Advantiv® VSP Library Specifications

Preliminary 1.6

Analog Devices Confidential Information. Not for public distribution.

June 2016

Revision History

Revision	Date	Remarks
1.0	Apr, 15th, 2010	Draft, Initial revision
1.1	May 17th, 2011	The revised version after modification based on integration with repeater
1.2	July 14th, 2011	The revised version to add functions to support middleware
1.3	Mar 26th, 2012	Update 8003 related functions.
1.4	Sep 2nd , 2013	Added, copyright and product information sections, performed editorial corrections and updates, used ADV800x instead of ADV8002/3
1.5	Nov 15th, 2013	Merged section-5 & section-6 - Section-5 was prepared for ADV8002, Section-6 was for ADV8003. - New section-5 is for all ADV800x chips, including 8002/8003/8005, each ADV800x specials are explained in each API function. Modified 5.1.1 to add HPS for ADV8005
1.6	Nov 4th, 2014	Added minor changes like replace UCHAR by UINT16 in Processor structure VSP_TIMING_FORMAT & added parameters Aspect ratio 64x27 and 256x135 in ADIAPI_VspSetScaleMode & ADIAPI_SecondaryVspSetScaleMode API

Table of contents

PRODUCT INFORMATION **5**
ANALOG DEVICES WEB SITE 5
ENGINEERZONE 5
COPYRIGHT INFORMATION **5**
DISCLAIMER 5
TRADEMARK AND SERVICE MARK NOTICE 5
1 INTRODUCTION **5**
1.1 PURPOSE 6
1.2 SCOPE 6
2 SOFTWARE ARCHITECTURE **7**
2.1 FOLDER STRUCTURE 8
3 CONFIGURATION **9**
3.1 COMPILATION SWITCHES 10
3.2 USER-SUPPLIED CONFIGURATION FILES 10
3.3 USER-SUPPLIED INITIALIZATION DATA 11
4 USAGE **12**
5 ADV800X VSP LIBRARY APIS **13**
5.1 GENERAL VSP APIS 13
5.1.1 ADIAPI_VspInit *13*
5.1.2 ADIAPI_VspRouteCfg *14*
5.1.3 ADIAPI_VspInOutCfg *15*
5.1.4 ADIAPI_ManVSPInOutCfg *17*
5.1.5 ADIAPI_SecondaryVspInOutCfg *19*
5.1.6 ADIAPI_ManSecondaryVSPInOutCfg *21*
5.1.7 ADIAPI_PrimaryVSPEnableMNR *23*
5.1.8 ADIAPI_PrimaryVSPEnableSharpness *24*
5.1.9 ADIAPI_PrimaryVSPEnableBNR *25*
5.1.10 ADIAPI_PrimaryVSPEnableRNR *26*
5.1.11 ADIAPI_PrimaryVSPEnableUELA *27*
5.1.12 ADIAPI_PrimaryVSPEnableCadence *28*
5.1.13 ADIAPI_PrimaryVSPEnableChroma *28*
5.1.14 ADIAPI_PrimaryVSPEnableCUE *29*
5.1.15 ADIAPI_PrimaryVSPEnablePanorama *29*
5.1.16 ADIAPI_SecondaryVSPEnablePanorama *30*
5.1.17 ADIAPI_ManPrimaryVSPP2I *31*
5.1.18 ADIAPI_ManSecondaryVSPP2I *31*
5.1.19 ADIAPI_PrimaryVSPMarginColor *32*
5.1.20 ADIAPI_SecondaryVSPMarginColor *32*
5.1.21 ADIAPI_VspRebootEntry *33*
5.1.22 ADIAPI_VspRebootExit *33*
5.1.23 ADIAPI_VspGenRebootEntry *34*

5.1.24 ADIAPI_VspGenRebootExit 34
5.1.25 ADIAPI_VspSetScaleMode 35
5.1.26 ADIAPI_SecondaryVspSetScaleMode 36
5.1.27 ADIAPI_VSPSetMemMode 36
5.1.28 ADIAPI_SecondaryVSPSetMemMode 37
5.1.29 ADIAPI_SecondaryVspRebootEntry 37
5.1.30 ADIAPI_SecondaryVspRebootExit 38
5.1.31 ADIAPI_SecondaryVspGenRebootEntry 39
5.1.32 ADIAPI_SecondaryVspGenRebootExit 39
5.1.33 ADIAPI_PrimaryVspGetMaxPixelSize 40
5.1.34 ADIAPI_SecondaryVspGetMaxPixelSize 40
5.1.35 ADIAPI_PrimaryVSPSetFrameBuffer 41
5.1.36 ADIAPI_SecondaryVSPSetFrameBuffer 42
5.1.37 ADIAPI_PrimaryVSPSetVfmPeriod 42
5.1.38 ADIAPI_EnablePVSPFrameTrackMode 43
5.2 GENERAL VSP ENUMERATION LIST 44
5.2.1 VSP_VIDEO_MODE 44
5.3 STANDARD DEFINITION PROCESSOR STRUCTURES 47
5.3.1 VSP_TIMING_FORMAT 47
6 NOTIFICATION EVENTS 49

PRODUCT INFORMATION

Product information can be obtained from the Analog Devices Web site and other Web sources.

Analog Devices Web Site

The Analog Devices Web site, www.analog.com, provides information about a broad range of productsanalog integrated circuits, amplifiers, converters, and digital signal processors. To access a complete technical library for each video product family, go to http://www.analog.com/en/audiovideo-products/products/index.html.
Also note, MyAnalog.com is a free feature of the Analog Devices Web site that allows customization of a Web page to display only the latest information about products you are interested in. You can choose to receive weekly e-mail notifications containing updates to the Web pages that meet your interests, including documentation errata against all manuals. MyAnalog.com provides access to books, application notes, data sheets, code examples, and more. Visit MyAnalog.com to sign up. If you are a registered user, just log on. Your user name is your e-mail address.

Engineer Zone

Engineer Zone is a technical support forum from Analog Devices. It allows you direct access to Analog Devices technical support engineers. You can search FAQs and technical information to get quick answers to your questions about Analog Devices video products at http://ez.analog.com/community/video.

