RENESAS

CONFIDENTIAL

ADSP TDM Renderer/Capture Interface for Linux
RCG3AHIFL4001ZDP

Application Note - TDM Renderer/Capture -

RCG3AHIFL4001ZDPE_AN_RDR

Rev. 1.00   Jul, 2017

Renesas Electronics
www.renesas.com

CONFIDENTIAL

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 disputes 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, drawing, chart, program, algorithm, 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 otherwise misappropriate 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, copy or otherwise misappropriation of Renesas Electronics products.

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; and 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.
   Renesas Electronics products are neither intended nor 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 damages (space and 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 third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.

6. When using the 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 radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond 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. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of 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 by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate 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. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use 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 not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.

10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.

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 majority-owned subsidiaries.

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

(Rev.3.0-1   November 2016)

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.

RENESAS

- Table of Contents -

RENESAS

- List of Figures -

- List of Tables -

1 Overview

1.1 Overview of this document.

In this chapter, overview of TDM Renderer/Capture interface is explained.

1.2 The architecture of the Software and scope of this document

The architecture of ADSP TDM Renderer/Capture Interface for Linux is shown in Figure 1-1. ADSP TDM Renderer/Capture Interface for Linux is a user space library which provides the interface to control ADSP TDM Renderer Plugin and ADSP TDM Capture Plugin.
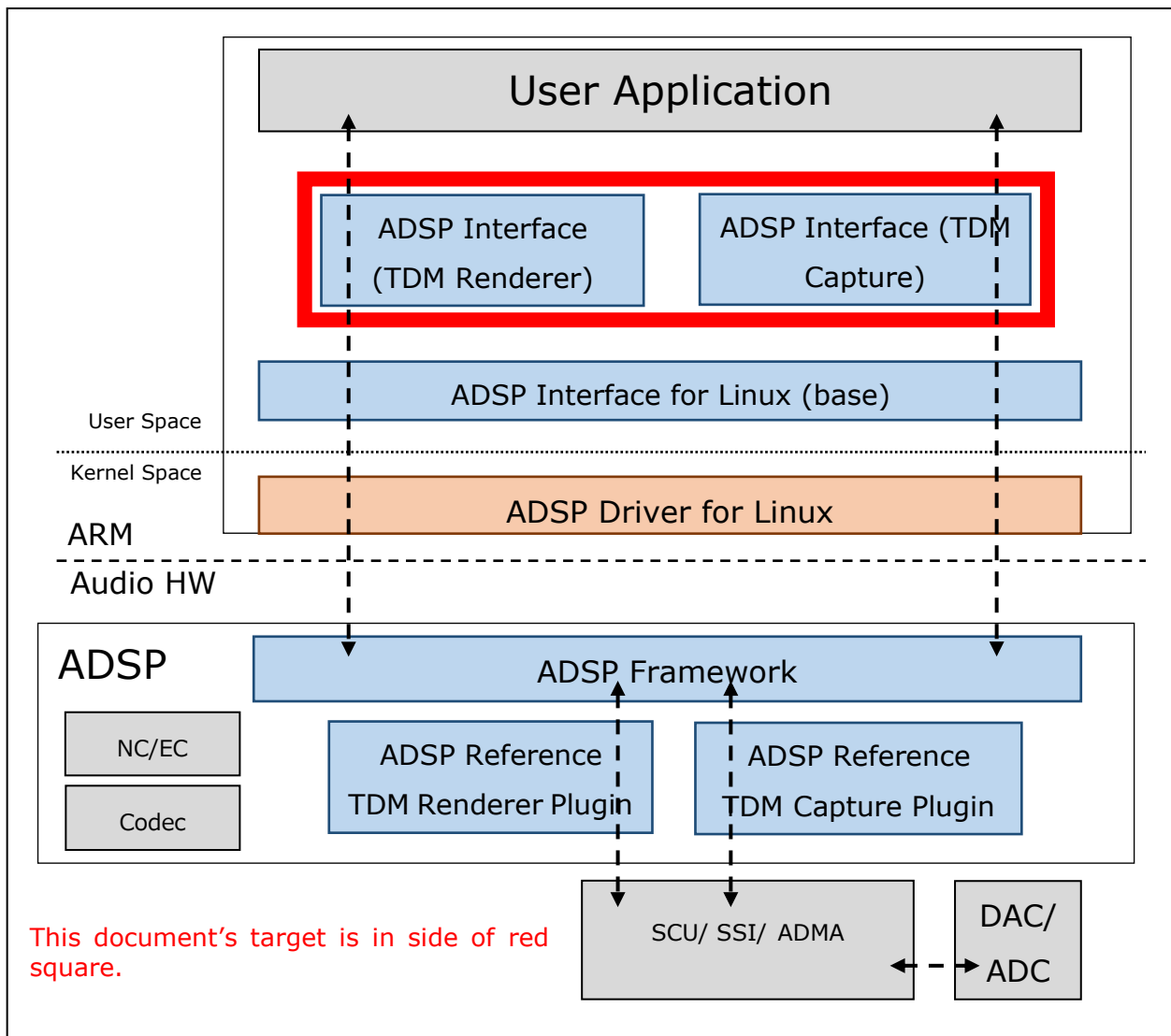
Figure 1-1 The software architecture

Note:

- TDM Renderer function is used to merge multiple raw Pulse Code Modulation (PCM) audio data stream and then output to the configured output device. The user can setup the data path before using TDM Renderer Interface or can setting the data path inside of TDM Renderer Interface.

