SGX EGL 1.4 Reference Driver

Software Architecture Specification

Filename	:	SGX EGL 1.4 Reference Driver. Software Architecture Specification.doc
Version	:	1.0.5a External Issue
Issue Date	:	14 Jul 2008
Author	:	PowerVR

Contents

1. Introduction 4
1.1. Scope 4
1.2. Related Documents 4
2. Product Overview 5
2.1. Goals and Objectives 5
2.2. Product Environment 5
2.3. Assumptions, Dependencies and Constraints 5
2.3.1. Assumptions 5
2.3.2. Constraints 5
3. Architecture Specification 6
3.1. Overview 6
3.2. Detailed Component Description 8

List of Figures

Figure 1 Architecture Overview 6
Figure 2 EGL 1.4 Driver Architecture Overview 8

Glossary of Terms

Table 1

Khronos Group	Body that controls the OpenGL ES, OpenVG and OpenMAX specifications
EGL 1.4	The Native Platform Graphics Interface implemented for this project.
OpenGL ES™ 1.1	The fixed function 3D API interfacing with EGL. OpenGL ES is a trademark of Silicon Graphics, Inc.
OpenGL ES™ 2.0	The programmable 3D API implemented for this project OpenGL ES is a trademark of Silicon Graphics, Inc.
OpenVG™ 1.x	The vector graphics 3D API interfacing with EGL OpenVG is a trademark of The Khronos Group, Inc.
SGX	IP that provides 2D/3D/Video acceleration

1. Introduction

1.1. Scope

This document provides a top-level view of the EGL software architecture for SGX with a brief description of the major architectural components. It is assumed that the reader is already familiar with the hardware functionality of SGX and its derivatives.

1.2. Related Documents

OpenVG 1.0 Specification, Khronos Group, Version 1.0
OpenGL ES 1.1 Specification, Khronos Group, Version 1.1.04
OpenGL ES 2.0 Specification, Khronos Group, Version 1.03
EGL 1.4 Specification, Khronos Group, Version 1.0 (May 2008)

2. Product Overview

The EGL 1.4 component delivers a reference EGL 1.4 driver for SGX and variants, but as this cannot be developed in isolation of its environment, the driver will also require the presence of other components for services functionality and 2D functionality.

2.1. Goals and Objectives

The major high-level design goals for this project's software architecture are detailed below.

- Support for all client APIs allowed in EGL 1.4, namely OpenGL ES 2.0, OpenGL ES 1.1 and OpenVG.
- Platform portability. The code must be easily ported to different operating systems potentially including Linux, Symbian, WinCE, others.
- Does not impede performance.

2.2. Product Environment

The product is designed to support the SGX hardware and variants including SGX 520, 530, 535 with MMU. The EGL 1.4 software component will be developed in a reference linux environment.

2.3. Assumptions, Dependencies and Constraints

2.3.1. Assumptions

- None.

2.3.2. Constraints

- The EGL 1.4 schedule is aggressive; any chosen architecture must be very low risk. This then requires re-use of successfully integrated components from the target environment as far as possible.

3. Architecture Specification

3.1. Overview

The following diagram 'Figure 1 Architecture Overview' details a top-level overview of the architecture of the software stack, including the EGL 1.4 reference driver's position within it.

