Application Note

RENESAS

CONFIDENTIAL

# ADSP Driver for Android
# RCG3AHPDA9001ZDO

## Application Note - ALSA -

RCG3AHPDA9001ZDOE_AN_ALSAIF



Rev. 1.00 May, 2019

Renesas Electronics
www.renesas.com

# 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

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 platform interface.
- APIs for platform driver.

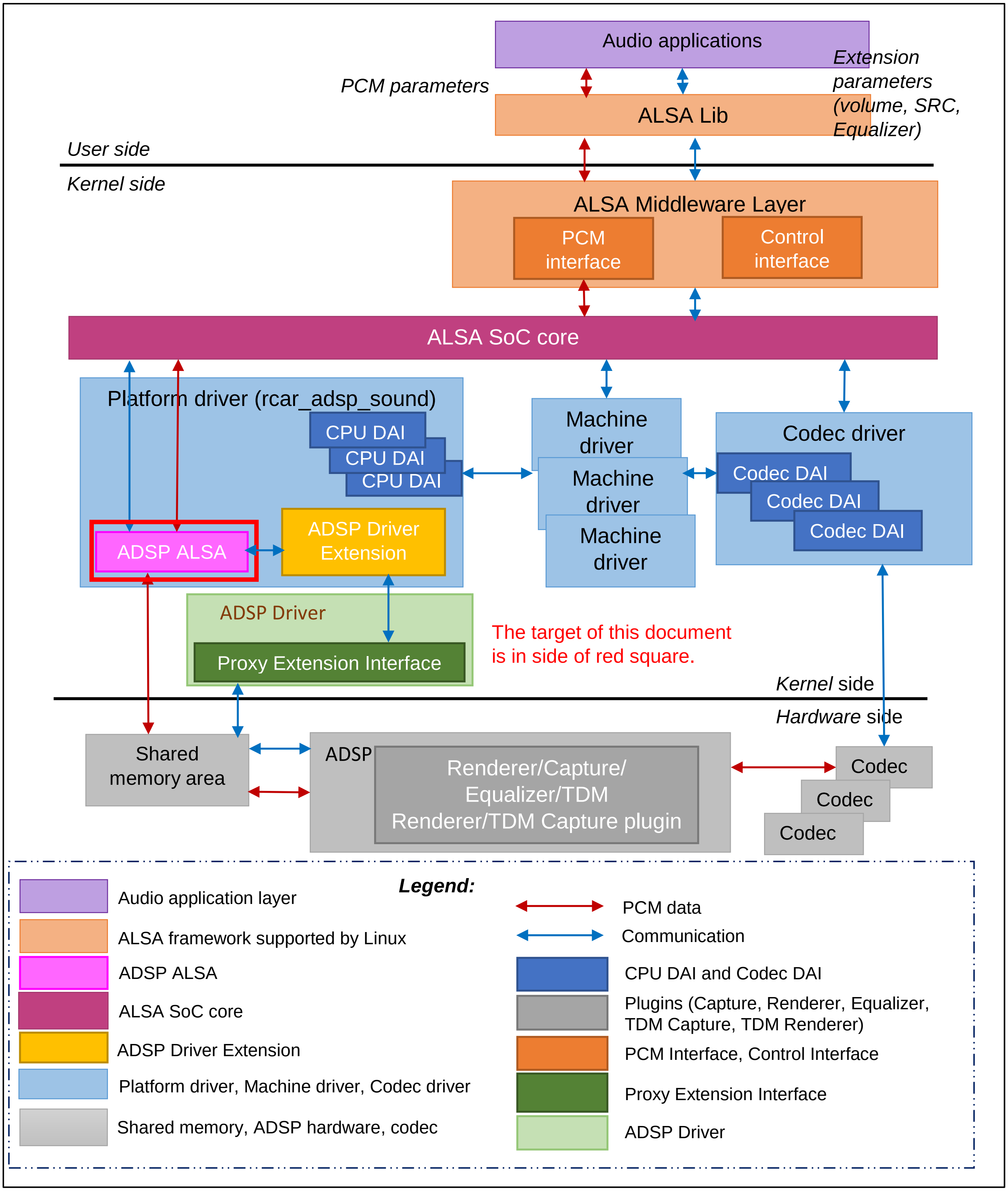



Figure 1-1 Overview architecture of ADSP driver.

- **Audio applications (tinycap, tinyplay, tinymix, etc)**:
  The user applications that support to play or record sound by using ALSA library.

- **ALSA Lib**:
  The ALSA library APIs are the interface to the ALSA drivers.

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

- **ALSA SoC core:**
  It is part of ALSA Framework and does processing of PCM data

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

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

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

- **Codec driver**:
  It represents interface for codecs.

- **Codec DAI**:
  The DAI for codecs to communicate with other drivers

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

- **Proxy Extension Interface**:
  APIs of methods through which ADSP Driver Extension communicates with shared memory area in Hardware side.

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

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

## 1.1. Device Tree

Below table describes which DTS files need to be considered to update when using ADSP sound driver for playback/record.

As Salvator-X/XS and Ebisu does not support TDM multi-channel, it is checked only in Starter-Kit and Kingfisher environment. Since the Starter-Kit and Kingfisher environment are not supported by the standard, it is necessary for users to prepare by yourself.

Table 1-1 Device tree files

| Target CPU | Target board | Device tree files (for example : Yocto v3.15.0) |
|---|---|---|
| R-Car H3 | Salvator-X/XS | arch/arm64/boot/dts/renesas/r8a7795-es1-salvator-x.dts<br>arch/arm64/boot/dts/renesas/r8a7795-salvator-x.dts<br>arch/arm64/boot/dts/renesas/r8a7795-salvator-xs.dts<br><br>each include below:<br>arch/arm64/boot/dts/renesas/r8a7795.dtsi<br>arch/arm64/boot/dts/renesas/salvator-common.dtsi |
| R-Car M3 | Salvator-X/XS | arch/arm64/boot/dts/renesas/r8a7796-salvator-x.dts<br>arch/arm64/boot/dts/renesas/r8a7796-salvator-xs.dts<br><br>each include below:<br>arch/arm64/boot/dts/renesas/r8a7796.dtsi<br>arch/arm64/boot/dts/renesas/salvator-common.dtsi |
| R-Car M3N | Salvator-X/XS | arch/arm64/boot/dts/renesas/r8a77965-salvator-x.dts<br>arch/arm64/boot/dts/renesas/r8a77965-salvator-xs.dts<br><br>each include below:<br>arch/arm64/boot/dts/renesas/r8a77965.dtsi<br>arch/arm64/boot/dts/renesas/salvator-common.dtsi |
| R-Car E3 | Ebisu | arch/arm64/boot/dts/renesas/r8a77990-es10-ebisu.dts<br><br>include below:<br>arch/arm64/boot/dts/renesas/r8a77990.dtsi |

Table 1-2 Required property for ADSP sound node

| Property name | Value |
|---|---|
| compatible | "renesas,rcar_adsp_sound_gen3" |

❖ Example setting route:

a) Use ADSP sound with MIX function on Salvator-XS/H3:

- Select Machine driver: Simple SCU Card or Audio Graph SCU Card which support MIX function. This example will use Audio Graph SCU Card for illustration.
  (Refer to *Documentation/devicetree/bindings/sound/simple-scu-card.txt* or *Documentation/devicetree/bindings/sound/audio-graph-scu-card.txt* for more detail setting of sound card).
- Select Codec driver: AK4613 codec which supports I2C format and controls audio chip on Salvator-XS/H3 board.
  (Refer to *Documentation/devicetree/bindings/sound/ak4613.txt* for more information)

- Add ADSP sound node to device tree.
  The node is placed in the root node of *arch/arm64/boot/dts/renesas/r8a7795.dtsi*

```
/ {
        …
        rcar_adsp_sound: adsp_sound {
                compatible = "renesas,rcar_adsp_sound_gen3";
                status = "disabled";
        };
        …
};
```

- Add routing between ADSP sound driver and codec driver.
  Update routing configuration in *arch/arm64/boot/dts/renesas/salvator-common.dtsi*

Define DAI indexes for each Mixing port, each DAI connects to AK4613's DAI as below:

```
&rcar_adsp_sound {
	status = "okay";
	/* Multiple DAI */
	#sound-dai-cells = <1>;

	ports {
		#address-cells = <1>;
		#size-cells = <0>;
		adsp_port0: port@0 {
			reg = <0>;
			adsp_endpoint0: endpoint {
				remote-endpoint = <&ak4613_endpoint0>;
				dai-format = "left_j";
			};
		};
		adsp_port1: port@1 {
			reg = <1>;
			adsp_endpoint1: endpoint {
				remote-endpoint = <&ak4613_endpoint1>;
				dai-format = "left_j";
			};
		};
		adsp_port2: port@2 {
			reg = <2>;
			adsp_endpoint2: endpoint {
				remote-endpoint = <&ak4613_endpoint2>;
				dai-format = "left_j";
			};
		};
		adsp_port3: port@3 {
			reg = <3>;
			adsp_endpoint3: endpoint {
				remote-endpoint = <&ak4613_endpoint3>;
				dai-format = "left_j";
			};
		};
	};
};
```

Keep rcar-sound card connects to AK4613 codec with 4th end point:

```
&rcar_sound {
	…

	ports {
		…
		rsnd_port0: port@0 {
			reg = <0>;
			rsnd_endpoint0: endpoint {
				remote-endpoint = <&ak4613_endpoint4>;

	…
};
```

The codec DAIs are also connect to ADSP's DAIs, and rcar-sound's DAI:

```
ak4613: codec@10 {
	compatible = "asahi-kasei,ak4613";
	…

	ports {
		#address-cells = <1>;
		#size-cells = <0>;
		ak4613_endpoint0: endpoint@0 {
			reg = <0>;
			remote-endpoint = <&adsp_endpoint0>;
		};
		ak4613_endpoint1: endpoint@1 {
			reg = <1>;
			remote-endpoint = <&adsp_endpoint1>;
		};
		ak4613_endpoint2: endpoint@2 {
			reg = <2>;
			remote-endpoint = <&adsp_endpoint2>;
		};
		ak4613_endpoint3: endpoint@3 {
			reg = <3>;
			remote-endpoint = <&adsp_endpoint3>;
		};
		ak4613_endpoint4: endpoint@4 {
			reg = <4>;
			remote-endpoint = <&rsnd_endpoint0>;
		};
	};
};
```

Update sound node to routing ADSP sound driver and codec AK4613 driver:

```
sound_card: sound {
        compatible = "audio-graph-scu-card";

        label = "rcar-sound";

        prefix = "ak4613";
        routing = "ak4613 Playback", "Playback0",
                        "ak4613 Playback", "Playback1",
                        "ak4613 Playback", "Playback2",
                        "ak4613 Playback", "Playback3",
                        "Capture0", "ak4613 Capture",
                        "Capture1", "ak4613 Capture",
                        "Capture2", "ak4613 Capture",
                        "Capture3", "ak4613 Capture",
                        "ak4613 Playback", "DAI0 Playback",
                        "DAI0 Capture", "ak4613 Capture";

        dais = <&adsp_port0
                &adsp_port1
                &adsp_port2
                &adsp_port3
                &rsnd_port0>;
};
```

[Note]

- If current kernel-source configured HDMI audio for rcar-sound. It means rcar-sound is routed to HDMI codec, and AK4613 codec. In such cases, integrating ADSP sound is done by removing HDMI audio from rcar-sound.

For example, remove below content in *arch/arm64/boot/dts/renesas/r8a7795-salvator-xs.dts*

```
&sound_card {
        dais = <&rsnd_port0     /* ak4613 */
                &rsnd_port1     /* HDMI0  */
                &rsnd_port2>;   /* HDMI1  */
};
```

- The device index of R-Car sound (*rcar_sound*) changed to 4 as above configurations.

So, it is necessary to add the index number when setting its control.

b) Setting device, DMA and MIX control in device tree:

- By default, MIX function is disabled. The device parameters (devices and DMA) are assigned default values shown in the table below:

Table 1-3 Default value of devices for DAIs in playback direction

| | 1st device | DMA type for 1st device | 2nd device | DMA type for 2nd device |
|---|---|---|---|---|
| DAI-0 | SRC0 | ADMAC0 | SSI0 | PDMA0 |
| DAI-1 | SRC0 | ADMAC0 | SSI0 | PDMA0 |
| DAI-2 | SRC0 | ADMAC0 | SSI0 | PDMA0 |
| DAI-3 | SRC0 | ADMAC0 | SSI0 | PDMA0 |
| DAI-4 | SRC0 | ADMAC0 | SSI3 | PDMA0 |

Table 1-4 Default value of devices for DAIs in capture direction

| | 1st device | DMA type for 1st device | 2nd device | DMA type for 2nd device |
|---|---|---|---|---|
| DAI-0 | SRC1 | ADMAC0 | SSI1 | PDMA0 |
| DAI-1 | SRC1 | ADMAC0 | SSI1 | PDMA0 |
| DAI-2 | SRC1 | ADMAC0 | SSI1 | PDMA0 |
| DAI-3 | SRC1 | ADMAC0 | SSI1 | PDMA0 |
| DAI-4 | SRC1 | ADMAC0 | SSI4 | PDMA0 |

- To set paramters for devices, DMA and MIX control, update configuration in the content of node &rcar_adsp_sound

Table 1-5 Required nodes for setting HW devices and route

| Node | Description |
|---|---|
| device_params | Node whose content contains parameters of HW and DMAs for DAIs to set. Its parent node is &rcar_adsp_sound. |
| dai-x<br>(x = 0, 1, 2, 3, 4) | DAIx's route info. This is the sub-node of device_params<br>x = 0, 1, 2, 3: These DAI indexes for playback and capture stream<br>x = 4: This DAI index is for TDM |
| playback | Playback direction info of a DAI. Sub-node of dai-x |
| capture | Record direction info of a DAI. Sub-node of dai-x |

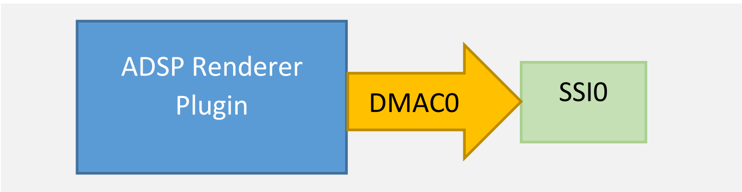
Table 1-6 Required properties for DAI's HW parameters

| Properties | Description | Values | |
|---|---|---|---|
| dev | HW module | "src-x" | SRCx (x = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9)<br>[Note] Only SRC 0,1,3,4 are supported for dai-4 (TDM). Others SRCs are unavailable. for it |
| | | "ssi-x" | SSIx (x = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9) |
| dma | DMA type | "pdma-x" | PDMAx (x = 0, 1, 2, …, 28) |

| | | | |
|---|---|---|---|
| | | | [Note] PDMA is not supported for the 1st device. Only selected for 2nd device. |
| | | “dmac-x” | ADMACx (x = 0, 1, 2, …, 31) |
| mix_usage | Turn on MIX function. This is just only used in playback. | - | |

Examples:

- Set DAI0’s playback without MIX control turned on:

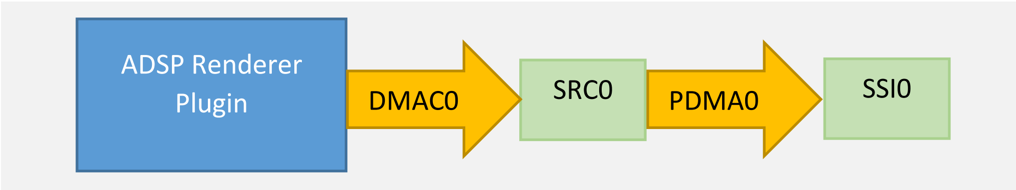



```
&rcar_adsp_sound {
        status = "okay";
        /* Multiple DAI */
        #sound-dai-cells = <1>;

        device_params {
                dai-0 {
                        playback {
                                dev = “ssi-0”;
                                dma = “dmac-0”;
                        };
                };
        };

        …
};
```

- Set DAI0's playback without MIX control turned on:

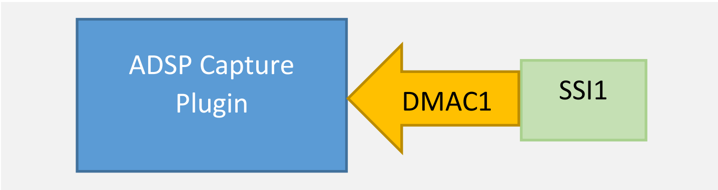



```
&rcar_adsp_sound {
        status = "okay";
        /* Multiple DAI */
        #sound-dai-cells = <1>;

        device_params {
                dai-0 {
                        playback {
                                dev = “src-0”, “ssi-0”;
                                dma = “dmac-0”, “pdma-0”;
                        };
                };
        };

        ...
};
```

- Set DAI0’s Capture as below:

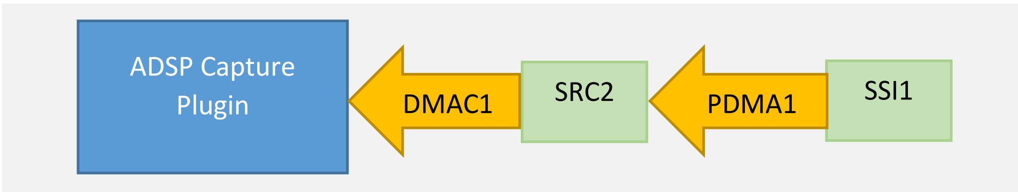



```
&rcar_adsp_sound {
	status = "okay";
	/* Multiple DAI */
	#sound-dai-cells = <1>;

	device_params {
		dai-0 {
			capture {
				dev = “ssi-1”;
				dma = “dmac-1”;
			};
		};
	};

	….
};
```

- Set DAI0's Capture as below:

