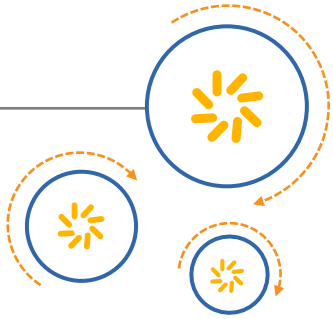
Qualcomm Technologies, Inc.



DragonBoard™ 410c based on Qualcomm® Snapdragon™ 410E processor

OMX Video Decoder - Android

September 2016

Questions or comments: https://www.96boards.org/DragonBoard410c/forum

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

LM80-P0436-16 Rev C

Revision history

Revision	Date	Description
C	September 2016	Updated to ‘E’ part
B	June 10, 2015	Revised OMX source code information & port definition
A	May 27, 2015	Initial release

Contents

Figures

Tables

1 Introduction

1.1 Purpose

This document is a reference for integrating hardware video decoders using the Qualcomm Technologies, Inc. (QTI) implementation of the OpenMAX (OMX) Integration Layer (IL).

1.2 Scope

This document applies to devices using the APQ8016E processor.

It is intended for third-party implementers who have multimedia frameworks that will use QTI hardware codecs for the video decoder.

This document is based on the OpenMAX 1.1 specification (*OpenMAX Integration Layer Application Programming Interface Specification, Version 1.1.2, 2008*) and assumes prior knowledge of the OpenMAX standard.

1.3 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

Commands to be entered appear in a different font, e.g., **`copy a:*.* b:`**.

Button and key names appear in bold font, for example, click **Save** or press **Enter**.

Shading indicates content that has been added or changed in this revision of the document.

1.4 Acronyms and terms

Term	Definition
BDU	Bitstream Data Unit
Client	Multimedia frameworks
Component	QTI OpenMAX component
IL	Integration Layer
MDP	Mobile Display Processor
MPEG	Motion Picture Experts Group
NAL	Network Abstraction Layer
OMX	OpenMAX

Term	Definition
OMX core	QTI OpenMAX core
PPS	Picture Parameter Set
QCIF	Quarter Common Intermediate Format
QTI	Qualcomm Technologies, Inc.
SC	Start Code
SPS	Sequence Parameter Set
VOL	Visual Object Layer

2 OpenMAX Components

2.1 OpenMAX core

The OpenMAX core component is the top-level interface exposed to the multimedia frameworks for use with all QTI hardware codecs.

This component supports the standard OMX IL interface specification (*OpenMAX Integration Layer Application Programming Interface Specification, Version 1.1.2, 2008*) for easy plug-in and is constructed as a dynamic library.

2.2 Integration procedure

The OMX core component is the only shared library that must be linked with the client's framework to use the QTI hardware video decoders.

This library is available in the Android build as `libOmxCore.so`.

2.3 OpenMAX components for the video decoder

The OMX video decoder components are specific to decoders and are loaded by the OMX core at runtime based on a client request. These components are derived from a common interface and implement the functionality of the specific decoder.

3 OpenMAX Sequence

3.1 Prerequisite

Before proceeding with the OMX sequence, link to the OMX core library identified in Section 2.2.

3.2 Determine support for the role

To determine whether the QTI OMX core supports the required role, the client can enumerate the component name based on the role from the OMX core. If no component supports the given role, the output parameter for the OMX_GetRolesOfComponent is 0.

Figure 3-1 shows the IL client flow for querying the names of all installed components that support a given role.

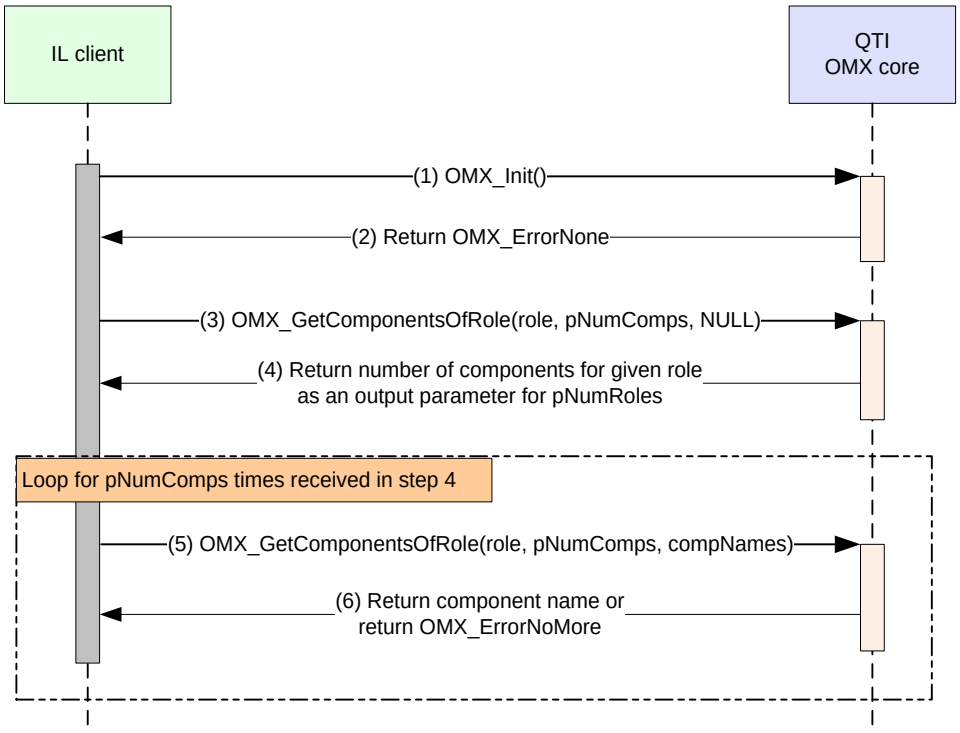


Figure 3-1 Sequence diagram for initialization

3.3 Load the component

Figure 3-2 shows the OMX_GetHandle API used to get the component handle for the required role.

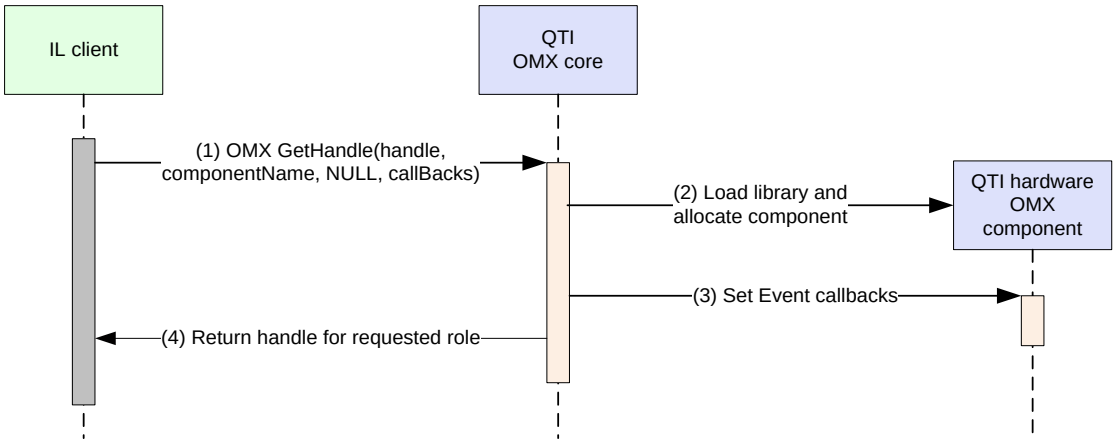


Figure 3-2 Sequence diagram for loading a component

The following items correspond to the steps in Figure 3-2:

- The OMX_GetHandle call loads the component library dynamically and initializes the components (2).
- Callback events are set to directly notify the client for FillBufferDone, EmptyBufferDone, and events from the components (3).

