User's Manual

RENESAS

CONFIDENTIAL

ADSP Interface for Linux
RCG3AHIFL4001ZDP

Application Note - Renderer/Capture -

RCG3AHIFL4001ZDPE_AN_RDR

Rev. 1.00 Jul, 2017

Renesas Electronics
www.renesas.com

How to Use This Manual

1. Purpose and Target Reader

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

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

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

2. Restrictions on the Use of this Middleware

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

3. Related Manuals

4. Technical Terms and Abbreviation

- Table of Contents -

- List of Figures -

- List of Tables -

1 Overview

1.1 Overview of this document.

In this chapter, overview of Renderer/Capture 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 Renderer Plugin and ADSP Capture Plugin.

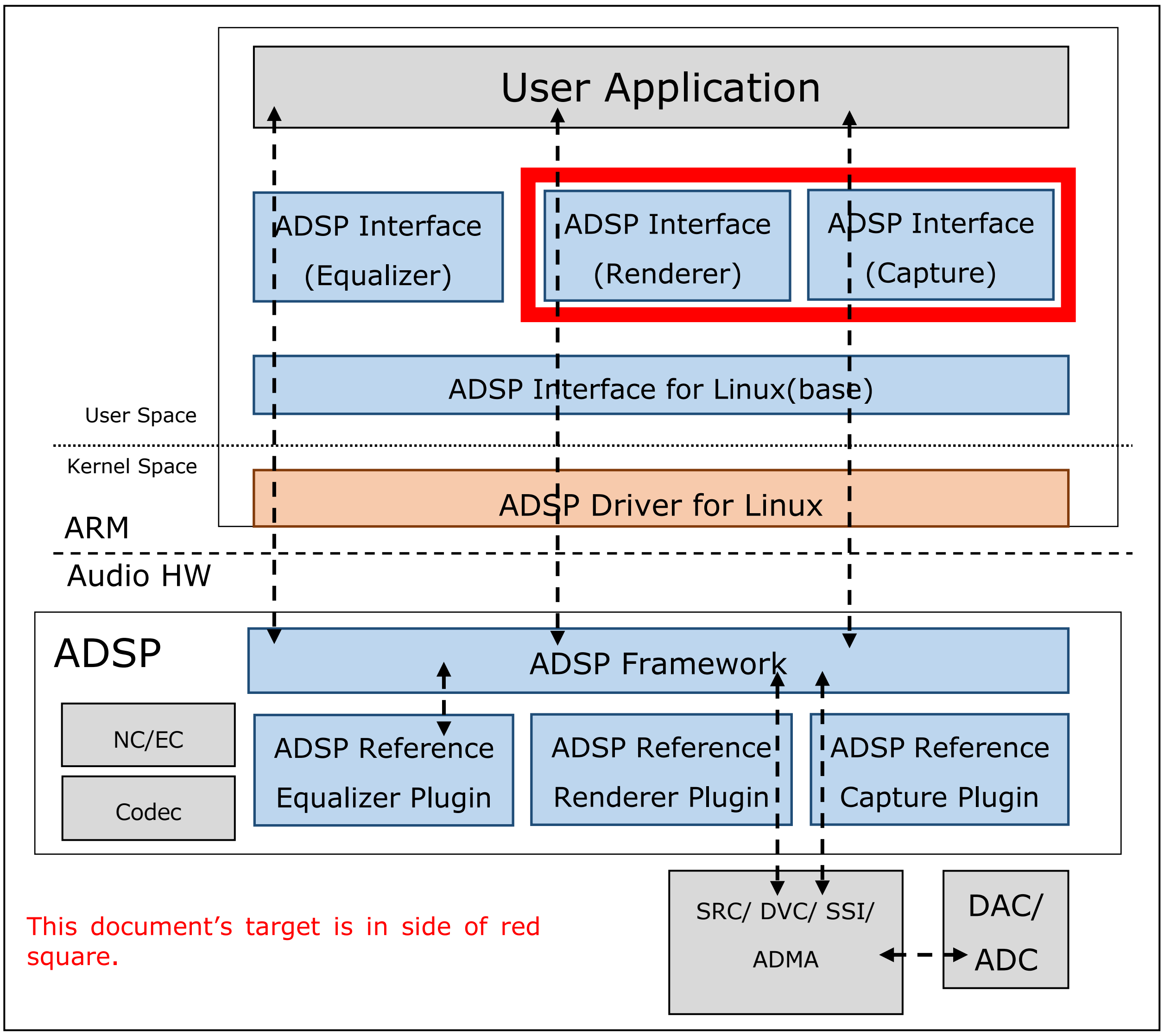
User Application
ADSP Interface (Equalizer)
ADSP Interface (Renderer)
ADSP Interface (Capture)
ADSP Interface for Linux(base)
User Space
Kernel Space
ADSP Driver for Linux
ARM
Audio HW
ADSP
ADSP Framework
NC/EC
Codec
ADSP Reference Equalizer Plugin
ADSP Reference Renderer Plugin
ADSP Reference Capture Plugin
SRC/ DVC/ SSI/ ADMA
DAC/ ADC
This document's target is in side of red square.

