Software OSD

User Guide

Document Version 1.00

31st January 2012

IMPORTANT NOTICE

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

Products		**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
Clocks and Timers	www.ti.com/clocks	Digital Control	www.ti.com/digitalcontrol
Interface	interface.ti.com	Medical	www.ti.com/medical
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
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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

Revision History

Version	Date	Revision History
1.0	31 Jan 2012	Updated document for GA release 2.0

TABLE OF CONTENTS

1 Introduction

This module represents the implementation of software OSD functionality. The software OSD runs in a separate link called as SWOSD link which gets the video frames from its previous link as input and does the OSD imprinting and outputs the SWOSD applied video frames to its next link in the chain. The data flow is explained in the figure 3.

The OSD API takes the YUV or predefined string as input from the application and imprints the OSD along with live video data. Presently OSD library supports drawing upto 15 windows per stream. It can be increased to more than 15 windows but the library needs to be re-built.

OSD window can be enabled/disabled for each stream specified by application. If Application gives YUV data then it can be drawn directly, if application wants to display the string pattern say ”IPNetCam” then it has to converted to YUV data first then it can be drawn on live data.

2 Product Requirements

High level functional requirements are as follows:

- There should be a Handle Object for each video data resolution
- Handle Object contains information of the live video window along with all the bitmap windows that need to be drawn on the video window
- Multiple Handle Objects for each resolution have to be initialized in the beginning during application start-up
- SW OSD is implemented as a separate link which receives messages from the application.
- SW OSD can allow the parameters for different windows to be changed at any time, but the effect will be seen only in next frame
- Bitmap data to be created by the application for different resolutions
- Memory management for Bitmap window done at application level
- Blending would not be supported. Only transparency is available
- If no transparency is required, EDMA should be used
- Support YUV422, YUV420 planar (UV interleaved)
- Does not support RGB or AlphaRGB format.
- When multiple bitmap windows are overlapped, bitmap window 1 gets priority over bitmap window 2 and so on.

Data required to initialize	Input	Output
• Input image width, Height • YUV data or predefined string. • Font width, font height, character width, character height and offset. • Window width, Window height, line offset, format • Coordinates like startX and startY	• YUV data or predefined string • Input image width, Height • Font width, font height, character width, character height and offset. • Window width, Window height, line offset, format • Coordinates like startX and startY	• OSD draw for multiple windows

2.1 Input Image Requirement

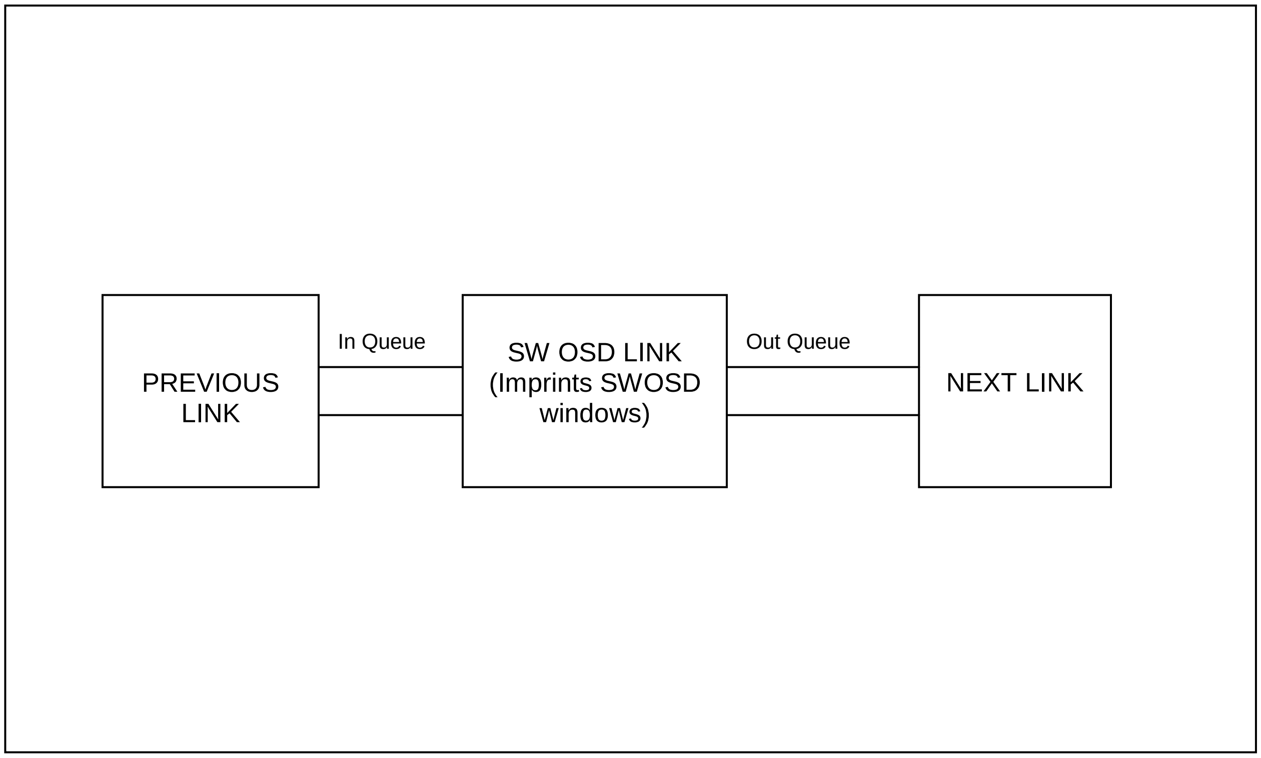
The following input parameters are required to be programmed by the application. The OSD API's updates the new value from the user such as main width/ height, font width/height, character width/height, predefined string/YUV Data, startX, startY.

