CSSD DESIGN SPECIFICATION

Nucleus® LOCOSTO

Document Revision: 0.1

Issue Date: 10 May 2006

MakingWireless

Table of Contents

List of Figures

List of Tables

Revision History

REV	DATE	AUTHOR	NOTES
0.1	10th May 2006	J Raghuram Karthik	First Version

Approvals

REV	APPROVAL 1	DATE	APPROVAL 2	DATE
0.1	Rajan Narendran		Prabhavathy S	

Please read the “Important Notice” on the next page.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

1 Products		2 Applications	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

1 Introduction

This document describes the design of the Screen Services Layer (SSL) core. The core features are usable through the OMX_SSL Layer, which is a client layer designed as per the Open Max 1.0 specifications. The typical operating environment for the SSL Layer is as follows:

- Nucleus® Operating System on LOCOSTO™'s ARM7 processor.
- SSL Core core to expose a standardized API to the application.
- OMX SSL Client for the Applications to make use of the SSL Core functionality
- I-SAMPLE BOARD (REV3)
- Generic Protocol Framework (GPF) for System resource access.

Purpose

This document details the design specifications for SSL Core on LOCOSTO.

Scope

This document addresses only design specifications.

Additional technical data can be found by referring to the OMAP™SS&P Technical Perspective and Data Package document.

The document provides information about technical data artifacts, including their title, standard ClearCase® VOB location, a brief description and the System or Software Checkpoint where the artifact is first introduced into the development process.

File Path

This design specification document shall be captured in ClearCase® path defined in the project CM Plan:

```
\OMAPSW_SysDev\LOCOSTO\Multimedia\System_Core\Docs
```

File Name

The file name of this document is CSSD_DESIGNSPEC_LOCOSTO_SSL_TASK.doc.

References

All References can be found on the Cellular Systems web site or the World Wide Process and Tools Group web site.

Definitions

Terms used in this document can be found in the Cellular Systems Glossary Document.

Terms that are introduced in this document are detailed below:

Table 1 Terms and Acronyms

ACRONYM	DEFINITION
DSP	Digital Signal Processor
GPP	General Purpose Processor
OMX	OMX and SSL Core are used interchangeably in the document.
API	Application Programming Interface
ARM	Advanced RISC Machines
OSAL	Operating System Adaptation Layer
UI	User Interface
DMM	Dynamic Mapped Memory

2 Architectural Overview

The Screen Services Layer (SSL) is middle-level layer that provides for an OpenMAX™ 1.0 compliant interface to applications for the display feature on the LOCOSTO™ platform. The operating environment for the SSL consists of the Nucleus® operating system and the Generic Protocol Framework (GPF). The application layer uses the SSL for all of the display functionalities. With respect to the Imaging Subsystem, the SSL operates in tandem with the IMG_Client and the CAMD_Client Components to provide for the view-finder and capture features. The SSL Layer makes use of the IMG_Client component to compose the multiple planes that it gets to the final frame buffer. At the lowest level, the SSL interacts with the LCD Manager to access the physical display.

2.1 System diagram

Figure 1 shows the architecture of **SSL Core** in context of the imaging sub-system.

