- Table of Contents -

[1 Overview 3](#_Toc529973355)

[1.1 Overview of this document. 3](#_Toc529973356)

[1.2 The architecture of the Software and scope of this document 3](#_Toc529973357)

[1.3 Software necessary to be prepared in advance 4](#_Toc529973358)

[1.4 Related documents 4](#_Toc529973359)

[2 Software specification 5](#_Toc529973360)

[2.1 The list of functions 5](#_Toc529973361)

[2.2 The list of structures 7](#_Toc529973362)

[2.3 Function specification 8](#_Toc529973363)

[2.3.1 IL Core method 8](#_Toc529973364)

[2.3.1.1 OMX\_Init 8](#_Toc529973365)

[2.3.1.2 OMX\_Deinit 9](#_Toc529973366)

[2.3.1.3 OMX\_GetHandle 10](#_Toc529973367)

[2.3.1.4 OMX\_FreeHandle 11](#_Toc529973368)

[2.3.2 Component API 12](#_Toc529973369)

[2.3.2.1 OMX\_SendCommand 12](#_Toc529973370)

[2.3.2.2 OMX\_GetParameter 13](#_Toc529973371)

[2.3.2.3 OMX\_SetParameter 14](#_Toc529973372)

[2.3.2.4 OMX\_GetState 15](#_Toc529973373)

[2.3.2.5 OMX\_AllocateBuffer 16](#_Toc529973374)

[2.3.2.6 OMX\_FreeBuffer 17](#_Toc529973375)

[2.3.2.7 OMX\_EmptyThisBuffer 18](#_Toc529973376)

[2.3.2.8 OMX\_FillThisBuffer 19](#_Toc529973377)

[2.4 Callback function specification 20](#_Toc529973378)

[2.4.1 EventHandler 21](#_Toc529973379)

[2.4.2 EmptyBufferDone 22](#_Toc529973380)

[2.4.3 FillBufferDone 23](#_Toc529973381)

[2.5 Structure specification 24](#_Toc529973382)

[2.5.1 XAOMX\_AUDIO\_PARAM\_TDM\_RENDERER 24](#_Toc529973383)

[2.5.2 XAOMX\_AUDIO\_PARAM\_TDM\_CAPTURE 26](#_Toc529973384)

[3 Process sequence 28](#_Toc529973385)

[3.1 Initialize Component 28](#_Toc529973386)

[3.2 Decoding sequence 29](#_Toc529973387)

[3.3 De-initialize Component 30](#_Toc529973388)

[4 Notes 31](#_Toc529973389)

[4.1 Function Call 31](#_Toc529973390)

[4.2 Other notes 31](#_Toc529973391)

[4.2.1 Allocation of memory 31](#_Toc529973392)

[4.2.2 Out of range memory access 31](#_Toc529973393)

[4.2.3 Combination with other applications 31](#_Toc529973394)

[4.2.4 Monitoring on Performance 31](#_Toc529973395)

- List of Figures -

[Figure 1‑1 The software architecture 3](#_Toc529973396)

[Figure 3‑1 Initialize the Component and preparation phase 28](#_Toc529973397)

[Figure 3‑2 Decoding sequence 29](#_Toc529973398)

[Figure 3‑3 De-Initialize Component and OMX IL Core 30](#_Toc529973399)

- List of Tables -

[Table 1‑1 The list of related documents 4](#_Toc529973400)

[Table 2‑1 List of functions 5](#_Toc529973401)

[Table 2‑2 List of available functions between TDM Renderer and TDM Capture 6](#_Toc529973402)

[Table 2‑3 List of structures 7](#_Toc529973403)

[Table 2‑4 Parameters Structure of TDM Renderer 25](#_Toc529973404)

[Table 2‑5 Parameters Structure of TDM Capture 27](#_Toc529973405)

ADSP TDM Renderer/Capture Interface for Linux

RCG3AHIFL4001ZDPE\_AN\_RDR

Rev. 1.00

Jul. 04, 2017

# Overview

## Overview of this document.

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

ADSP

ARM

User Application

Codec

DAC/ADC

ARM

Audio HW

SCU/ SSI/ ADMA

ADSP Interface for Linux (base)

NC/EC

ADSP Driver for Linux

ADSP Framework

ADSP Reference TDM Renderer Plugin

User Space

ADSP Interface (TDM Capture)

Kernel Space

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

ADSP Interface (TDM Renderer)

ADSP Reference 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.