2.2 Memory Requirement for the System

The application need to allocate memory for different OSD handle and predefined input string.

3 IPNC Flow Diagram

The software OSD flow diagrams will give details about the implementation in IPNC



4 API Definition

SWOSD_init

Input arguments: NILL

Return value – int – Return 0 on success or -1 on failure.

Definition- This function sets the OSD parameters to null and allocate the memory, create message queue.

SWOSD_exit

Input arguments: NILL

Return value – int – Return 0 on success or -1 on failure.

Definition- This function deletes the message queue which was created during SWOSD_init.

SWOSD_setMainWinPrm

Arg1 - SWOSD_Hndl - Input – pointer to the SWOSD_Hndl structure which contains the details about OSD Bitmap Window, Main Video Window and Font Information.

Arg2 - SWOSD_MainWinPrm * - Input – Structure which contains the format, main window width, height and line offset.

Return value – int – Return 0 on success or -1 on failure.

Definition – This API sets the main window width, height, lineoffset and format based on input stream.

SWOSD_setBmpWinPrm

Arg1 - SWOSD_Hndl - Input – pointer to the SWOSD_Hndl structure which contains the details about OSD Bitmap Window, Main Video Window and Font Information details.

Arg2 - bmpWinId - Input – window ID.

Arg3 - SWOSD_BmpWinPrm - Input – pointer to the SWOSD_BmpWinPrm which contains the details about OSD window parameters.

Return value – int – Return 0 on success or -1 on failure.

Definition – This API sets the OSD window width, height, format, x, y, transparency range and transparency value.

SWOSD_setBmpWinEnable

Arg1 - SWOSD_Hndl - Input – pointer to the SWOSD_Hndl structure which contains the details about OSD Bitmap Window, Main Video Window and Font Information details.

Arg2 - bmpWinId - Input – window ID.

Arg3 – enable - Input – flag which enables/disables the OSD window.

Return value – int – Return 0 on success or -1 on failure.

Definition – This API enables/disables the OSD Window and sends the message. This API will be called from application.

SWOSD_setBmpchangeWinXYPrm

Arg1 - SWOSD_Hndl - Input – pointer to the SWOSD_Hndl structure which contains the details about OSD Bitmap Window, Main Video Window and Font Information details.

Arg2 - bmpWinId - Input – window ID.

Arg3 – startX - Input – X coordinate of starting point for the bitmap window.

Arg4 – startY - Input – Y coordinate of starting point for the bitmap window.

Return value – int – Return 0 on success or -1 on failure.

Definition – This API sets the OSD x, y values and sends the message. This API will be called from application.

SWOSD_winChangeTransperency

Arg1 - SWOSD_Hndl - Input – pointer to the SWOSD_Hndl structure which contains the details about OSD Bitmap Window, Main Video Window and Font Information details.

Arg2 - bmpWinId - Input – window ID.

Arg3 – bmpTransValue- Input –transparency value

Arg4 – bmpTransRange- Input –range of transparency. Library will allow +/- bmpTransRange from bmpTransValue to be used as transparency value.

Return value – int Return 0 on success or -1 on failure.

Definition – This API sets the OSD transparency range and value and sends the message.This API will be called from application.

SWOSD_setBmpWinAddr

Arg1 - SWOSD_Hndl - Input – pointer to the SWOSD_Hndl structure which contains the details about OSD Bitmap Window, Main Video Window and Font Information details.

Arg2 - bmpWinAddr- Input – BMP Window address (Used as Y address in YUV420planar mode)

Arg3 - bmpWinAddrUV- Input – BMP Window UV address (NULL in YUV422 interleaved mode)

Arg4 - bmpWinId - Input – window ID.

Return value – int – Return 0 on success or -1 on failure.

Definition – This API sets the OSD Window buffer address and sends the message.This API will be called from application.