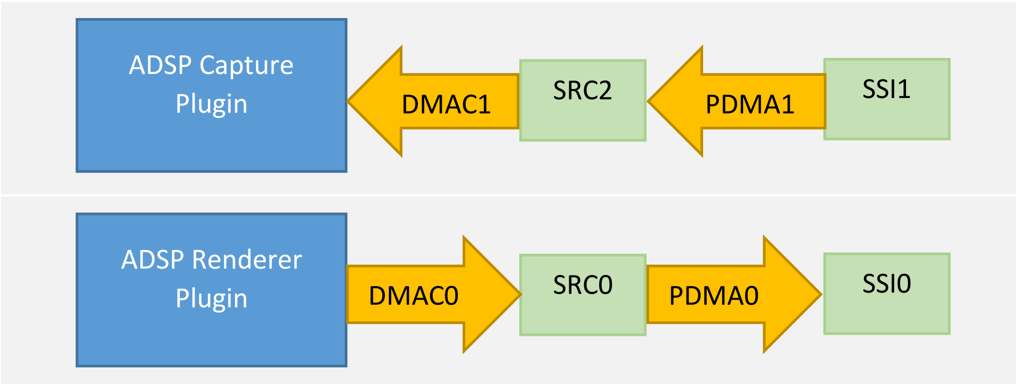



```
&rcar_adsp_sound {
        status = "okay";
        /* Multiple DAI */
        #sound-dai-cells = <1>;

        device_params {
                dai-0 {
                        capture {
                                dev = "src-2", "ssi-1";
                                dma = "dmac-1", "pdma-1";
                        };
                };
        };

        ....
};
```

- Set DAI0 for playback and capture as below:



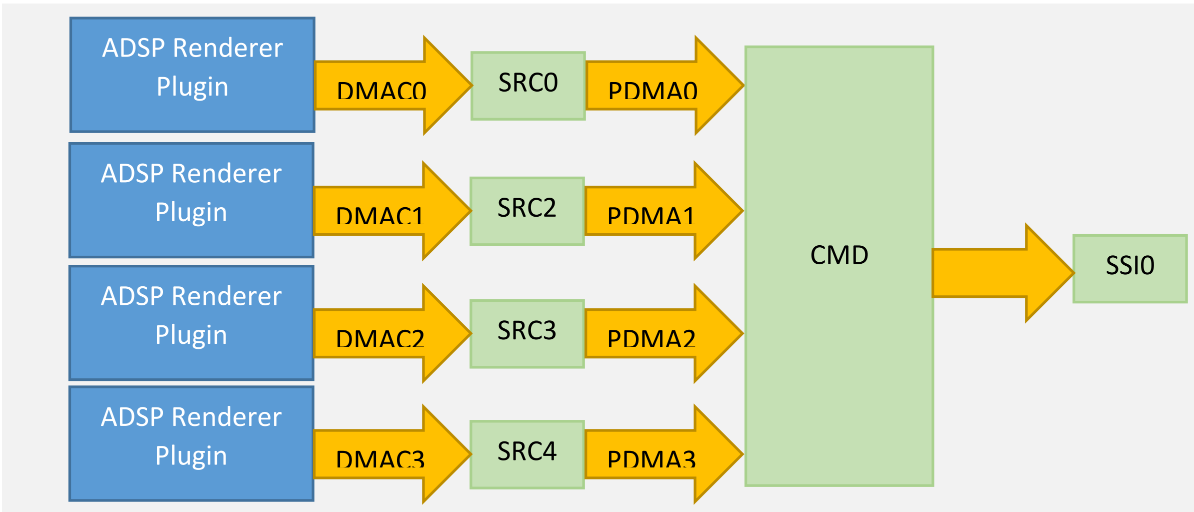

```
&rcar_adsp_sound {
        status = "okay";
        /* Multiple DAI */
        #sound-dai-cells = <1>;

        device_params {
                dai-0 {
                        playback {
                                dev = “src-0”, “ssi-0”;
                                dma = “dmac-0”, “pdma-0”;
                        };

                        capture {
                                dev = “src-2”, “ssi-1”;
                                dma = “dmac-1”, “pdma-1”;
                        };
                };
        };

        ....
};
```

- Set multi-DAIs as below using MIX for playback:



```
&rcar_adsp_sound {
	status = "okay";
	/* Multiple DAI */
	#sound-dai-cells = <1>;

	device_params {
		dai-0 {
			playback {
				dev = “src-0”, “ssi-0”;
				dma = “dmac-0”, “pdma-0”;
				mix_usage;
			};
		};
		dai-1 {
			playback {
				dev = “src-2”, “ssi-0”;
				dma = “dmac-1”, “pdma-1”;
				mix_usage;
			};
		};
		dai-2 {
			playback {
				dev = “src-3”, “ssi-0”;
				dma = “dmac-2”, “pdma-2”;
				mix_usage;
			};
		};
		dai-3 {
			playback {
				dev = “src-4”, “ssi-0”;
				dma = “dmac-3”, “pdma-3”;
				mix_usage;
			};
		};
	};
```

# 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.

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

| Number | API functions | Description |
|---|---|---|
| 1 | snd_adsp_pcm_open | Register a Capture/Renderer plugin or a TDM Capture/Renderer plugin. It also registers Equalizer plugin in case of Capture/Renderer if Equalizer is used. It also gets range of hardware parameter into substream. |
| 2 | snd_adsp_pcm_close | Unregister Capture/Renderer plugin or TDM Capture/Renderer plugin. It also unregisters Equalizer plugin in case of Capture/Renderer if Equalizer is used. |
| 3 | snd_adsp_pcm_hw_params | This callback is used to allocate buffer pool for data transfer. In Capture/Renderer, it maps ALSA buffer to shared memory. In TDM, it allocates ALSA buffer to transfer data. |
| 4 | snd_adsp_pcm_hw_free | This callback is used to deallocate ALSA buffer in TDM. In Capture/Renderer, this callback is just dummy. |
| 5 | snd_adsp_pcm_prepare | This callback helps prepare necessary parameters to set to the plugin before it is ready. If user do not set volume, the volume will get the default value of 100%. In Capture/Renderer, if Equalizer is used, it will route Capture/Renderer to Equalizer, otherwise, it requests to map shared memory to data buffer in the plugin.<br>On the occasion of data overrun or underrun error occurrence, this callback waits until all the buffers return. In playback/record case, without Equalizer, it changes the state of the component to reset. |
| 6 | snd_adsp_pcm_trigger | Start, stop, resume, suspend pcm substream.<br>In Capture/TDM Capture, when it does not running, this callback kicks init with Start/Resume command. When it is running, Start/Resume command is sent in case of overrun/underrun occurrence. |
| 7 | snd_adsp_pcm_ack | This callback is called in read/write operation and in Renderer/TDM Renderer when it starts or resumes PCM substream. Then: |

| | | |
|---|---|---|
| | | In TDM Capture/TDM Renderer, it copies data from/to DMA buffer to/from ALSA buffer, respectively.<br>In Renderer/Capture, it only get the current DMA buffer for data transfer.<br>The DMA buffer is then submitted to ADSP side with EMPTY_THIS_BUFFER (Renderer/TDM Renderer) or FILL_THIS_BUFFER(Capture/TDM Capture). |
| 8 | snd_adsp_pcm_pointer | Update HW buffer position and return the position of the offset on hardware buffer in sample unit |
| 9 | snd_adsp_pcm_mmap | This callback is used to map kernel memory to user space for use. |

## 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_rdr_out_channel_info | Get detail information on Renderer output channel control |
| 4 | snd_adsp_control_volume_get | Get volume setting value |
| 5 | snd_adsp_control_sample_rate_get | Get sample rate output setting value |
| 6 | snd_adsp_control_rdr_out_channel_get | Get Renderer output channel setting value |
| 7 | snd_adsp_control_volume_put | Set volume value |
| 8 | snd_adsp_control_sample_rate_put | Set sample rate output value |
| 9 | snd_adsp_control_rdr_out_channel_put | Set Renderer output channel |

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 for ASoC Platform interface

Below table shows APIs used for ASoC Platform interface

Table 3-4 List of API functions for ASoC Platform interface

| Number | API function | Description |
|---|---|---|
| 1 | snd_adsp_pcm_new | This API registers necessary control interfaces for ADSP soundcard based on CPU DAI type (playback/capture type or TDM playback/TDM capture type). In playback/capture case, when registering control interfaces for the second/third/fourth playback/capture. In TDM, it also pre-allocates a memory region for ALSA buffer for transferring data. |

## 3.4 APIs for for ASoC Platform driver

Below table shows APIs used for ASoC Platform driver

Table 3-5 List of API for ASoC Platform driver

| Number | API function | Description |
|---|---|---|
| 1 | snd_adsp_probe | Register platform driver with 5 CPU DAIs (4 DAIs for playback/record and 1 DAI for TDM playback/TDM record) and add ADSP sound card component to ASoC framework. It also assign default values for device parameters, DMA parameters for DAIs. It also parses values for DAIs defined in device tree. |
| 2 | snd_adsp_remove | Unregister platform driver and remove ADSP sound card component from ASoC framework. |

## 3.5 Detail of APIs for PCM Interface

### 3.5.1 snd_adsp_pcm_open

| snd_adsp_pcm_open | | |
|---|---|---|
| Synopsis | This callback is used for record/playback streams and TDM streams.<br>In Capture/Renderer case, it registers a Capture/Renderer plugin. It also registers Equalizer plugin if Equalizer switch is set to 1. Get range of hardware parameter into substream.<br>In TDM Capture/TDM Renderer case, it registers a TDM Capture/TDM Renderer plugin. It also gets range of hardware parameter into substream. | |
| Syntax | static int snd_adsp_pcm_open(struct snd_pcm_substream *substream) | |
| Parameter | struct snd_pcm_substream *substream | Pointer to a pcm substream |
| Return value | 0 | Success |
| | -EINVAL | The registering of the Capture/Renderer or Equalizer plugin, or the TDM Capture/TDM Renderer plugin fails. |

### 3.5.2 snd_adsp_pcm_close

| snd_adsp_pcm_close | | |
|---|---|---|
| Synopsis | This callback is used for record/playback streams and TDM streams.<br>In Capture/Renderer case, it unregisters the Capture/Renderer plugin, free all buffer pool. It also unregisters Equalizer plugin if Equalizer switch is set to 1.<br>In TDM Capture/Renderer case, it unregisters the TDM Capture/Renderer plugin. | |
| Syntax | static int snd_adsp_pcm_close(struct snd_pcm_substream *substream) | |
| Parameter | struct snd_pcm_substream *substream | Pointer to a pcm substream |
| Return value | 0 | Success |
| | -EINVAL | The unregistering of the Capture/Renderer or Equalizer plugin, or TDM Capture/Renderer plugin fails. |

### 3.5.3 snd_adsp_pcm_hw_params

| snd_adsp_pcm_hw_params | | |
|---|---|---|
| Synopsis | This callback is used to allocate buffer pool for data transfer. In Capture/Renderer, it maps ALSA buffer to shared memory. In TDM, it allocates ALSA buffer to transfer data. | |
| Syntax | static 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 |
| | struct snd_pcm_hw_params *hw_params | Hardware parameter is set up by the application |
| Return value | 0 | Success |
| | -ENOMEM | Cannot allocate ALSA buffer in TDM |
| | -EINVAL | Period size is not the power of 2<br>Cannot allocate buffer pool |

### 3.5.4 snd_adsp_pcm_hw_free

| snd_adsp_pcm_hw_free | | |
|---|---|---|
| Synopsis | This callback is used to deallocate ALSA buffer in TDM. In Capture/Renderer, this callback is just dummy. | |
| Syntax | static 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 | Success |
| | -EINVAL | Cannot deallocate ALSA buffer |

### 3.5.5 snd_adsp_pcm_prepare

| snd_adsp_pcm_prepare | | |
|---|---|---|
| Synopsis | This callback helps prepare necessary parameters to set to the plugin before it is ready to run. If user do not set volume, the volume will get the default value of 100%. In Capture/Renderer, if Equalizer is used, it will route Capture/Renderer to Equalizer, otherwise, it requests to map shared memory to data buffer in the plugin.<br>On the occasion of data overrun or underrun error occurrence, this callback waits until all the buffers return. In playback/record case, without Equalizer, it changes the state of the component to reset. | |
| Syntax | static 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 the plugin<br>Cannot route Equalizer and Capture/Renderer<br>Frame sizes between Capture/Renderer and Equalizer are not the same.<br>Runtime error when overrun/underrun occurs |

### 3.5.6 snd_adsp_pcm_trigger

| snd_adsp_pcm_trigger | | |
|---|---|---|
| Synopsis | Start, stop, resume, suspend pcm substream.<br>In Capture/TDM Capture, when it does not running, this callback kicks init with Start/Resume command. When it is running, Start/Resume command is sent in case of overrun/underrun occurrence. | |
| Syntax | static 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, resume, suspend, stop command |
| Return value | 0 | Success |
| | -EINVAL | Invalid command, cannot send fill buffer command to ADSP when performing capture, TDM capture functions |

### 3.5.7 snd_adsp_pcm_ack

| snd_adsp_pcm_ack | | |
|---|---|---|
| Synopsis | This callback is called in read/write operation and in Renderer/TDM Renderer when it starts or resumes PCM substream. Then:<br>In TDM Capture/TDM Renderer, it copies data from/to DMA buffer to/from ALSA buffer, respectively.<br>In Renderer/Capture, it only get the current DMA buffer for data transfer. The DMA buffer is then submitted to ADSP side with EMPTY_THIS_BUFFER (Renderer/TDM Renderer) or FILL_THIS_BUFFER(Capture/TDM Capture). | |
| Syntax | static int snd_adsp_pcm_ack(struct snd_pcm_substream *substream) | |
| Parameter | struct snd_pcm_substream *substream | Pointer to a pcm substream |
| Return value | 0 | Success |
| | -EINVAL | Runtime error |

### 3.5.8 snd_adsp_pcm_pointer

| snd_adsp_pcm_pointer | | |
|---|---|---|
| Synopsis | Update HW buffer position and return the position of the offset on hardware buffer in sample unit | |
| Syntax | static snd_pcm_uframes_t snd_adsp_pcm_pointer(struct snd_pcm_substream *substream) | |
| Parameter | struct snd_pcm_substream *substream | struct snd_pcm_substream *substream |
| Return value | value | Value of the offset on hardware buffer in sample unit |

### 3.5.9 snd_adsp_pcm_mmap

| snd_adsp_pcm_mmap | | |
|---|---|---|
| Synopsis | This callback is to map ALSA buffer to user space. | |
| Syntax | static int snd_adsp_pcm_mmap(struct snd_pcm_substream *substream, struct vm_area_struct *vma) | |
| Parameter | struct snd_pcm_substream *substream | Pointer to a pcm substream. |
| | struct vm_area_struct *vma | Virtual memory area struct |
| Return value | 0 | Success |
| | -EINVAL | Cannot map DMA buffer to userspace |

## 3.3 Detail of APIs for hardware control

❖ APIs get detailed information from the control

### 3.3.1 snd_adsp_control_volume_info

| snd_adsp_control_volume_info | | |
|---|---|---|
| Synopsis | Get detailed information on volume control in playback, capture, TDM playback, TDM capture cases | |
| Syntax | static 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 |

### 3.3.2 snd_adsp_control_sample_rate_info

| snd_adsp_control_sample_rate_info | | |
|---|---|---|
| Synopsis | Get detailed information on sample rate control in playback, capture, TDM playback, TDM capture cases | |
| Syntax | static 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 |

### 3.3.3 snd_adsp_control_eqz_switch_info

| snd_adsp_control_eqz_switch_info | | |
|---|---|---|
| Synopsis | Get detailed info on the equalizer switch control in playback, capture cases | |
| Syntax | static 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 |

### 3.3.4 snd_adsp_control_eqz_info

| snd_adsp_control_eqz_info | | |
|---|---|---|
| Synopsis | Get detailed info on the equalizer control in playback, capture cases | |
| Syntax | static 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 |

### 3.3.5 snd_adsp_control_rdr_out_channel_info

| snd_adsp_control_rdr_out_channel_info | | |
|---|---|---|
| Synopsis | Get detailed info of Renderer output channel control in playback case | |
| Syntax | static int snd_adsp_control_rdr_out_channel_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 Renderer output channel |
| Return value | 0 | Always return 0 |

### 3.3.6 snd_adsp_control_volume_get

| snd_adsp_control_volume_get | | |
|---|---|---|
| Synopsis | Get the PCM volume rate setting value in playback, capture, TDM playback, TDM capture cases | |
| Syntax | static 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. |

### 3.3.7 snd_adsp_control_sample_rate_get

| snd_adsp_control_sample_rate_get | | |
|---|---|---|
| Synopsis | Get sample rate value of hardware output in playback/TDM playback case and sample rate of hardware input in capture/TDM capture case. | |
| Syntax | static 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. |

### 3.3.8 snd_adsp_control_eqz_get

| snd_adsp_control_eqz_get | | |
|---|---|---|
| Synopsis | Get parameters info of equalizer control in playback, capture cases | |
| Syntax | static 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. |

### 3.3.9 snd_adsp_control_eqz_switch_get

| snd_adsp_control_eqz_switch_get | | |
|---|---|---|
| Synopsis | Get status of Equalizer control in playback, capture cases | |
| Syntax | static 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 | Always return 0 |

### 3.3.10 snd_adsp_control_rdr_out_channel_get

| snd_adsp_control_rdr_out_channel_get | | |
|---|---|---|
| Synopsis | Get value of Renderer output channel in playback case | |
| Syntax | static int snd_adsp_control_rdr_out_channel_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 Renderer output channel value |
| Return value | 0 | Success |
| | -EINVAL | Can't get value of output channel from Renderer plugin |

### 3.3.11 snd_adsp_control_volume_put

| snd_adsp_control_volume_put | | |
|---|---|---|
| Synopsis | Set the PCM volume rate value in playback, capture, TDM playback, TDM capture cases | |
| Syntax | static 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.<br>Set volume TDM Capture/TDM Renderer at runtime |

### 3.3.12 snd_adsp_control_sample_rate_put

| snd_adsp_control_sample_rate_put | | |
|---|---|---|
| Synopsis | Set sample rate for hardware output in playback and TDM playback case, input in the capture and TDM capture case | |
| Syntax | static 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 | 1 | Hardware parameter is changed. |
| | -EINVAL | Can't set parameter to ADSP. |