3.4 Handshake with the component

After the component handle is acquired, the next step is to initialize and configure the component. To initialize and configure the component, the client can retrieve the port definition (OMX_PARAM_PORTDEFINITIONTYPE) and format of the components and set them accordingly.

Because QTI OMX components and video drivers are optimized for the specific chipsets, QTI recommends that the client use the port definitions of the components.

Figure 3-3 shows handshaking of the IL client with the QTI hardware OMX component for port configuration.

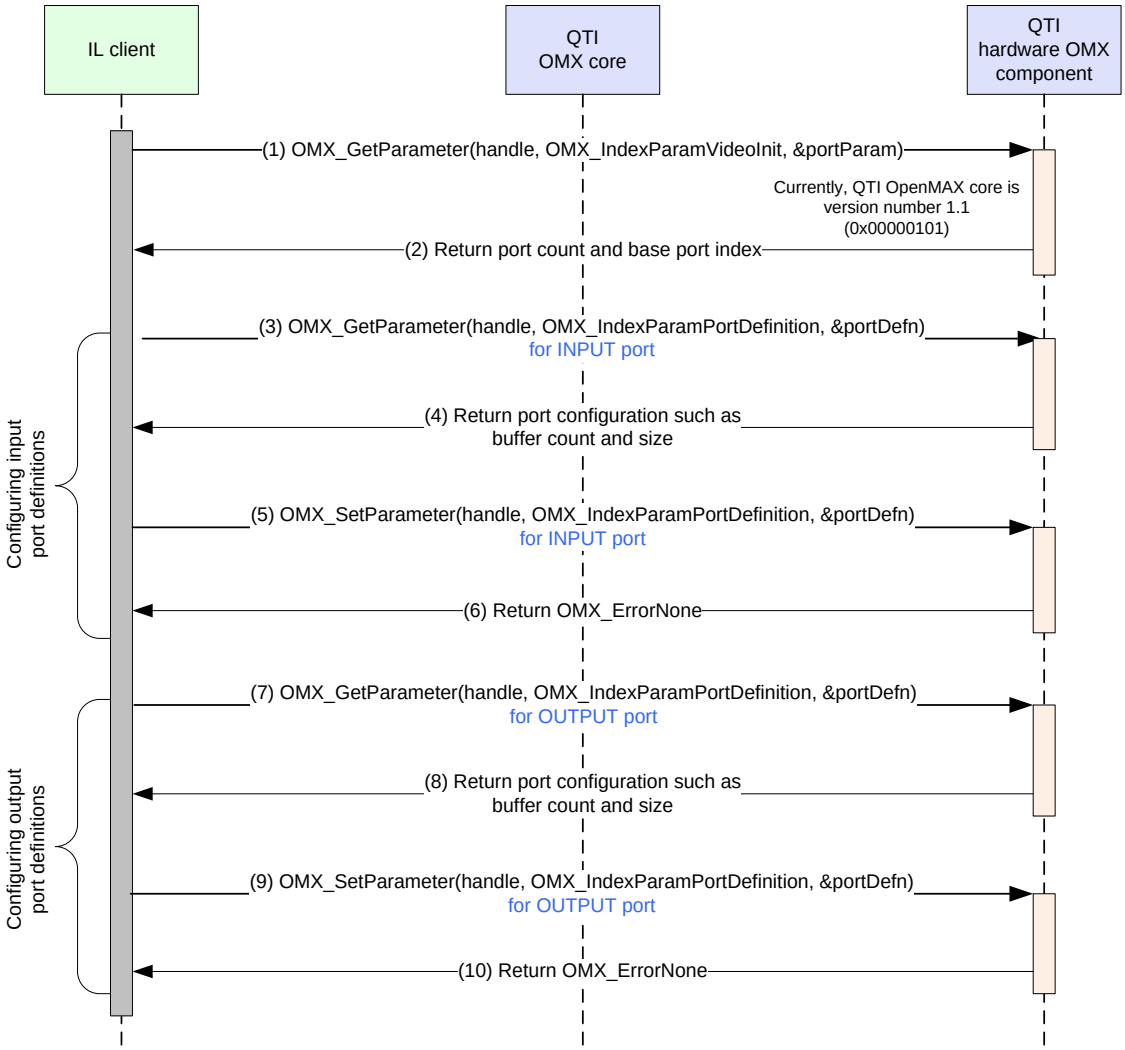


Figure 3-3 Sequence diagram for configuring a component port

The following items correspond to the steps in Figure 3-3:

- OMX_GetParameter (with OMX_IndexVideoInit) is called (1) to retrieve the number of ports supported by the QTI OMX component.
- The component returns the number of ports supported as well as the base port index to retrieve the individual port configuration (2).
- For each port, the client can get information about the preferred buffer size, number of buffers, etc., of the component (3 and 7).
- The IL client can call OMX_SetParameter to configure the port with either the configuration retrieved (4 and 8) or with a different configuration.

Table 3-1 shows the configuration parameters that can be set using OMX_IndexParamPort Definition for the input port.

Table 3-1 Configuration parameters set using OMX_IndexParamPortDefinition for input port

OMX_Index	OMX parameter	Comments
OMX_IndexParam PortDefinition (input port)	OMX_PARAM_PORTDEFINITIONTYPE; .nBufferCountmin	This parameter represents the minimum buffer count as recommended by the driver. It is returned after the omx_getparameter call using index OMX_IndexParamPort Definition.
	OMX_PARAM_PORTDEFINITIONTYPE; .nBufferCountActual	Make sure this value is greater than .nBufferCountmin (see above)
	OMX_VIDEO_PORTDEFINITIONTYPE; format.video.nFrameWidth	Frame width
	OMX_VIDEO_PORTDEFINITIONTYPE; format.video.nFrameHeight	Frame height
	OMX_VIDEO_PORTDEFINITIONTYPE; format.video.xFramerate	Frame rate in Q16 format
	OMX_VIDEO_PORTDEFINITIONTYPE; .nBufferSize	Minimum buffer size in bytes allocated for this port

Table 3-2 shows the configuration parameters that can be set using OMX_IndexParamPort Definition for the output port.

Table 3-2 Configuration parameters set using OMX_ IndexParamPortDefinition for output port

OMX_Index	OMX parameter	Comments
OMX_IndexParam PortDefinition (output port)	OMX_PARAM_PORTDEFINITIONTYPE; .format.video.pNativeWindow	Handle to display component
	OMX_PARAM_PORTDEFINITIONTYPE; .nBufferCountmin	This parameter represents the minimum buffer count as recommended by the driver. It is returned after the omx_getparameter call using index OMX_IndexParamPort Definition.
	OMX_PARAM_PORTDEFINITIONTYPE; .nBufferCountActual	Make sure this value is greater than .nBufferCountmin (see above)
	OMX_VIDEO_PORTDEFINITIONTYPE; .nBufferSize	Minimum buffer size in bytes allocated for this port

3.5 Decoder configuration parameters

The OMX IL component is able to accept various encode parameters in addition to the mandatory parameters, such as input frame width and height, target frame rate, target bit rate, profile, and level. Figure 3-4 shows how to set specific parameters.

