Application Note

CONFIDENTIAL

ADSP Interface for Linux
RCG3AHIFL4101ZDP

Application Note - Equalizer -

RCG3AHIFL4101ZDPE_AN_EQZ

Rev. 2.00 Dec, 2018

Renesas Electronics
www.renesas.com

Notice

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

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

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

(Rev.4.0-1 November 2017)

Trademarks

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

How to Use This Manual

1. Purpose and Target Reader

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

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

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

2. Restrictions on the Use of this Middleware

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

3. Related Manuals

4. Technical Terms and Abbreviation

- Table of Contents -

- List of Figures -

- List of Tables -

1 Overview

1.1 Overview of this document.

In this chapter, overview of Equalizer interface is explained.

1.2 The architecture of the Software and scope of this document

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

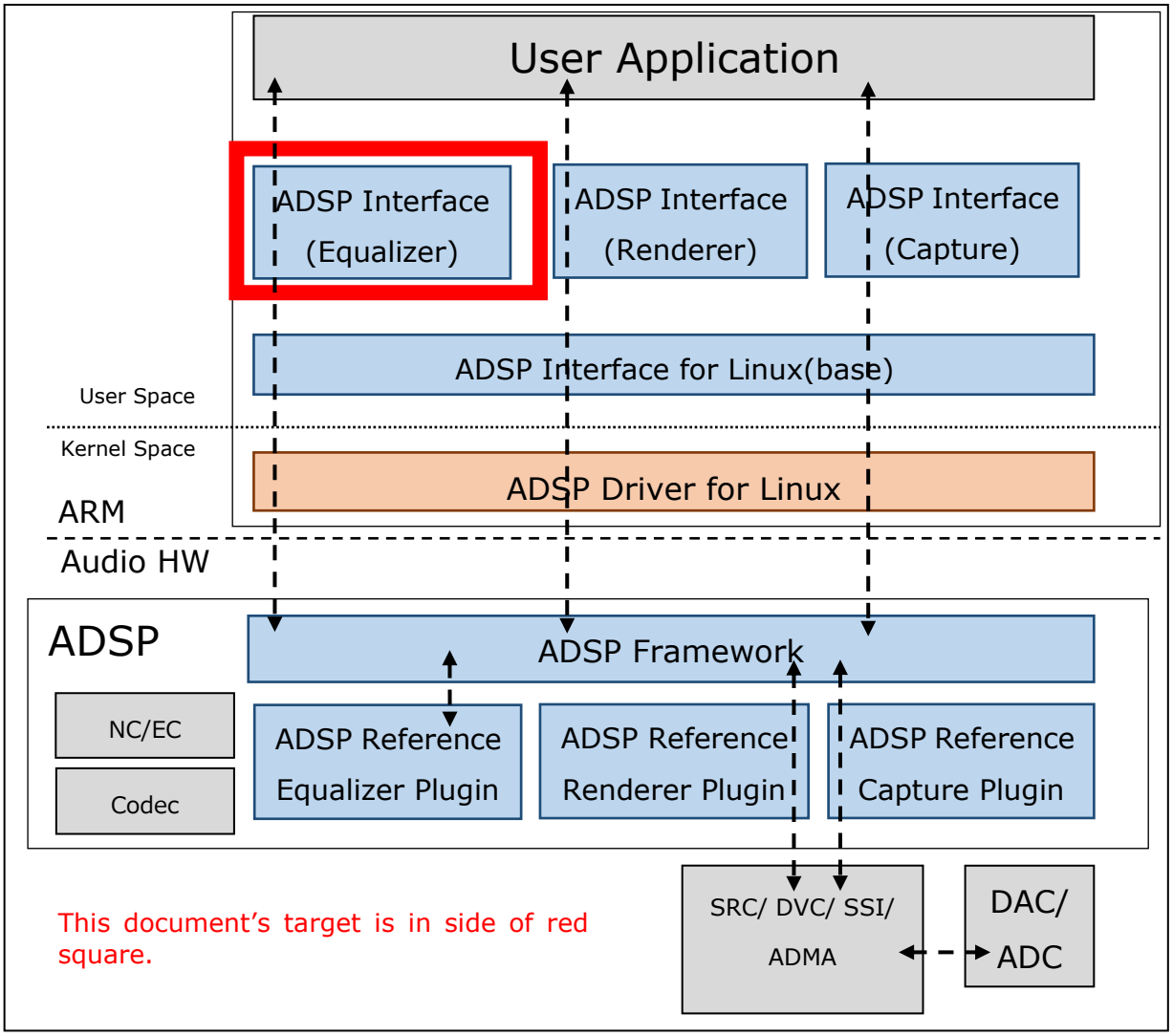

Figure 1-1 The software architecture

1.3 Software necessary to be prepared in advance

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

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]	OpenMAX IL Specification 1.1.2	Renesas Electronics Corporation

2 Software specification

2.1 The list of functions

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

Table 2-1 List of functions

	name	outline
IL Core Methods	OMX_Init	Initialize the OpenMAX™ IL core
	OMX_Deinit	De-initialize the OpenMAX™ IL core
	OMX_GetHandle	Load that component into memory, validate it and return the component handle via the output parameter
	OMX_FreeHandle	Free a component handle (allocated by the OMX_GetHandle)
	OMX_SetupTunnel	Establish a tunnel between components
	OMX_TeardownTunnel	Clears tunneled communication between components
Component APIs	OMX_SendCommand	Send the command from application (IL-client) to component
	OMX_GetParameter	Retrieve the parameter from the component
	OMX_SetParameter	Setup the parameter from the component
	OMX_GetState	Get the current state of the component
	OMX_UseBuffer	Pass the handle to the buffer allocated by application
	OMX_AllocateBuffer	Allocate buffer on behalf of a component
	OMX_FreeBuffer	De-allocate buffer structure
	OMX_EmptyThisBuffer	Pass filled input buffer to the component
	OMX_FillThisBuffer	Pass the free output buffer to the component

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 further detailed descriptions of the structures.

