- Table of Contents -

[1 Overview 3](#_Toc529974355)

[1.1 Overview of this document. 3](#_Toc529974356)

[1.2 The architecture of the Software and scope of this document 3](#_Toc529974357)

[1.3 Software necessary to be prepared in advance 4](#_Toc529974358)

[1.4 Related documents 4](#_Toc529974359)

[2 Software specification 5](#_Toc529974360)

[2.1 The list of functions 5](#_Toc529974361)

[2.2 The list of structures 7](#_Toc529974362)

[2.3 Function specification 8](#_Toc529974363)

[2.3.1 IL Core method 8](#_Toc529974364)

[2.3.1.1 OMX\_Init 8](#_Toc529974365)

[2.3.1.2 OMX\_Deinit 9](#_Toc529974366)

[2.3.1.3 OMX\_GetHandle 10](#_Toc529974367)

[2.3.1.4 OMX\_FreeHandle 11](#_Toc529974368)

[2.3.1.5 OMX\_SetupTunnel 12](#_Toc529974369)

[2.3.1.6 OMX\_TeardownTunnel 13](#_Toc529974370)

[2.3.2 Component API 14](#_Toc529974371)

[2.3.2.1 OMX\_SendCommand 14](#_Toc529974372)

[2.3.2.2 OMX\_GetParameter 15](#_Toc529974373)

[2.3.2.3 OMX\_SetParameter 16](#_Toc529974374)

[2.3.2.4 OMX\_GetState 17](#_Toc529974375)

[2.3.2.5 OMX\_UseBuffer 18](#_Toc529974376)

[2.3.2.6 OMX\_AllocateBuffer 19](#_Toc529974377)

[2.3.2.7 OMX\_FreeBuffer 20](#_Toc529974378)

[2.3.2.8 OMX\_EmptyThisBuffer 21](#_Toc529974379)

[2.3.2.9 OMX\_FillThisBuffer 22](#_Toc529974380)

[2.3.2.10 OMX\_SetConfig 23](#_Toc529974381)

[2.4 Callback function specification 24](#_Toc529974382)

[2.4.1 EventHandler 25](#_Toc529974383)

[2.4.2 EmptyBufferDone 26](#_Toc529974384)

[2.4.3 FillBufferDone 27](#_Toc529974385)

[2.5 Structure specification 28](#_Toc529974386)

[2.5.1 XAOMX\_AUDIO\_PARAM\_RENDERER 28](#_Toc529974387)

[2.5.2 XAOMX\_AUDIO\_PARAM\_CAPTURE 31](#_Toc529974388)

[3 Process sequence 34](#_Toc529974389)

[3.1 Initialize Component 34](#_Toc529974390)

[3.2 Decoding sequence 35](#_Toc529974391)

[3.3 De-initialize Component 36](#_Toc529974392)

[4 Notes 37](#_Toc529974393)

[4.1 Function Call 37](#_Toc529974394)

[4.2 Other notes 37](#_Toc529974395)

[4.2.1 Allocation of memory 37](#_Toc529974396)

[4.2.2 Out of range memory access 37](#_Toc529974397)

[4.2.3 Combination with other applications 37](#_Toc529974398)

[4.2.4 Monitoring on Performance 37](#_Toc529974399)

- List of Figures -

[Figure 1‑1 The software architecture 3](#_Toc529974342)

[Figure 3‑1 Initialize the Component and preparation phase 34](#_Toc529974343)

[Figure 3‑2 Decoding sequence 35](#_Toc529974344)

[Figure 3‑3 De-Initialize Component and OMX IL Core 36](#_Toc529974345)

- List of Tables -

[Table 1‑1 Supported features for OMX Interface 4](#_Toc529974346)

[Table 1‑2 The list of related documents 4](#_Toc529974347)

[Table 2‑1 List of functions 5](#_Toc529974348)

[Table 2‑2 List of available functions between Renderer and Capture 6](#_Toc529974349)

[Table 2‑3 List of structures 7](#_Toc529974350)

[Table 2‑4 Parameters Structure of Renderer 29](#_Toc529974351)

[Table 2‑5 PCM stream setting of Renderer 30](#_Toc529974352)

[Table 2‑6 Parameters Structure of Capture 32](#_Toc529974353)

[Table 2‑7 PCM stream setting of Capture 33](#_Toc529974354)

ADSP Interface for Linux

RCG3AHIFL4001ZDPE\_AN\_RDR

Rev. 1.00

Jul. 06, 2017

# Overview

## Overview of this document.

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

## The architecture of the Software and scope of this document

The architecture of is shown in Figure 1‑1. is a user space library which provides the interface to control ADSP Renderer Plugin and ADSP Capture Plugin.