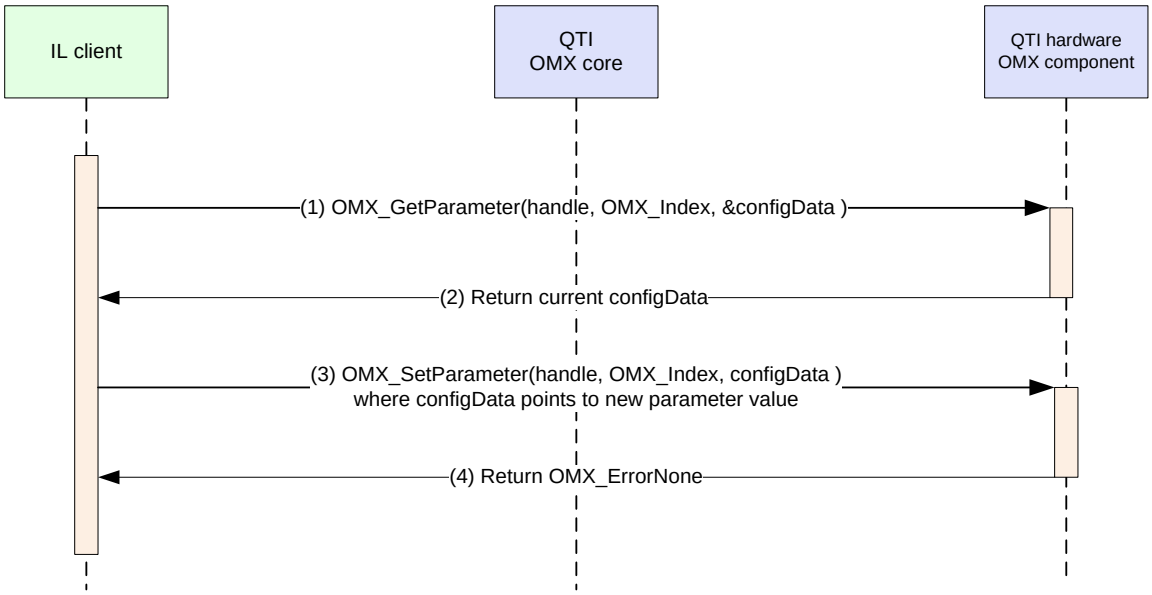


Figure 3-4 Sequence diagram for configuring encoder with OMX_Index parameter

The following items correspond to the steps in Figure 3-4:

- The IL client can obtain the current configuration using OMX_GetParameter (1).
- The IL client must pass the OMX parameter information using OMX_SetParameter with the corresponding OMX_Index before sending any input data through OMX_EmptyThisBuffer (3).

Table 3-3 shows the configuration parameters that can be set using OMX_Index.

Table 3-3 Configuration parameters set using OMX_Index

OMX_Index	OMX parameter	Comments
OMX_IndexParamVideo PortFormat* (output port)	OMX_VIDEO_PARAM_PORTFORMATTYPE; .eColorFormat	Set color format: ▪ 0→ QOMX_COLOR_ FORMATYUV420 PackedSemiPlanar32m (NV12) ▪ 1→ OMX_COLOR_ FormatYUV420Planar (YUV420 Planar)
OMX_QcomIndexPort Defn (input port)	OMX_QCOM_PARAM_PORTDEFINITIONTY PE; .nFramePackingFormat	Possible values: ▪ OMX_QCOM_FrameP acking_Arbitrary ▪ OMX_QCOM_FrameP acking_OnlyOne CompleteFrame

OMX_Index	OMX parameter	Comments
OMX_IndexParam StandardComponentRole	OMX_PARAM_COMPONENTROLETYPE; .cRole	Specifies codec
OMX_QcomIndexParamH 264TimeInfo	QOMX_ENABLETYPE; .bEnable	Enables parsing for H.264 SEI NAL unit for additional timestamp information; used for ISDBT digital television standard
OMX_QcomIndexParamVi deoDecoderPicture Order	QOMX_VIDEO_DECODER_PICTURE_ ORDER; .eOutputPictureOrder	Possible values: ▪ QOMX_VIDEO_ DISPLAY_ORDER ▪ QOMX_VIDEO_ DECODE_ORDER
OMX_QcomIndexParam FrameInfoExtraData	QOMX_ENABLETYPE; .bEnable	Provides information about: ▪ Picture type ▪ Interlace type ▪ panScan ▪ AspectRatio ▪ Displayaspectratio ▪ nconcealed macroblock count ▪ nframerate ▪ ntimestamp
OMX_QcomIndexParam InterlaceExtraData	QOMX_ENABLETYPE; .bEnable	Provides information about if progressive or interlaced
OMX_QcomIndexParamVi deoSyncFrame DecodingMode	No parameter available	Enables Thumbnail mode
OMX_QcomIndexParamEn ableSmoothStreaming	No parameter available	Enables smooth streaming feature; for more information, see Salesforce solution 00027736
OMX_QcomIndexParamEn ableTimeStamp Reorder	QOMX_INDEXTIMESTAMPREORDER; .bEnable	In QOMX_VIDEO_ DISPLAY_ORDER mode, manages ordering of frames

The QTI decoder supports a proprietary color format, requiring an extra step to integrate QTI OMX decoder components.

The QTI decoder supports the YUV420SemiPlanar format, that is, YUV planar format, organized with a first plane containing Y pixels and a second plane containing interleaved U and V pixels. U and V pixels are subsampled by a factor of two, both horizontally and vertically.

Figure 3-5 shows the method for retrieving a color format using the OMX_SetParameter call.

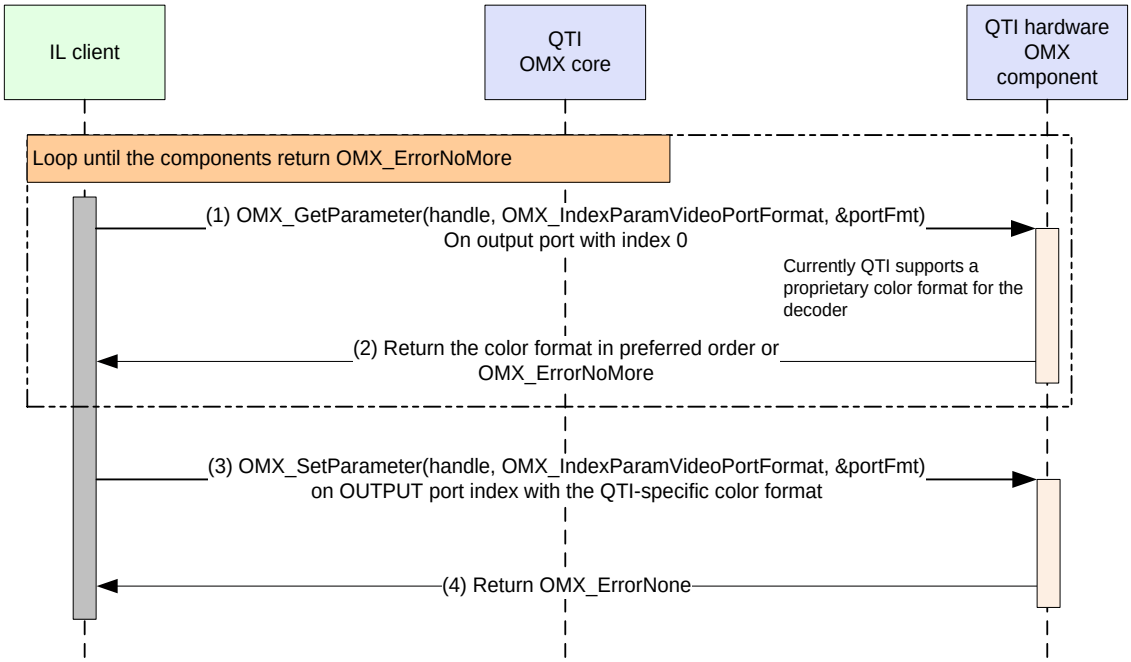


Figure 3-5 Sequence diagram for retrieving a color format supported by a component

The following items correspond to the steps in Figure 3-5:

- The IL client can query the component in the loop shown for supported color formats until the component returns OMX_ErrorNoMore (1).
- Index 0 is currently reserved for the YUV420PackedSemiPlanar32m color format, which is used to set the color format of the component (3).

3.6 Buffer allocation

After the component is configured, the next step is to allocate the buffers. QTI hardware components use physical memory for output buffers to interface with the DSP, so the client uses the AllocateBuffer model for the output port. For the input port, there is no such restriction.

