User's Manual

CONFIDENTIAL

ADSP Interface for Android
RCG3AHIFA9001ZDP

User's Manual

RCG3AHIFA9001ZDPE

Rev. 1.00 May, 2019

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)

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. Overview of this document.

In this chapter, overview of ADSP Interface for Android is explained.

1.2. The architecture of the Software and scope of this document

The architecture of ADSP Interface for Android is shown in Figure 1-1. ADSP Interface for Android is a user space library which provides the interface to control ADSP Framework via ADSP Driver for Android.

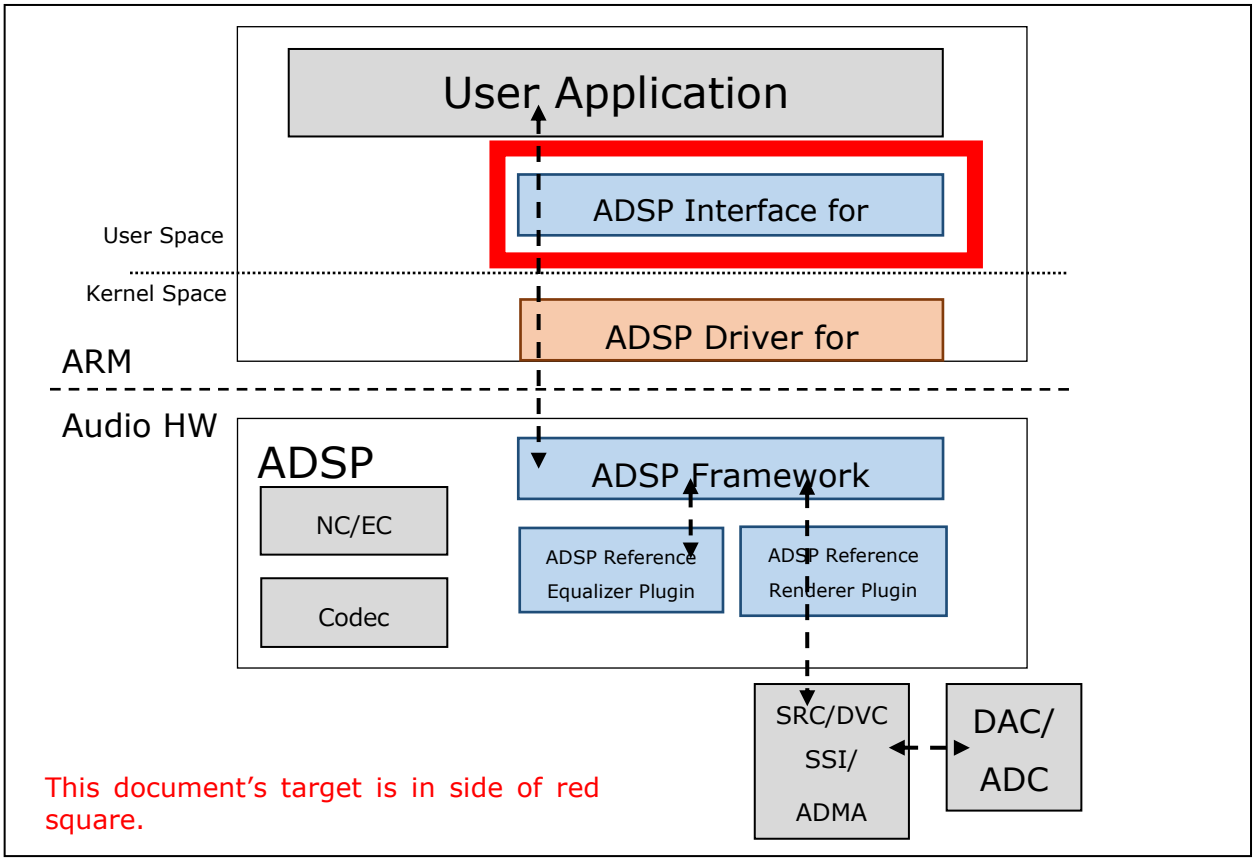

Figure 1-1 The software architecture

1.3. Software necessary to be prepared in advance

ADSP Driver for Android should be loaded in advance to use ADSP Interface for Android.

1.4. Related documents

Table 1-1 shows related documents.

Table 1-1 The list of related documents

No.	Name	Published by
[1]	R-Car Series, 3rd Generation User's Manual: Hardware	Renesas Electronics Corporation
[2]	ADSP Framework User's Manual	Renesas Electronics Corporation

1.5. The definition of common types

Table 1-2 shows the list of type definitions used in ADSP Interface for Android.

Table 1-2 The list of common types

type	size [byte]		
s8	1	signed 8 bit integer	-128 to 127
s16	2	signed 16 bit integer	-32768 to 32767
s32	4	signed 32 bit integer	-2147483648 to 2147483647
u8	1	unsigned 8 bit integer	0 to 255
u16	2	unsigned 16 bit integer	0 to 65535
u32	4	unsigned 32 bit integer	0 to 4294967295

[notice] the size of a pointer depends on architecture.

2. Software specification

2.1. The list of functions

Table 2-1 shows the functions provided by this software. See 2.3 for more detailed specification of the functions.

Table 2-1 The list of functions

name	outline
xf_proxy_init	ADSP Interface for Android initialization
xf_proxy_close	ADSP Interface for Android close
xf_proxy_pool	Get pointer to auxiliary shared buffer pool from ADSP Interface for Android struct
xf_pool_alloc	Allocate buffer pool
xf_pool_free	Release buffer pool
xf_buffer_get	Get buffer from buffer pool
xf_buffer_put	Return buffer to buffer pool
xf_buffer_data	Get buffer address
xf_buffer_length	Get buffer size
xf_open	Get handle of ADSP Plugin
xf_close	Release handle of ADSP Plugin
xf_handle_aux	Get the address of shared buffer from the handle of ADSP Plugin
xf_handle_auxlen	Get the size of shared buffer from the handle of ADSP Plugin.
xf_route	Set routing of ADSP Plugin
xf_unroute	Release routing of ADSP Plugin
xf_command	Send a command to ADSP Plugin
xf_response_put	Send a message to pipe
xf_response_get	Get a message from pipe
xf_response_cb	Callback function

2.2. The list of structures

Table 2-2 shows the list of structures which user should allocate memory in using the software. See 2.5 for more detailed specification of the structures.

Table 2-2 The list of structures

name	outline	remark
xf_proxy_t	The structure of ADSP Interface for Android	Direct access to a member of the structure is prohibited.
xf_handle_t	The handle structure of ADSP Plugin	Direct access to a member of the structure is prohibited.
xf_pool_t	The structure of buffer pool	Direct access to a member of the structure is prohibited.
xf_buffer_t	The structure of buffer	Direct access to a member of the structure is prohibited.
xf_set_param_msg_t	The structure of XF_SET_PARAM	See the manual of ADSP Framework for details
xf_get_param_msg_t	The structure of XF_GET_PARAM	See the manual of ADSP Framework for details
xf_start_msg_t	The structure of Runtime Initialization Stage	See the manual of ADSP Framework for details
xf_user_msg_t	The structure of user message	See the manual of ADSP Framework for details

2.3. Function specification

2.3.1. xf_proxy_init

xf_proxy_init		
Synopsis	This function initializes ADSP Interface for Android. It should be called once in advance to call other functions.	

Syntax	`int xf_proxy_init(`
	`  xf_proxy_t *proxy,`
	`  u32 core);`	

Parameter	proxy	The pointer to ADSP Interface for Android structure
	core	Specify 0.
Return values	success	0
	fail	non-zero

2.3.2. xf_proxy_close

xf_proxy_close		
Synopsis	This function terminates ADSP Interface for Android. After calling this function, other function can't be called until executing xf_proxy_init.	
Syntax	`void xf_proxy_close(` `  xf_proxy_t *proxy);`	
Parameter	proxy	The pointer to ADSP Interface for Android structure
Return value	None	

2.3.3. xf_proxy_pool

xf_proxy_pool		
Synopsis	This function gets the member of auxiliary shared buffer pool from ADSP Interface for Android structure. This function is a function-like macro.	
Syntax	`#define xf_proxy_pool(proxy) ((proxy)->aux)`	
Parameter	proxy	The pointer to ADSP Interface for Android structure
Return value	The pointer to AUX shared buffer pool	

2.3.4. xf_pool_alloc

xf_pool_alloc			
Synopsis	This function allocates buffer pool.		

Syntax	`int xf_pool_alloc(`
	`  xf_proxy_t *proxy,`
	`  u32 number,`
	`  u32 length,`
	`  xf_pool_type_t type,`
	`  xf_pool_t **pool);`		

Parameter	proxy	The pointer to ADSP Interface for Android structure	
	number	The number of buffers to allocate	
	length	The length of each buffer	
	type	The kind of buffer pool to allocate	
		XF_POOL_AUX	AUX buffer
		XF_POOL_INPUT	Input buffer
		XF_POOL_OUTPUT	Output buffer
	pool	The double pointer to buffer pool to allocate. When allocating AUX buffer, specify the pointer to xf_proxy_pool macro.	
Return value	success	0	
	fail	non-zero	

2.3.5. xf_pool_free

xf_pool_free		
Synopsis	This functions releases buffer pool	
Syntax	`void xf_pool_free(` `  xf_pool_t *pool);`	
Parameter	pool	The pointer to buffer pool to be released.
Return value	None	

2.3.6. xf_buffer_get

xf_buffer_get		
Synopsis	This function gets buffer from buffer pool.	
Syntax	`xf_buffer_t * xf_buffer_get(` `  xf_pool_t *pool);`	
Parameter	pool	The pointer to buffer pool from which buffer is obtained.
Return value	success	The address of buffer
	fail	0

2.3.7. xf_buffer_put

xf_buffer_put		
Synopsis	This functions returns buffer to buffer pool.	
Syntax	`void xf_buffer_put(` `  xf_buffer_t *buffer);`	
Parameter	buffer	The pointer to the buffer to be returned
Return value	None	

2.3.8. xf_buffer_data

xf_buffer_data		
Synopsis	This function gets buffer address.	
Syntax	`static inline void* xf_buffer_data(` `  xf_buffer_t *buffer);`	
Parameter	buffer	The pointer to the buffer
Return value	The address of the buffer	

2.3.9. xf_buffer_length

xf_buffer_length		
Synopsis	This function gets buffer size.	
Syntax	`static inline size_t xf_buffer_length(` `  xf_buffer_t *buffer);`	
Parameter	buffer	The pointer to the buffer
Return value	The size of the buffer	

2.3.10. xf_open

xf_open		
Synopsis	This function gets the handle of ADSP Plugin.	

Syntax	`int xf_open(xf_proxy_t *proxy,`
	`  xf_handle_t *handle,`
	`  xf_id_t id,`
	`  u32 core,`
	`  xf_response_cb response);`	

Parameter	proxy	The pointer to ADSP Interface for Android structure
	handle	The pointer to ADSP Plugin handle structure
	id	The component ID of the plugin.
	core	Specify 0
	response	The pointer to the callback function
Return value	success	0
	fail	non-zero

2.3.11. xf_close

xf_close		
Synopsis	This function releases the handle of ADSP Plugin.	
Syntax	`void xf_close(` `  xf_handle_t *handle);`	
Parameter	handle	The pointer to ADSP Plugin handle structure
Return value	None	

2.3.12. xf_handle_aux

xf_handle_aux		
Synopsis	This function gets the address of shared buffer from ADSP Plugin handle.	
Syntax	`static inline void* xf_handle_aux(` `  xf_handle_t *handle);`	
Parameter	handle	The pointer to ADSP Plugin handle structure
Return value	The address of shared buffer	

2.3.13. xf_handle_auxlen

xf_handle_auxlen		
Synopsis	This function gets the size of shared buffer from ADSP Plugin handle.	
Syntax	`static inline size_t xf_handle_auxlen(` `  xf_handle_t *handle);`	
Parameter	handle	The pointer to ADSP Plugin handle structure
Return value	The size of shared buffer	

2.3.14. xf_route

xf_route		
Synopsis	This function connects the ports between 2 ADSP Plugins. It enables to use the data of source ADSP Plugin as the input of target ADSP Plugin. It is impossible to connect one port to multiple ports.	

Syntax	`int xf_route(xf_handle_t *src,`
	`  u32 src_port,`
	`  xf_handle_t *dst,`
	`  u32 dst_port,`
	`  u32 num,`
	`  u32 size,`
	`  u32 align);`	

Parameter	src	The pointer of source ADSP Plugin handle structure
	src_port	The source port
	dst	The pointer of target ADSP Plugin handle structure
	dst_port	The target port
	num	The number of buffers allocated between the ports
	size	The size of buffers allocated between the ports
	align	The align of buffers allocated between the ports
Return value	success	0
	fail	non-zero

2.3.15. xf_unroute

xf_unroute		
Synopsis	This function disconnects the connection between the ports of 2 ADSP Plugins.	
Syntax	`int xf_unroute(xf_handle_t *src,` `  u32 src_port);`	
Parameter	src	The pointer of source ADSP Plugin handle structure
	src_port	The source port
Return value	success	0
	fail	non-zero

2.3.16. xf_command

xf_command		
Synopsis	This function sends a command to ADSP Plugin.	