ARM

User Application

ADSP

Codec

DAC/ADC

ARM

Audio HW

SRC/CTU/MIX/ DVC/ SSI/

ADMA

ADSP Interface for Linux(base)

NC/EC

ADSP Driver for Linux

ADSP Framework

ADSP Reference Renderer Plugin

ADSP Reference Equalizer Plugin

User Space

ADSP Interface (Capture)

Kernel Space

This document’s target is in side of red square.

ADSP Interface (Equalizer)

ADSP Interface (Renderer)

ADSP Reference Capture Plugin

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.

Table 1‑1 shows supported features for OMX CAPTURE and OMX RENDERER

Table 1‑1 Supported features for OMX Interface

|  |  |  |
| --- | --- | --- |
| Name | Renderer Interface | Capture Interface |
| ADMAC | O | O |
| ADMACPP | O | O |
| SSIU/SSI | O | O |
| SRC | O | O |
| DVC | O | O |
| CTU | O | X |
| MIX | O | X |

Implementation*:*

O: Supported.

X: Not supported.

Note: ADSP Capture Plugin does not support CTU and MIX functionalities.

## Software necessary to be prepared in advance

should be loaded in advance to use .

## Related documents

Table 1‑2 shows related documents.

Table 1‑2 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 |

# Software specification

## 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 OMX component |
| OMX\_SetParameter | Setup the parameter to the OMX 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 |
| OMX\_SetConfig | Set ADSP Plugin-in configuration value |

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 | OMX\_SendCommand | O | O |
| OMX\_GetParameter | O | O |
| OMX\_SetParameter | O | O |
| OMX\_GetState | O | O |
| OMX\_UseBuffer | O | O |
| OMX\_AllocateBuffer | O | O |
| OMX\_FreeBuffer | O | O |
| OMX\_EmptyThisBuffer | O | O |
| OMX\_FillThisBuffer (\*\*\*) | X | O |
| OMX\_SetConfig | O | 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.

## 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.

## Function specification

### IL Core method

#### 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.

#### 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.

#### OMX\_GetHandle

FD\_API\_RDR\_001

|  |  |  |
| --- | --- | --- |
| 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 |

[Covers: RD\_006, RD\_007]

#### OMX\_FreeHandle

FD\_API\_RDR\_002

|  |  |  |
| --- | --- | --- |
| 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 |

[Covers: RD\_006, RD\_007]

#### 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 are 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 |

#### 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 |

### Component API

#### OMX\_SendCommand

FD\_API\_RDR\_003

|  |  |  |
| --- | --- | --- |
| 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 | The transition is invalid such as changing from OMX\_StateExecuting to OMX\_StatePause, etc. |
| 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. |

[Covers: RD\_006, RD\_007, RD\_015]

#### OMX\_GetParameter

FD\_API\_RDR\_004

|  |  |  |
| --- | --- | --- |
| OMX\_GetParameter | | |
| Synopsis | Get the current parameter settings from the OMX 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. |

[Covers: RD\_006, RD\_007]

#### OMX\_SetParameter

FD\_API\_RDR\_005

|  |  |  |
| --- | --- | --- |
| OMX\_SetParameter | | |
| Synopsis | Send a parameter structure to a OMX 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. |

[Covers: RD\_006, RD\_007]

#### 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. |

#### 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. |

#### OMX\_AllocateBuffer

|  |  |  |
| --- | --- | --- |
| OMX\_AllocateBuffer | | |
| Synopsis | Allocate the buffer and the buffer header, request mapping the buffer with data buffer in ADSP 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. |

#### 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. |

#### 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.

#### 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.

#### OMX\_SetConfig

FD\_API\_RDR\_006

|  |  |  |
| --- | --- | --- |
| OMX\_SetConfig | | |
| Synopsis | Set parameter to ADSP Plugin when this command is called. This command can be called in any OMX\_State except OMX\_StateInvalid. This function is only support for setting volume rate parameter. | |
| Syntax | OMX\_ERRORTYPE SetConfig(  OMX\_IN OMX\_HANDLETYPE hComponent,  OMX\_IN OMX\_INDEXTYPE nIndex,  OMX\_IN OMX\_PTR pComponentConfigStructure) | |
| 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:  XAOMX\_IndexParamAudioRenderer  XAOMX\_IndexParamAudioCapture |
| pComponentConfigStructure | A pointer to the IL client-allocated structure that the component fills.  For Renderer and Capture parameters setting structure:  XAOMX\_AUDIO\_PARAM\_RENDERER  XAOMX\_AUDIO\_PARAM\_CAPTURE  This function is only support for volume rate parameter in the setting structure. |
| Return value | OMX\_ErrorBadParameter | pComponentConfigStructure points to an invalid memory area. |
| OMX\_ErrorIncorrectStateOperation | Component is in OMX\_StateInvalid |
| OMX\_ErrorUnsupportedIndex | The index of parameter structure is not supported by component. |
| OMX\_ErrorNone | Normal ends. Transferring buffer to client is successful. |

