SGX Services 4

Software Architectural Specification

Filename	:	SGX Services 4.Software Architectural Specification.doc
Version	:	4.0.42 External Issue
Issue Date	:	08 Dec 2009
Author	:	PowerVR

Contents

List of Figures

1. Introduction

1.1. Scope

This document provides a top-level view of the Consumer Services software architecture with a brief description of the major architectural components. It is assumed that the reader has a basic knowledge of driver development.

1.2. Related Documents

Services 4.0 Software Functional Specification
Services Porting Guide for Services 4.0
Consumer Services 3rd Party Display API

1.3. Nomenclature

Due to the over use of the word *platform* within the consumer projects and industry, it was decided to define terminology that would be clear and yet not conflict with general external use. Thus the following is recommended:

Term	Description
API:	D3D, OpenGL, OpenGL ES, OpenVG, OpenCL, DirectDraw…
Components:	Delivered files for each API or Tool.
Build:	Name should include components that uniquely describe the build: <OS> + <Device> + <platform or system> e.g. Linux_SGX535_FPGAPC
Device:	SGX535, SGX530, MSVDX, PDP … AKA Core
Environment:	Linux, Vista, WinXP, WinCE, WinMobile, Symbian, uITRON… AKA OS
System:	Typically the customers project name, e.g. LongRun, SilverFish… AKA SoC.
Platform:	Nokia N93… - real physical item - no files or directory.

2. Product Overview

The components of the Consumer Services are entirely software. The Consumer Services is intended for use across as many consumer projects as possible, limited only by the scope and flexibility of its overall design.

2.1. Goals and Objectives

The major high-level design goals of the software architecture are detailed below.

- Reduce time and effort in development of software for new consumer projects
- Increase code re-use and stability
- Flexibility in terms of porting to new platforms and environments
- Does not affect performance compared to a `one-off' solution
- Can be used in the place of the Services v3.0
- Does not affect performance when compared to the Services v3.0
- Provide Services GPL kernel driver

2.2. Product Environment

Consumer Services is designed to support all current and future embedded devices, including 3rd party devices that must cooperate with other devices on a given platform. Consumer Services must support a wide range of different platforms incorporating numerous combinations of devices, CPUs and environments.

2.3. Constraints

The Consumer Services are designed to operate in the most constrained environment, and where any of these constraints do not apply, compile options for these more `relaxed' environments may allow simpler and/or faster operation. Some of the constraints affecting the design and operation of the Consumer Services are as follows:

2.3.1. Kernel Space Operation

In many environments correct operation of the Consumer Services will require that they operate in privileged mode or the equivalent in that environment, this will be referred to as Kernel mode.

2.3.2. Kernel-User Separation

Address spaces may be different in the Kernel and Client Services. This requires that the Services APIs be designed to allow optional separation of these address spaces.

2.3.3. Multi-Process Operation

The Consumer Services should be designed to provide functionality to multiple user space processes at the same time. In some environments these processes will be operating in their own individual address spaces and as such access to common resources must be managed via constructs provided by the Consumer Services APIs.