Figure 1-1 The software architecture

Note:

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

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

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 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

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

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

	Name	Renderer Interface	Capture Interface
IL Core Method	OMX_Init (*)	O	O
	OMX_Deinit (**)	O	O
	OMX_GetHandle	O	O
	OMX_FreeHandle	O	O
	OMX_SetupTunnel	O	O
	OMX_TeardownTunnel	O	O
Component API	SendCommand	O	O
	GetParameter	O	O
	SetParameter	O	O
	GetState	O	O
	UseBuffer	O	O
	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 Equalizer, OMX Renderer, OMX Capture).

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

(***) OMX_FillThisBuffer function is not supported for 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_RENDERER	The structure of parameters for OMX MC Renderer
XAOMX_AUDIO_PARAM_CAPTURE	The structure of parameters for OMX MC Capture

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

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 Equalizer, OMX MC Renderer and OMX MC 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 Equalizer, OMX MC Renderer and OMX MC 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 Renderer and Capture respectively are: “OMX.RENESAS.AUDIO.DSP.RENDERER” “OMX.RENESAS.AUDIO.DSP.CAPTURE”
	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_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. 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_IN OMX_HANDLETYPE hComponent,`
	`    OMX_IN OMX_INDEXTYPE nParamIndex,`
	`    OMX_INOUT 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 XAOMX_IndexParamAudioRenderer XAOMX_IndexParamAudioCapture
	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 Renderer and Capture parameters setting structure: XAOMX_AUDIO_PARAM_RENDERER XAOMX_AUDIO_PARAM_CAPTURE
Return value	OMX_ErrorBadParameter	pComponentParameterStructure 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.

2.3.2.3 OMX_SetParameter

OMX_SetParameter		
Synopsis	Send a parameter structure to a 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	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 XAOMX_IndexParamAudioRenderer XAOMX_IndexParamAudioCapture
	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 Renderer and Capture parameters setting structure: XAOMX_AUDIO_PARAM_RENDERER XAOMX_AUDIO_PARAM_CAPTURE
Return value	OMX_ErrorBadParameter	pComponentParameterStructure 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.

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	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_INOUT 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	ppBufferHdr 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_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	ppBuffer 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_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	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	pBuffer points to an invalid memory area. Input length is zero.
	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 Capture Interface to start-up Capture function.

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_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	pBuffer points to an invalid memory area.
	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 Capture interface.