- TDM Capture function is used to capture/record the raw Pulse Code Modulation (PCM) audio data from configured input device and then spitting the data into multiple output streams. The user can setup the data path before using TDM Capture Interface or can setting the data path inside of TDM Capture Interface.

- Due to development environment (Salvator-X board) does not supports TDM format for input/output hardware. TDM input/output changes to 2 channels.

1.3 Software necessary to be prepared in advance
ADSP Driver for Linux should be loaded in advance to use ADSP TDM Renderer/Capture 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 more detailed specification of the functions.

Table 2-1 List of functions

	Name	Outline
IL Core Method	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 API	OMX_SendCommand	Send the command from application (IL-client) to component
	OMX_GetParameter	Retrieve the parameter from the component
	OMX_SetParameter	Setup the parameter to 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

ADSP TDM Renderer/Capture Interface for Linux Application Note - TDM R
enderer/Capture -

Table 2-2 shows the different functions between TDM Renderer and TDM Capture Interface.

Table 2-2 List of available functions between TDM Renderer and TDM Capture

	Name	TDM Renderer Interface	TDM Capture Interface
IL Core Method	OMX_Init (*)	O	O
	OMX_Deinit (**)	O	O
	OMX_GetHandle	O	O
	OMX_FreeHandle	O	O
	OMX_SetupTunnel	X	X
	OMX_TeardownTunnel	X	X
Component API	SendCommand	O	O
	GetParameter	O	O
	SetParameter	O	O
	GetState	O	O
	UseBuffer	X	X
	AllocateBuffer	O	O
	FreeBuffer	O	O
	EmptyThisBuffer	O	O
	FillThisBuffer (***)	X	O

Implementation:
O: Supported.
X: Not supported.

Note:

(*) OMX_Init function will be called only one time for the using of all the OpenMAX Media
component (OMX TDM Renderer, OMX TDM Capture).

(**) OMX_Deinit function will be called only one time for the using of all the OpenMAX Media
component (OMX TDM Renderer, OMX TDM Capture).

(***) OMX_FillThisBuffer function is not supported for TDM Renderer Interface due to the output
data will be output to speaker device.

2.2 The list of structures

Table 2-3 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 2-3 List of structures

Name	Outline
XAOMX_AUDIO_PARAM_TDM_RENDERER	The structure of parameters for OMX MC TDM Renderer
XAOMX_AUDIO_PARAM_TDM_CAPTURE	The structure of parameters for OMX MC TDM Capture

For more detail about OpenMAX IL Structures, please refer to OMX IL Specification 1.1.2, section
3.1 and section 4.1.

ADSP TDM Renderer/Capture Interface for Linux Application Note - TDM R
enderer/Capture -

2.3 Function specification

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. The OpenMAX™ IL core functions are ready to be used when this function returns successfully. (*)	
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

(*) OMX_Init shall be the first call made into OpenMAX IL and should be executed only one time
without an intervening OMX_Deinit call. It will be called only one time for the using of OMX MC TDM
Renderer, OMX MC TDM Capture.

2.3.1.2 OMX_Deinit

OMX_Deinit		
Synopsis	De-initializes OMX IL core, including its allocated memory and objects use to load/manage components. (*)	
Syntax	`OMX_API OMX_ERRORTYPE OMX_APIENTRY OMX_Deinit();`	
Parameter	None	
Return values	OMX_ErrorUndefined	Undefined error while processing command
	OMX_ErrorNone	Normal ends. De-initialize successfully

(*) OMX_Deinit should be the last call made into the OpenMAX IL core after all OpenMAX IL-related resources have been released. It will be called only one time for the using of OMX MC TDM Renderer, OMX MC TDM Capture.