Syntax	`int xf_command(xf_handle_t *handle,`
	`  u32 port,`
	`  u32 opcode,`
	`  void *buffer,`
	`  u32 length)`	

Parameter	handle	The pointer of target ADSP Plugin handle structure
	port	The target port of ADSP Plugin
	opcode	Command code (see user manuals of ADSP Framework and ADSP Plugin)
	buffer	The pointer to the buffer in which command is stored.
	length	The size of the buffer in which command is stored.
Return value	success	0
	fail	non-zero

2.3.17. xf_response_put

xf_response_put		
Synopsis	This function sends a message to pipe. The main usage is to send a message from callback function to main process.	
Syntax	`static inline int xf_response_put(xf_handle_t *handle,` `  xf_user_msg_t *msg);`	
Parameter	handle	The pointer to ADSP Plugin handle structure which sends a command.
	msg	The pointer to user message structure
Return value	success	0
	fail	non-zero

2.3.18. xf_response_get

xf_response_get		
Synopsis	This function gets a message from pipe. It waits for reception of a message if no message exists. The main usage is to wait for the finish of callback process after transmission of a command.	
Syntax	`static inline int xf_response_get(xf_handle_t *handle,` `  xf_user_msg_t *msg);`	
Parameter	handle	The pointer to ADSP Plugin handle structure which sends a command.
	msg	The pointer to user message structure
Return value	success	0
	fail	non-zero

2.4. Callback function specification

2.4.1. xf_response_cb

xf_response_cb		
Synopsis	This function performs ADSP Plugin callback feature. It is necessary for the user to make the callback function with the format of the syntax. Register the pointer to the callback function as the response parameter when executing xf_open.	
Syntax	`void (*xf_response_cb)(xf_handle_t *h,` `  xf_user_msg_t *msg);`	
Parameter	h	The pointer to ADSP Plugin handle structure
	msg	The pointer to user message structure
Return value	None	

2.5. Structures specification

2.5.1. xf_proxy_t

xf_proxy_t		
Synopsis	This is ADSP Interface for Android structure. Direct access to a member of the structure is prohibited.	
Access function	xf_proxy_pool	This function gets the pointer to AUX shared buffer pool in this structure.

2.5.2. xf_handle_t

xf_handle_t		
Synopsis	This is ADSP Plugin handle structure. Direct access to a member of the structure is prohibited.	
Access function	xf_handle_aux	This function gets the address of shared buffer in this structure.
	xf_handle_auxlen	This function gets the size of shared buffer in this structure.

2.5.3. xf_pool_t

xf_pool_t		
Synopsis	This is buffer pool structure. Direct access to a member of the structure is prohibited. Allocate this structure as many as necessary in / out buffer pools.	
Access function	xf_buffer_get	This function gets buffer from buffer pool allocated by this structure.
	xf_buffer_put	This function returns buffer to buffer pool.

2.5.4. xf_buffer_t

xf_buffer_t		
Synopsis	This is buffer structure. Direct access to a member of the structure is prohibited. Allocate this structure as many as necessary in / out buffers.	
Access function	xf_buffer_data	This function gets the address of buffer in this structure.
	xf_buffer_length	This function gets the size of buffer in this structure.

2.5.5. xf_user_msg_t

xf_user_msg_t			
Synopsis	This is user message structure. Specify the pointer of this structure as a parameter of callback function.		
Member	u32	id	Component ID (only used for administration and not used by user)
	u32	opcode	Command code (see user's manuals of ADSP Framework and ADSP Plugin for details)
	u32	length	The length of the data stored in the buffer
	void *	buffer	The address of the buffer

3. Process sequence

3.1. Initialization flow

Figure 3-1 shows the initialization flow.

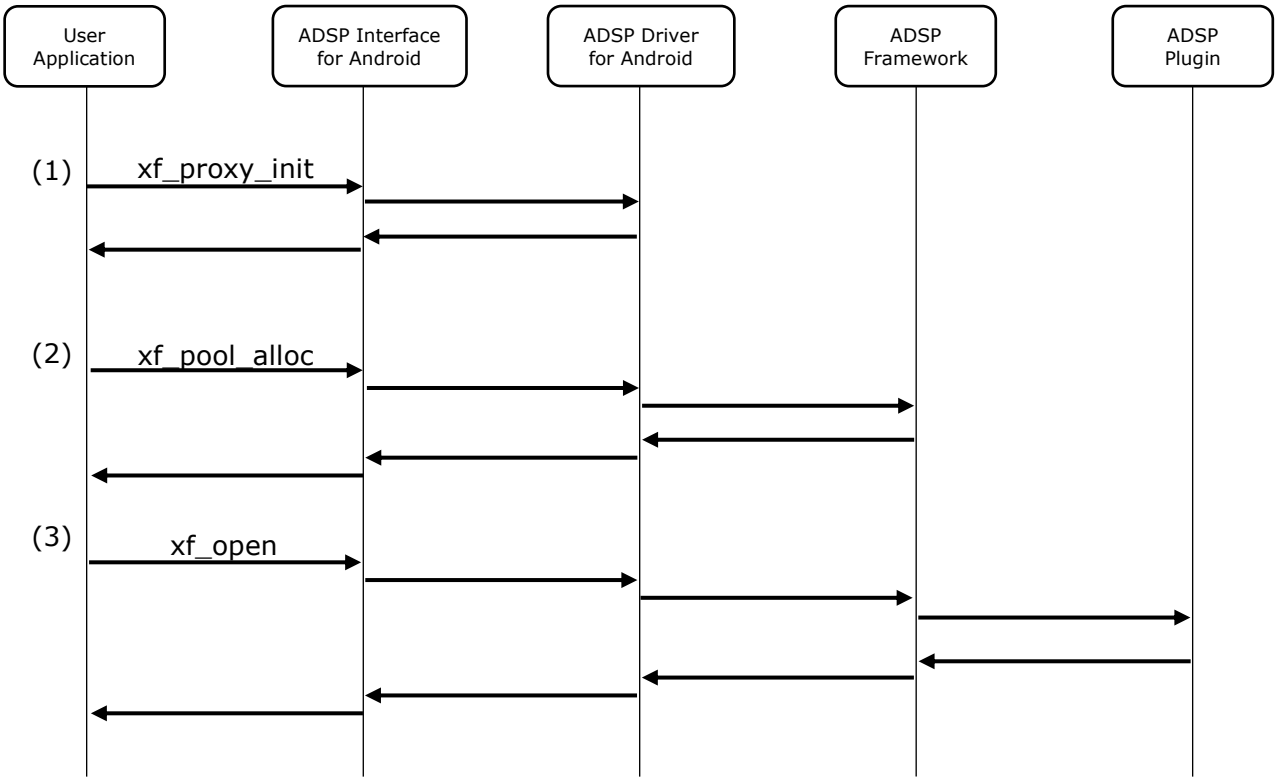

Figure 3-1 Initialization flow

(1) The xf_proxy_init performs initialization of ADSP Interface for Android and ADSP Driver for Android.
(2) The xf_pool_alloc performs allocation of the shared buffer pool whose type is XF_POOL_AUX. Specify the pointer returned from xf_proxy_pool in advance.
(3) xf_open performs initialization of ADSP Plugin. A shared buffer whose type is XF_POOL_AUX is allocated to ADSP Plugin from shared buffer pool.

3.2. Flow of sending a command

Figure 3-2 shows the reference flow to send / receive a command.

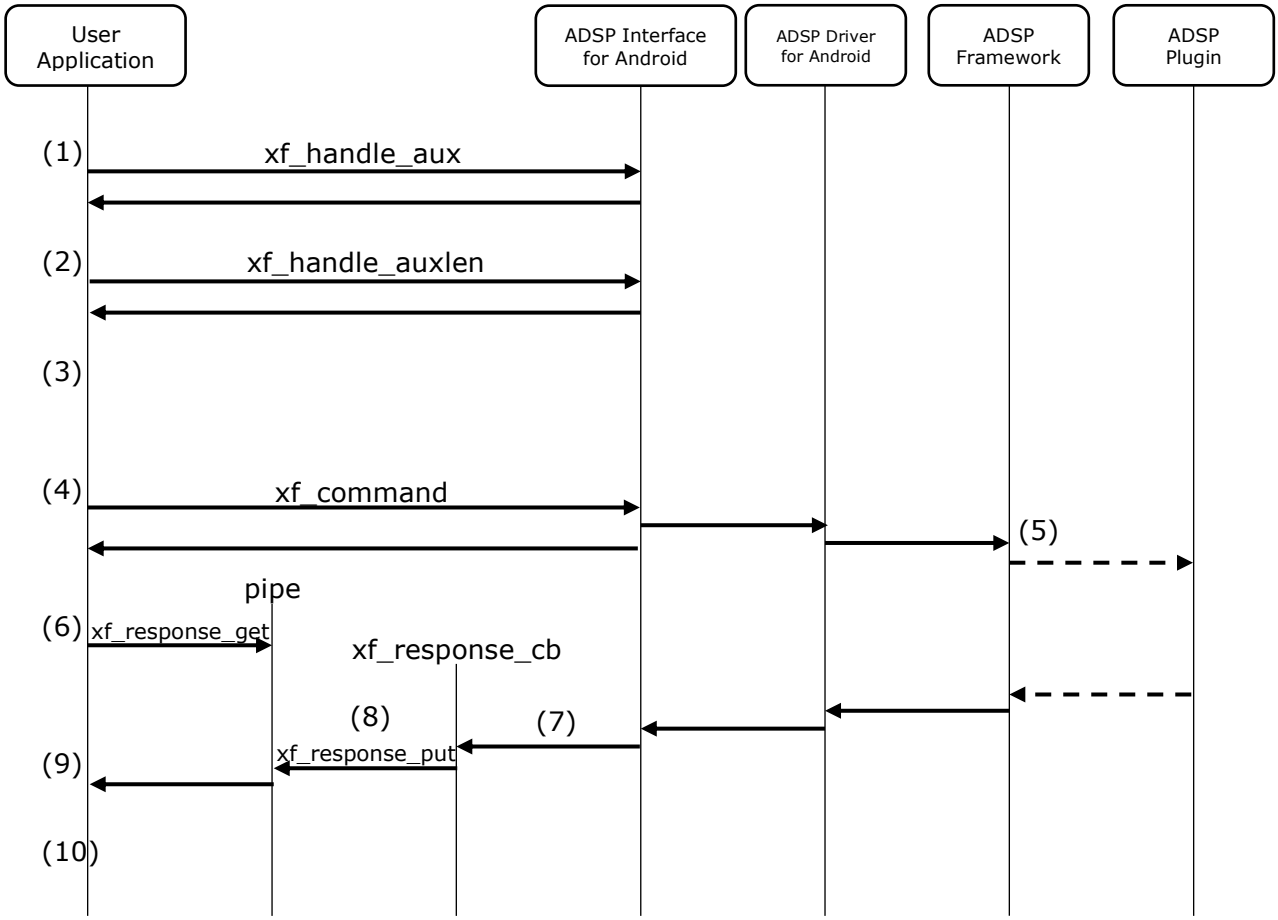

Figure 3-2 The standard flow to send / receive a command

(1) xf_handle_aux gets the address of buffer in which command is stored.
(2) xf_handle_auxlen gets the size of the buffer.
(3) Store a command to the buffer. Take care that the command does not exceed the size of the buffer.
(4) xf_command sends the command. The function ends after the finish of sending.
(5) Some commands are processed by ADSP Framework, others by ADSP Plugin.
(6) xf_response_get waits for a message from pipe.
(7) The callback function registered by xf_open is called after finish of the process of command.
(8) xf_response_put sends the received message to pipe.
(9) When xf_response_put is executed, xf_response_get stops. If the order of (6) and (8) is inverted, xf_response_get stops immediately.
(10) Continue process according to received message.

3.3. Flow to allocate input / output buffer

Figure 3-1 shows the reference flow to allocate input / output buffer.