### 3.3.13 snd_adsp_control_eqz_put

| snd_adsp_control_eqz_put | | |
|---|---|---|
| Synopsis | Set equalizer parameter in playback, capture cases | |
| Syntax | static 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 | 1 | Hardware parameter is changed. |
| | -EINVAL | Can’t set parameter to ADSP. |

### 3.3.14 snd_adsp_control_eqz_switch_put

| snd_adsp_control_eqz_switch_put | | |
|---|---|---|
| Synopsis | Enable or disable Equalizer control in playback, capture cases | |
| Syntax | static 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 | 1 | Success |
| | -EINVAL | Cannot enable EQZ when Renderer/Capture plugin is running. |

### 3.3.15 snd_adsp_control_rdr_out_channel_put

| snd_adsp_control_rdr_out_channel_put | | |
|---|---|---|
| Synopsis | Set Renderer output channel in playback case | |
| Syntax | static int snd_adsp_control_rdr_out_channel_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 Renderer output channel value |
| Return value | 1 | Parameter after setting is changed. |
| | -EINVAL | Cannot set output channel for Renderer |

## 3.4 Detail of APIs for ASoC Platform interface

### 3.4.1 snd_adsp_pcm_new

| snd_adsp_pcm_new | | |
|---|---|---|
| Synopsis | This API registers necessary control interfaces for ADSP soundcard based on CPU DAI type (playback/capture type or TDM playback/TDM capture type). In playback/capture case, when registering control interfaces for the second/third/fourth playback/capture. In TDM, it also pre-allocates a memory region for ALSA buffer for transferring data. | |
| Syntax | static int snd_adsp_pcm_new(struct snd_soc_pcm_runtime *runtime) | |
| Parameter | struct snd_soc_pcm_runtime *runtime | PCM runtime data |
| Return value | 0 | Success |
| | -EINVAL | Cannot register a control into ALSA framework |

## 3.5 Detail of APIs for ASoC Platform driver

### 3.5.1 snd_adsp_probe

| snd_adsp_probe | | |
|---|---|---|
| Synopsis | Register platform driver with 5 CPU DAIs (4 DAIs for playback/record and 1 DAI for TDM playback/TDM record) and add ADSP sound card component to ASoC framework. It also assign default values for device parameters, DMA parameters for DAIs. It also parses values for DAIs defined in device tree. | |
| Syntax | static int snd_adsp_probe(struct platform_device *pdev) | |
| Parameter | struct platform_device *pdev | Pointer to platform device structure |
| Return value | 0 | Success |
| | -ENOMEM | Cannot allocate ADSP card data structure |
| | -EINVAL | Cannot register platform device<br>Cannot add ADSP sound card component to ASoC framework<br>Cannot parse parameters for devices, DMAs |

### 3.5.2 snd_adsp_remove

| snd_adsp_remove | | |
|---|---|---|
| Synopsis | This callback is used to unregister platform driver and remove ADSP sound card component from ASoC framework. | |
| Syntax | static int snd_adsp_remove(struct platform_device *pdev) | |
| Parameter | struct platform_device *pdev | Pointer to platform device structure |
| Return value | 0 | Success |
| | -ENODEV | ADSP sound card is invalid |