[Covers: RD\_006, RD\_007]

## 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;

### 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

### 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.

### 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.

## Structure specification

### 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;

OMX\_U32 nPCM\_in\_channel;

OMX\_U32 nPCM\_out\_channel;

OMX\_U32 nPCM\_mix\_control;

} 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 | 1024/2048 | 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. |
| nPCM\_in\_channel | I/O | Valid value: 1/2/4/6/8 | 2 | Set the number of PCM input channels |
| nPCM\_out\_channel | I/O | Valid value: 1/2 | 2 | Set the number of PCM output channels |
| nPCM\_mix\_control (\*\*\*) | I/O | Valid value: 0/1 | 0 | Set mixer control flag. (0: mixer disable, 1: mixer enable) |

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** |
| nBitPerSample | I/O | 16 or 24 bits | 16 | Set the PCM width of PCM stream. Supporting mode (nBitPerSample - nPCM\_out\_channel) is 24bit-2channel, 16bit-2channel, 16bit-1channel. |

Note:

(\*) Frame size is 1024 is the best performance. The value 2048 does not guarantee the performance of Renderer plugin.

(\*\*) 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 two CMD modules.

Therefore, user can only use volume control or mixing stream up to maximum two different SSI output.

### 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 | 1024/2048 | 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 (nBitPerSample – nChannels) is 24bit-2channel, 16bit-2channel, 16bit-1channel |
| nBitPerSample | I/O | 16 or 24 bits | 16 | Set the PCM width of PCM stream. Supporting mode (nBitPerSample – nChannels) is 24bit-2channel, 16bit-2channel, 16bit-1channel |

Note:

(\*) Frame size is 1024 is the best performance. The value 2048 does not guarantee the performance of Capture plugin.

(\*\*) 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 two CMD modules.

Therefore, user can only use volume control up to maximum two different SSI output.