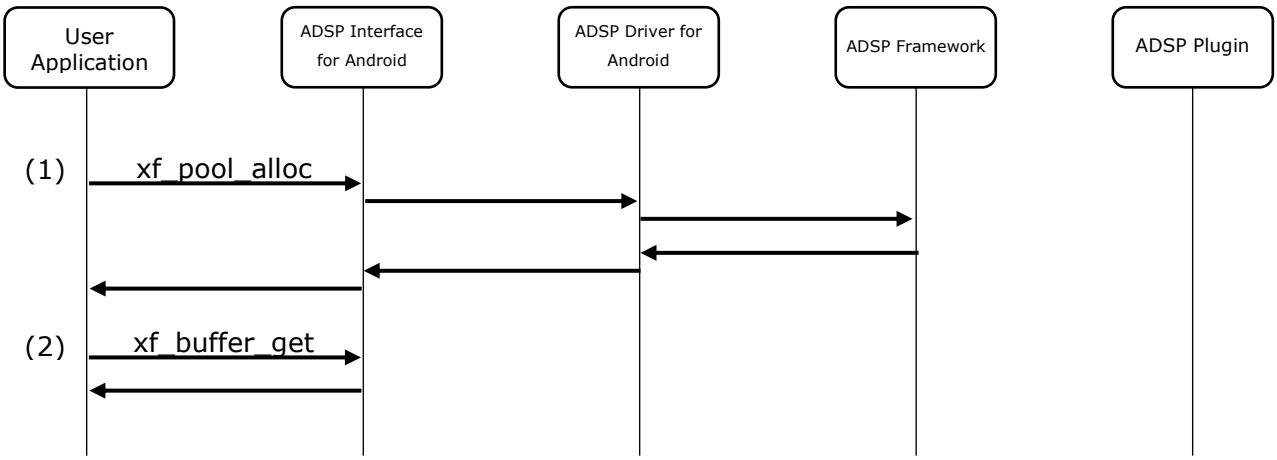

Figure 3-3 The standard flow to allocate input / output buffer

(1) xf_pool_alloc allocates buffer pool whose type is XF_POOL_INPUT or XF_POOL_OUTPUT.
(2) xf_buffer_get gets buffer from the buffer pool.

3.4. Flow of send / receive input buffer

Figure 3-2 shows the reference flow to send / receive input buffer.

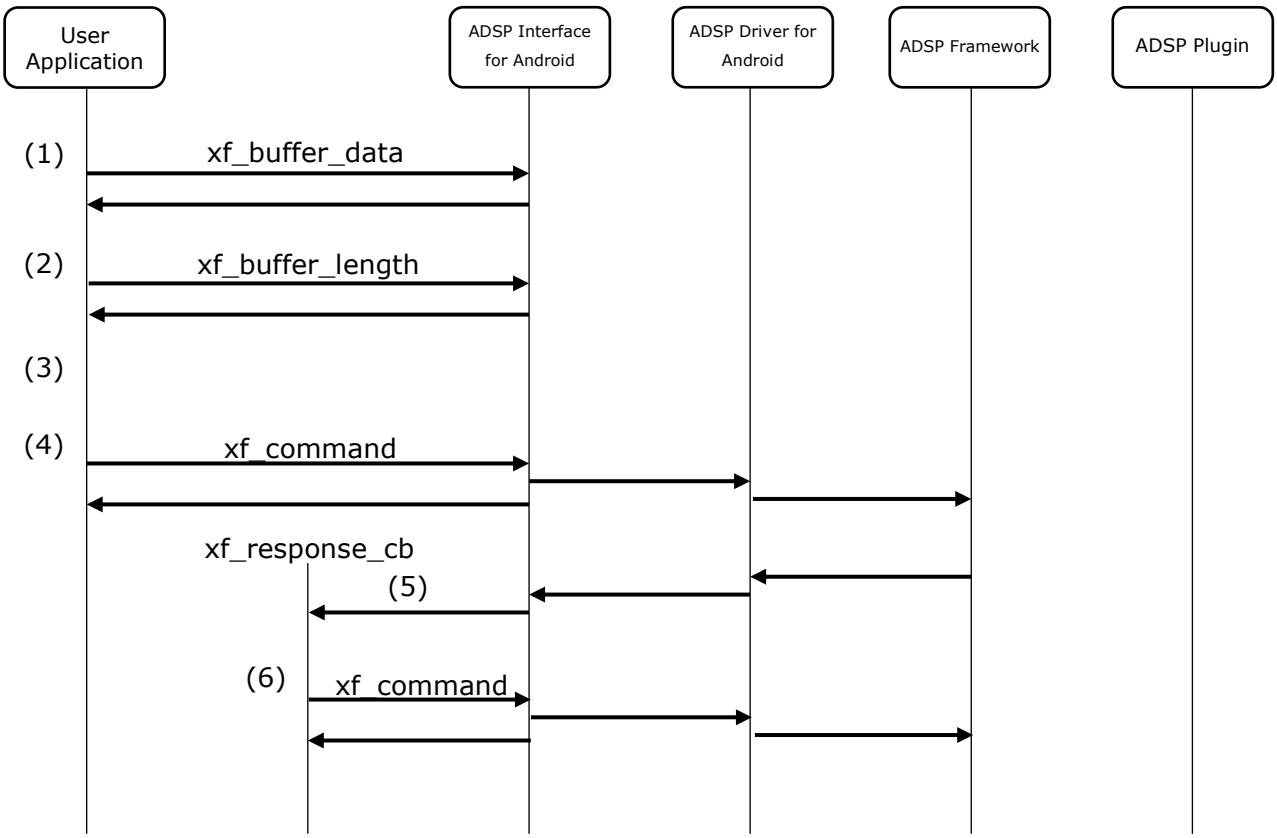

Figure 3-4 The standard flow to send / receive data

(1) xf_buffer_data gets the address of buffer to send data.
(2) xf_buffer_length gets the size of the buffer.
(3) Store data to the buffer. Take care that the data does not exceed the size of the buffer.
(4) xf_command sends the data in the buffer. The function ends after the finish of sending. Specify XF_EMPTY_THIS_BUFFER for the command code. See ADSP Framework user's manual for details.
(5) The callback function registered by xf_open is called after finish of the process.
(6) If necessary, continue to store next data to the buffer and send again. It is also possible to use another thread, not by callback function.

3.5. Flow to send / receive output buffer

Figure 3-2 shows the reference flow to send / receive output buffer.

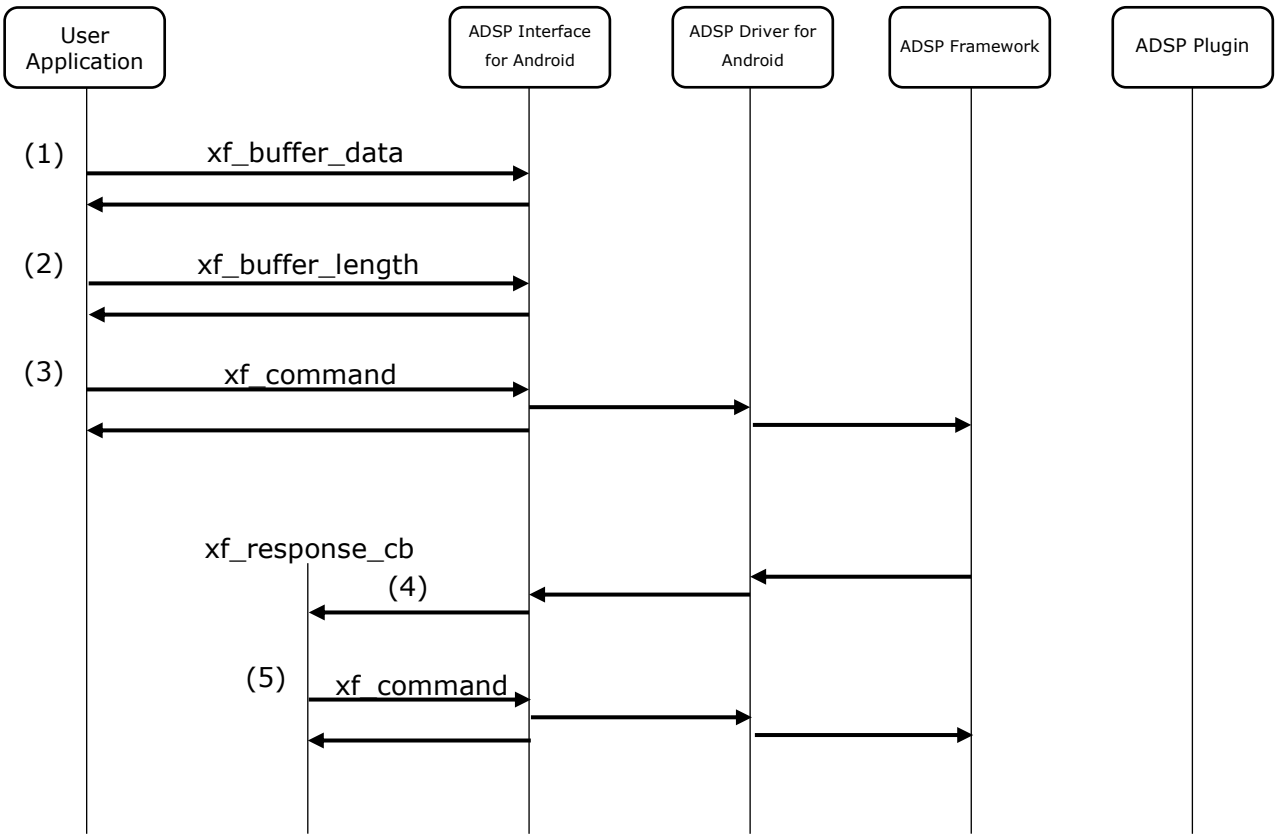

Figure 3-5 The flow to send / receive output buffer

(1) xf_buffer_data gets the address of buffer to send data.
(2) xf_buffer_length gets the size of the buffer.
(3) xf_command sends the data in the buffer. The function ends after the finish of sending. Specify XF_FILL_THIS_BUFFER for the command code. See ADSP Framework user's manual for details.
(4) The callback function registered by xf_open is called after finish of the process.
(5) Get the output data from the buffer after the process is finished. If necessary, continue to store next data to the buffer and send again. It is also possible to use another thread, not by callback function.

3.6. Flow to release input / output buffer

Figure 3-1 shows the reference flow to release input / output buffer.

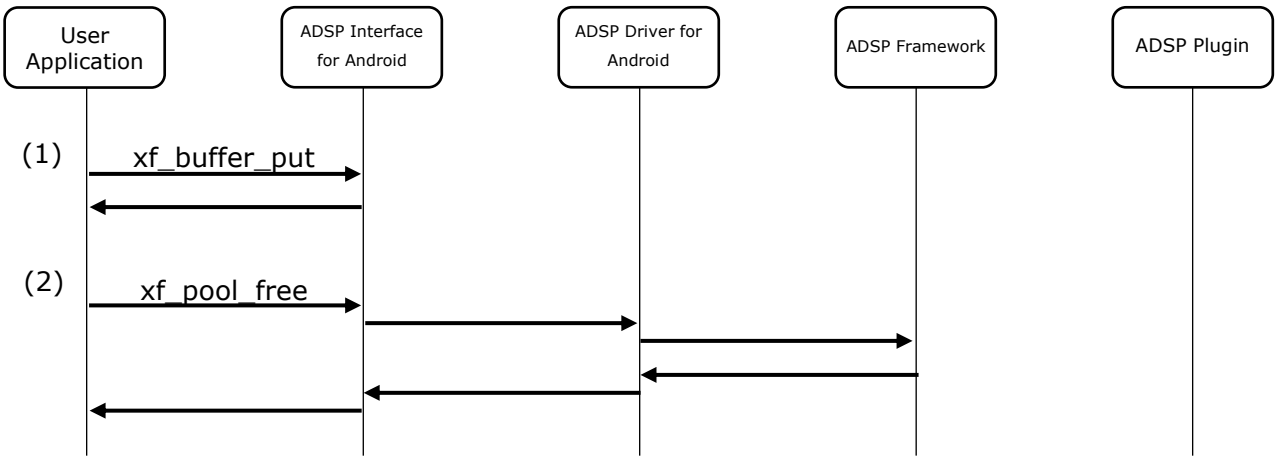

Figure 3-6 The standard flow to release input / output buffer

(1) xf_buffer_put returns all grabbed buffers to the buffer pool.
(2) xf_pool_free releases the buffer pool.

3.7. Termination flow

Figure 3-1 shows termination flow.