ADSP TDM Renderer/Capture Interface for Linux Application Note - TDM R
enderer/Capture -

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 names for TDM Renderer and TDM Capture respectively are: "OMX.RENESAS.AUDIO.DSP.TDMRENDERER" "OMX.RENESAS.AUDIO.DSP.TDMCAPTURE"
	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 values	OMX_ErrorInvalidState	The proxy is not initialized.
	OMX_ErrorInsufficientResources	Failed to locate the component due to not enough resource
	OMX_ErrorInvalidComponentName	The component name parameter is invalid.
	OMX_ErrorNone	Normal ends. Get handle successfully

2.3.1.4　　OMX_FreeHandle

OMX_FreeHandle		
Synopsis	Free a handle allocated by the OMX_GetHandle method. The IL client should call OMX_FreeHandle only when the component is in the OMX_StateLoaded and 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 values	OMX_ErrorBadParameter	hComponent points to an invalid memory area.
	OMX_ErrorNone	Normal ends. Free handle successfully

2.3.1.5　　　OMX_SetupTunnel

OMX_SetupTunnel		
Synopsis	Handle the necessary calls to the components to setup the specified tunnel the two components. This method shall not be called unless the component is in the OMX_StateLoaded state except 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 as the destination for the tunnel.
	nPortInput	Select the destination port on component to be used in the tunnel
Return value	OMX_ErrorBadParameter	Both hOutput and hInput component point to invalid memory area.
	OMX_ErrorUndefined	Undefined error while processing command
	OMX_ErrorPortsNotCompatible	One or both components is a non-interop component and does not support tunneling.
	OMX_ErrorBadPortIndex	Port index is invalid.
	OMX_ErrorIncorrectStateOperation	Component is not in OMX_StateLoaded.
	OMX_ErrorNone	Normal ends. Setup tunnel successfully

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 are 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 are being used as the destination for the tunnel.
	nPortInput	Select the destination port on component being used in the tunnel
Return value	OMX_ErrorBadParameter	hOutput or hInput component points to invalid memory area.
	OMX_ErrorBadPortIndex	Port index is invalid.
	OMX_ErrorIncorrectStateOperation	Component is not in OMX_StateLoaded.
	OMX_ErrorNone	Normal ends. Teardown tunnel successfully

2.3.2 Component API

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_HANDLETYPE hComponent,`
	`    OMX_COMMANDTYPE Cmd,`
	`    OMX_U32 nParam1,`
	`    OMX_PTR pCmdData);`	

Parameter	hComponent	Pointer to memory area of component handle
	Cmd	Type of command. For more detail about type of command, please refer to OMX IL Specification 1.1.2, section 3.1.1.1.
	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 (mark buffer header).
Return value	OMX_ErrorBadParameter	Invalid command Invalid mark buffer area Invalid number of ports Destination state could not be recognized
	OMX_ErrorSameState	State transition is requested between same states.
	OMX_ErrorIncorrectStateTransition	Transition is incorrect.
	OMX_ErrorInvalidState	The current state is OMX_StateInvalid. The destination state is OMX_StateInvalid.
	OMX_ErrorNotImplemented	Don't support OMX_StatePause and OMX_StateWaitForResources
	OMX_ErrorInsufficientResources	Failed to initial codec setup due to not enough resource
	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	Get the current parameter settings from the component	

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

Parameter	hComponent	Pointer to memory area of component handle
	nParamIndex	It indicates which structure is requested from the component. This value is from the OMX_INDEXTYPE enumeration. Supported index are: OMX_IndexParamPortDefinition OMX_IndexParamAudioPortFormat OMX_IndexParamPriorityMgmt OMX_IndexParamAudioPcm OMX_IndexParamCompBufferSupplier XAOMX_IndexParamAudioTDMRenderer XAOMX_IndexParamAudioTDMCapture
	pComponentParameterStructure	A pointer to the IL client-allocated structure that the component fills. For OpenMAX IL parameters setting structure please refer OMX IL Specification 1.1.2, section 3.1 and section 4.1. For TDM Renderer and TDM Capture parameters setting structure: XAOMX_AUDIO_PARAM_TDM_RENDERER XAOMX_AUDIO_PARAM_TDM_
Return value	OMX_ErrorBadParameter	pParam points to invalid memory area.
	OMX_ErrorIncorrectStateOperation	Current state is OMX_StateInvalid.
	OMX_ErrorBadPortIndex	Port index of parameter is invalid.
	OMX_ErrorUnsupportedIndex	The index of parameter structure is not supported by component.
	OMX_ErrorNone	Normal ends. Getting parameter from component is successful.

RENESAS

2.3.2.3 OMX_SetParameter

OMX_SetParameter		
Synopsis	Send a parameter structure to a component	

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

Parameter	hComponent	Pointer to memory area of component handle
	nIndex	It indicates which structure is passed to the component. This value is from the OMX_INDEXTYPE enumeration. Supported index are: OMX_IndexParamPortDefinition OMX_IndexParamAudioPortFormat OMX_IndexParamPriorityMgmt OMX_IndexParamStandardComponentRole OMX_IndexParamAudioPcm OMX_IndexParamCompBufferSupplier XAOMX_IndexParamAudioTDMRenderer XAOMX_IndexParamAudioTDMCapture
	pComponentParameterStructure	A pointer to the IL client-allocated structure that the component fills. For OpenMAX IL parameters setting structure please refer OMX IL Specification 1.1.2, section 3.1 and section 4.1. For TDM Renderer and TDM Capture parameters setting structure: XAOMX_AUDIO_PARAM_TDM_RENDERER XAOMX_AUDIO_PARAM_TDM_CAPTURE
Return value	OMX_ErrorBadParameter	pParam points to 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 index of parameter structure is not supported by component.
	OMX_ErrorNone	Normal ends. Setting parameter to component is successful.

