Capture In Rate** 48000 to set sample rate of hardware input to 48000. **'Capture In Rate'** string maps with a control defined in ADSP ALSA Driver.

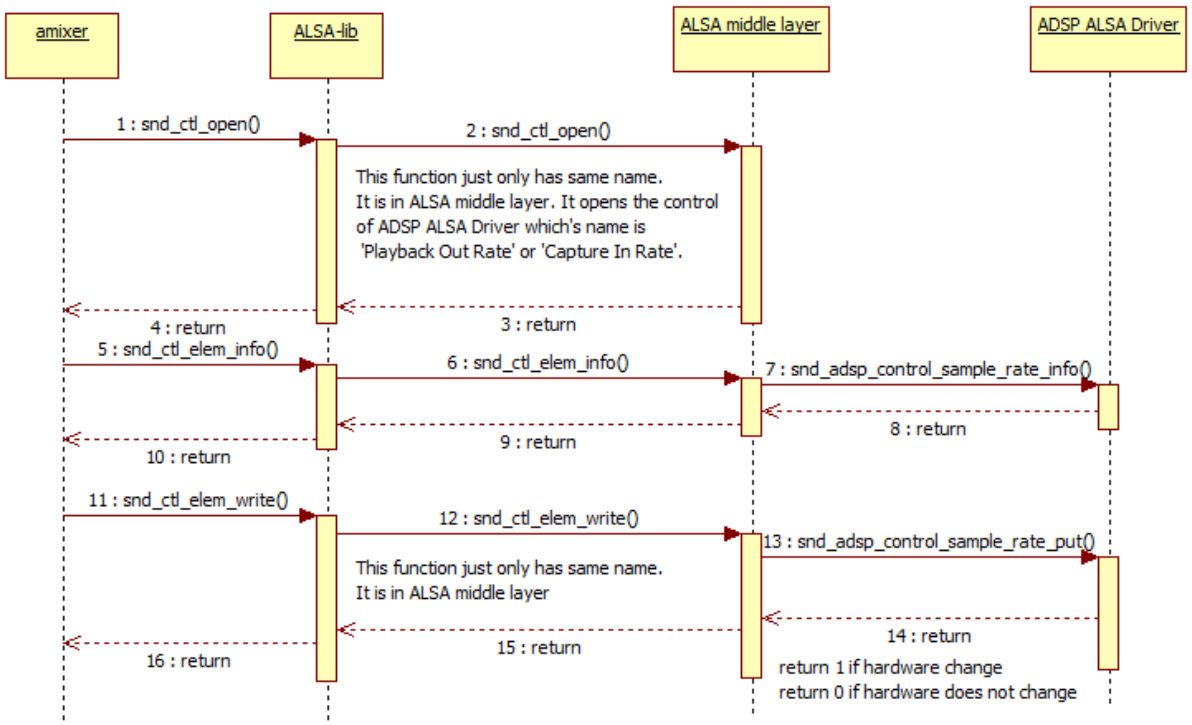

Figure 4-10 Flow of setting sample rate of hardware

4.5. Equalizer Control Flow

4.5.1. Get status of Equalizer control

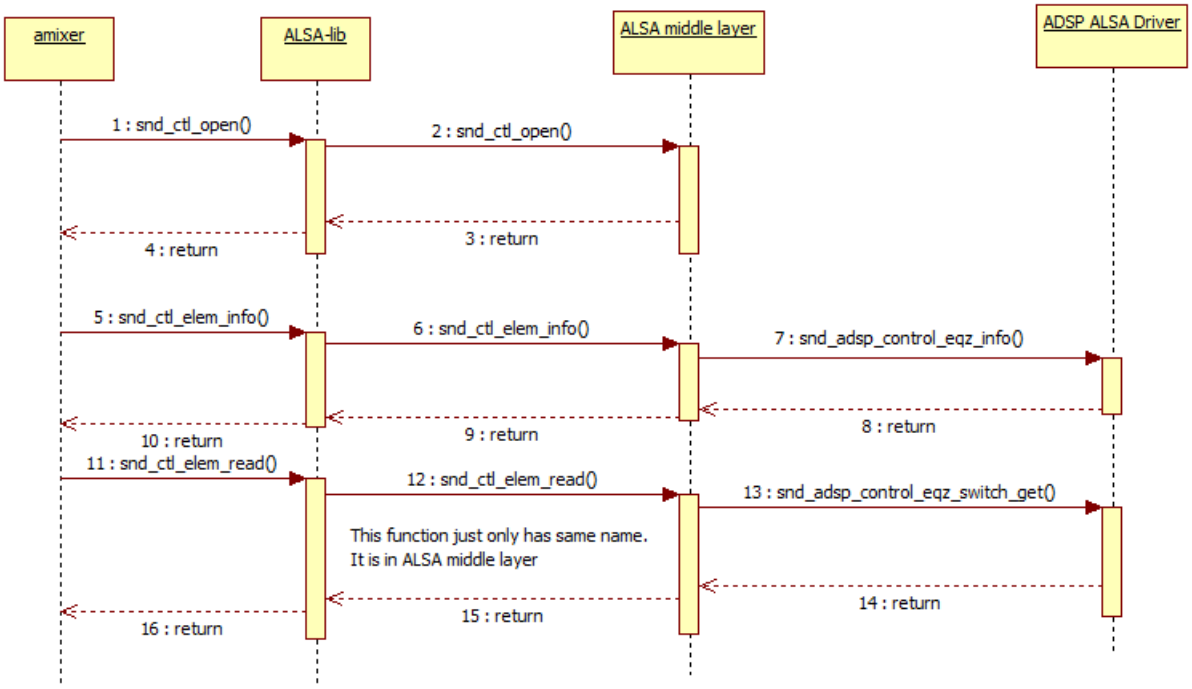

Figure 4-11 Flow of Equalizer control's status getting

4.5.2. Enable Equalizer control

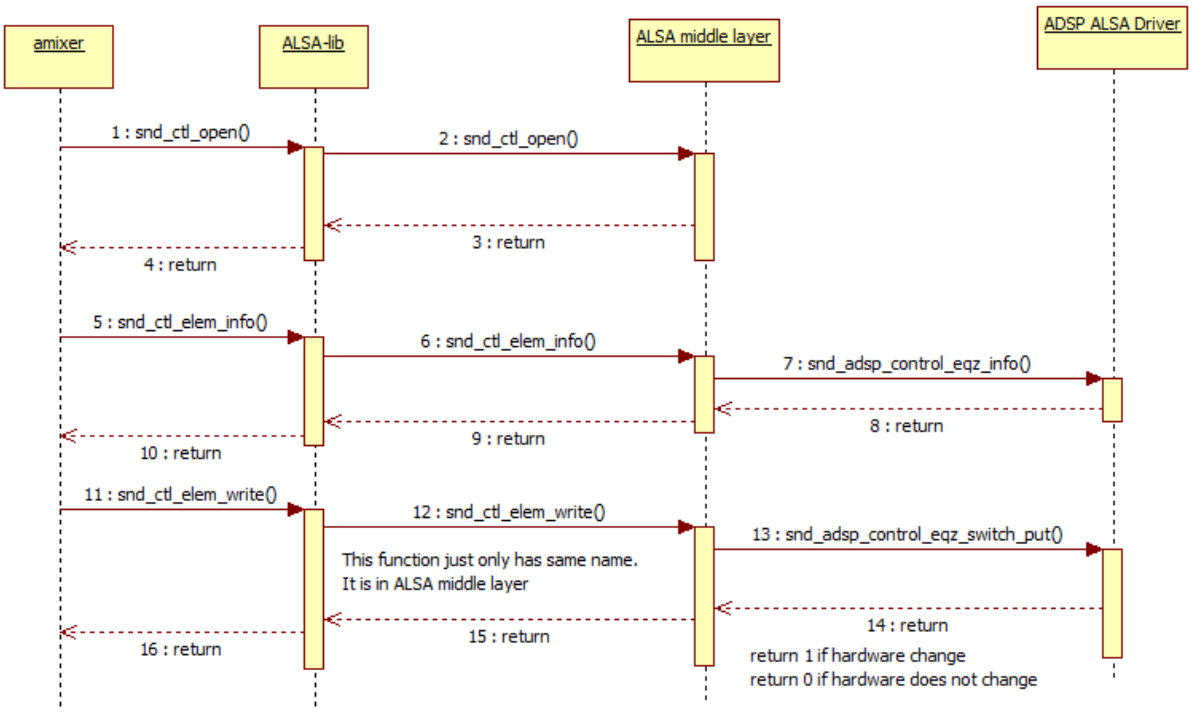

Figure 4-12 Flow of Equalizer activation setting

4.5.3. Get Equalizer parameters of hardware

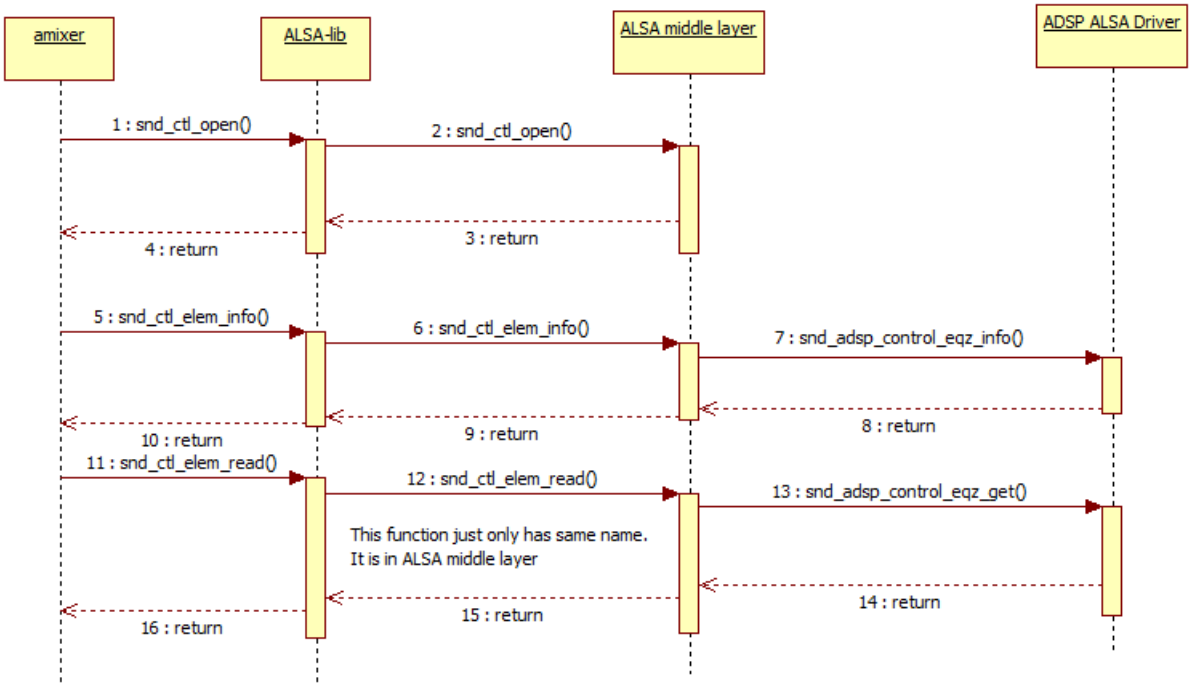

Figure 4-13 Flow of getting Equalizer parameters of hardware

4.5.4. Set Equalizer parameters of hardware

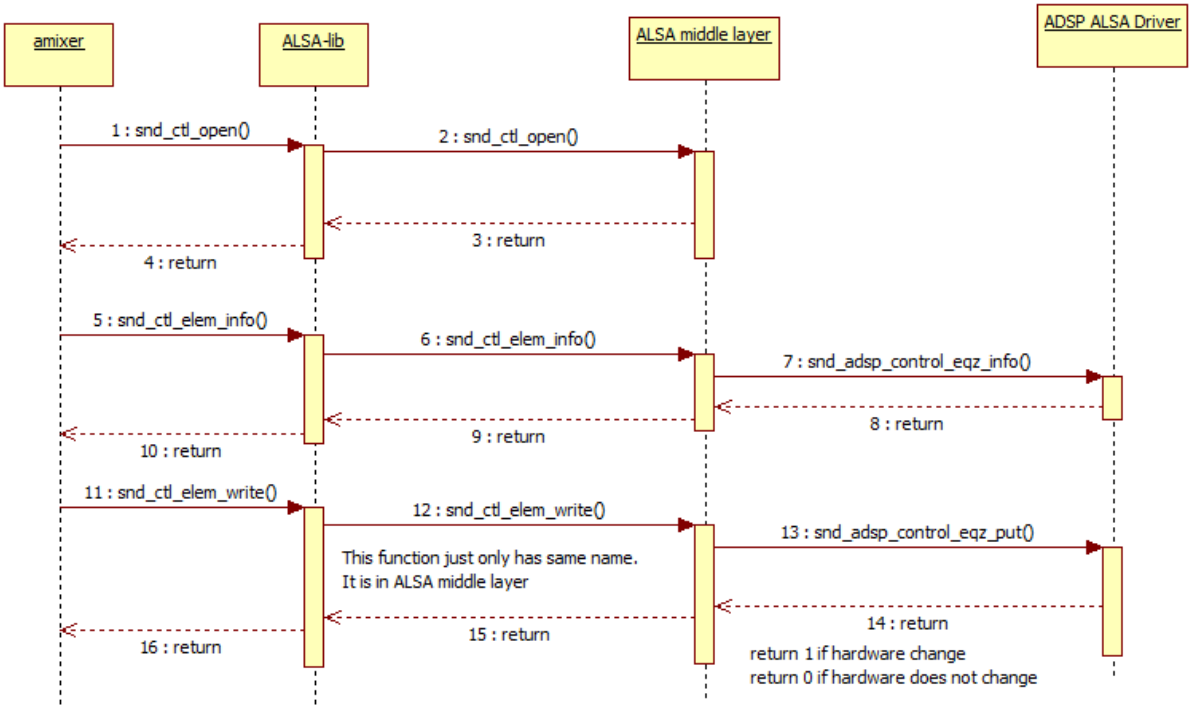

Figure 4-14 Flow of setting Equalizer parameters of hardware

5. Appendix

5.1. Error code

Below table shows error types and corresponding value of callback functions in ALSA Interface

Table 5-1 Error code for ALSA callback functions

Error code	Description	Reference
EINVAL	Invalid argument or some functions get failed	https://elixir.free-electrons.com/linux/v4.0/source/include/uapi/asm-generic/errno-base.h