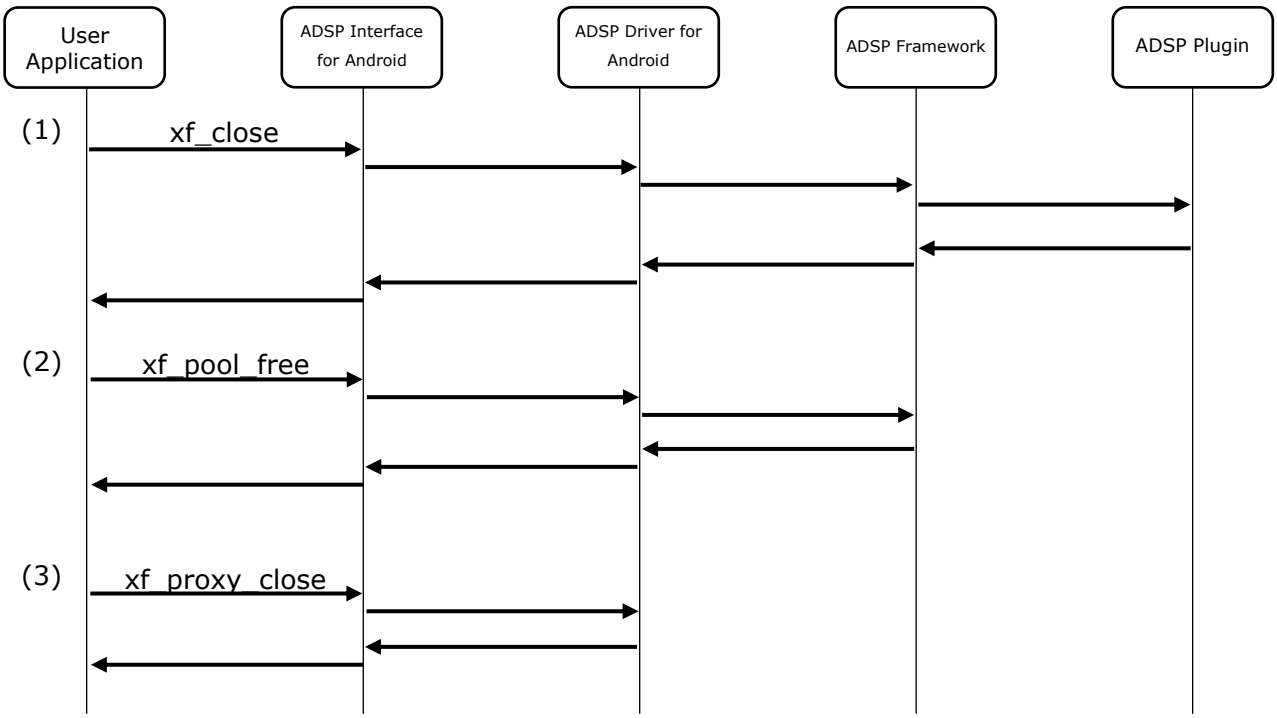

Figure 3-7 Termination flow

(1) xf_close performs ADSP Plugin termination.

(The following process should be performed only when ADSP Interface for Android is also terminated.)

(2) xf_pool_free releases the shared buffer pool whose type is XF_POOL_AUX by using the return value of xf_proxy_pool.
(3) xf_proxy_close terminates ADSP Interface for Android and ADSP Driver for Android.

4. Notes

This section describes the notice of developing user programs.

4.1. Function Call

User programs which calls the functions in this specification should obey the calling rules of compiler.

4.2. Other notes

4.2.1. Allocation of memory

Before calling the functions in this specification, allocate necessary memory area and each structure used for the parameters of each function.

4.2.2. Out of range memory access

The functions in this specification never access out of allocated memory or related I/O.

4.2.3. Combination with other applications

Take care not to duplicate symbol names when other applications are combined with other programs.

4.2.4. Monitoring on Performance

The products embedding this Software shall observe performance of the Software periodically with Watch Dog timer or such functions in order not to damage system performance.

5. Appendix

This section is to help user understand the usage of the OMX interface.
Below table show target platforms support for each use case.
Table 5-1 Target environment for each use case.

Use case	Chip	Board
Playback/Record/Route	H3/M3/M3N/E3	Salvator, Ebisu board
TDM	H3/M3	Starter KIT –Kingfisher board

5.1. Standard Use Case: Playback

Figure 5-1, shows an example of the audio data path are transferred which use renderer function.

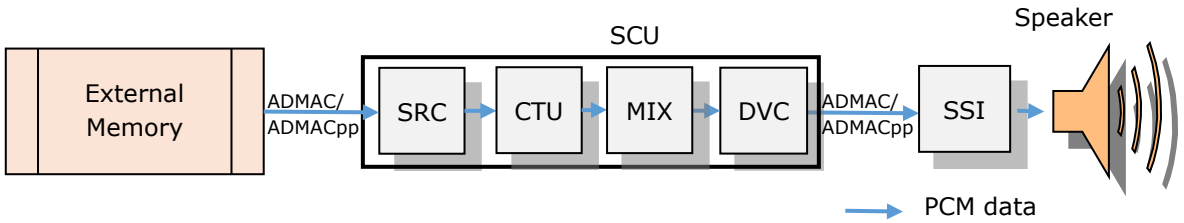

Figure 5-1 Audio data path for renderer function

Below table shows supported features and its usage for OMX renderer.
Table 5-2 Supported features for OMX Renderer Interface

Name	Usage
Audio-DMAC/ Audio-DMACpp	Transfer data between two audio modules; memory and audio module. **[Note]**: About Rcar E3: - ADMAC supports from channels 0 to 15. - ADMAC cannot be used to transfer data between SCU modules and SSI0.
SSI	Send audio data to speaker.
SRC	Convert sampling rate between 32 kHz, 44.1 kHz, and 48 kHz. **[Note]**: In Rcar E3 only two SRC modules supporting multi-channels are SRC1, SRC3. In Rcar H3/M3/M3N four SRC modules supporting multi-channels are SRC0, SRC1, SRC3, and SRC4.
DVC	Volume control which can modify volume value.
CTU	Channel transfer (from 1, 2, 4, 6, 8 channel to 1 or 2 channel).
MIX	It is used for mixing (adding) streams from two to four audio stream sources into a single stream. It also support the volume control (gain level) for each input stream.

Below table show parameter of data is supported.

Table 5-3 Supported data for playback

Item	Description				
Input data format		Channel number	PCM bit-width (fix-point)		Sample rate (Hz)
			16-bit	24-bit	48000/44100/32000
		1ch	O	X	O
		2ch	O	O	O
		4ch	O	O	O
		6ch	O	O	O
		8ch	O	O	O
Output data format		Channel number	PCM bit-width (fix-point)		Sample rate (Hz)
			16-bit	24-bit	48000/44100/32000
		1ch	O	X	O
		2ch	O	O	O
Frame size	1024 bytes				

Implementation:
O: Supported.
X: Not supported.

User can use ADSP Smoketest application to play the stream.
ADSP Smoketest program usage guideline:

```
#./adsp-omx-launch -card 0,4 -i <name> –w <value> -o <name> -rdr <name>]
```

Explain:

`-i <name>:`	Input file (.pcm or .wav).
`-card 0,4:`	Select rcar-sound card is to open codec.
`–w <value>:`	PCM Bit per sample (16/24).
`–o <name>:`	Output device (renderer).
`–rdr <name>:`	Renderer configuration file.

Renderer configuration file example, parameters will be set in a row follow the order:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
```

Parameters meaning:

In_fs:	Renderer input sampling frequency (32000/44100/48000).
Out_fs:	Renderer output sampling frequency (32000/44100/48000).
Dmach1:	Renderer DMA channel 1 (ADMAC_CH00 to ADMAC_CH31/ADMACPP_CH00 to ADMACPP_CH28).
Output1:	Renderer output source 1 (SSI00/SRC0 to SRC9).
Dmach2:	Renderer DMA channel2 (ADMAC_CH00 to ADMAC_CH31/ADMACPP_CH00 to ADMACPP_CH28).
Output2:	Renderer output source 2 (SSI00/NONCONFIG).
Volume:	Renderer volume gain (gain from 0 to 8. If DVC module is not used, this value must be -1)
Framesize:	Renderer frame size (1024).
In_ch:	Number of Renderer input channels (1/2/4/6/8).
Out_ch:	Number of Renderer output channels (1/2).
Mix_ctrl:	Renderer Mix enable control (0/1) (0 is disable MIX function, 1 is enable MIX function).

5.1.1. Playback flow 1: Normal

In this flow, data without data conversion is transfer to speaker.

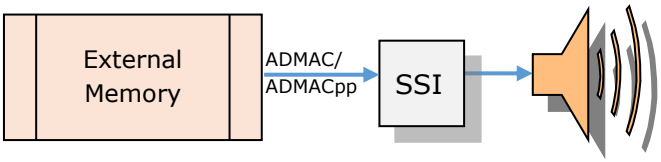

Figure 5-2 Data path for payback normal

- Example use smoke test:

```
#./adsp-omx-launch -card 0,4 -i thetest_FULL_s_32000_16.pcm -w 16 -o renderer
-rdr rdrconfig_ADMACPP.txt
```

Content of rdrconfig_ADMACPP.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
32000 32000 ADMACPP_CH15 SSI00 ADMACPP_CH00 NONCONFIG -1 1024 2 2 0
```

5.1.2. Playback flow 2: Sampling Rate Conversion

In this flow, data is converted from input sample rate to output sample rate, then transferred to speaker.

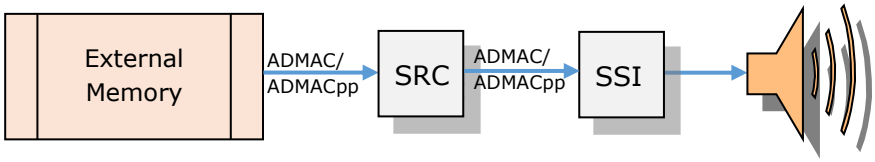

Figure 5-3 Data path for playback with sampling rate conversion in R-Car H3/M3/M3N

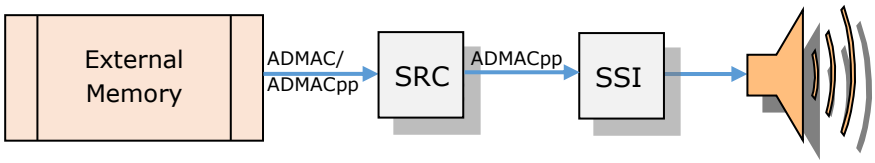

Figure 5-4 Data path for playback with sampling rate conversion in R-Car E3

- Range value:

  In_fs: 32000/44100/48000
  Out_fs: 32000/44100/48000
  Output1: SRC0-SRC9
  [Note]:
  In R-Car E3 two SRC modules supporting multi-channels are SRC1, SRC3.
  In R-Car H3/M3/M3N four SRC modules supporting multi-channels are SRC0, SRC1, SRC3, SRC4.

- Example use smoke test:

```
#./adsp-omx-launch -card 0,4 -i thetest_FULL_s_32000_16.pcm -w 16 -o renderer
-rdr rdrconfig_ADMACPP.txt
```

Content of rdrconfig_ADMACPP.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
32000 48000 ADMACPP_CH15 SRC0 ADMACPP_CH00 SSI00 -1 1024 2 2 0
```

5.1.3. Playback flow 3: Channel Transfer

In this flow, data is converted from input channel to output channel, then transferred to speaker.

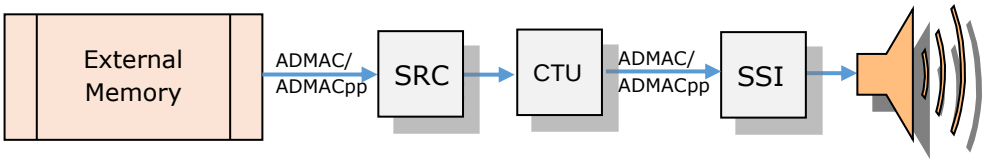

Figure 5-5 Data path for playback with channel transfer in R-Car H3/M3/M3N

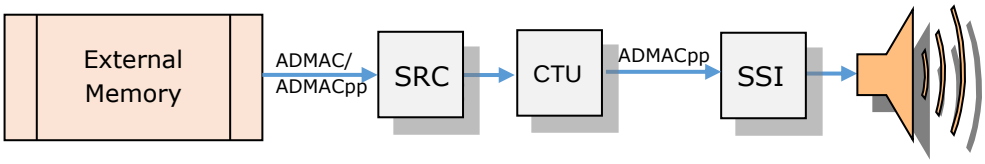

Figure 5-6 Data path for playback with channel transfer in R-Car E3

- Range value:

```
In_ch: 1, 2, 4, 6, 8
Out_ch: 1, 2
Ouput1: SRC0-SRC9
[Note]:
In R-Car E3 two SRC modules supporting multi-channels are SRC1, SRC3.
In R-Car H3/M3/M3N four SRC modules supporting multi-channels are SRC0, SRC1, SRC3, SRC4.
```

- Example use smoke test:

```
#./adsp-omx-launch -card 0,4 -i thetest_FULL_8ch_32000_16.pcm -w 16 -o renderer -rdr rdrconfig_ADMACPP.txt
```

Content of rdrconfig_ADMACPP.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
32000 32000 ADMACPP_CH15 SRC1 ADMACPP_CH00 SSI00 -1 1024 8 2 0
```

5.1.4. Playback flow 4: Volume Control

In this flow, data is changed volume, then transferred to speaker.