SWOSD_winDrawHandle

Arg1 - SWOSD_Hndl - Input – pointer to the SWOSD_Hndl structure which contains the details about OSD Bitmap Window, Main Video Window and Font Information details.

Arg2 – mainWinAddr- Input – Main window Address.

Return value – int – Return SWOSD_0 on success.

Definition – This API will draw the OSD bitmap windows on the main video window for the input handle, based on number of bitmap windows.

SWOSD_winDraw

Arg1 - SWOSD_Hndl - Input – pointer to the SWOSD_Hndl structure which contains the details about OSD Bitmap Window, Main Video Window and Font Information details.

Arg2 - mainWinAddr- Input – main window address.

Arg3 - bmpWinAddr- Input – OSD window address.

Arg4 - bmpWinAddrUV- Input – OSD window UV address.

Arg5 - bmpWinId - Input – window ID.

Return value – int – Return 0 on success or -1 on failure.

Definition – This API imprints the OSD window along with main window address based on window ID.This API also checks the boundary condition like whether OSD window width has exceeded main window width and height of the OSD is exceeded the main widow height.

SWOSD_MakeOsdwinstring

Arg1 - SWOSD_Fontdata - Input – pointer to the SWOSD_Fontdata structure which contains the details about OSD font such as character width, character height, string width, string height, string line offset, format, font address.

Arg2 - pInputstr- Input – predefined input string given by application.

Arg3 - pBuff- Input/Output – buffer pointer which contains the YUV data of input string.

Arg4 - SWOSD_BmpWinPrm - Input – pointer to the SWOSD_BmpWinPrm which has OSD window parameters such as window enable, format, x, y, width, height, Transparency value ,range, Transparency Enable/Disable and BMP window address.

Arg5 – stringLength – length of the string to be imprinted for particular OSD bitmap window.

Return value – int – Return 0 on success or -1 on failure.

Definition – This API converts input string to YUV data. It also updates the OSD bitmap window width, height and line offset.

SWOSD_setPrivBmpWinEnable

Arg1 - SWOSD_Hndl - Input – pointer to the SWOSD_Hndl structure which contains the details about OSD Bitmap Window, Main Video Window and Font Information details.

Arg2 - bmpWinId - Input – window ID.

Arg3 – enable – Input – Window Enable/Disable.

Return value – int –Return 0 on success or -1 on failure.

Definition –This API updates the OSD window enable/disable to the OSD bitmap window structure.

SWOSD_setPrivBmpchangeWinXYPrm

Arg1 - SWOSD_Hndl - Input – pointer to the SWOSD_Hndl structure which contains the details about OSD Bitmap Window, Main Video Window and Font Information details.

Arg2 - bmpWinId - Input – window ID.

Arg3 – startX – Input – X coordinate.

Arg4 – startY – Input – Y coordinate.

Return value – int – Return 0 on success or -1 on failure.

Definition –This is the private API which updates the OSD x and y coordinates to the OSD bitmap window structure.

SWOSD_setPrivwinChangeTransperency

Arg1 - SWOSD_Hndl - Input – pointer to the SWOSD_Hndl structure which contains the details about OSD Bitmap Window, Main Video Window and Font Information details.

Arg2 - bmpWinId - Input – window ID.

Arg3 – bmpTransValue – Input – bitmap transparency value.

Arg4 – bmpTransRange– Input – bitmap transparency range.

Return value – int – Return 0 on success or -1 on failure.

Definition –This is the private API which updates the OSD Transparency value and range to the OSD bitmap window structure.

SWOSD_setPrivBmpWinAddr

Arg1 - SWOSD_Hndl - Input – pointer to the SWOSD_Hndl structure which contains the details about OSD Bitmap Window, Main Video Window and Font Information details.

Arg2 - bmpWinAddr- Input – bitmap window address.

Arg3 - bmpWinAddrUV- Input – bitmap window UV address.

Arg4 – bmpWinId– Input – bitmap window ID.

Return value – int - Return 0 on success or -1 on failure.

Definition –This is the private API which updates the OSD window address to the OSD bitmap window structure.

SWOSD_getOSDHndls

Arg1 – numHndls – Input – pointer to the number of handles.

Return value – int – Return pointer to an array of OSD handles.

Definition –This API gets the number of active handles created by application for different resolutions/streams.

SWOSD_createHandle

Arg1 – SWOSD_Hndl – Input – pointer to the address of OSD handle.

Arg2 – indexHndl– Input – pointer to the index.

Return value – int – Return 0 on success or -1 on failure.

Definition –This API creates the handle for different resolutions/streams and returns the handle.

SWOSD_deleteHandle

Arg1 – indexHndl– Input –index of the handle.

Return value – int – Return 0 on success or -1 on failure.

Definition –This API deletes the handle based on input index.

