- Table of Contents -

[1 Overview 3](#_Toc529975038)

[1.1 Overview of this document. 3](#_Toc529975039)

[1.2 The architecture of the Software and scope of this document 3](#_Toc529975040)

[1.3 Software necessary to be prepared in advance 4](#_Toc529975041)

[1.4 Related documents 5](#_Toc529975042)

[2 Software specification 5](#_Toc529975043)

[2.1 The list of functions 5](#_Toc529975044)

[2.2 The list of structures 6](#_Toc529975045)

[2.3 Function specifications 7](#_Toc529975046)

[2.3.1 IL Core method 7](#_Toc529975047)

[2.3.1.1 OMX\_Init 7](#_Toc529975048)

[2.3.1.2 OMX\_Deinit 8](#_Toc529975049)

[2.3.1.3 OMX\_GetHandle 9](#_Toc529975050)

[2.3.1.4 OMX\_FreeHandle 10](#_Toc529975051)

[2.3.1.5 OMX\_SetupTunnel 11](#_Toc529975052)

[2.3.1.6 OMX\_TeardownTunnel 12](#_Toc529975053)

[2.3.2 Component APIs 13](#_Toc529975054)

[2.3.2.1 OMX\_SendCommand 13](#_Toc529975055)

[2.3.2.2 OMX\_GetParameter 14](#_Toc529975056)

[2.3.2.3 OMX\_SetParameter 15](#_Toc529975057)

[2.3.2.4 OMX\_GetState 16](#_Toc529975058)

[2.3.2.5 OMX\_UseBuffer 17](#_Toc529975059)

[2.3.2.6 OMX\_AllocateBuffer 18](#_Toc529975060)

[2.3.2.7 OMX\_FreeBuffer 19](#_Toc529975061)

[2.3.2.8 OMX\_EmptyThisBuffer 20](#_Toc529975062)

[2.3.2.9 OMX\_FillThisBuffer 21](#_Toc529975063)

[2.4 Callback function specification 22](#_Toc529975064)

[2.4.1 EventHandler 23](#_Toc529975065)

[2.4.2 EmptyBufferDone 24](#_Toc529975066)

[2.4.3 FillBufferDone 25](#_Toc529975067)

[2.5 Structures specification 26](#_Toc529975068)

[2.5.1 XAOMX\_AUDIO\_PARAM\_PARAMETRIC\_EQUALIZER 26](#_Toc529975069)

[2.5.2 XAOMX\_AUDIO\_PARAM\_GRAPHIC\_EQUALIZER 26](#_Toc529975070)

[2.5.3 XAOXM\_AUDIO\_PARAM\_EQUALIZER 27](#_Toc529975071)

[3 Process sequence 29](#_Toc529975072)

[3.1 Initialize Component 29](#_Toc529975073)

[3.2 Decoding sequence 30](#_Toc529975074)

[3.3 De-initialize Component 31](#_Toc529975075)

[4 Notes 32](#_Toc529975076)

[4.1 Function Call 32](#_Toc529975077)

[4.2 Other notes 32](#_Toc529975078)

[4.2.1 Allocation of memory 32](#_Toc529975079)

[4.2.2 Out of range memory access 32](#_Toc529975080)

[4.2.3 Combination with other applications 32](#_Toc529975081)

[4.2.4 Monitoring on Performance 32](#_Toc529975082)

- List of Figures -

[Figure 1‑1 The software architecture 3](#_Toc529975083)

[Figure 3‑1 Initialize the Component and preparation phase 29](#_Toc529975084)

[Figure 3‑2 Decoding sequence 30](#_Toc529975085)

[Figure 3‑3 De-Initialize Component and OMX IL Core 31](#_Toc529975086)

- List of Tables -

[Table 1‑1 The list of related documents 5](#_Toc529975087)

[Table 2‑1 List of functions 5](#_Toc529975088)

[Table 2‑2 List of structures 6](#_Toc529975089)

[Table 2‑3 PCM stream setting of Equalizer 28](#_Toc529975090)

ADSP Interface for Linux

RCG3AHIFL4001ZDPE\_AN\_EQZ

Rev. 1.00

Jun. 28, 2017

# Overview

## Overview of this document.

In this chapter, overview of Equalizer 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 via .

