DecodeCore™

User Guide

Document # 11-141-10492

FEBRUARY 22, 2019

Notice

This document and the information contained herein are the property of Core Avionics & Industrial Inc. Any reproduction, disclosure or use thereof is prohibited except as authorized in writing by Core Avionics & Industrial Inc. Recipient accepts the responsibility for maintaining the confidentiality of the contents of this document.

Core Avionics & Industrial Inc.
www.coreavi.com

[redacted]

[redacted]				
[redacted]			[redacted]	[redacted]
[redacted]		[redacted]	[redacted]	[redacted]
[redacted]			[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]			[redacted]	[redacted]
[redacted]		[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

Table of Contents

[redacted]

List of Figures

1. Introduction

1.1. Purpose

This document presents the DecodeCore API. This document describes at a high level the architecture of the software, drivers, and hardware interface to decode video files and streaming video via GPU-based video decoding. This design currently provides support for the H.264 video standard as described in this user guide.

Hardware video decoding allows users to play encoded video files and streams with minimal CPU overhead. The video frames are decoded by the GPU and then exposed to the application as OpenGL® 2D textures, allowing the application to manipulate the decoded video using standard OpenGL texturing functionality.

1.2. References

ITU-T H.264 AVC Spec (freely available in .pdf)
http://www.itu.int/rec/T-REC-H.264-201201-I/en

VDPAU source code and header files
http://cgit.freedesktop.org/~aplattner/libvdpau

VDPAU documentation
ftp://download.nvidia.com/XFree86/vdpau/doxygen/html/index.html

DXVA H.264 documentation
http://www.microsoft.com/en-us/download/details.aspx?id=11323

1.3. Acronyms and Abbreviations

Acronym	Expanded
API	Application Programming Interface
CPU	Central Processing Unit
FOC	Field Order Count; synonymous with POC
GART	Graphics Address Remapping Table
GPU	Graphics Processing Unit
MB	Mega Bytes
MBAFF	Macroblock Adaptive Field Frame
NAL	Network Abstraction Layer
NV12	Video format for YUV420 planar data
OS	Operating System
PAFF	Picture Adaptive Field Frame
PCI	Peripheral Component Interconnect
PCIe	Peripheral Component Interconnect Express
POC	Picture Order Count; synonymous with FOC

PPS	Picture Parameter Set
RAM	Random Access Memory
RGBA	Red Green Blue Alpha
RTOS	Real Time Operating System
RTP	Real Time Protocol
SPS	Sequence Parameter Set
VDPAU	Video Decode and Presentation API for Unix.
VRAM	Video RAM