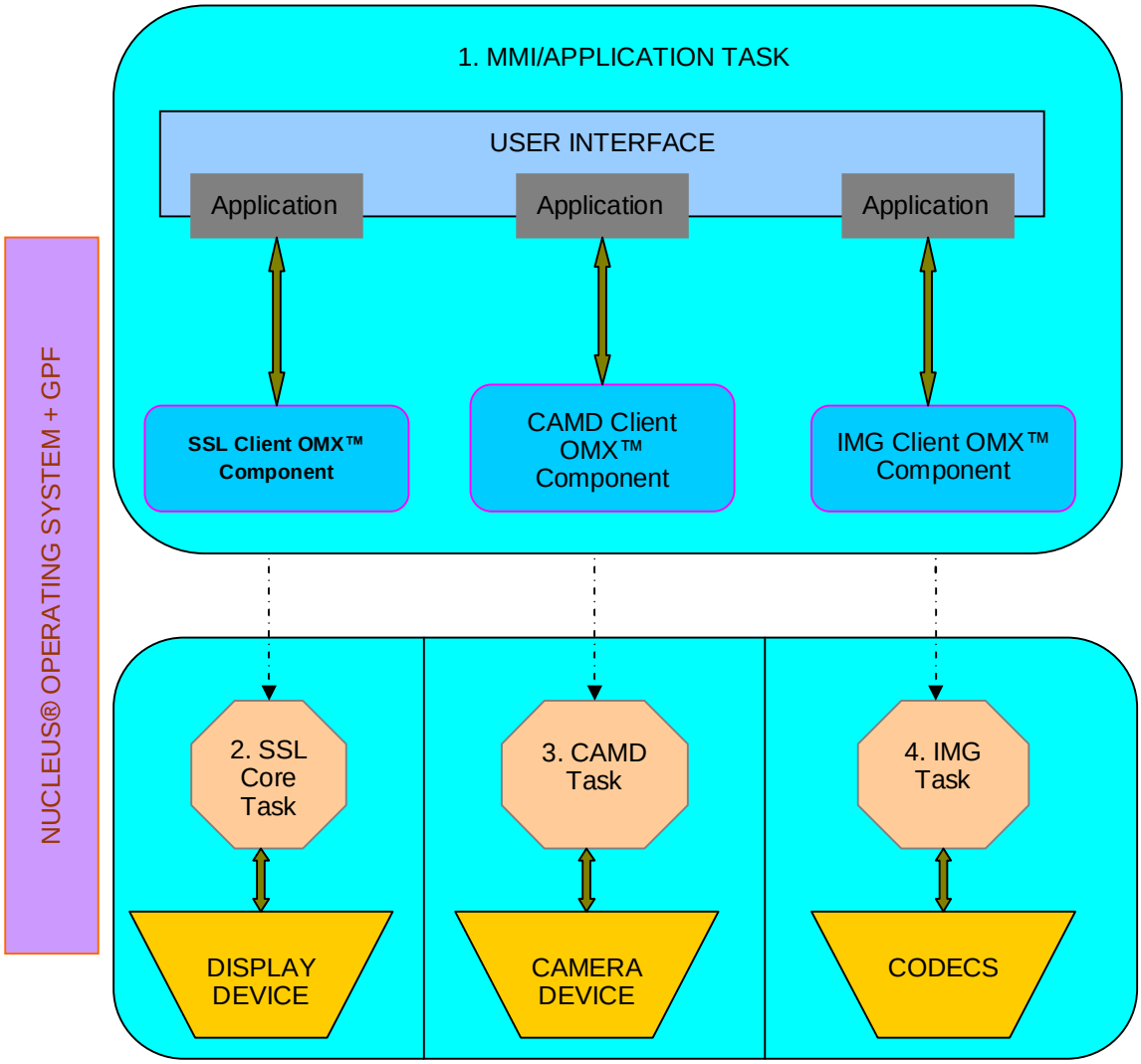

Figure 1 System diagram

From the diagram, it is seen that there are 4 tasks in the system, namely:

1. MMI/Application Task
2. SSL Core Task (includes the LCD Manager)
3. Camera Driver Task (CAMD)
4. Imaging Wrapper (IMG) Task

The SSL Task is actually the LCD Manager Task with added functionality. The CAMD Task is the Camera Driver task with enhanced functionality. The IMG Task includes the in it.

All these tasks make use of the GPF for using the system resources. The GPF abstracts the functionalities offered by the operating system.

2.2 Architecture diagram

Figure 2 shows the architecture of the SSL Core, as well as its interfaces with the MMI/Application Layer. The SSL Core runs in the same task context as the MMI/Appliacation Layer. The SSL Core executes as a separates task and includes the LCD Manager in it.