ARM

User Application

ADSP

Codec

DAC/ADC

ARM

Audio HW

SRC/ 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

## 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 further detailed descriptions of the functions.

Table 2‑1 List of functions

|  |  |  |
| --- | --- | --- |
|  | name | outline |
| IL Core Methods | OMX\_Init | Initialize the OpenMAX™ IL core |
| OMX\_Deinit | De-initialize the OpenMAX™ IL core |
| OMX\_GetHandle | Load that component into memory, validate it and return the component handle via the output parameter |
| OMX\_FreeHandle | Free a component handle (allocated by the OMX\_GetHandle) |
| OMX\_SetupTunnel | Establish a tunnel between components |
| OMX\_TeardownTunnel | Clears tunneled communication between components |
| Component APIs | OMX\_SendCommand | Send the command from application (IL-client) to component |
| OMX\_GetParameter | Retrieve the parameter from the component |
| OMX\_SetParameter | Setup the parameter from the component |
| OMX\_GetState | Get the current state of the component |
| OMX\_UseBuffer | Pass the handle to the buffer allocated by application |
| OMX\_AllocateBuffer | Allocate buffer on behalf of a component |
| OMX\_FreeBuffer | De-allocate buffer structure |
| OMX\_EmptyThisBuffer | Pass filled input buffer to the component |
| OMX\_FillThisBuffer | Pass the free output buffer to the component |

## The list of structures

Table 2‑2 shows the list of structures which user should allocate memory in using the software.

See 2.5 for further detailed descriptions of the structures.

Table 2‑2 List of structures

|  |  |
| --- | --- |
| name | outline |
| XAOMX\_AUDIO\_PARAM\_PARAMETRIC\_EQUALIZER | The structure of parameters of parametric equalizer |
| XAOMX\_AUDIO\_PARAM\_GRAPHIC\_EQUALIZER | The structure of parameters of graphic equalizer |
| XAOXM\_AUDIO\_PARAM\_EQUALIZER | The structure of parameters of equalizer |
| OMX\_AUDIO\_PARAM\_PCMMODETYPE | PCM mode type structures  For further information, refer to OpenMAX IL Specification 1.1.2, section 4.1.7 |

## Function specifications

### IL Core method

#### OMX\_Init

|  |  |  |
| --- | --- | --- |
| OMX\_Init | | |
| Synopsis | Initialize the OpenMAX™ IL core, including memory allocation and preparation for loading components. It is used as the very first call into OpenMAXTM IL core. | |
| Syntax | OMX\_API OMX\_ERRORTYPE OMX\_APIENTRY OMX\_Init(); | |
| Parameter | None | |
| Return values | OMX\_ErrorInsufficientResources | Failed to initialize due to not enough resource |
| OMX\_ErrorUndefined | Undefined error while processing command |
| OMX\_ErrorNone | Normal ends. Initialize successfully |

[notice] this function is called only once.

#### OMX\_Deinit

|  |  |  |
| --- | --- | --- |
| OMX\_Deinit | | |
| Synopsis | De-initialize OMX IL core, including its allocated memory and objects used to load/manage components. It is used as the very last call into OpenMAXTM IL core. | |
| Syntax | OMX\_API OMX\_ERRORTYPE OMX\_APIENTRY OMX\_Deinit(); | |
| Parameter | None | |
| Return value | OMX\_ErrorUndefined | Undefined error while processing command |
| OMX\_ErrorNone | Normal ends. De-initialize successfully |

[notice] this function is called only once.

#### OMX\_GetHandle

FD\_API\_EQZ\_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 name for Equalizer is:  “OMX.RENESAS.AUDIO.DSP.EQUALIZER” | |
| pAppData | A pointer to an IL client-defined value that will be returned during callbacks so that the IL client can identify the source of the callback | |
| pCallBacks | A pointer to an OMX\_CALLBACKTYPE structure containing the callbacks that the component will use for this IL client | |
| Return value | OMX\_ErrorInsufficientResources | | Failed to locate the component due to not enough resource |
| OMX\_ErrorInvalidState | | The proxy is not initialized. |
| OMX\_ErrorInvalidComponentName | | The component name parameter is invalid. |
| OMX\_ErrorNone | | Normal ends. |

[Covers: RD\_008]