Table 2-2 List of structures

name	outline
XAOMX_AUDIO_PARAM_PARAMETRIC_EQUALIZER	The structure of parameters of parametric equalizer
XAOMX_AUDIO_PARAM_GRAPHIC_EQUALIZER	The structure of parameters of graphic equalizer
XAOXM_AUDIO_PARAM_EQUALIZER	The structure of parameters of equalizer
OMX_AUDIO_PARAM_PCMMODETYPE	PCM mode type structures For further information, refer to OpenMAX IL Specification 1.1.2, section 4.1.7

2.3 Function specifications

2.3.1 IL Core method

2.3.1.1 OMX_Init

OMX_Init		
Synopsis	Initialize the OpenMAX™ IL core, including memory allocation and preparation for loading components. It is used as the very first call into OpenMAX™ IL core.	
Syntax	`OMX_API OMX_ERRORTYPE OMX_APIENTRY OMX_Init();`	
Parameter	None	
Return values	OMX_ErrorInsufficientResources	Failed to initialize due to not enough resource
	OMX_ErrorUndefined	Undefined error while processing command
	OMX_ErrorNone	Normal ends. Initialize successfully

[notice] this function is called only once.

2.3.1.2 OMX_Deinit

OMX_Deinit		
Synopsis	De-initialize OMX IL core, including its allocated memory and objects used to load/manage components. It is used as the very last call into OpenMAX™ IL core.	
Syntax	`OMX_API OMX_ERRORTYPE OMX_APIENTRY OMX_Deinit();`	
Parameter	None	
Return value	OMX_ErrorUndefined	Undefined error while processing command
	OMX_ErrorNone	Normal ends. De-initialize successfully

[notice] this function is called only once.

2.3.1.3 OMX_GetHandle

OMX_GetHandle			
Synopsis	Locate the component specified by the component name given, load that component into memory, validate it and return the component handle via the output parameter		

Syntax	`OMX_API OMX_ERRORTYPE OMX_APIENTRY OMX_GetHandle (`
	`  OMX_OUT OMX_HANDLETYPE * pHandle,`
	`  OMX_IN OMX_STRING cComponentName,`
	`  OMX_IN OMX_PTR pAppData,`
	`  OMX_IN OMX_CALLBACKTYPE * pCallBacks );`		

Parameter	pHandle	A pointer to OMX_HANDLETYPE to be filled in by this method	
	cComponentName	A pointer to a string specifies the component name Supported name for Equalizer is: “OMX.RENESAS.AUDIO.DSP.EQUALIZER”	
	pAppData	A pointer to an IL client-defined value that will be returned during callbacks so that the IL client can identify the source of the callback	
	pCallBacks	A pointer to an OMX_CALLBACKTYPE structure containing the callbacks that the component will use for this IL client	
Return value	OMX_ErrorInsufficientResources		Failed to locate the component due to not enough resource
	OMX_ErrorInvalidState		The proxy is not initialized.
	OMX_ErrorInvalidComponentName		The component name parameter is invalid.
	OMX_ErrorNone		Normal ends.

2.3.1.4 OMX_FreeHandle

OMX_FreeHandle			
Synopsis	Free a handle allocated by the OMX_GetHandle. The IL client should call OMX_FreeHandle only when the component is in the OMX_StateLoaded and when all the ports are not connected via any tunnels.		
Syntax	`OMX_API OMX_ERRORTYPE OMX_APIENTRY OMX_FreeHandle(` `  OMX_IN OMX_HANDLETYPE hComponent);`		
Parameter	hComponent	The handle of the component to be freed	
Return value	OMX_ErrorBadParameter		hComponent points to an invalid memory area.
	OMX_ErrorNone		Normal ends.

2.3.1.5 OMX_SetupTunnel

OMX_SetupTunnel		
Synopsis	Handle the necessary calls to the components to set up the specified tunnel the two components. This method shall be called only when the component is in the OMX_StateLoaded state or when the ports used for the tunnel are disabled (OMX_StateExecuting, OMX_StatePause, or OMX_StateIdle states).	

Syntax	`OMX_API OMX_ERRORTYPE OMX_APIENTRY OMX_SetupTunnel(`
	`  OMX_IN  OMX_HANDLETYPE hOutput,`
	`  OMX_IN  OMX_U32 nPortOutput,`
	`  OMX_IN  OMX_HANDLETYPE hInput,`
	`  OMX_IN  OMX_U32 nPortInput);`	

Parameter	hOutput	Handle of the component whose port, specified in the nPortOutput parameter, will be used as the source for the tunnel
	nPortOutput	Select the source port on component to be used in the tunnel
	hInput	Handle of the component whose port, specified in the nPortInput parameter, will be used the destination for the tunnel
	nPortInput	Select the destination port on component to be used in the tunnel
Return value	OMX_ErrorBadParameter	hOutput and hInput point to invalid memory area.
	OMX_ErrorBadPortIndex	Port index of parameter is invalid.
	OMX_ErrorIncorrectStateOperation	Component is not in OMX_StateLoaded
	OMX_ErrorUndefined	Undefined error while processing command
	OMX_ErrorPortsNotCompatible	One or both components are non-interop components which do not support tunneling.
	OMX_ErrorNone	Normal end

2.3.1.6 OMX_TeardownTunnel

OMX_TeardownTunnel		
Synopsis	Clear tunneled communication between an output port and an input port. After OMX_TeardownTunnel returns successfully, these ports are no longer connected together.	