RENESAS

ADSP TDM Renderer/Capture Interface for Linux Application Note - TDM R
enderer/Capture -

2.3.2.4 OMX_GetState

OMX_GetState		
Synopsis	Return the current state of the component	

Syntax	OMX_ERRORTYPE OMX_GetState(
	    OMX_HANDLETYPE hComponent,
	    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	pState points to an invalid memory area.
	OMX_ErrorNone	Normal end. Getting the state of the component is successful.

ADSP TDM Renderer/Capture Interface for Linux Application Note - TDM R enderer/Capture -

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_HANDLETYPE hComponent,
	    OMX_BUFFERHEADERTYPE **ppBufHdr,
	    OMX_U32 nPortIndex,
	    OMX_PTR pAppPrivate,
	    OMX_U32 nSizeBytes,
	    OMX_U8 *pBuffer);	

Parameter	hComponent	Pointer to memory area of component handle
	**ppBufHdr	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	ppBufHdr points to an invalid memory area. Target port is invalid. Buffer size is not suitable.
	OMX_ErrorIncorrectStateOperation	Port is not populated.
	OMX_ErrorUndefined	Undefined error while processing command
	OMX_ErrorInsufficientResources	Failed to allocate the buffer due to lack of needed resources
	OMX_ErrorNone	Normal end. Setting the buffer to the target port is successful.

RENESAS

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_HANDLETYPE hComponent, 　　OMX_BUFFERHEADERTYPE **ppBufHdr, 　　OMX_U32 nPortIndex, 　　OMX_PTR pAppPrivate, 　　OMX_U32 nSizeBytes);	
Parameter	hComponent	Pointer to memory area of component handle
	**ppBufHdr	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	ppBufHdr points to an invalid memory area. Target port is invalid.
	OMX_ErrorInsufficientResources	Failed to allocate the buffer due to lack of needed resources
	OMX_ErrorIncorrectStateOperation	Port is not populated.
	OMX_ErrorUndefined	Undefined error while processing command
	OMX_ErrorNone	Normal end. Allocating the buffer is successful.

2.3.2.7 OMX_FreeBuffer

OMX_FreeBuffer		
Synopsis	De-allocate buffer structure	

Syntax	`OMX_ERRORTYPE OMX_FreeBuffer(`
	`    OMX_HANDLETYPE hComponent,`
	`    OMX_U32 nPortIndex,`
	`    OMX_BUFFERHEADERTYPE *pBufHdr);`	

Parameter	hComponent	Pointer to memory area of component handle
	nPortIndex	Target port (index into the port definition array of the component)
	*pBufHdr	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	pBufHdr 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.

RENESAS

ADSP TDM Renderer/Capture Interface for Linux Application Note - TDM Renderer/Capture -

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_HANDLETYPE hComponent,
	    OMX_BUFFERHEADERTYPE *pBufHdr);	

Parameter	hComponent	Pointer to memory area of component handle
	*pBufHdr	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	pBufHdr points to an invalid memory area. Input length is zero.
	OMX_ErrorVersionMismatch	OMX structure version is not compliance.
	OMX_ErrorBadPortIndex	Port index of buffer is invalid.
	OMX_ErrorIncorrectStateOperation	Input port is disable or busy. Component is not in OMX_StateExecuting. Receiving a buffer after end-of-stream has been reported.
	OMX_ErrorNone	Normal end. Buffer is transferred to the input port of a component successfully.

(*)This API only need to call once in TDM Capture Interface to start-up TDM Capture function.

RENESAS

ADSP TDM Renderer/Capture Interface for Linux Application Note - TDM R enderer/Capture -

2.3.2.9　　　OMX_FillThisBuffer

OMX_FillThisBuffer		
Synopsis	Send an empty buffer to an output port of a component then fill it with appropriate output data. (*)	

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

Parameter	hComponent	Pointer to memory area of component handle
	*pBufHdr	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	pBufHdr points to an invalid memory area.
	OMX_ErrorVersionMismatch	OMX structure version is not compliance.
	OMX_ErrorBadPortIndex	Port index of buffer is invalid.
	OMX_ErrorIncorrectStateOperation	Output port is disable or busy. Component is not in OMX_StateExecuting. Sending a buffer after end-of-stream has been reported.
	OMX_ErrorNone	Normal ends. Transferring buffer to client is successful.