Figure 2 SSL CORE architecture diagram

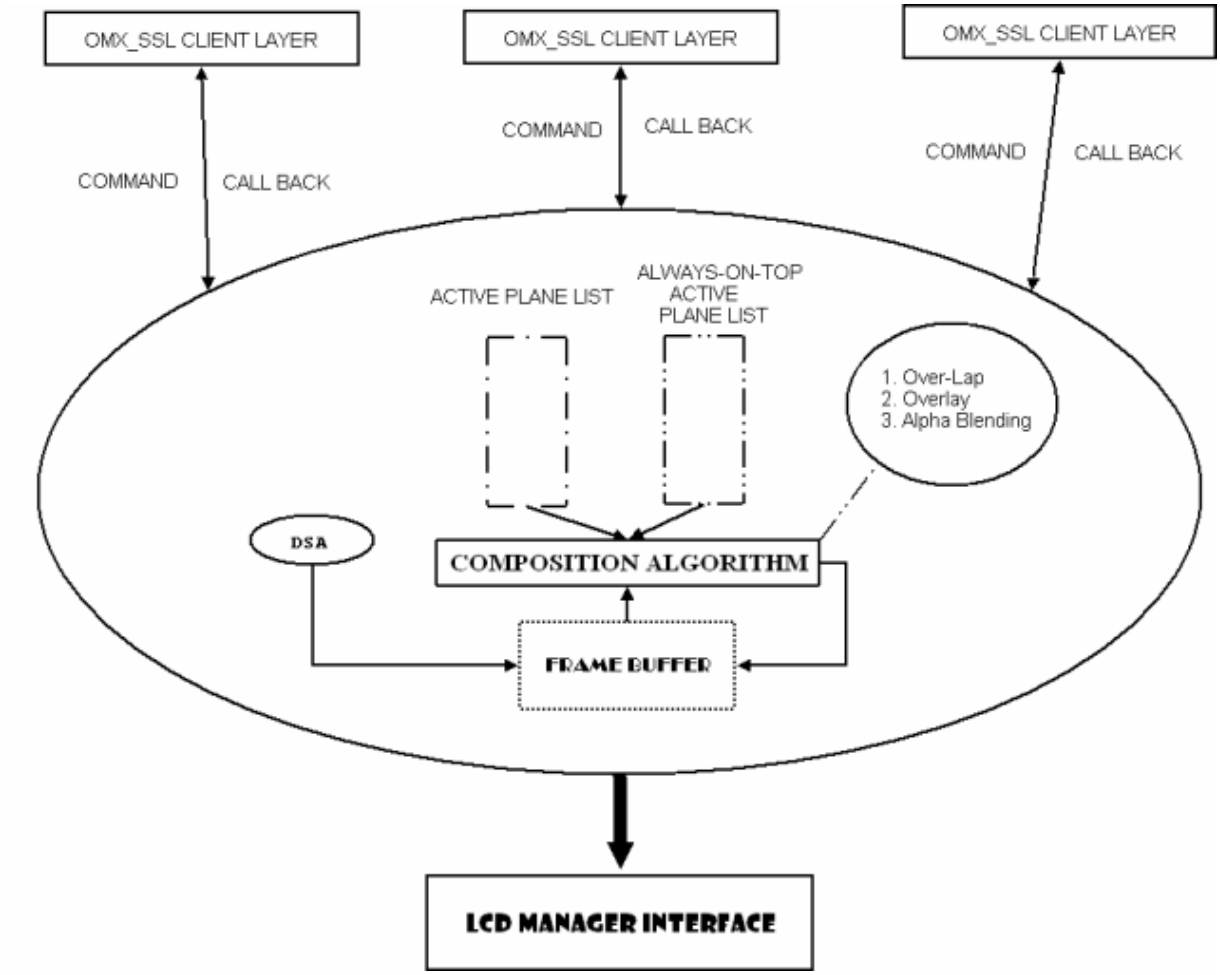

Figure 3 indicates multiple applications each creating an instance of the OMX SSL Client for displaying a 'plane'. A 'plane' as will be described later, is a logical region of disaply access to the physical display device.

