ANALOG DEVICES

ADV8005 Hardware Reference Manual

UG-707

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

ADV8005 Functionality and Features

Advantiv® Advanced Television Solutions by Analog Devices | NatureVue™ Video Upconversion and Enhancement by Analog Devices

TABLE OF CONTENTS

REVISION HISTORY

1/15—Rev. 0 to Rev. A

Changes to Spare Packets and VSI Support Section 201
Added VSYNC Interrupt Section and Table 53; Renumbered Sequentially 205

6/14—Revision 0: Initial Version

UNDERSTANDING THE ADV8005 HARDWARE MANUAL

DESCRIPTION OF THE HARDWARE MANUAL

This manual provides a detailed description of the functionality and features supported by the ADV8005.

DISCLAIMER

The information contained in this document is proprietary of Analog Devices Inc. (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 Analog Devices.

TRADEMARK AND SERVICE MARK NOTICE

The Analog Devices logo is a registered trademark of Analog Devices, Inc. All other brand and product names are trademarks or service marks of their respective owners.

NUMBER NOTATIONS

Notation	Description
bit N	Bits are numbered in little endian format, that is, the least significant bit of a number is referred to as Bit 0
V[X:Y]	Bit field representation covering bit X to Y of a value or a field V
0xNN	Hexadecimal (base-16) numbers are preceded by the prefix '0x'
0bNN	Binary (base-2) numbers are preceded by the prefix '0b'
NN	Decimal (base-10) are represented using no additional prefixes or suffixes

REGISTER ACCESS CONVENTIONS

Mode	Description
R/W	Memory location has read and write access.
R	Memory location has read access only. A read always returns 0 unless specified otherwise.
W	Memory location has write access only.

ACRONYMS AND ABBREVIATIONS

This is a list of common acronyms and abbreviations found in Analog Devices Hardware Manuals.

Acronym/Abbreviation	Description
ACP	Audio Content Protection
ACR	Audio Clock Regeneration
ADC	Analog to Digital Converter
AFE	Analog Front End
AGC	Automatic Gain Control
Ainfo	HDCP register. Refer to HDCP documentation.
AKSV	HDCP Transmitter Key Selection Vector. Refer to HDCP documentation.
An	64-bit pseudo-random value generated by HDCP cipher function of device A
ARC	Audio Return Channel
AUD_IN	Audio Input Pin
AVI	Auxiliary Video Information
Aux	Auxiliary
Bcaps	HDCP register. Refer to HDCP documentation.

Acronym/Abbreviation	Description
BGA	Ball Grid Array
BKSV	HDCP Receiver Key Selection Vector. Refer to HDCP documentation.
BNR	Block Noise Reduction
CEC	Consumer Electronics Control
CP	Component Processor
CSC	Color Space Converter/Conversion
CSync	Composite Synchronization
CTS	Cycle Time Stamp
CUE	Color Upsampling Error
CVBS	Composite Video
DCM	Decimation
DDR	Double Data Rate
DDFS	Direct Digital Frequency Synthesizer
DE	Data Enable
DID	Data Identification Word
DLL	Delay Locked Loop
DMA	Direct Memory Access
DNR	Digital Noise Reduction
DPP	Data Preprocessor
DSD	Direct Stream Digital
DST	Direct Stream Transfer
DUT	Device Under Test (designate the ADV8005 unless stated otherwise)
DVD	Digital Video Disc
DVI	Digital Visual Interface
EAV	End of Active Video
ED	Enhanced Definition
ENC	Encoder
EQ	Equalizer
FFS	Field Frame Scheduler
FRC	Frame Rate Conversion/Converter
HBR	High Bit Rate
HD	High Definition
HDCP	High Bandwidth Digital Content Protection
HDMI	High Definition Multimedia Interface
HDTV	High Definition Television
HEAC	HDMI Ethernet and Audio Channels
HEC	HDMI Ethernet Channel
HPA	Hot Plug Assert
HPD	Hot Plug Detect
HSync/HS	Horizontal Synchronization
IC	Integrated Circuit
ISRC	International Standard Recording Code
I^2S	Inter IC Sound
I^2C	Inter Integrated Circuit
KSV	Key Selection Vector
LLC	Line Locked Clock
LQFP	Low-profile Quad Flat Package
LSB	Least Significant Bit
L-PCM	Linear Pulse Code Modulation
Mbps	Megabit per Second
MNR	Mosquito Noise Reduction

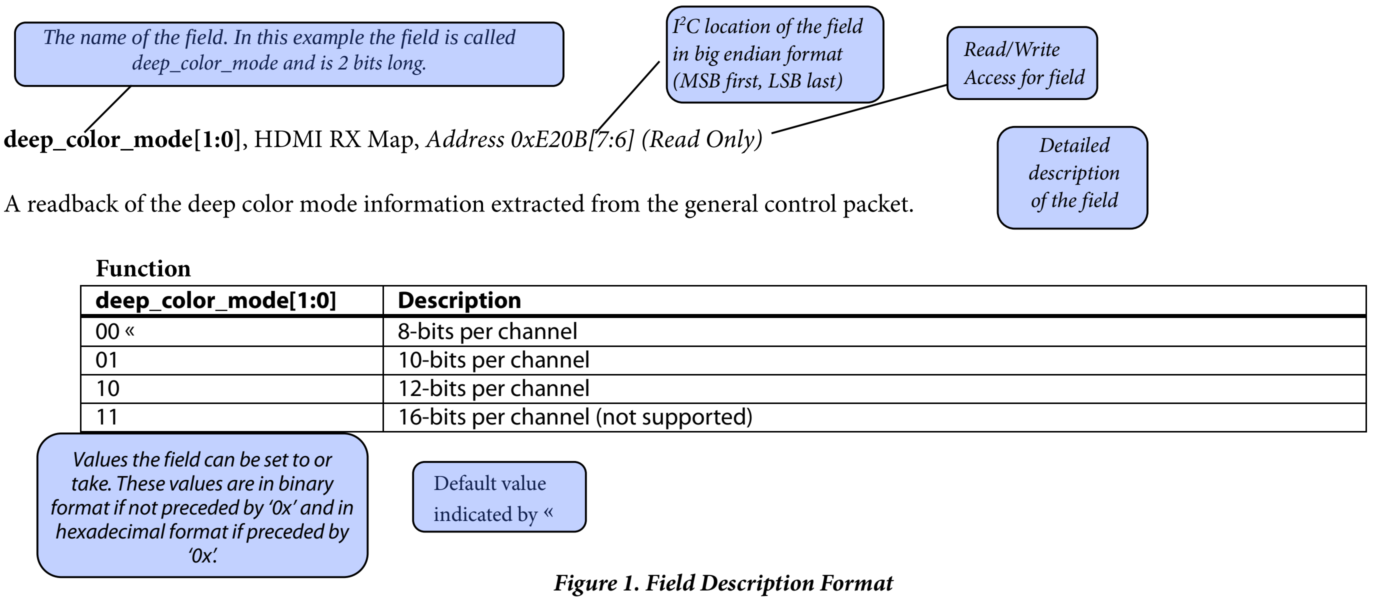
Acronym/Abbreviation	Description
MPEG	Moving Picture Expert Group
ms	Millisecond
MSB	Most Significant Bit
NC	No Connect
NSV	Noise Shaped Video
OSD	On Screen Display
OTP	One Time Programmable
PtoI	Progressive to Interlaced
Pj’	HDCP Enhanced Link Verification Response. Refer to HDCP documentation.
PVSP	Primary VSP
Ri’	HDCP Link verification response. Refer to HDCP documentation.
RNR	Random Noise Reduction
Rx	Receiver
SA	Slave Address
SAV	Start of Active Video
SD	Standard Definition
SDP	Standard Definition Processor
SDR	Single Data Rate
SMPTE	Society of Motion Picture and Television Engineers
SNR	Signal to Noise Ratio
SOG	Sync on Green
SOY	Sync on Y
SPA	Source Physical Address
SPD	Source Production/Product Descriptor
SPDIF	Sony/Philips Digital Interface
SPI	Serial Peripheral Interface
SRM	System Renewability Message
SSPD	Synchronization Source Polarity Detector
STDI	Standard Identification
SVSP	Secondary VSP
TBC	Timebase Correction
TMDS	Transition Minimized Differential Signaling
Tx	Transmitter
ULAI	Ultra Low Angle Interpolation
US	Up Sampling
VBI	Video Blanking Interval
VDP	VBI Data Processor
VIC	Video Identification Code
VIM	Video Input Module
VOM	Video Output Module
VSDP	Vendor Specific Data Block
VSP	Video Signal Processor/Processing
VSync/VS	Vertical Synchronization
XTAL	Crystal Oscillator