(*) This API supports only TDM Capture interface.

ADSP TDM Renderer/Capture Interface for Linux Application Note - TDM R
enderer/Capture -

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 3-11 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 3.1.3.9

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 3.1.3.9.

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 3.1.3.9.

2.5 Structure specification

2.5.1 XAOMX_AUDIO_PARAM_TDM_RENDERER

To configure the input port of component, OMX MC TDM Renderer receives the
XAOMX_AUDIO_PARAM_TDM_RENDERER structure from user (with the index param
XAOMX_IndexParamAudioTDMRenderer). User can also obtain the information of input port by get
this structure from the component.

```
typedef struct XAOMX_AUDIO_PARAM_TDM_RENDERER
{
    OMX_U32                     nSize;
    OMX_VERSIONTYPE             nVersion;
    OMX_U32                     nPCM_frame_size;
    OMX_U32                     nPCM_channel_mode;
    OMX_U32                     nPCM_in_sample_rate;
    OMX_U32                     nPCM_out_sample_rate;
    OMX_U32                     nPCM_output1;
    OMX_U32                     nPCM_dma_channel1;
    OMX_U32                     nPCM_output2;
    OMX_U32                     nPCM_dma_channel2;
    OMX_U32                     nPCM_volume_rate;
} XAOMX_AUDIO_PARAM_TDM_RENDERER;
```

For more detail about OMX_VERSIONTYPE please refer to OMX IL Specification 1.1.2, section
3.1.2.4.

Table 2-4 shows the detail explanations of this structure. I/O column indicates the element is input
or output; Input Value column indicate the valid input value can be set from user.

ADSP TDM Renderer/Capture Interface for Linux Application Note - TDM R
enderer/Capture -

Table 2-4 Parameters Structure of TDM Renderer

Element	I/O	Input Value	Default	Description
nSize	O	None	The size of structure	Fixed by component
nVersion	O	None	1.1.2.0	OMX specification version information
nPCM_frame_size	I/O	512 / 1024 / 2048 (*)	1024	PCM frame size in sample
nPCM_channel_mode	I/O	0 (4 stereo stream), 1 (1 eight channel stream), 3 (3 stereo stream), 4 (1 six channel stream)	0	Set the channel mode of TDM plugin
nPCM_in_sample_rate	I/O	32,000/44,100/48,000 Hz	44100	Set the PCM input sampling rate
nPCM_out_sample_rate	I/O	0/32,000/44,100/48,000 Hz	0	Set the PCM output sampling rate. When SRC module is not used, this value should be 0
nPCM_output1	I/O	-SSI device: SSI00, SSI10, SSI20, SSI30, SSI40, SSI90 -SRC device: SCU_SRCI0, SCU_SRCI1, SCU_SRCI3, SCU_SRCI4 -No use: NONCONFIG	SSI00	Set the 1st output destination device
nPCM_dma_channel1	I/O	(ADMACPP_CH00 to ADMACPP_CH28) or (ADMAC_CH00 to ADMAC_CH31)	ADMACPP_CH00	Set the data transfer method control for the 1st output device
nPCM_output2	I/O	-SSI device: SSI00, SSI10, SSI20, SSI30, SSI40, SSI90 -SRC device: SCU_SRCI0, SCU_SRCI1, SCU_SRCI3, SCU_SRCI4 -No use: NONCONFIG	NONCONFIG	Set the 2nd output destination device
nPCM_dma_channel2	I/O	(ADMACPP_CH00 to ADMACPP_CH28) or (ADMAC_CH00 to ADMAC_CH31)	ADMACPP_CH01	Set the data transfer method control for the 2nd output device
nPCM_volume_rate	I/O	0xFFFFFFFF and (0 to 0x7FFFF)	0xFFFFFFFF	Set the volume control value. To disable volume control, this value is set to FFFFFFFF

Note:
(*) The CPU performance also influence to data transfer between user application and ADSP side.
So, in case TDM Renderer sets 512 for frame size to play multiple stream. If CPU cannot transfer
data to ADSP timely, the quality of output sound will be affected.

2.5.2 XAOMX_AUDIO_PARAM_TDM_CAPTURE

To configure the output port of component, OMX MC TDM Capture receives the
XAOMX_AUDIO_PARAM_TDM_CAPTURE structure from user (with the index param
XAOMX_IndexParamAudioTDMCapture). User can also obtain the information of output port by get
this structure from the component.

```
typedef struct XAOMX_AUDIO_PARAM_TDM_CAPTURE
{
    OMX_U32                         nSize;
    OMX_VERSIONTYPE                 nVersion;
    OMX_U32                         nPCM_frame_size;
    OMX_U32                         nPCM_channel_mode;
    OMX_U32                         nPCM_in_sample_rate;
    OMX_U32                         nPCM_out_sample_rate;
    OMX_U32                         nPCM_input1;
    OMX_U32                         nPCM_dma_channel1;
    OMX_U32                         nPCM_input2;
    OMX_U32                         nPCM_dma_channel2;
    OMX_U32                         nPCM_volume_rate;
} XAOMX_AUDIO_PARAM_TDM_CAPTURE;
```

For more detail about OMX_VERSIONTYPE please refer to OMX IL Specification 1.1.2, section
3.1.2.4.