# 4 Sequence diagram

## 4.1 Playback stream flow

### 4.1.1 Open a renderer substream

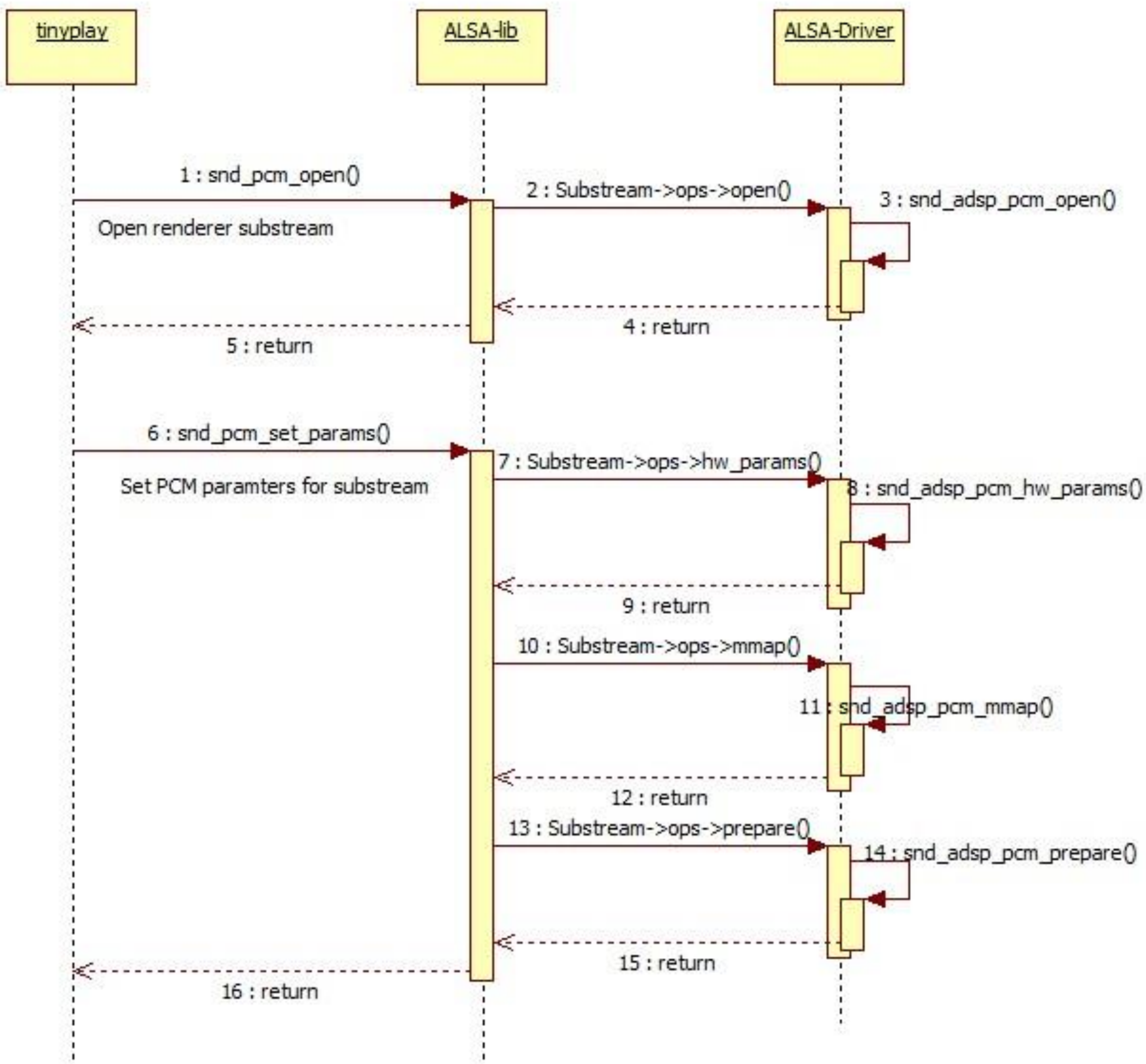



Figure 4-1 Open flow for playback stream

### 4.1.2 Open a TDM renderer substream

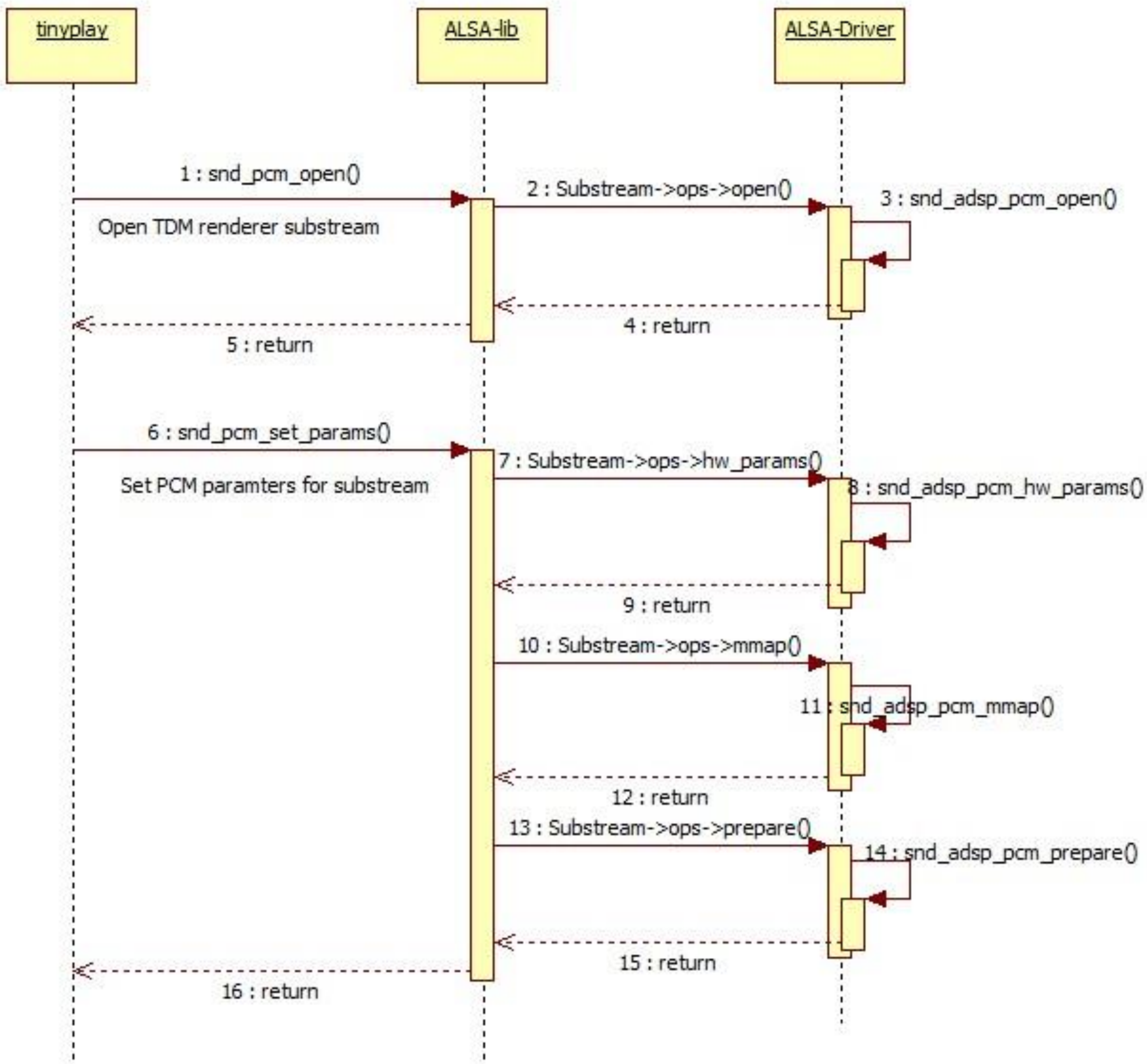



Figure 4-2 Open flow for TDM playback stream

### 4.1.3 Write data flow

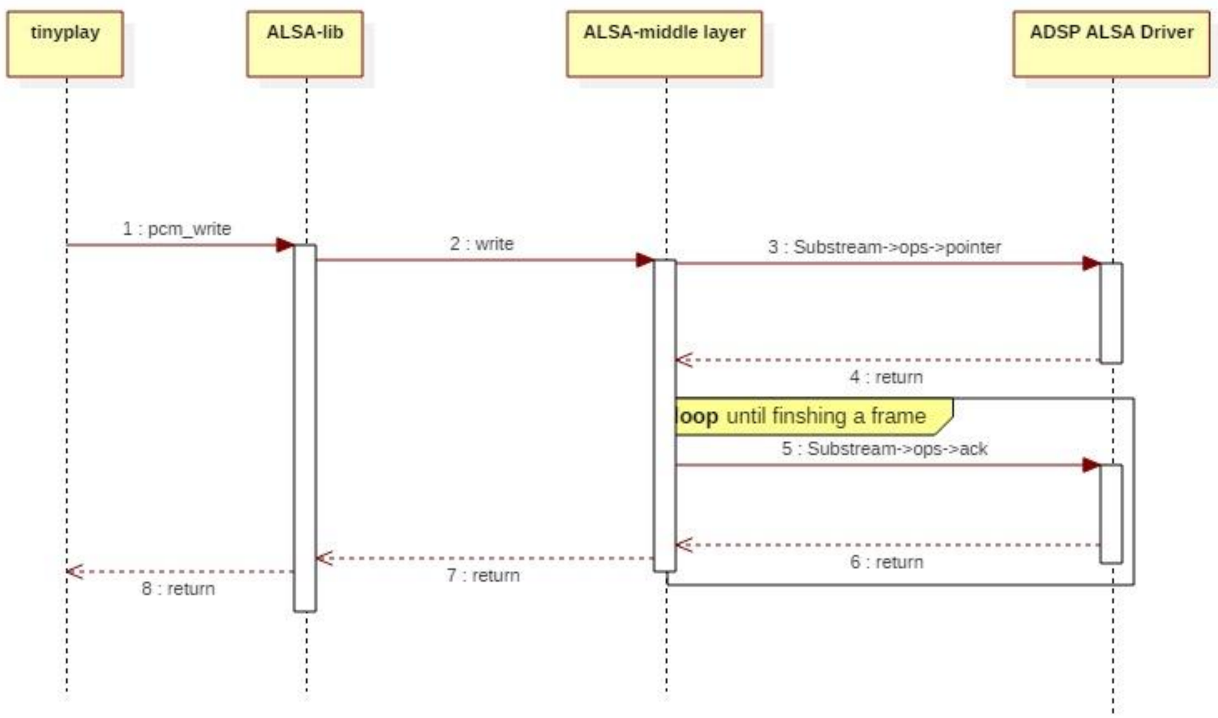



Figure 4-3 Write data flow

### 4.1.4 Close a playback/TDM playback substream

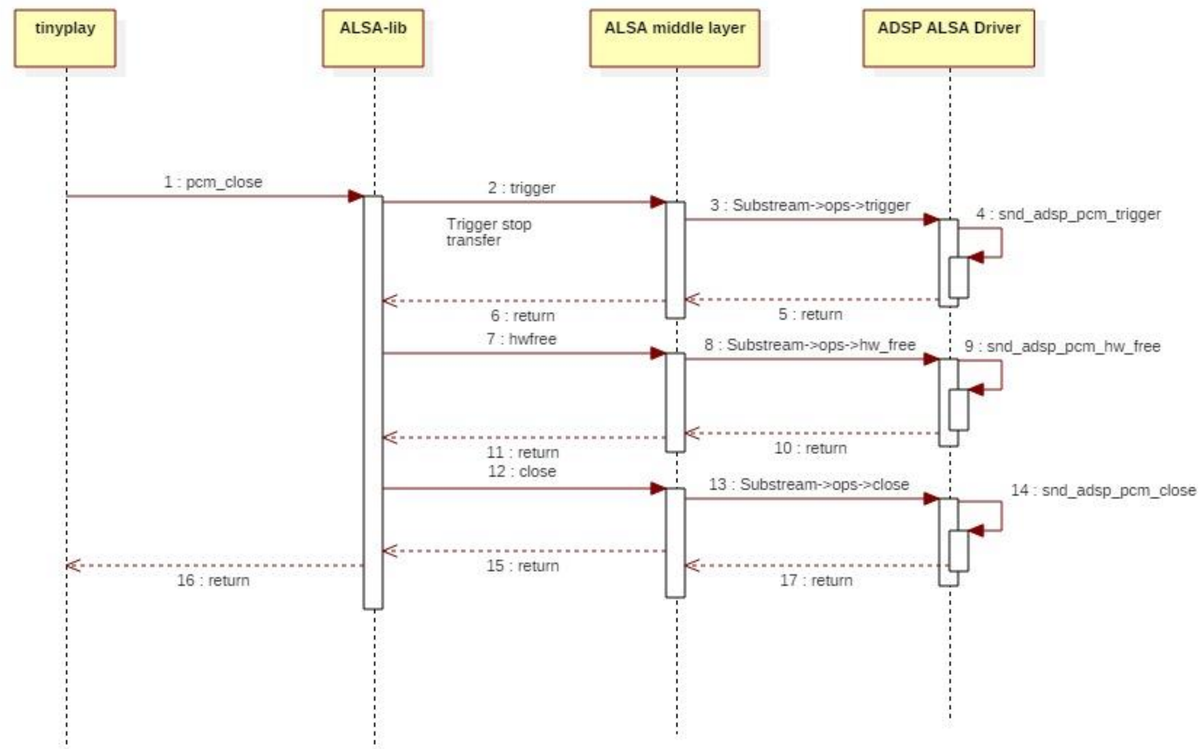



Figure 4-4 Close flow for playback/TDM playback stream

## 4.2 Capture/TDM capture streams flow

### 4.2.1 Open a capture substream

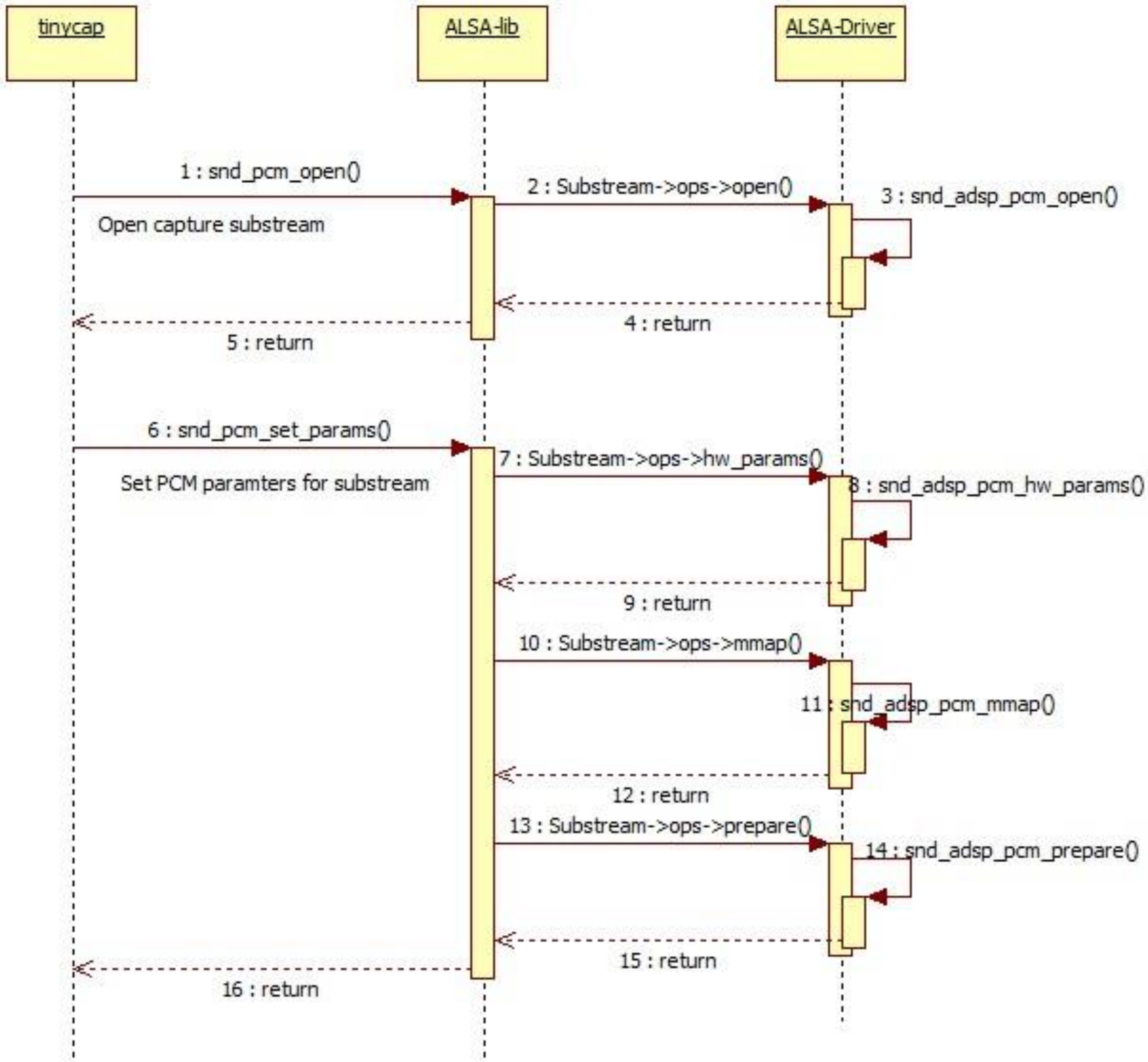



Figure 4-5 Open flow for capture stream

### 4.2.2 Open a TDM capture substream

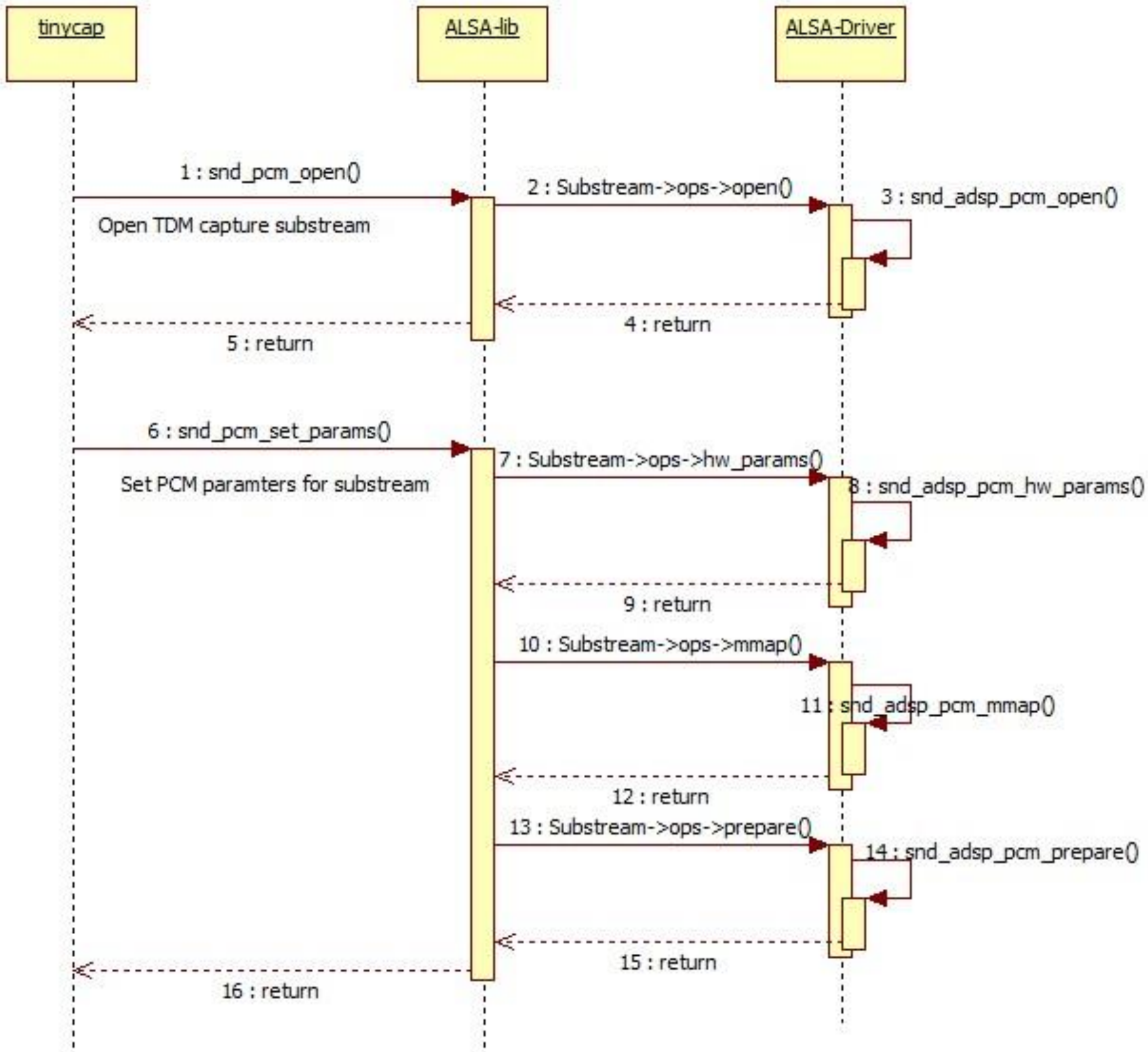



Figure 4-6 Open flow for TDM capture stream

### 4.2.3 Read data flow

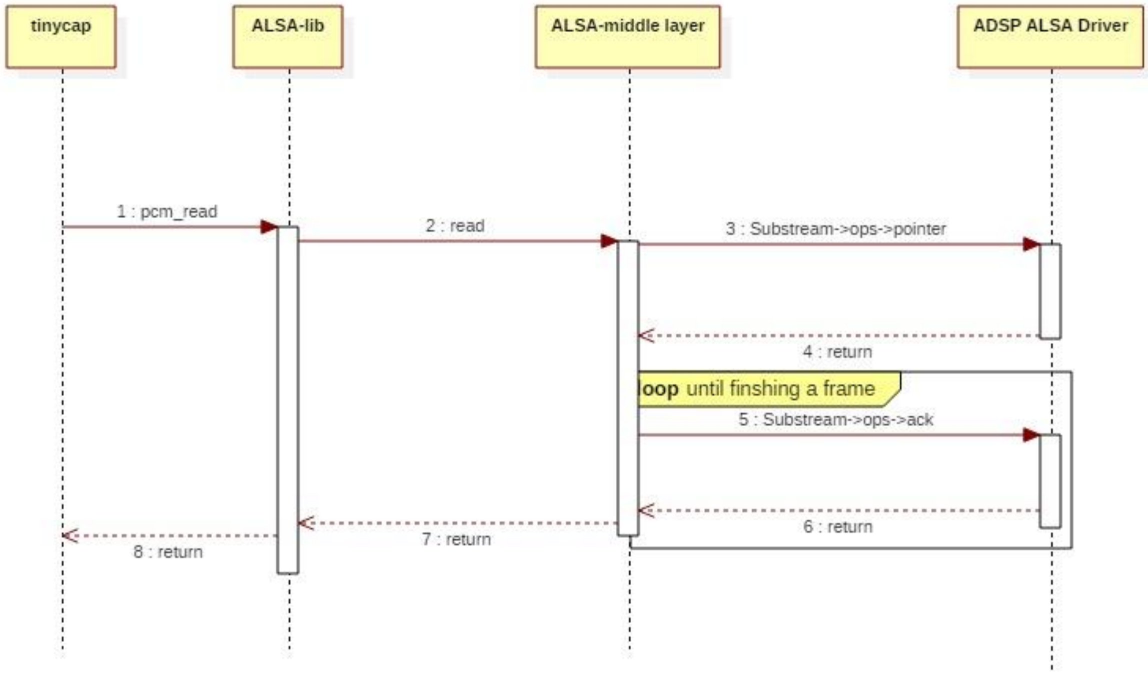



Figure 4-7 Read data flow

### 4.2.4 Close a capture/TDM capture substream

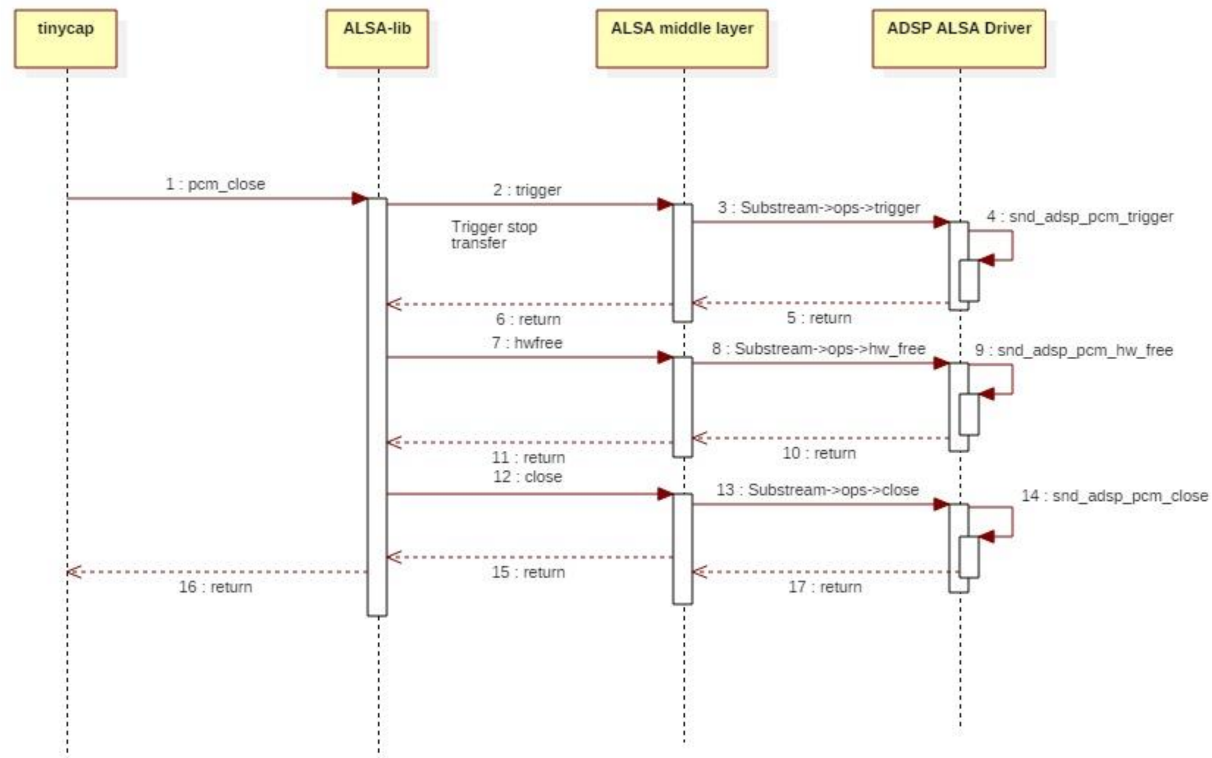



Figure 4-8 Close flow for capture/TDM capture stream

## 4.3 Volume Control Flow

### 4.3.1 Get volume value from hardware

In playback case, the control name is **"PlaybackVolume"**. This string maps to the control defined in ADSP ALSA Driver.

In capture case, the control name is **"CaptureVolume"**. This string maps to the control defined in ADSP ALSA Driver.

In TDM playback case, the control name is **"TDMPlaybackVolume"**. This string maps to the control defined in ADSP ALSA Driver.

In TDM capture case, the control name is **"TDMCaptureVolume"**. This string maps to the control defined in ADSP ALSA Driver.

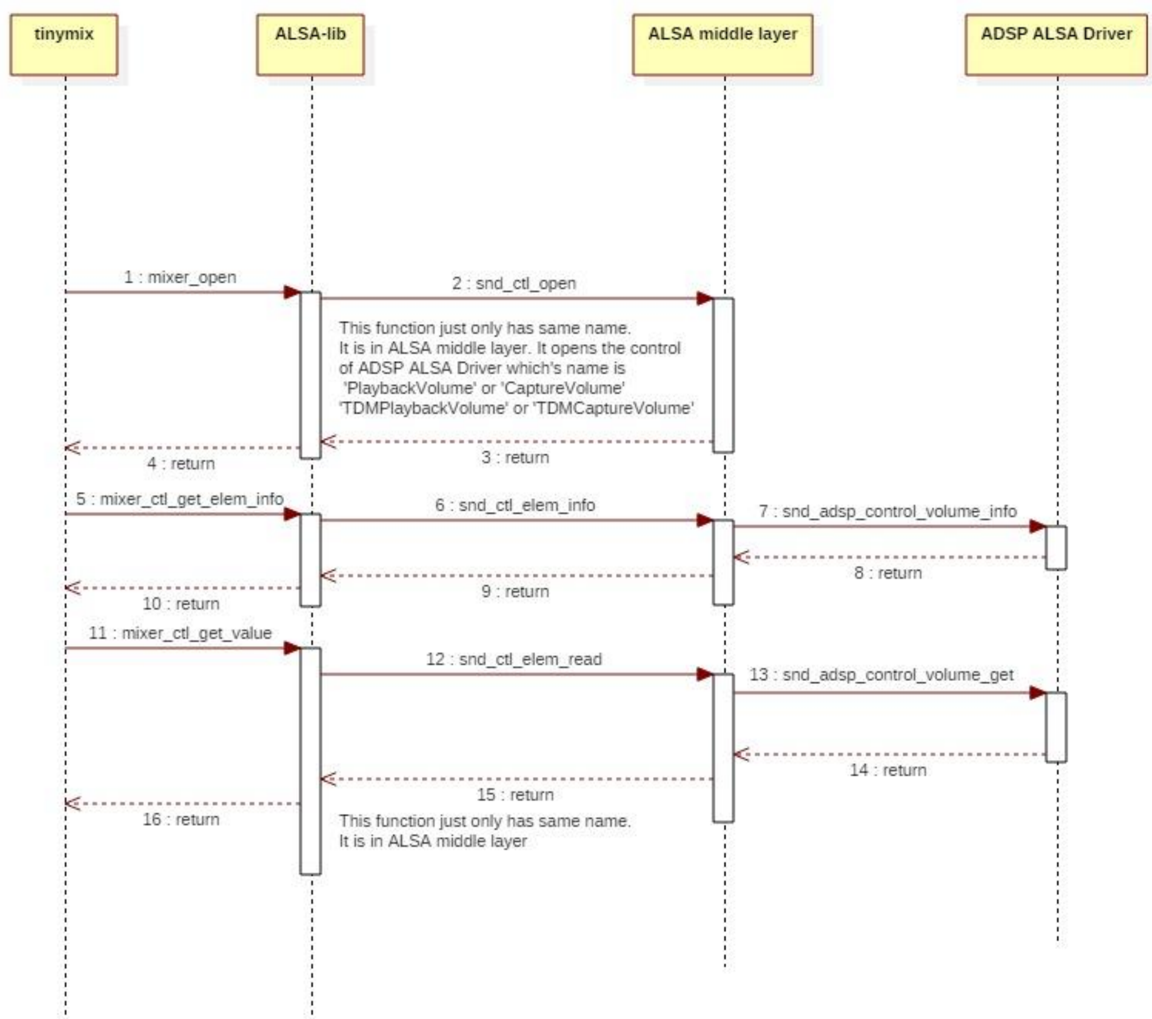



Figure 4-9 Flow of getting volume information from hardware

### 4.3.2 Set volume value to hardware

In playback case, the control name is **"PlaybackVolume"**. This string maps to the control defined in ADSP ALSA Driver.

In capture case, the control name is **"CaptureVolume"**. This string maps to the control defined in ADSP ALSA Driver.

In TDM playback case, the control name is **"TDMPlaybackVolume"**. This string maps to the control defined in ADSP ALSA Driver.

In TDM capture case, the control name is **"TDMCaptureVolume"**. This string maps to the control defined in ADSP ALSA Driver.