FIELD FUNCTION DESCRIPTION

The function of a field is described in a table preceded by the bit name, a short function description, the I^{2}C map, the register location within the I^{2}C map, and a detailed description of the field. Refer to Figure 1 for more details.

The detailed description consists of:

- For a readable field, the values the field can take
- For a writable field, the values the field can be set to


The name of the field. In this example the field is called deep_color_mode and is 2 bits long.
I^{2}C location of the field in big endian format (MSB first, LSB last)
Read/Write Access for field
Detailed description of the field
Values the field can be set to or take. These values are in binary format if not preceded by '0x' and in hexadecimal format if preceded by '0x'.
Default value indicated by «

deep_color_mode[1:0], HDMI RX Map, *Address 0xE20B[7:6] (Read Only)*

A readback of the deep color mode information extracted from the general control packet.

Function

deep_color_mode[1:0]	Description
00 «	8-bits per channel
01	10-bits per channel
10	12-bits per channel
11	16-bits per channel (not supported)

Figure 1. Field Description Format

REFERENCES

HDMI Licensing and LLC, High-Definition Multimedia Interface, Revision 1.4a, March 4, 2010

Digital Content Protection (DCP) LLC, High-bandwidth Digital Content Protection System, Revision 1.3, December 21, 2006

CEA, CEA-861-E, A DTV Profile for Uncompressed High Speed Digital Interfaces, Revision E, September 11, 2007

ITU, ITU-R BT.656-4, Interface for Digital Component Video Signals in 525-Line and 625-Line Television Systems Operating at the 4:2:2 Level of Recommendation ITU-R BT.601, February 1998

ITU, ITU-R BT.601-5 Studio encoding parameters of digital television for standard 4:3 and widescreen 16:9 aspect ratios, December 1995

ITU, ITU-R BT.709-5 Parameter values for the HDTV standards for production and international programme exchange, April 2002

CENELEC, EN 50157, Part 1, Domestic and similar electronic equipment interconnection requirements: AV.link

CENELEC, EN 50157, Part 2-1, Domestic and similar electronic equipment interconnection requirements: AV.link

CENELEC, EN 50157, Part 2-2, Domestic and similar electronic equipment interconnection requirements: AV.link

CENELEC, EN 50157, Part 2-3, Domestic and similar electronic equipment interconnection requirements: AV.link

1. INTRODUCTION TO THE ADV8005

1.1. OVERVIEW

The ADV8005 is a video signal processor (VSP) with TTL and Serial Video inputs that can de-interlace and scale input video, generate and blend a bitmap based on-screen display (OSD) and output the blended video using one or more of the part's outputs; dual HDMI transmitters and a 6-DAC encoder with SD and HD support.

The ADV8005 has three video inputs – the video TTL input, the EXOSD TTL input and the Serial Video receiver (Rx). The combined video TTL input and EXOSD TTL input constitute the flexible 60-bit TTL input port. The 60-bit TTL input port can be arranged in a variety of fashions to accept one input video stream (for example, a 48-bit 3 GHz input video stream from ADV7619) or two input video streams (for example, a 36-bit input video stream from ADV7844 and a 24-bit input video stream from an external OSD generator). Once the data is received, the video TTL and EXOSD TTL inputs can be connected to either the primary input channel or the secondary input channel. From these input channels, the video data can be sent to the internal video processing blocks (for example, primary VSP or secondary VSP).

The Serial Video Rx is connected to the RX input channel. The Serial Video Rx accommodates inter-chip transfer of data over an HDMI compatible interface (for example, from an HDMI (Rx) such as ADV7850 or transceiver such as ADV7623). The ADV8005 does not support EDID or DDC activity on this port.

The motion adaptive de-interlacer in the ADV8005 provides excellent edge detection and excellent ultra-low angle performance. Per-pixel motion-adaptive de-interlacing is used for natively interlaced input video (for example, a live sport broadcast) where still parts of the image are reconstructed from information on both the odd and even fields, and moving parts of the image are interpolated by an advanced interpolation algorithm. The de-interlacer can also recognize when interlaced input video originally came from progressive content (for example, 24 Hz movie content or 30 Hz documentary content) and reconstructs the original frames.

Dual video scalers allow the ADV8005 to support two different output resolutions on its outputs, for example, 1080p60 on HDMI Tx1 and 720p on HDMI Tx2 and the HD encoder. The primary VSP (PVSP) in the ADV8005 is capable of upscaling from 480i to 4k x 2k formats. The secondary VSP (SVSP) in the ADV8005 is used to provide a second output resolution to accommodate dual zone systems. The ADV8005 is also capable of downscaling a single 4Kx2K input to 1080p or lower using a combination of a Horizontal Pre-Scaler (HPS) and the SVSP. Also available in the ADV8005 are image enhancing features such as random noise reduction (RNR), mosquito noise reduction (MNR) and block noise reduction (BNR), detail enhancement and automatic contrast enhancement (ACE).

The ADV8005 features an internal bitmap based OSD generator capable of generating OSDs of up to 4k x 2k. External solutions can also be implemented and fed into the ADV8005 for blending with the main video. A bitmap based OSD is an advanced form of OSD display, which can add effects such as scrolling, animation and 3D depth to OSD displays. This allows customers to create advanced OSD designs to differentiate their products. Once created, OSD designs are stored in an external SPI flash memory connected to the ADV8005. The control of the OSD must be performed from the system microcontroller via SPI. OSD designs can be created using ADI's software development tool, *Blimp OSD*.

The ADV8005 offers flexible configuration of its internal circuitry allowing the output of one, two or three input channels simultaneously. The output of multiple ADV8005s can be synchronized using the master clock, horizontal sync and vertical sync inputs. This facilitates, with the incorporation of a simple FPGA, seamless per pixel switching of multiple synchronized ADV8005 inputs. The ADV8005 can also measure the picture position and sample quality of the video being processed; this assists in identifying the exact video format and the optimum sampling phase of the video front end's ADC.

Video can be output from the ADV8005 via one or both of the HDMI transmitters, the 6-DAC SD/HD video encoder or using the TTL interface. Both HDMI transmitters support the HDMI v1.4b specifications of increased resolutions, 3D video and audio return channel (ARC). The ADV8005 supports both S/PDIF and 8-channel I^2S audio. . The audio can be sourced from either the external audio interface or using the audio pass through feature of the Serial Video Rx.

The ADV8005 includes a high-speed digital-to-analog video encoder available with and without Rovi content protection. Six high speed, Noise Shaped Video (NSV), 12-bit video DACs provide support for composite (CVBS), S-video (Y/C), and component (YPrPb/RGB) analog outputs in either SD, ED, or HD video formats up to 1080p. In addition, simultaneous SD and ED/HD formats are supported. 216 MHz (SD and ED) and 297 MHz (HD) oversampling ensures that external output filtering is not required. The final option to output video from the ADV8005 is the TTL interface which allows up to 36 of the pins to be reconfigured as outputs. This facilitates the output of up to 1080p 12-bit deep color from the ADV8005 to an FPGA without the requirement of an expensive FPGA-based HDMI phy.