COPYRIGHT INFORMATION

DISCLAIMER

Analog Devices, Inc. (ADI) reserves the right to change this product without prior notice. Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use; nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under the patent rights of Analog Devices, Inc.

The information contained in this document is proprietary of ADI. This document must not be made available to anybody other than the intended recipient without the written permission of ADI.
The content of this document is believed to be correct. If any errors are found within this document or if clarification is needed, contact the video community on EZ forum (http://ez.analog.com/community/video).
TRADEMARK AND SERVICE MARK NOTICE
The Analog Devices logo is a registered trademark of Analog Devices, Inc. The Advantiv® and Blackfin® are registered trademarks of Analog Devices Inc. All other brand and product names are trademarks or service marks of their respective owners.
All other brand and product names are trademarks or service marks of their respective owners.

Analog Devices' Trademarks and Service Marks may not be used without the express written consent of Analog Devices, such consent only to be provided in a separate written agreement signed by Analog Devices. Subject to the foregoing, such Trademarks and Service Marks must be used according to Analog Devices' Trademark Usage guidelines. Any licensee wishing to use Analog Devices' Trademarks and Service Marks must obtain and follow these guidelines for the specific marks at issue.

1 Introduction

1.1 *Purpose*

This document serves as a programmer's reference for using and utilising various aspects of the ADI A/V VSP Library, as a module of the ATV software stack, with a detailed list of Application Program Interface (APIs) and associated data structures, macros and defines.

The document only describes the API specification and any parts of the VSP library that are particular to the VSP module. For a complete description of the ATV software stack, please refer to the ATV software architecture specifications document.

1.2 *Scope*

This document is intended to be used in conjunction with, and it largely depends on, the ATV software architecture specifications document. It is strongly advised to go through the ATV software architecture specifications before using this document to get an understanding of the overall system control flow and how the VSP library fits into this architecture.

2 Software Architecture

The VSP library is a collection of APIs that provide a consistent interface to a variety of VSP hardware modules. The APIs are designed so that they can be used on different, closely-related, VSP chip families, thus enhancing application portability.

The library is a software layer that sits between the application and the VSP hardware. The library is intended to serve two purposes:

1- Provide the application with a set of APIs that can be used to configure VSP hardware without the need for low-level register access. This makes the application portable across different revisions of the hardware and even across different hardware modules.

2- Provide basic services to aid the application in controlling the VSP module, such as route control and status information.

The driver does not, in any shape or form, alter the configuration or state of VSP module on its own. It is the responsibility of the application to poll for status, prepare and send info-frames, compose/parse EDID, and configure VSP module accordingly. The library acts only as an abstraction layer between the application and the hardware.

As an example, the application is responsible for the following:

- Configuring video interface
- Configuring primary VSP
- Configuring secondary VSP
- Handling of video enhancement
- Setting up internal video route
- Configuring cadence detection

The application should access the VSP module only through the VSP library's exported APIs. This enhances application portability and reduces dependencies on any particular VSP hardware. If the application chooses to directly access VSP module hardware (e.g., using I2C or using VSP HAL macros,) this should be done in a very limited scope, when it is absolutely necessary, and it should be understood that this practice will effectively reduce application portability.

2.1 *Folder structure*

The collective files of the VSP library reside in a sub-folder under the ATV software main folder. The library is supplied in source format. All source files are in standard ANSI C to simplify porting to any platform. The VSP library folder is structured as follows:

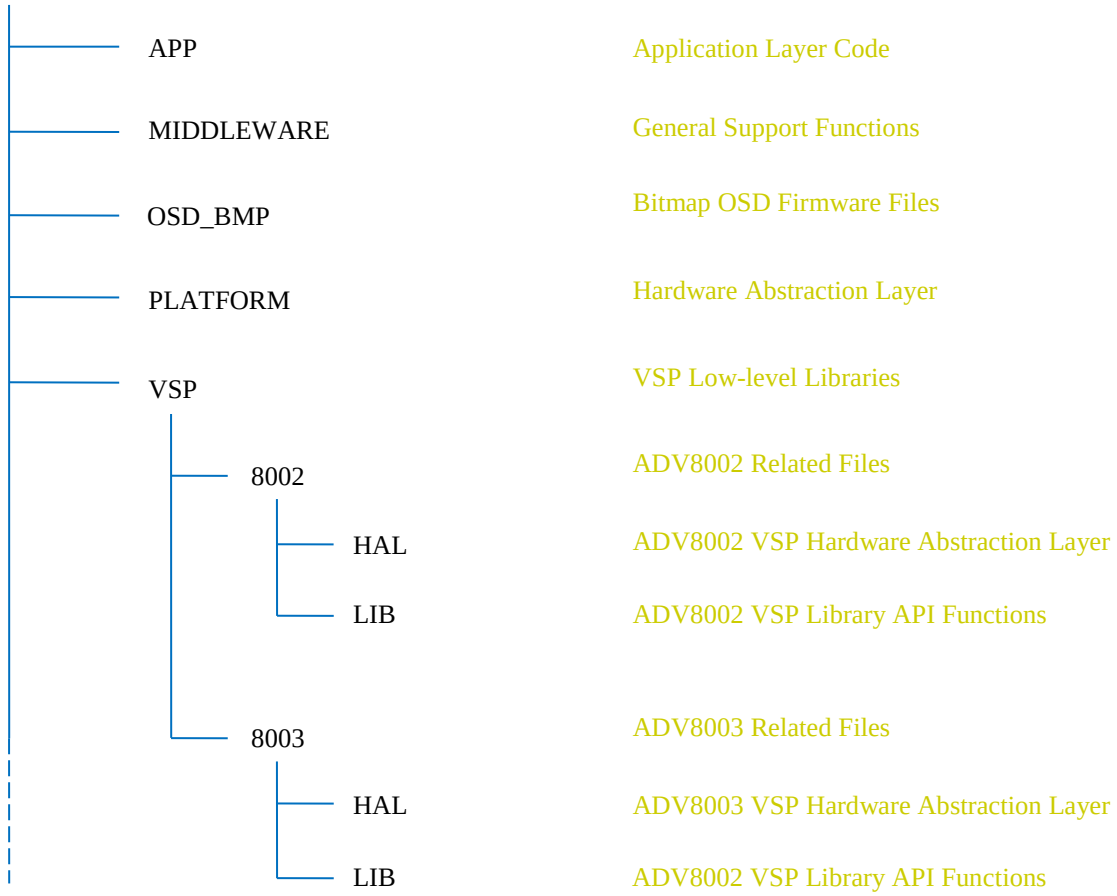

Figure 1 – Folder Structure

/ VSP / 800x

The VSP module root folder contains all the header files needed by upper level layers (the application) to access the VSP library. This is the only folder in the VSP folders tree that should be "seen" by the application. Only one file, "vsp_lib.h", exists in this folder. This file contains all necessary prototypes, macros and defines needed to access the VSP software library.