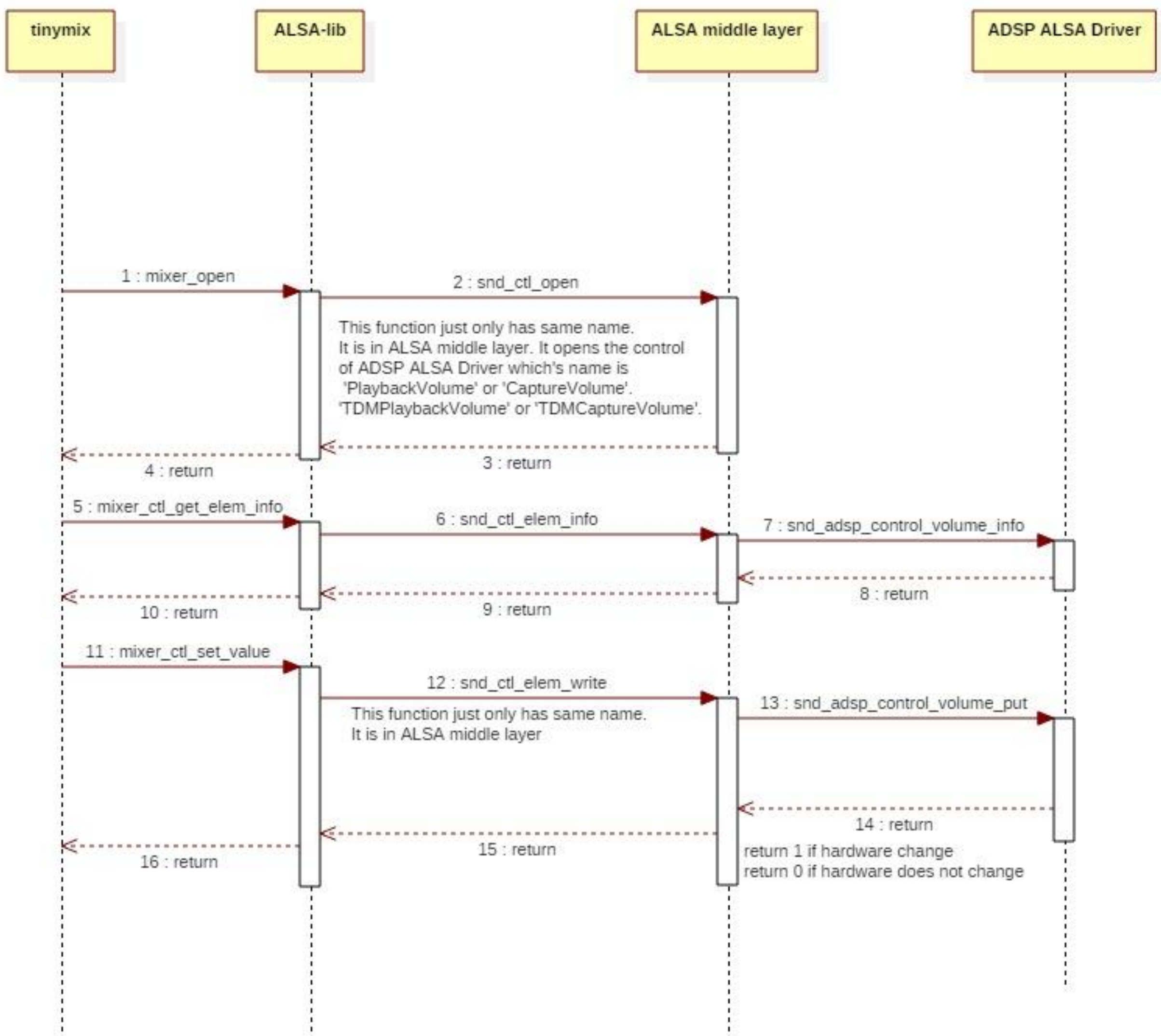



Figure 4-10 Flow of setting volume of hardware

## 4.4 Sample Rate Converter Control Flow

### 4.4.1 Get sample rate of hardware

In playback case, the control name is **"PlaybackOutRate"**. This string maps to the control defined in ADSP ALSA Driver.

In capture case, the control name is **"CaptureInRate"**. This string maps to the control defined in ADSP ALSA Driver.

In TDM playback case, the control name is **"TDMPlaybackOutRate"**. This string maps to the control defined in ADSP ALSA Driver.

In TDM capture case, the control name is **"TDMCaptureInRate"**. This string maps to the control defined in ADSP ALSA Driver.

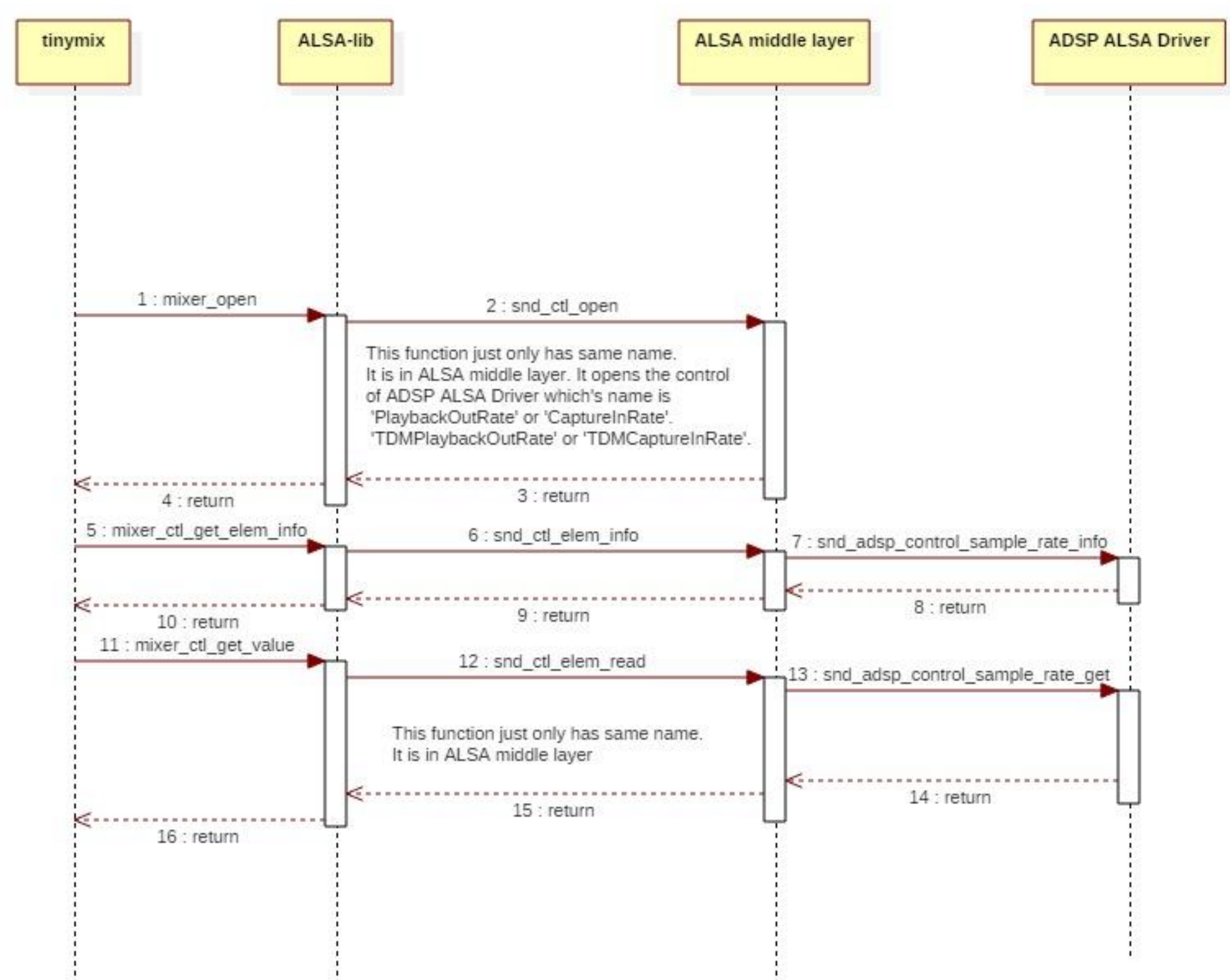



Figure 4-11 Flow of getting sample rate information from hardware

### 4.4.2 Set sample rate of hardware

In playback case, the control name is **"PlaybackOutRate"**. This string maps to the control defined in ADSP ALSA Driver.

In capture case, the control name is **"CaptureInRate"**. This string maps to the control defined in ADSP ALSA Driver.

In TDM playback case, the control name is **"TDMPlaybackOutRate"**. This string maps to the control defined in ADSP ALSA Driver.

In TDM capture case, the control name is **"TDMCaptureInRate"**. This string maps to the control defined in ADSP ALSA Driver.

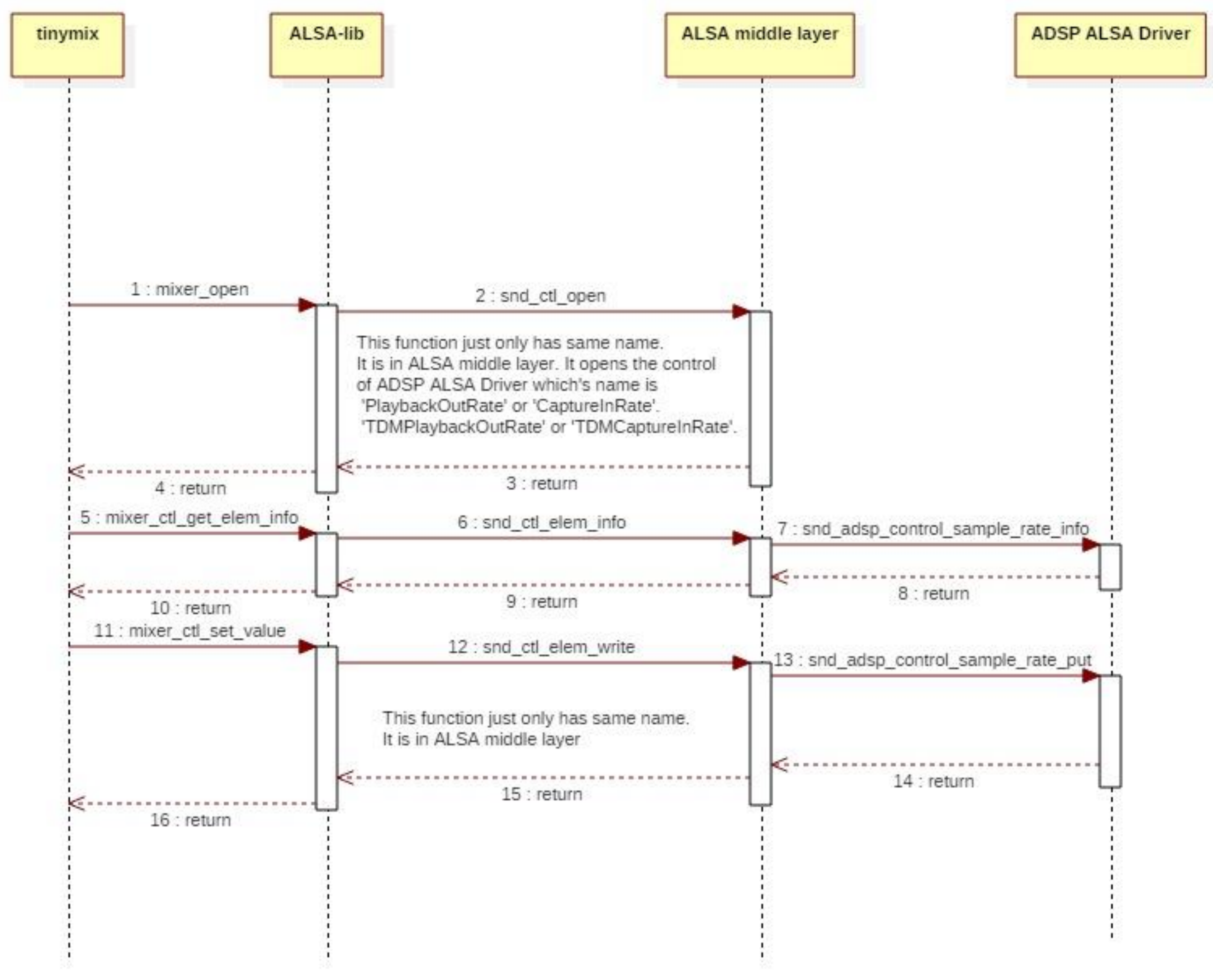



Figure 4-12 Flow of setting sample rate of hardware

## 4.5 Output Channel Control Flow

### 4.5.1 Get output channel of Renderer

The control name is **"PlaybackOutChannel"**. This string maps to the control defined in ADSP ALSA Driver.

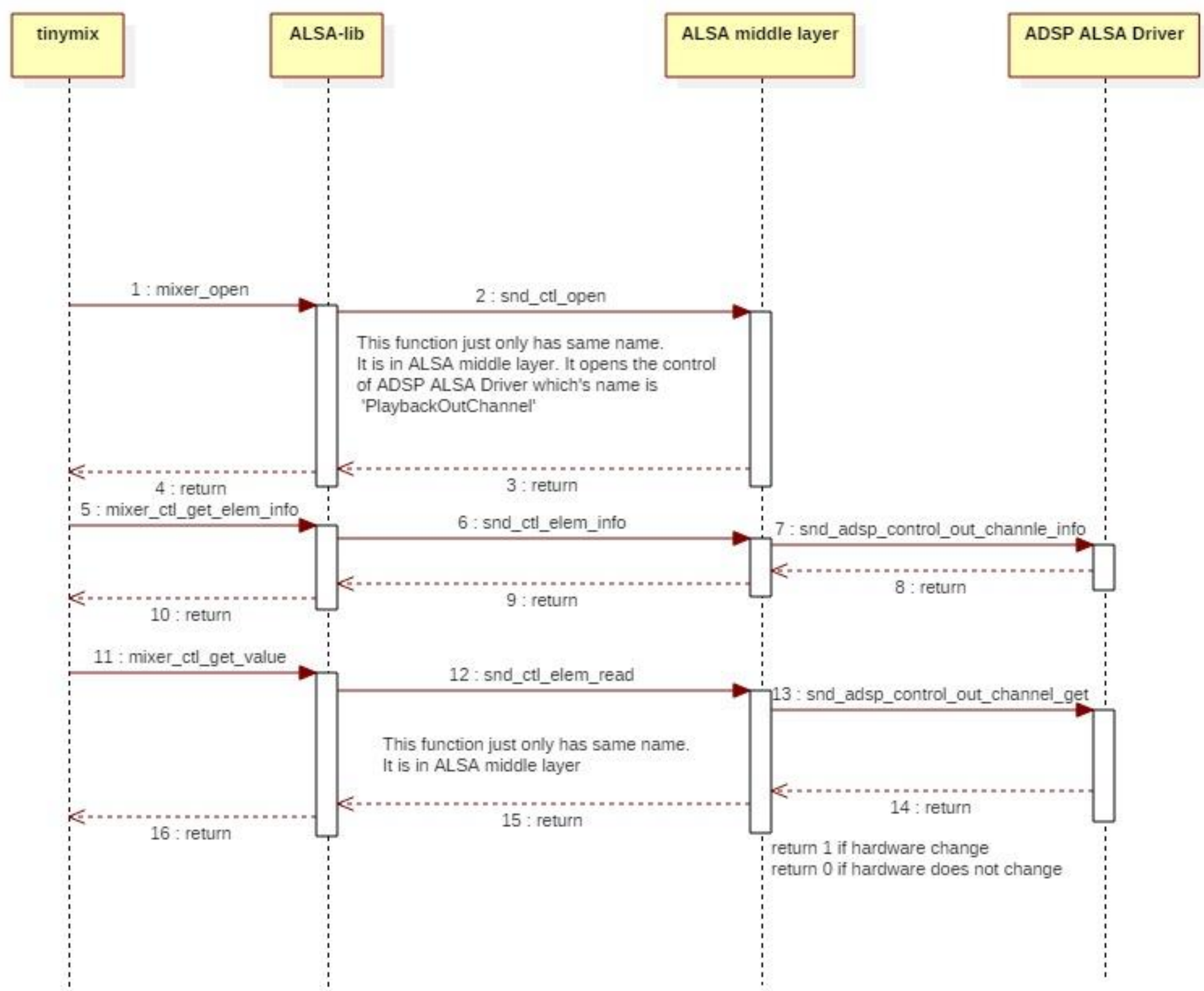



Figure 4-13 Flow of getting Renderer output channel information

### 4.5.2 Set output channel of Renderer

The control name is **"PlaybackOutChannel"**. This string maps to the control defined in ADSP ALSA Driver.

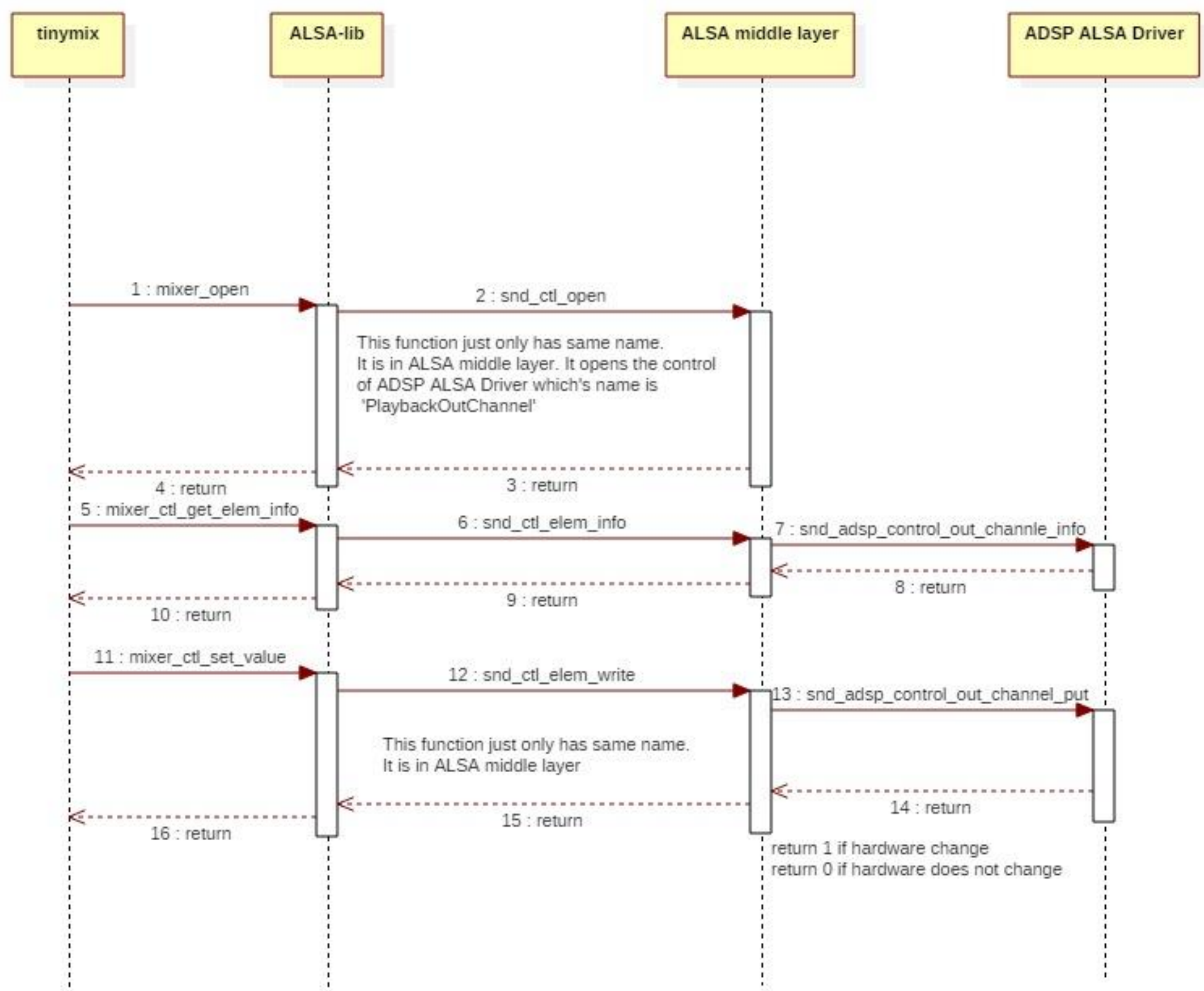



Figure 4-14 Flow of setting Renderer output channel

## 4.6 Equalizer Control Flow

### 4.6.1 Get status of Equalizer control

In playback case, the control name is **”PlaybackEQZSwitch”**. This string maps to the control defined in ADSP ALSA Driver.