The ADV8005 supports all common consumer formats as outlined in the EIA-861 specification and many common professional output formats as outlined in the VESA specification.

The part supports the I2C® and SPI protocols for communication with the system microcontroller.

Note: There are four options within the ADV8005 family of parts, each with different capabilities but all in the same CSPBGA-425 package. These are described in Table 1.

Table 1: Available Features Within ADV8005 Family of ICs

Part Number	Max Speed	Maximum Resolution	HDMI Tx Outputs	Analog Outputs	Rovi	VSP	OSD	TTL Out
ADV8005KBCZ-8A	3 Gbps	4k × 2k at 30 Hz (8-bit)	2	Six 12-bit DACs	Yes	2	Yes	Yes
ADV8005KBCZ-8N	3 Gbps	4k × 2k at 30 Hz (8-bit)	2	Six 12-bit DACs	No	2	Yes	Yes
ADV8005KBCZ-8B	3 Gbps	4k × 2k at 30 Hz (8-bit)	1	No	N/A	1	Yes	No
ADV8005KBCZ-8C	3 Gbps	4k × 2k at 30 Hz (8-bit)	2	No	N/A	2	Yes	No

Note that ADV8005KBCZ-8A and ADV8005KBCZ-8N functionality is described throughout this manual (figures, functional blocks, and so on). Some sections of this manual are not relevant to the ADV8005KBCZ-8B and ADV8005KBCZ-8C as they do not include those blocks. If a section is not relevant to a particular generic, this is indicated in the introduction to that section.

1.1.1. *Digital Video Input*

Video data can be input into the ADV8005 in a number of ways. The flexible 60-bit TTL input port can be configured for dual video inputs (video TTL input and EXOSD TTL input), for a single video input (interleaved TTL data from an ADV7619) or for a single video input and an external alpha channel. The 60-bit TTL input port is extremely flexible and can be configured into a number of different arrangements; for more information, refer to Table 88 and Table 89. Video can also be input into the ADV8005 via the Serial Video Rx which can be used for device to device interconnect, for example, a serial video link between the ADV7850 and the ADV8005 or a serial video link between the ADV7623 and the ADV8005. Using such front end devices located before the ADV8005 allows the audio to be extracted and processed in a DSP before being reinserted into the ADV8005.

A mux after the TTL inputs allows the video TTL input pins and the EXOSD TTL input pins to be connected to either the primary or the secondary input channel. The primary input channel features an input formatter, manually programmable CSC, updither function, ACE, contrast, brightness and saturation controls. The secondary input channel features an input formatter, manually programmable CSC and updither function. The Serial Video Rx is connected directly to the Rx input channel and features an input formatter, manually programmable CSC and updither function.

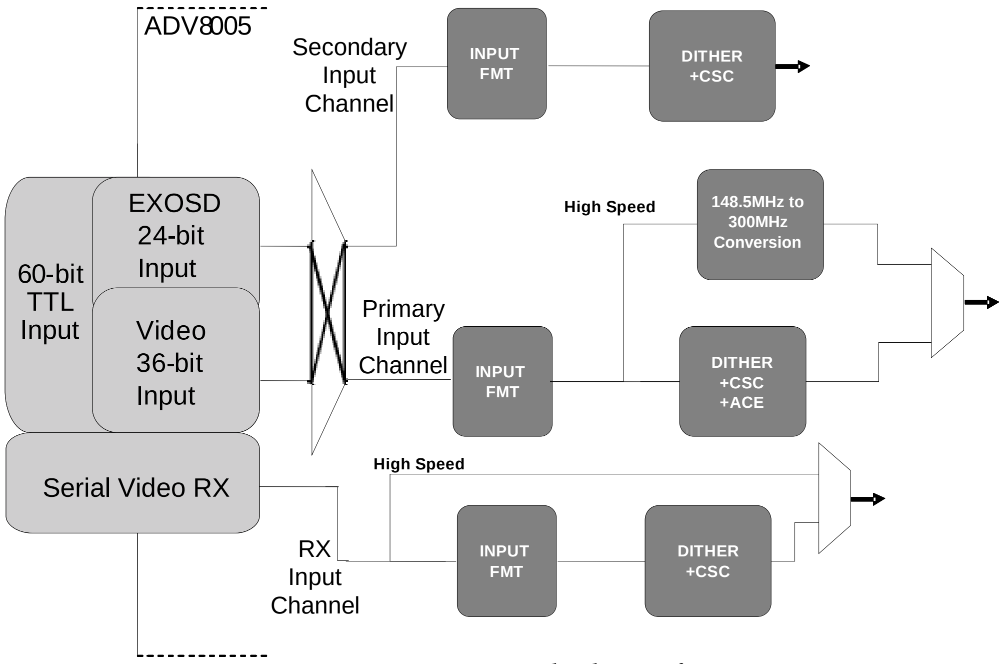


Figure 2: ADV8005 Digital Video Interface

1.1.2. *Flexible Digital Core*

The ADV8005 has a flexible digital core, allowing multiple options for the routing of video data. This allows the user to place the OSD in front of the video processing so the OSD will be overlaid on one or more outputs. Alternatively, video processing can be placed before the OSD ensuring all outputs are processed to the highest quality. The digital core can also be configured so that the ADV8005 can output one or all of the inputs in various arrangements, for example, picture in picture with one input appearing as a window within another or two inputs routed to two outputs.

Several common modes of operation are defined to assist the user to quickly integrate the ADV8005 into a system. Refer to Section 2 for more details.

1.1.3. *Video Signal Processor*

The motion adaptive de-interlacer in the ADV8005 offers excellent edge detection and ultra low angle performance. The per-pixel de-interlacing algorithm used delivers excellent performance which can be seen with specialist test patterns, on facial features like eyebrows or on shirt collars. This algorithm decides on whether an area of an image is moving or not and then applies the appropriate de-interlacing approach accordingly. The de-interlacer can also determine when interlaced video originated as progressive and can reconstruct the original frames.

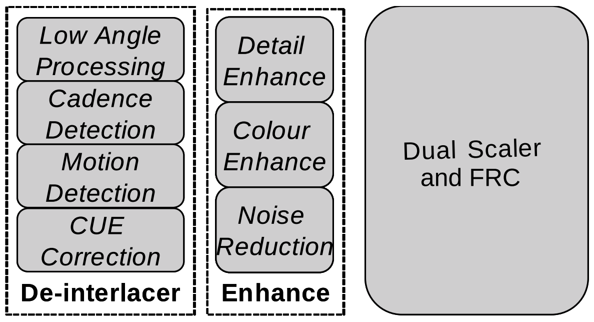


Figure 3: ADV8005 Video Processing

The ADV80038005 features dual scalers referred to as Primary Video Signal Processor (PVSP) and the Secondary Video Signal Processor (SVSP). The PVSP uses a contour-based interpolation scaler which can upscale from 480i to 4k x 2k. The PVSP can arbitrarily upscale between 480p and 4k x 2k and down scale between 1080p and 480p. The advanced scaling algorithm used in the ADV8005 eradicates many common problems associated with scaling video such as ringing and jagged or blurred edges. Using a combination of the Horizontal Pre Scaler (HPS) and the SVSP, the ADV8005 can downscale from 4k x 2k to 1080p. The PVSP can be employed to further scale the downscaled 4k x 2k content. When using the PVSP as the primary scaler, the SVSP can also be used to provide a second lower resolution output format. The PVSP and SVSP can be connected in parallel or in series.