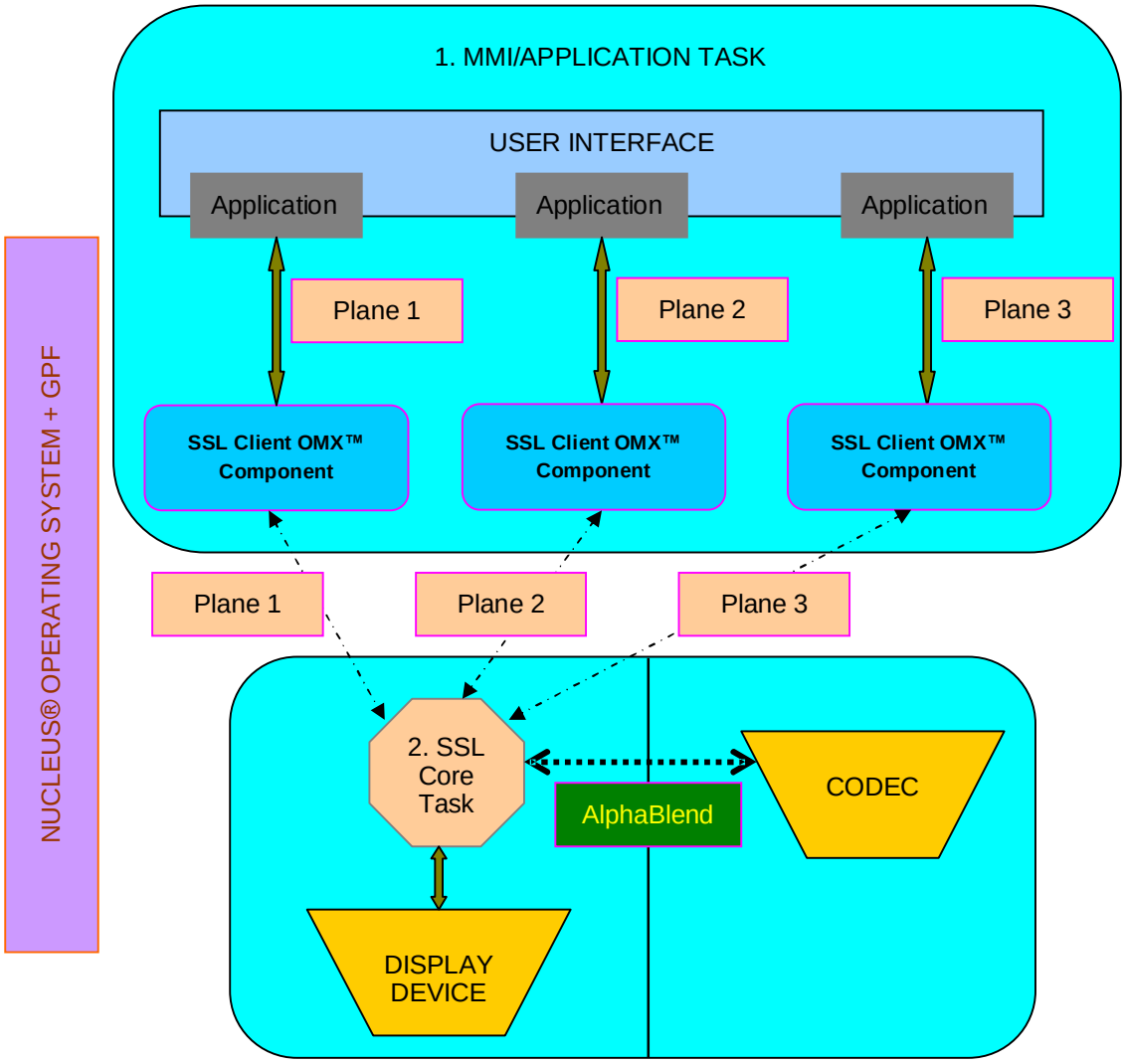

Figure 3 Multiple Plane Creation

Each application that requests a plane creates a new SSL OMX component configures it and uses it.

Alpha-blending based composition is shown as an example above. The codec algorithm is used as a library and there is not inter-task interaction. Hence the SSL can function independent of the IMG Task.

2.3 Software Design Interfaces

The SSL Task is a plane composer and display manager. It provides for logical planes to applications and also provides for a choice of composition algorithms for these planes. These are however compile time options only.

2.4 Features

The SSL is a generic display manager layer. It uses the services of an Image Processing Library and LCD Manager to provide for multiple logical planes and combines them to a single display. The interface exposed by SSL to applications is Open MAX 1.0 compliant. Following are the key features of the SSL:

- **Multiple Plane Handling**
- **Screen Composition using Alpha Blending, Overlay or Overlap**
- **Always on Top plane specification**
- **Background configuration**
- **Direct Screen Access (DSA)**

3 Design Rationale

The SSL Core taks supports a set of commands that map to a state transition and/or API for the OMX_SSL client.