/ VSP / 800x / HAL

This folder contains the VSP Hardware Abstraction Layer files and sub-folders. The VSP HAL APIs and macros should not be referenced directly by the application.

/ VSP / 800x / LIB

This folders contains the VSP module library exported APIs and interrupt service routine.

3 Configuration

Some of the receiver module parameters can be configured by the application at compile time. Parameters such as I2C device addresses can be specified by the application if required.

A combination of compile-time switches and user-supplied configuration files are used to configure the receiver library as described in the following sections.

3.1 *Compilation Switches*

Some parameters of the VSP library are configured through compiler switches (symbols). The following is a list of compiler switches required to compile the receiver library:

VSP__DEVICE = 800x

This is the target VSP device. Possible values are 8002 for the ADV8002, 8003 for the ADV8003 and 8005 for the ADV8005.

VSP_USER_CONFIG = 1

This switch is used to enable the application to configure the VSP library at compile time using a configuration file. If set to 1, the VSP library will include an application-supplied header file named "vsp_config.h" that contains all the configuration parameters needed by the library. This file must reside outside the VSP folders tree (typically in the application folder) and the path for this file must be set in the compiler include path. If this switch is set to 0, the application does not need to provide the "vsp_config.h" file and a default value for all configuration parameters will be used. The User-supplied configuration files section describes the structure of the receiver configuration file and the default values for all configuration parameters.

VSP_USER_INIT = 1

This switch is used to enable the application to initialize the VSP hardware with certain values during runtime. If this switch is set to 1, in addition to default initialization sequence, the library will initialize the receiver hardware using a set of application-supplied tables containing register values that needs to be set during initialization. The structure of this table is defined in the User-supplied initialization data section. This initialization data must reside outside the VSP folders tree (typically in the application folder). If this switch is set to 0, the application does not need to provide the initialization tables and only the default initialization values will be used. It should be noted that the use of application-supplied initialization data is highly discouraged as it binds the application to a specific VSP hardware module. This feature is provided as a last resort to initialize the VSP hardware when all else fails.

3.2 *User-supplied Configuration Files*

If the compilation switch VSP_USER_CONFIG is set to 1 (Refer to the Compilation switches section), the application must provide a header file named "vsp_config.h" that contains RX module compile-time configuration settings that need to be adjusted according to the target application or platform.

The configuration parameters that need to be defined in the "vsp_config.h" file are described below along with the default values that will be used if the VSP_USER_CONFIG switch is set to 0.

#define VSP_USE_NOTIFICATION 1

This macro is used if the application wishes to be notified of status change. If this switch is set to 1, an application-defined function with a predefined prototype will be called from the library (from the ISR) to notify the application of such events. The details of the notification function and the events it receives from the library are defined in the Notification events section. If this switch is omitted or set to 0, the application will not be notified of any status changes and will have to poll the library for status information. Please see ADIAPI_VspIsr for more details.

3.3 *User-supplied Initialization Data*

The VSP library provides the ADIAPI_VspInit API to initialize the VSP hardware module. Some of the VSP module registers are initialized by this API to fixed values that cannot be controlled directly from the application. However, the application can specify additional settings to any of the VSP module registers by setting the VSP_USER_INIT compilation switch to 1. (Please see Compilation switches for details)

When VSP_USER_INIT is set to 1, the application must define two tables containing VSP registers initialization sequences. Those tables will be used by the ADIAPI_VspInit API to perform additional VSP hardware initialization after completing the default initialization sequence.

The structure of the two tables is as follows:

UCHAR UserVspRegInitTable [] = {device, register, value, …, 0, 0, 0};

UCHAR UserVspFieldInitTable[] = {device, register, mask, value, …, 0, 0, 0, 0};

The first table is used to initialize a set of VSP registers. The table consists of a concatenated list of 3-byte entries, where each entry specifies the required setting for one register. Each entry starts with the I2C device address, followed by the register address and the value to be written to that register. The next 3 bytes represent the initialization value for the next register and so on. The end of the table is marked by a three 0 bytes.

The second table is used to initialize a set of VSP registers fields. The table consists of a concatenated list of 4-byte entries, where each entry specifies the required setting for a register field. Each entry starts with the I2C device address, followed by the register address, followed by the filed mask then the field value to be written to that register. The next 4 bytes represent the initialization value for the next field and so on. The end of the table is marked by a four 0 bytes. An example is defined below:

```
UCHAR UserVspFieldInitTable[] = {
0xE0, 0x14, 0xF0, 0x90,      /* Set device 0xE0 register 0x14 bits 4-7 = 0x09 */
0xE1, 0x55, 0x60, 0x20,      /* Set device 0xE1 register 0x55 bits 5-6 = 0x01 */
0x00, 0x00, 0x00, 0x00              /* Table end */
};
```

It should be noted that the use of application-supplied initialization data is highly discouraged as it binds the application to a specific VSO hardware module. This feature is provided only as a last resort to initialize the VSP hardware when all else fails.

4 Usage

The VSP library is designed to be a self-contained module. The library, however, relies on the platform hardware abstraction layer (The HAL, see ATV architecture specifications document) to gain access to the underlying hardware.
It is the responsibility of the library's user to provide platform HAL APIs, either in source format or as a binary module to be linked with the receiver library. All the data types used by the library must also be provided as specified in the ATV architecture specifications document.

The steps needed to compile the receiver library are summarized below

- Provide platform HAL APIs and data types
- Set the compilation switches as described in Compilation switches section
- If required, create the notification events handling function as described in the User-supplied configuration files and Notification events
- If required, create the user-configuration file and user-initialization tables as described in the User-supplied configuration files and User-supplied initialization data sections.