#### OMX\_FreeHandle

FD\_API\_EQZ\_002

|  |  |  |  |
| --- | --- | --- | --- |
| OMX\_FreeHandle | | | |
| Synopsis | Free a handle allocated by the OMX\_GetHandle. The IL client should call OMX\_FreeHandle only when the component is in the OMX\_StateLoaded and when all the ports are not connected via any tunnels. | | |
| Syntax | OMX\_API OMX\_ERRORTYPE OMX\_APIENTRY OMX\_FreeHandle(  OMX\_IN OMX\_HANDLETYPE hComponent); | | |
| Parameter | hComponent | The handle of the component to be freed | |
| Return value | OMX\_ErrorBadParameter | | hComponent points to an invalid memory area. |
| OMX\_ErrorNone | | Normal ends. |

[Covers: RD\_008]

#### OMX\_SetupTunnel

|  |  |  |
| --- | --- | --- |
| OMX\_SetupTunnel | | |
| Synopsis | Handle the necessary calls to the components to set up the specified tunnel the two components. This method shall be called only when the component is in the OMX\_StateLoaded state or when the ports used for the tunnel are disabled (OMX\_StateExecuting, OMX\_StatePause, or OMX\_StateIdle states). | |
| Syntax | OMX\_API OMX\_ERRORTYPE OMX\_APIENTRY OMX\_SetupTunnel(  OMX\_IN OMX\_HANDLETYPE hOutput,  OMX\_IN OMX\_U32 nPortOutput,  OMX\_IN OMX\_HANDLETYPE hInput,  OMX\_IN OMX\_U32 nPortInput); | |
| Parameter | hOutput | Handle of the component whose port, specified in the nPortOutput parameter, will be used as the source for the tunnel |
| nPortOutput | Select the source port on component to be used in the tunnel |
| hInput | Handle of the component whose port, specified in the nPortInput parameter, will be used the destination for the tunnel |
| nPortInput | Select the destination port on component to be used in the tunnel |
| Return value | OMX\_ErrorBadParameter | hOutput and hInput point to invalid memory area. |
| OMX\_ErrorBadPortIndex | Port index of parameter is invalid. |
| OMX\_ErrorIncorrectStateOperation | Component is not in OMX\_StateLoaded |
| OMX\_ErrorUndefined | Undefined error while processing command |
| OMX\_ErrorPortsNotCompatible | One or both components are non-interop components which do not support tunneling. |
| OMX\_ErrorNone | Normal end |

#### OMX\_TeardownTunnel

|  |  |  |
| --- | --- | --- |
| OMX\_TeardownTunnel | | |
| Synopsis | Clear tunneled communication between an output port and an input port. After OMX\_TeardownTunnel returns successfully, these ports are no longer connected together. | |
| Syntax | OMX\_API OMX\_ERRORTYPE OMX\_APIENTRY OMX\_TeardownTunnel(  OMX\_IN OMX\_HANDLETYPE hOutput,  OMX\_IN OMX\_U32 nPortOutput,  OMX\_IN OMX\_HANDLETYPE hInput,  OMX\_IN OMX\_U32 nPortInput); | |
| Parameter | hOutput | Handle of the component whose port, specified in the nPortOutput parameter, is being used as the source for the tunnel |
| nPortOutput | Select the source port on component being used in the tunnel |
| hInput | Handle of the component whose port, specified in the nPortInput parameter, is being used the destination for the tunnel |
| nPortInput | Select the destination port on component being used in the tunnel |
| Return value | OMX\_ErrorBadParameter | hOutput or hInput components point to invalid memory area. |
| OMX\_ErrorBadPortIndex | Port index is invalid. |
| OMX\_ErrorIncorrectStateOperation | Component is not in OMX\_StateLoaded. |
| OMX\_ErrorNone | Normal end |