A display scan is where the ‘Active’ queue is examined in the SSL Core and blended to form the FrameBuffer. After the ‘Active Queue’ scan is completer, the ‘Active Always on Top’ Queue is examined and the planes are binary blended with the current framebuffer.

An exception to the plane described arises in the case of Direct Screen Access (DSA). When a DSA is requested by the application, the SSL Core returns the FrameBuffer pointer to the application through the OMX Component. The application uses this to dump the display data. Thus a DSA Plane has the framebuffer pointer, the offset and dimensions and the active flag associated with it. Even if a DSA plane is active, other planes can be active and can be blended with the FrameBuffer to produce composite images. The DSA feature is especially useful for ViewFinder type of applications. There can be more than on DSA plane, but only to mutually exculsive regions. This can provide for exciting options like display screen windowing etc.

The SSL Core is an ACTIVE GPF task. This means that, the core task’s pei_run function waits for messages from the OMX Client. Once a primitive is received, the pei_primitive function then processes the primitive appropriately. Commands supported are manily those that provide for 1. Plane creation

2. Plane destruction 3. Plane activation 4. Plane deactivation 5. Plane configuration 6. Plane pausing 7. Plane suspension 8. Input buffer mapping

The various commands and their effects are found in the table below:

COMMAND	FUNCTION
OMX_SSL_CMD_SETPLANEPROPERTIES	Sets the plane configuration properties; refreshes plane if the plane is active.
OMX_SSL_CMD_CREATEPLANE	Creates the plane; shadow buffer allocated for non-DSA planes.
OMX_SSL_CMD_ACTIVATEPLANE	Puts the plane as the top most plane in the Active List; refreshes the LCD. If the plane is an Always In Top plane, it puts the plane into that list and refreshes accordingly.

OMX_SSL_CMD_SUSPENDPLANE	LCD refreshed ignoring this plane's shadow buffer alone.
OMX_SSL_CMD_DEACTIVATEPLANE	Removes the plane from the Active queue or Always On Top queue.
OMX_SSL_CMD_RESUMEPLANE	Refreshes the LCD considering this plane also in the list. Suspend/Resume maintains the plane position in the list.
OMX_SSL_CMD_DESTROYPLANE	Destroys the shadow buffer resource and removes from the lists as well. Also refresh the LCD.
OMX_SSL_CMD_SETINBUFF	Provides an input buffer to the LCD for a plane. This causes data update to the Shadow Buffer. Composition and LCD update are performed if the plane is active.

An important point to be noted: Whenever a DSA Plane is active, update for all other planes happen (onto LCD) only for the DSA plane refresh.

3.1 Relevant Specifications

Refer to the following specifications for additional information:

- SSL Core Standard (www.khronos.org)
- CSSD Design Specification for CAMD Client OMX Component
- CSSD Design Specification for IMG Client Component
- CSSD Design Specification for OMX SSL Client
- GPF Architecture Documentation

3.2 Design Trade-offs

- The use of DSA provides for a fast mechanism for the application to display frames, but restricts the number of planes that can be simultaneously displayed.
- The SSL Core runs as a separate task for design simplicity. It is a better approach to have it in the LCD Task context.
- Only one DSA plane is permitted in the design for simplicity. This also seems sufficient for most cases.

3.3 Hardware Dependencies

This component is designed to run on the Locosto platform. The development board used was an I-Sample Revision 3 board.

3.4 Other Pertinent Design Issues

There are no design issues.

4 Memory Requirements

Memory requirements of the **SSL Core** are listed in Table 2.

Table 2 Memory requirements of the **SSL Core**

Type	Size	Structure	Comments

From the above table, the total dynamic memory required for the **SSL Core** can be obtained using the following calculation. Sizes of buffer headers and buffers for a port will need to be included only if the **SSL Core** allocates buffers for that port.

Total dynamic memory size in bytes =

4.1 Memory Allocation

All source memory regions are allocated by the application (MMI). For each Non-DSA plane, a shadow buffer is allocated and the size of the buffer is decided by the width, height and the image data format.