The ADV8005 features a number of video enhancement controls such as detail enhancement, block noise reduction, mosquito noise reduction and random noise reduction. Block and mosquito noise are related to the compression of video for transmission or encoding onto a DVD or BD disc. Random noise is related to noise picked up during the transmission of video. The automatic contrast enhancement feature offered by the ADV8005 intelligently stretches the brightness of an image to enhance the dark areas without saturating the dark areas.

Note that the dual scaler variants of the ADV8005 are the following:

- ADV8005KBCZ-8A/8N
- ADV8005KBCZ-8C

The single scaler variants of the ADV8005 are the following:

- ADV8005KBCZ-8B

1.1.4. *Bitmap On Screen Display*

The ADV8005 incorporates an OSD core capable of generating an internal bitmap based OSD. Customers can generate elaborate OSD designs that can include bitmap images, 3D overlay and animation. Up to 256 regions in total can be created and displayed. These 256 regions are bitmap images defined during the design stage and can be characters, pictures, buttons, and so on. Individual regions can be alpha blended and prioritized versus other regions.

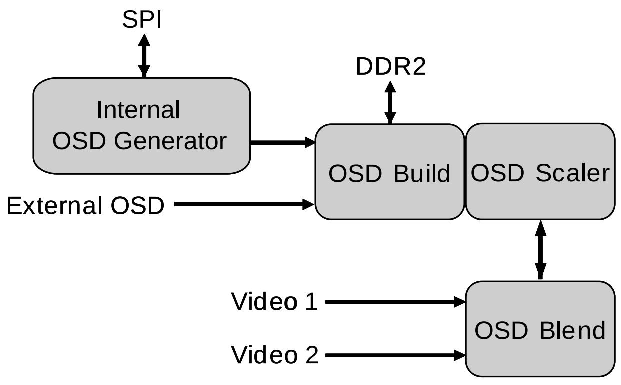


Figure 4: ADV8005 Bitmap OSD

The OSD is controlled by the host microcontroller via the ADV8005 SPI slave (serial port 1). In response to commands, the ADV8005 loads the data from the external SPI flash memory via the SPI master (serial port 2). The ADV8005 uses DDR2 memory when rendering and blending the OSD. In order to lower the load of the DDR2 memory, there is a block in the ADV8005 OSD hardware called the OSD co-processor. The OSD co-processor is responsible for handling upper level commands from the microcontroller and translating them into lower level operations for the OSD and DMA which retrieves data from the external DDR2 memories.

The OSD blend can be switched between either of the two video streams routed through the OSD blend block without disturbing the output video. This enables seamless OSD blending in dual zone systems.

Bitmap OSDs can be created and compiled using ADI's software development tool, *Blimp OSD*. This allows users to create their custom OSDs and emulate them before integrating them into their system, abstracting the design task from the underlying OSD hardware. For more details on the operation of the external OSD, design and system techniques, refer to the *Blimp OSD* documentation.

1.1.5. *External DDR2 Memory*

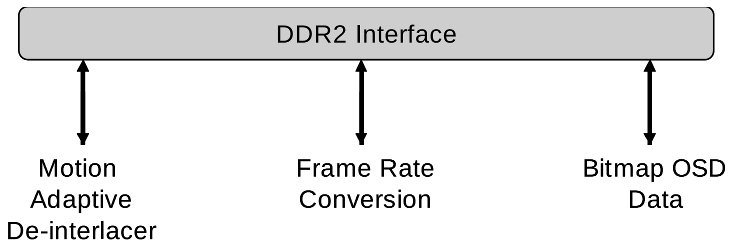


Figure 5: External DDR2 Memory Interface

External DDR2 memory is required for motion adaptive de-interlacing, Frame Rate Conversion (FRC), and OSD bitmap overlay. ADV8005 supports various memory options using one or two DDR2 memories of various sizes (1 Gb maximum). For full processing capabilities, two DDR2 memories are required which use data transfers up to 250 MHz. Refer to Section 3 for more details on the operations using the external DDR2 memory.

1.1.6. *HDMI Transmitter*

The ADV8005 features two HDMI v1.4b transmitters. The transmitters feature an audio return channel (ARC), which allows a Sony/Philips Digital Interface (SPDIF) audio connection between the source and sink. Each transmitter features an on-chip MPU with an I^{2}C master to perform HDCP operations and EDID operations.

Note: The dual transmitter variants of the ADV8005 are ADV8005KBCZ-8A, ADV8005KBCZ-8N and ADV8005KBCZ-8C. The single transmitter variant of the ADV8005 is the ADV8005KBCZ-8B.

1.1.7. *Video Encoder*

The ADV8005 features a high speed digital to analog video encoder. Six high speed, NSV, 3.3 V, 12-bit video DACs provide support for worldwide composite (CVBS), S-Video (Y-C), and component (YPrPb/RGB) analog outputs in standard definition (SD), enhanced definition (ED), or high definition (HD) video formats. It is also possible to enable the ADV8005 video encoder to work in simultaneous mode where both an SD and ED/HD format are being output.

Note: The video encoder variants of the ADV8005 are the ADV8005KBCZ-8A and the ADV8005KBCZ-8N. The variants of ADV8005 with no encoder are the ADV8005KBCZ-8B and the ADV8005KBCZ-8B.

1.1.8. *Digital Video Output*

Video can be output from the ADV8005 via the flexible TTL port. Reusing up to 36 of the flexible TTL port pins means that video can be routed in and out of the ADV8005 without using HDMI, a useful cost reduction in systems which utilize FPGA interconnects (e.g. 30-bit TTL input and 30-bit TTL output allowing 1080p 10-bit input and output). The possible configurations of the TTL output port are captured in Table 91 and Table 92. The video TTL output port has a manually programmable CSC.

1.2. MAIN FEATURES OF THE ADV8005

1.2.1. *Video Signal Processor*

1.2.1.1. Primary VSP

- 12-bit internal processing
- Fixed frame latency capability
- Input timing up to 1080p
- Output timing up to 4k x 2k
- Input/output format YCbCr at 4:4:4
- Motion adaptive de-interlacing with motion detection
- Ultra low angle interpolation on edge regions of interlaced video
- Cadence detection (any cadence detection possible)
- Progressive cadence supported
- Super resolution video scaler
- Aspect ratio conversion/panorama scaling
- Arbitrary upscaling and downscaling for both horizontal and vertical direction
- Sharpness detail and edge enhancement
- Noise reduction for random, mosquito, and block noise
- Frame Rate Conversion
- Color Upsampling Error (CUE) correction
- Progressive to interlaced (PtoI) converter
- Game mode supported
- Album mode supported
- Demo window

1.2.1.2. Horizontal Pre-Scaler

- 8-bit internal processing
- Downscales video standards of greater than 162MHz and/or more than 2048 pixels/line

1.2.1.3. Secondary VSP

- 8-bit internal processing
- Input and output timing up to 1080p
- Input and output format YCbCr at 4:4:4
- Up-scaling and down-scaling for both horizontal and vertical direction
- Aspect ratio conversion and panorama scaling
- Frame Rate Conversion
- Progressive to interlaced (PtoI) converter

1.2.2. *OSD*

- Internally generated bitmap based OSD allowing overlay of bitmap images on one or more video outputs
- Dual video paths through the OSD blend block to support dual zone OSD
- Dedicated OSD scaler – allows OSDs to be rendered at a single resolution reducing external memory bandwidth
- Blending onto 3 GHz video formats
- Pixel-by-pixel alpha blending of OSD data on video data
- Option of externally generated OSD
- OSD can be overlaid in the main 3D video format timings
- *Blimp OSD* software tool and provided ANSI-C libraries cover the full design flow of any OSD