# Process sequence

## Initialize Component

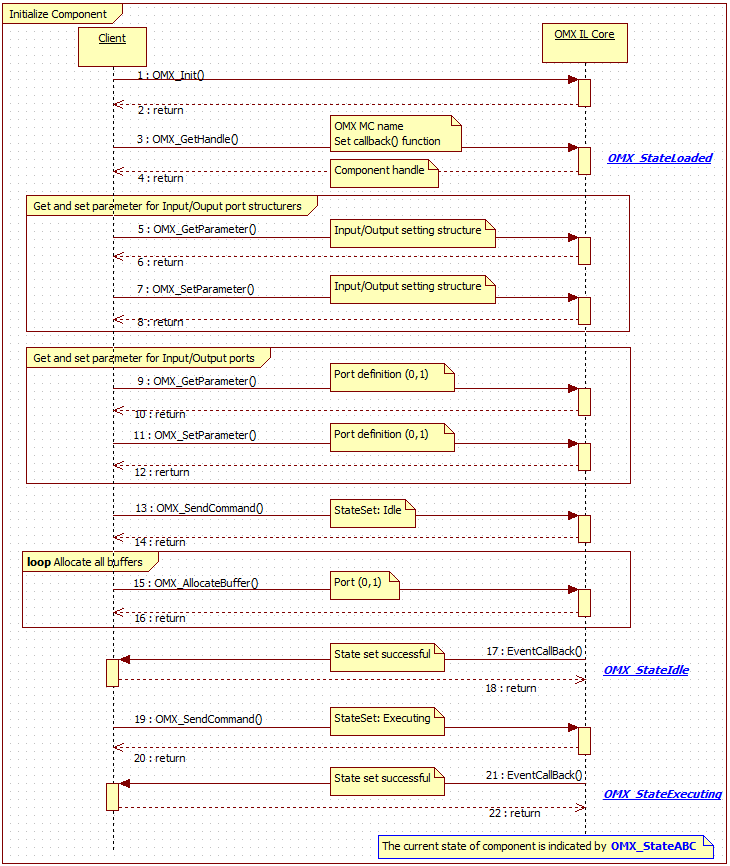


Figure 3‑1 Initialize the Component and preparation phase

## Decoding sequence

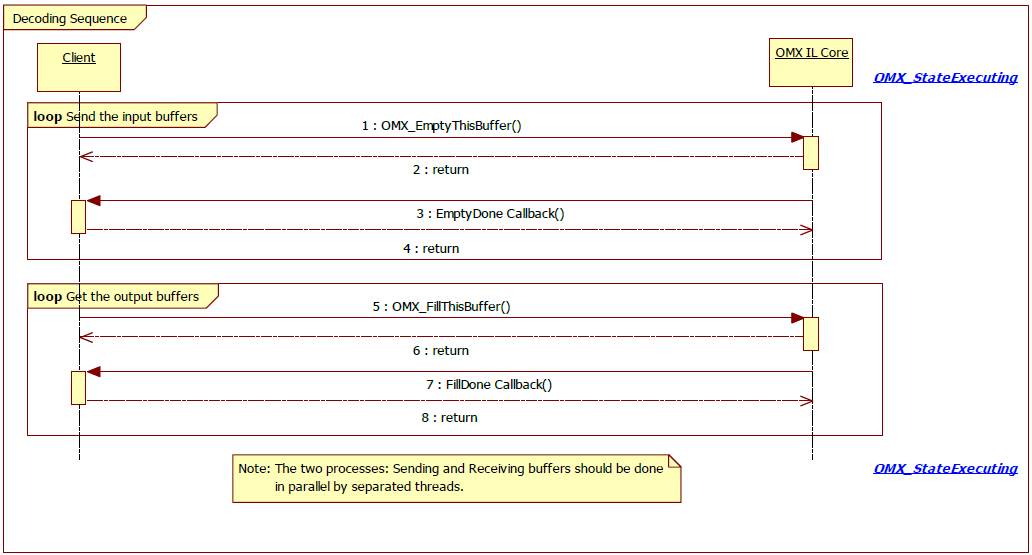


Figure 3‑2 Decoding sequence

## De-initialize Component

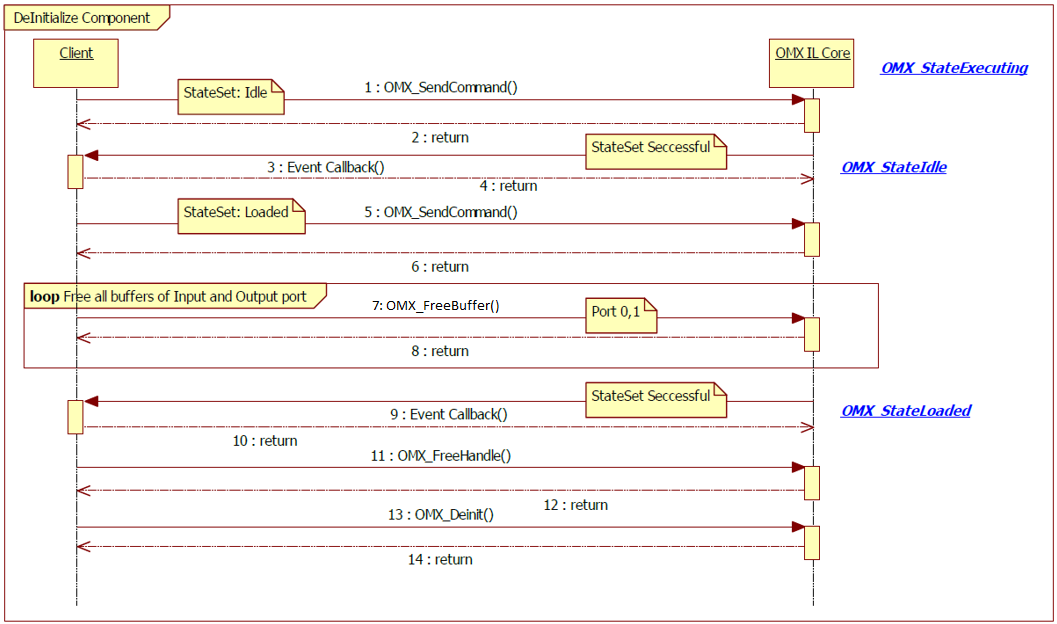


Figure 3‑3 De-Initialize Component and OMX IL Core

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

# Notes

This section describes the notice of developing user programs.

## Function Call

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

## Other notes

### Allocation of memory

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

### Out of range memory access

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

### Combination with other applications

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

### Monitoring on Performance

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