Figure 3-6 shows the buffer allocation sequence model.

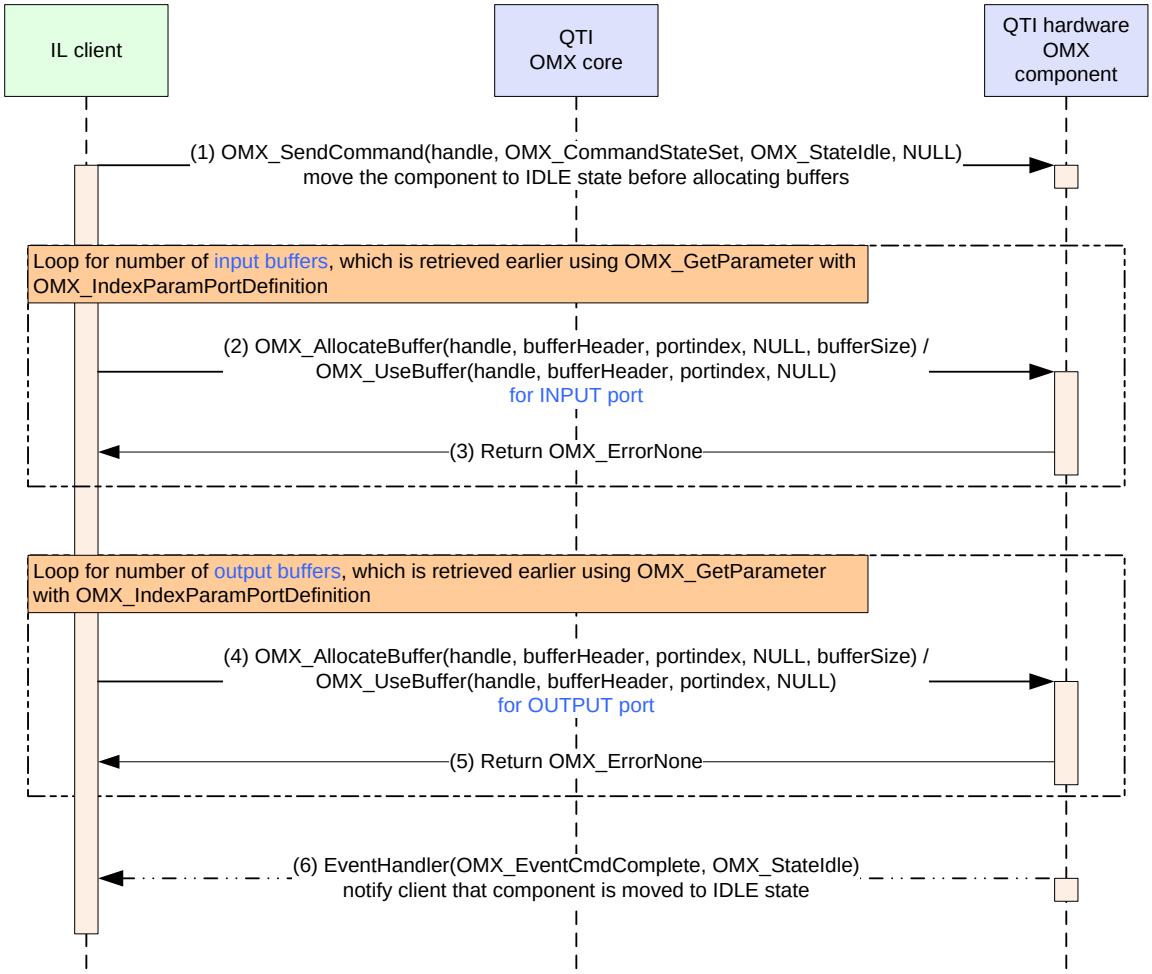


Figure 3-6 Sequence diagram for buffer allocations

The following items correspond to the steps in Figure 3-6:

- To allocate the buffer, the components are moved from the Loaded state to the Idle state (1).
- On the input port, the client can use either OMX_UseBuffer or OMX_AllocateBuffer calls to the OMX component (2). The component is called for the number of input buffers in a loop.
- On the output port, the client can use either OMX_UseBuffer or OMX_AllocateBuffer calls to the OMX component (4). The component is called for the number of input buffers in a loop.

NOTE: For better performance, use OMX_AllocateBuffer on the output port. This avoids the memcpy of output YUV frames from physical to virtual memory.

- Once the buffers are successfully allocated on the input and output ports, the OMX components generate the OMX_EventCmdComplete event for the Loaded-to-Idle state transition and send it to the client using EventHandlerCallback (6).

3.7 Dynamic port configuration

Figure 3-7 shows changing parameters dynamically using OMX_SetConfig.

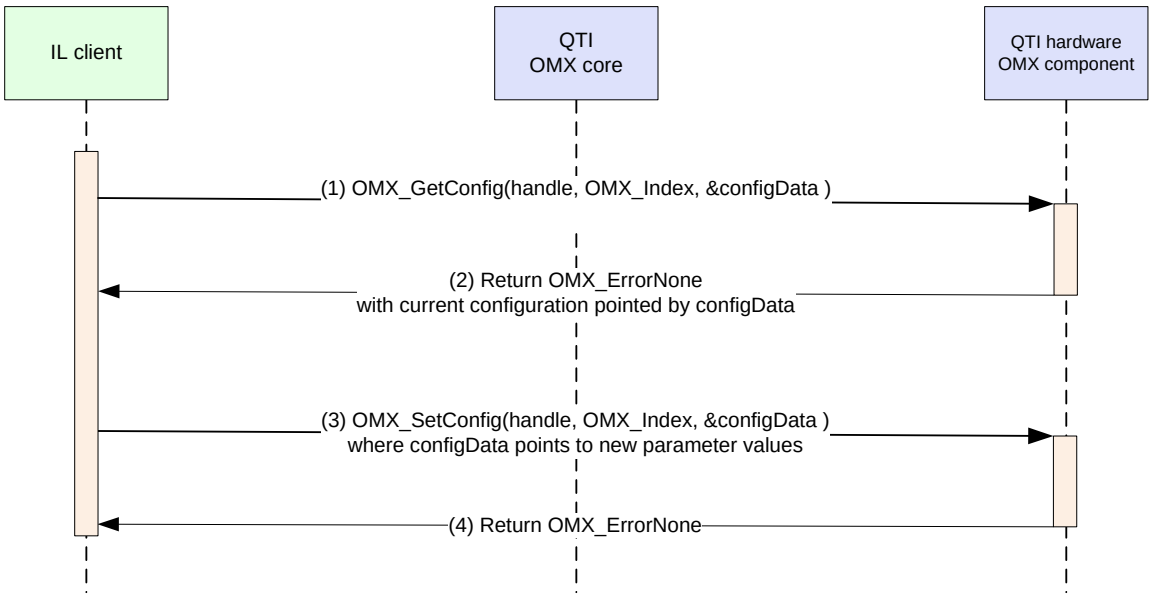


Figure 3-7 Sequence diagram for dynamically changing parameters

Table 3-4 Configuration parameters set dynamically using OMX_SetConfig

Index	OMX parameter	Comments
OMX_IndexVendor VideoExtraData (see Figure 3-8 and Figure 3-11Figure 3-8)	OMX_VENDOR_EXTRA DATATYPE; .pData	See Figure 3-8 and Figure 3-11 for more information
	OMX_VENDOR_EXTRA DATATYPE; .nDataSize	
OMX_IndexConfigVideo NalSize (see Figure 3-11)	OMX_VIDEO_CONFIG_ NALSIZE; .nNaluBytes	Possible values are: ▪ 0→V4L2_MPEG_VIDC_VIDEO_NAL_ FORMAT_STARTCODES ▪ 2→V4L2_MPEG_VIDC_VIDEO_NAL_ FORMAT_TWO_BYTE_LENGTH ▪ 4→V4L2_MPEG_VIDC_VIDEO_NAL_ FORMAT_FOUR_BYTE_LENGTH
OMX_IndexVendorVideo FrameRate (input port)	OMX_VENDOR_VIDEO FRAMERATE; .nFps	Frame rate set by OMX client

3.8 MPEG-4 decoder-specific configuration

The MPEG-4 decoder requires the VOL header to extract the height and width of the clip. By default, the IL client can pass the VOL header as part of the first OMX_EmptyThisBuffer call. If the IL client wants to set the VOL header by using OMX_SetConfig, it can use the following sequence.

