RENESAS

CONFIDENTIAL

ADSP Interface for Android
RCG3AHIFA8101ZDP

User's Manual

RCG3AHIFA8101ZDPE

Rev. 0.10   May, 2018

Renesas Electronics
www.renesas.com

CONFIDENTIAL

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

How to Use This Manual

1. Purpose and Target Reader

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

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

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

2. Restrictions on the Use of this Middleware

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

3. Related Manuals

4. Technical Terms and Abbreviation

All trademarks and registered trademarks are the property of their respective owners.

- Table of Contents -

- List of Figures -

- List of Tables -

ADSP Interface for Android User's Manual

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.
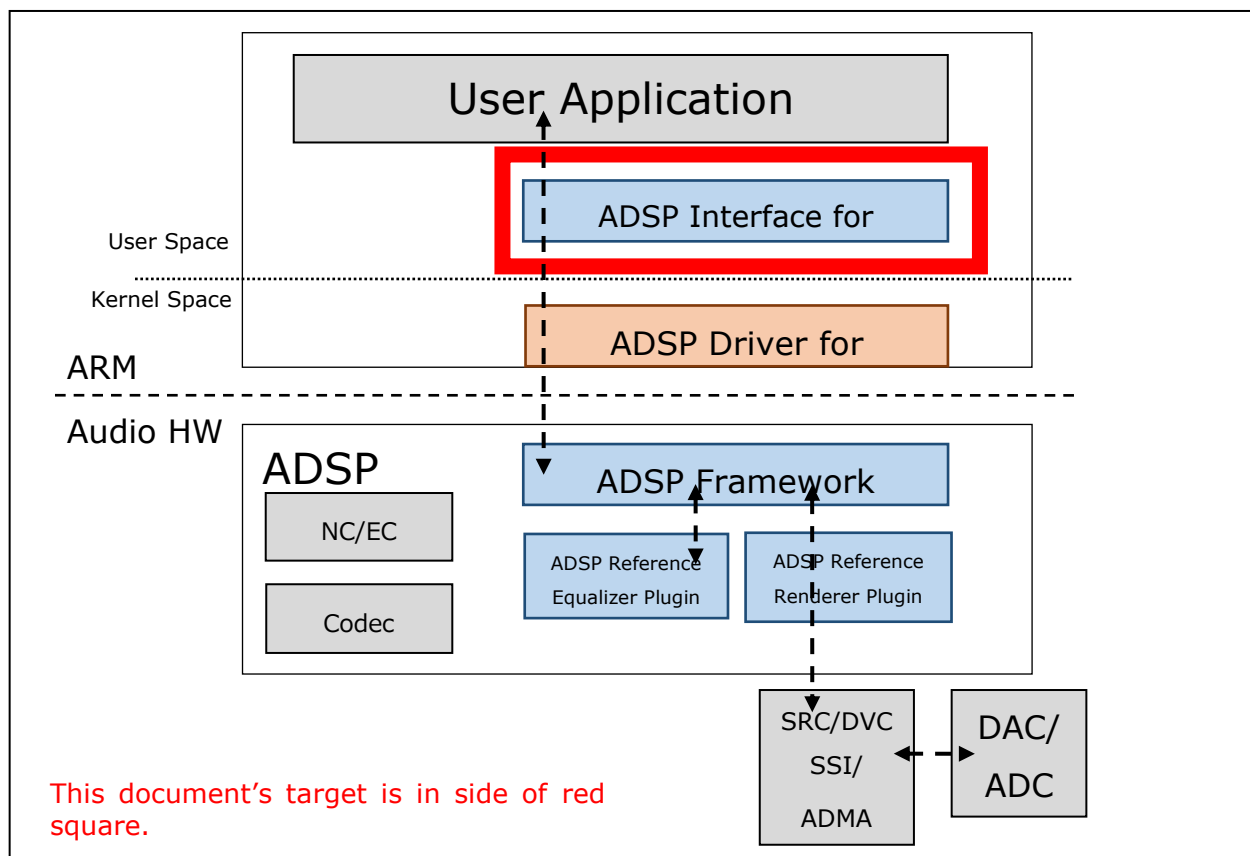
Figure 1-1 The software architecture