Syntax	`OMX_API OMX_ERRORTYPE OMX_APIENTRY OMX_TeardownTunnel(`
	`  OMX_IN OMX_HANDLETYPE hOutput,`
	`  OMX_IN OMX_U32 nPortOutput,`
	`  OMX_IN OMX_HANDLETYPE hInput,`
	`  OMX_IN OMX_U32 nPortInput);`	

Parameter	hOutput	Handle of the component whose port, specified in the nPortOutput parameter, is being used as the source for the tunnel
	nPortOutput	Select the source port on component being used in the tunnel
	hInput	Handle of the component whose port, specified in the nPortInput parameter, is being used the destination for the tunnel
	nPortInput	Select the destination port on component being used in the tunnel
Return value	OMX_ErrorBadParameter	hOutput or hInput components point to invalid memory area.
	OMX_ErrorBadPortIndex	Port index is invalid.
	OMX_ErrorIncorrectStateOperation	Component is not in OMX_StateLoaded.
	OMX_ErrorNone	Normal end

2.3.2 Component APIs

2.3.2.1 OMX_SendCommand

OMX_SendCommand		
Synopsis	Receive a command from the client and make a queue for serial execution in separated component thread	
Syntax	OMX_ERRORTYPE OMX_SendCommand(OMX_IN OMX_HANDLETYPE hComponent, OMX_IN OMX_COMMANDTYPE Cmd, OMX_IN OMX_U32 nParam1, OMX_IN OMX_PTR pCmdData);	
Parameter	hComponent	Pointer to memory area of component handle
	Cmd	Type of command
	nParam1	Integer parameter for the command that is to be executed (represented for STATETYPE, number of ports)
	pCmdData	Pointer to a memory area contains specific parameters (marked buffer header)
Return value	OMX_ErrorBadParameter	Parameter(s) is invalid: Command could not be recognized. Invalid mark buffer area Invalid number of ports Invalid destination state (state could not be recognized)
	OMX_ErrorSameState	State transition is requested between same states.
	OMX_ErrorInsufficientResources	Fail to initialize codec setup due to insufficient resources
	OMX_ErrorNotImplemented	Don’t support OMX_StatePause and OMX_StateWaitForResources
	OMX_ErrorIncorrectStateTransition	The transition is invalid such as changing from OMX_StateExecuting to OMX_StatePause, etc.
	OMX_ErrorInvalidState	The executing state is not proper.
	OMX_ErrorBadPortIndex	Port index is invalid.
	OMX_ErrorIncorrectStateOperation	Execution is invalid in the current state of component.
	OMX_ErrorUndefined	Undefined error while processing command
	OMX_ErrorNone	Normal end. Command sending succeeds

2.3.2.2 OMX_GetParameter

OMX_GetParameter		
Synopsis	Receive a parameter structure from IL Client and fill it with appropriate data of component	

Syntax	`OMX_ERRORTYPE OMX_GetParameter(`
	`  OMX_IN OMX_HANDLETYPE hComponent,`
	`  OMX_IN OMX_INDEXTYPE nParamIndex,`
	`  OMX_INOUT OMX_PTR pComponentParameterStructure);`	

Parameter	hComponent	Pointer to memory area of component handle
	nParamIndex	The index of the structure that is to be sent. This value is from the OMX_INDEXTYPE enumeration. Supported index are: OMX_IndexParamPortDefinition OMX_IndexParamAudioPortFormat OMX_IndexParamPriorityMgmt OMX_IndexParamAudioPcm XAOMX_IndexParamAudioEqualizer
	pComponentParameterStructure	Pointer to the IL client-allocated parameter structure
Return value	OMX_ErrorUnsupportedIndex	Cannot recognize parameters
	OMX_ErrorBadParameter	Parameter is invalid for execution: pComponentParameterStructure points to an invalid memory area.
	OMX_ErrorIncorrectStateOperation	Current state is OMX_StateInvalid.
	OMX_ErrorBadPortIndex	Port index of parameter is invalid.
	OMX_ErrorNone	Normal ends. Getting parameter from component is successful.

2.3.2.3 OMX_SetParameter

OMX_SetParameter		
Synopsis	Fill a parameter structure allocated by IL Client with appropriate data of component	

Syntax	`OMX_ERRORTYPE OMX_SetParameter(`
	`  OMX_IN OMX_HANDLETYPE hComponent,`
	`  OMX_IN OMX_INDEXTYPE nIndex,`
	`  OMX_IN OMX_PTR pComponentParameterStructure);`	

Parameter	hComponent	Pointer to memory area of component handle
	nIndex	The index of the structure that is to be sent. It indicates which structure is requested by IL Client. This value is from the OMX_INDEXTYPE enumeration. Supported indexes are: OMX_IndexParamPortDefinition OMX_IndexParamAudioPortFormat OMX_IndexParamPriorityMgmt OMX_IndexParamStandardComponentRole OMX_IndexParamAudioPcm XAOMX_IndexParamAudioEqualizer
	pComponentParameterStructure	Pointer to the IL client-allocated parameter structure to be filled
Return value	OMX_ErrorBadParameter	Parameter is invalid for execution: pComponentParameterStructure points to an invalid memory area.
	OMX_ErrorIncorrectStateOperation	Current state is OMX_StateInvalid. Port is locked. Current state is not OMX_StateLoaded. (for OMX_IndexParamPriorityMgmt and OMX_IndexParamStandardComponentRole)
	OMX_ErrorBadPortIndex	Port index of parameter is invalid.
	OMX_ErrorUnsupportedIndex	The type (index) of parameter structure is not supported by component.
	OMX_ErrorNone	Normal ends. Setting parameter to component is successful.