1.2.3. *Video Encoder*

- Six NSV 12-bit video DACs
- Compliant with all common SMPTE formats
- Multiformat video output support
  - NTSC M, PAL B/D/G/H/I/M/N, PAL 60 support
  - Composite (CVBS) and S-Video (Y/C) component/YPrPb/RGB (SD, ED and HD)
- Macrovision® Rev 7.1.L1 (SD) and Rev 1.2 (ED) compliant
- Simultaneous SD and ED/HD operation

1.2.4. *HDMI 1.4 Transmitter*

- 3 GHz video output (ADV8005KBCZ-8A/8N models only)
- Incorporates HDMI™ (v.1.4 with Deep Color, x.v.Color™)
  - Content Type Bits
  - ARC (Audio Return Channel) Support
  - 3D support
- Supports standard S/PDIF for stereo LPCM compressed audio up to 192 kHz
- Six-channel uncompressed LPCM I2S audio up to 192 kHz
- Six-channel DSD audio inputs

1.2.5. *Additional Features*

- Auto-phase and position detection
- External Sync Timing mode employing Master clock, horizontal sync and vertical sync inputs

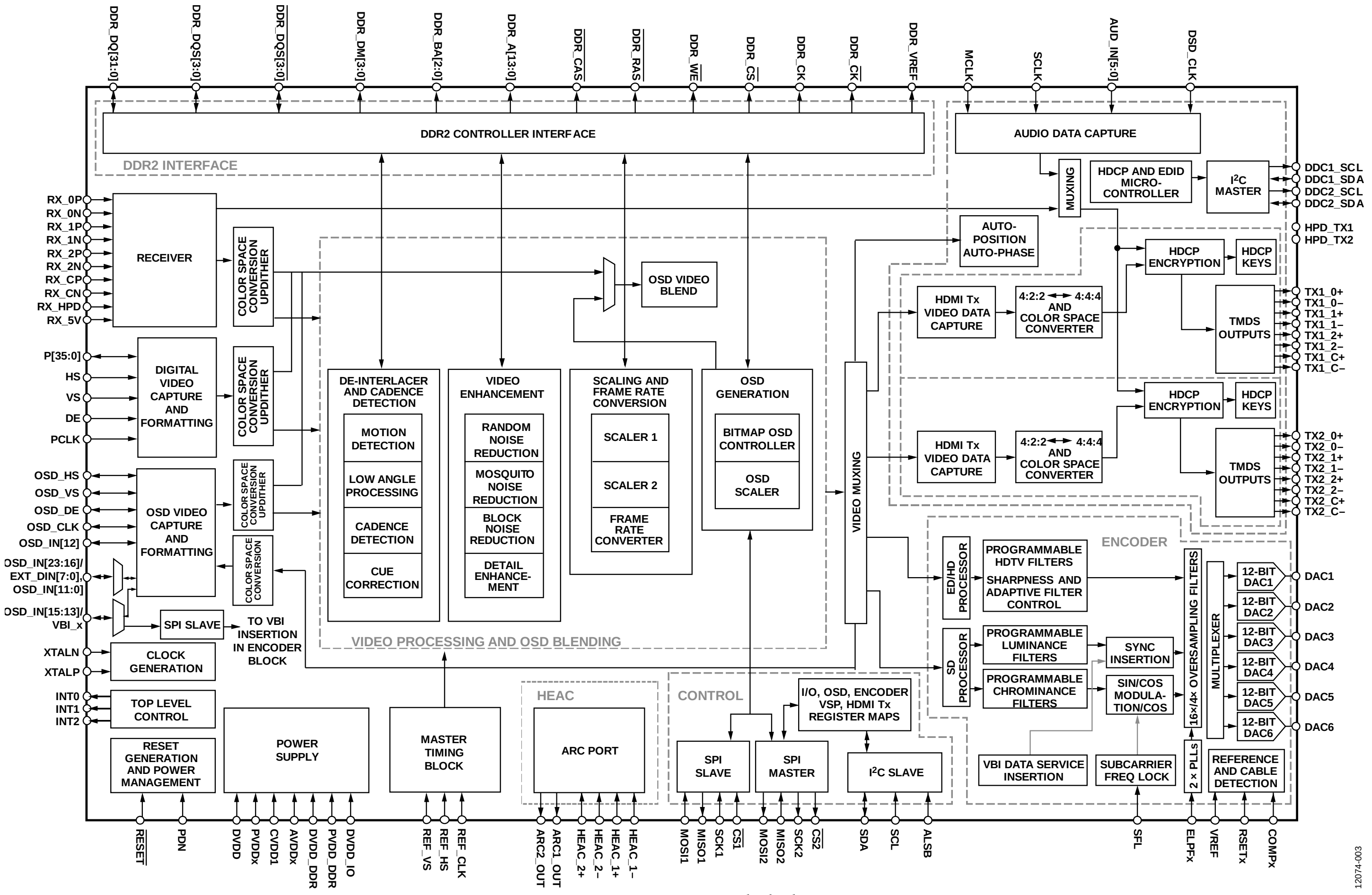

DDR2 CONTROLLER INTERFACE
DDR2 INTERFACE
AUDIO DATA CAPTURE
RECEIVER
DIGITAL VIDEO CAPTURE AND FORMATTING
OSD VIDEO CAPTURE AND FORMATTING
COLOR SPACE CONVERSION UPDITHER
OSD VIDEO BLEND
DE-INTERLACER AND CADENCE DETECTION
MOTION DETECTION
LOW ANGLE PROCESSING
CADENCE DETECTION
CUE CORRECTION
VIDEO ENHANCEMENT
RANDOM NOISE REDUCTION
MOSQUITO NOISE REDUCTION
BLOCK NOISE REDUCTION
DETAIL ENHANCEMENT
SCALING AND FRAME RATE CONVERSION
SCALER 1
SCALER 2
FRAME RATE CONVERTER
OSD GENERATION
BITMAP OSD CONTROLLER
OSD SCALER
VIDEO PROCESSING AND OSD BLENDING
VIDEO MUXING
MUXING
HDCP AND EDID MICRO-CONTROLLER
I2C MASTER
AUTO-POSITION AUTO-PHASE
HDMI Tx VIDEO DATA CAPTURE
4:2:2 ↔ 4:4:4 AND COLOR SPACE CONVERTER
HDCP ENCRYPTION
HDCP KEYS
TMDS OUTPUTS
ENCODER
ED/HD PROCESSOR
SD PROCESSOR
PROGRAMMABLE HDTV FILTERS
SHARPNESS AND ADAPTIVE FILTER CONTROL
PROGRAMMABLE LUMINANCE FILTERS
PROGRAMMABLE CHROMINANCE FILTERS
SYNC INSERTION
SIN/COS MODULATION/COS
16×/4× OVERSAMPLING FILTERS
MULTIPLEXER
12-BIT DAC1 … DAC6
2 × PLLs
VBI DATA SERVICE INSERTION
SUBCARRIER FREQ LOCK
REFERENCE AND CABLE DETECTION
SPI SLAVE
TO VBI INSERTION IN ENCODER BLOCK
CLOCK GENERATION
TOP LEVEL CONTROL
RESET GENERATION AND POWER MANAGEMENT
POWER SUPPLY
MASTER TIMING BLOCK
HEAC
ARC PORT
CONTROL
I/O, OSD, ENCODER VSP, HDMI Tx REGISTER MAPS
SPI SLAVE
SPI MASTER
I2C SLAVE
DDR_DQ[31:0] DDR_DQS[3:0] DDR_DQS[3:0] DDR_DM[3:0] DDR_BA[2:0] DDR_A[13:0] DDR_CAS DDR_RAS DDR_WE DDR_CS DDR_CK DDR_CK DDR_VREF MCLK SCLK AUD_IN[5:0] DSD_CLK
RX_0P RX_0N RX_1P RX_1N RX_2P RX_2N RX_CP RX_CN RX_HPD RX_5V
P[35:0] HS VS DE PCLK
OSD_HS OSD_VS OSD_DE OSD_CLK OSD_IN[12]
OSD_IN[23:16]/EXT_DIN[7:0], OSD_IN[11:0]
OSD_IN[15:13]/VBI_x
XTALN XTALP INT0 INT1 INT2
DDC1_SCL DDC1_SDA DDC2_SCL DDC2_SDA HPD_TX1 HPD_TX2
TX1_0+ TX1_0– TX1_1+ TX1_1– TX1_2+ TX1_2– TX1_C+ TX1_C–
TX2_0+ TX2_0– TX2_1+ TX2_1– TX2_2+ TX2_2– TX2_C+ TX2_C–
DAC1 DAC2 DAC3 DAC4 DAC5 DAC6
RESET PDN DVDD PVDDx CVDD1 AVDDx DVDD_DDR PVDD_DDR DVDD_IO REF_VS REF_HS REF_CLK ARC2_OUT ARC1_OUT HEAC_2+ HEAC_2– HEAC_1+ HEAC_1– MOSI1 MISO1 SCK1 CS1 MOSI2 MISO2 SCK2 CS2 SDA SCL ALSB SFL ELPFx VREF RSETx COMPx
12074-003