The steps that need to be taken by the application at run time are summarized below:

- Initialize receiver hardware and software stack. This should be done by calling the ADIAPI_VspInit API.
- Configure the VSP module using the library APIs. This can include, for example, setting the internal route, setting the primary vsp, setting the secondary vsp, etc.
- If a receiver interrupt is detected, the application should call the Receiver ISR ADIAPI_VspIsr to process the interrupt. Since the VSP state usually changes following an interrupt, the application should enquire about VSP state following interrupt processing. Alternatively, if the VSP_USE_NOTIFICATION macro is used, the library will call-back the application's notification events handling function as described in the User-supplied configuration files section. Please see the Notification events section and ADIAPI_VspIsr API for more details.

A sample application is provided in the /APP folder. This application perform all of the above steps starting from the main entry function, main().

5 ADV800x VSP Library APIs

The VSP library provides a comprehensive set of APIs to control, configure and provide status on all aspects of the VSP module. All library APIs are available for the application software to use at any time.

For a description of the ATV APIs structure, data types and return values please refer to the ATV software architecture specification document.

5.1 *General VSP APIs*

5.1.1 ADIAPI_VspInit

Description

This function is used to initialize Primary VSP and Secondary VSP after power-up or reset.

ADV8005 adds HPS module. It is initialized in this function also.

Synopsis

```
#include "vsp_lib.h"
ATV_ERR     ADIAPI_VspInit ()
```

Parameters

None

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.2 ADIAPI_VspRouteCfg

Description

This function is used to set the internal route of VSP.

Synopsis

```
#include "vsp_lib.h"
ATV_ERR     ADIAPI_VspRouteCfg ()
```

Parameters

None

Return value

ATVERR_OK
Operation completed successfully

ADIERR_INV_PARM
Invalid Parameter

Remarks

None

5.1.3 ADIAPI_VspInOutCfg

Description

This function is used to configure the primary VSP's input and output timing format.

Synopsis

ADV8002
#include "vsp_lib.h"
ATV_ERR ADIAPI_VspInOutCfg (pVSP_AUTO_TIMING_CHARAC pVideoInputCharac, pVSP_AUTO_TIMING_CHARAC pVideoOutputCharac)

ADV8003 and ADV8005
#include "vsp_lib.h"
ATV_ERR ADIAPI_VspInOutCfg (pVSP_VIDEO_TIMING_CHARAC VideoInputCharac, pVSP_VIDEO_TIMING_CHARAC VideoOutputCharac)

Parameters

VideoInputCharac

Set the video input format of primary VSP. Possible input video format is defined in enum VSP_VIDEO_FMT.

It is the pointer to a structure to describe the video input timing format of Primary VSP. In the structure, the following parameters need to be set.

- *VideoFmt: Input video timing format defined in enum VSP_VIDEO_FMT.*
- *CropEn: Enable cropping*
- *CropStartX: Start position (x,y) of cropping*
- *CropStartY: Start position (x,y) of cropping*
- *CropH: Horizontal pixels of cropping*
- *CropV: Vertical lines of cropping*

VideoOutputCharac

It is the pointer to a structure to describe the video output timing format of Primary VSP. In the structure, the following parameters need to be set.

- *VideoFmt: Output video timing format defined in enum VSP_VIDEO_FMT.*
- *CfgMode: VSP configuration mode. The configuration modes are defined in enum VSP_CFG_MODE.*
  - *AUTO_MODE: Generating video output with processing of video input.*
  - *GAME_MODE: Extremely low latency mode with nearly zero frame latency.*
  - *BYPASS_MODE: Bypass video input to video output with no processing.*
  - *OSD_MODE: Generate video output without video input.*
  - *VIDEOGEN_MODE: Writing external OSD data into DDR2 memory.*
  - *LOWLATENCY_MODE: With low latency, generating video output with processing of video input.*
- *CropEn: Enable album mode*
- *CropStartX: Start position (x,y) of album output*

- *CropStartY: Start position (x,y) of album output*
- *CropH: Horizontal pixels of album output*
- *CropV: Vertical lines of album output*
- *OverscanH: Horizontal overscan percentage*
- *OverscanV: Vertical overscan percentage*

Return value

ATVERR_OK
Operation completed successfully

ADIERR_INV_PARM
Invalid Parameter

Remarks

None

5.1.4 ADIAPI_ManVSPInOutCfg

Description

This function is used to manually configure the primary VSP block.

Synopsis

ADV8002
#include "vsp_lib.h"
ATV_ERR ADIAPI_ManVSPInOutCfg(pVSP_MANUAL_TIMING_CHARAC pVideoInputCharac,
pVSP_MANUAL_TIMING_CHARAC pVideoOutputCharac)

ADV8003 and ADV8005
#include "vsp_lib.h"
ATV_ERR ADIAPI_ManVSPInOutCfg(pVSP_TIMING_FORMAT pVideoInputTiming,
pVSP_OUTPUT_TIMING pVideoOutputCharac)

Parameters

pVideoInputTiming
Pointer to a structure to describe the timing, describing video output timing format of Primary VSP.

- *TimingFmt = Output video timing structure. Pointer to a structure to describe the video input timing format of Primary VSP.*
  - *HFrontPorch = Horizontal Front Porch Number*
  - *HSyncDur = Horizontal Sync Duration Number*
  - *HBackPorch = Horizontal Back Porch Number*
  - *VFrontPorch = Vertical Front Porch Number*
  - *VSyncDur = Vertical Sync Duration Number*
  - *VBackPorch = Vertical Back Porch Number*
  - *HActive = Horizontal Active Pixel Number*
  - *VActive = Vertical Active Pixel Number*
  - *IsIntl = Set to TRUE if input is interlaced*
  - *FrameRate = Frame Rate Number*
  - *VideoID = Video ID*
  - *HPol = HSync Polarity*
  - *VPol = VSync Polarity*
  - *VfmPeriod = Vfm Period Number*
- *CropEn: Enable cropping*
- *CropStartX: Start position (x,y) of cropping*
- *CropStartY: Start position (x,y) of cropping*
- *CropH: Horizontal pixels of cropping*
- *CropV: Vertical lines of cropping*

pVideoOutputCharac

Pointer to a structure to describe the timing, describing video output timing format of Primary VSP.