2. DecodeCore Overview

2.1. System Overview

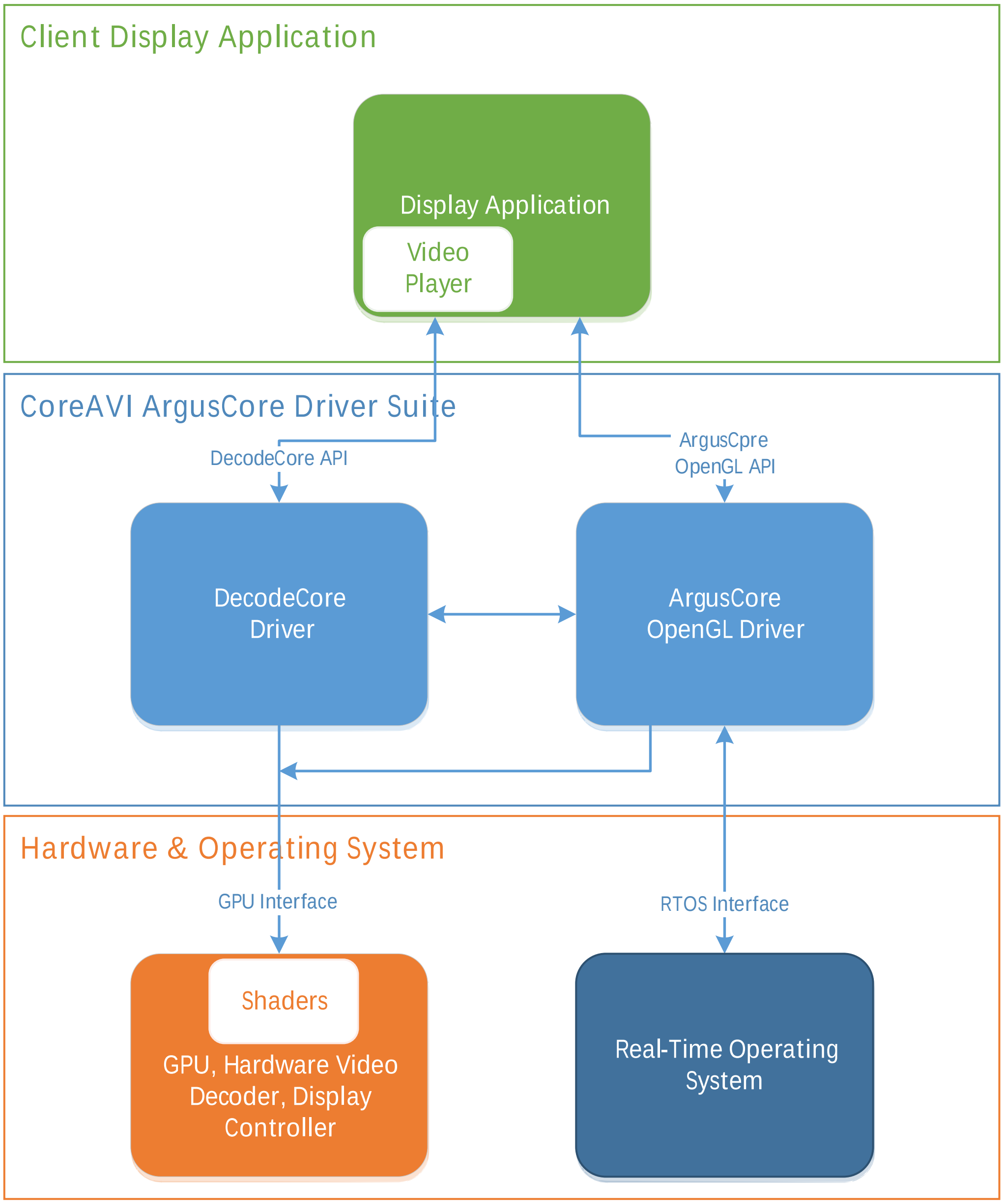
Client Display Application
Display Application
Video Player
CoreAVI ArgusCore Driver Suite
DecodeCore API
ArgusCpre OpenGL API
DecodeCore Driver
ArgusCore OpenGL Driver
Hardware & Operating System
GPU Interface
RTOS Interface
Shaders
GPU, Hardware Video Decoder, Display Controller
Real-Time Operating System

Figure 1 DecodeCore - System Perspective

2.2. Driver Software Requirements

- The video decode driver shall be thread safe and shall support up to 10 video decode contexts simultaneously. The GPU-specific hardware decoder will impose additional limits such as maximum video resolution, maximum number of concurrent streams, etc. Please see the GPU-specific user guide for these details.
- The video decode driver shall perform a built-in-test during GPU initialization to ensure that the video decode hardware is operational.
- The video decode driver shall provide a method to initialize and destroy video decode contexts.
- The video decode driver shall support the decompression of H.264 video bitstream data.
- The video decode driver shall work on video “slice” data.

2.3. Client Application Requirements

The video decode driver works in tandem with the ArgusCore™ OpenGL drivers. The application must use an ArgusCore OpenGL API in conjunction with the DecodeCode API in order to display decoded video.

The video decode driver accepts only "slice" data from the bitstream and performs all required processing of those slices (e.g. VLD decoding, IDCT, motion compensation, in-loop deblocking, etc.).

The client application is responsible for:

- Extracting the raw H.264 stream, if the video stream is encapsulated in a container (e.g. .avi, .mkv, etc.) or RTP.
- Parsing the H.264 stream’s NAL units, headers, and metadata (e.g. sequence parameter set, picture parameter set, slice headers, etc.) Various fields from the parsed structures need to be provided to the DecodeCore API alongside the slice’s raw bitstream data and size.
- Tracking the list of reference frames using information derived from the NAL units.
- Using OpenGL texturing to display the decoded video.

[redacted]

3. DecodeCore Driver APIs

This section describes the custom CoreAVI DecodeCore driver API enumerations, structures, and functions. For specifics on individual structures, please refer to the corresponding header file.

[redacted]

[redacted]

[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]

[redacted]

[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

4. DecodeCore Setup

4.1. Initialization

The DecodeCore driver only works with an ArgusCore OpenGL driver, so much of the initialization is done via the same mechanism used to initialize OpenGL. [redacted] See the ArgusCore OpenGL User Guide for more details on driver and GPU initialization.

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

4.3. Context Creation

Once the driver, GPU, and OpenGL are initialized then a video decode context can be created. [redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]