3.8.1 Configure the VOL header

The IL client must pass the VOL header using OMX_SetConfig (with OMX_IndexVendorVideo ExtraData) before sending input data through OMX_EmptyThisBuffer. See (1) in Figure 3-8 .

The component parses the VOL header and uses the information for configuring the video driver. Figure 3-8 shows configuration of the MPEG-4 decoder component using the VOL header.

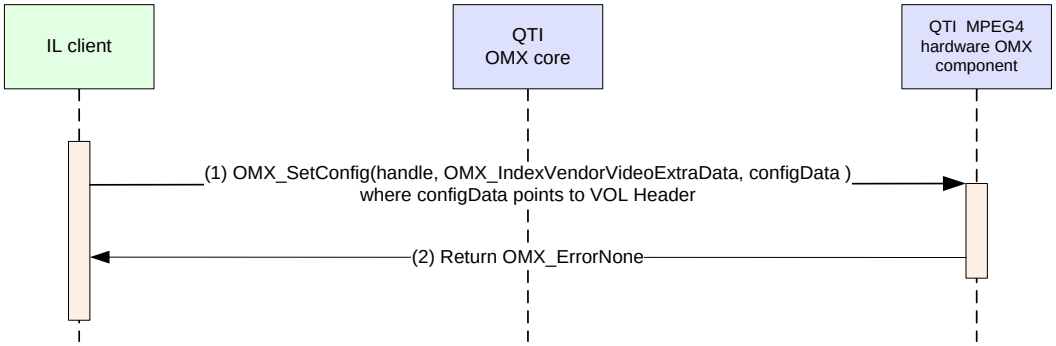


Figure 3-8 Sequence diagram for configuring MPEG-4 decoder with VOL header

3.9 VC1 decoder-specific configuration

The VC1 decoder requires a sequence start code to extract the height and width of the clip. Based on the clip type (Simple/Advanced), the following two methods can be used.

3.9.1 Simple/Main profile

Simple and Main profile clips can only be passed as a sequence header and one complete frame data in two different buffers. VC1 decoders do not support arbitrary bytes with Simple/Main profile. The OMX frame packing format for input port 6 must be defined as OMX_QCOM_ FramePacking_OnlyOneCompleteFrame.

```
inputPortFmt.nFramePackingFormat =
OMX_QCOM_FramePacking_OnlyOneCompleteFrame
```

The OMX IL client shall do OMX_SetParameter (handle,OMX_QcomIndexPortDefn, (OMX_PTR)&inputPortFmt), as shown in Figure 3-9.

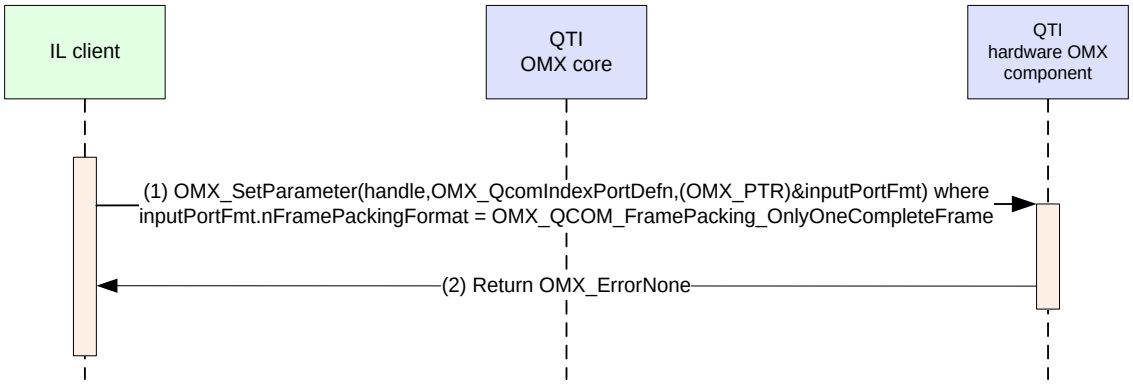


Figure 3-9 OpenMax IL client process

At the first OMX_EmptyThisBuffer(), the OpenMax IL client shall push the entire sequence Layer Data as referred by the SMPTE 421M standard *(VC-1 Compressed Video Bitstream Format and Decoding Process)* structC, along with the height and width (if not already passed as part of codec config).

With the next EmptyThisBuffer(), the OpenMax IL client shall push the Frame Data at the first Frame Layer as referred by Annex L.3 of the reference document *VC-1 Compressed Video Bitstream Format and Decoding Process (SMPTE 421M)*.

3.9.2 Advanced profile

For Advanced profile clips, the OMX supports sending data as:

- One complete Frame Data option, or
- Arbitrary bitstream option, where the frame can be divided into multiple fragments before sending them to the OMX component.

These two methods are described below.

3.9.2.1 Arbitrary bitstream (default mode)

The OMX IL client can push an arbitrary bitstream to the OMX using the OMX_EmptyThisBuffer() function.

The steps that pass the header are as follows; all three fragments can be in different buffers:

1. < sequence SC - 0x0000010F > + sequence header
2. < entry point SC - 0x0000010E > + entry point header
3. < Frame Start Code - 0x0000010D > + Frame Data – Frame SC is needed for each frame data

3.9.2.2 One complete frame data

The OMX IL client shall do OMX_SetParameter (handle,OMX_QcomIndexPortDefn, (OMX_PTR)&inputPortFmt), as shown in Figure 3-10, to pass one complete Frame Data.

```
inputPortFmt.nFramePackingFormat =
OMX_QCOM_FramePacking_OnlyOneCompleteFrame
```

The OMX IL client shall push one complete BDU per buffer using OMX_EmptyThisBuffer(), as shown in Figure 3-10.

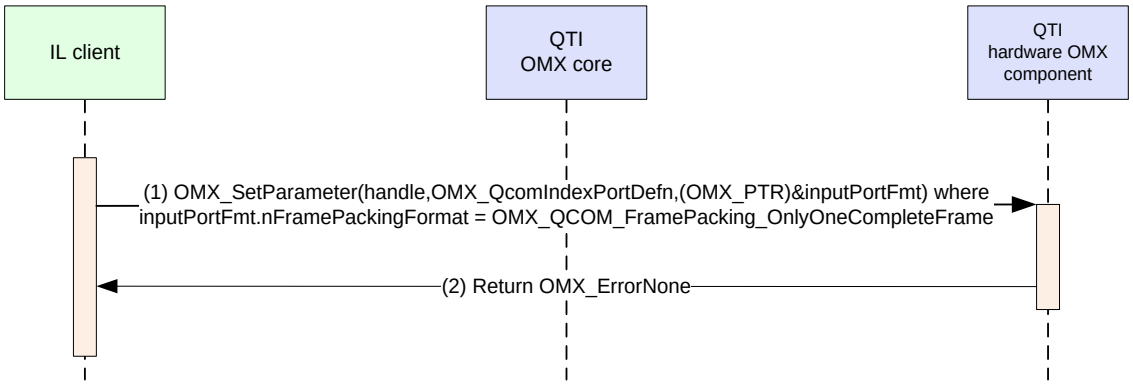


Figure 3-10 OpenMax IL client pushing one complete BDU per buffer

For BDU details, see Annex E and Annex G of the reference document *VC-1 Compressed Video Bitstream Format and Decoding Process (SMPTE 421M)*. The frame passed as one complete frame consists of:

1. < sequence SC - 0x0000010F > + sequence header
2. < entry point SC - 0x0000010E > + entry point header
3. < Frame Start Code - 0x0000010D > + Frame Data – Frame start code is needed for each Frame Data

3.10 H.263 decoder-specific configuration

The H.263 decoder requires a short header to extract the height and width of the clip. The IL client can pass the short header as part of the first OMX_EmptyThisBuffer call.