- *TimingFmt = Output video timing structure*
- *CfgMode = VSP working mode*
  - *AUTO_MODE: Generating video output with processing of video input.*
  - *GAME_MODE: Extremely low latency mode with nearly zero frame latency.*
  - *BYPASS_MODE: Bypass video input to video output with no processing.*
  - *OSD_MODE: Generate video output without video input.*
  - *VIDEOGEN_MODE: Writing external OSD data into DDR2 memory.*
  - *LOWLATENCY_MODE: With low latency, generating video output with processing of video input.*
- *CropEn = Enable album mode*
- *CropStartX = Start position (x,y) of album output*
- *CropStartY = Start position (x,y) of album output*
- *CropH = Horizontal pixels of album output*
- *CropV = Vertical lines of album output*
- *OverscanH = Horizontal overscan percentage*
- *OverscanV = Vertical overscan percentage*

Return value

ATVERR_OK

Operation completed successfully

ADIERR_INV_PARM

Invalid Parameter

Remarks

None

5.1.5 ADIAPI_SecondaryVspInOutCfg

Description

This function is used to configure the secondary VSP's input and output timing format.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_SecondaryVspInOutCfg (pVSP_AUTO_TIMING_CHARAC VideoInputCharac,
pVSP_VIDEO_TIMING_CHARAC VideoOutputCharac)

Parameters

VideoInputCharac

Set the video input format of secondary VSP. Possible input video format is defined in enum VSP_VIDEO_FMT.

It is the pointer to a structure to describe the video input timing format of Secondary VSP. In the structure, the following parameters need to be set.

- *VideoFmt: Input video timing format defined in enum VSP_VIDEO_FMT.*
- *CropEn: Enable cropping*
- *CropStartX: Start position (x,y) of cropping*
- *CropStartY: Start position (x,y) of cropping*
- *CropH: Horizontal pixels of cropping*
- *CropV: Vertical lines of cropping*

VideoOutputCharac

It is the pointer to a structure to describe the video output timing format of secondary VSP. In the structure, the following parameters need to be set.

- *VideoMode: Output video timing format defined in enum VSP_VIDEO_FMT.*
- *CfgMode: VSP configuration mode. The configuration modes are defined in enum VSP_CFG_MODE.*
  - *AUTO_MODE: Generating video output with processing of video input.*
  - *GAME_MODE: Extremely low latency mode with nearly zero frame latency.*
  - *BYPASS_MODE: Bypass video input to video output with no processing.*
  - *OSD_MODE: Generate video output without video input.*
  - *VIDEOGEN_MODE: Writing external OSD data into DDR2 memory.*
  - *LOWLATENCY_MODE: With low latency, generating video output with processing of video input.*
- *CropEn: Enable album mode*
- *CropStartX: Start position (x,y) of album output*
- *CropStartY: Start position (x,y) of album output*
- *CropH: Horizontal pixels of album output*
- *CropV: Vertical lines of album output*
- *OverscanH: Horizontal overscan percentage*

- *OverscanV: Vertical overscan percentage*

Return value

ATVERR_OK
Operation completed successfully

ADIERR_INV_PARM
Invalid Parameter

Remarks

None

5.1.6 ADIAPI_ManSecondaryVSPInOutCfg

Description

This function is used to manually configure the secondary VSP block.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_ManSecondaryVSPInOutCfg(pVSP_MANUAL_TIMING_CHARAC VideoInputCharac, pVSP_OUTPUT_TIMING pVideoOutputCharac)

Parameters

VideoInputCharac

Pointer to a structure to describe the timing, describing video output timing format of Secondary VSP.

- *TimingFmt = Output video timing structure. Pointer to a structure to describe the video input timing format of Secondary VSP.*
  - *HFrontPorch = Horizontal Front Porch Number*
  - *HSyncDur = Horizontal Sync Duration Number*
  - *HBackPorch = Horizontal Back Porch Number*
  - *VFrontPorch = Vertical Front Porch Number*
  - *VSyncDur = Vertical Sync Duration Number*
  - *VBackPorch = Vertical Back Porch Number*
  - *HActive = Horizontal Active Pixel Number*
  - *VActive = Vertical Active Pixel Number*
  - *IsIntl = Set to TRUE if input is interlaced*
  - *FrameRate = Frame Rate Number*
  - *VideoID = Video ID*
  - *HPol = HSync Polarity*
  - *VPol = VSync Polarity*
  - *VfmPeriod = Vfm Period Number*
- *CropEn: Enable cropping*
- *CropStartX: Start position (x,y) of cropping*
- *CropStartY: Start position (x,y) of cropping*
- *CropH: Horizontal pixels of cropping*
- *CropV: Vertical lines of cropping*

pVideoOutputCharac

Pointer to a structure to describe the timing, describing video output timing format of Secondary VSP.

- *TimingFmt = Output video timing structure*
- *CfgMode = VSP working mode*
  - *AUTO_MODE: Generating video output with processing of video input.*
  - *GAME_MODE: Extremely low latency mode with nearly zero frame latency.*
  - *BYPASS_MODE: Bypass video input to video output with no processing.*

    - *OSD_MODE: Generate video output without video input.*
    - *VIDEOGEN_MODE: Writing external OSD data into DDR2 memory.*
    - *LOWLATENCY_MODE: With low latency, generating video output with processing of video input.*
- *AlbumEn = Enable album mode*
- *AlbumStartX = Start position (x,y) of album output*
- *AlbumStartY = Start position (x,y) of album output*
- *AlbumH = Horizontal pixels of album output*
- *AlbumV = Vertical lines of album output*
- *OverscanH = Horizontal overscan percentage*
- *OverscanV = Vertical overscan percentage*

Return value

ATVERR_OK
Operation completed successfully

ADIERR_INV_PARM
Invalid Parameter

Remarks

None

5.1.7 ADIAPI_PrimaryVSPEnableMNR

Description

This function is used to enable the MNR (Mosquito Noise Reduction) of primary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_PrimaryVSPEnableMNR(BOOL Enable, VSP_NR_LEVEL level)

Parameters

Enable

TRUE	Enable MNR
FALSE	Disable MNR

Level

LOW	Low level noise reduction
MIDDLE	Middle level noise reduction
HIGH	High level noise reduction

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.8 ADIAPI_PrimaryVSPEnableSharpness

Description

This function is used to enable the Sharpness of primary VSP.