3. Architecture Specification

3.1. Overview

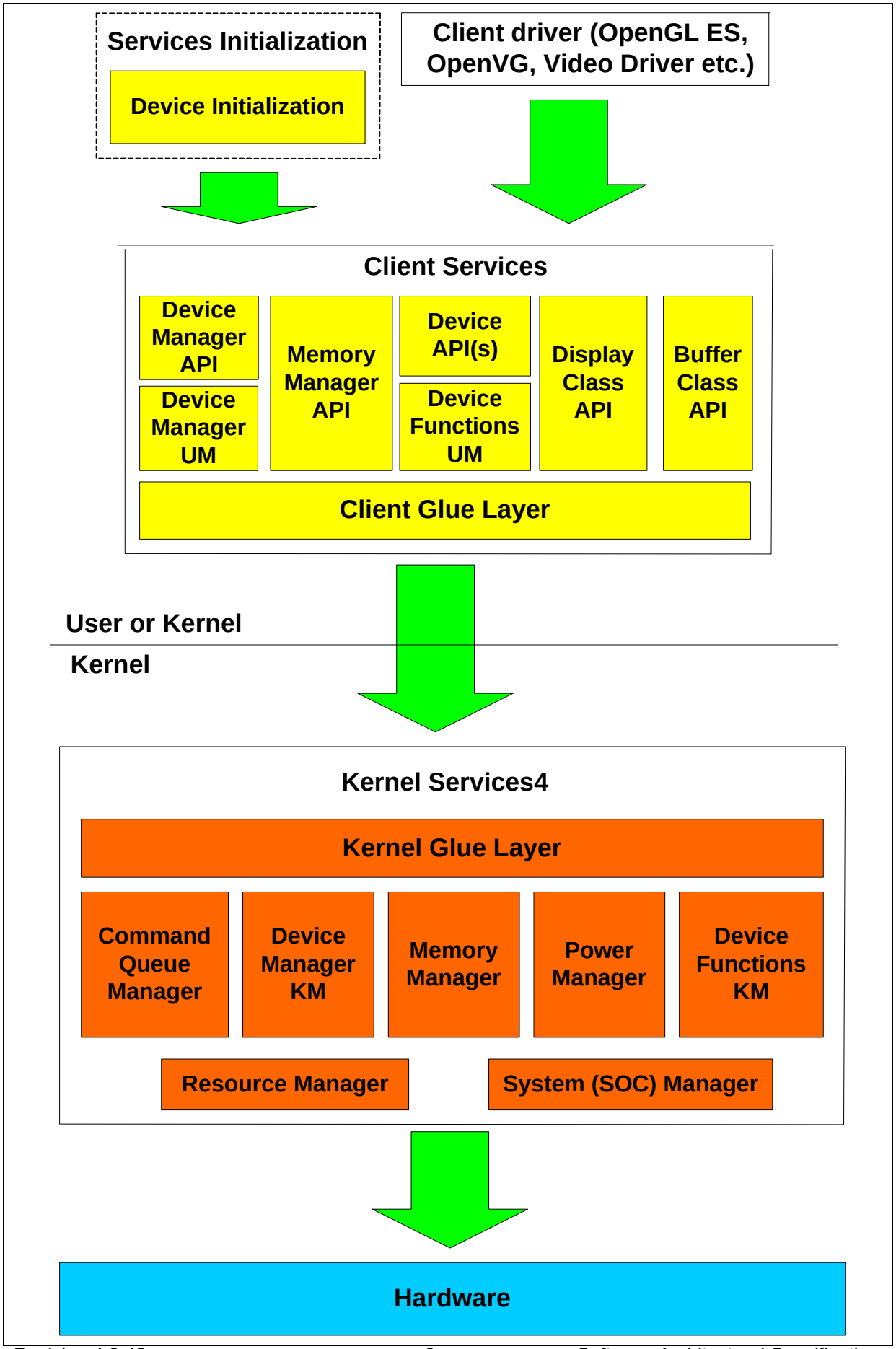
Services Initialization
Device Initialization
Client driver (OpenGL ES, OpenVG, Video Driver etc.)
Client Services
Device Manager API
Device Manager UM
Memory Manager API
Device API(s)
Device Functions UM
Display Class API
Buffer Class API
Client Glue Layer
User or Kernel
Kernel
Kernel Services4
Kernel Glue Layer
Command Queue Manager
Device Manager KM
Memory Manager
Power Manager
Device Functions KM
Resource Manager
System (SOC) Manager
Hardware

Figure 1 Architectural view of Consumer Services Software Components

Figure 1 details a high-level architectural overview of the Consumer Services software components.

The architecture chosen represents a typical layered approach.

Note: Figure 1 incorporates a user/kernel divide that may or may not apply depending on the platform.

The Consumer Services architectural design is the result of addressing the goals and objectives laid out in section 2.1. The following elaborates on how these goals have been addressed in the Services design:

- 'Reduce time and effort in development of software for new consumer projects' - the architectural design limits the scope of Services knowledge required to integrate new platforms.
- 'Increase code re-use and stability' - large amounts of the services code base are truly common and unchanged as new platforms are added.
- 'Flexibility in terms of porting to new platforms and environments' - the architecture makes few assumptions about the nature of new environments, systems and devices.
- 'Does not affect performance compared to a 'one-off' solution' - it is not necessary for 'critical' functions to bore through multiple abstraction layers.

3.2. Detailed Component Description

The following tables detail various attributes of the consumer services software components described in Figure 1.

3.2.1. User Components

Device Manager API	
Component Type	Built as part of client driver
Purpose	Provides clients with access to devices managed by consumer services
Functionality	A thin component providing an external API for the device manager component (see Device Manager for details)
External Interfaces	Provides a 'C' interface as defined by the Consumer Services external API header, see Software Function Specification
Internal Interfaces	None
Dependencies and Inter-relationships with other major components	Device Manager Component (All functions call through client/kernel glue)

Device Manager UM	
Component Type	Built as part of client services
Purpose	Provides clients with access to devices managed by consumer services device integration and control
Functionality	Provides an abstraction framework that allows other components to control different devices in a consistent manner. Responsible for device enumeration and all aspects of device initialisation.
External Interfaces	None
Internal Interfaces	Services the Device Manager API

Device Manager UM	
Dependencies and Inter-relationships with other major components	Device Manager API, Device Manager KM

Memory Manager API	
Component Type	Built as part of client driver
Purpose	Provides clients with access to device addressable memory management functions in consumer services
Functionality	A thin component providing an external API for the memory manager component (see Memory Manager for details)
External Interfaces	Provides a 'C' interface as defined by the Consumer Services external API header, see Software Function Specification
Internal Interfaces	None
Dependencies and Inter-relationships with other major components	Memory Manager Component (All functions call through client/kernel glue)