In capture case, the control name is **”CaptureEQZSwitch”**. This string maps to the control defined in ADSP ALSA Driver.

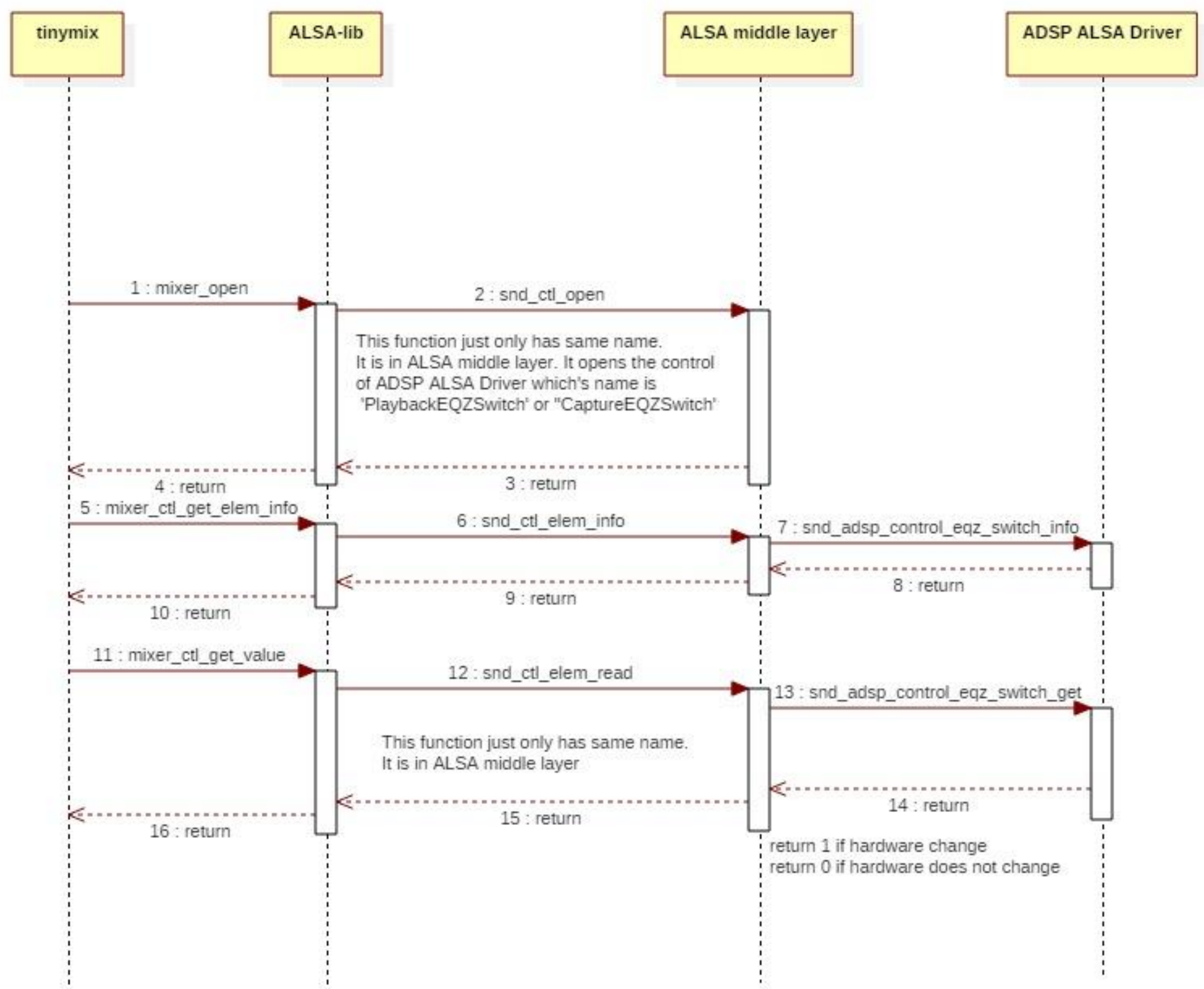



Figure 4-15 Flow of Equalizer control’s status getting

### 4.6.2 Enable Equalizer control

In playback case, the control name is **"PlaybackEQZSwitch"**. This string maps to the control defined in ADSP ALSA Driver.

In capture case, the control name is **"CaptureEQZSwitch"**. This string maps to the control defined in ADSP ALSA Driver.

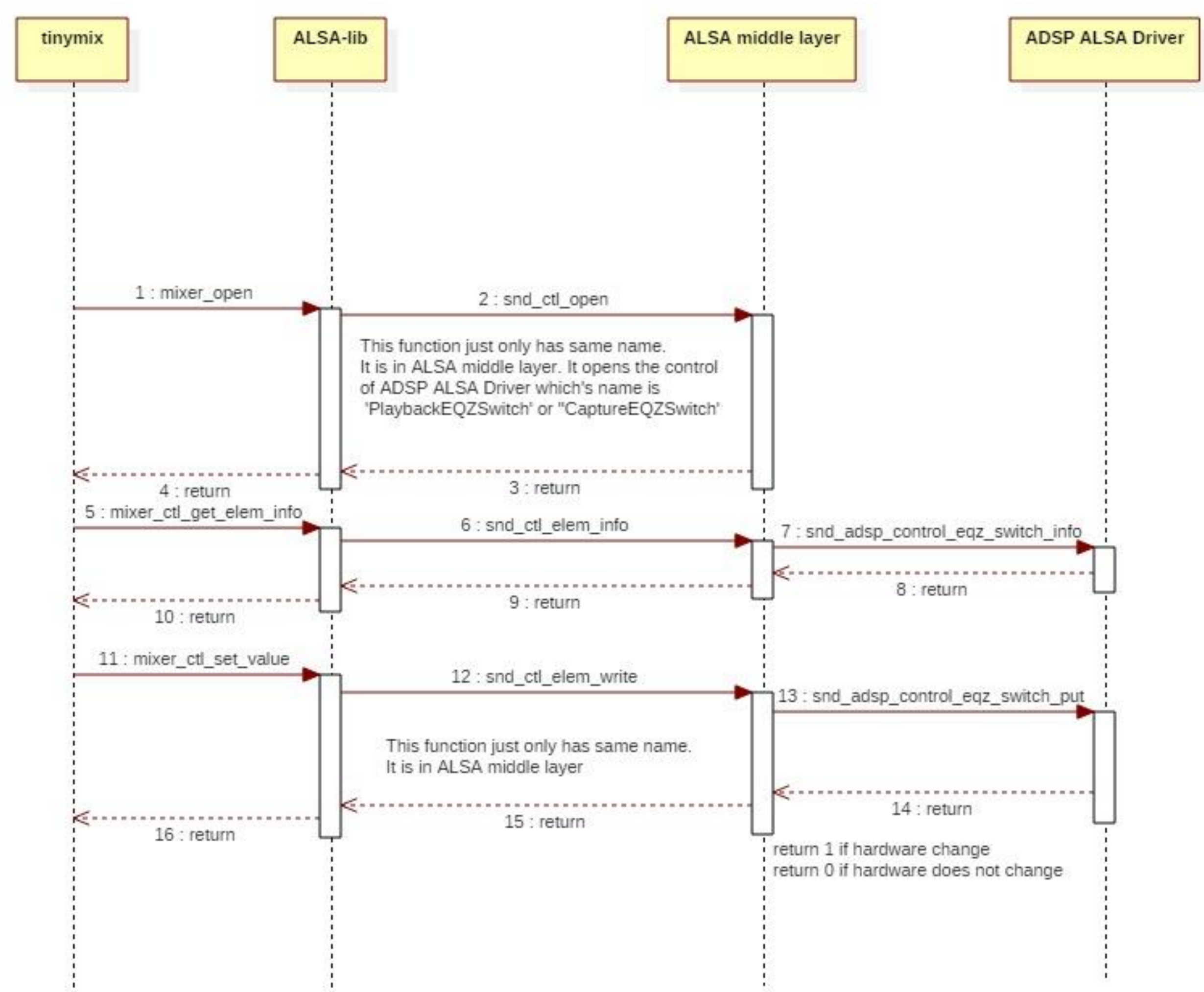



Figure 4-16 Flow of Equalizer activation setting

### 4.6.3 Get Equalizer parameters of hardware

In playback case, the control name is **"PlaybackEQZControl"**. This string maps to the control defined in ADSP ALSA Driver.

In capture case, the control name is **"CaptureEQZControl"**. This string maps to the control defined in ADSP ALSA Driver.

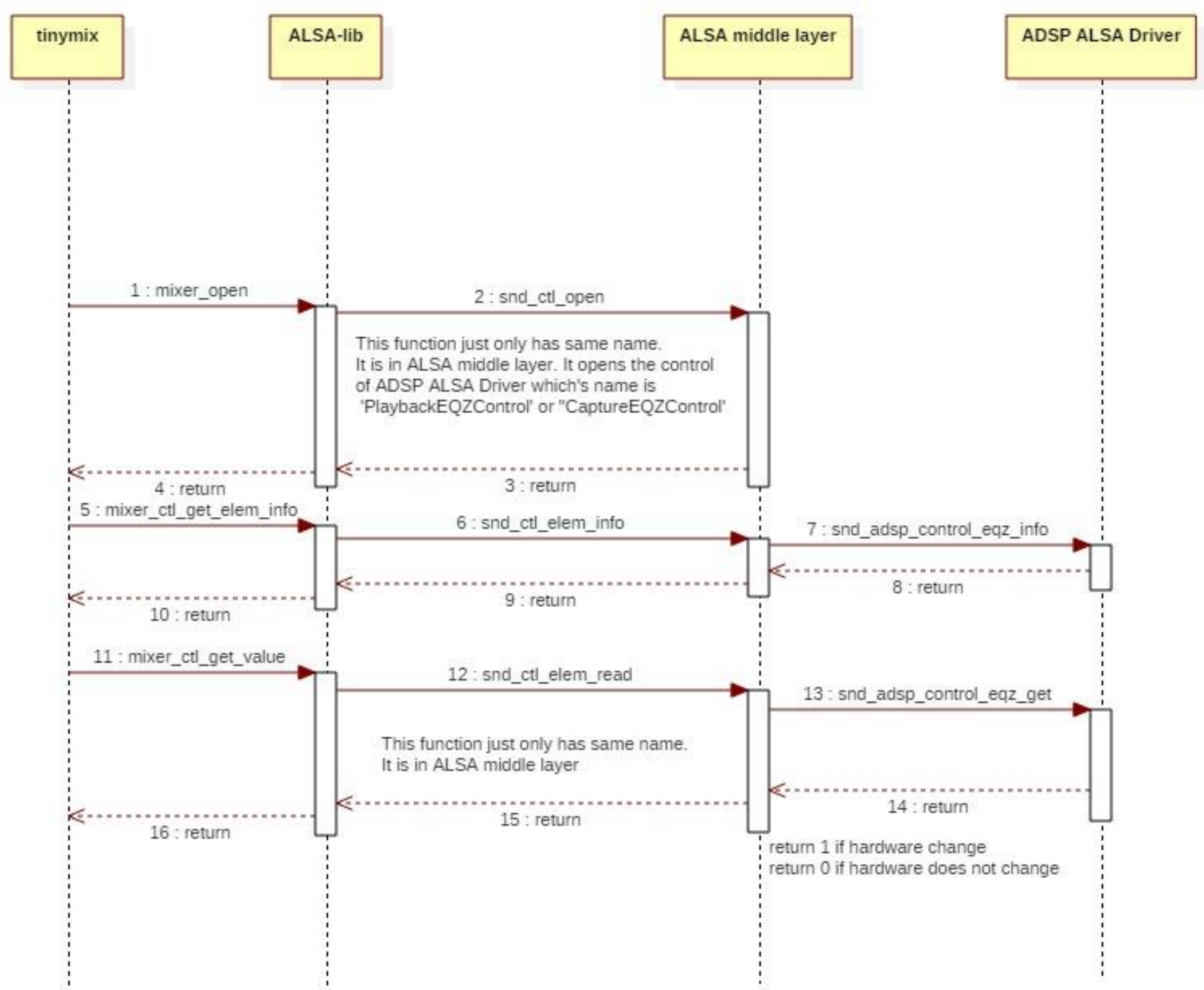



Figure 4-17 Flow of getting Equalizer parameters of hardware

### 4.6.4 Set Equalizer parameters of hardware

In playback case, the control name is **"PlaybackEQZControl"**. This string maps to the control defined in ADSP ALSA Driver.

In capture case, the control name is **"CaptureEQZControl"**. This string maps to the control defined in ADSP ALSA Driver.

In Parametric Equalizer, there are 9 filters. Each filter has its own parameters: frequency center, bandwidth, filter type, gain base, gain. Therefore, there are 55 values to set as the table below:

| **Parameters** | **Value range** | **Number** |
|---|---|---|
| Equalizer type | 0 (for Parametric) | 1 |
| Filter index | 1 - 9 | 9 |
| Frequency center | 20 – 20000 | 9 |
| Bandwidth | $0.2 \times 2^{27} – 15 \times 2^{27}$ | 9 |
| Filter type | 0 – 2 | 9 |
| Gain base | 20 - 20000 | 9 |
| Gain | $10^{-10/20} \times 2^{28} – 10^{10/20} \times 2^{28}$ | 9 |
| | | Total: 55 |

Order of parameters to set for Parametric Equalizer:

*Equalizer type, 9 x (filter index, frequency center, bandwidth, filter type, gainbase, gain)*

In Graphic Equalizer, there are 5 filters, each of which has its own parameter: graphic gain. The settings are described below:

| **Parameters** | **Value range** | **Number** |
|---|---|---|
| Equalizer type | 1 (for Graphic) | 1 |
| Filter index | 1 - 5 | 5 |
| Graphic gain | $10^{-10/20} \times 2^{28} – 10^{10/20} \times 2^{28}$ | 5 |
| | | Total: 11 |

Order of parameters to set for Graphic Equalizer:

*Equalizer type, 5 x (filter index, graphic gain)*

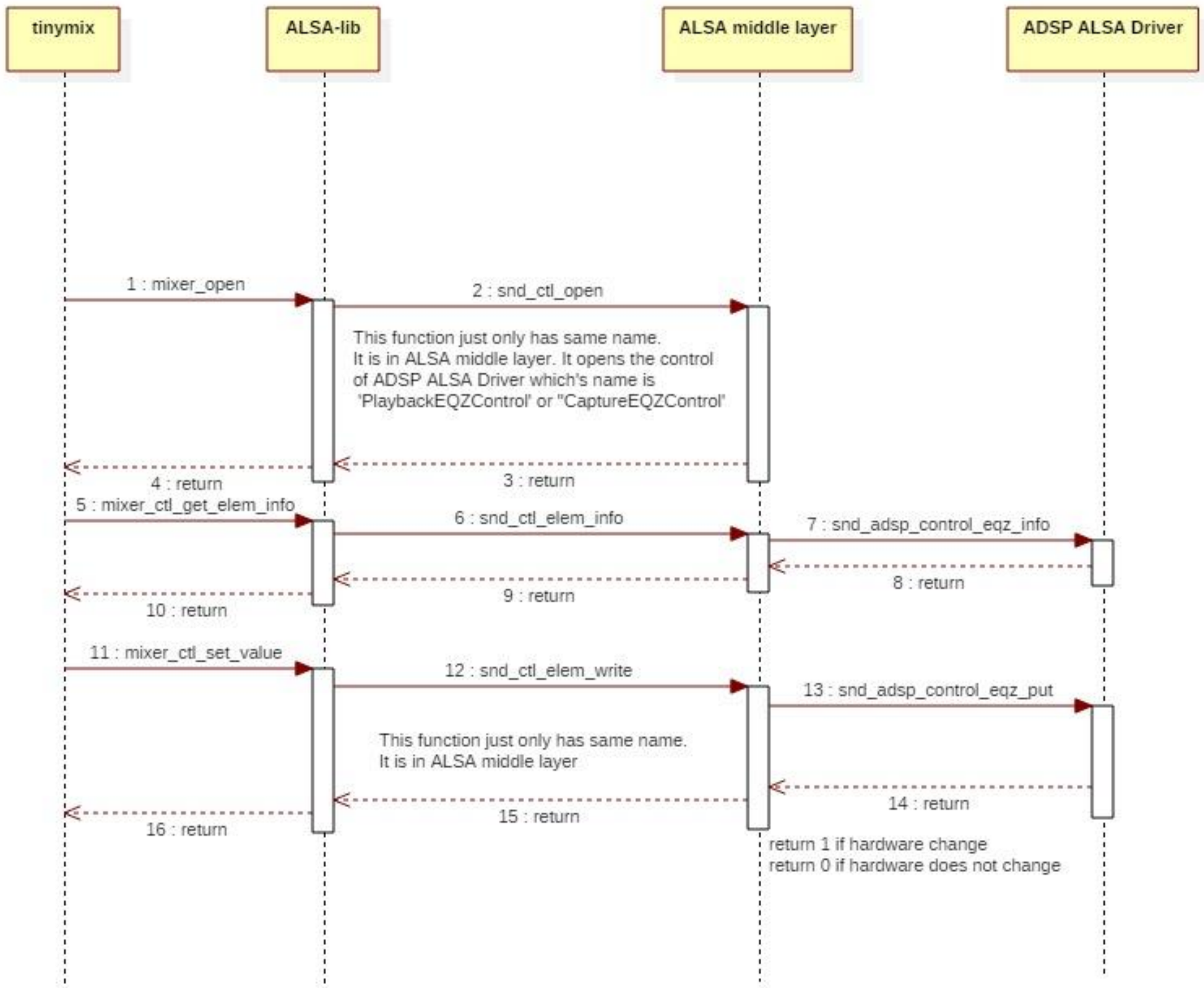



Figure 4-18 Flow of setting Equalizer parameters of hardware

# 5. List of Usage

This section is to help user understand the usage of the ADSP ALSA interface.
Below table show target platforms support for each use case.

| Use case | Chip | Board |
|---|---|---|
| Playback/Capture | H3/M3/M3N/E3 | Salvator, Ebisu board |
| TDM Playback/ TDM Capture | H3/M3 | Starter KIT –Kingfisher board |

Table 5-1 Target environment for each use case.