2.3.2.4 OMX_GetState

OMX_GetState		
Synopsis	Return the current state of the component	

Syntax	`OMX_ERRORTYPE OMX_GetState(`
	`  OMX_IN OMX_HANDLETYPE hComponent,`
	`  OMX_OUT OMX_STATETYPE *pState);`	

Parameter	hComponent	Pointer to memory area of component handle
	*pState	Pointer to an allocated memory area used to store component state
Return value	OMX_ErrorBadParameter	Parameter is invalid for execution: pState points to an invalid memory area.
	OMX_ErrorNone	Normal end. Getting the state of the component is successful.

2.3.2.5 OMX_UseBuffer

OMX_UseBuffer		
Synopsis	Use a buffer allocated by the IL Client to a port or supplied by a tunneling component	

Syntax	`OMX_ERRORTYPE OMX_UseBuffer(`
	`  OMX_IN OMX_HANDLETYPE hComponent,`
	`  OMX_OUT OMX_BUFFERHEADERTYPE **ppBufferHdr,`
	`  OMX_IN OMX_U32 nPortIndex,`
	`  OMX_IN OMX_PTR pAppPrivate,`
	`  OMX_IN OMX_U32 nSizeBytes,`
	`  OMX_IN OMX_U8 *pBuffer);`	

Parameter	hComponent	Pointer to memory area of component handle
	**ppBufferHdr	Pointer to OMX_BUFFERHEADERTYPE which contains meta-information about the buffer. It receives the pointer to the buffer header.
	nPortIndex	Target port that uses the buffer (index into the port definition array of the component)
	pAppPrivate	Pointer to the private memory area of IL Client. It is used to initialize the pAppPrivate member of the buffer header structure.
	nSizeBytes	The size (byte) of the buffer to allocate
	*pBuffer	Pointer to the allocated buffer to be used
Return value	OMX_ErrorBadParameter	Parameter is invalid for execution: ppBufferHdr points to an invalid memory area. Target port is invalid.
	OMX_ErrorIncorrectStateOperation	Port is not populated.
	OMX_ErrorInsufficientResources	Not enough resources
	OMX_ErrorUndefined	Undefined error while processing command
	OMX_ErrorNone	Normal end. Setting the buffer to the target port is successful.

2.3.2.6 OMX_AllocateBuffer

OMX_AllocateBuffer		
Synopsis	Allocate the buffer and the buffer header and return the pointer to the buffer header	

Syntax	`OMX_ERRORTYPE OMX_AllocateBuffer(`
	`  OMX_IN OMX_HANDLETYPE hComponent,`
	`  OMX_INOUT OMX_BUFFERHEADERTYPE **ppBuffer,`
	`  OMX_IN OMX_U32 nPortIndex,`
	`  OMX_IN OMX_PTR pAppPrivate,`
	`  OMX_IN OMX_U32 nSizeBytes);`	

Parameter	hComponent	Pointer to memory area of component handle
	**ppBuffer	Pointer to OMX_BUFFERHEADERTYPE which contains meta-information about the buffer. It receives the pointer to the buffer header.
	nPortIndex	Target port (index into the port definition array of the component)
	pAppPrivate	Pointer to the private memory area of IL Client. It is used to initialize the pAppPrivate member of the buffer header structure.
	nSizeBytes	The size (byte) of the buffer to allocate
Return value	OMX_ErrorBadParameter	Parameter is invalid for execution: ppBuffer points to an invalid memory area. Target port is invalid.
	OMX_ErrorIncorrectStateOperation	Port is not populated.
	OMX_ErrorInsufficientResources	Not enough resources
	OMX_ErrorUndefined	Undefined error while processing command
	OMX_ErrorNone	Normal end. Setting the buffer to the target port is successful.

2.3.2.7 OMX_FreeBuffer

OMX_FreeBuffer		
Synopsis	De-allocate buffer structure	
Syntax	OMX_ERRORTYPE OMX_FreeBuffer(OMX_IN OMX_HANDLETYPE hComponent, OMX_IN OMX_U32 nPortIndex, OMX_IN OMX_BUFFERHEADERTYPE *pBuffer);	
Parameter	hComponent	Pointer to memory area of component handle
	nPortIndex	Target port (index into the port definition array of the component)
	*pBuffer	Pointer to OMX_BUFFERHEADERTYPE structure which contains meta-information about the buffer. It specifies the index of the input port that receives the buffer.
Return value	OMX_ErrorBadParameter	Parameter is invalid: pBuffer points to an invalid memory area. Target port is invalid.
	OMX_ErrorIncorrectStateOperation	The port is not unpopulated (all buffers of the port is active (being used), so cannot free the buffer).
	OMX_ErrorUndefined	Undefined error while processing command
	OMX_ErrorNone	Normal end. Transferring the buffer to the client is successful.

2.3.2.8 OMX_EmptyThisBuffer

OMX_EmptyThisBuffer		
Synopsis	Send a filled buffer to an input port of a component	

Syntax	`OMX_ERRORTYPE OMX_EmptyThisBuffer(`
	`  OMX_IN OMX_HANDLETYPE hComponent,`
	`  OMX_IN OMX_BUFFERHEADERTYPE *pBuffer);`	

Parameter	hComponent	Pointer to memory area of component handle
	*pBuffer	Pointer to OMX_BUFFERHEADERTYPE structure which contains meta-information about the buffer. It specifies the index of the input port that receives the buffer.
Return value	OMX_ErrorBadParameter	Parameter is invalid: pBuffer points to an invalid memory area. Invalid buffer header Buffer filled length is zero.
	OMX_ErrorBadPortIndex	Port index of buffer is invalid.
	OMX_ErrorIncorrectStateOperation	Execution is invalid in current state of the component. Component is not in OMX_StateExecuting. Sending a buffer after end-of-stream
	OMX_ErrorNone	Normal end. Buffer is transferred to the input port of a component successfully.