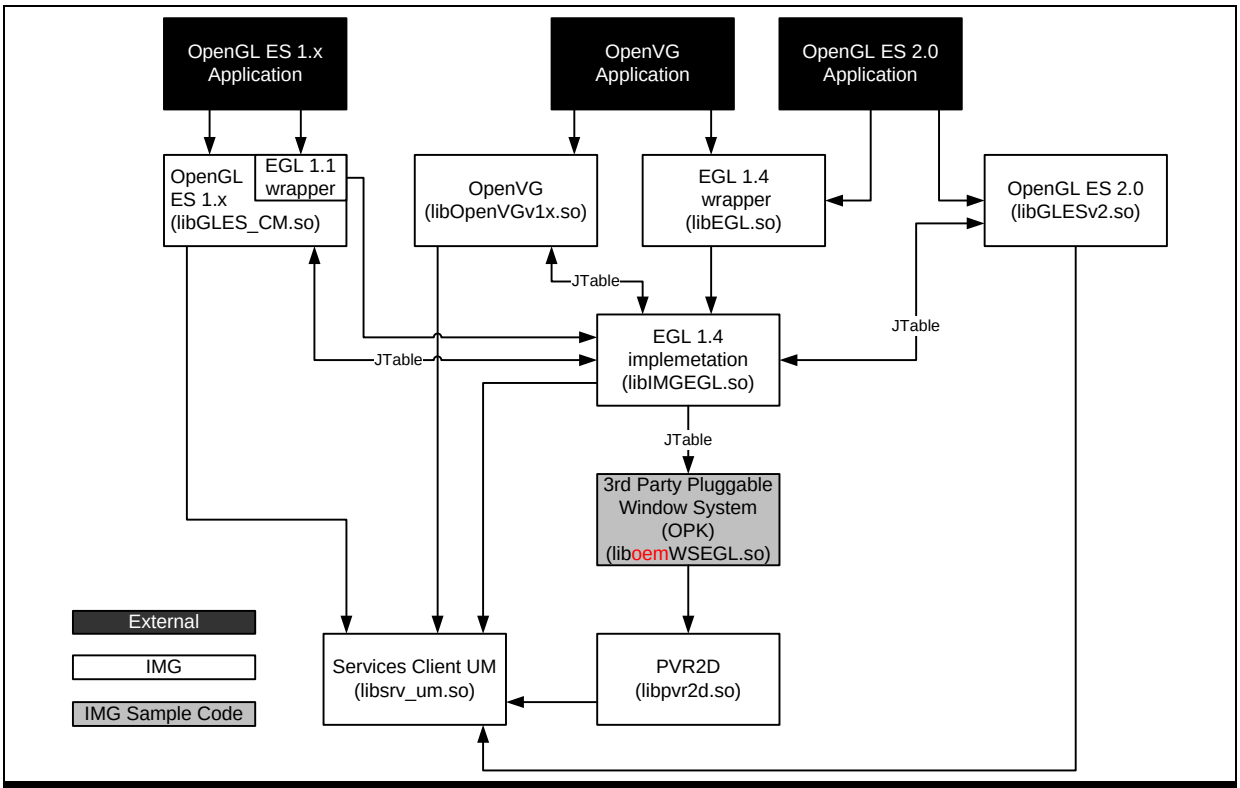

Figure 1 Architecture Overview

Elements in grey are 3rd party components for which example code will be supplied

The architecture chosen meets the design goals defined in section 2.1 (see below for details).

- Proven architecture. The basic architecture chosen has been successfully implemented on a variety of platforms including WinCE, Symbian, Linux and uiTRON. This provides a very low risk design. The architecture re-uses substantial amounts of proven code from the MBX EGL 1.2 code base.
- Platform portable. Platform specific code abstracted to services components to simplify porting to another platform.

EGL 1.4 wrapper	
Component Type	Library (libEGL)
Purpose	Provides the EGL 1.4 API for applications binding directly to the libEGL module.
Functionality	Provides a thin transport layer to the internal EGL implementation (libIMGEGL), marking any relevant calls has having come from the EGL specific (libEGL) module.
External Interfaces	This component provides a 'C' interface as defined by the EGL 1.4 Specification as defined by the Khronos group.
Dependencies and Inter-relationships with other major components	This component interfaces with the internal EGL implementation (libIMGEGL) through a function list closely resembling the EGL 1.4 interface.

Table 2

EGL 1.1 wrapper	
Component Type	Portion of Library (libGLES_CM)
Purpose	Provides the legacy EGL 1.1 API through the legacy OpenGL ES 1.1 module.
Functionality	Provides a thin transport layer to the internal EGL implementation (libIMGEGL), marking any relevant calls has having come from the legacy OpenGL ES 1.1 (libGLES_CM) module.
External Interfaces	This component provides a 'C' interface as defined by the EGL 1.1 Specification as defined by the Khronos group.
Dependencies and Inter-relationships with other major components	This component interfaces only with the internal EGL implementation (libIMGEGL), through a function list closely resembling the EGL 1.1 interface, although it is a part of the OpenGL ES 1.1 module (libGLES_CM).

Table 3

Internal EGL Implementation	
Component Type	Library (libIMGEGL)
Purpose	Allows both legacy EGL 1.1 and future compatibility EGL 1.4 to be layered on a single functional module.
Functionality	Provides the implementation of EGL, allowing for difference in EGL API version, via the thin layered modules described above. Deals with configurations, creation of surfaces and contexts, and binding them together.
External Interfaces	This component provides a 'C' interface similar to the one defined by the EGL 1.4 Specification as defined by the Khronos group. This interface allows for differentiation to be supported between EGL 1.1 and EGL1.4

Internal EGL Implementation	
Dependencies and Inter-relationships with other major components	This component interfaces with the EGL 1.1 and 1.4 thin layers. It also provides functionality to the client API drivers (OpenVG/OpenGL ES) and gets a jump table of functions from them. This component interfaces with the WSEGL (window system integration) module for window system specific information about surfaces, and with the services to get support for SGX allocation and management routines.

Table 4

3.2. Detailed Component Description

The following diagram 'Figure 2 EGL 1.4 Driver Architecture Overview' details an overview of the architecture of the EGL 1.3 reference driver.