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

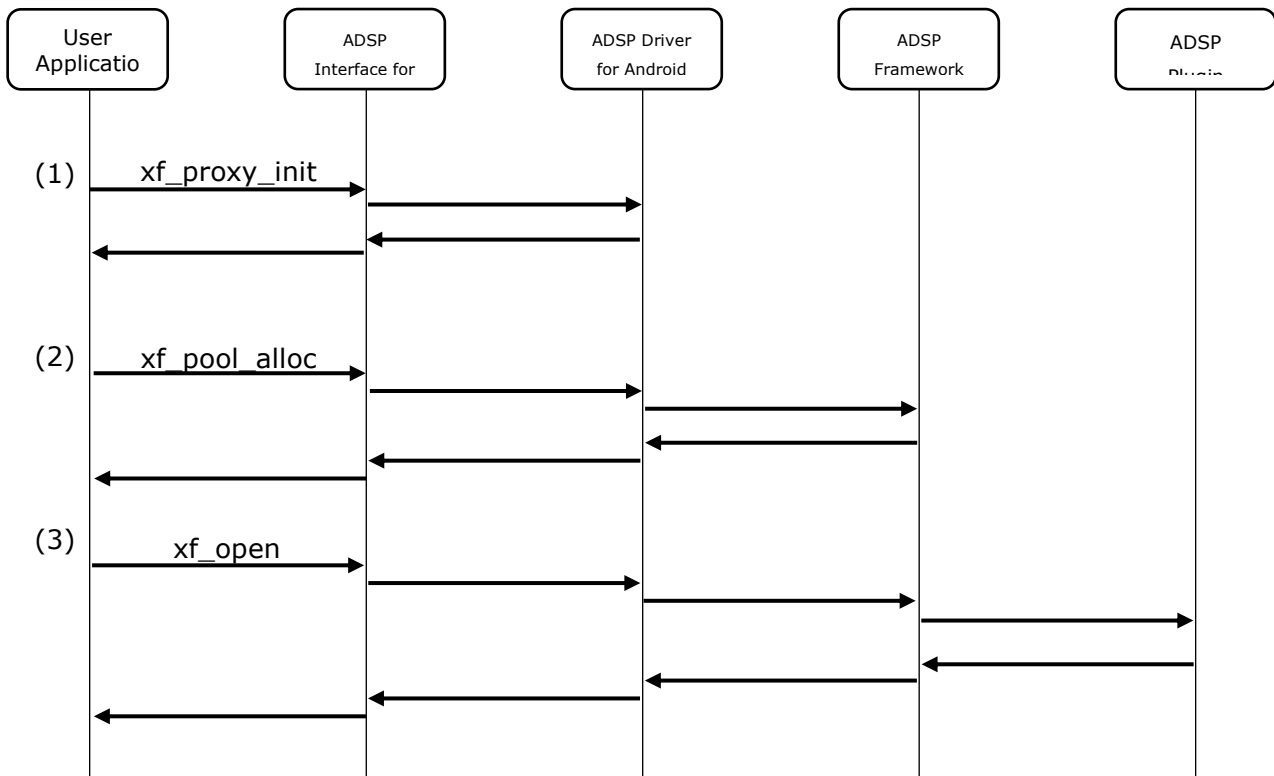## 3.1. Initialization flow

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

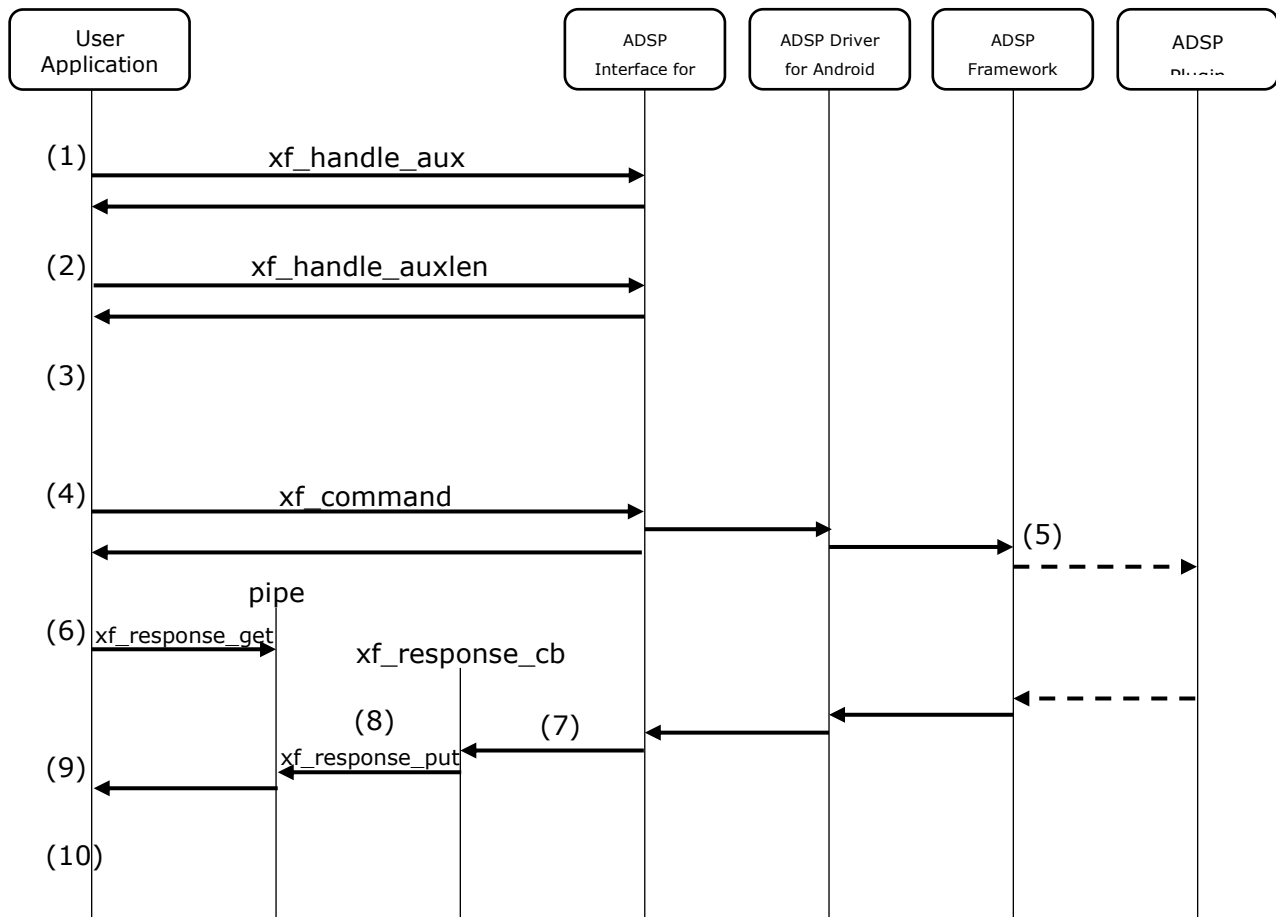Figure 