There is no method to set the short header using the OMX_SetConfig method.

3.11 Data processing

When the component is ready for decoding after buffer allocation and configuration, the client can transition the component to the Executing state, after which the client can start sending the input bitstream for processing.

Figure 3-11 shows the sequence of calls for processing the input bitstream.

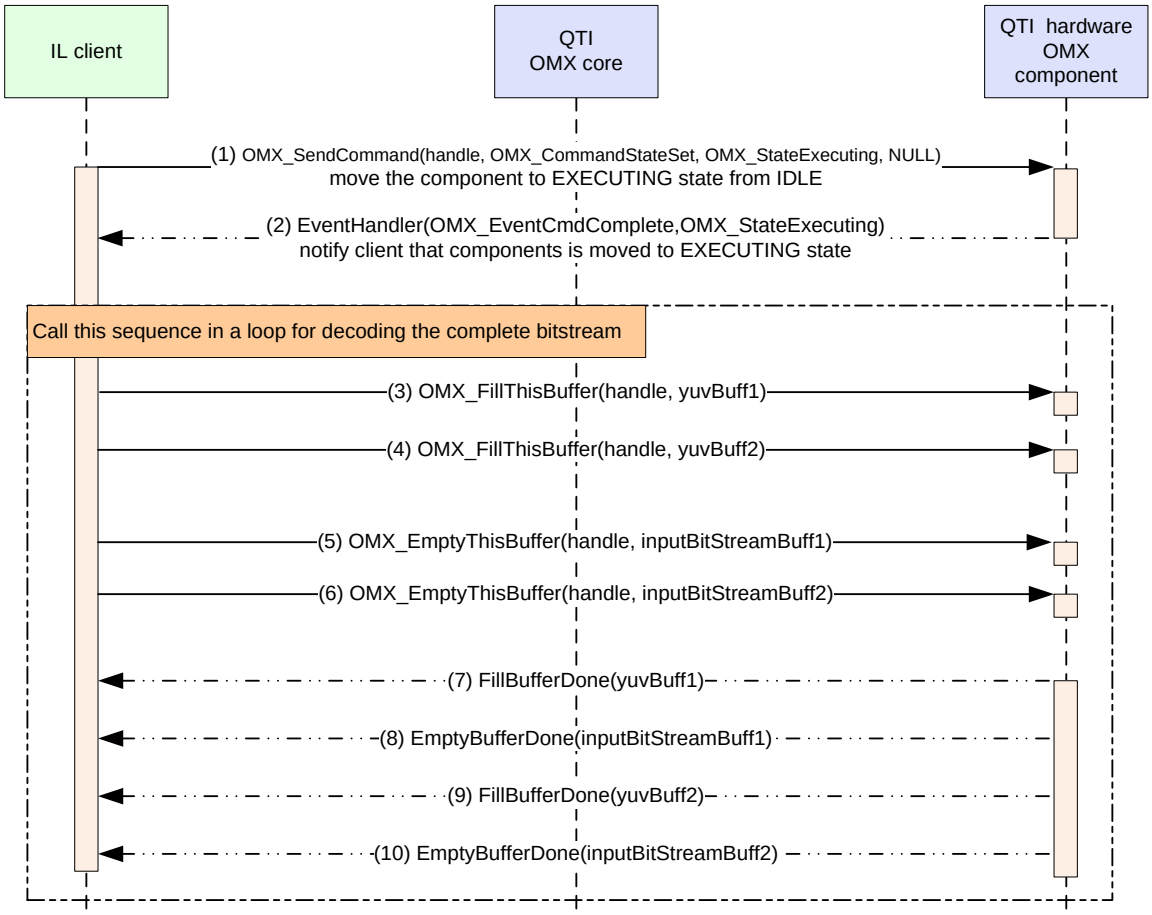


Figure 3-11 Sequence diagram for data flow

The following items correspond to the steps in Figure 3-11 :

- Transition the component from the Idle state to the Executing state (1).
- Wait for OMX_EventCmdComplete (2).
- After the IL client receives the Command Complete event for state transition, it can start sending data to the component.
- Call OMX_FillThisBuffer (3 and 4) with the output buffers that can hold the YUV data.
- Call OMX_EmptyThisBuffer (5 and 6) with the input bitstream data that is to be decoded.
- The component generates the EmptyBufferDone (8 and 10) and FillBufferDone (7 and 9) callbacks to notify the client after processing the data.

3.12 Dynamic port reconfiguration

All clients should be capable of handling the OMX_EventPortSettingsChanged function. The video decoder parses a sequence header and discovers the frame size of the output pictures. If the frame size of the clip does not match the default frame size, buffers associated with its output ports are rearranged.

The default height and width of the component is the QCIF resolution (176 x 144 pixels). When the component detects the correct height and width of the clip, it generates the OMX_EventPortSettingsChanged event.

Figure 3-12 shows the current sequence diagram that the client is to follow.