Synopsis

ADV8002
#include "vsp_lib.h"
ATV_ERR ADIAPI_PrimaryVSPEnableSharpness(BOOL Enable, UINT8 Detail_gain, UINT8 Edge_gain)

ADV8003 and ADV8005
#include "vsp_lib.h"
ATV_ERR ADIAPI_PrimaryVSPEnableSharpness(BOOL Enable, UINT16 gain)

Parameters

ADV8002
Enable
TRUE Enable Sharpness
FALSE Disable Sharpness
Detail_gain
0~127 detail gain of sharpness
Edge_gain
0~127 edge gain of sharpness

ADV8003 and ADV8005
Enable
TRUE Enable Sharpness
FALSE Disable Sharpness
gain
-2048~2047 gain of sharpness, negative is represented in complemental code

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.9 ADIAPI_PrimaryVSPEnableBNR

Description

This function is used to enable the BNR (Block Noise Reduction) of primary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_PrimaryVSPEnableBNR(BOOL Enable, VSP_NR_LEVEL Level)

Parameters

Enable

TRUE	Enable BNR
FALSE	Disable BNR

Level

LOW	Low level noise reduction
MIDDLE	Middle level noise reduction
HIGH	High level noise reduction

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.10 ADIAPI_PrimaryVSPEnableRNR

Description

This function is used to enable the RNR (Random Noise Reduction) of primary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_PrimaryVSPEnableRNR(BOOL Enable, VSP_NR_LEVEL level)

Parameters

Enable

TRUE	Enable RNR
FALSE	Disable RNR

Level

LOW	Low level noise reduction
MIDDLE	Middle level noise reduction
HIGH	High level noise reduction

Return value

ATVERR_OK
Operation completed successfully

ATVERR_FAILED
Indicates the function is not completed successfully

Remarks

RNR can't be enabled for 1080p input otherwise output would crash.

5.1.11 ADIAPI_PrimaryVSPEnableUELA

Description

This function is used to enable the UELA of primary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_PrimaryVSPEnableUELA(BOOL Enable)

Parameters

Enable
TRUE Enable UELA
FALSE Disable UELA

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.12 ADIAPI_PrimaryVSPEnableCadence

Description

This function is used to enable the film detection of primary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_PrimaryVSPEnableCadence(BOOL Enable)

Parameters

Enable
TRUE Enable film detection
FALSE Disable film detection

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.13 ADIAPI_PrimaryVSPEnableChroma

Description

This function is used to enable the chroma of primary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_PrimaryVSPEnableChroma(BOOL Enable)

Parameters

Enable
TRUE Enable Chroma
FALSE Disable Chroma

Return value

ATVERR_OK

Operation completed successfully

Remarks

None

5.1.14 ADIAPI_PrimaryVSPEnableCUE

Description

This function is used to enable the CUE of primary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_PrimaryVSPEnableCUE(BOOL Enable)

Parameters

Enable
TRUE Enable CUE
FALSE Disable CUE

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.15 ADIAPI_PrimaryVSPEnablePanorama

Description

This function is used to enable the Panorama of primary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_PrimaryVSPEnablePanorama(BOOL Enable , UINT16 Value)

Parameters

Enable

TRUE Enable Panorama
FALSE Disable Panorama

Value

The width of the output video frame not stretched

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.16 ADIAPI_SecondaryVSPEnablePanorama

Description

This function is used to enable the Panorama of secondary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_SecondaryVSPEnablePanorama(BOOL Enable , UINT16 Value)

Parameters

Enable

TRUE Enable Panorama
FALSE Disable Panorama

Value

The width of the output video frame not stretched

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.17 ADIAPI_ManPrimaryVSPP2I

Description

This function is used to configure P2I of primary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_ManPrimaryVSPP2I(BOOL enable, VSP_VIDEO_FMT vid)

Parameters

Enable
TRUE Enable P2I
FALSE Disable P2I
Vid
Primary VSP Output Progressive VID

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.18 ADIAPI_ManSecondaryVSPP2I

Description

This function is used to configure P2I of secondary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_ManSecondaryVSPP2I(BOOL enable, VSP_VIDEO_FMT vid)

Parameters

Enable
TRUE Enable P2I
FALSE Disable P2I
Vid
Primary VSP Output Progressive VID

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.19 ADIAPI_PrimaryVSPMarginColor

Description

This function is used to set margin color of Primary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_PrimaryVSPMarginColor(UINT32 MarginColor)

Parameters

MarginColor
The margin color in YCbCr space

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.20 ADIAPI_SecondaryVSPMarginColor

Description

This function is used to set margin color of Secondary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_SecondaryVSPMarginColor(UINT32 MarginColor)

Parameters

MarginColor
The margin color in YCbCr space

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.21 ADIAPI_VspRebootEntry

Description

Entry of Primary VSP Reboot protocol.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_VspRebootEntry(void)

Parameters

None

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.22 ADIAPI_VspRebootExit

Description

Exit of Primary VSP Reboot protocol.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_VspRebootExit(void)

Parameters

None

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.23 ADIAPI_VspGenRebootEntry

Description

Entry of Primary VSP Gently Reboot protocol.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_VspGenRebootEntry(void)

Parameters

None

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.24 ADIAPI_VspGenRebootExit

Description

Exit of Primary VSP Gently Reboot protocol.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_VspGenRebootExit (void)

Parameters

None

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.25 ADIAPI_VspSetScaleMode

Description

Set the aspect ratio of Primary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_VspSetScaleMode(VSP_ASPECT_RATIO AspectRatio)

Parameters

AspectRatio

- SNORMAL Normal aspect ratio
- S4TO3 4:3 aspect ratio
- S16TO9 16:9 aspect ratio
- S64TO27 64:27 aspect ratio
- S256TO135 256:135 aspect ratio
- SWH Horizontal wide screen aspect ratio
- SWV Vertical wide screen aspect ratio