[Note]

- For the case of 24-bit streams with Equalizers involved, the maximum of streams to run concurrently is 2 due to memory limitation.
- tinycap and tinyplay do not support to run 24 bit wav files to ADSP sound driver.
- tinymix cannot set control index along with control name. If you want to set or get control values from other DAI indexes. You have to use control number. This value can be gotten using tinymix command as below (these numbers – **ctl** column can be different from other user environments):

```
console: tinymix -D 0
Mixer name: 'audio-card'
Number of controls: 54
ctl    type   num    name
6      INT    1      PlaybackVolume
7      INT    1      CaptureVolume
8      INT    1      PlaybackOutRate
9      INT    1      CaptureInRate
10     INT    55     PlaybackEQZControl
11     INT    55     CaptureEQZControl
12     INT    1      PlaybackEQZSwitch
13     INT    1      CaptureEQZSwitch
14     INT    1      PlaybackOutChannel
15     INT    1      PlaybackVolume
16     INT    1      CaptureVolume
17     INT    1      PlaybackOutRate
18     INT    1      CaptureInRate
19     INT    55     PlaybackEQZControl
20     INT    55     CaptureEQZControl
21     INT    1      PlaybackEQZSwitch
22     INT    1      CaptureEQZSwitch
23     INT    1      PlaybackOutChannel
24     INT    1      PlaybackVolume
25     INT    1      CaptureVolume
26     INT    1      PlaybackOutRate
27     INT    1      CaptureInRate
28     INT    55     PlaybackEQZControl
29     INT    55     CaptureEQZControl
30     INT    1      PlaybackEQZSwitch
31     INT    1      CaptureEQZSwitch
32     INT    1      PlaybackOutChannel
33     INT    1      PlaybackVolume
34     INT    1      CaptureVolume
35     INT    1      PlaybackOutRate
36     INT    1      CaptureInRate
37     INT    55     PlaybackEQZControl
38     INT    55     CaptureEQZControl
39     INT    1      PlaybackEQZSwitch
40     INT    1      CaptureEQZSwitch
41     INT    1      PlaybackOutChannel
```

From later examples in this document, setting control numbers will be referred from above result.

## 5.1. Playback

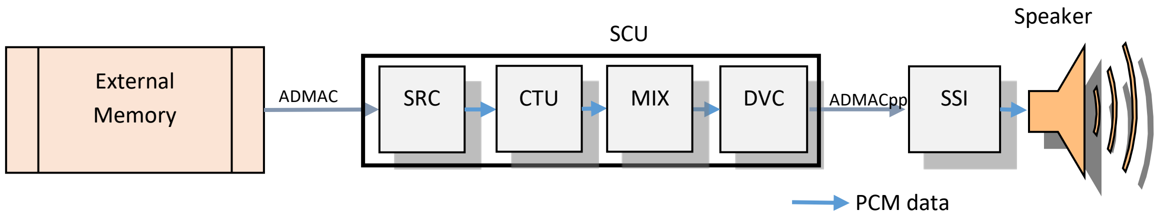



Figure 5-1 Data path for playback

Below table shows information of ADSP ALSA Driver sound card.

| Sound card | Description |
|---|---|
| hw:0,0,0 | ADSP ALSA sound card (card 0) with DAI 0 |
| hw:0,1,0 | ADSP ALSA sound card (card 0) with DAI 1 |
| hw:0,2,0 | ADSP ALSA sound card (card 0) with DAI 2 |
| hw:0,3,0 | ADSP ALSA sound card (card 0) with DAI 3 |

Table 5-2 Detailed information of ADSP ALSA sound card

The sound card is configured in device tree as default card (card 0). User can run multi playback stream with different DAIs. More information about mixing multi stream, please refer 5.1.5

### 5.1.1. Playback stream

ADSP ALSA Driver supports playback stream (PCM width 16/24, sample rate 32/44.1/48 kHz, frame size 4/8/16/32/64/128/256/512/1024, 1/2 channels, period count 2/4).

[Note]: In 16 bit mono case, ADSP ALSA Driver only supports frame size 1024. Other frame sizes are not guarantee.

- User setting:
  tinyplay thetest_FULL_s_32000_16.wav -D 0 -d 0 -p 1024 -n 4

### 5.1.2. Setting Playback Volume

ADSP ALSA Driver supports setting volume with range from 0 to 799. Value 799 means that increase volume to 8 time.

- User setting: perform below steps to set volume
  - Run stream
    tinyplay thetest_FULL_s_48000_16.wav -D 0 -d 0
  - Set volume
    tinymix -D 0 PlaybackVolume 100
    or
    tinymix -D 0 6 100
  - Get volume information
    tinymix -D 0 PlaybackVolume
    or
    tinymix -D 0 6

### 5.1.3. Setting Output Sample Rate

ADSP ALSA Driver supports converting data's sample rate to other value. Sample rate supported with range (32/44.1/48 kHz).

- User setting
  - Set output sample rate
    tinymix -D 0 PlaybackOutRate 48000
    or
    tinymix -D 0 8 48000
  - Run stream
    tinyplay thetest_FULL_s_32000_16.wav -D 0 -d 0
  - Get information output sample rate
    tinymix -D 0 PlaybackOutRate
    or
    tinymix -D 0 8

### 5.1.4. Setting Output Channel

ADSP ALSA Driver supports convert data's channel number to other value as below table.

| Number | Input data | Output data | Supported |
|---|---|---|---|
| 1 | 16 bit & 1 channel | 16 bit & 2 channel | O |
| 2 | 16 bit & 2 channel | 16 bit & 1 channel | O |
| 3 | 24 bit & 2 channel | 24 bit & 1 channel | X |

Table 5-3 List of channel number conversation

O means supported
X means unsupported

- User setting:
  - Set output channel number
    tinymix -D 0 PlaybackOutChannel 2
    or
    tinymix -D 0 14 2
  - Run stream
    tinyplay thetest_FULL_s_48000_16.wav -D 0 -d 0
  - Get information about channel number
    tinymix -D 0 PlaybackOutChannel
    or
    tinymix -D 0 14

### 5.1.5. MIX function

ADSP ALSA Driver supports mixing multi (2/3/4) playback stream with same sample rate.
But due to hardware performance and memory, some limitation showed as below:

- In case routing between Equalizer and Renderer playback 24 bit stream only mixing 2 stream is supported.
- H3 can support mixing 2, 3 or 4 stream but M3 only supports mixing 2 stream.

#### 5.1.5.1. Mix 2 playback stream

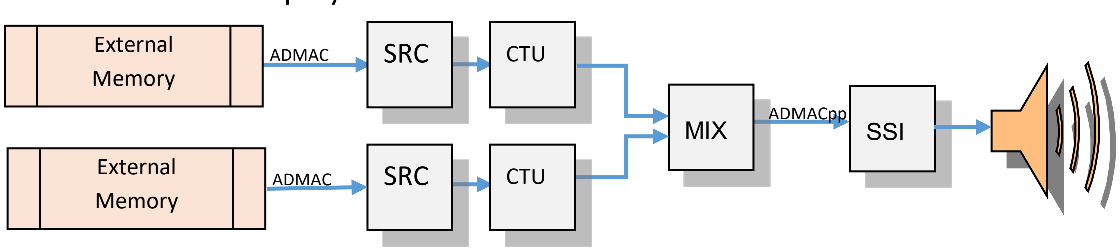



Figure 5-2 Data path when mixing 2 streams

- User setting
  - Playback 1st stream:
    tinyplay thetest_FULL_s_44100_16.wav -D 0 -d 0

  - Playback 2nd stream:
    tinymix -D 0 17 44100
    tinyplay thetest_FULL_s_48000_16.wav -D 0 -d 1

### 5.1.5.2. Mix 3 playback stream

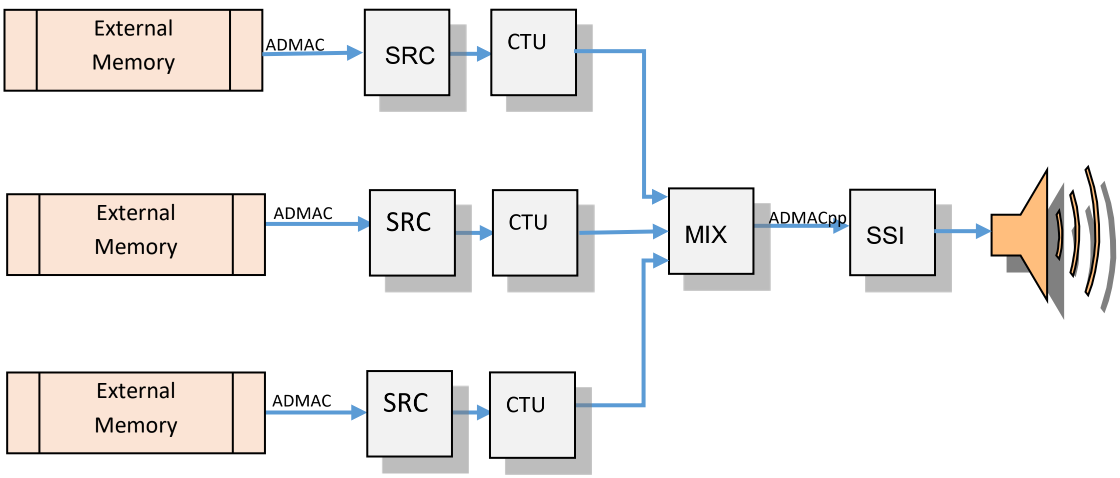



Figure 5-3 Data path when mixing 3 streams

- User setting
  - Playback 1st stream:
    tinyplay thetest_FULL_s_44100_16.wav -D 0 -d 0

  - Playback 2nd stream:
    tinymix -D 0 17 44100
    tinyplay thetest_FULL_s_48000_16.wav -D 0 -d 1

  - Playback 3rd stream:
    tinymix -D 0 26 44100
    tinyplay thetest_FULL_s_32000_16.wav -D 0 -d 2

### 5.1.5.3. Mix 4 playback stream

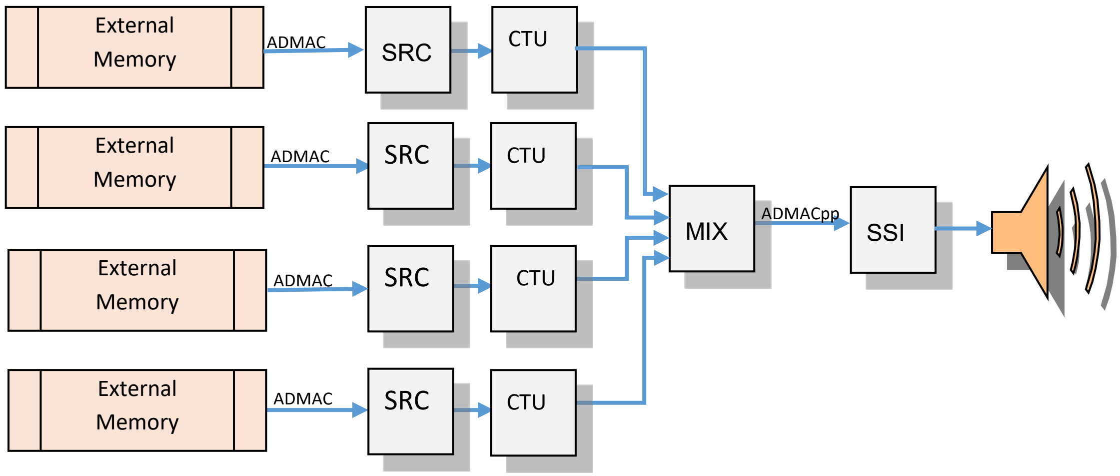



Figure 5-4 Data path when mixing 4 streams

- User setting
  - Playback 1st stream:
    tinyplay thetest_FULL_s_44100_16.wav -D 0 -d 0

  - Playback 2nd stream:
    tinymix -D 0 17 44100
    tinyplay thetest_FULL_s_32000_16.wav -D 0 -d 1

  - Playback 3rd stream:
    tinymix -D 0 26 44100
    tinyplay thetest_FULL_s_32000_16.wav -D 0 -d 2

  - Playback 4th stream:
    tinymix -D 0 35 44100
    tinyplay thetest_FULL_s_48000_16.wav -D 0 -d 3

### 5.1.6. Optimize audio output latency

ADSP ALSA Driver supports audio output latency optimization by decreasing frame size. The smaller frame size the smaller latency. The best result is confirmed **at frame size 64 bytes, period count 2** without any degradation. Although frame size smaller than 64 bytes can be set but there is no guarantee for them.

- User setting:

  tinyplay thetest_FULL_s_32000_16.wav -D 0 -d 0 -p 64 -n 2

## 5.2. Capture

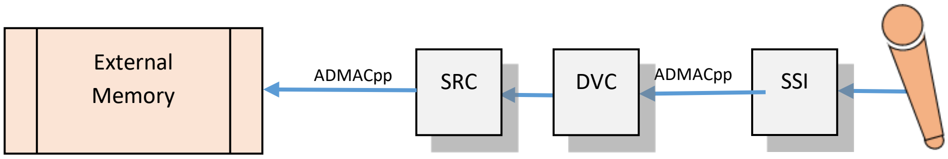



Figure 5-5 Data path for capture stream

Below table shows information of ADSP ALSA Driver sound card.

| Sound card | Description |
|---|---|
| hw:0,0,0 | ADSP ALSA sound card (card 0) with DAI 0 |
| hw:0,1,0 | ADSP ALSA sound card (card 0) with DAI 1 |
| hw:0,2,0 | ADSP ALSA sound card (card 0) with DAI 2 |
| hw:0,3,0 | ADSP ALSA sound card (card 0) with DAI 3 |

Table 5-4 Detailed information of ADSP ALSA sound card

The sound card is configured in device tree as default card (card 0).

### 5.2.1. Capture stream

ADSP ALSA Driver supports recording stream (PCM width 16/24, sample rate 32/44.1/48 kHz, frame size 4/8/16/32/64/128/256/512/1024, channel 1/2, period count 2/4).

[Note]: In 16 bit mono case, ADSP ALSA Driver only supports frame size 1024. Other frame sizes are not guarantee.

- User setting:

  tinycap cap_s_32000_16.wav -D 0 -d 3 -c2 -r32000 -b 16 -T 5 -p 512 -n 4

### 5.2.2. Setting Capture Volume

ADSP ALSA Driver supports setting volume for record stream. Value range from 0 to 799. Value 799 means that increase volume to 8 times.

- User setting:
  - Record stream

    tinycap thetest_FULL_s_32000_16.wav -D 0 -d 0 -c2 -r32000 -b 16 -T 5

  - Set volume

    tinymix -D 0 CaptureVolume 50

    or

    tinymix -D 0 7 50

  - Get information about volume

    tinymix -D 0 CaptureVolume

    or

    tinymix -D 0 7

### 5.2.3. Setting Input Sample Rate

ADSP ALSA Driver supports converting data's sample rate to value. Sample rate supported with range (32/44.1/48 KHz).

- User setting
  - Set input sample rate
    tinymix -D 0 CaptureInRate 44100
    or
    tinymix -D 0 9 44100
  - Record stream
    tinycap cap_s_48000_16.wav -D 0 -d 0 -c2 -r48000 -b 16 -T 5
  - Get information about input sample rate
    tinymix -D 0 CaptureInRate
    or
    tinymix -D 0 9

### 5.2.4. Optimize audio input latency

ADSP ALSA Driver supports audio input latency optimization by decreasing frame size. The smaller frame size the smaller latency. The best result is confirmed **at frame size 64 bytes, period count 2** without any degradation. Although frame size smaller than 64 bytes can be set but there is no guarantee for them.

- User setting:
  tinycap cap_s_32000_16.wav -D 0 -d 3 -c2 -r32000 -b 16 -T 5 -p 64 -n 2

## 5.3. TDM Playback

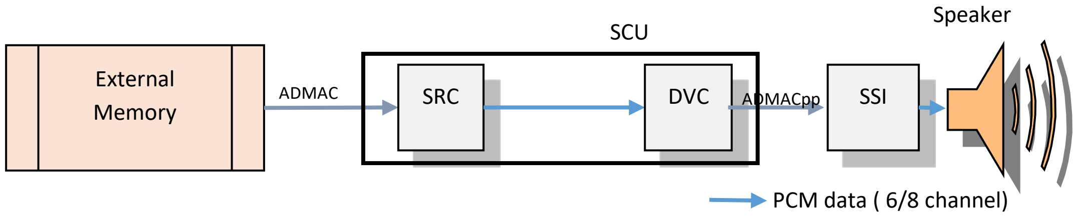



Figure 5-6 Data path for multichannel

Command is used when running aplay to play TDM stream:

# tinyplay <input> -D 0 -d 0

Explanation:

| | |
|---|---|
| -D 0 -d 0: | Card selected is 0, DAI index is 0, sub-device is 0 |
| <input>: | input file (.wav) |

### 5.3.1. TDM Playback stream

ADSP ALSA Driver supports run multichannel stream (PCM width 16/24, 6/8 channels, frame size 1024, sample rate 44.1/48 kHz). If user run a stream 32 kHz, must convert sample rate to 44.1/48 kHz, please refer 5.3.3.