5 Sub-Components

5.1 Include files for the SSL Core

Table 3 Include files for the **SSL Core**

File	Function
Typedefs.h, vsi.h, pei.h, tools.h	GPF Includes
mon_ssl.h	Memory monitor includes
OMX_Types.h	To get access to OMX defined data types
omx_sslcomponent.h	For access to the OMX Client defined commands and data types
OMX_SSL_Private.h	Private header for SSL
OMX_TIImage.h	To have access to the common image data types
Ssl.h	To get global entity definitions
alpha_overlay.h	Access to the alpha blending/overlay functions
exp_colorconv_rotate_scaling.h	Access to the export header for image processing functions
lcd_interface.h	Access to the LCD Manager APIs

6 Control and Data flow

6.1 Component States

The SSL Core functionality is controlled by the OMX_SSL Component's state flow. Each OMX_SSL state transition maps to a functionality in the SSL Core. The following table lists the Core functionality and the OMX_SSL state change that achieves it.

SSL CORE FUNCTIONALITY	OMX_SSL STATE TRANSITION
CREATE PLANE	LOADED->IDLE
ACTIVATE PLANE	IDLE->EXECUTE
DELETE PLANE	IDLE->LOADED
DE-ACTIVATE PLANE	EXECUTE->IDLE
SUSPEND PLANE	IDLE->PAUSE; EXECUTE->PAUSE
RESUME PLANE	PAUSE->EXECUTE; PAUSE->IDLE

Table 4 Client<->Core Functionality mapping

6.2 Component Phases

The SSL Core's processing element is a 'Plane'. A plane is a logical region of display that is used in a composition algorithm and rendered.

The SSL Plane goes through the phases typically as per figure below. The figure also includes the SSL Core task initialization actions.