SWOSD_createFontDatabase

Arg1 – fontDatabaseY – Input/Output – buffer pointer contains Y font address for each character of OSD string pattern.

Arg2 – fontDatabaseUV – Input/Output – buffer pointer contains UV font address for each character of OSD string pattern.

Arg3 - swosd_stringPattern – this is a macro contains string pattern of all characters

Arg4 - SWOSD_Fontdata - Input – pointer to the OSD_Fontdata structure which contains the details about OSD font such as character width, character height, string width, string height, string line offset, format, font address.

Return value – int – Return 0 on success or -1 on failure.

Definition – This API creates a database of Y and UV address of each character in string pattern.

5 Software OSD Usage Details

5.1 Creation of bmp image of string pattern:

We have used Microsoft Paint to create bmp image of string pattern with different font size and different font/background color. While creating bmp image of string pattern make sure that character length must be an integer value.

5.2 Conversion of bmp format to jpeg format:

Use any tools to convert bmp image into jpeg format. We have used "ReaJpeg Pro" tool to convert bmp image into jpeg format.

5.3 Conversion of JPEG format to YUV422 format:

Steps to convert JPG format to YUV422 format:

1. Install any tool which converts JPG to YUV format. We have installed a freeware tool called ImageMagick-6.4.3-Q16.
2. After Installing ImageMagick-6.4.3-Q16**,** convert JPG format to YUV format as below.
   a. Open the command prompt and go to the path where ImageMagick-6.4.3-Q16 tool is installed.
   Ex: C:\Program Files\ImageMagick-6.4.3-Q16>

   b. Then from the path give **convert <JPEG input file > <output file.uyvy>**
   Ex: C:\Program Files\ImageMagick-6.4.3-Q16>convert A1.jpg B1.uyvy
   Where A1.jpg is input file and B1.uyvy is the output file.

   Note: ImageMagick uses the extension in the filenames as the format of the file. In this case .UYVY means YUV422 interleaved format.

5.4 Conversion of YUV422 format to YUV420 planar (UV interleaved) format:

Steps to convert YUV422 format to YUV420 planar format:

1. Use any tools which convert YUV422 format to yuv420 format. We have used "image_convert" tool which is provided in the package.
2. Copy the image_convert folder in some location.
   Ex: Copy inside D:\
3. Put the YUV422 file in to bin folder inside "image_convert"
   Ex: D:\image_convert\bin
4. Open the command prompt and go to the path where image_convert tool is copied
   Ex: D:\image_convert\bin>
5. Use image_convet command to convert YUV422 format to yuv420 format.

5.5 Extract the y data and uv data:

Extract y data and uv data from YUV420 planar image and put in to different binary files.

5.6 Conversion of YUV files to TEXT file:

These steps are needed to convert the binary data to a text form, so that it can be included in the software code as an array. This way we can directly link the Y and UV data to the executable binary that uses SWOSD library.

Steps to convert Y and UV file to TEXT file:

1. Install bin2c tool, provided in the package.
2. Open the command prompt and go to the path where bin2c tool is installed
   Ex: C:\Program Files\bin2c>
3. Then from the path give **bin2c <YUV Input file> >output file>**
   Ex: bin2c B1.yuv >C_YUV.c, convert the contents of C_YUV.c into an array of characters as shown below.
   char <Array Name> [] = {
   ………………….
   ………………….
   };
4. Place C_YUV.c file in the folder along with the application code**.** For example, in IPNC Reference Design, place the files in
   mcfw\src_bios6\links_m3vpss\alg\sw_osd\fonts
   Add array name (<Array Name>) of C_YUV.c in SwosdThr.c and use these array to create font.
5. Change the character width and height, string width and height for different set of fonts in application code.
   Ex: Change SWOSD_CHARWIDTHxxxx, SWOSD_CHARHEIGHTxxxx, SWOSD_STRINGWIDTHxxxx, and SWOSD_STRINGHEIGHTxxxx for different fonts in Swosd.h of IPNC application.

5.7 Steps to convert Y and UV file to TEXT file:

Steps to convert Y and UV file to TEXT file:

1. Install bin2c tool, provided in the package.
2. Open the command prompt and go to the path where bin2c tool is installed
   Ex: C:\Program Files\bin2c>
3. Then from the path give **bin2c <Y Input file> >output file>**
   Ex: bin2c B1.y >C_Y.c, convert the contents of C_Y.c into an array of characters as shown below.
   char <Array Name> [] = {
   ………………….
   ………………….
   };
4. Also repeat above step to convert UV file to text. Let's say the output file is C_UV.c.
5. Place C_Y.c and C_UV.c files in the folder along with the application code**.** For example, in IPNC Reference Design, place the files in
   mcfw\src_bios6\links_m3vpss\alg\sw_osd\fonts
   Add array name (<Array Name>) of C_Y.c and C_UV.c in videoSwosdThr.c and use these array to create font.
6. Change the character width and height, string width and height for different set of fonts in application code.
   Ex: Change SWOSD_CHARWIDTHxxxx, SWOSD_CHARHEIGHTxxxx, SWOSD_STRINGWIDTHxxxx, and SWOSD_STRINGHEIGHTxxxx for different fonts in Swosd.h of IPNC application.