### Component APIs

#### OMX\_SendCommand

FD\_API\_EQZ\_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 |
| nParam1 | Integer parameter for the command that is to be executed (represented for STATETYPE, number of ports) |
| pCmdData | Pointer to a memory area contains specific parameters (marked buffer header) |
| Return value | OMX\_ErrorBadParameter | Parameter(s) is invalid: Command could not be recognized.  Invalid mark buffer area  Invalid number of ports  Invalid destination state (state could not be recognized) |
| OMX\_ErrorSameState | State transition is requested between same states. |
| OMX\_ErrorInsufficientResources | Fail to initialize codec setup due to insufficient resources |
| OMX\_ErrorNotImplemented | Don’t support OMX\_StatePause and OMX\_StateWaitForResources |
| OMX\_ErrorIncorrectStateTransition | The transition is invalid such as changing from OMX\_StateExecuting to OMX\_StatePause, etc. |
| OMX\_ErrorInvalidState | The executing state is not proper. |
| OMX\_ErrorBadPortIndex | Port index is invalid. |
| OMX\_ErrorIncorrectStateOperation | Execution is invalid in the current state of component. |
| OMX\_ErrorUndefined | Undefined error while processing command |
| OMX\_ErrorNone | Normal end. Command sending succeeds |

[Covers: RD\_008, RD\_015]

#### OMX\_GetParameter

FD\_API\_EQZ\_004

|  |  |  |
| --- | --- | --- |
| OMX\_GetParameter | | |
| Synopsis | Receive a parameter structure from IL Client and fill it with appropriate data of component | |
| Syntax | OMX\_ERRORTYPE OMX\_GetParameter(  OMX\_IN OMX\_HANDLETYPE hComponent,  OMX\_IN OMX\_INDEXTYPE nParamIndex,  OMX\_INOUT OMX\_PTR pComponentParameterStructure); | |
| Parameter | hComponent | Pointer to memory area of component handle |
| nParamIndex | The index of the structure that is to be sent. This value is from the OMX\_INDEXTYPE enumeration.  Supported index are:  OMX\_IndexParamPortDefinition  OMX\_IndexParamAudioPortFormat  OMX\_IndexParamPriorityMgmt  OMX\_IndexParamAudioPcm  XAOMX\_IndexParamAudioEqualizer |
| pComponentParameterStructure | Pointer to the IL client-allocated parameter structure |
| Return value | OMX\_ErrorUnsupportedIndex | Cannot recognize parameters |
| OMX\_ErrorBadParameter | Parameter is invalid for execution:  pComponentParameterStructure points to an invalid memory area. |
| OMX\_ErrorIncorrectStateOperation | Current state is OMX\_StateInvalid. |
| OMX\_ErrorBadPortIndex | Port index of parameter is invalid. |
| OMX\_ErrorNone | Normal ends. Getting parameter from component is successful. |

[Covers: RD\_008]

#### OMX\_SetParameter

FD\_API\_EQZ\_005

|  |  |  |
| --- | --- | --- |
| OMX\_SetParameter | | |
| Synopsis | Fill a parameter structure allocated by IL Client with appropriate data of component | |
| Syntax | OMX\_ERRORTYPE OMX\_SetParameter(  OMX\_IN OMX\_HANDLETYPE hComponent,  OMX\_IN OMX\_INDEXTYPE nIndex,  OMX\_IN OMX\_PTR pComponentParameterStructure); | |
| Parameter | hComponent | Pointer to memory area of component handle |
| nIndex | The index of the structure that is to be sent. It indicates which structure is requested by IL Client. This value is from the OMX\_INDEXTYPE enumeration.  Supported indexes are:  OMX\_IndexParamPortDefinition  OMX\_IndexParamAudioPortFormat  OMX\_IndexParamPriorityMgmt  OMX\_IndexParamStandardComponentRole  OMX\_IndexParamAudioPcm  XAOMX\_IndexParamAudioEqualizer |
| pComponentParameterStructure | Pointer to the IL client-allocated parameter structure to be filled |
| Return value | OMX\_ErrorBadParameter | Parameter is invalid for execution:  pComponentParameterStructure points to an invalid memory area. |
| OMX\_ErrorIncorrectStateOperation | Current state is OMX\_StateInvalid.  Port is locked.  Current state is not OMX\_StateLoaded.  (for OMX\_IndexParamPriorityMgmt and OMX\_IndexParamStandardComponentRole) |
| OMX\_ErrorBadPortIndex | Port index of parameter is invalid. |
| OMX\_ErrorUnsupportedIndex | The type (index) of parameter structure is not supported by component. |
| OMX\_ErrorNone | Normal ends. Setting parameter to component is successful. |

[Covers: RD\_008]