## Software necessary to be prepared in advance

should be loaded in advance to use .

## 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 |

# 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) |
| 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\_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 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.

TDM does not support routing function. Therefore, OMX\_SetupTunnel, OMX\_TeardownTunnel and UseBuffer are unsupported.

## 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.

## 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 TDM Renderer, OMX MC TDM 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 TDM Renderer, OMX MC TDM Capture.

#### OMX\_GetHandle

FD\_API\_TDM\_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 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 |

[Covers: RD\_009]

#### OMX\_FreeHandle

FD\_API\_TDM\_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\_009]

### Component API

#### OMX\_SendCommand

FD\_API\_TDM\_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\_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 | 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\_009]

#### OMX\_GetParameter

FD\_API\_TDM\_004

|  |  |  |
| --- | --- | --- |
| 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\_CAPTURE |
| 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. |

[Covers: RD\_009]

#### OMX\_SetParameter

FD\_API\_TDM\_005

|  |  |  |
| --- | --- | --- |
| 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. |

[Covers: RD\_009]

#### 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. |

#### 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. |

#### 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. |

#### 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.

#### 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.

## 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\_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.

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 | 0/48,000/44,100 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 0x7FFFFF) | 0xFFFFFFFF | Set the volume control value. To disable volume control, this value is set to FFFFFFFF |

Note:

(\*): Frame size is 1024 is the best performance. Frame size are 512 or 2048 not guarantee the performance of TDM Renderer plugin.

### 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.

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 | 0/48,000/44,100 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 0x7FFFFF) | 0xFFFFFFFF | Set the volume control value. To disable volume control, this value is set to FFFFFFFF |

Note:

(\*): Frame size is 1024 is the best performance. Frame size are 512 or 2048 not guarantee the performance of TDM Capture plugin.