2.3.2.9 OMX_FillThisBuffer

OMX_FillThisBuffer		
Synopsis	Receive an empty buffer to an output port of a component and fill it with appropriate output data	

Syntax	`OMX_ERRORTYPE OMX_FillThisBuffer(`
	`  OMX_IN OMX_HANDLETYPE hComponent,`
	`  OMX_IN OMX_BUFFERHEADERTYPE *pBuffer);`	

Parameter	hComponent	Pointer to memory area of component handle
	*pBuffer	Pointer to OMX_BUFFERHEADERTYPE which contains meta-information about the buffer. It specifies the index of the output port that receives the buffer.
Return value	OMX_ErrorBadParameter	Parameter is invalid for execution: pBuffer points to an invalid memory area. Invalid buffer header
	OMX_ErrorBadPortIndex	Port index of buffer is invalid.
	OMX_ErrorIncorrectStateOperation	Execution is invalid in current state of component. Output port is enabled. Sending a buffer after end-of-stream
	OMX_ErrorNone	Normal ends. Transferring buffer to client is successful.

2.4 Callback function specification

The OpenMAX IL includes a callback mechanism that allows a component to communicate the IL client. To accomplish a callback, the IL client has three callback functions defined: a generic event handler and two callbacks related to the dataflow (`EmptyBufferDone` and `FillBufferDone`).

The IL client is responsible for filling in an `OMX_CALLBACKTYPE` structure with its callback entry points and passing the structure to the OpenMAX IL core at initialization (init) time.

`OMX_CALLBACKTYPE` is defined as follows.

```
typedef struct OMX_CALLBACKTYPE {
        OMX_ERRORTYPE (*EventHandler)(
                OMX_IN OMX_HANDLETYPE hComponent,
                OMX_IN OMX_PTR pAppData,
                OMX_IN OMX_EVENTTYPE eEvent,
                OMX_IN OMX_U32 nData1,
                OMX_IN OMX_U32 nData2,
                OMX_IN OMX_PTR pEventData);
        OMX_ERRORTYPE (*EmptyBufferDone)(
                OMX_IN OMX_HANDLETYPE hComponent,
                OMX_IN OMX_PTR pAppData,
                OMX_IN OMX_BUFFERHEADERTYPE* pBuffer);
        OMX_ERRORTYPE (*FillBufferDone)(
                OMX_IN OMX_HANDLETYPE hComponent,
                OMX_IN OMX_PTR pAppData,
                OMX_IN OMX_BUFFERHEADERTYPE* pBuffer);
} OMX_CALLBACKTYPE;
```

2.4.1 EventHandler

A component uses the EventHandler method to notify the IL client when an event of interest occurs within the component. The OMX_EVENTTYPE enumeration defines the set of OpenMAX IL events; refer to the definition of this enumeration for the meaning of each event.

The EventHandler method is defined as follows.

```
OMX_ERRORTYPE(*OMX_CALLBACKTYPE::EventHandler)(
                        OMX_IN OMX_HANDLETYPE hComponent,
                        OMX_IN OMX_PTR pAppData,
                        OMX_IN OMX_EVENTTYPE eEvent,
                        OMX_IN OMX_U32 nData1,
                        OMX_IN OMX_U32 nData2,
                        OMX_IN OMX_PTR pEventData)
```

The information carried within nData1, nData2 and pEventData varies depending on OMX_EVENTTYPE, refer to Table 3-7 of OMX IL Specification v1.1.2 for specific details.

During the processing, component may update some information of output port from default values to exact values. User should take into account the OMX_EventPortSettingsChanged to correct their configurations by getting parameters from component again. Note that, for output port, user has to perform necessary steps to reconfigure the port (see 3.4.5 of OMX IL Specification v1.1.2 for more detail of sequence). However, for input port, user just has to get the parameter again and must not process any further step.

For more detail, please refer to OMX IL Specification 1.1.2, section 3.1.2.9.1.

2.4.2 EmptyBufferDone

A component uses the EmptyBufferDone callback to pass a buffer from an input port back to the IL client. A component updates the nOffset and nFilledLen values of the buffer header to reflect the portion of the buffer it consumed; for example, nFilledLen is set equal to 0 if completely consumed.

In addition to facilitating normal data flow between an executing component and the IL client, a component uses the EmptyBufferDone function to return input buffers to the IL client in the following cases:

- The IL client commands a transition from OMX_StateExecuting or OMX_StatePause to OMX_StateIdle.
- The IL client flushes or disables a port.

In these cases, a component may also return a partially consumed input buffer to the IL client. The EmptyBufferDone call is defined as follows.

```
OMX_ERRORTYPE(*OMX_CALLBACKTYPE::EmptyBufferDone)(
                        OMX_IN OMX_HANDLETYPE hComponent,
                        OMX_IN OMX_PTR pAppData,
                        OMX_IN OMX_BUFFERHEADERTYPE* pBuffer)
```

For more detail, please refer to OMX IL Specification 1.1.2, section 3.1.2.9.2.

2.4.3 FillBufferDone

Component uses the FillBufferDone callback to pass a buffer from an output port back to the IL client. Component sets the nOffset and nFilledLen of the buffer header to reflect the portion of the buffer it filled; for example, nFilledLen is equal to 0 if it contains no data).