5.8 Steps to display user defined image file stored in SD card:

Suppose if we want to display some icon/image (say TI Logo) in IPNC Reference Design, first we have to convert it to YUV.

1. Convert the icon/image as explained above.
2. Create folder called BITMAP in Memory card and place the converted YUV data as ICON.bin
3. Change width and height of the icon/image in application code. Ex: SWOSD_CHARWIDTHTILOGOxxxx, SWOSD_CHARHEIGHTTILOGOxxxx in videoSwosd.h according to the size of the newly created YUV or Y and UV data.
4. If ICON.bin is found during OSD (On screen display) operation then this file would be read and copy into a buffer. This image/icon will then be displayed during OSD operation.
5. If ICON.bin is not found during OSD operation then a static array which is already present in application will be displayed, Ex. In application Static array is already present in swosdThr.c**,** will be displayed.

5.9 Adding New OSD String Pattern:

If application wants to create its own string pattern say "aAbBcCdD….." then it needs to provide the same string pattern in the YUV format. It can be done using a JPEG to YUV tool converter. Then string pattern has to be updated in sw_osd.h.

Example string pattern is provided in the release package

#define SWOSD_STRINGPATTERN "aAbBcCdD…."

5.10 User Defined Transparency versus Fixed Transparency

5.10.1 User Defined Transparency

In user defined transparency mode there is no restriction on the color of the OSD bitmap image that means user can make OSD bitmap image with any colors. In this mode, if pixel value of OSD bitmap window is within the range of transparency value plus/minus transparency range, the pixel becomes transparent. The pixel value of live video data remains same.

Otherwise the pixel value of OSD bitmap image is assigned to corresponding pixel value of live video data. This mode needs checking of each pixel of OSD bitmap images and hence takes long time with increase in the CPU load.

5.10.2 Fixed Transparency

In fixed transparency mode or "optimized transparency", there is the restriction on pixel value of OSD bitmap image - Each pixel value can be either black(0x00) or white(0xFF). Unlike user transparency, it simply does the "Logical Or" operation of each 4 bytes of y or uv data of osd image with actual live video data. Applying only "Logical Or" operation for 4 bytes considerably reduce the CPU load compared to user transparency. In this mode, only the OSD bitmap image pixel with black (0x00) color becomes transparent.

For fixed transparency, the OSD bitmap window width must be multiple of 4. It is recommended that before calling function **SWOSD_MakeOsdwinstring** make sure that OSD bitmap window width is multiple of 4. If it is not multiple of 4, then API

SWOSD_MakeOsdwinstring takes care of it by extending the OSD bitmap window width to be a multiple of 4 and filling the extra area with 0x00.

Software OSD library support fixed transparency only for YUV420 planar (UV interleaved) format.

5.11 Steps to Enable/Disable Transparency

5.11.1 Transparency Disable

Steps to disable transparency:

1. Set SWOSD_BmpWinPrm→userTransparency = SWOSD_RBOOLTRUE for each OSD window.
2. Set SWOSD_BmpWinPrm→transperencyEnable = SWOSD_BMPWINDISABLE for each OSD window parameter.

5.11.2 User defined Transparency Enable

Steps to enable user defined transparency:

1. Set SWOSD_BmpWinPrm→userTransparency = SWOSD_RBOOLTRUE for each OSD window.
2. Set SWOSD_BmpWinPrm→transperencyEnable = SWOSD_BMPWINENABLE for each OSD window.
3. Set SWOSD_BmpWinPrm→transperencyVal for each OSD window.
4. Set SWOSD_BmpWinPrm→transperencyRange for each OSD window.

5.11.3 Fixed Transparency Enable

Steps to enable fixed transparency:

1. Set SWOSD_BmpWinPrm→userTransparency = SWOSD_RBOOLFALSE for each OSD window.
2. Set SWOSD_BmpWinPrm→transperencyEnable = SWOSD_BMPWINENABLE for each OSD window.

6 Introduction to OSD done utilizing SIMCOP

This part of the library module represents the implementation of software OSD functionality on DM8148/DM8107 based DVRs utilising SIMCOP HW accelerators. The software OSD runs in a separate link called as SWOSD link which gets the video frames from its previous link as input and does the OSD imprinting and outputs the SWOSD applied video frames to its next link in the chain.
The OSD API takes the YUV or predefined string as input from the application and imprints the OSD along with live video data.
OSD window can be enabled/disabled for each stream specified by application. If Application gives YUV data then it can be drawn directly, if application wants to display

the string pattern then it has to converted to YUV data first then it can be drawn on live data.