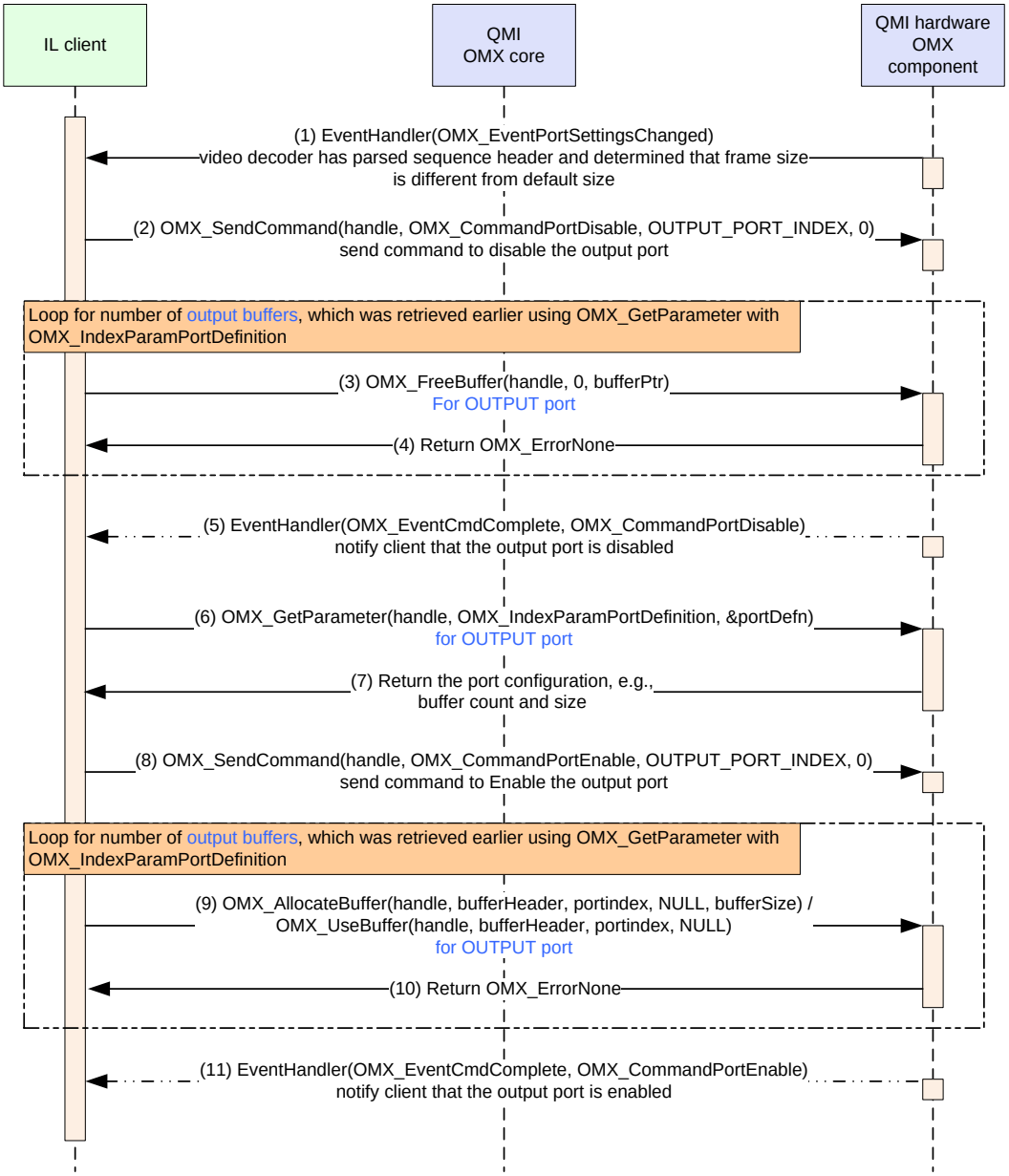


Figure 3-12 Sequence diagram for dynamic port reconfiguration

The following items correspond to the steps in Figure 3-12 :

- The IL client receives OMX_EventPortSettingsChanged (1) when the component parses the sequence header, and the frame size does not match the default frame size of the component.
- Send a command to disable the output port (2).
- Free all buffers on the output port (3).

- The OMX component generates the OMX_EventCmdComplete event (5) to disable the port.
- Because the height and width of the clip are different from the default height and width, the output buffer size and count are changed to hold the different clip dimensions. At this point, the client must get the new configuration of the output port (6 and 7).
- Send a command to enable the output port (8).
- Allocate the buffers with the new configuration on the output port (9).
- When all the buffers are allocated, the component generates the OMX_EventCmdComplete event to enable the port (11).

3.13 Deinitialize the component and OpenMAX core

To deinitialize the OpenMAX core, active components must be freed. To free the component, move it to the Executing→Idle→Loaded state, and then free all the input and output buffers.

Figure 3-13 shows the teardown process of the component and OMX core.

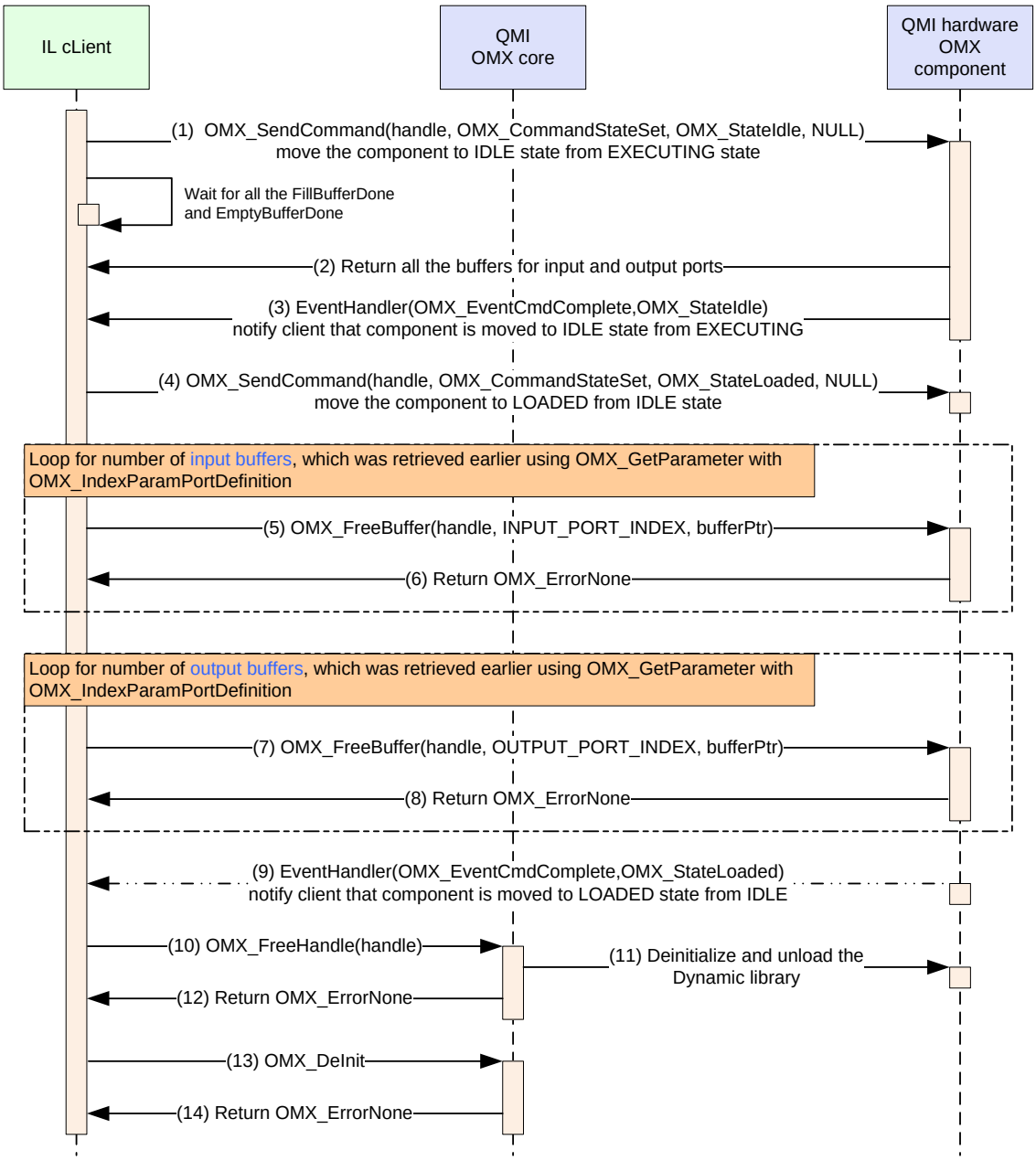


Figure 3-13 Sequence diagram for deinitialization

The following items correspond to the steps in Figure 3-13 :

- Move the component from the Executing state to the Idle state (1).
- Wait for all buffers to be returned by the component.
- The component returns all buffers to the IL client (2).
- The component generates OMX_EventCmdComplete for the Execute→Idle state transition (3).
- Transition the component from the Idle state to the Loaded state (4).
- Free all input and output buffers (5 to 8).
- Wait for OMX_EventCmdComplete for the Idle→Loaded state transition.
- The client receives the command OMX_EventCmdComplete for the Loaded→Idle state transition (9).
- Call OMX_FreeHandle to the OMX core to release the component handle (10).
- Call OMX_DeInit to deinitialize the OMX core (13).

4 Limitations

4.1 Generic limitation for decoders

The following are common limitations of decoders.

- The OMX default color format, OMX_COLOR_FormatYVU420Planar, is not supported. The only color format supported by the decoders is the QTI-defined OMX_QCOM_COLOR_FormatYVU420SemiPlanar.
  - YVU420SemiPlanar – YVU planar format is organized with a first plane containing Y pixels and a second plane containing interleaved V and U pixels. V and U pixels are subsampled by a factor of two, both horizontally and vertically.
- There is no support for changing the decoder configuration during playback.
- The OMX core and codec interfaces are OMX 1.1-based – Backward-compatibility with the OMX 1.0 core and OMX 1.0 IL clients is not supported.