Device API(s)	
Component Type	Built as part of client driver
Purpose	Provides clients with access to device specific functionality of devices managed by consumer services.
Functionality	The functionality is dependent on the device type. APIs may be thin components calling through to device kernel services or self-contained functions within user services (or any combination of both).
External Interfaces	Provides a 'C' interface as defined by the Consumer Services external API header, see Software Function Specification
Internal Interfaces	None
Dependencies and Inter-relationships with other major components	Device Functions Component (APIs may or may not call through to kernel Device Functions)

Device Functions UM	
Component Type	Built as part of client services
Purpose	Provides clients with access to device specific functionality of devices managed by consumer services.
Functionality	The functionality is dependent on the device type. This component provides optional user mode services device functions
External Interfaces	None.
Internal Interfaces	Services the Device API component.

Device Functions UM	
Dependencies and Inter-relationships with other major components	Device API, Device Functions KM

Display Class API	
Component Type	Built as part of client driver
Purpose	Provides clients with access to display devices
Functionality	Provides control of display hardware and display addressable memory.
External Interfaces	Provides a 'C' interface as defined by the Consumer Services external API header, see Software Function Specification
Internal Interfaces	See 3rd Party Display Class API documentation
Dependencies and Inter-relationships with other major components	Device Manager and Memory Manager Components (All functions call through client/kernel glue)

Buffer Class API	
Component Type	Built as part of client driver
Purpose	Provides clients with access to 'buffer class' type devices
Functionality	Provides control of buffer class hardware and associated resources.
External Interfaces	Provides a 'C' interface as defined by the Consumer Services external API header, see Software Function Specification
Internal Interfaces	See 3rd Party Buffer Class API documentation
Dependencies and Inter-relationships with other major components	Device Manager and Memory Manager Components (All functions call through client/kernel glue)

3.2.2. Services Initialization

Services Initialization	
Component Type	Built as separate component
Purpose	Initializes devices managed by the consumer services
Functionality	A thin component performing second step of services initialization
External Interfaces	None
Internal Interfaces	None
Dependencies and Inter-relationships with other major components	Services Client Component (All functions call through services client)

3.2.3. Kernel Components

Command Queue Manager	
Component Type	Built as part of kernel services
Purpose	Provides asynchronous software command scheduling
Functionality	Provides Command Queue create, destroy, insert command and process command functionality
External Interfaces	None
Internal Interfaces	Kernel Services Display Class (Flip) code as well as L/MISRs for command processing
Dependencies and Inter-relationships with other major components	Resource Manager Component

Device Manager KM	
Component Type	Built as part of kernel services
Purpose	Provides clients with access to devices managed by consumer services as well as handling 3rd Party device integration and control
Functionality	Provides an abstraction framework that allows other components to control different devices in a consistent manner. Responsible for device enumeration and all aspects of device initialisation.
External Interfaces	None
Internal Interfaces	Services the Device Manager UM via client/kernel glue
Dependencies and Inter-relationships with other major components	Resource Manager Component

Memory Manager KM	
Component Type	Built as part of kernel services
Purpose	Provides clients with access to device ‘addressable’ memory management functions in consumer services
Functionality	Manages all types of device addressable memory mapping and allocation
External Interfaces	None
Internal Interfaces	Services the Memory Manager API via client/kernel glue as well as other kernel services components.
Dependencies and Inter-relationships with other major components	Resource Manager Component

Power Manager	
Component Type	Built as part of kernel services
Purpose	Provides clients with the ability to be power aware, responding appropriately to power events from the OS.
Functionality	Provides a consistent framework for devices' power to be managed effectively
External Interfaces	None
Internal Interfaces	Services the OS power event function call-back as well as device power interfaces.
Dependencies and Inter-relationships with other major components	Resource Manager Component

Device Functions KM	
Component Type	Built as part of kernel services
Purpose	Provides clients with access to device specific functionality of devices managed by consumer services.
Functionality	The functionality is dependent on the device type. This component provides optional kernel services device functions
External Interfaces	Services the Device Function UM component via client/kernel glue.
Internal Interfaces	None
Dependencies and Inter-relationships with other major components	Resource Manager Component, Buffer Manager Component

Resource Manager	
Component Type	Built as part of kernel services
Purpose	Provides hardware and software resource management for consumer services
Functionality	The functionality is dependent on the device type. This component provides optional kernel services device functions
External Interfaces	None
Internal Interfaces	Resource Manager initialisation and resource registration/recovery
Dependencies and Inter-relationships with other major components	

System (SOC) Manager	
Component Type	Built as part of kernel services
Purpose	Provides an API framework for controlling underlying system/SOC

System (SOC) Manager	
Functionality	Handles system initialisation/de-initialisation, SOC power control
External Interfaces	None
Internal Interfaces	Coordinates other kernel-side components
Dependencies and Inter-relationships with other major components	All other kernel-side components