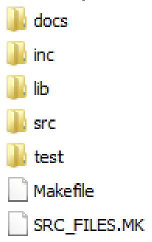
7 Product requirement

High level functional requirements are as follows:

- 16CIF @ 30fps + 16QCIF @ 30fps channels with each channel having 7 windows of different sizes.
    - Max size of window is original video size.
    - Should operate when input is tiled/non-tiled.
    - The window resolution and number of windows can be changed dynamically
    - The overlay bitmap can be changed dynamically
- The scatter-gather is implemented using EDMA3. Basically a list of param entries representing each window transfer is prepared in advance. During the scatter-gather execution, M3 fires and waits for each of the transfer without needing to reprogram any param entry
    - All EDMA resources should be allocated using EDMA_LLD and no hard coding of channel numbers/params usage should be made.
    - No Ducati-L2 /OCMC memory usage should be assumed by algorithm.
    - Should support 4:2:0 and 422ILE YUV video frame format.
    - Should work on field and frame input.
- Projected loading for M3 is as follow: 25 MHz of M3 (under full system load) to gather the blocks into one frame and then scatter them back after alpha blending.

8 Source and Library location

The SWOSD is integrated to the ISS package (version 2.0) and is available in the path *iss_02_xx_xx_xx\packages\ti\psp\iss\alg\swosd\* folder. The folder contains a sample application to exercise SWOSD library APIs. Following figure depicts the folder structure within the SWOSD package.


docs
inc
lib
src
test
Makefile
SRC_FILES.MK

The top level Makefile which is available at *iss_02_xx_xx_xx* folder is used to generate the library from source as well as sample executable.

9 API Definition

9.1 Symbolic constants and enumerated data types

Group or	Symbolic constant name	Description

enumeration class		
`SWOSD_DataFormat`	`SWOSD_FORMAT_YUV422I_YUYV`	Color format is YUV422ILE with YUYV as spacing
	`SWOSD_FORMAT_YUV422I_UYVY`	Color format is YUV422ILE with UYVY as spacing
	`SWOSD_FORMAT_YUV420SP_UV`	Color format is YUV420 Semi planar where is Y is planar and UV are interleaved
	`SWOSD_FORMAT_DEFAULT`	Default is YUV420SP

9.2 Constants

Constant Name	Value
`SWOSD_MAX_FRAMES_PER_BLEND_FRAME`	16
`SWOSD_MAX_CHANNELS`	48
`SWOSD_MAX_PLANES`	2
`SWOSD_MAX_MEM_BLOCKS`	16
`SWOSD_MAX_WINDOWS`	8

9.3 Data structures

9.3.1 SWOSD_WindowPrm

This structure describes the properties of the window parameters.

Fields	Data Type	Description
`startX`	`UInt16`	Relative to start of video window in pixels. Must be multiple of 2.
`startY`	`UInt16`	Relative to start of video window in lines.
`width`	`UInt16`	Width of the window in pixels. Must be multiple of 2.
`height`	`UInt16`	Height of the window in lines.
`alpha`	`UInt8`	8 bit global alpha value.
`lineOffset[SWOSD_MAX_PLANES]`	`UInt32`	Line offset in pixels for graphics window buffer. Must be multiple of 4, recommended to be 32 for efficiency
`graphicsWindowAddr[SWOSD_MAX_PLANES]`	`Ptr`	Points to graphic window for Y and UV incase of YUV420SP. In case of YUV422ILE, single pointer will be provided. **Note:** Graphic window is assumed to be in non-tiled memory, so library expects non-tiled buffer addresses.

9.3.2 SWOSD_DynamicPrm

This structure encapsulates properties for each of the window parameters in which user can use it for changing these properties dynamically.

Fields	Data Type	Description
numWindows	UInt16	Number of windows to be blended per channel.
winPrm[SWOSD_MAX_WINDOWS]	SWOSD_WindowPrm	Parameters for each of the window.

9.3.3 SWOSD_StaticPrm

This structure is used to define the static parameters of window which are constant throughout the instance of OSD.

Fields	Data Type	Description
maxWidth	UInt16	Maximum width of the window.
maxHeight	UInt16	Maximum height of the window.
dataFormat	SWOSD_DataFormat	Supported color format specified by enum SWOSD_DataFormat. **Note:** User must configure the colour format for each channel. This is to make sure that the OSD library will prepare the memory layout such that maximum of two will be considered. The library expects all the input frames to be of same data format in one process call for blend frames. Only YUV420SP format is validated on DVR RDK, YUV422ILE format is validate only in stand-alone environment.
isTiledMem	Bool	If this flag is true, then the video data will be expected to be in Tiler buffer. **Note:**For YUV420SP case, it is assumed that Y data will be in Tiler 8 bit and UV will be in Tiler 16 bit container. For YUV422ILE, it is assumed to be in Tiler page mode/Tiled 8 bit.
isInterlaced	Bool	Specifies whether input video is progressive or interlaced. This feature is validated only at standalone level not at DVR level.
videoLineOffset[SWOSD_MAX_PLANES]	Uint32	Specifies strides for both Y and UV components in video buffer. OSD library expects this parameter in no. of pixels. If isTiledMem is enabled, OSD library takes care of offsets accordingly.