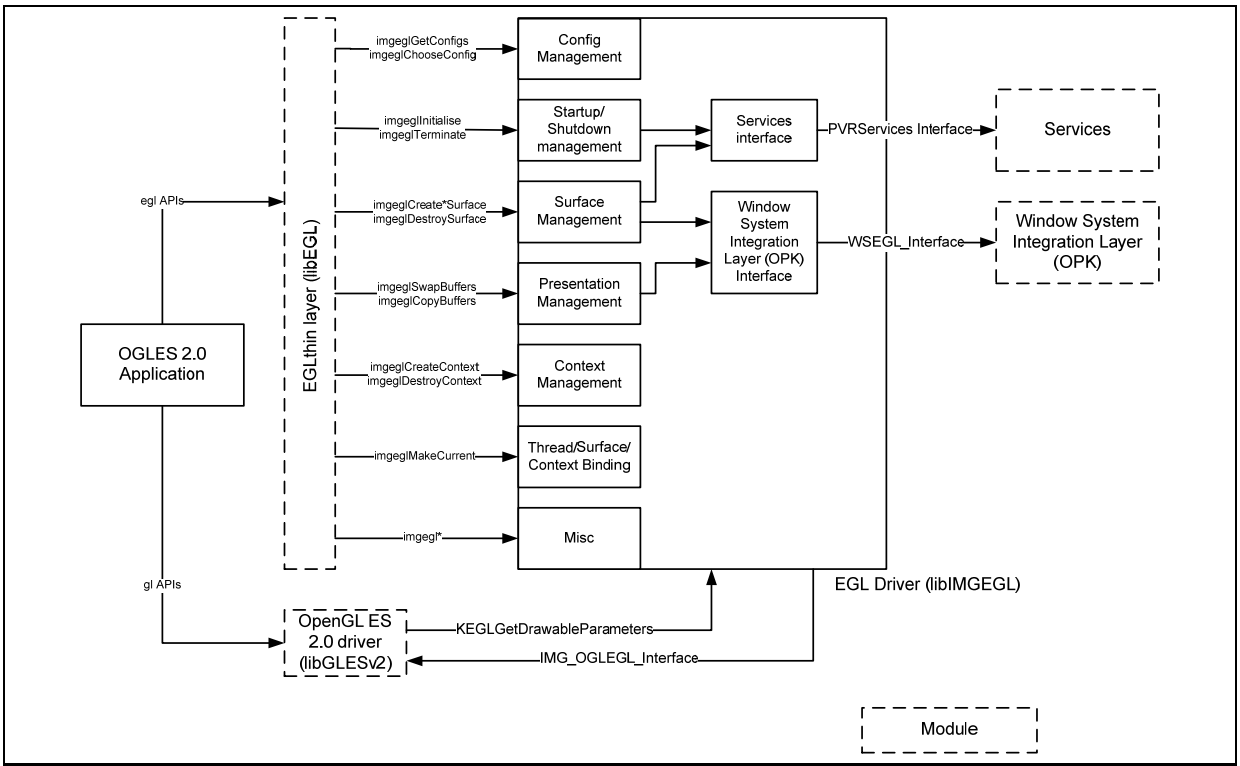

Figure 2 EGL 1.4 Driver Architecture Overview

Config Management	
Component Type	C Code
Functionality	Defines supported configs, and integrates these with WSEGL support. Provides config sorting, attribute querying and config compatibility checks.
External Interfaces	None
Internal Interfaces	None

Config Management	
Dependencies and Inter-relationships with other major components	This component interfaces with the WSEGL component to get information about the WSEGL config support.

Table 5

Startup/Shutdown Management	
Component Type	C Code
Functionality	Performs initialisation and deinitialisation of services constructs, such as memory contexts and render contexts, and also manages thread storage.
External Interfaces	None
Internal Interfaces	None
Dependencies and Inter-relationships with other major components	This component interfaces with the services to initialise/deinitialise devices, contexts etc.

Table 6

Surface Management	
Component Type	C Code
Functionality	Allocates and deallocates support structures for render surfaces. Interacts with WSEGL module to create WSEGL representation of a surface (including back buffers). Provides support for pbuffer surfaces and gives client APIs drawable parameter information.
External Interfaces	None
Internal Interfaces	None
Dependencies and Inter-relationships with other major components	This component interfaces with the WSEGL module to create/destroy surface's backbuffers. Provides information about the current readback surface's attributes (size, format etc) to client API modules.

Table 7

Presentation Management	
Component Type	C Code
Functionality	Orchestrates flushing of renders and presentation of results.
External Interfaces	None
Internal Interfaces	None
Dependencies and Inter-relationships with other major components	This component interfaces with the client APIs to perform flushing, and interfaces with the WSEGL module to perform presentation.

Table 8

Context Management	
Component Type	C Code
Functionality	Calls relevant client APIs for context creation/ destruction. Provides thread support for currently bound context.
External Interfaces	None
Internal Interfaces	None
Dependencies and Inter-relationships with other major components	This component interfaces with the client API modules to perform client specific context creation/destruction.

Table 9

Thread/Surface/Context Binding	
Component Type	C Code
Functionality	Brings contexts, surfaces and threads together. Checks for valid combinations of these entities and destroys them as necessary when unbinding.
External Interfaces	None
Internal Interfaces	None
Dependencies and Inter-relationships with other major components	This component interfaces with the client APIs to make surfaces and contexts current/uncurrent.

Table 10

Misc	
Component Type	C Code
Functionality	Supports miscellaneous EGL functionality, including error handling, getting of EGL state, waiting for native/client rendering etc.
External Interfaces	None
Internal Interfaces	None
Dependencies and Inter-relationships with other major components	n/a

Table 11