In addition to facilitating normal dataflow between an executing component and the IL client, a component uses this function to return output buffers to the IL client in the following cases:

- The IL client commands a transition from OMX_StateExecuting or OMX_StatePause to OMX_StateIdle.
- The IL client flushes or disables a port.

FillBufferDone is defined as follows.

```
OMX_ERRORTYPE(*OMX_CALLBACKTYPE::FillBufferDone)(
                        OMX_IN OMX_HANDLETYPE hComponent,
                        OMX_IN OMX_PTR pAppData,
                        OMX_IN OMX_BUFFERHEADERTYPE* pBuffer)
```

For more detail, please refer to OMX IL Specification 1.1.2, section 3.1.2.9.3.

2.5 Structures specification

2.5.1 XAOMX_AUDIO_PARAM_PARAMETRIC_EQUALIZER

XAOMX_AUDIO_PARAM_PARAMETRIC_EQUALIZER			
Synopsis	This is the structure of parameters of parametric equalizer. Specify the parameters of parametric filters		
Member	OMX_U32	nSize	The size of the structure in bytes
	OMX_VERSIONTYPE	nVersion	OMX specification version information
	OMX_S32	Type[9]	The type of the filter Value range: Peaking, Bass, Treble, Through filter Through is default filter type.
	OMX_S32	FreqCenter[9]	The frequency center of a filter Value range: -Peaking filter: 20-20kHz (or less than Nyquist frequency) -Bass filter: 50-500Hz -Treble filter: 5k - 11kHz
	OMX_S32	Gain[9]	The gain of a filter Value range: $10^{-\frac{15}{20}} \times 2^{28}$ to $10^{\frac{15}{20}} \times 2^{28}$ (fixed point Q4.28)
	OMX_S32	BandWidth[9]	The bandwidth of a filter Value range: 0.2×2^{27} to 15×2^{27} (fixed point Q5.27)
	OMX_S32	GainBase[9]	The gain base of a filter Value range: $10^{-\frac{10}{20}} \times 2^{28}$ to $10^{\frac{10}{20}} \times 2^{28}$ (fixed point Q4.28)

2.5.2 XAOMX_AUDIO_PARAM_GRAPHIC_EQUALIZER

XAOMX_AUDIO_PARAM_GRAPHIC_EQUALIZER			
Synopsis	This is the structure of parameters of graphic equalizer. Specify the parameters of graphic filter		
Member	OMX_U32	nSize	The size of the structure in bytes
	OMX_VERSIONTYPE	nVersion	OMX specification version information
	OMX_S32	Gain_g[5]	The gain of the graphic filter Value range: $10^{-\frac{10}{20}} \times 2^{28}$ to $10^{\frac{10}{20}} \times 2^{28}$ (fixed point Q4.28)

2.5.3 XAOXM_AUDIO_PARAM_EQUALIZER

XAOXM_AUDIO_PARAM_EQUALIZER			
Synopsis	This is the structure of parameters of equalizer. Specify the parameters of the equalizer		
Member	OMX_U32	nSize	The size of the structure in bytes
	OMX_VERSIONTYPE	nVersion	OMX specification version information
	OMX_S32	Eqz_type	The type of the equalizer Value range: - Parametric Equalizer: 0 - Graphic Equalizer: 1
	XAOMX_AUDIO_PARAM_PARAMETRIC_EQUALIZER	stEqCoef	Parametric equalizer coefficient setting parameters
	XAOMX_AUDIO_PARAM_GRAPHIC_EQUALIZER	stEqGCoef	Graphic equalizer coefficient setting parameters

For detail about PCM stream structure of Equalizer, the software supports the below settings in table 2-7.

The table 2-7 reveals the detailed definitions of PCM structure. I/O column indicates the element is input or output; Input Values column indicates the valid input values set by user.which are to make the system work properly.

Table 2-3 PCM stream setting of Equalizer

Element	**I/O**	**Input Values**	**Default**	**Description**
nChannels	I/O	1 or 2 channels	2	Set channels of PCM stream
nBitPerSample	I/O	16 or 24 bits	16	Set the PCM width of PCM stream
nSamplingRate	I/O	valid values: 32,000 / 44,100 / 48,000 Hz	44100	Set the sampling frequency of PCM stream