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_RENDERER

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

```
typedef struct XAOMX_AUDIO_PARAM_RENDERER
{
    OMX_U32                          nSize;
    OMX_VERSIONTYPE                  nVersion;
    OMX_U32                          nPCM_frame_size;
    OMX_U32                          nPCM_output1;
    OMX_U32                          nPCM_DMAchannel1;
    OMX_U32                          nPCM_output2;
    OMX_U32                          nPCM_DMAchannel2;
    OMX_U32                          nPCM_in_sample_rate;
    OMX_U32                          nPCM_out_sample_rate;
    OMX_U32                          nPCM_volume_rate;
} XAOMX_AUDIO_PARAM_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.

Table 2-4 Parameters Structure of 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	256/512/1024 /2048/4096	1024	Set the PCM frame size
nPCM_output1 (**)	I/O	Expected output device 1	SSI00	Set the expected output device 1: +SSI device: SSIx0, x from 0 to 9 +SCU_SRC device: SCU_SRCI0 to SCU_SRCI9
nPCM_DMAchannel1	I/O	ADMAC channel / ADMACPP channel	ADMACPP_CH00	Set the data transfer method control for output device 1: +ADMACPP_CH00 to ADMACPP_CH28 +ADMAC_CH00 to ADMAC_CH31
nPCM_output2	I/O	Expected output device 2	NONCONFIG	Set the expected output device 2: +No device: NONCONFIG (If output1 is SSI device) +SSI device: SSIx0, x form 0 to 9 (If output1 is SCU_SRC device)
nPCM_DMAchannel2	I/O	ADMAC channel / ADMACPP channel	ADMACPP_CH01	Set the data transfer method control for output device 2: +ADMACPP_CH00 to ADMACPP_CH28 +ADMAC_CH00 to ADMAC_CH31
nPCM_in_sample_rate	I/O	Valid value: 32,000 / 44,100 / 48,000 Hz	44100	Set the PCM input sampling rate.
nPCM_out_sample_rate (***)	I/O	Valid value: 32,000 / 44,100 / 48,000 Hz	44100	Set the PCM output sampling rate.
nPCM_volume_rate (****)	I/O	0x00000000 to 0x007FFFFF	0xFFFFFFFF	Set the volume control value. To disable volume control, this value is set to 0xFFFFFFFF.

For detail about PCM stream structure of Renderer, the software supports the below settings in table 2-5.

The table 2-5 shows the detail explanations of PCM structure. I/O column indicates the element is input or output; Input Value column indicate the valid input value can be set from user.

Table 2-5 PCM stream setting of Renderer

Element	**I/O**	**Input Value**	**Default**	**Description**
nChannels	I/O	1 or 2 channels	2	Set channels of PCM stream. Supporting mode is 24bit-2channel, 16bit-2channel, 16bit-1channel
nBitPerSample	I/O	16 or 24 bits	16	Set the PCM width of PCM stream. Supporting mode is 24bit-2channel, 16bit-2channel, 16bit-1channel

Note:

(*): When two or more components are routing, framesize of each component must be the same. When two or more components are routing, framesize is only supported 256/512/1024.
When component is run independently, framesize is supported 256/512/1024/2048/4096.

(**): When module DVC is not used, SRC module range is form SRC0 to SRC9.
When module DVC is used, SRC module range is from SRC0 to SRC6, SRC9 (SRC7 and SRC8 are not available)

(***) When routing between Capture-Renderer and using SRC, In_fs of Capture and Out_fs of Renderer must be the same.

(****) Because hardware limitation has only 2 DVC modules. Therefore, user can only use maximum of 2 DVC control module.

2.5.2 XAOMX_AUDIO_PARAM_CAPTURE

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

```
typedef struct XAOMX_AUDIO_PARAM_CAPTURE
{
    OMX_U32                             nSize;
    OMX_VERSIONTYPE                     nVersion;
    OMX_U32                             nPCM_frame_size;
    OMX_U32                             nPCM_input1;
    OMX_U32                             nPCM_DMAchannel1;
    OMX_U32                             nPCM_input2;
    OMX_U32                             nPCM_DMAchannel2;
    OMX_U32                             nPCM_in_sample_rate;
    OMX_U32                             nPCM_out_sample_rate;
    OMX_U32                             nPCM_volume_rate;
} XAOMX_AUDIO_PARAM_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.

Table 2-6 Parameters Structure of 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	256/512/1024 /2048/4096	1024	Set the PCM frame size
nPCM_input1 (**)	I/O	Expected input device 1	SSI10	Set the expected input device 1: +SSI device: SSIx0, x from 0 to 9 +SCU_SRC device: SCU_SRCI0-SCU_SRCI9
nPCM_DMAchannel1	I/O	ADMAC channel / ADMACPP channel	ADMACPP_CH10	Set the data transfer method control for input device 1: +ADMACPP_CH00 to ADMACPP_CH28 +ADMAC_CH00 to ADMAC_CH31
nPCM_input2	I/O	Expected input device 2	NONCONFIG	Set the expected input device 2: +No device: NONCONFIG (If input1 is SSI device) +SSI device: SSIx0, x form 0 to 9 (If input1 is SCU_SRC device)
nPCM_DMAchannel2	I/O	ADMAC channel / ADMACPP channel	ADMACPP_CH02	Set the data transfer method control for input device 2: +ADMACPP_CH00 to ADMACPP_CH28 +ADMAC_CH00 to ADMAC_CH31
nPCM_in_sample_rate (***)	I/O	Valid value: 32,000 / 44,100 / 48,000 Hz	44100	Set the PCM input sampling rate.
nPCM_out_sample_rate	I/O	Valid value: 32,000 / 44,100 / 48,000 Hz	44100	Set the PCM output sampling rate.
nPCM_volume_rate (****)	I/O	0x00000000 to 0x007FFFFF	0xFFFFFFFF	Set the volume control value. To disable volume control, this value is set to 0xFFFFFFFF.