EXHIBIT **1**

PLEASE READ THIS LICENSE AGREEMENT ("AGREEMENT") CAREFULLY. THIS AGREEMENT IS A BINDING LEGAL AGREEMENT ENTERED INTO BY AND BETWEEN YOU (OR IF YOU ARE ENTERING INTO THIS AGREEMENT ON BEHALF OF AN ENTITY, THEN THE ENTITY THAT YOU REPRESENT) AND QUALCOMM TECHNOLOGIES, INC. ("QTI" "WE" "OUR" OR "US"). THIS IS THE AGREEMENT THAT APPLIES TO YOUR USE OF THE DESIGNATED AND/OR ATTACHED DOCUMENTATION AND ANY UPDATES OR IMPROVEMENTS THEREOF (COLLECTIVELY, "MATERIALS"). BY USING OR COMPLETING THE INSTALLATION OF THE MATERIALS, YOU ARE ACCEPTING THIS AGREEMENT AND YOU AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO THESE TERMS, QTI IS UNWILLING TO AND DOES NOT LICENSE THE MATERIALS TO YOU. IF YOU DO NOT AGREE TO THESE TERMS YOU MUST DISCONTINUE AND YOU MAY NOT USE THE MATERIALS OR RETAIN ANY COPIES OF THE MATERIALS. ANY USE OR POSSESSION OF THE MATERIALS BY YOU IS SUBJECT TO THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT.

1.1 **License.** Subject to the terms and conditions of this Agreement, including, without limitation, the restrictions, conditions, limitations and exclusions set forth in this Agreement, Qualcomm Technologies, Inc. ("QTI") hereby grants to you a nonexclusive, limited license under QTI's copyrights to use the attached Materials; and to reproduce and redistribute a reasonable number of copies of the Materials. You may not use Qualcomm Technologies or its affiliates or subsidiaries name, logo or trademarks; and copyright, trademark, patent and any other notices that appear on the Materials may not be removed or obscured. QTI shall be free to use suggestions, feedback or other information received from You, without obligation of any kind to You. QTI may immediately terminate this Agreement upon your breach. Upon termination of this Agreement, Sections 1.2-4 shall survive.

1.2 **Indemnification.** You agree to indemnify and hold harmless QTI and its officers, directors, employees and successors and assigns against any and all third party claims, demands, causes of action, losses, liabilities, damages, costs and expenses, incurred by QTI (including but not limited to costs of defense, investigation and reasonable attorney's fees) arising out of, resulting from or related to: (i) any breach of this Agreement by You; and (ii) your acts, omissions, products and services. If requested by QTI, You agree to defend QTI in connection with any third party claims, demands, or causes of action resulting from, arising out of or in connection with any of the foregoing.

1.3 **Ownership.** QTI (or its licensors) shall retain title and all ownership rights in and to the Materials and all copies thereof, and nothing herein shall be deemed to grant any right to You under any of QTI's or its affiliates' patents. You shall not subject the Materials to any third party license terms (e.g., open source license terms). You shall not use the Materials for the purpose of identifying or providing evidence to support any potential patent infringement claim against QTI, its affiliates, or any of QTI's or QTI's affiliates' suppliers and/or direct or indirect customers. QTI hereby reserves all rights not expressly granted herein.

1.4 **WARRANTY DISCLAIMER.** YOU EXPRESSLY ACKNOWLEDGE AND AGREE THAT THE USE OF THE MATERIALS IS AT YOUR SOLE RISK. THE MATERIALS AND TECHNICAL SUPPORT, IF ANY, ARE PROVIDED "AS IS" AND WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED. QTI ITS LICENSORS AND AFFILIATES MAKE NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE MATERIALS OR ANY OTHER INFORMATION OR DOCUMENTATION PROVIDED UNDER THIS AGREEMENT, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR AGAINST INFRINGEMENT, OR ANY EXPRESS OR IMPLIED WARRANTY ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. NOTHING CONTAINED IN THIS AGREEMENT SHALL BE CONSTRUED AS (I) A WARRANTY OR REPRESENTATION BY QTI, ITS LICENSORS OR AFFILIATES AS TO THE VALIDITY OR SCOPE OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT OR (II) A WARRANTY OR REPRESENTATION BY QTI THAT ANY MANUFACTURE OR USE WILL BE FREE FROM INFRINGEMENT OF PATENTS, COPYRIGHTS OR OTHER INTELLECTUAL PROPERTY RIGHTS OF OTHERS, AND IT SHALL BE THE SOLE RESPONSIBILITY OF YOU TO MAKE SUCH DETERMINATION AS IS NECESSARY WITH RESPECT TO THE ACQUISITION OF LICENSES UNDER PATENTS AND OTHER INTELLECTUAL PROPERTY OF THIRD PARTIES.

1.5 **LIMITATION OF LIABILITY.** IN NO EVENT SHALL QTI, QTI'S AFFILIATES OR ITS LICENSORS BE LIABLE TO YOU FOR ANY INCIDENTAL, CONSEQUENTIAL OR SPECIAL DAMAGES, INCLUDING BUT NOT LIMITED TO ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL DAMAGES, ARISING OUT OF THE USE OR INABILITY TO USE, OR THE DELIVERY OR FAILURE TO DELIVER, ANY OF THE MATERIALS, OR ANY BREACH OF ANY OBLIGATION UNDER THIS AGREEMENT, EVEN IF QTI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE FOREGOING LIMITATION OF LIABILITY SHALL REMAIN IN FULL FORCE AND EFFECT REGARDLESS OF WHETHER YOUR REMEDIES HEREUNDER ARE DETERMINED TO HAVE FAILED OF THEIR ESSENTIAL PURPOSE. THE ENTIRE LIABILITY OF QTI, QTI's AFFILIATES AND ITS LICENSORS, AND THE SOLE AND EXCLUSIVE REMEDY OF YOU, FOR ANY CLAIM OR CAUSE OF ACTION ARISING HEREUNDER (WHETHER IN CONTRACT, TORT, OR OTHERWISE) SHALL NOT EXCEED US$10.

2. **COMPLIANCE WITH LAWS; APPLICABLE LAW.** You agree to comply with all applicable local, international and national laws and regulations and with U.S. Export Administration Regulations, as they apply to the subject matter of this Agreement. This Agreement is governed by the laws of the State of California, excluding California's choice of law rules.

3. **CONTRACTING PARTIES.** If the Materials are downloaded on any computer owned by a corporation or other legal entity, then this Agreement is formed by and between QTI and such entity. The individual accepting the terms of this Agreement represents and warrants to QTI that they have the authority to bind such entity to the terms and conditions of this Agreement.

4. **MISCELLANEOUS PROVISIONS.** This Agreement, together with all exhibits attached hereto, which are incorporated herein by this reference, constitutes the entire agreement between QTI and You and supersedes all prior negotiations, representations and agreements between the parties with respect to the subject matter hereof. No addition or modification of this Agreement shall be effective unless made in writing and signed by the respective representatives of QTI and You. The restrictions, limitations, exclusions and conditions set forth in this Agreement shall apply even if QTI or any of its affiliates becomes aware of or fails to act in a manner to address any violation or failure to comply therewith. You hereby acknowledge and agree that the restrictions, limitations, conditions and exclusions imposed in this Agreement on the rights granted in this Agreement are not a derogation of the benefits of such rights. You further acknowledges that, in the absence of such restrictions, limitations, conditions and exclusions, QTI would not have entered into this Agreement with You. Each party shall be responsible for and shall bear its own expenses in connection with this Agreement. If any of the provisions of this Agreement are determined to be invalid, illegal, or otherwise unenforceable, the remaining provisions shall remain in full force and effect. This Agreement is entered into solely in the English language, and if for any reason any other language version is prepared by any party, it shall be solely for convenience and the English version shall govern and control all aspects. If You are located in the province of Quebec, Canada, the following applies: The Parties hereby confirm they have requested this Agreement and all related documents be prepared in English.