Figure 6. ADV8005 Functional Block Diagram

1.3. PROTOCOL FOR MAIN I^{2}C PORT

The system controller initiates a data transfer by establishing a start condition, defined by a high to low transition on SDA while SCL remains high. This transition indicates that an address/data stream will follow. All peripherals respond to the start condition and shift the next eight bits (7-bit address and R/W bit). The bits are transferred from MSB down to LSB. The peripheral that recognizes the transmitted address responds by pulling the data line low during the ninth clock pulse. This is known as an acknowledge bit. All other devices withdraw from the bus at this point and maintain an idle condition.

In the idle condition, the device monitors the SDA and SCL lines for the start condition and the correct transmitted address. The R/W bit determines the direction of the data. A logic 0 on the LSB of the first byte means that the master will write information to the peripheral. A logic 1 on the LSB of the first byte means that the master will read information from the peripheral.

The ADV8005 has a single 8-bit I^{2}C slave address. All register maps within the ADV8005 can be accessed through this I^{2}C address through 16-bit addressing and 8-bit data registers. The ADV8005 acts as a standard slave device on the I^{2}C bus. It interprets the first byte as the I^{2}C address and the second byte and third bytes as the appropriate subaddress. The fourth byte is then considered the data for this subaddress register. This means that I^{2}C writes to the part will be in the form <I^{2}C Address>, <Address MSBs>, <Address LSBs>, <Data>.

For example, to write 0xFF to the encoder register map, register 0x59AF, the I^{2}C writes needed are 0x1A, 0x59, 0xAF, 0xFF. The addresses are outlined in Table 2. Figure 7 shows the register map architecture for the ADV8005.

Table 2: ADV8005 I^{2}C Address and Register Address Range for Different HW Blocks

Register Map Name	I^{2}C Address	Register Address
IO Map	0x1A (0x18 with LSB low)	0x1A00 to 0x1BFF
Primary VSP Map		0xE800 to 0xE8FF
Primary VSP Map 2		0xE900 to 0xE9FF
Secondary VSP Map		0xE600 to 0xE6FF
Rx Main Map		0xE200 to 0xE2FF
Rx InfoFrame Map		0xE300 to 0xE3FF
Tx1 Main Map		0xEC00 to 0xECFF
Tx1 EDID Map		0xEE00 to 0xEEFF
Tx1 UDP Map		0xF200 to 0xF2FF
Tx1 Test Map		0xF300 to 0xF3FF
Tx2 Main Map		0xF400 to 0xF4FF
Tx2 EDID Map		0xF600 to 0xF6FF
Tx2 UDP Map		0xFA00 to 0xFAFF
Tx2 Test Map		0xFB00 to 0xFBFF
Encoder Map		0xE400 to 0xE4FF
DPLL Map		0xE000 to 0xE0FF

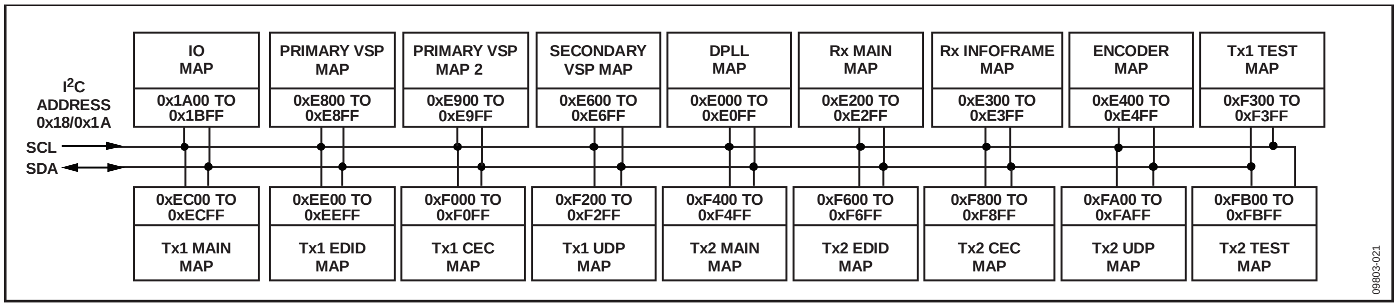

I²C ADDRESS 0x18/0x1A
SCL
SDA
IO MAP 0x1A00 TO 0x1BFF
PRIMARY VSP MAP 0xE800 TO 0xE8FF
PRIMARY VSP MAP 2 0xE900 TO 0xE9FF
SECONDARY VSP MAP 0xE600 TO 0xE6FF
DPLL MAP 0xE000 TO 0xE0FF
Rx MAIN MAP 0xE200 TO 0xE2FF
Rx INFOFRAME MAP 0xE300 TO 0xE3FF
ENCODER MAP 0xE400 TO 0xE4FF
Tx1 TEST MAP 0xF300 TO 0xF3FF
0xEC00 TO 0xECFF Tx1 MAIN MAP
0xEE00 TO 0xEEFF Tx1 EDID MAP
0xF000 TO 0xF0FF Tx1 CEC MAP
0xF200 TO 0xF2FF Tx1 UDP MAP
0xF400 TO 0xF4FF Tx2 MAIN MAP
0xF600 TO 0xF6FF Tx2 EDID MAP
0xF800 TO 0xF8FF Tx2 CEC MAP
0xFA00 TO 0xFAFF Tx2 UDP MAP
0xFB00 TO 0xFBFF Tx2 TEST MAP
09803-021

Figure 7: Register Map Architecture

It is possible to use the subaddresses auto-increment feature, which allows data to be accessed from the starting subaddress. A data transfer is always terminated by a stop condition. The user can also access any unique subaddress register on a one-by-one basis without having to update all the registers.

Stop and start conditions can be detected at any stage during the data transfer. If these conditions are asserted out of sequence with normal read and write operations, these cause an immediate jump to the idle condition. During a given SCLK high period, the user should issue only one start condition, one stop condition, or a single stop condition followed by a single start condition. If an invalid subaddress is issued by the user, the ADV8005 does not issue an acknowledge and returns to the idle condition.

If the user exceeds the highest subaddress in auto increment mode, the following actions are taken:

- In read mode, the highest subaddress register contents continue to be output until the master device issues a no acknowledge. This indicates the end of a read. A no acknowledge condition is where the SDA line is not pulled low on the ninth pulse.
- In write mode, the data for the invalid byte is not loaded into any subaddress register. A no acknowledge is issued by the ADV8005 and the part returns to the idle condition.