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.26 ADIAPI_SecondaryVspSetScaleMode

Description

Set the aspect ratio of Secondary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_SecondaryVspSetScaleMode (VSP_ASPECT_RATIO AspectRatio)

Parameters

AspectRatio

- SNORMAL Normal aspect ratio
- S4TO3 4:3 aspect ratio
- S16TO9 16:9 aspect ratio
- S64TO27 64:27 aspect ratio
- S256TO135 256:135 aspect ratio
- SWH Horizontal wide screen aspect ratio
- SWV Vertical wide screen aspect ratio

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.27 ADIAPI_VSPSetMemMode

Description

Set memory storage mode of Primary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_VSPSetMemMode(VSP_MEM_MODE Mode)

Parameters

Mode

- TWELVE444 12-bit 444 style (32-bit per pixel)
- EIGHT444 8-bit 444 style (24-bit per pixel)
- TWELVE422 12-bit 422 style (24-bit per pixel)
- EIGHT422 8-bit 422 style (16-bit per pixel)

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.28 ADIAPI_SecondaryVSPSetMemMode

Description

Set memory storage mode of Secondary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_SecondaryVSPSetMemMode(VSP_MEM_MODE Mode)

Parameters

Mode

- TWELVE444 12-bit 444 style (32-bit per pixel, still treat as EIGHT444)
- EIGHT444 8-bit 444 style (24-bit per pixel)
- TWELVE422 12-bit 422 style (24-bit per pixel, still treat as EIGHT422)
- EIGHT422 8-bit 422 style (16-bit per pixel)

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.29 ADIAPI_SecondaryVspRebootEntry

Description

Entry of Secondary VSP Reboot protocol.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_SecondaryVspRebootEntry(void)

Parameters

None

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.30 ADIAPI_SecondaryVspRebootExit

Description

Exit of Secondary VSP Reboot protocol.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_SecondaryVspRebootExit(void)

Parameters

None

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.31 ADIAPI_SecondaryVspGenRebootEntry

Description

Entry of Secondary VSP Gently Reboot protocol.

Synopsis

```
#include “vsp_lib.h”
ATV_ERR ADIAPI_SecondaryVspGenRebootEntry(void)
```

Parameters

None

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.32 ADIAPI_SecondaryVspGenRebootExit

Description

Exit of Secondary VSP Gently Reboot protocol.

Synopsis

```
#include “vsp_lib.h”
ATV_ERR ADIAPI_SecondaryVspGenRebootExit(void)
```

Parameters

None

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.33 ADIAPI_PrimaryVspGetMaxPixelSize

Description

Get Maximum Pixel Size of Primary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_PrimaryVspGetMaxPixelSize(UINT16 InHactive,
UINT16 InVactive, UINT16 OutHactive, UINT16 OutVactive, VSP_MEM_MODE *MaxPixelSize)

Parameters

InHactive
Input timing Horizontal Active Lines
InVactive
Input timing Vertical Active Lines
OutHactive
Output timing Horizontal Active Lines
OutVactive
Output timing Vertical Active Lines
MaxPixellSize
the max byte size that can be supported per pixel

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.34 ADIAPI_SecondaryVspGetMaxPixelSize

Description

Get Maximum Pixel Size of Secondary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_SecondaryVspGetMaxPixelSize (UINT16 InHactive,

UINT16 InVactive, UINT16 OutHactive, UINT16 OutVactive, VSP_MEM_MODE *MaxPixelSize)

Parameters

InHactive
Input timing Horizontal Active Lines
InVactive
Input timing Vertical Active Lines
OutHactive
Output timing Horizontal Active Lines
OutVactive
Output timing Vertical Active Lines
MaxPixellSize
the max byte size that can be supported per pixel

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.35 ADIAPI_PrimaryVSPSetFrameBuffer

Description

Set Frame Buffer of Primary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_PrimaryVSPSetFrameBuffer(UINT32 FrameBufferSize, BOOL VinIntl)

Parameters

FrameBufferSize
The frame Buffer size for Primary VSP
VinIntl
Interlaced input

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.36 ADIAPI_SecondaryVSPSetFrameBuffer

Description

Set Frame Buffer of Secondary VSP.

Synopsis

#include "vsp_lib.h"
ATV_ERR ADIAPI_SecondaryVSPSetFrameBuffer(UINT32 FrameBufferSize)

Parameters

FrameBufferSize
The frame Buffer size for Secondary VSP

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.37 ADIAPI_PrimaryVSPSetVfmPeriod

Description

Set VFM Period of Primary VSP.

Synopsis

#include "vsp_lib.h"
ADIAPI_PrimaryVSPSetVfmPeriod(UCHAR Vic, UCHAR Idx)

Parameters

VIC
The VIC id that needs to generate corresponding VFM value

idx
0 to generate 23.97/59.94 Hz, 1 to generate the rest frequency value (24/50/60Hz ...)

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.1.38 ADIAPI_EnablePVSPFrameTrackMode

Description

Enable or Disable frame tracking mode of Primary VSP.

Synopsis

#include "vsp_lib.h"
ADIAPI_EnablePVSPFrameTrackMode(BOOL Enable)

Parameters

Enable
=1 to Enable Frame tracking, =0 to Disable Frame tracking

Return value

ATVERR_OK
Operation completed successfully

Remarks

None

5.2 *General VSP Enumeration List*

This section described all the enumerations that are used by the General VSP APIs.