9.3.4 SWOSD_Obj

This structure specifies the algorithm pointers required for OSD library

Fields	Data Type	Description

`algHndl`	`void *`	Pointer holds address of OSD algorithm handle.
`acquire`	`void (*)(void *)`	Acquire function pointer to acquire SIMCOP resource before blending.
`release`	`void (*)(void *)`	Release function pointer to release SIMCOP resource after blending.

9.3.5 SWOSD_CreatePrm

This structure specifies the create time parameters that used to create the OSD library instance.

Fields	Data Type	Description
`numChannels`	`UInt16`	Specifies the maximum number of channels(video channels) supported for the current instance. Current algorithm supports
`colorKey[SWOSD_MAX_PLANES]`	`SWOSD_WindowPrm`	8 bit color key for both Luma and Chroma. If Graphics window pixel value == *colorkey* then blending is not done for that pixel and video pixel is copied to output window. If Graphics window pixel value != *colorkey* then graphics window pixel is blended with video window pixel as usual
`transparencyEnable`	`Bool`	When transparency is enabled, - if Graphics window pixel value == ColorKey then blending is not done for that pixel and video pixel is copied to output window - if Graphics window pixel value != ColorKey then graphics window pixel is blended with video window pixel as usual When transparency is disabled, - graphics window pixel is blended with video window pixel as usual. Colorkey has no existence. TRUE – Enable transparency FALSE – Disable transparency
`useGlobalAlpha`	`Uint8`	Global Alpha flag. - 0 : disable global alpha - 1 : global alpha used for all channels and corresponding windows
`globalAlphaValue`	`Uint8`	Global alpha value in the range 0 – 255. Valid only if `useGlobalAlpha` flag is enabled. - 255 : Allows to see foreground - 0 : Allows to see background
`chStaticPrm[SWOSD_MAX_CHANNELS]`	`SWOSD_StaticPrm`	Static parameter structure for each video channel
`chDynamicPrm[SWOSD_MAX_CHANNELS]`	`SWOSD_DynamicPrm`	Dynamic parameter structure for each video channel

9.3.6 SWOSD_Frame

This structure specifies the video frame properties that is fed to OSD library

Fields	Data Type	Description
channelNum	UInt32	Specifies channel number associated with corresponding video frame
Fid	UInt32	Field id in case of interlaced video frames. Currently not supported.
Addr[SWOSD_MAX_PLANES]	Ptr	Pointers pointing to video buffers for both Y and C. These buffers can be either in Tiled or non-tiled. OSD library expects the address to be physical not virtual.

9.3.7 SWOSD_BlendFramePrm

This structure is used to pass to frame processing by OSD library which specifies actual no. of channels to be processed along with frame properties.

Fields	Data Type	Description
numFrames	UInt32	Specifies actual number of channels to be processed by OSD process function which blends video and graphics windows. Here the assumption is that only 1 frame per channel is allowed i.e. multiple frames of same channel will not be provided in `SWOSD_BlendFramePrm`
frames[SWOSD_MAX_FRAMES_PER_BLEND_FRAME]	SWOSD_Frame	Video frame properties for each of the video channel

9.3.8 SWOSD_MemAllocPrm

This structure is used to get and pass the persistent memories required by the OSD library. Application will call an API with this structure to get actual memory requirement and then calls an API that will be used by library to initialize memory pointers.

Fields	Data Type	Description
numMemBlocks	UInt32	No. of memory blocks requested by the library. Maximum supported value is `SWOSD_MAX_MEM_BLOCKS`
memBlockAddr[SWOSD_MAX_MEM_BLOCKS]	Ptr	Memory block addresses provided by the application
memBlockSize[SWOSD_MAX_MEM_BLOCKS]	UInt32	Sizes in bytes for each of the memory block
memBlockAlign[SWOSD_MAX_MEM_BLOCKS]	UInt32	Alignment requirement for each of the memory block

9.4 Interface functions

This section describes the Application Programming Interfaces (APIs) used in the SWOSD library.