3 Process sequence

3.1 Initialize Component

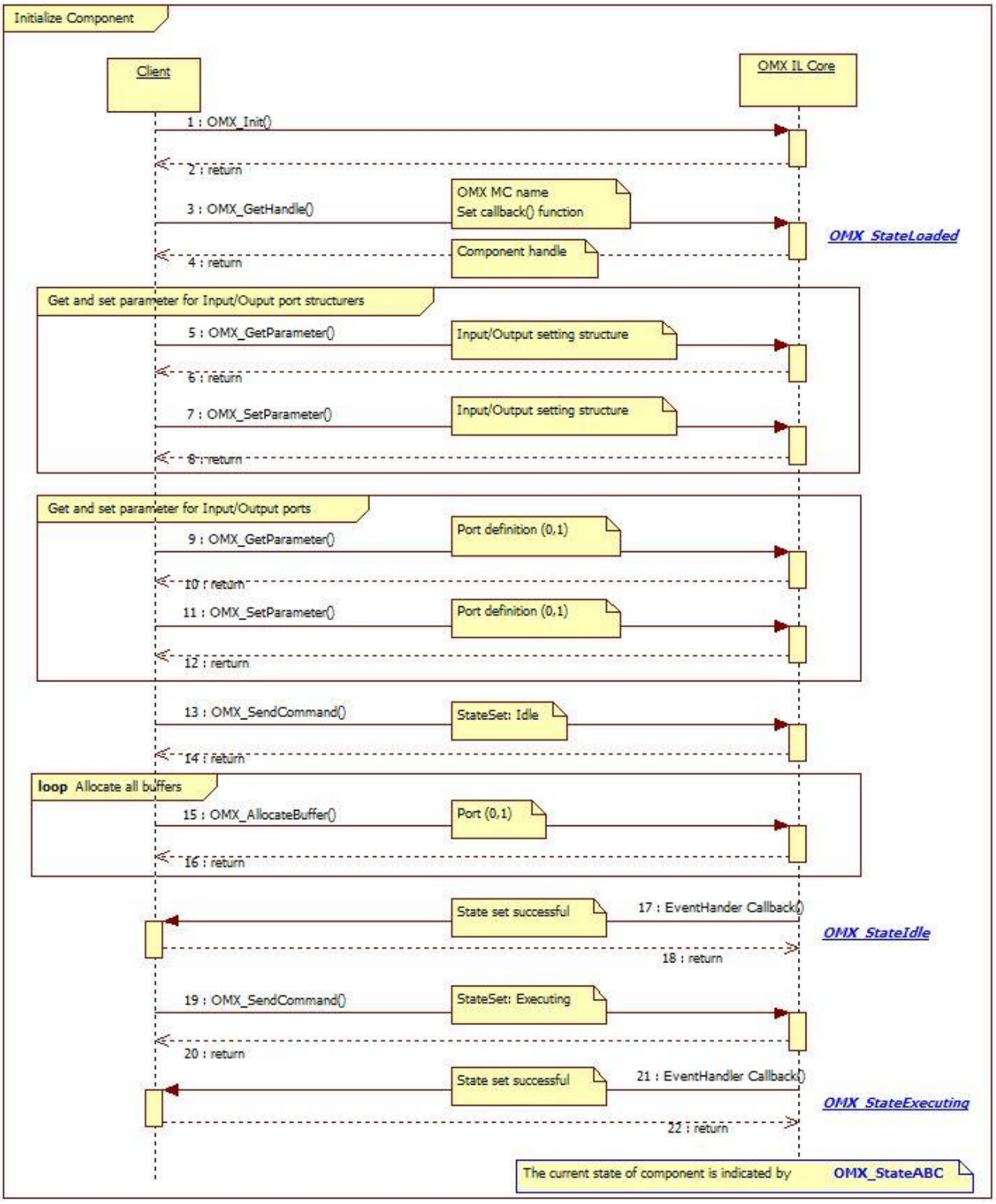

Figure 3-1 Initialize the Component and preparation phase

3.2 Decoding sequence

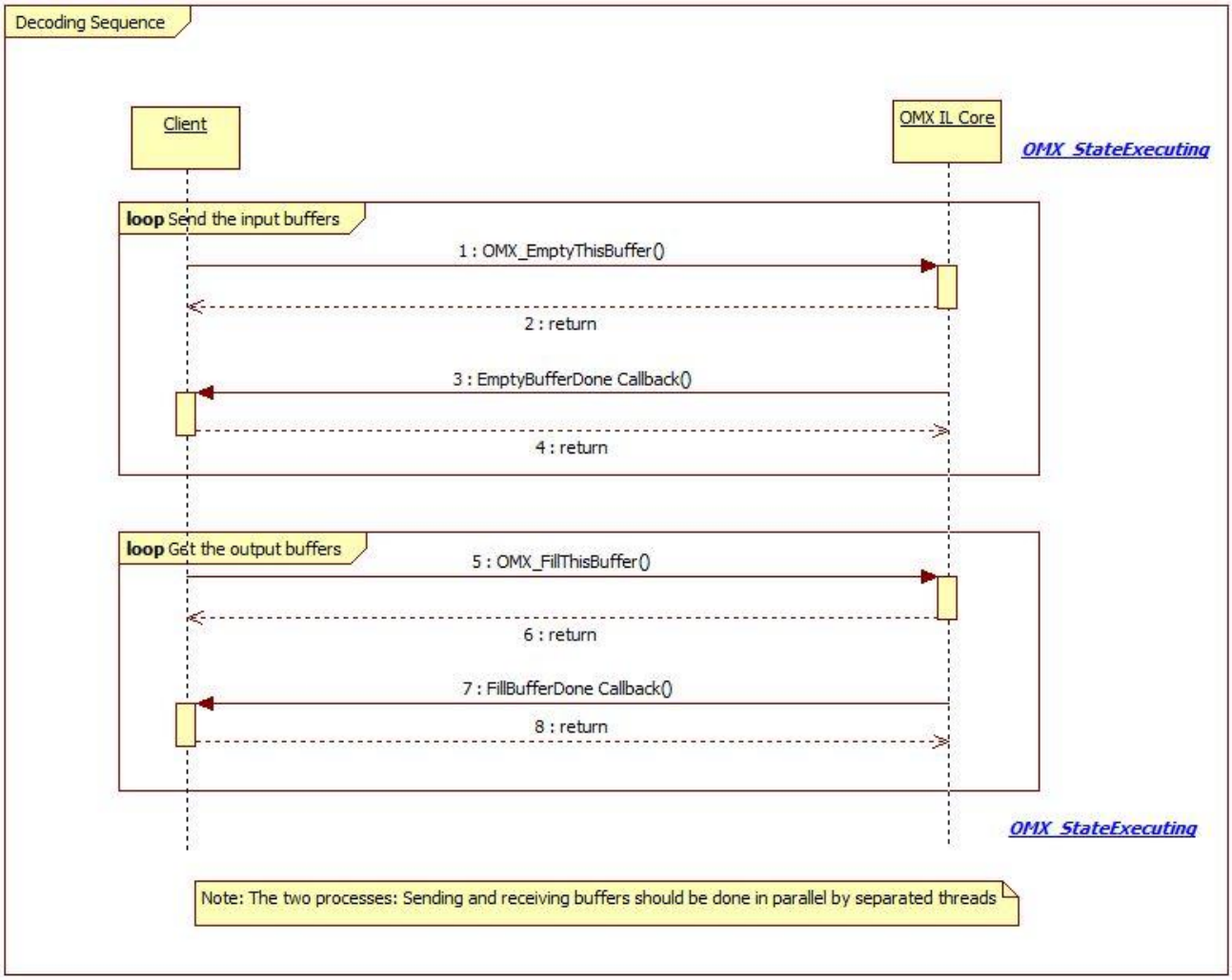

Figure 3-2 Decoding sequence

3.3 De-initialize Component

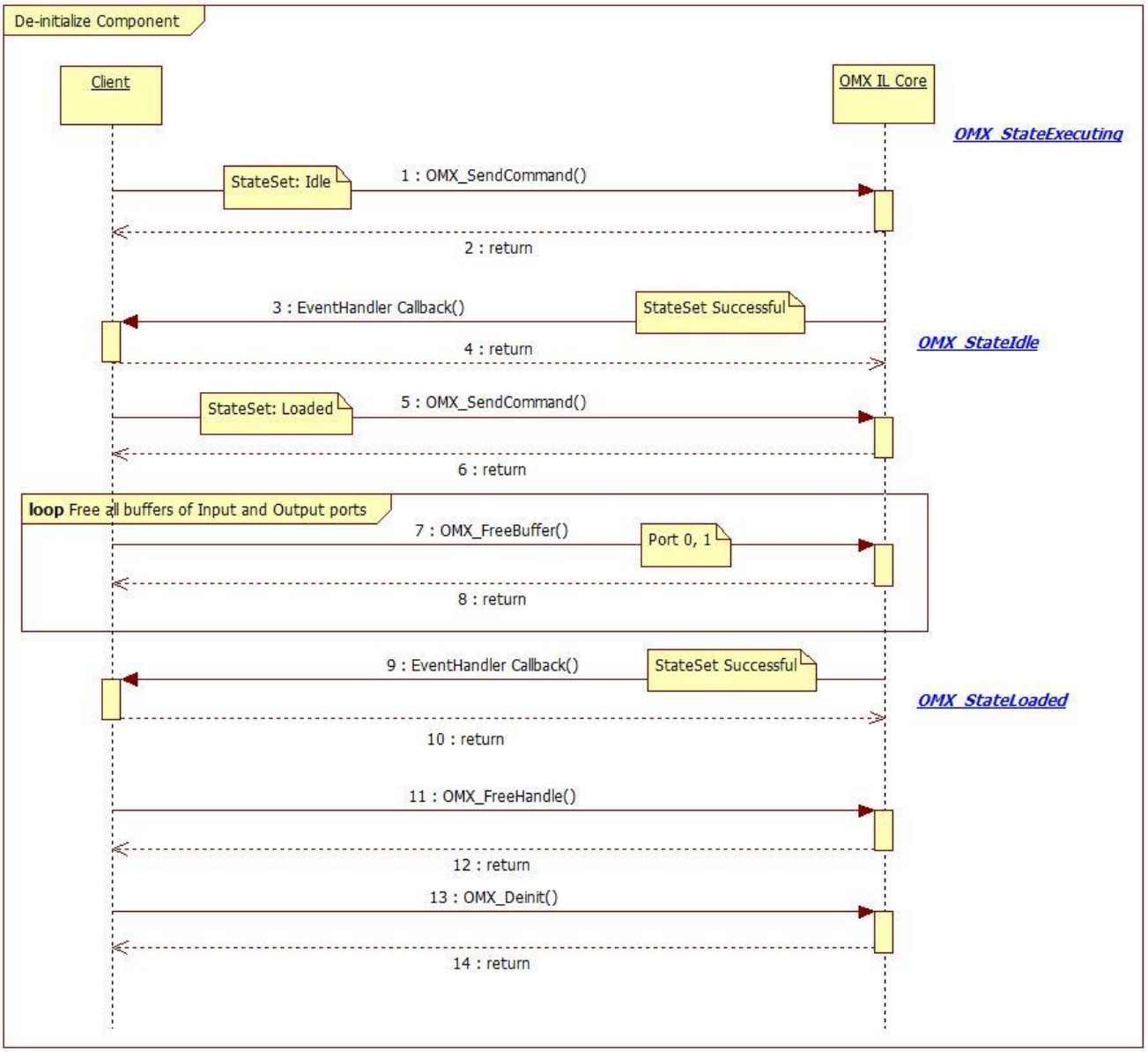

Figure 3-3 De-Initialize Component and OMX IL Core

Note: The order of 2: return (of SendCommand) and 3: EventCallback is not guaranteed. It depends on the current status of component.

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 Linux Application Note - Equalizer -

Rev.	Date	Description	
		Page	Summary
1.00	Jan. 29, 2018	-	New Create
1.01	Jun. 28, 2018	-	Style Modify
2.00	Dec. 25, 2018	-	Official Release
		13	Change details of “OMX_ErrorIncorrectStateTransition”

ADSP Interface for Linux Application Note - Equalizer -

Publication Date: Dec 25, 2018 Rev. 2.00

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 Linux
RCG3AHIFL4101ZDP

RCG3AHIFL4101ZDPE_AN_EQZ

Renesas Electronics Corporation