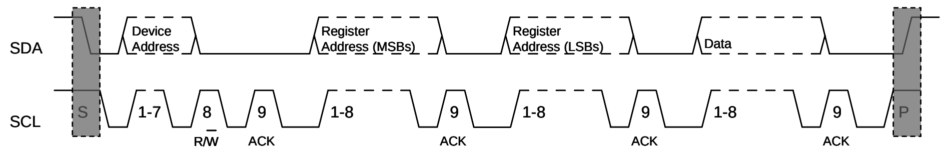


Figure 8: Bus Data Transfer

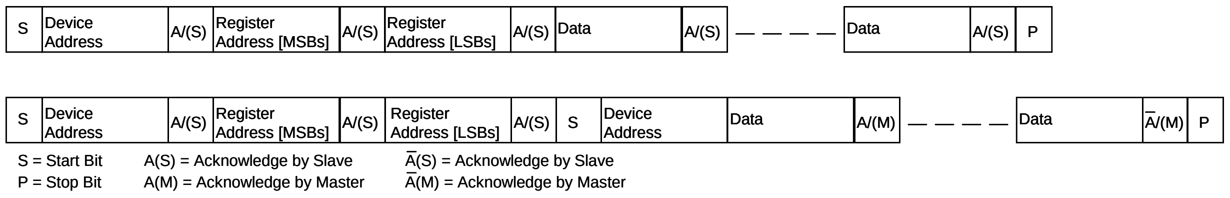

S | Device Address | A/(S) | Register Address [MSBs] | A/(S) | Register Address [LSBs] | A/(S) | Data | A/(S) | — — — — | Data | A/(S) | P
S | Device Address | A/(S) | Register Address [MSBs] | A/(S) | Register Address [LSBs] | A/(S) | S | Device Address | Data | A/(M) | — — — — | Data | Ā/(M) | P
S = Start Bit
P = Stop Bit
A(S) = Acknowledge by Slave
A(M) = Acknowledge by Master
Ā(S) = Acknowledge by Slave
Ā(M) = Acknowledge by Master

Figure 9: Read and Write Sequence

1.4. CONFIGURING THE ADV8005

The ADV8005 requires a number of configuration settings for each mode of operation. To ensure the part is correctly configured, refer to either the recommended settings configuration script (supplied with the ADV8005 evaluation software) or the reference software driver. Failure to follow these recommended settings will result in the part not operating to its optimum performance.

2. ADV8005 TOP LEVEL CONTROL

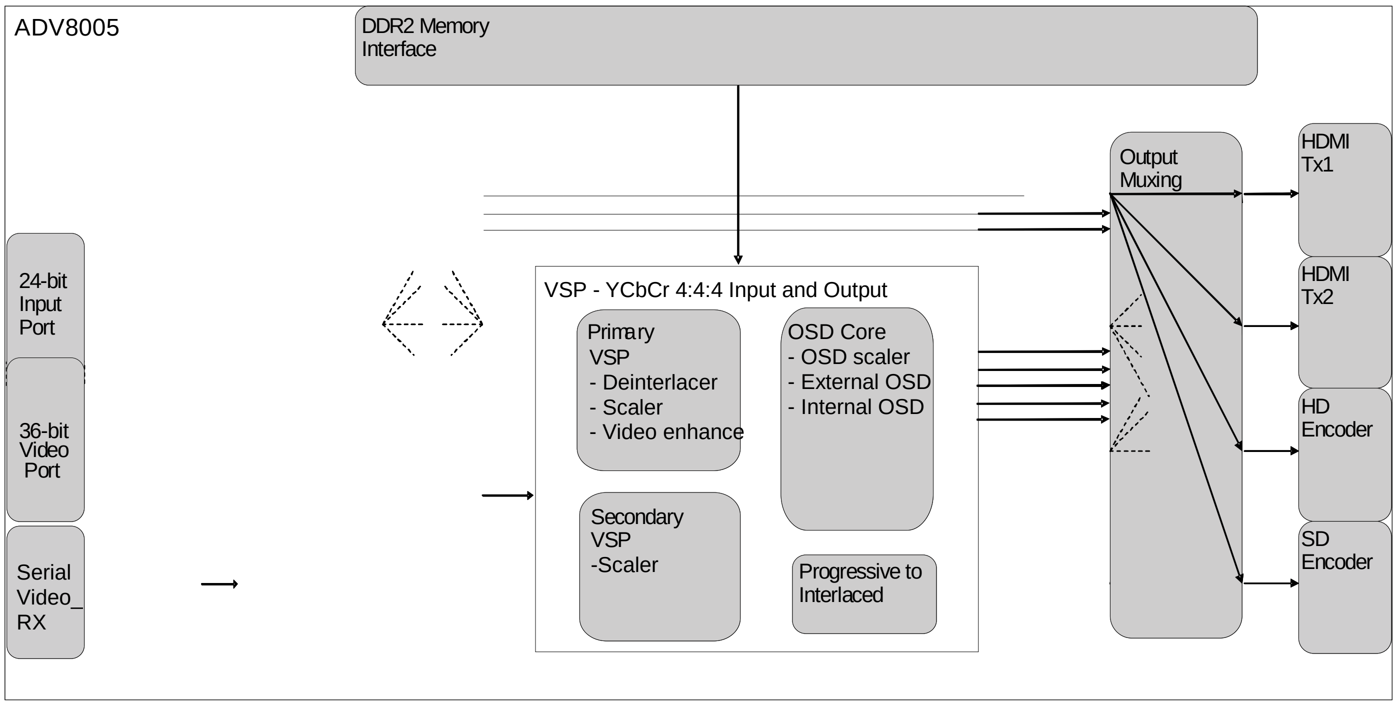


Figure 10: ADV8005 Simplified Block Diagram

A simplified block diagram of the ADV8005 can be seen in Figure 10. Video can be routed through the ADV8005 in a number of ways, for example, the OSD can be blended before the PVSP to display the OSD on all outputs, the OSD can be blended before the output to display the OSD on a single output. This has been divided into several modes of operation which are recommended by Analog Devices. These modes of operation are documented in Section 2.1 and outline the most practical modes in which to configure the ADV8005.

The four main processing blocks of the ADV8005 are described as follows.

PVSP: This is the main scaler of the ADV8005 and contains many of the signal processing functions. This block performs motion adaptive de-interlacing as well as scaling, ACE, FRC, cadence detection, CUE correction, RNR, BNR and MNR. PVSP utilizes the external DDR2 memory for such processes as FRC, de-interlacing and RNR. (Refer to Section 3.2 for more details on the PVSP.)

SVSP: This is the secondary scaler in the ADV8005 and is useful when providing an additional output resolution. The input to this block can only be progressive. This means an input format can only be connected to the SVSP if it is progressive or if it has been de-interlaced by the PVSP block. (Refer to Section 3.3 for more details on the SVSP.)

OSD Blend: This block overlays the generated OSD on the incoming video signal, from the Serial Video input lines or from the video TTL port. This is determined by an alpha factor as to how transparent the OSD will be. Depending on the source of the OSD data (from an external OSD solution or DDR2 memory), this is then synchronized with the incoming video signal. If the generated resolution is the same as the video, the OSD is simply overlaid on the video. If both are at different resolutions, the OSD scaler will first scale the OSD data to match the incoming video. (Refer to Section 3 for more details on the OSD.)

Progressive to Interlaced: The ADV8005 has two progressive to interlaced (PtoI) blocks, one of these is included as part of the SVSP. The second is a standalone block. The function of this block is exactly as named and can be used, for example, if the user was to convert an ED format such as 480p to a HD format such as 1080i. The PtoI block would be required as part of this conversion. (Refer to Section 3.2.3 and Section 3.3.3 for more details on the PtoI hardware blocks.)