# Process sequence

## Initialize Component

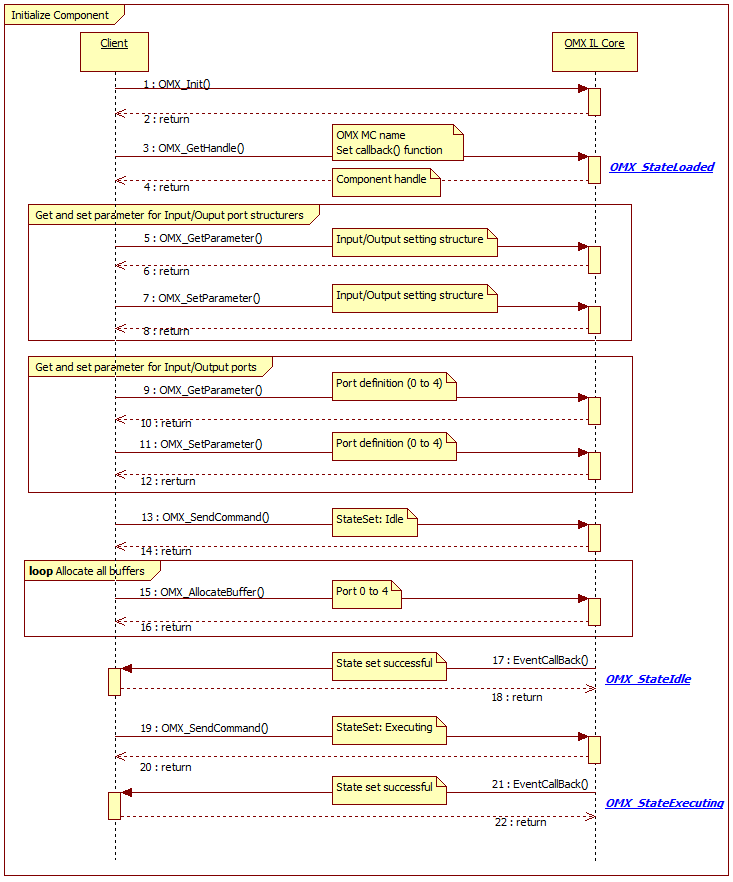


Figure 3‑1 Initialize the Component and preparation phase

## Decoding sequence

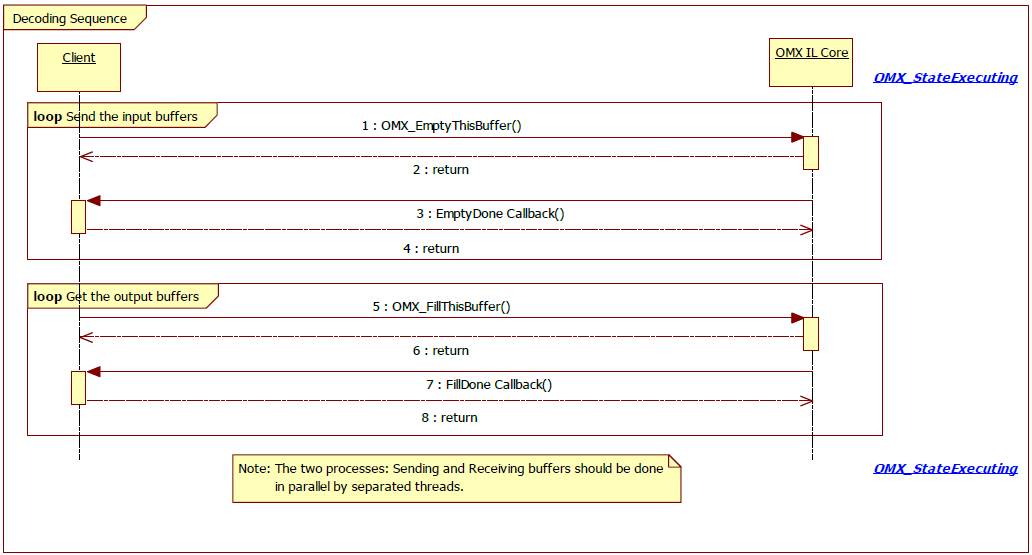


Figure 3‑2 Decoding sequence

## De-initialize Component

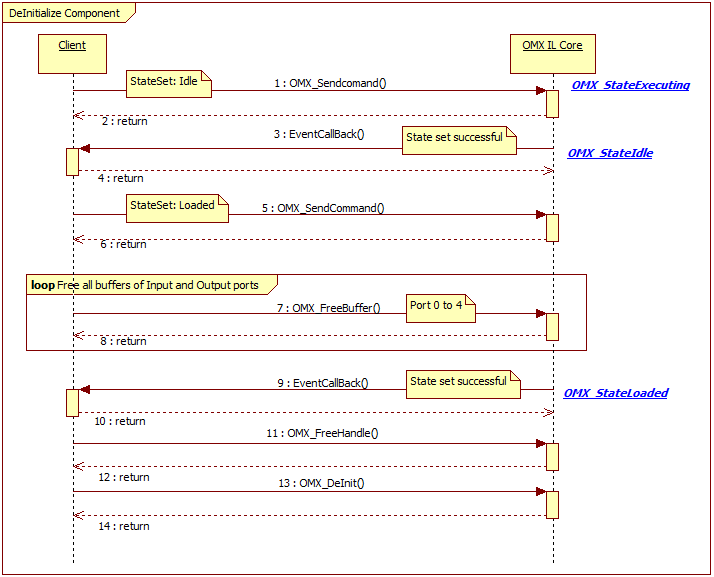


Figure 3‑3 De-Initialize Component and OMX IL Core

*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.