3-2 shows the reference flow to send / receive a command.

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

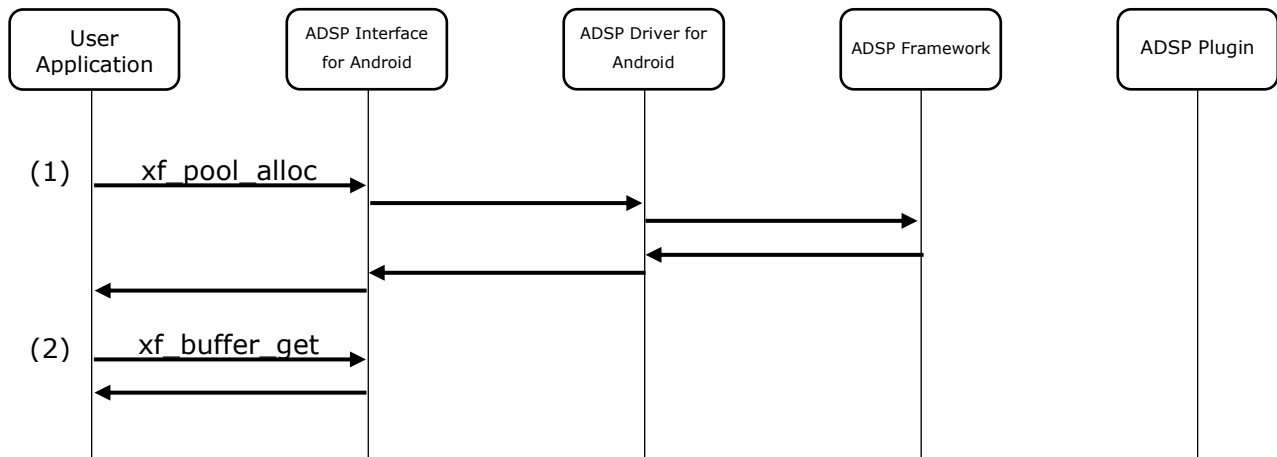Figure 3-1 shows the reference flow to allocate input / output buffer.