ENOMEM	Cannot allocate memory	
ENODEV	Invalid driver data in platform device	

5.2. Structure

Below tables shows structure used in this material.

Table 5-2 shows structures are defined in ALSA middle layer.

Structure	Description	Reference
snd_pcm_substreams	It contains elements of PCM substream's information, only one of which is used for above callback functions – **runtime object**. This object contains PCM parameters (pcm width, channel, sample rate, buffer size). Other elements are used by PCM middle layer.	*https://elixir.free-electrons.com/linux/latest/source/include/sound/pcm.h.*
snd_kcontrol	It is used to communicate a control of user space with the control interface on kernel. It contains control index, elements point the get, set, info callback inside the control.	*https://elixir.free-electrons.com/linux/latest/source/include/sound/control.h*
snd_ctl_elem_value	It contains value to set parameter to ADSP or get parameter from ADSP.	*http://elixir.free-electrons.com/linux/v4.2/source/include/uapi/sound/asound.h*
snd_ctl_elem_info	It contains detail information on the control.	*http://elixir.free-electrons.com/linux/v4.3/source/include/uapi/sound/asound.h*

Table 5-3 shows list of structures are defined in ADSP ALSA Driver.

Table 5-3 Structures defined in ADSP ALSA Driver

Structure	Size	Description
snd_adsp_control	456 (bytes)	It is used to store parameters from use
snd_adsp_card	592 (bytes)	It is used to store data for ALSA sound card

5.2.1. snd_adsp_control structure

Table 5-4 snd_adsp_control structure information

Member name		Outline
int	vol_rate[DIRECT_NUM]	Volume rate value
int	sample_rate[DIRECT_NUM]	Out sample rate with Renderer, in sample rate with Capture
xf_adsp_equalizer_params_t	eqz_params[DIRECT_NUM]	Equalizer parameters
int	eqz_switch[DIRECT_NUM]	Equalizer switch

[Note] DIRECT_NUM is a macro presents number of stream can execute concurrently. In this material DIRECT_NUM is 2 means that it support running a playback stream and a capture stream concurrently.

5.2.2. snd_adsp_card structure

Table 5-5 snd_adsp_card structure information

Member name		Outline
xf_adsp_renderer_t	*renderer	Renderer plugin
xf_adsp_equalizer_t	*equalizer[DIRECT_NUM]	Equalizer plugin
xf_adsp_capture_t	*capture	Capture plugin
int	rdr_state	Renderer state
int	eqz_state[DIRECT_NUM]	Equalizer sate
int	cap_state	Capture sate
snd_adsp_control_t	ctr_if	Structure contains parameters information for control
void	*ipc	Pointer to stored proxy IPC data
struct snd_card	*card	Pointer to ALSA sound card object
char	*playback_buffer[XF_BUF_POOL_SIZE]	Data buffer for Playback
char	*capture_buffer[XF_BUF_POOL_SIZE]	Data buffer for Capture
int	phw_idx	HW index for Playback
int	chw_idx	HW index for Capture

[Note] DIRECT_NUM is a macro presents number of stream can execute concurrently. In this material DIRECT_NUM is 2 means that it support running a playback stream and a capture stream concurrently.

XF_BUF_POOL_SIZE is a macro presents number of buffer used to send/receive data. In this material XF_BUF_POOL_SIZE is 4.

Revision History	ADSP Interface for Android Application Note - ALSA -

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

ADSP Driver for Android Application Note - ALSA -

Publication Date: May 30, 2018 Rev. 1.00

Published by: Renesas Electronics Corporation

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

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

Renesas Electronics 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_ALSAIF

Renesas Electronics Corporation