#### OMX\_GetState

|  |  |  |
| --- | --- | --- |
| OMX\_GetState | | |
| Synopsis | Return the current state of the component | |
| Syntax | OMX\_ERRORTYPE OMX\_GetState(  OMX\_IN OMX\_HANDLETYPE hComponent,  OMX\_OUT OMX\_STATETYPE \*pState); | |
| Parameter | hComponent | Pointer to memory area of component handle |
| \*pState | Pointer to an allocated memory area used to store component state |
| Return value | OMX\_ErrorBadParameter | Parameter is invalid for execution: pState points to an invalid memory area. |
| OMX\_ErrorNone | Normal end. Getting the state of the component is successful. |

#### OMX\_UseBuffer

|  |  |  |
| --- | --- | --- |
| OMX\_UseBuffer | | |
| Synopsis | Use a buffer allocated by the IL Client to a port or supplied by a tunneling component | |
| Syntax | OMX\_ERRORTYPE OMX\_UseBuffer(  OMX\_IN OMX\_HANDLETYPE hComponent,  OMX\_OUT OMX\_BUFFERHEADERTYPE \*\*ppBufferHdr,  OMX\_IN OMX\_U32 nPortIndex,  OMX\_IN OMX\_PTR pAppPrivate,  OMX\_IN OMX\_U32 nSizeBytes,  OMX\_IN OMX\_U8 \*pBuffer); | |
| Parameter | hComponent | Pointer to memory area of component handle |
| \*\*ppBufferHdr | Pointer to OMX\_BUFFERHEADERTYPE which contains meta-information about the buffer. It receives the pointer to the buffer header. |
| nPortIndex | Target port that uses the buffer (index into the port definition array of the component) |
| pAppPrivate | Pointer to the private memory area of IL Client. It is used to initialize the pAppPrivate member of the buffer header structure. |
| nSizeBytes | The size (byte) of the buffer to allocate |
| \*pBuffer | Pointer to the allocated buffer to be used |
| Return value | OMX\_ErrorBadParameter | Parameter is invalid for execution: ppBufferHdr points to an invalid memory area.  Target port is invalid. |
| OMX\_ErrorIncorrectStateOperation | Port is not populated. |
| OMX\_ErrorInsufficientResources | Not enough resources |
| OMX\_ErrorUndefined | Undefined error while processing command |
| OMX\_ErrorNone | Normal end. Setting the buffer to the target port is successful. |

#### OMX\_AllocateBuffer

|  |  |  |
| --- | --- | --- |
| OMX\_AllocateBuffer | | |
| Synopsis | Allocate the buffer and the buffer header, request mapping the buffer with the 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 | Parameter is invalid for execution: ppBuffer points to an invalid memory area.  Target port is invalid. |
| OMX\_ErrorIncorrectStateOperation | Port is not populated. |
| OMX\_ErrorInsufficientResources | Not enough resources |
| OMX\_ErrorUndefined | Undefined error while processing command |
| OMX\_ErrorNone | Normal end. Setting the buffer to the target port is successful. |

#### OMX\_FreeBuffer

|  |  |  |
| --- | --- | --- |
| OMX\_FreeBuffer | | |
| Synopsis | De-allocate buffer structure | |
| Syntax | OMX\_ERRORTYPE OMX\_FreeBuffer(  OMX\_IN OMX\_HANDLETYPE hComponent,  OMX\_IN OMX\_U32 nPortIndex,  OMX\_IN OMX\_BUFFERHEADERTYPE \*pBuffer); | |
| Parameter | hComponent | Pointer to memory area of component handle |
| nPortIndex | Target port (index into the port definition array of the component) |
| \*pBuffer | Pointer to OMX\_BUFFERHEADERTYPE structure which contains meta-information about the buffer. It specifies the index of the input port that receives the buffer. |
| Return value | OMX\_ErrorBadParameter | Parameter is invalid: pBuffer points to an invalid memory area.  Target port is invalid. |
| OMX\_ErrorIncorrectStateOperation | The port is not unpopulated (all buffers of the port is active (being used), so cannot free the buffer). |
| OMX\_ErrorUndefined | Undefined error while processing command |
| OMX\_ErrorNone | Normal end. Transferring the buffer to the client is successful. |