9.4.1 SWOSD_open()

- Prototype

```
Int32 SWOSD_open(SWOSD_Obj *pObj, SWOSD_CreatePrm *pPrm)
```

- Description

  Open API to initialize SWOSD object parameters

  Operations:

  - Initializes algorithm handle
  - Assigning resources to the algorithm (eg. CPIS_Init() call)

- Parameters

Parameter	Description
`pObj`	Pointer to SWOSD object
`pPrm`	Pointer to create time parameter `SWOSD_CreatePrm`

- Return Value

  SWOSD_OK – If creation is successful

  SWOSD_ERROR – Failed to create instance of the OSD library

9.4.2 SWOSD_getMemAllocInfo ()

- Prototype

```
Int32 SWOSD_getMemAllocInfo(SWOSD_Obj *pObj, SWOSD_MemAllocPrm *pPrm)
```

- Description

  Application will call this API to get memory allocation required info from algorithm. Algorithm fills the memory requirements through the structure `SWOSD_MemAllocPrm`.

- Parameters

Parameter	Description
`pObj`	Pointer to SWOSD object
`pPrm`	Pointer to Memory alloc parameter structure `SWOSD_MemAllocPrm`

- Return Value

  None

9.4.3 SWOSD_setMemAllocInfo ()

- Prototype
  `Int32 SWOSD_setMemAllocInfo(SWOSD_Obj *pObj, SWOSD_MemAllocPrm *pPrm)`
- Description
  API for user to provide allocated pointers to the algorithm.
- Parameters

Parameter	Description
pObj	Pointer to SWOSD object
pPrm	Pointer to Memory alloc parameter structure SWOSD_MemAllocPrm

- Return Value
  None.

9.4.4 SWOSD_setDynamicPrm ()

- Prototype
  `Int32 SWOSD_setDynamicPrm(SWOSD_Obj *pObj, UInt32 chNum, SWOSD_DynamicPrm *pPrm)`
- Description
  This API is called by the user to change the properties of particular channel parameters.
- Parameters

Parameter	Description
pObj	Pointer to SWOSD object
chNum	Channel number for which the properties need to be dynamically changed
pPrm	Pointer to the dynamic parameter structure SWOSD_DynamicPrm

- Return Value
  SWOSD_OK – Dynamic parameters set successfully
  SWOSD_ERROR – Failed to apply dynamic parameter, probably due to incorrect value of new parameter

9.4.5 SWOSD_updateTransparencyEnableFlag ()

- Prototype
  `Int32 SWOSD_updateTransparencyEnableFlag (SWOSD_Obj *pObj, Bool transparencyEnable, Uint8 colorKey[SWOSD_MAX_PLANES])`

- Description
  This API is called by the user to set the transparency flag and color keys for all the channels at create time.

- Parameters

Parameter	Description
`pObj`	Pointer to SWOSD object
`transparencyEnable`	Transparency flag set by user
`colorKey[SWOSD_MAX_PLANES]`	Color keys for all the channels

- Return Value
  SWOSD_OK – Dynamic parameters set successfully
  SWOSD_ERROR – Failed to set create parameter, probably due to incorrect value of new parameter

9.4.6 SWOSD_updateGlobalAlpha ()

- Prototype
  `Int32 SWOSD_updateGlobalAlpha(SWOSD_Obj *pObj, Bool useGlobalAlpha, Uint8 globalAlphaValue)`
- Description
  This API is called by the user to set the global alpha value at create time.

- Parameters

Parameter	Description
`pObj`	Pointer to SWOSD object
`useGlobalAlpha`	Global alpha enable/disable flag
`globalAlphaValue`	Global alpha value

- Return Value
  SWOSD_OK – Dynamic parameters set successfully
  SWOSD_ERROR – Failed to apply dynamic parameter, probably due to incorrect value of new parameter

9.4.7 SWOSD_blendFrames ()

- Prototype
  `Int32 SWOSD_blendFrames(SWOSD_Obj *pObj, SWOSD_BlendFramePrm *pPrm)`
- Description
  API to all frames blending. This API performs actual OSD functinality. Here the Video buffer pointer along with DDR pointers should be populated in pPrm by the user/application before calling this API.
  Note: The API assumes all the channels/video frames to have same data format.
  i.e. Call `SWOSD_blendFrames()` to process X number of channels – all of YUV422I.
  Similarly call `SWOSD_blendFrames()` to process Y number of channels – all of YUV420SP

- Parameters

Parameter	Description
`pObj`	Pointer to SWOSD object
`pPrm`	Pointer to the structure `SWOSD_BlendFramePrm`

- Return Value
  SWOSD_OK – Successfully blended video with graphics windows
  SWOSD_ERROR – Failed to perform blend frames

9.4.8 SWOSD_close ()

- Prototype
  `Int32 SWOSD_close(SWOSD_Obj *pObj)`

- Description
  SWOSD close API which release the resource and deletes the handle.

- Parameters

Parameter	Description
`pObj`	Pointer to SWOSD object

- Return Value
  SWOSD_OK – Successfully deleted the handle
  SWOSD_ERROR – Failed to delete the handle