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

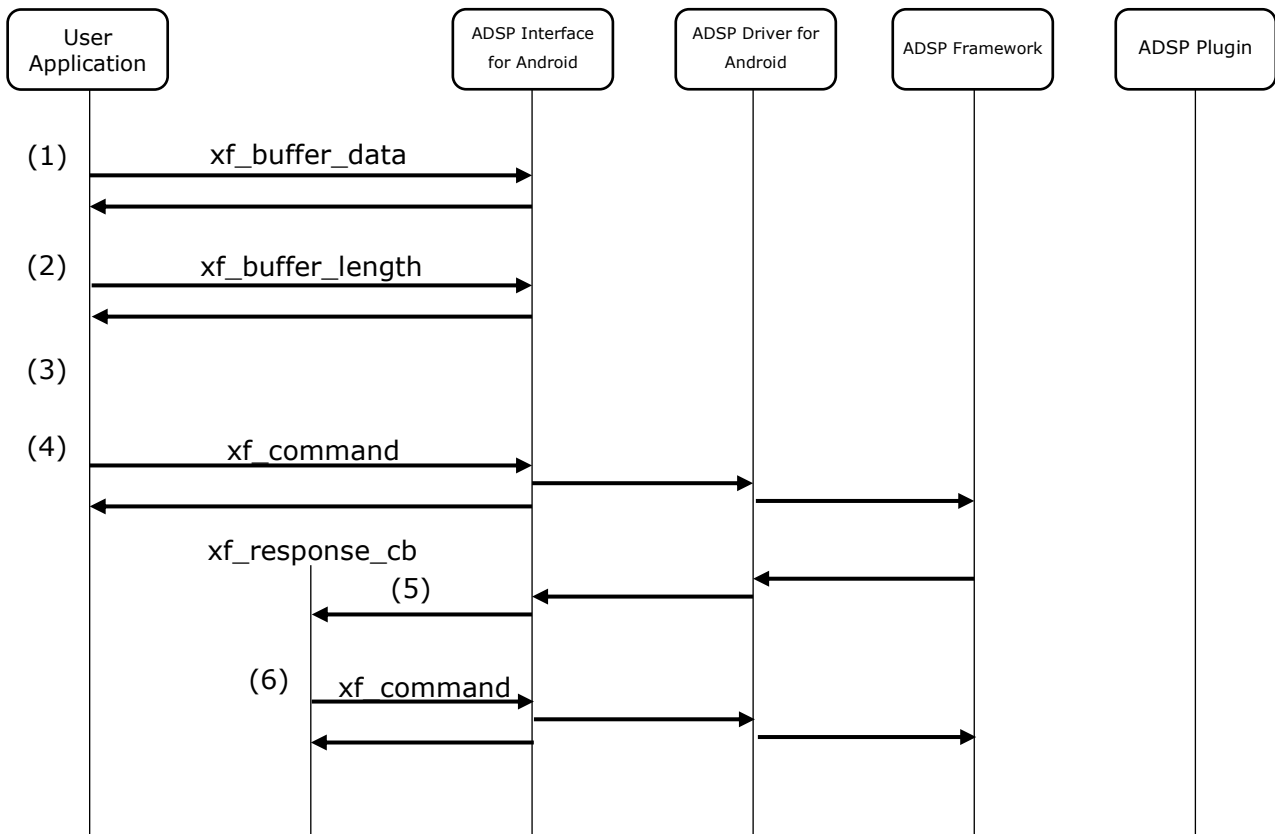Figure 3-2 shows the reference flow to send / receive input buffer.