[Note] Do not support sample rate conversion between 32 and 44.1 kHz, and between 48 and 44.1 kHz.

- User setting
  - Set output sample rate if it is different from 48kHz
    tinymix -D 0 TDMPlaybackOutRate 48000

  - Playback multichannel stream
    tinyplay thetest_FULL_6ch_32000_16.wav -D 0 -d 0

  - Get information about sample rate
    tinymix -D 0 TDMPlaybackOutRate

### 5.3.2. Setting TDM Playback Volume

ADSP ALSA Driver supports setting volume for TDM Playback stream. Value range from 0 to 799. But updating volume runtime is unsupported. So user needs to set volume value before running multichannel stream.

- User setting
  - Set volume
    tinymix -D 0 TDMPlaybackVolume 100

  - Playback multichannel stream
    tinyplay thetest_FULL_6ch_48000_16.wav -D 0 -d 0

  - Get information about volume
    tinymix -D 0 TDMPlaybackVolume

### 5.3.3. Setting TDM Output Sample Rate

ADSP ALSA Driver supports convert data's sample rate to other value. Range of output sample rate supported (44.1/48 kHz).

- User setting
  - Set output sample rate
    tinymix -D 0 TDMPlaybackOutRate 48000

  - Run stream
    tinyplay thetest_FULL_6ch_44100_16.wav -D 0 -d 0

  - Get information output sample rate
    tinymix -D 0 TDMPlaybackOutRate

## 5.4. TDM Capture

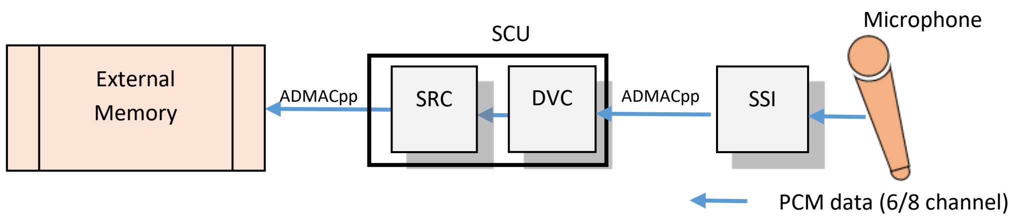



Figure 5-7 Data path for recording multichannel stream

Command are used when running tinycap to record multichannel stream:

tinycap <output> -D 0 -d 0 –c<value> -r<value> -b<name> -T <value>

Explanation:

| | |
|---|---|
| -D 0 -d 0: | Card selected is 0, DAI index is 0, sub-device is 0 |
| -c<value>: | Channel number (6/8) |
| -r<value>: | Sampling rate (32000/44100/48000) |
| -b<name>: | Format of PCM width |
| -T <value>: | Recording duration (second) |
| <output>: | Output file is raw type (.wav) |

### 5.4.1. TDM Capture stream

ADSP ALSA Driver supports recording multichannel stream (PCM width 16/24, sample rate 32/44.1/48 KHz, frame size 1024, channel 6/8). If user record a stream 32 kHz, must convert input sample rate to 44.1/48 kHz, please refer 5.4.3.

[Note] Do not support sample rate conversion between 32 and 44.1 kHz, and between 48 and 44.1 kHz.

- User setting
  tinycap output.wav -D 0 -d 0 -c8 -r48000 -b 16 -T 15

### 5.4.2. Setting TDM Capture Volume

ADSP ALSA Driver supports setting volume for record multichannel stream. Value range from 0 to 799. However, updating volume runtime is unsupported. Value 799 means that increase volume to 8 times.

- User setting
  - Set volume

    tinymix -D 0 TDMCaptureVolume 100

  - Record multichannel stream

    tinycap out.wav -D 0 -d 0 -c8 -r48000 -b 16 -T 15

  - Get information about volume

    tinymix -D 0 TDMCaptureVolume

### 5.4.3. Setting TDM Input Sample Rate

ADSP ALSA Driver supports convert input sample rate to other value. Range of input sample rate supported (44.1/48 kHz).

- User setting
  - Set input sample rate

    tinymix -D 0 TDMCaptureInRate 48000

  - Record stream

    tinycap output.wav -D 0 -d 0 -c8 -r44100 -b 16 -T 15

  - Get information input sample rate

    tinymix -D 0 TDMCaptureInRate

## 5.5. Equalizer

❖ User setting parameters for Parametric Equalizer:

| Parameter | Data format | Range | Step | Description |
|---|---|---|---|---|
| Type | 32-bit integer | 0 to 3<br>T: Through<br>P: Peaking<br>B: Bass<br>R: Treble | 1 | Specify filter type of one filter. |
| Fc[Hz] | 32-bit integer | It is specified with respect to each filter type. | 1Hz | Specify center frequency of a peaking filter. |
| Gain | Fixed point decimal (Q4.28) | -15dB to 15dB | 0.125dB | Specify gain at a center frequency of a peaking filter. |
| Base Gain | Fixed point decimal (Q4.28) | -10dB to 10dB | 0.125dB | Specify a base gain.<br>It is used for Bass/Treble filter and it is ignored for Peaking filter.<br>Summed gain of Gain and Base Gain do not have to exceed -15～15dB. |
| Q | Fixed point decimal (Q5.27) | 0.2 to 15 | 0.1 | Specify band width of a peaking/notch filter |

Table 5-5 User setting parameters for Parametric Equalizer

❖ User setting parameters for Graphic Equalizer:

| Parameter | Data format | Range | Step | Description |
|---|---|---|---|---|
| Fs[Hz] | 32-bit integer | 48kHz<br>(44.1kHz)<br>(32kHz) | 1Hz | Specify sampling frequency of input signal. |
| Gain | Fixed point decimal (Q4.28) | -10dB to 10dB | 0.125dB | Specify gain at a center frequency of a peaking/notch filter. |
| Channel | 32-bit integer | 0 to 1 | 1 | Specify which channel to set. |
| Band | 32-bit integer | 0 to 4 | 1 | Specify which band to set |

Table 5-6 User setting parameters for Graphic Equalizer

### 5.5.1. Equalizer for Playback

ADSP ALSA Driver supports setting Parametric Equalizer and Graphic Equalizer for playback stream.

Equalizer plugin does not support setting in runtime. It only runs with frame size 1024.

#### 5.5.1.1. Setting Parametric Equalizer

- Setting flow
  - Enable Equalizer control

    tinymix -D 0 PlaybackEQZSwitch 1
    or
    tinymix -D 0 12 1

    Value: 1 enable Equalizer control; 0: disable Equalizer control

  - Set Parametric Equalizer

    tinymix -D 0 PlaybackEQZControl 0 1 0 15000 94891933 268435456 268435456 2 0 15000 94891933 268435456 268435456 3 0 15000 94891933 268435456 268435456 4 0 15000 94891933 268435456 268435456 5 0 15000 94891933 268435456 268435456 6 0 15000 94891933 268435456 268435456 7 0 15000 94891933 268435456 268435456 8 0 15000 94891933 268435456 268435456 9 0 15000 94891933 268435456 268435456

    or

    tinymix -D 0 10 0 1 0 15000 94891933 268435456 268435456 2 0 15000 94891933 268435456 268435456 3 0 15000 94891933 268435456 268435456 4 0 15000 94891933 268435456 268435456 5 0 15000 94891933 268435456 268435456 6 0 15000 94891933 268435456 268435456 7 0 15000 94891933 268435456 268435456 8 0 15000 94891933 268435456 268435456 9 0 15000 94891933 268435456 268435456

    Please refer to Set Equalizer parameters of hardware for detail information about the order and range for each parameters.

  - Playback stream:

    tinyplay thetest_FULL_s_48000_16.wav -D 0 -d 0 -p 1024

  - Get information of Equalizer parameter:

    tinymix -D 0 PlaybackEQZControl

  - Get information of Equalizer status:

    tinymix -D 0 PlaybackEQZSwitch

### 5.5.1.2. Setting Graphic Equalizer

- Setting flow
  - Enable Equalizer control

    tinymix -D 0 PlaybackEQZSwitch 1

    or

    tinymix -D 0 12 1

    Value: 1 enable Equalizer control; 0: disable Equalizer control

  - Set Graphic Equalizer parameter:

    tinymix -D 0 PlaybackEQZControl 1 1 84886744 2 84886744 3 84886744 4 84886744 5 84886744

    or

    tinymix -D 0 10 1 1 84886744 2 84886744 3 84886744 4 84886744 5 84886744

    Please refer to Set Equalizer parameters of hardware for detail information about the order and range for each parameters.

  - Playback stream:

    tinyplay thetest_FULL_s_48000_16.wav -D 0 -d 0 -p 1024

  - Get information of Equalizer parameter:

    tinymix -D 0 PlaybackEQZControl

  - Get information of Equalizer status:

    tinymix -D 0 PlaybackEQZSwitch

### 5.5.2. Equalizer for Capture

ADSP ALSA Driver supports setting Parametric Equalizer and Graphic Equalizer for record stream.

Equalizer plugin does not support setting in runtime. It only runs with frame size 1024.

#### 5.5.2.1. Setting Parametric Equalizer

- Setting flow
  - Enable Equalizer control

    tinymix -D 0 CaptureEQZSwitch 1
    or
    tinymix -D 0 13 1

    Value 1: enable Equalizer control; 0: disable Equalizer control

  - Set Parametric Equalizer

    tinymix -D 0 CaptureEQZControl 0 1 0 15000 94891933 268435456 268435456 2 0 15000 94891933 268435456 268435456 3 0 15000 94891933 268435456 268435456 4 0 15000 94891933 268435456 268435456 5 0 15000 94891933 268435456 268435456 6 0 15000 94891933 268435456 268435456 7 0 15000 94891933 268435456 268435456 8 0 15000 94891933 268435456 268435456 9 0 15000 94891933 268435456 268435456
    or
    tinymix -D 0 11 0 1 0 15000 94891933 268435456 268435456 2 0 15000 94891933 268435456 268435456 3 0 15000 94891933 268435456 268435456 4 0 15000 94891933 268435456 268435456 5 0 15000 94891933 268435456 268435456 6 0 15000 94891933 268435456 268435456 7 0 15000 94891933 268435456 268435456 8 0 15000 94891933 268435456 268435456 9 0 15000 94891933 268435456 268435456

    Please refer to Set Equalizer parameters of hardware for detail information about the order and range for each parameters.

  - Record stream:

    tinycap cap_s_32000_16.wav -D 0 -d 0 -c2 -r32000 -b 16 -T 5 -p 1024

  - Get information of Equalizer parameter:

    tinymix -D 0 CaptureEQZControl

  - Get information of Equalizer status:

    tinymix -D 0 CaptureEQZSwitch

### 5.5.2.2. Setting Graphic Equalizer

- Setting flow
  - Enable Equalizer control

    tinymix -D 0 CaptureEQZSwitch 1

    or

    tinymix -D 0 13 1

    Value 1 to enable, 0 to disable Equalizer control

  - Set Graphic Equalizer parameter:

    tinymix -D 0 CaptureEQZControl 1 1 84886744 2 84886744 3 84886744 4 84886744 5 84886744

    or

    tinymix -D 0 11 1 1 84886744 2 84886744 3 84886744 4 84886744 5 84886744

    Please refer to Set Equalizer parameters of hardware for detail information about the order and range for each parameters.

  - Record stream:

    tinycap cap_s_32000_16.wav -D 0 -d 0 -c2 -r32000 -b 16 -T 5  -p 1024

  - Get information of Equalizer parameter:

    tinymix -D 0 CaptureEQZControl

    or

    tinymix -D 0 11

  - Get information of Equalizer status:

    tinymix -D 0 CaptureEQZSwitch

    or

    tinymix -D 0 13

# 6 Appendix

## 6.1 Error code

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

Table 6-1 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 | |

## 6.2 Structure and type definitions

Below tables shows structures and type definitions used in this material.

Table 6-2 Structures or type definition 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* |
| spinlock_t | It contains detail information of spinlock. | *https://elixir.bootlin.com/linux/latest/source/include/linux/spinlock_types.h* |

Table 6-3 shows list of structures are defined in ADSP ALSA Driver.

Table 6-3 Structures defined in ADSP ALSA Driver

| Structures | Size (bytes) | Description |
|---|---|---|
| snd_adsp_control | 1856 | It is used to store parameters from use |
| snd_adsp_card | 2136 | It is used to store data for ALSA sound card |
| snd_adsp_base_info | 72 | It is used to store base data for ADSP sound card |
| snd_adsp_playback | 96 | It is used to store necessary information for Renderer |
| snd_adsp_record | 96 | It is used to store necessary information for Capture |
| snd_adsp_tdm_playback | 88 | It is used to store necessary information for TDM Renderer |
| snd_adsp_tdm_record | 88 | It is used to store necessary information for TDM Capture |

### 6.2.1 snd_adsp_control structure

Table 6-4 snd_adsp_control structure information

| Member name | | Outline |
|---|---|---|
| int | vol_rate[DIRECT_NUM][MAX_DAI_IDX – 1] | Volume rate value for Capture/Renderer |
| int | tdm_vol_rate[DIRECT_NUM] | Volume rate for TDM Capture/TDM Renderer |
| int | sample_rate[DIRECT_NUM][MAX_DAI_IDX – 1] | Out sample rate with Renderer, in sample rate with Capture |
| int | tdm_sample_rate[DIRECT_NUM] | Out sample rate with TDM Renderer, in sample rate with TDM Capture |
| int | rdr_out_ch[MAX_DAI_IDX – 1] | Output channel of Renderer |
| struct xf_adsp_equalizer_params | eqz_params[DIRECT_NUM][MAX_DAI_IDX – 1] | Equalizer parameters |
| int | eqz_switch[DIRECT_NUM][MAX_DAI_IDX – 1] | Equalizer switch |

[Note] DIRECT_NUM is a macro representing the number of streams that can execute concurrently. In this material, DIRECT_NUM is 2, which means it supports running a playback stream and a capture stream concurrently.

MAX_DAI_IDX is a macro representing the number of stream types. In this material, MAX_DAI_IDX is 5, which means it supports 4 playback/record streams and 1 TDM playback/TDM record stream.

### 6.2.2 snd_adsp_device_params structure

Table 6-5 snd_adsp_device_params structure information

| Member name | | Outline |
|---|---|---|
| int | dev[MAX_DEV_NUM] | Device parameter |
| int | dma[MAX_DEV_NUM] | DMA parameter |
| int | mix_usage | MIX control |

[Note] MAX_DEV_NUM is 2 because we have maximum 2 devices supported.

### 6.2.3 snd_adsp_base_info structure

Table 6-6 snd_adsp_base_info structure information

| Member name | | Outline |
|---|---|---|
| int | state | Indicator to show whether the component is running or not |
| int | handle_id | Target handle ID of ALSA driver |
| unsigned char * | buffer | Data buffer |
| struct xf_pool * | buf_pool | Buffer pool for data transfer |
| int | buf_bytes | Size of each allocated data buffer |
| int | buf_cnt | Total buffer count of shared memory |
| int | buf_queue | Number of data buffer in the queue |
| int | hw_idx | Hardware index in bytes |
| int | period_bytes | Number of bytes in one period |
| struct snd_pcm_substream * | substream | Substream runtime object |
| spinlock_t | lock | Spinlock data |
| int | runtime_err | Runtime error indicator |
| int | old_app_ptr | Old application buffer position |

### 6.2.4 snd_adsp_playback structure

Table 6-7 snd_adsp_playback structure information

| Member name | | Outline |
|---|---|---|
| struct snd_adsp_base_info | base | Base information of stream |
| struct xf_adsp_renderer * | renderer | Renderer component’s data |
| struct xf_adsp_equalizer * | equalizer | Equalizer component’s data |
| int | rdr_state | Renderer component’s state |
| int | eqz_state | Equalizer component’s state |

### 6.2.5 snd_adsp_record structure

Table 6-8 snd_adsp_record structure information

| Member name | | Outline |
|---|---|---|
| struct snd_adsp_base_info | base | Base information of stream |
| struct xf_adsp_capture * | capture | Capture component's data |
| struct xf_adsp_equalizer * | equalizer | Equalizer component's data |
| int | cap_state | Capture component's state |
| int | eqz_state | Equalizer component's state |

### 6.2.6 snd_adsp_card structure

Table 6-9 snd_adsp_card structure information

| Member name | | Outline |
|---|---|---|
| struct snd_adsp_playback * | playback[MAX_DAI_IDX – 1] | Playback data |
| struct snd_adsp_record * | record[MAX_DAI_IDX – 1] | Record data |
| struct snd_adsp_tdm_playback * | tdm_playback | TDM playback data |
| struct snd_adsp_tdm_record * | tdm_record | TDM record data |
| struct snd_adsp_control | ctr_if | Structure containing parameter information for control |
| struct snd_adsp_device_params | dev_params[MAX_DAI_IDX][DIRECT_NUM] | Device parameters for DAIs |

### 6.2.7 snd_adsp_tdm_playback structure

Table 6-10 snd_adsp_tdm_playback structure information

| Member name | | Outline |
|---|---|---|
| struct snd_adsp_base_info | base | Base information of stream |
| struct xf_adsp_tdm_renderer * | tdm_renderer | TDM Renderer component's data |
| int | state | TDM Renderer component's state |

### 6.2.8 snd_adsp_tdm_record structure

Table 6-11 snd_adsp_tdm_record structure information

| Member name | | Outline |
|---|---|---|
| struct snd_adsp_base_info | base | Base information of stream |
| struct xf_adsp_tdm_capture * | tdm_capture | TDM Capture component's data |
| int | state | TDM Capture component's state |

| Revision History | ADSP Interface for Android Application Note - ALSA - |
|---|---|

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | May. 24, 2019 | - | New Create |

**CONFIDENTIAL**

ADSP Driver for Android Application Note - ALSA -

Publication Date: May 24, 2019 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 Corporation**
TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

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

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

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

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

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

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

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

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

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

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

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

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





Colophon 5.2

RENESAS

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

RCG3AHPDA9001ZDO



Renesas Electronics Corporation