Table 2-6 shows the detail explanations of this structure. I/O column indicates the element is input
or output; Input Value column indicate the valid input value can be set from user.

ADSP TDM Renderer/Capture Interface for Linux Application Note - TDM R
enderer/Capture -

Table 2-5 Parameters Structure of TDM Capture

Element	I/O	Input Value	Default	Description
nSize	O	None	The size of structure	Fixed by component
nVersion	O	None	1.1.2.0	OMX specification version information
nPCM_frame_size	I/O	512 / 1024 / 2048	1024	PCM frame size in sample
nPCM_channel_mode	I/O	0 (4 stereo stream), 1 (1 eight channel stream), 3 (3 stereo stream), 4 (1 six channel stream)	0	Set the channel mode of TDM plugin
nPCM_in_sample_rate	I/O	0/32,000/44,100/48,000 Hz	0	Set the PCM input sampling rate. When SRC module is not used, this value should be 0
nPCM_out_sample_rate	I/O	32,000/44,100/48,000 Hz	44100	Set the PCM output sampling rate.
nPCM_input1	I/O	+SSI device: SSI00, SSI10, SSI20, SSI30, SSI40, SSI90 -SRC device: SCU_SRCI0, SCU_SRCI1, SCU_SRCI3, SCU_SRCI4 -No use: NONCONFIG	SSI10	Set the 1st input destination device
nPCM_dma_channel1	I/O	(ADMACPP_CH00 to ADMACPP_CH28) or (ADMAC_CH00 to ADMAC_CH31)	ADMACPP_CH00	Set the data transfer method control for the 1st input device
nPCM_input2	I/O	-SSI device: SSI00, SSI10, SSI20, SSI30, SSI40, SSI90 -SRC device: SCU_SRCI0, SCU_SRCI1, SCU_SRCI3, SCU_SRCI4 -No use: NONCONFIG	NONCONFIG	Set the 2nd input destination device
nPCM_dma_channel2	I/O	(ADMACPP_CH00 to ADMACPP_CH28) or (ADMAC_CH00 to ADMAC_CH31)	ADMACPP_CH01	Set the data transfer method control for the 2nd input device
nPCM_volume_rate	I/O	0xFFFFFFFF and (0 to 0x7FFFF)	0xFFFFFFFF	Set the volume control value. To disable volume control, this value is set to FFFFFFFF

ADSP TDM Renderer/Capture Interface for Linux Application Note - TDM R enderer/Capture -