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

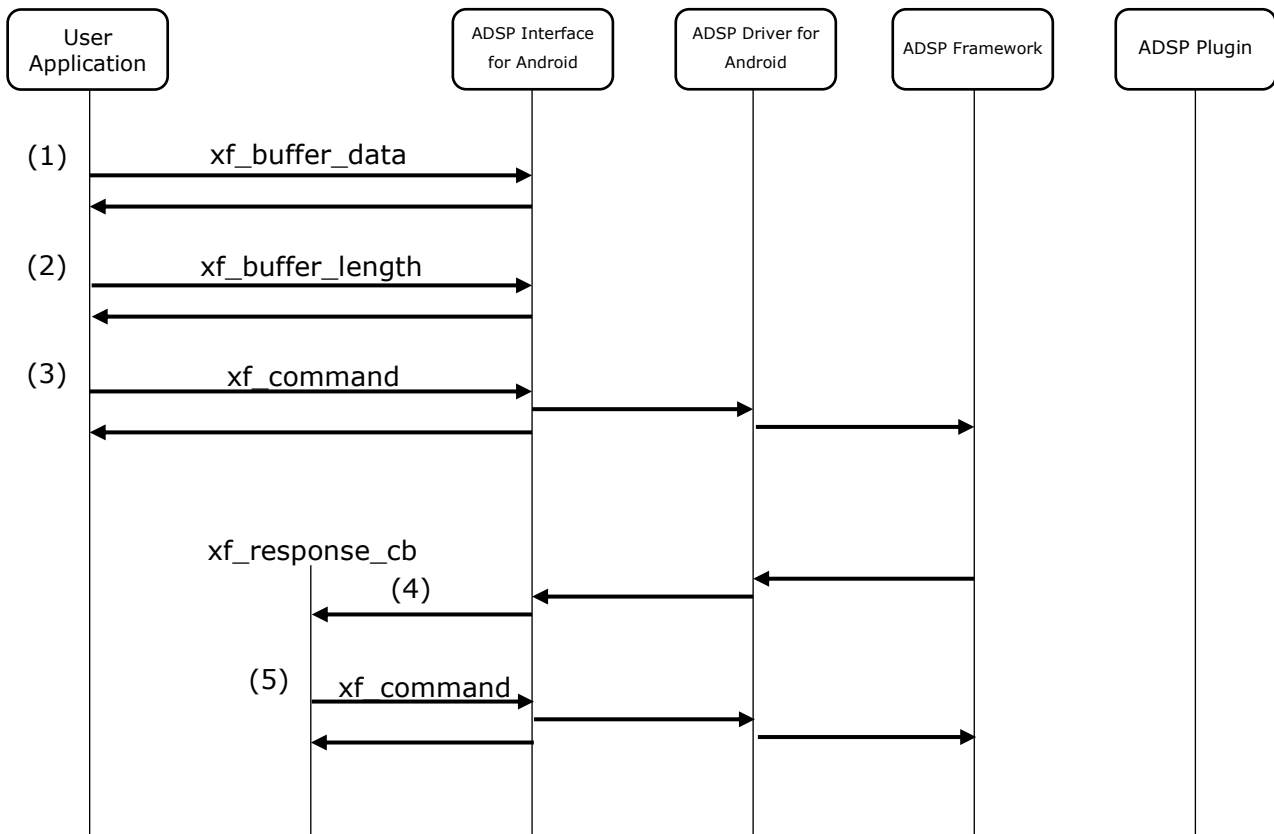Figure 3-2 shows the reference flow to send / receive output buffer.

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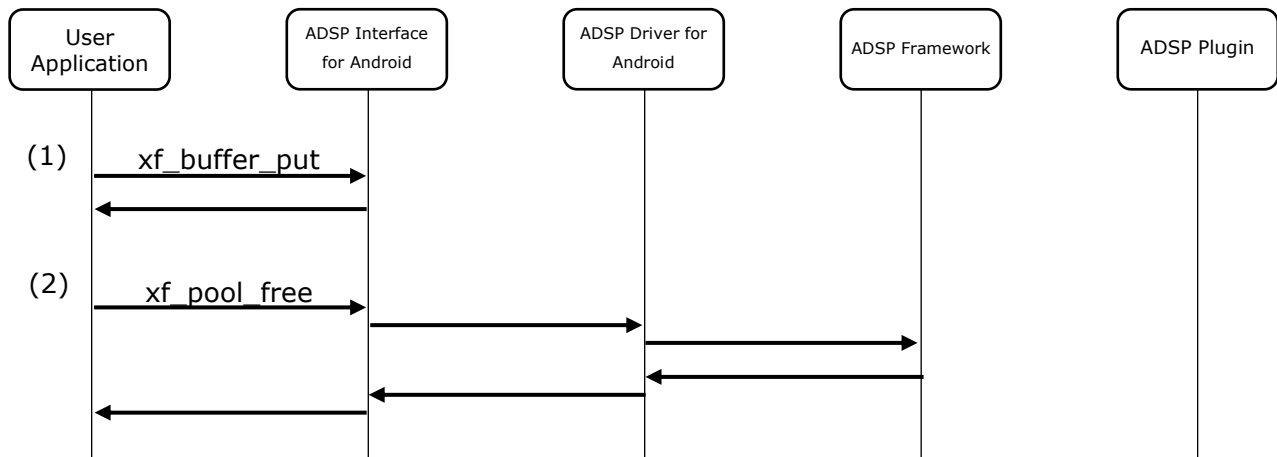
Figure 3-6 The standard flow to release input / output buffer