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

The table 2-7 shows the detail explanations of PCM structure. I/O column indicates the element is input or output; Input Value column indicate the valid input value can be set from user.

Table 2-7 PCM stream setting of Capture

Element	**I/O**	**Input Value**	**Default**	**Description**
nChannels	I/O	1 or 2 channels	2	Set channels of PCM stream. Supporting mode is 24bit-2channel, 16bit-2channel, 16bit-1channel
nBitPerSample	I/O	16 or 24 bits	16	Set the PCM width of PCM stream. Supporting mode is 24bit-2channel, 16bit-2channel, 16bit-1channel

Note:

(*): When two or more components are routing, framesize of each component must be the same.
When two or more components are routing, framesize is only supported 256/512/1024.
When component is run independently, framesize is supported 256/512/1024/2048/4096.

(**): When module DVC is not used, SRC module range is form SRC0 to SRC9.
When module DVC is used, SRC module range is from SRC0 to SRC6, SRC9 (SRC7 and SRC8 are not available)

(***) When routing between Capture-Renderer and using SRC, In_fs of Capture and Out_fs of Renderer must be the same.

(****) Because hardware limitation has only 2 DVC modules. Therefore, user can only use maximum of 2 DVC control module.

3 Process sequence

3.1 Initialize Component

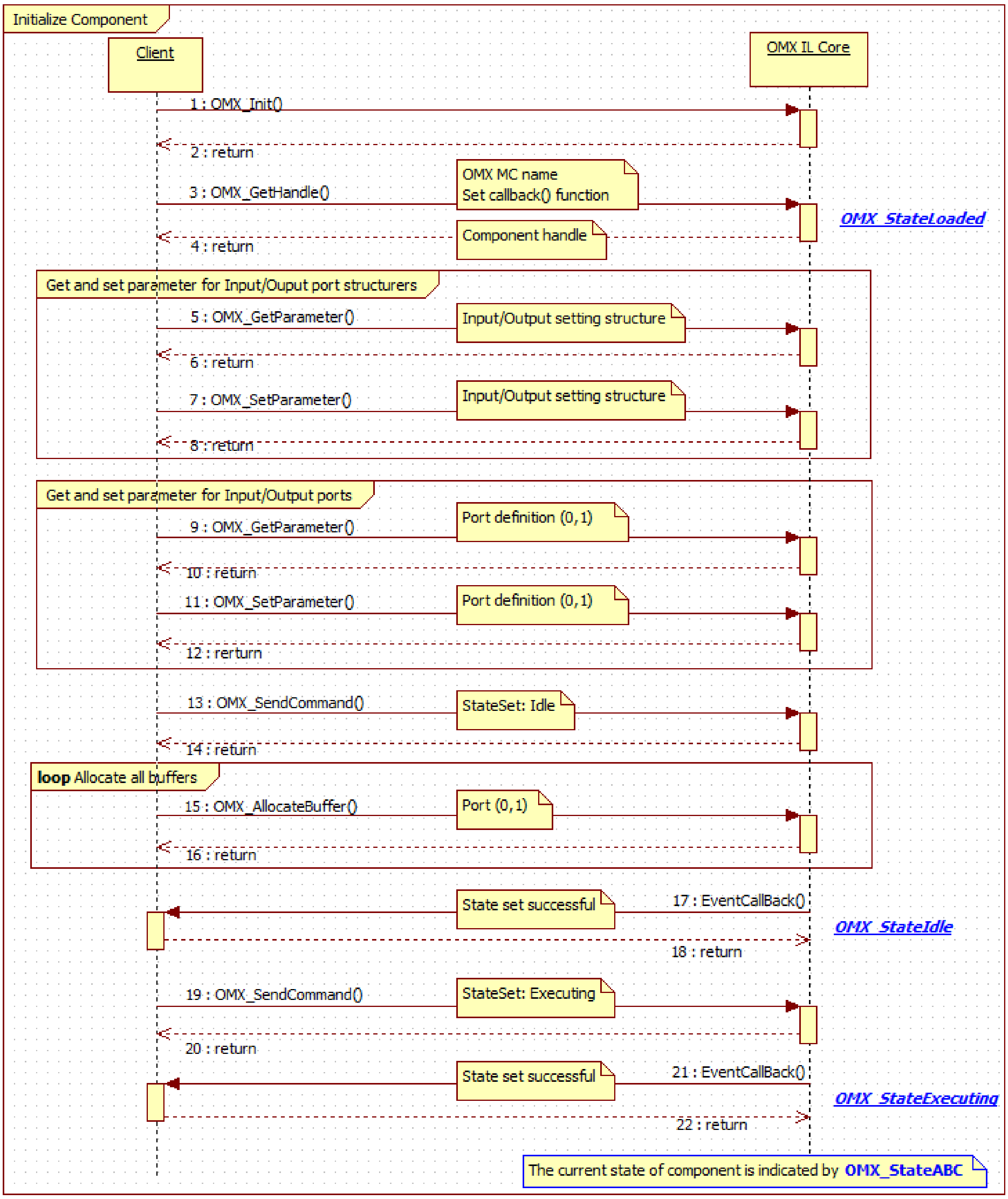

Figure 3-1 Initialize the Component and preparation phase

3.2 Decoding sequence

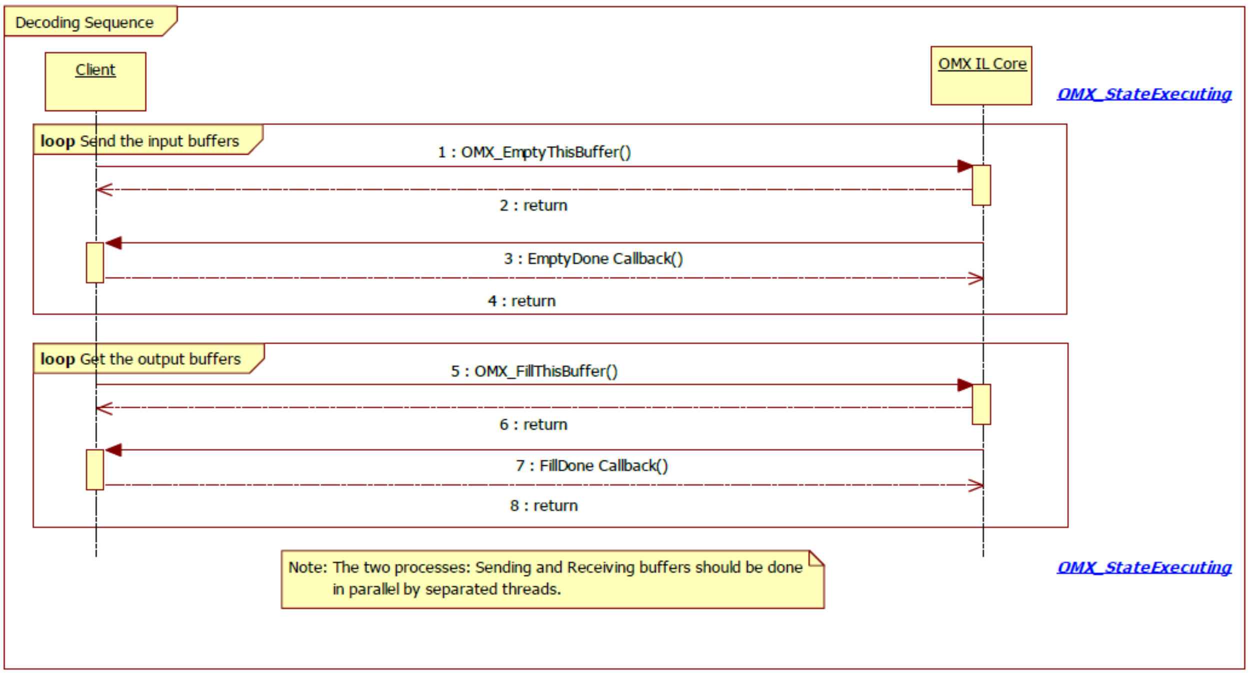
Decoding Sequence
Client
OMX IL Core
OMX_StateExecuting
loop Send the input buffers
1 : OMX_EmptyThisBuffer()
2 : return
3 : EmptyDone Callback()
4 : return
loop Get the output buffers
5 : OMX_FillThisBuffer()
6 : return
7 : FillDone Callback()
8 : return
Note: The two processes: Sending and Receiving buffers should be done in parallel by separated threads.
OMX_StateExecuting