#### OMX\_EmptyThisBuffer

|  |  |  |
| --- | --- | --- |
| OMX\_EmptyThisBuffer | | |
| Synopsis | Send a filled buffer to an input port of a component | |
| Syntax | OMX\_ERRORTYPE OMX\_EmptyThisBuffer(  OMX\_IN OMX\_HANDLETYPE hComponent,  OMX\_IN OMX\_BUFFERHEADERTYPE \*pBuffer); | |
| Parameter | hComponent | Pointer to memory area of component handle |
| \*pBuffer | Pointer to OMX\_BUFFERHEADERTYPE structure which contains meta-information about the buffer. It specifies the index of the input port that receives the buffer. |
| Return value | OMX\_ErrorBadParameter | Parameter is invalid: pBuffer points to an invalid memory area.  Invalid buffer header  Buffer filled length is zero. |
| OMX\_ErrorBadPortIndex | Port index of buffer is invalid. |
| OMX\_ErrorIncorrectStateOperation | Execution is invalid in current state of the component.  Component is not in OMX\_StateExecuting.  Sending a buffer after end-of-stream |
| OMX\_ErrorNone | Normal end. Buffer is transferred to the input port of a component successfully. |

#### OMX\_FillThisBuffer

|  |  |  |
| --- | --- | --- |
| OMX\_FillThisBuffer | | |
| Synopsis | Receive an empty buffer to an output port of a component and fill it with appropriate output data | |
| Syntax | OMX\_ERRORTYPE OMX\_FillThisBuffer(  OMX\_IN OMX\_HANDLETYPE hComponent,  OMX\_IN OMX\_BUFFERHEADERTYPE \*pBuffer); | |
| Parameter | hComponent | Pointer to memory area of component handle |
| \*pBuffer | Pointer to OMX\_BUFFERHEADERTYPE which contains meta-information about the buffer. It specifies the index of the output port that receives the buffer. |
| Return value | OMX\_ErrorBadParameter | Parameter is invalid for execution:  pBuffer points to an invalid memory area.  Invalid buffer header |
| OMX\_ErrorBadPortIndex | Port index of buffer is invalid. |
| OMX\_ErrorIncorrectStateOperation | Execution is invalid in current state of component.  Output port is enabled.  Sending a buffer after end-of-stream |
| OMX\_ErrorNone | Normal ends. Transferring buffer to client is successful. |

## 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 3-7 of OMX IL Specification v1.1.2 for specific details.

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

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

### EmptyBufferDone

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

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

* The IL client commands a transition from OMX\_StateExecuting or OMX\_StatePause to OMX\_StateIdle.
* The IL client flushes or disables a port.

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

OMX\_ERRORTYPE(\*OMX\_CALLBACKTYPE::EmptyBufferDone)(

OMX\_IN OMX\_HANDLETYPE hComponent,

OMX\_IN OMX\_PTR pAppData,

OMX\_IN OMX\_BUFFERHEADERTYPE\* pBuffer)

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

### FillBufferDone

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

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

* The IL client commands a transition from OMX\_StateExecuting or OMX\_StatePause to OMX\_StateIdle.
* The IL client flushes or disables a port.

FillBufferDone is defined as follows.

OMX\_ERRORTYPE(\*OMX\_CALLBACKTYPE::FillBufferDone)(

OMX\_IN OMX\_HANDLETYPE hComponent,

OMX\_IN OMX\_PTR pAppData,

OMX\_IN OMX\_BUFFERHEADERTYPE\* pBuffer)

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

## Structures specification

### XAOMX\_AUDIO\_PARAM\_PARAMETRIC\_EQUALIZER

|  |  |  |  |
| --- | --- | --- | --- |
| XAOMX\_AUDIO\_PARAM\_PARAMETRIC\_EQUALIZER | | | |
| Synopsis | This is the structure of parameters of parametric equalizer.  Specify the parameters of parametric filters | | |
| Member | OMX\_U32 | nSize | The size of the structure in bytes |
| OMX\_VERSIONTYPE | nVersion | OMX specification version information |
| OMX\_S32 | Type[9] | The type of the filter  Value range: Peaking, Bass, Treble, Through filter  Through is default filter type. |
| OMX\_S32 | FreqCenter[9] | The frequency center of a filter  Value range:  -Peaking filter: 20-20kHz (or less than Nyquist frequency)  -Bass filter: 50-500Hz  -Treble filter: 5k - 11kHz |
| OMX\_S32 | Gain[9] | The gain of a filter  Value range: to (fixed point Q4.28) |
| OMX\_S32 | BandWidth[9] | The bandwidth of a filter  Value range: to (fixed point Q5.27) |
| OMX\_S32 | GainBase[9] | The gain base of a filter  Value range: to (fixed point Q4.28) |