3 Process sequence

3.1 Initialize Component
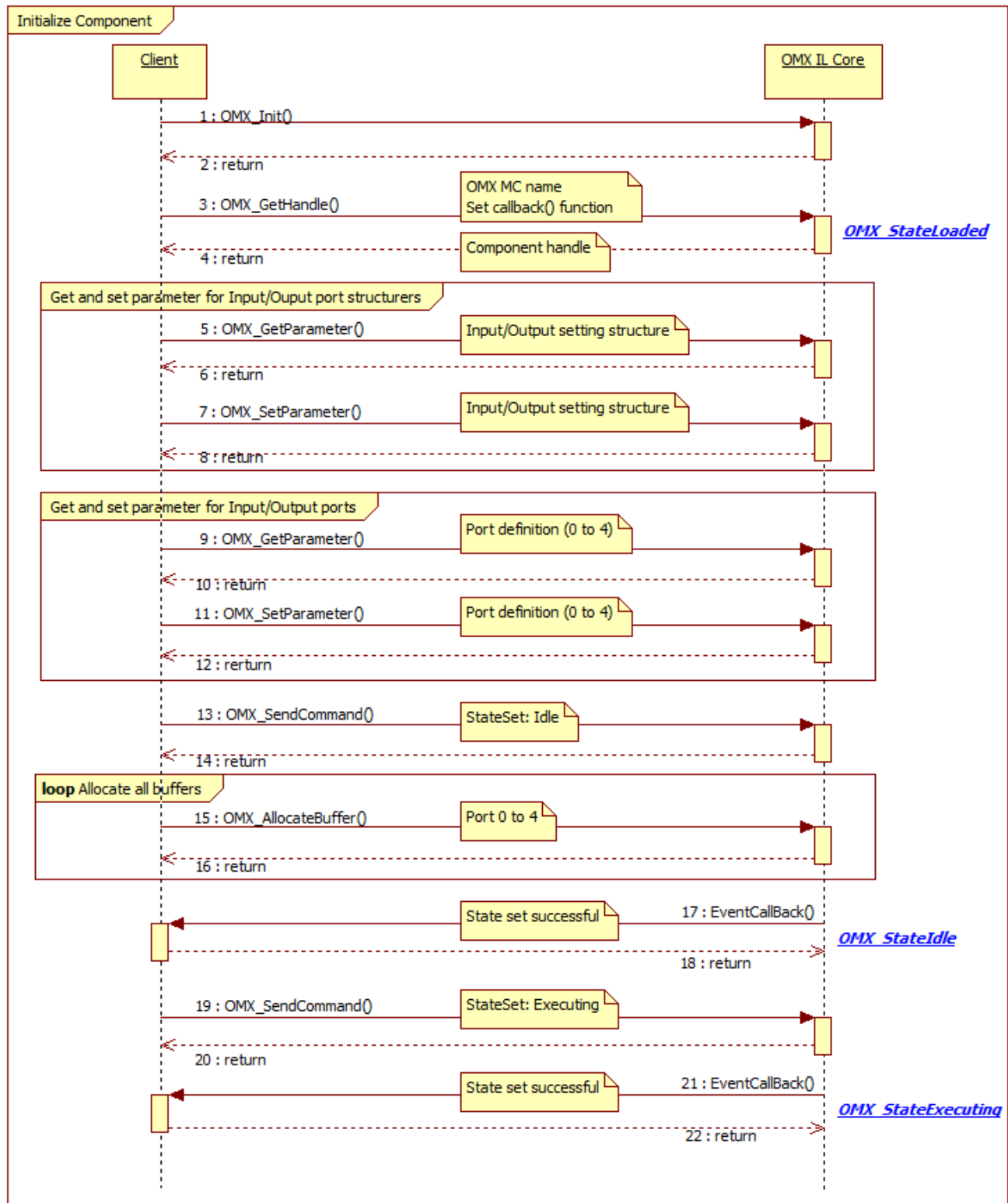
Figure 3-1 Initialize the Component and preparation phase