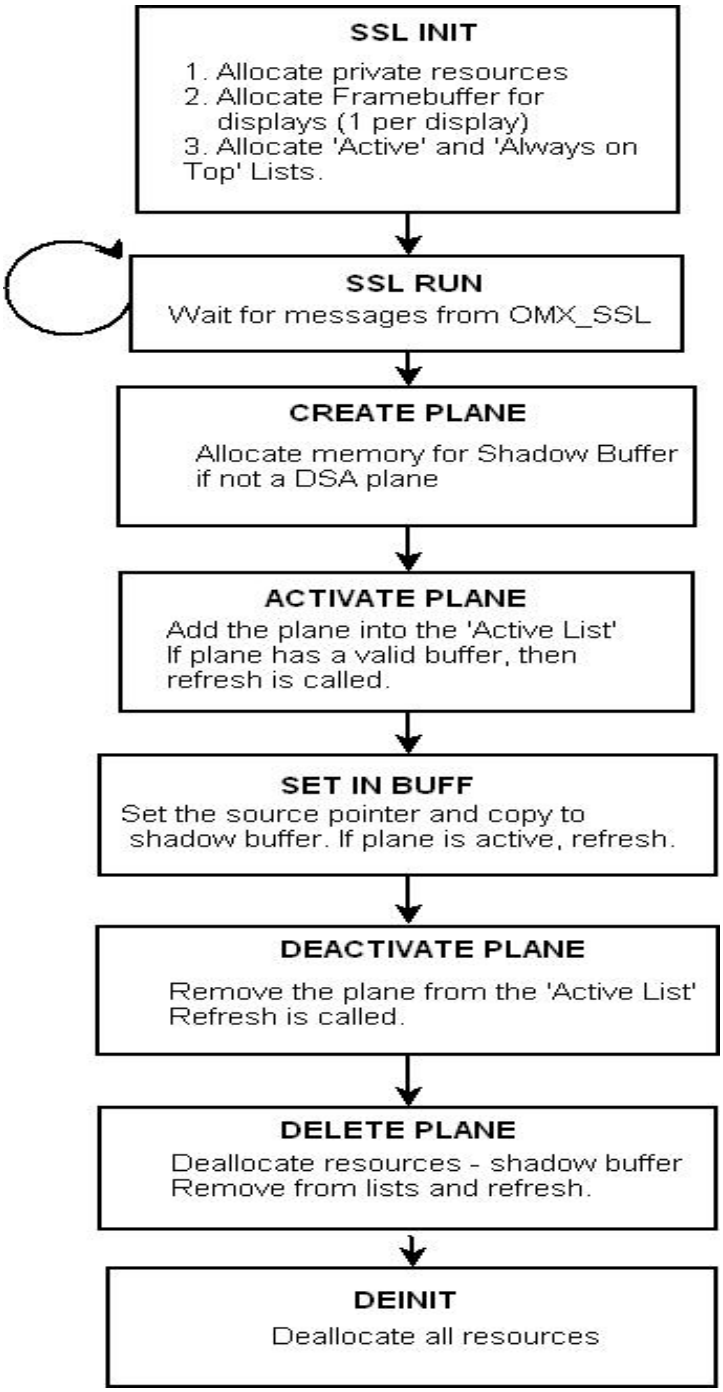

Figure 4 SSL Core State Flow

6.3 Input and Output Buffer Allocation Scenarios

The SSL Core is a 'Sink Task'. This means that it has only output buffers and no buffers are expected to read from. The core allocates Shadow Buffers for each of the Non-DSA planes and maintains them. The size of each of these is given by: Plane Width * Plane Height * Bits per pixel / 8 bytes.

Allocation and deallocation of shadow buffers is handled by the core along with plane creation and deletion. In case of DSA planes, there is no shadow buffer allocation since the framebuffer is directly accessed by the application, thereby saving the copy.

7 Software Requirements

This section needs to filled with the requirements table.

8 Requirements Traceability

This section describes the Data types, data structures, callbacks and macros, which are supported by the SSL Core.

8.1 Defined Types

The SSL Core uses the data types as mentioned in the OMX_SSL Design specifications. This is not a good design and will be changed in a future release.

8.2 Data Structures

The SSL core data structures are ones defined in the OMX_SSL Design specifications.

8.3 API Requirements Coverage

The SSL Core core provides a set of macros that are used by Application to perform various operations like loading the component, communicating with OMX component etc. These macros are defined in OMX_Core.h. Each macro maps to a function implemented by the OMX component. Detailed description of each function implemented by the OMX SSL Component and is given in following sub sections. Application (OMX) must not call any of these functions directly and instead use the macros provided by the OMX core.

8.3.1 Common pre conditions

Before OMX can invoke the functions of the OMX SSL Component, the component must be loaded. OMX loads the component by calling OMX_GetHandle, which is an OMX core function. For details on component loading and unloading refer to SSL Core Core Design Specification. The SSL Core can be accessed only by a client (OMX_SSL), with the OMX_ APIs.

The SSL Core exposes only a command based interface. The commands are all accessible through the OMX_ APIs. Please refer to the OMX_SSL Desing for API details.

8.4 Application callbacks

The SSL Core specification requires OMX to provide three callbacks for buffer exchange and event handling. At loading time, the OMX component receives a structure containing pointers to the callback functions. The OMX component makes a copy of this structure in the private data area. This section describes the callback functions.

8.4.1 OMX_SSL_Callback

```
void __OMX_SSL_Callback (OMX_HANDLETYPE hComponent,
          OMX_ERRORTYPE hComponent,
          OMX_SSL_CMDTYPE tCommand,
          OMX_PTR pVoid//To be used later
          ); //Call back function pointer
```

Description

The EventHandler method is used to notify OMX when an event of interest occurs. This event may be change of state, an error occurred etc.

Parameters

Name	Type	Description
hComponent	IN	This input argument is the component handle.
hComponent	IN	Error Status from the core
tCommand	IN	Command for which this is the callback
pVoid	IN	Auxulary data pointer

Return

None

Requirement Coverage

This method addresses requirement:

SR14062: This interface must comply with SSL Core specification.

Implementation

- EventHandler is called to notify state changes in the SSL Core to the OMX Client.
- In the SSL OMX Component, this callback chains the application callback.

8.5 Internal Functions

The LCD Driver APIs are used in the SSL Core. Also, imag processing APIs are used for the screen composition.

8.6 Non-API Requirements Coverage

This SSL Core core will comply with the TI coding guidelines, located in the Clear Case® VOB path:

```
\\OMAPSW_docs\Process\Coding_Standards\OMAPSW_C_CodingStandards.doc
```

9 Assumptions

- OMX and the SSL Client Component run in the same task.
- The SSL Core task runs as a separate thread.
- The IMG Server is active and running and supports alpha-blending.
- The LCD Manager is active.