Figure 3-2 Decoding sequence

3.3 De-initialize Component

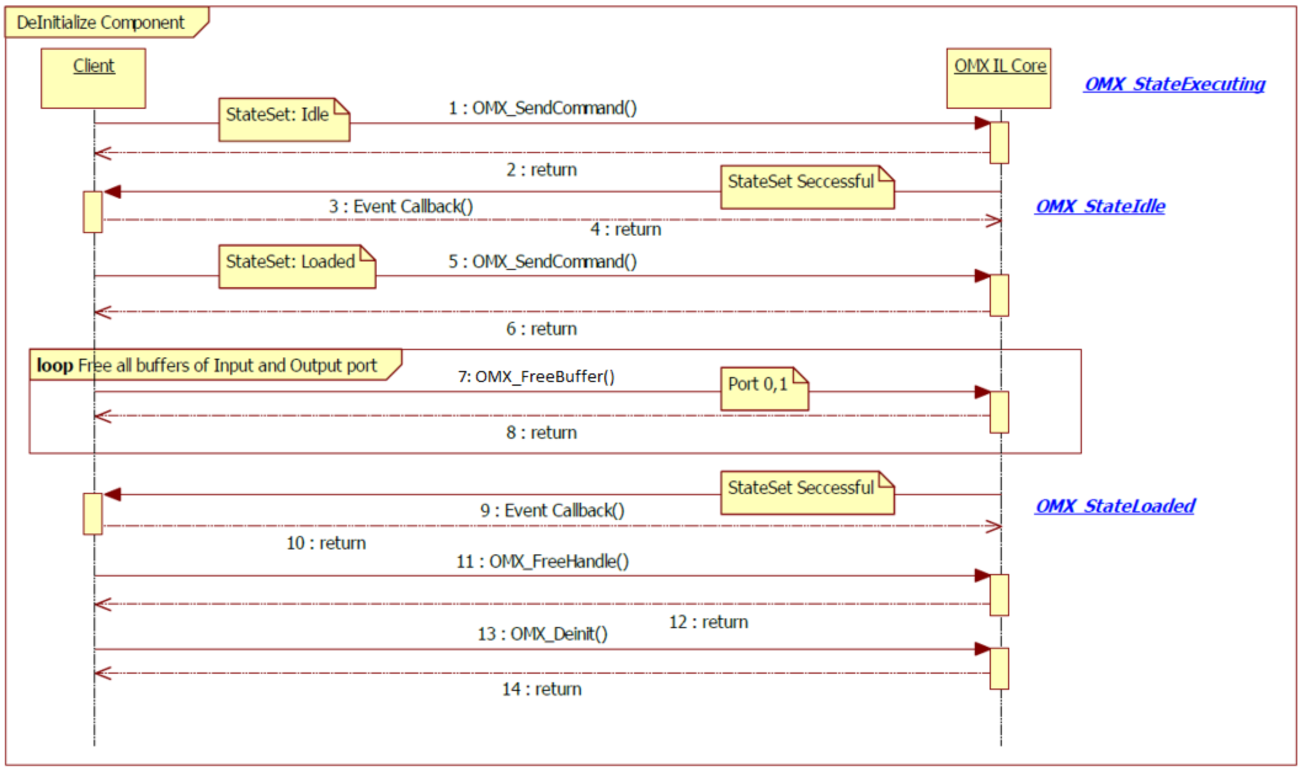

Figure 3-3 De-Initialize Component and OMX IL Core

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 - Renderer/Capture -

Rev.	Date	Description	
		Page	Summary
0.10	Aug. 04, 2016	-	Preliminary Edition
0.11	Nov. 03, 2016	-	Error correction
0.12	Mar. 15, 2017	-	Update for Extension
0.13	May. 31, 2017	28, 31	Update valid range of parameters for Extension (table 2-4 & table 2-6)
1.00	Jul. 06, 2017	-	Official Release

ADSP Interface for Linux Application Note - Renderer/Capture -

Publication Date: Jul 06, 2017 Rev. 1.00

Published by: Renesas Electronics Corporation

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

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: +44-1628-585-100, Fax: +44-1628-585-900

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, P. R. China 200333
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 II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.1

ADSP Interface for Linux

RCG3AHIFL4001ZDP

RCG3AHIFL4001ZDPE_AN_RDR

Renesas Electronics Corporation