3.2 Decoding sequence

Figure 3-2 Decoding sequence

ADSP TDM Renderer/Capture Interface for Linux Application Note - TDM R enderer/Capture -
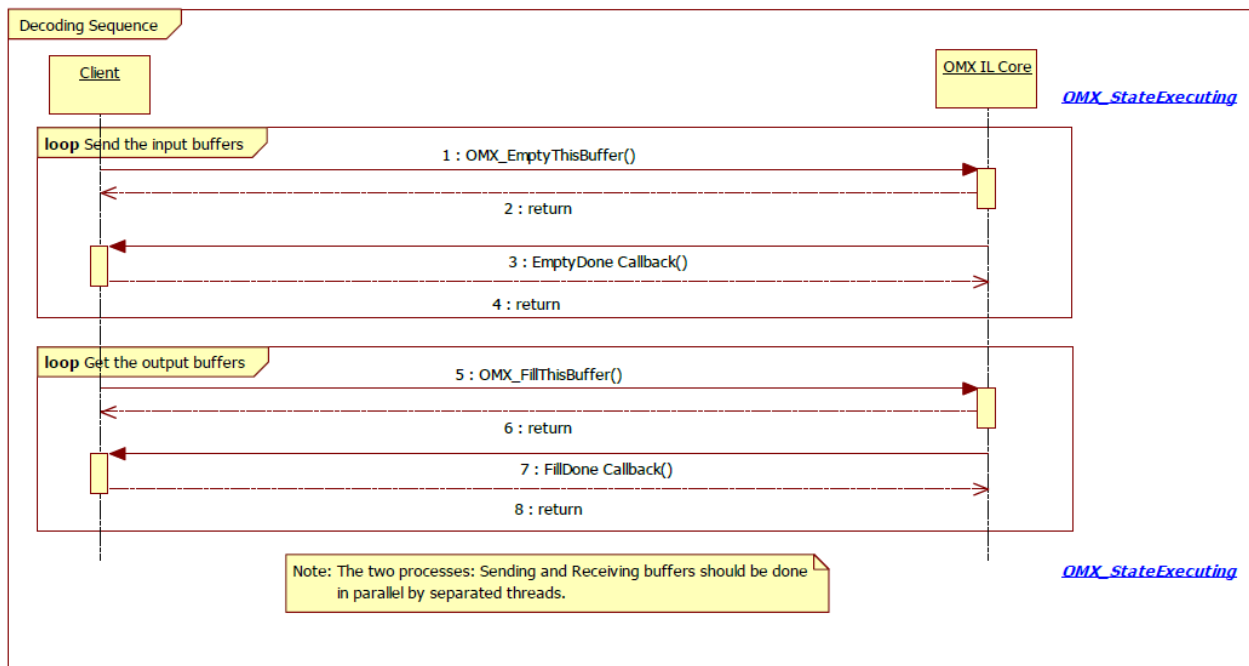
3.3 De-initialize Component
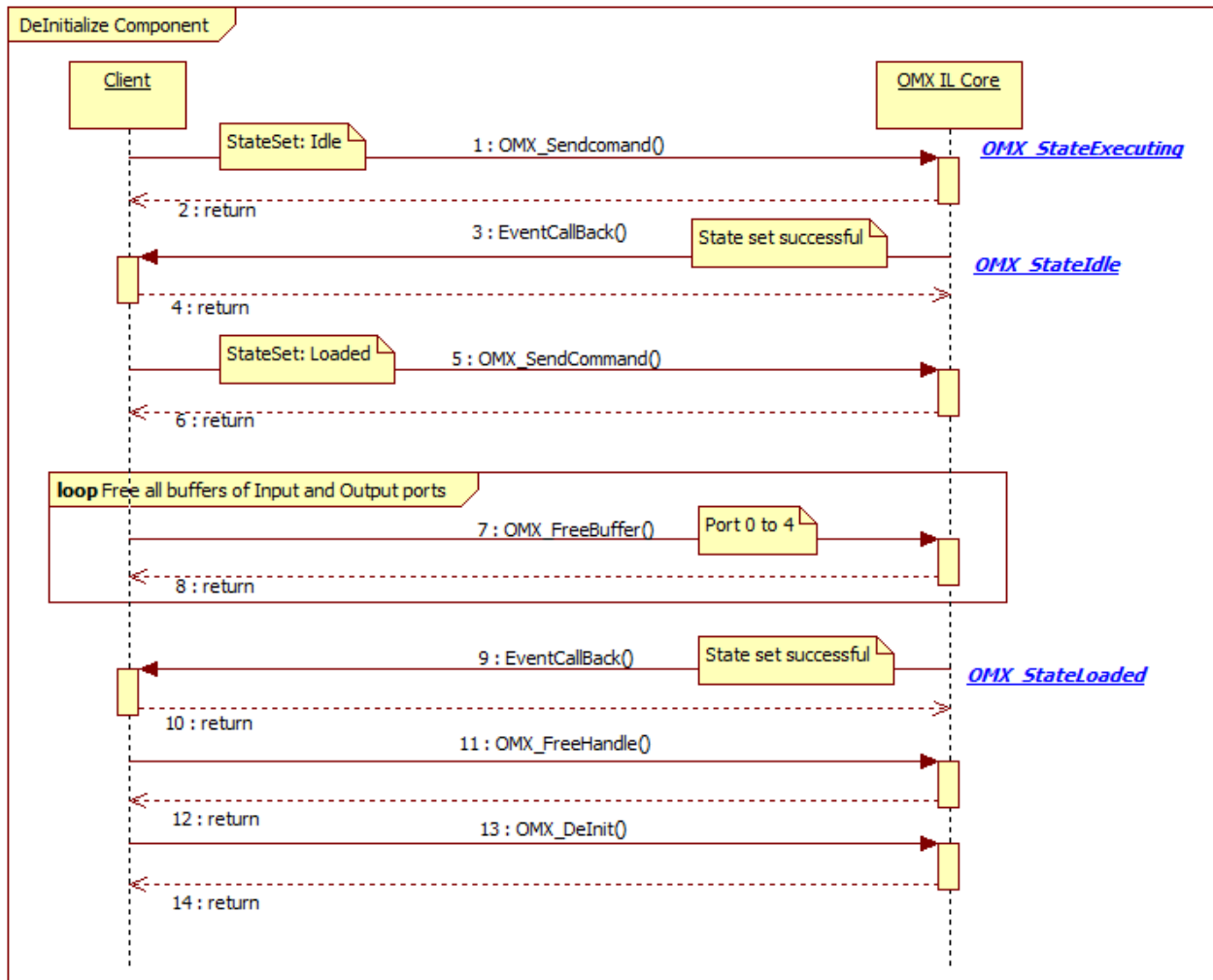
Figure 3-3 De-Initialize Component and OMX IL Core

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 - TDM Renderer/Capture -

Rev.	Date	Description	
		Page	Summary
0.10	Mar. 22, 2017	-	Preliminary Edition
0.11	Jun. 08, 2017	-	Update page number
		28, 30	Update valid range of parameters in table 2-4, table 2-6
0.12	Jul. 04, 2017	15, 16	Update the list of supported index of set/get parameter command
		18, 21, 22	Update return error description.
1.00	Jul. 04, 2017	-	Official Release

RENESAS

ルネサス エレクトロニクス株式会社

RENESAS

ADSP TDM Renderer/Capture Interface for Linux
RCG3AHIFL4001ZDP

Renesas Electronics Corporation