### XAOMX\_AUDIO\_PARAM\_GRAPHIC\_EQUALIZER

|  |  |  |  |
| --- | --- | --- | --- |
| XAOMX\_AUDIO\_PARAM\_GRAPHIC\_EQUALIZER | | | |
| Synopsis | This is the structure of parameters of graphic equalizer.  Specify the parameters of graphic filter | | |
| Member | OMX\_U32 | nSize | The size of the structure in bytes |
| OMX\_VERSIONTYPE | nVersion | OMX specification version information |
| OMX\_S32 | Gain\_g[5] | The gain of the graphic filter  Value range: to (fixed point Q4.28) |

### XAOXM\_AUDIO\_PARAM\_EQUALIZER

|  |  |  |  |
| --- | --- | --- | --- |
| XAOXM\_AUDIO\_PARAM\_EQUALIZER | | | |
| Synopsis | This is the structure of parameters of equalizer.  Specify the parameters of the equalizer | | |
| Member | OMX\_U32 | nSize | The size of the structure in bytes |
| OMX\_VERSIONTYPE | nVersion | OMX specification version information |
| OMX\_S32 | Eqz\_type | The type of the equalizer  Value range:   * Parametric Equalizer: 0 * Graphic Equalizer: 1 |
| XAOMX\_AUDIO\_PARAM\_PARAMETRIC\_EQUALIZER | stEqCoef | Parametric equalizer coefficient setting parameters |
| XAOMX\_AUDIO\_PARAM\_GRAPHIC\_EQUALIZER | stEqGCoef | Graphic equalizer coefficient setting parameters |