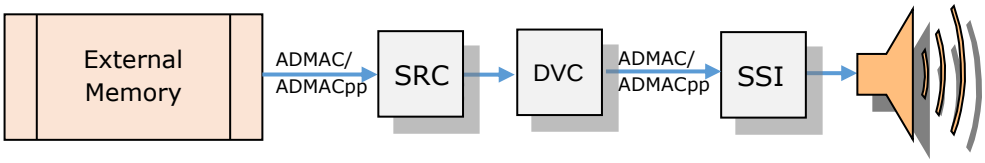

Figure 5-7 Data path for playback with volume control in R-Car H3/M3/M3N

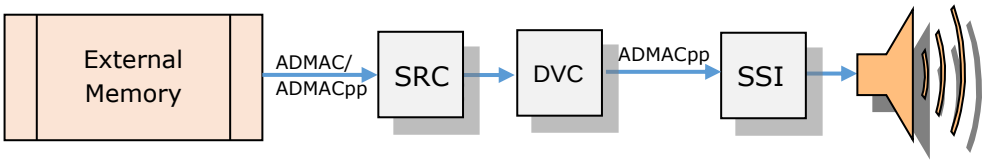

Figure 5-8 Data path for playback with volume control in R-Car E3

- Range value:

  `Volume: 0 – 8 (step 0.5).`

  `[Note]:` `If DVC module is not used, this value must be -1.`

- Example use smoke test:

```
#./adsp-omx-launch -card 0,4 -i thetest_FULL_s_32000_16.pcm -w 16 -o renderer
-rdr rdrconfig_ADMACPP.txt
```

  Content of rdrconfig_ADMACPP.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
32000 32000 ADMACPP_CH15 SRC0 ADMACPP_CH00 SSI00 1.5 1024 2 2 0
```

Beside that, stream can update volume during playback (DVC module must be enable – value of `Volume` is not -1) by pressing “u” to set volume up and “d” to set volume down.

5.1.5. Playback flow 5: MIX function

When mixing is used, the second stream and later must be consistent output sample rate, output channel and PCM width with the first one. We can convert sample rate, number of channel by SRC, CTU module.

Below table shows the maximum stream can play in MIX mode for R-Car E3/M3/M3N/H3
Table 5-4 Number of stream can play in MIX function

Number of stream	E3	M3	M3N	H3
2 streams	O	O	O	O
3 streams	O	O	O	O
4 streams	O	O	O	O

The order to playback stream when use Smoketest application is from the longest to the shortest duration stream.

- Range value:

```
Dmach1, Dmach2: ADMAC_CH00 to ADMAC_CH31/ADMACPP_CH00 to ADMACPP_CH28
Mix_ctrl: 0 or 1(0 is disable MIX function,
                 1 is enable MIX function)
[Note]:
If Dmach1 is ADMACPP:
    - Input sample rate (In_fs) of multi-channel stream must be same.
    - 16-bit stream: the total number of input channel (In_ch) is
      maximum 10:
    - 24-bit stream: the total number of input channel (In_ch) is
      maximum 6.
If Dmach1 is ADMAC:
     - 16-bit stream: the total number of input channel (In_ch) is
       maximum 16:
     - 24-bit stream: the total number of input channel (In_ch) is
       maximum 8:
```

[Note]
For the case of 24-bit streams when Equalizers are involved, the maximum of streams to be mixed is 2 due to memory limitation.

5.1.5.1. MIX function flow 1: MIX 2 streams into 1 stream

In this flow, 2 stream is mixed into 1 stream. If the second stream is not consistent with the first one (sample rate or channel), stream must setup sample rate or channel to be the same as the first, then transferred to speaker.

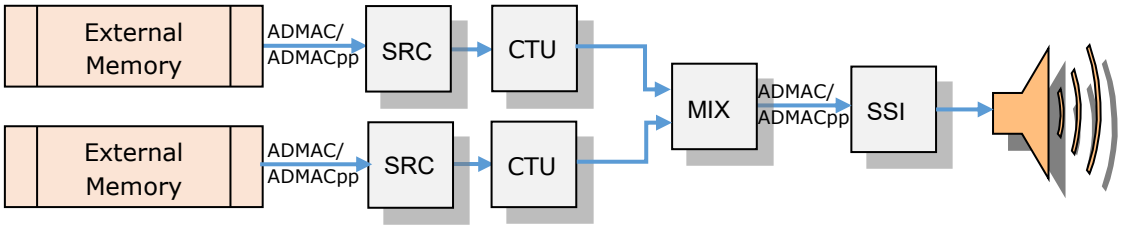

Figure 5-9 Data path for playback MIX 2 streams into 1 stream in R-Car H3/M3/M3N

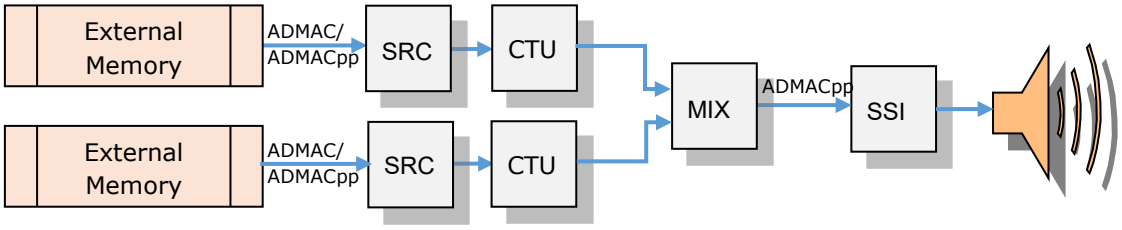

Figure 5-10 Data path for playback MIX 2 streams into 1 stream in R-Car E3

- Example use smoke test:

```
#./adsp-omx-launch -card 0,4 -i thetest_FULL_4ch_32000_16.pcm -w 16 -o renderer
-rdr rdrconfig_ADMAC_1.txt
#./adsp-omx-launch -card 0,4 -i thetest_FULL_6ch_32000_16.pcm -w 16 -o renderer
-rdr rdrconfig_ADMAC_2.txt
```

Content of rdrconfig_ADMAC_1.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
32000 32000 ADMAC_CH05 SRC3 ADMACPP_CH00 SSI00 -1 1024 4 2 1
```

Content of rdrconfig_ADMAC_2.txt:

```
 # In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
 32000 32000 ADMAC_CH13 SRC1 ADMACPP_CH10 SSI00 -1 1024 6 2 1
```

5.1.5.2. MIX function flow 2: MIX 3 streams into 1 stream

In this flow, 3 stream is mixed into 1 stream. If the later stream is not consistent with the first one (sample rate or channel) stream must setup sample rate or channel to be the same as the first, then transferred to speaker.

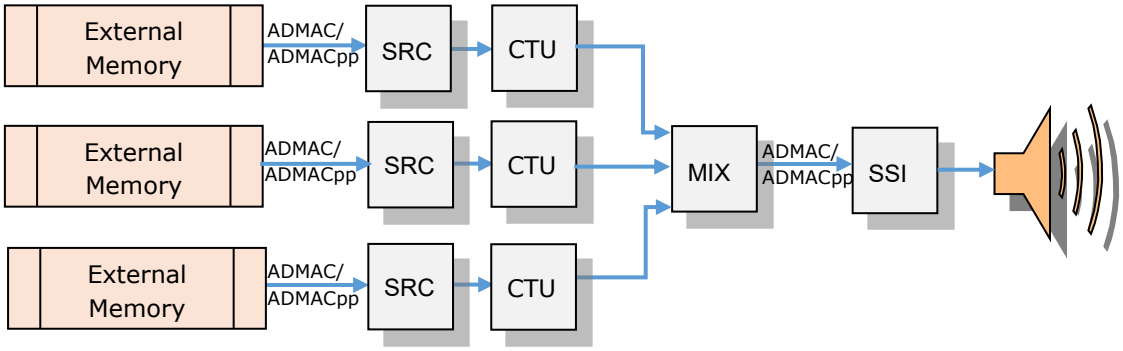

Figure 5-11 Data path for playback MIX 3 streams into 1 stream in R-Car H3/M3/M3N

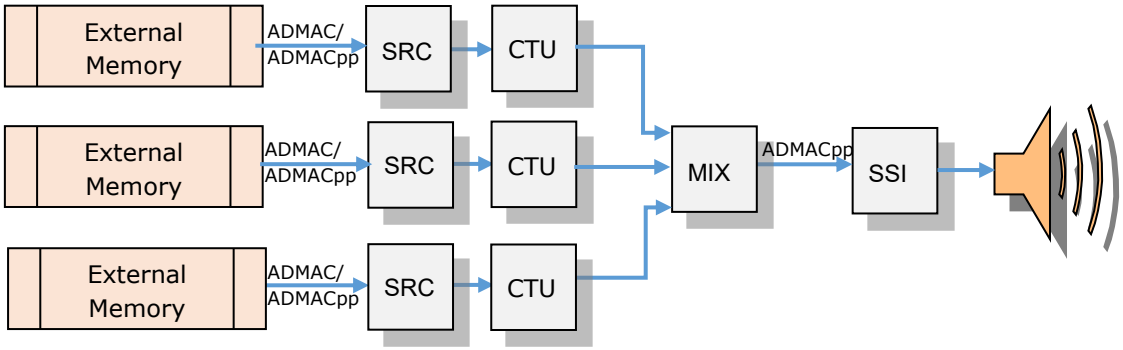

Figure 5-12 Data path for playback MIX 3 streams into 1 stream in R-Car E3

- Example use smoke test:

```
#./adsp-omx-launch -card 0,4 -i thetest_FULL_4ch_32000_16.pcm -w 16 -o renderer -rdr
rdrconfig_ADMAC_1.txt
#./adsp-omx-launch -card 0,4 -i thetest_FULL_4ch_32000_16.pcm -w 16 -o renderer -rdr
rdrconfig_ADMAC_2.txt
#./adsp-omx-launch -card 0,4 -i thetest_FULL_s_32000_16.pcm -w 16 -o renderer -rdr
rdrconfig_ADMAC_3.txt
```

Content of rdrconfig_ADMAC_1.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
32000 32000 ADMAC_CH05 SRC3 ADMACPP_CH00 SSI00 -1 1024 4 2 1
```

Content of rdrconfig_ADMAC_2.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
32000 32000 ADMAC_CH13 SRC1 ADMACPP_CH10 SSI00 0.5 1024 4 2 1
```

Content of rdrconfig_ADMAC_3.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
32000 32000 ADMAC_CH03 SRC0 ADMACPP_CH04 SSI00 2 1024 2 2 1
```

5.1.5.3. MIX function flow 3: MIX 4 streams into 1 stream

In this flow, 4 stream is mixed into 1 stream, if the later stream is not consistent with the first one (sample rate or channel) stream must setup sample rate or channel to be the same as the first, then transferred to speaker.

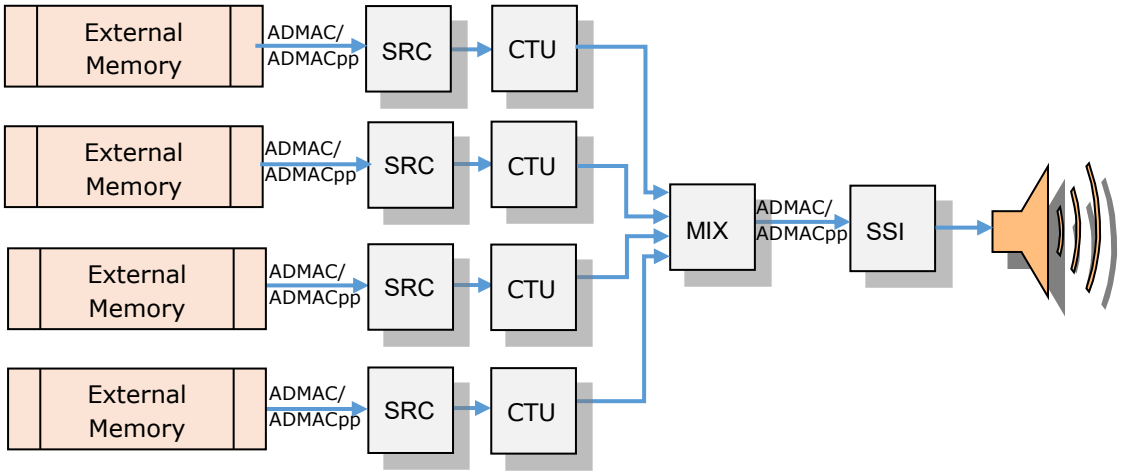

Figure 5-13 Data path for playback MIX 4 streams into 1 stream in R-Car H3/M3/M3N

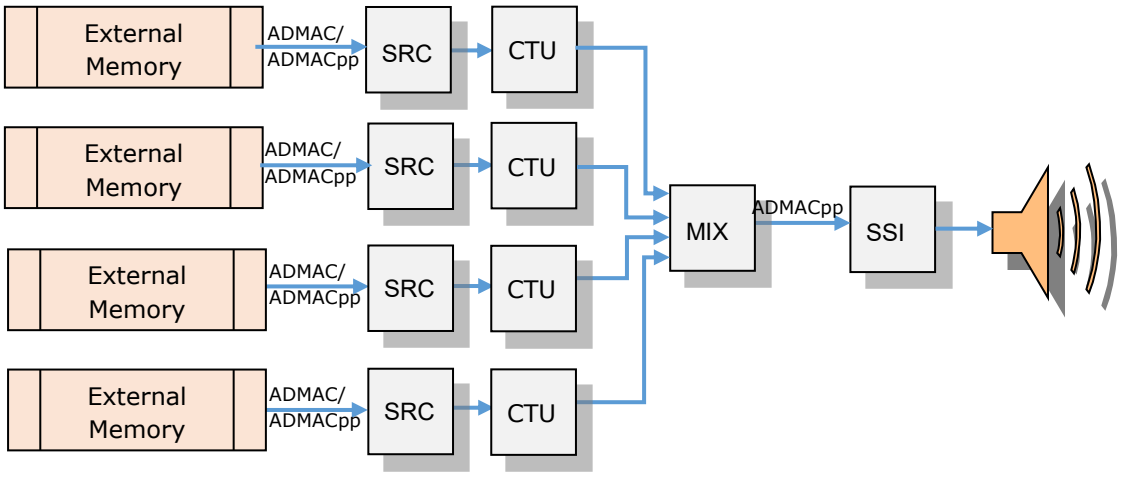

Figure 5-14 Data path for playback MIX 4 streams into 1 stream in R-Car E3

- Example use smoke test:

```
#./adsp-omx-launch -card 0,4 -i thetest_FULL_s_32000_16.pcm -w 16 -o renderer
-rdr rdrconfig_ADMAC_1.txt
#./adsp-omx-launch -card 0,4 -i thetest_FULL_s_32000_16.pcm -w 16 -o renderer
-rdr rdrconfig_ADMAC_2.txt
#./adsp-omx-launch -card 0,4 -i thetest_FULL_4ch_32000_16.pcm -w 16 -o renderer
-rdr rdrconfig_ADMAC_3.txt
#./adsp-omx-launch -card 0,4 -i thetest_FULL_6ch_32000_16.pcm -w 16 -o renderer
-rdr rdrconfig_ADMAC_4.txt
```

Content of rdrconfig_ADMAC_1.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
32000 32000 ADMAC_CH05 SRC0 ADMACPP_CH00 SSI00 -1 1024 2 2 1
```

Content of rdrconfig_ADMAC_2.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
32000 32000 ADMAC_CH13 SRC5 ADMACPP_CH10 SSI00 -1 1024 2 2 1
```

Content of rdrconfig_ADMAC_3.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
32000 32000 ADMAC_CH03 SRC1 ADMACPP_CH04 SSI00 -1 1024 4 2 1
```

Content of rdrconfig_ADMAC_4.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
32000 32000 ADMAC_CH06 SRC3 ADMACPP_CH6 SSI00 -1 1024 6 2 1
```

5.2. Standard Use Case: Record

Figure 5-15, shows an example of the audio data path are transferred which use capture function.

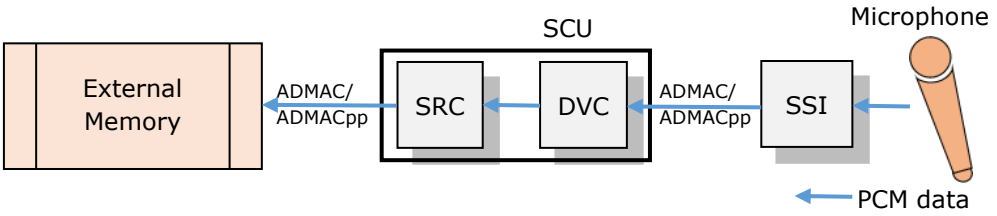

Figure 5-15 Audio data path for capture function

Below table shows supported features and its usage for OMX capture.
Table 5-5 Supported features for OMX Capture Interface

Name	Usage
Audio-DMAC/ Audio-DMACpp	Transfer data between two audio modules; memory and audio module. **[note]**: About R-Car E3: - ADMAC supports from channels 0 to 15. - ADMAC cannot be used to transfer data between SSI10 modules and SCU.
SSI	Receive audio data from microphone.
SRC	Convert sampling rate between 32 kHz, 44.1 kHz, and 48 kHz.
DVC	Volume control which can modify volume value.

Below table Show parameter of data is supported.
Table 5-6 Supported data for record

Item	Description			
Input data format	Channel number	PCM bit-width (fix-point)		Sample rate (Hz)
		16-bit	24-bit	48000/44100/32000
	1ch	O	X	O
	2ch	O	O	O
Output data format	Channel number	PCM bit-width (fix-point)		Sample rate (Hz)
		16-bit	24-bit	48000/44100/32000
	1ch	O	X	O
	2ch	O	O	O
Frame size	1024 bytes			

Implementation:
O: Supported.
X: Not supported.

User can use ADSP Smoketest application to record the stream.
ADSP Smoketest program usage guideline:

```
#./adsp-omx-launch -card 0,4 -i <name>  -c <value> -w <value> -l <value>
-o <name> -cap <name>]
```

Explain:

`-i <name>:`	Input device (capture)
`-card 0,4:`	Select rcar-sound card is to open codec
`-c <value>:`	PCM channel number for Capture (1/2)
`-w <value>:`	PCM Bit per sample (16/24)
`-l <value>:`	Recording time (second)
`-o <name>:`	Output file (.pcm)
`-cap <name>:`	Capture configuration file

Capture configuration file example, parameters will be set in a row follow the order:

```
# In_fs Out_fs Dmach1 Input1 Dmach2 Input2 Volume Framesize
```

Parameters meaning:

`In_fs:`	Capture input sampling frequency (32000/44100/48000)
`Out_fs:`	Capture output sampling frequency (32000/44100/48000)
`Dmach1:`	Capture DMA channel 1 (ADMAC_CH00 to ADMAC_CH31/ADMACPP_CH00 to ADMACPP_CH28)
`Input1:`	Capture input source 1 (SSI10/SRC0 to SRC9)
`Dmach2:`	Capture DMA channel2 (ADMAC_CH00 to ADMAC_CH31/ADMACPP_CH00 to ADMACPP_CH28)
`Input2:`	Capture input source 2 (SSI10/NONCONFIG)
`Volume:`	Capture volume gain (gain from 0 to 8. If DVC module is not used, this value must be -1)
`Framesize:`	Capture frame size (1024)

5.2.1. Capture flow 1: Normal

In this flow, data without data conversion is transferred from microphone to external memory

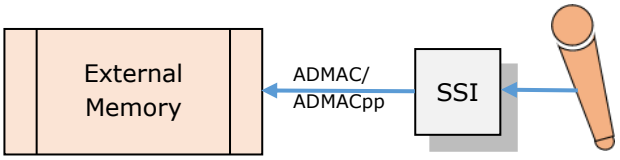

Figure 5-16 Data path for case record normal

- Example use smoke test:

```
#./adsp-omx-launch -card 0,4 -i capture -c 2 -w 16 -l 15 -o out.pcm -cap
capconfig_ADMACPP.txt
```

Content of capconfig_ADMACPP.txt:

```
#In_fs Out_fs Dmach1 Input1 Dmach2 Input2 Volume Framesize
48000 48000 ADMACPP_CH15 SSI10 ADMACPP_CH00 NONCONFIG -1 1024
```

5.2.2. Capture flow 2: Sampling Rate Conversion

In this flow, data, which is converted from input sample rate to output sample rate, is transferred from microphone to external memory

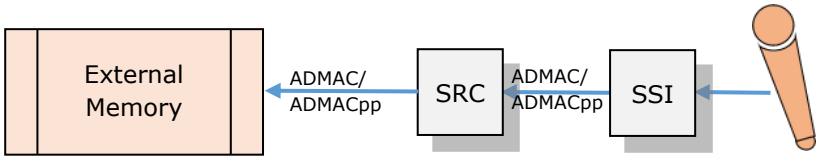

Figure 5-17 Data path for record sampling rate conversion in R-Car H3/M3/M3N

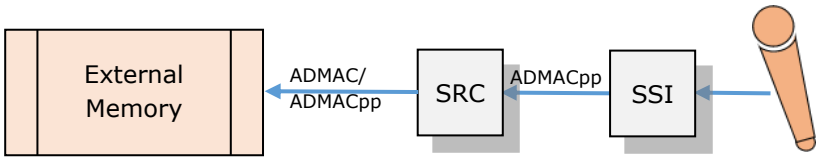

Figure 5-18 Data path for record sampling rate conversion in R-Car E3

- Range value.

```
In_fs: 32000/44100/48000
Out_fs: 32000/44100/48000
Input1: SRC0-SRC9
```

- Example use smoke test:

```
#./adsp-omx-launch -card 0,4 -i capture -c 2 -w 16 -l 15 -o out.pcm -cap
capconfig_ADMACPP.txt
```

Content of capconfig_ADMACPP.txt:

```
#In_fs Out_fs Dmach1 Input1 Dmach2 Input2 Volume Framesize
32000 48000 ADMACPP_CH15 SRC0 ADMACPP_CH00 SSI10 -1 1024
```

5.2.3. Capture flow 3: Volume Control

In this flow, data, which is update volume, is transferred from microphone to external memory

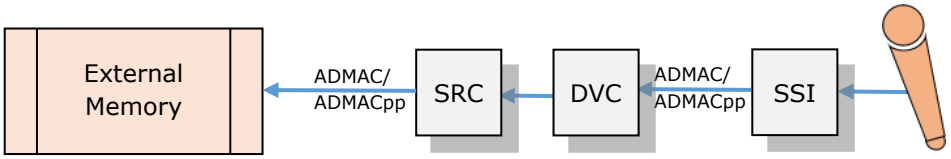

Figure 5-19 Data path for record volume control in R-Car H3/M3/M3N

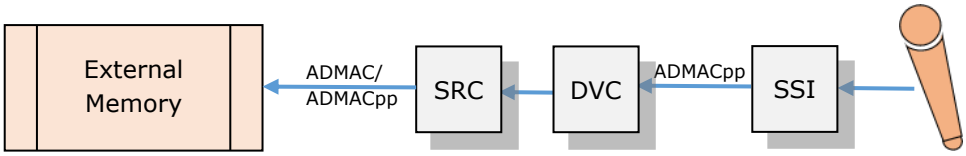

Figure 5-20 Data path for record volume control in R-Car E3

- Range value.

```
Volume: 0 – 8 (step 0.5).
[Note]: If DVC module is not used, this value must be -1.
```

- Example use smoke test:

```
#./adsp-omx-launch -card 0,4 -i capture -c 2 -w 16 -l 15 -o out.pcm -cap
capconfig_ADMACPP.txt
```

Content of capconfig_ADMACPP.txt:

```
#In_fs Out_fs Dmach1 Input1 Dmach2 Input2 Volume Framesize
32000 48000 ADMACPP_CH15 SRC0 ADMACPP_CH00 SSI10 2 1024
```

Record processing can stop by pressing “s”.
Besides that, stream can update volume during record (DVC module must be enable – value of `Volume` is not -1) by pressing “u” to set volume up and “d” to set volume down.

5.3. Standard Use Case: Equalizer

Equalizer plugin does not support setting in runtime.
User can use ADSP Smoketest application to equalize stream:
ADSP Smoketest program usage guideline:

```
#./adsp-omx-launch -i <name> -w <value> -c <value> -o <name>
-eq <value> -eqzfs <value> -eqz <name>]
```

Explain:

`-i <name>:`	Input file (.pcm or .wav).
`-o <name>:`	Output file (.pcm).
`-w <value>:`	PCM Bit per sample (16/24).
`-c <value>:`	PCM channel number for Equalizer (1/2).
`-eq <value>:`	Enable/Disable equalizer (on/off) (default: off).
`-eqzfs <value>:`	Equalizer PCM sampling frequency (32000/44100/48000).
`-eqz <name>:`	Equalizer configuration file.

Equalizer configuration file example:

- Parametric configuration file
  - Start 1st line with "Parametric" identity.
  - 2nd line to 10th line will contain information structure like below:

    `Type Fc Bandwidth Gain BaseGain`

o `Type:`	Filter type (P: Peak, R: Treble, B: Bass, T: Through)
o `Fc:`	Frequency center (Peak|Through: 20-20000, Treble: 5000-11000, Bass: 50-500)
o `Bandwidth:`	0.5 - 15
o `Gain:`	-15.0 - 15.0 (dB)
o `BaseGain:`	-10.0 - 10.0 (dB)

- Graphic configuration file
  - Start 1st line with "Graphic" identity.
  - 2nd line to 6th line with Gain (-10.0 - 10.0) dB value.

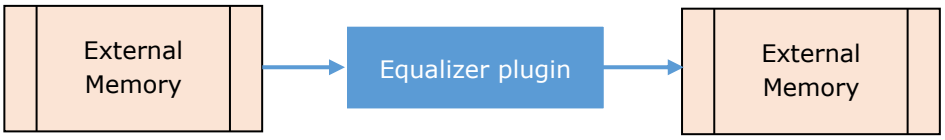

Figure 5-21 Data path for equalizer

- Example use smoke test:

```
#./adsp-omx-launch -i thetest_FULL_s_32000_16.pcm -w 16 -c 2
-o out.pcm -eq on -eqzfs 32000 -eqz parametric_config.txt
```

Content of parametric_config.txt:

```
Parametric
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
```

5.4. Standard Use Case: Route

User can use ADSP Smoketest application to route between Capture, Renderer and Equalizer

5.4.1. Route flow 1: Capture – Equalizer

In this flow, data is transferred from microphone to Capture and Equalizer plugin and stored in external memory.

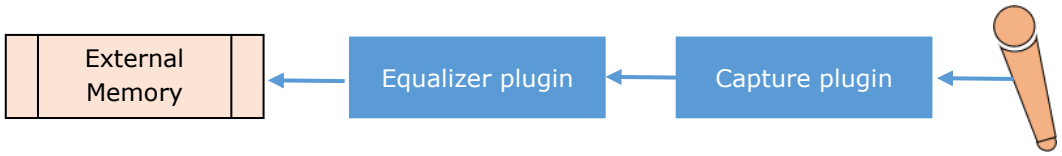

Figure 5-22 Data path for route Capture – Equalizer

- Example use smoke test:

```
#./adsp-omx-launch -o out.pcm -card 0,4 -i capture -w 24 -c 2 -eqzfs 48000 -eqz
parametric_config.txt -eq on -cap capconfig_ADMACPP.txt
```

Content of capconfig_ADMACPP.txt:

```
#In_fs Out_fs Dmach1 Input1 Dmach2 Input2 Volume Framesize
32000 48000 ADMACPP_CH15 SRC6 ADMACPP_CH04 SSI10 0.5 1024
```

Content of parametric_config.txt:

```
Parametric
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
```

5.4.2. Route flow 2: Equalizer – Renderer

In this flow, data is transferred from external memory to Equalizer and Renderer plugin and to speaker.

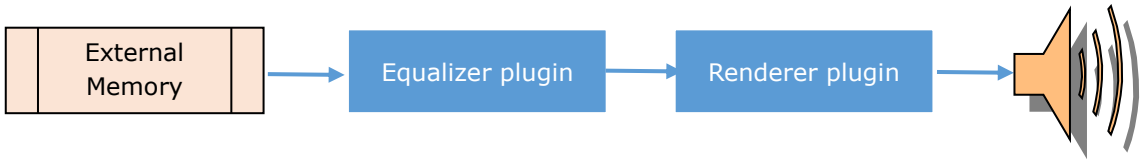

Figure 5-23 Data path for route Equalizer – Renderer

- Example use smoke test:

```
#./adsp-omx-launch -card 0,4 -o renderer -i thetest_FULL_s_48000_24.pcm -w 24 -c 2 -
eqzfs 48000 -eqz graphic_config.txt -eq on -rdr rdrconfig_ADMACPP.txt
```

Content of rdrconfig_ADMACPP.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
48000 32000 ADMACPP_CH25 SRC3 ADMACPP_CH00 SSI00 0.5 1024 2 2 0
```

Content of graphic_config.txt:

```
Graphic
-10.0
-10.0
-10.0
-10.0
-10.0
```

5.4.3. Route flow 3: Capture – Equalizer – Renderer

In this flow, data is transferred from microphone to Capture and Equalizer plugin and then to Renderer plugin and to speaker.

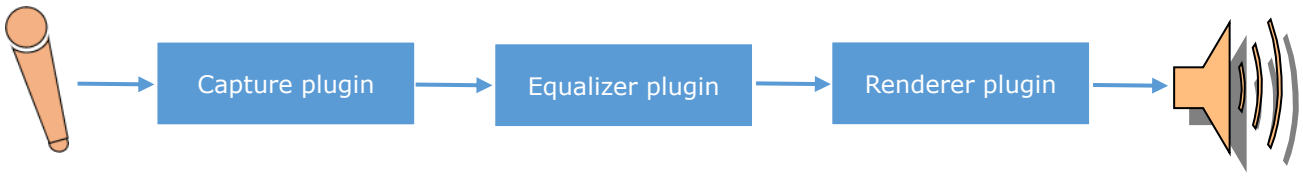

Figure 5-24 Data path for route Capture – Equalizer – Renderer

[Note] The input sample rate of Capture and output sample rate of Renderer should be specified to the same value.

- Example use smoke test:

```
#./adsp-omx-launch -card 0,4 -o renderer -i capture -w 24 -c 2 -eqzfs 48000 -eqz
parametric_config.txt -eq on -rdr rdrconfig_ADMACPP.txt -cap capconfig_ADMACPP.txt
```

Content of capconfig_ADMACPP.txt:

```
#In_fs Out_fs Dmach1 Input1 Dmach2 Input2 Volume Framesize
32000 48000 ADMACPP_CH15 SRC6 ADMACPP_CH04 SSI10 0.5 1024
```

Content of rdrconfig_ADMACPP.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
48000 32000 ADMACPP_CH25 SRC3 ADMACPP_CH00 SSI00 0.5 1024 2 2 0
```

Content of parametric_config.txt:

```
Parametric
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
```

5.4.4. Route flow 4: Capture – Renderer

In this flow, data is transferred from microphone to Capture, Renderer plugin and to speaker.

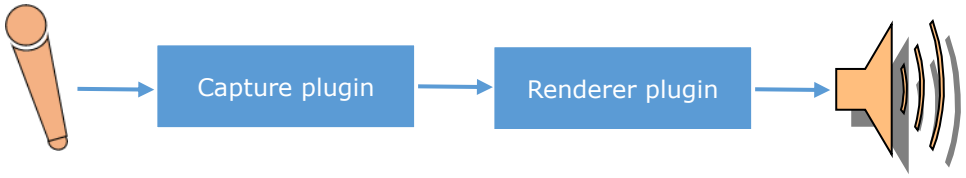

Figure 5-25 Data path for route Equalizer – Renderer

[Note] The input sample rate of Capture and output sample rate of Renderer should be specified to the same value.

- Example use smoke test:

```
#./adsp-omx-launch -card 0,4 -o renderer -i capture -w 24 -c 2 -rdr
rdrconfig_ADMACPP.txt -cap capconfig_ADMACPP.txt
```

Content of capconfig_ADMACPP.txt:

```
#In_fs Out_fs Dmach1 Input1 Dmach2 Input2 Volume Framesize
32000 48000 ADMACPP_CH15 SRC6 ADMACPP_CH04 SSI10 0.5 1024
```

Content of rdrconfig_ADMACPP.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
48000 32000 ADMACPP_CH25 SRC3 ADMACPP_CH00 SSI00 0.5 1024 2 2 0
```

Content of parametric_config.txt:

```
Parametric
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
T 15000 0.707 1.0 1.0
```

5.4.5. Route flow 5: Equalizer – Renderer use MIX function

In this flow, streams are transferred from external memory to Equalizer and Renderer plugin and to speaker. Below is a simple image when using mix function with Equalizer routing. For detail configuration, please refer to 5.1.5.3

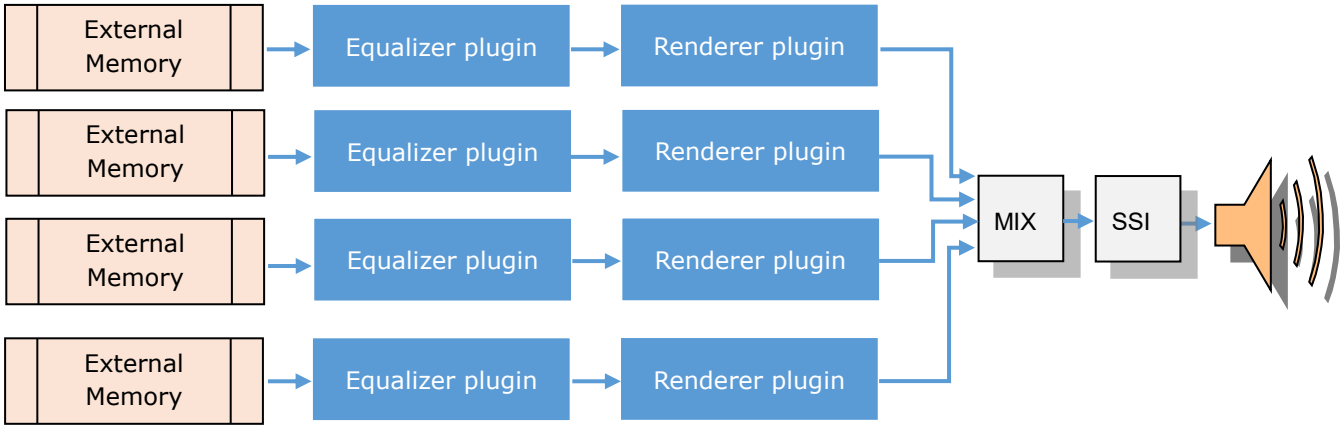

Figure 5-26 Data path for route Equalizer – Renderer use MIX function

- Range:
  If using 24 bit/stereo, maximum 3 mixing streams are supported.

- Example use smoke test:

```
#./adsp-omx-launch -card 0,4 -o renderer -i thetest_FULL_s_32000_16.pcm -w 16 -c 2 -
eqzfs 32000 -eqz graphic_config.txt -eq on -rdr rdrconfig_ADMACPP1.txt
#./adsp-omx-launch -card 0,4 -o renderer -i thetest_FULL_s_48000_16.pcm -w 16 -c 2 -
eqzfs 48000 -eqz graphic_config.txt -eq on -rdr rdrconfig_ADMACPP2.txt
#./adsp-omx-launch -card 0,4 -o renderer -i thetest_FULL_s_44100_16.pcm -w 16 -c 2 -
eqzfs 44100 -eqz graphic_config.txt -eq on -rdr rdrconfig_ADMACPP3.txt
#./adsp-omx-launch -card 0,4 -o renderer -i thetest_FULL_s_48000_16.pcm -w 16 -c 2 -
eqzfs 48000 -eqz graphic_config.txt -eq on -rdr rdrconfig_ADMACPP4.txt
```

Content of rdrconfig_ADMACPP1.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
32000 32000 ADMACPP_CH00 SRC0 ADMACPP_CH02 SSI00 0.5 1024 2 2 1
```

Content of rdrconfig_ADMACPP2.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
48000 32000 ADMACPP_CH05 SRC1 ADMACPP_CH04 SSI00 1 1024 2 2 1
```

Content of rdrconfig_ADMACPP3.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
44100 32000 ADMACPP_CH10 SRC3 ADMACPP_CH06 SSI00 -1 1024 2 2 1
```

Content of rdrconfig_ADMACPP4.txt:

```
# In_fs Out_fs Dmach1 Output1 Dmach2 Output2 Volume Framesize In_ch Out_ch Mix_ctrl
48000 32000 ADMACPP_CH13 SRC5 ADMACPP_CH08 SSI00 2 1024 2 2 1
```

Content of graphic_config.txt:

```
Graphic
-10.0
-10.0
-10.0
-10.0
-10.0
```

5.5. Standard Use Case: TDM

Figure 5-27, shows an example of the audio data path are transferred which use TDM renderer function.

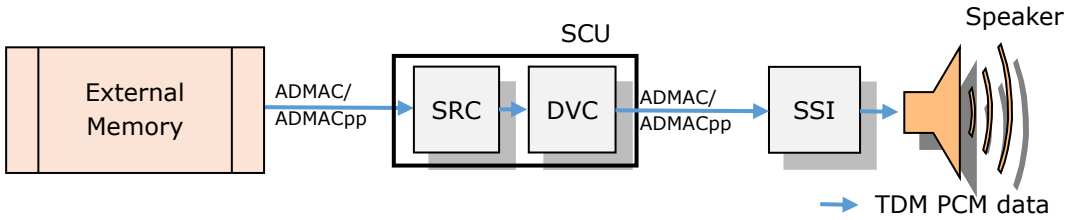

Figure 5-27 Audio data path for TDM renderer function

Figure 5-28, shows an example of the audio data path are transferred which use TDM capture function.

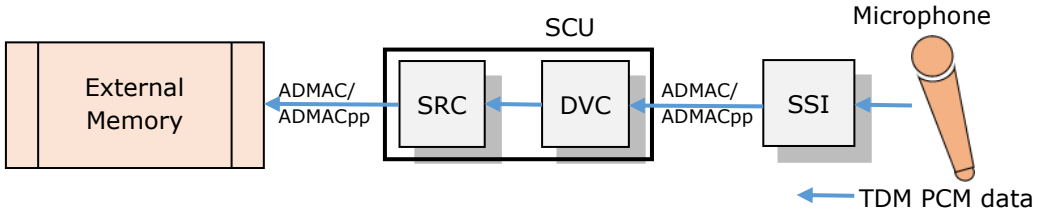

Figure 5-28 Audio data path for TDM capture function

Table 5-7 Supported features for OMX TDM Interface

Name	Usage
Audio-DMAC/ Audio-DMACpp	Transfer data between two audio modules; memory and audio module.
SSI	Send audio data to speaker (SSI30) Receive audio data from microphone (SSI40)
SRC	Convert sampling rate between 32 kHz, 44.1 kHz and 48 kHz.
DVC	Volume control Non-supported setting in runtime

Two below table show parameter of data is supported.

Table 5-8 Supported data for TDM Renderer

Item	Description				
Input data format	Channel number		PCM bit-width (fix-point)		Sample rate (Hz)
			16-bit	24-bit	48000/44100/32000
	6ch	3 x 2ch	O	O	O
		1 x 6ch	O	O	O
	8ch	4 x 2ch	O	O	O
		1 x 8ch	O	O	O
Output data format	Channel number		PCM bit-width (fix-point)		Sample rate (Hz)
			16-bit	24-bit	44100/48000
	6ch	3 x 2ch	O	O	O
		1 x 6ch	O	O	O
	8ch	4 x 2ch	O	O	O
		1 x 8ch	O	O	O
Frame size	1024 bytes				

Table 5-9 Supported data for TDM Capture

Item	Description				
Input data format	Channel number		PCM bit-width (fix-point)		Sample rate (Hz)
			16-bit	24-bit	44100/48000
	6ch	3 x 2ch	O	O	O
		1 x 6ch	O	O	O
	8ch	4 x 2ch	O	O	O
		1 x 8ch	O	O	O
Output data format	Channel number		PCM bit-width (fix-point)		Sample rate (Hz)
			16-bit	24-bit	48000/44100/48000
	6ch	3 x 2ch	O	O	O
		1 x 6ch	O	O	O
	8ch	4 x 2ch	O	O	O
		1 x 8ch	O	O	O
Frame size	1024 bytes				

Implementation:

O: Supported.

X: Not supported.

User can use ADSP Smoketest application to run TDM Capture.
ADSP Smoketest program usage guideline:

```
#./adsp-omx-tdm-launch -card 0,1 -i ssi1 –w <value> -chmod <value> -o1 <name> -o2
<name> -o3 <name> -o4 <name> -l <value> [-<command> <value>]
```

Explain:

`-i ssi1:` Use TDM Capture.

`-card 0,1:` Select rcar-sound card is to open codec.

`–w <value>:` PCM Bit per sample (16/24).

`–chmod <value>:` Channel mode.

Value	Description
0	4 streams stereo
1	1 stream 8 channels
3	3 streams stereo
4	1 stream 6 channels

`–o1 <name>:` 1st output stream (.pcm).

`–o2 <name>:` 2nd output stream (.pcm).

`–o3 <name>:` 3rd output stream (.pcm).

`–o4 <name>:` 4th output stream (.pcm).

`–l <value>:` Recording time (second).

`–capfs <value>:` Input sampling frequency.
(44100/48000, set '0' to disable SRC module).

`–capoutfs <value>:` Output sampling frequency (32000/44100/48000).

`–capdmachannel1 <value>:` 1st DMA channel.
(ADMAC_CH00 to ADMAC_CH31)/(ADMACPP_CH00 to ADMACPP_CH28).

`–capinsource1 <value>:` 1st input device.
(SSI40)/(SCU_SRCI0, SCU_SRCI1, SCU_SRCI3, SCU_SRCI4).

`–capdmachannel2 <value>:` 2nd DMA channel.
(ADMAC_CH00 to ADMAC_CH31)/(ADMACPP_CH00 to ADMACPP_CH28)

`–capinsource2 <value>:` 2nd input device (SSI40/NONCONFIG).
If `capinsource1` is 'SSI40', this value must be 'NONCONFIG'.

`–capvol <value>:` Volume gain (0-8, set 'FFFFFFFF' to disable DVC module).

`–capframe <value>:` Frame size (1024).

User can use ADSP Smoketest application to run TDM Renderer.
ADSP Smoketest program usage guideline:

```
#./adsp-omx-tdm-launch -card 0,1 -o ssi0 –w <value> -chmod <value> -o1 <name> -o2
<name> -o3 <name> -o4 <name> -l <value> [-<command> <value>]
```

Explain:

`-o ssi0:`	Use TDM Renderer.
`-card 0,1:`	Select rcar-sound card is to open codec.
`–w <value>:`	PCM Bit per sample (16/24).
`–chmod <value>:`	Channel mode.

Value	Description
0	4 streams stereo
1	1 stream 8 channels
3	3 streams stereo
4	1 stream 6 channels

`–i1 <name>:`	1st input stream (.pcm).
`–i2 <name>:`	2nd input stream (.pcm).
`–i3 <name>:`	3rd input stream (.pcm).
`–i4 <name>:`	4th input stream (.pcm).
`–rdrfs <value>:`	Input sampling frequency (32000/44100/48000).
`-rdroutfs <value>:`	Output sampling frequency. (44100/48000, set '0' to disable SRC module).
`–rdrdmachannel1 <value>:`	1st DMA channel. (ADMAC_CH00 to ADMAC_CH31)/(ADMACPP_CH00 to ADMACPP_CH28).
`–rdroutsource1 <value>:`	1st output device. (SSI30)/(SCU_SRCI0, SCU_SRCI1, SCU_SRCI3, SCU_SRCI4).
`–rdrdmachannel2 <value>:`	2nd DMA channel. (ADMAC_CH00 to ADMAC_CH31)/(ADMACPP_CH00 to ADMACPP_CH28)
`–rdroutsource2 <value>:`	2nd output device (SSI30/NONCONFIG). If `rdroutsource1` is 'SSI30', this value must be 'NONCONFIG'.
`–rdrvol <value>:`	Volume gain (0-8, set 'FFFFFFFF' to disable DVC module).
`–rdrframe <value>:`	Frame size (1024).

[Note] Do not support to convert sampling rate between 32000 and 44100 Hz, and between 48000 and 44100 Hz.

5.5.1. TDM Renderer flow 1: Normal

In this flow, TDM without data conversion data is transfer to speaker.

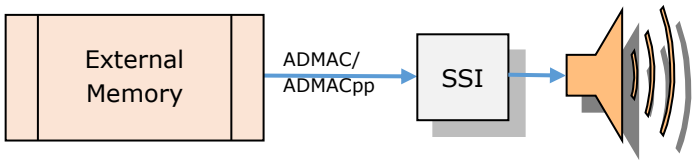

Figure 5-29 Data path for TDM renderer

- Example use smoke test:
```
#./adsp-omx-tdm-launch -card 0,1 -w 16 -chmod 3 -o ssi0 -i1 input1.pcm -i2 input2.pcm -i3 input3.pcm -rdrfs 32000 -rdrdmachannel1 ADMACPP_CH01 -rdroutsource1 SSI30
```

5.5.2. TDM Renderer flow 2: Sampling Rate Conversion

In this flow, TDM data, which is converted from input sample rate to output sample rate, is transfer to speaker.

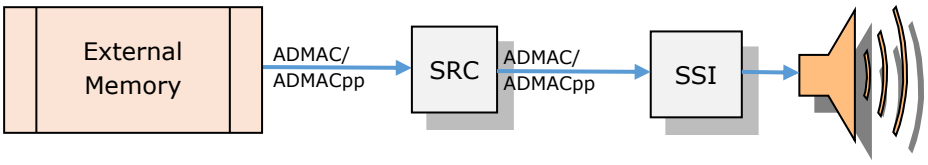

Figure 5-30 Data path for TDM renderer use sample rate conversation

- Example use smoke test:
```
#./adsp-omx-tdm-launch -card 0,1 -w 16 -chmod 0 -o ssi0 -i1 input1.pcm -i2 input2.pcm -i3 input3.pcm -i4 input4.pcm -rdrfs 32000 -rdroutfs 48000 -rdrdmachannel1 ADMACPP_CH01 -rdroutsource1 SSI30 -rdrdmachannel2 ADMACPP_CH00 -rdroutsource2 NONCONFIG -rdrvol FFFFFFFF
```

5.5.3. TDM Renderer flow 3: Volume Control

In this flow, TDM data is changed volume, then transferred to speaker.

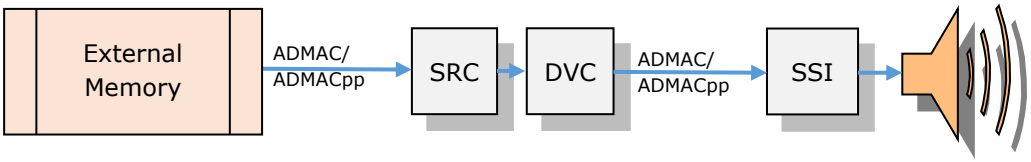

Figure 5-31 Data path for TDM renderer use volume control

- Example use smoke test:

```
#./adsp-omx-tdm-launch -card 0,1 -w 16 -chmod 0 -o ssi0 -i1 input1.pcm -i2 input2.pcm -i3 input3.pcm -i4 input4.pcm -rdrfs 32000 -rdroutfs 48000 -rdrdmachannel1 ADMACPP_CH01 -rdroutsource1 SCU_SRCI0 -rdrdmachannel2 ADMACPP_CH00 -rdroutsource2 SSI30 -rdrvol 1
```

5.5.4. TDM Capture flow 1: Normal

In this flow, data without data conversion is transferred from microphone to external memory

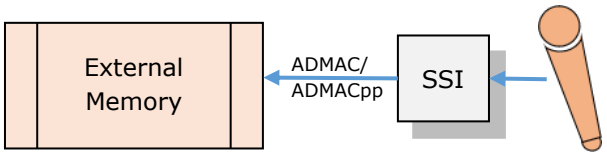

Figure 5-32 Data path for TDM Capture nomal

- Example use smoke test:

```
#./adsp-omx-tdm-launch -card 0,1 -w 16 -chmod 3 -i ssi1 -o1 out1.pcm -o2 out2.pcm -o3 out3.pcm -capoutfs 44100 -capdmachannel1 ADMACPP_CH01 -capinsource1 SSI40
```

5.5.5. TDM Capture flow 2: Sampling Rate Conversion

In this flow, data, which is converted from input sample rate to output sample rate, is transferred from microphone to external memory

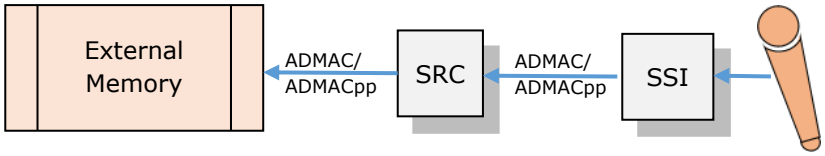

Figure 5-33 Data path for TDM capture use sampling rate conversion

- Example use smoke test:

```
#./adsp-omx-tdm-launch -card 0,1 -w 16 -chmod 0 -i ssi1 -o1 out1.pcm -o2
out2.pcm -o3 out3.pcm -o4 out4.pcm -capfs 48000 -capoutfs 44100 -
capdmachannel1 ADMACPP_CH01 -capinsource1 SCU_SRCI0 -capdmachannel2
ADMACPP_CH00 -capinsource2 SSI40 -capvol FFFFFFFF
```

5.5.6. TDM Capture flow 3: Volume Control

In this flow, data, which is changed volume, is transferred from microphone to external memory

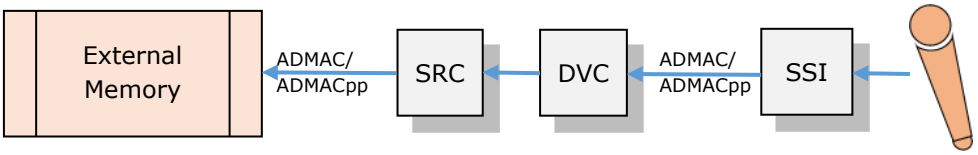

Figure 5-34 Data path for TDM capture use volume control

- Example use smoke test:

```
#./adsp-omx-tdm-launch -card 0,1 -w 16 -chmod 0 -i ssi1 -o1 out1.pcm -o2 out2.pcm -o3 out3.pcm -o4 out4.pcm -capfs 48000 -capoutfs 44100 -capdmachannel1 ADMACPP_CH01 -capinsource1 SCU_SRCI0 -capdmachannel2 ADMACPP_CH00 -capinsource2 SSI40 -capvol 1
```

Revision History	ADSP Interface for Android User's Manual

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

ADSP Interface for Android User's Manual

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

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 Interface for Android
RCG3AHIFA9001ZDP

RCG3AHIFA9001ZDPE

Renesas Electronics Corporation