5.2.1 VSP_VIDEO_MODE

Description

An enumeration of supported video formats

Synopsis

ADV8002	***ADV8003 and ADV8005***
#include "vsp_lib.h" ***typdef enum{*** ***VSP_576I_50HZ = 0,*** ***VSP_576P_50HZ,*** ***VSP_720P_50HZ,*** ***VSP_1080I_50HZ,*** ***VSP_1080P_50HZ,*** ***VSP_VGA_60HZ,*** ***VSP_480I_60HZ,*** ***VSP_480P_60HZ,*** ***VSP_720P_60HZ,*** ***VSP_1080I_60HZ,*** ***VSP_1080P_60HZ,*** ***VSP_1080P_24HZ*** ***} VSP_VIDEO_MODE***	***#include "vsp_lib.h"*** ***typdef enum{*** ***VSP_576I_50HZ = 0,*** ***VSP_576P_50HZ,*** ***VSP_720P_50HZ,*** ***VSP_1080I_50HZ,*** ***VSP_1080P_50HZ,*** ***VSP_VGA_60HZ,*** ***VSP_480I_60HZ,*** ***VSP_480P_60HZ,*** ***VSP_720P_60HZ,*** ***VSP_1080I_60HZ,*** ***VSP_1080P_60HZ,*** ***VSP_1080P_24HZ,*** ***VSP_1080P_25HZ,*** ***VSP_1080P_30HZ,*** ***VSP_SVGA_60HZ,*** ***VSP_XGA_60HZ,*** ***VSP_WXGA_60HZ,*** ***VSP_SXGA_60HZ,*** ***VSP_WXGA2_60HZ,*** ***VSP_UXGA_60HZ,*** ***VSP_WXGA3_60HZ,*** ***VSP_WUXGA_60HZ,*** ***VSP_240P_60HZ,*** ***VSP_288P_50HZ,*** ***VSP_1080IE_50HZ,*** ***VSP_1080I_100HZ,*** ***VSP_720P_100HZ,*** ***VSP_576P_100HZ,*** ***VSP_576I_100HZ,*** ***VSP_1080I_120HZ,***

	VSP_720P_120HZ, *VSP_480P_120HZ,* *VSP_480I_120HZ,* *VSP_576P_200HZ,* *VSP_576I_200HZ,* *VSP_480P_240HZ,* *VSP_480I_240HZ,* *VSP_4K2K_30HZ,* *VSP_4K2K_25HZ,* *VSP_4K2K_24HZ,* *VSP_4K2K_SMPTE* *} VSP_VIDEO_MODE*

Fields

ADV8002	*ADV8003 and ADV8005*
VSP_576I_50HZ = 0, *VSP_576P_50HZ,* *VSP_720P_50HZ,* *VSP_1080I_50HZ,* *VSP_1080P_50HZ,* *VSP_VGA_60HZ,* *VSP_480I_60HZ,* *VSP_480P_60HZ,* *VSP_720P_60HZ,* *VSP_1080I_60HZ,* *VSP_1080P_60HZ,* *VSP_1080P_24HZ*	*VSP_576I_50HZ = 0,* *VSP_576P_50HZ,* *VSP_720P_50HZ,* *VSP_1080I_50HZ,* *VSP_1080P_50HZ,* *VSP_VGA_60HZ,* *VSP_480I_60HZ,* *VSP_480P_60HZ,* *VSP_720P_60HZ,* *VSP_1080I_60HZ,* *VSP_1080P_60HZ,* *VSP_1080P_24HZ,* *VSP_1080P_25HZ,* *VSP_1080P_30HZ,* *VSP_SVGA_60HZ,* *VSP_XGA_60HZ,* *VSP_WXGA_60HZ,* *VSP_SXGA_60HZ,* *VSP_WXGA2_60HZ,* *VSP_UXGA_60HZ,* *VSP_WXGA3_60HZ,* *VSP_WUXGA_60HZ,* *VSP_240P_60HZ,* *VSP_288P_50HZ,* *VSP_1080IE_50HZ,* *VSP_1080I_100HZ,* *VSP_720P_100HZ,* *VSP_576P_100HZ,* *VSP_576I_100HZ,* *VSP_1080I_120HZ,*

	VSP_720P_120HZ, ***VSP_480P_120HZ,*** ***VSP_480I_120HZ,*** ***VSP_576P_200HZ,*** ***VSP_576I_200HZ,*** ***VSP_480P_240HZ,*** ***VSP_480I_240HZ,*** ***VSP_4K2K_30HZ,*** ***VSP_4K2K_25HZ,*** ***VSP_4K2K_24HZ,*** ***VSP_4K2K_SMPTE***

Remarks

None

5.3 *Standard Definition Processor Structures*

5.3.1 VSP_TIMING_FORMAT

Description

A list of the elements that define the parameters of supported video formats.

Synopsis

```
#include "vsp_lib.h"

typedef struct {
        UINT16  HFrontPorch;
        UINT16          HSyncDur;
        UINT16 HBackPorch;
        UINT16 VFrontPorch;
        UINT16 VSyncDur;
        UINT16 VBackPorch;
        UINT16 HActive;
        UINT16 VActive;
        BOOL  IsIntl;
        UCHAR FrameRate;
        UINT16 VideoID;
        UCHAR HPol;
        UCHAR VPol;
} VSP_TIMING_FORMAT, *pVSP_TIMING_FORMAT;
```

Fields

UINT16 HFrontPorch;
Hsync front porch

UINT16 HSyncDur;
Hsync duration

UINT16 HBackPorch;
Hsync back porch

UINT16 VFrontPorch;
Vsync front porch

UINT16 VSyncDur;
Vsync duration

UINT16 VBackPorch;
Vsync back porch

UINT16 HActive;
Active pixels per line

UINT16 VActive;
Active lines per frame

BOOL IsIntl;
Interlaced or progressive

UCHAR FrameRate;
Frame rate

UCHAR VideoID;
Video ID defined in CEA-861

UCHAR HPol;
Hsync polarity

UCHAR VPol;
Vsync polarity

Remarks
None

6 Notification events

The VSP library provide the option to notify the application of any changes in operating condition, thus relieving the application from polling the library for status changes. If the application wishes to be notified on status changes, it must define a function that will be called from the library (from inside ADIAPI_VspIsr) upon any change in operating conditions. The prototype for this function is as follows:

UINT16 App_VspNotificationEvent *(*ADI_VSP_EVENT EventID, UINT16 EventSize, void *EventData)

Where:

EventID

Is a unique value that identifies the operating condition that has changed.

EventSize

This parameter either specifies the size of the data pointed to by the EventData parameter or carries some other information related to the event. This value is event dependant as listed in the table below.

EventData

Is a pointer to a buffer that contains event-specific information. If the event does not have an associated data, this value will be undefined. The table below lists all events and their associated data.

Interrupt	EventID	EventSize	EventData

The return value of the notification event function is not currently used and should always be set to 0.