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

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

Table 2‑3 PCM stream setting of Equalizer

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

# Process sequence

## Initialize Component

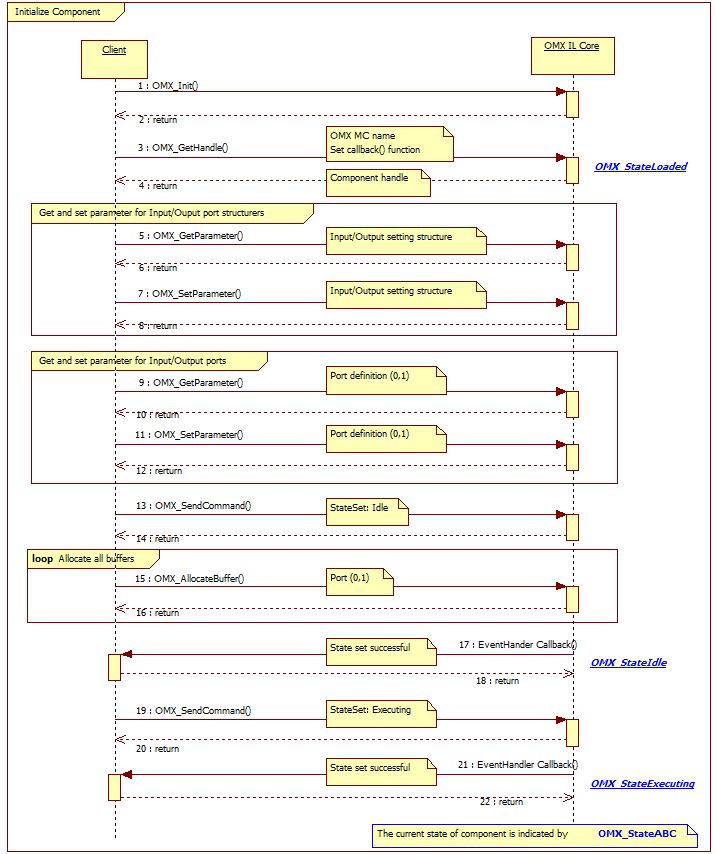


Figure 3‑1 Initialize the Component and preparation phase

## Decoding sequence

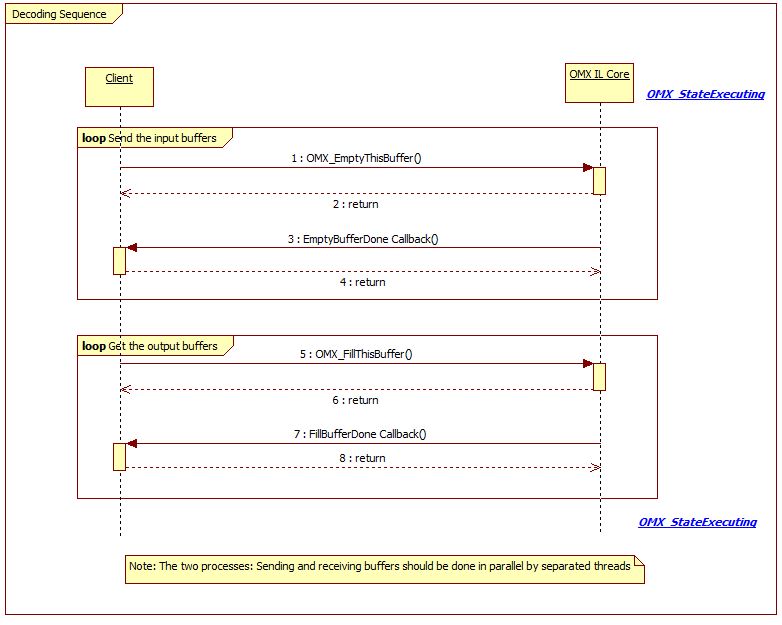


Figure 3‑2 Decoding sequence

## De-initialize Component

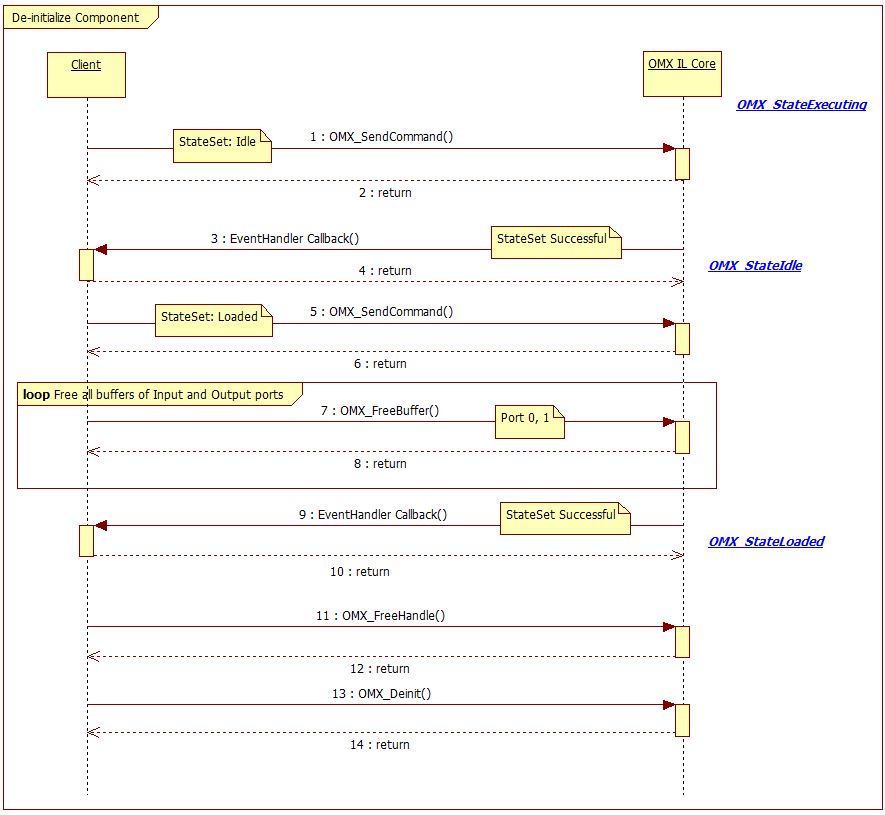


Figure 3‑3 De-Initialize Component and OMX IL Core

*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.