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

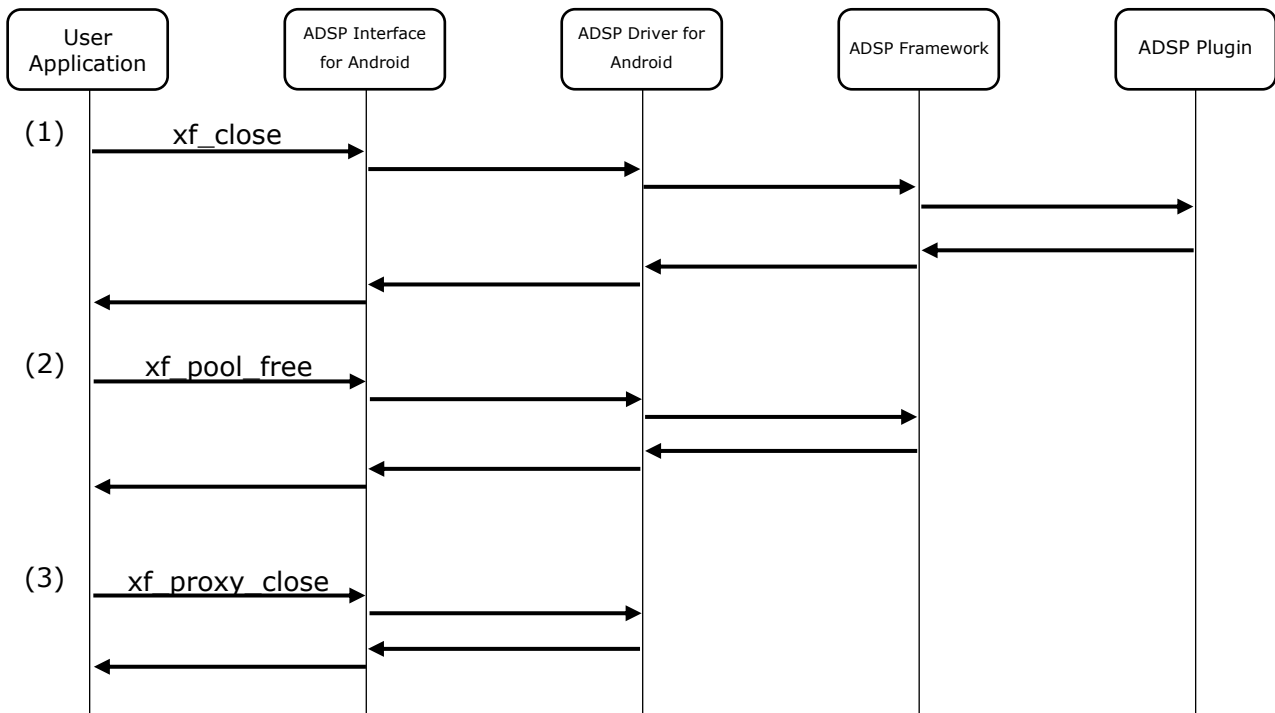## 3.7. Termination flow

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.

Revision History	ADSP Interface for Android User's Manual

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

CONFIDENTIAL

ADSP Interface for Android User's Manual

Publication Date: May 30, 2018 Rev. 0.10

Published by: Renesas Electronics Corporation

RENESAS

RENESAS

ルネサス エレクトロニクス株式会社

http://www.renesas.com

135-0061   3-2-24

https://www.renesas.com/contact/

Colophon 3.1

ADSP Interface for Android
RCG3AHIFA8101ZDP