2.1. ADV8005 MODES OF OPERATION

This section outlines the most practical modes in which the ADV8005 can be configured, as recommended by ADI. These modes describe the various ways to configure the VSP block, depending on the input formats as well as the outputs required. Table 3 outlines the various options afforded to the user in each mode. Depending on the desired output options, the appropriate mode should be chosen.

Table 3: ADV8005 Modes of Operation

	No. of Different Output Formats[1 2]	Interlaced Input Format Allowed	No. of Output Formats with OSD2	Input Video Copy-Protected
Mode 1	3	Yes	1	No
Mode 2	3	Yes	3	No
Mode 3	2	No (if using SVSP)	1	No
Mode 4	2	Yes	2	No
Mode 5	3	No (if using SVSP)	3	No
Mode 6	2	No (if using SVSP)	1	No
Mode 7	1-2	Yes	1-2	Yes
Mode 8	1-2	Yes	1-2	Yes
Mode 9 - Bypass	1	Yes	0	Yes
Mode 10 (PiP)	2	No (if using SVSP)	2	No
Mode 11 (PiP)	1	No (if using SVSP)	2	No
Mode 12 (Dual OSD)	1	Yes	2	Yes
Mode 13 (RX OSD)	2	Yes	2	Yes
Mode 14 (3 Inputs)	3	Yes	2	Yes

[1]For modes that offer four possible output formats, this means without reconfiguring the digital core. Only three possible output formats are supported at a single time: the input format and the two converted formats.
[2] The number of different output formats will be limited when using the ADV8005KBCZ-8B/8C.

Note: Table 3 does not list the definitive operation of the device in each mode. For example, in mode 3, it is listed as possible to have two different output resolutions and just have OSD on a single output. However, using the output muxing, it would also be possible to have a single output format (1080p in this case) with OSD going to several outputs. Table 3 provides only a guideline for the ADV8005 and should be used as such. Depending on user requirements, many of these modes could be tailored to a specific solution. Section 2.1.2 to Section 2.1.9 describe the usable modes of operation as recommended by ADI. These modes should be studied and the appropriate one selected for a given application.

2.1.1. *Selecting a Mode*

General guidelines for selecting a mode of operation involve selecting the location of certain blocks in the VSP section. For example, mode 5 and mode 6 both use the PVSP and SVSP in parallel. However, as the SVSP can only accept progressive formats, input video to the ADV8005 must be progressive. If interlaced, only the PVSP can be used. Therefore, a note should be kept of the input formats if selecting these in parallel mode.

The location of the OSD blend core must then be selected. This can be placed before the PVSP and both the input video and OSD can be scaled at the same time. However, depending on the application, the optimal solution may be to have both the input video and OSD scaled separately and then blended.

If blending the OSD after the PVSP, the OSD may need to be scaled to different resolutions. This can be done in two ways:

1. The OSD bitmap images are created at higher resolutions.
2. The OSD can be rendered at a single resolution and scaled internally in the ADV8005 using the OSD scaler.

There are limitations to both of these methods. Rendering OSDs at larger resolutions increases the system resources required to store these bitmaps. Alternatively, scaling the OSD internally in the part increases power consumption on the ADV8005.

The optimum solution to this depends on customer requirements and system capabilities. It should be chosen taking these considerations into account.

Note: For the following modes of operation, red indicates an active video path and black indicates a path is not used. If, for example, there are two red dashed lines, video may be available on one or the other but not on both.

2.1.2. *Mode 1*

Mode 1 should be used if:

- Three separate output formats are required
- Additional processing (BNR, RNR, and so on) is required on the new output formats
- OSD is required on a single output format (most likely the lowest quality of the converted formats)

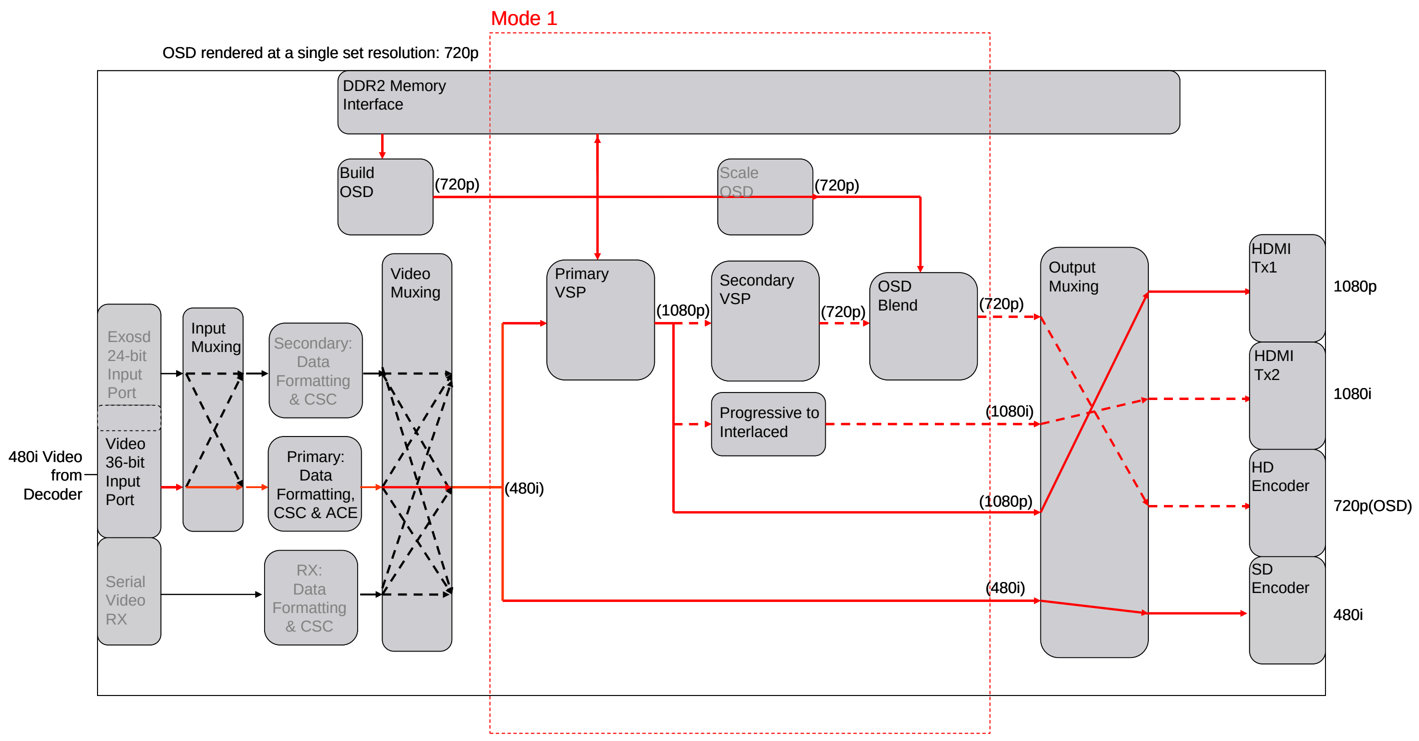


Figure 11: ADV8005 Mode 1 Configuration

Mode 1 places the PVSP after the input block. The output from this block is then sent to the SVSP or the PtoI converter. The OSD blend block can then be placed after the SVSP block.

In the example in Figure 11, the input resolution is taken to be 480i. This is then passed through the PVSP where it is converted to 1080p using motion adaptive de-interlacing. This can then be passed straight to the output, to the PtoI converter or, alternatively, to the SVSP and OSD blend. The example output formats generated using this mode are 720p (with OSD), 1080p and 1080i. The input SD format of 480i can also be passed to the SD encoder.

2.1.3. *Mode 2*

Mode 2 should be used if:

- Three separate output formats are required
- Additional processing (BNR, RNR, and so on) is required on the new output formats
- OSD is required on multiple outputs
- OSD and video scaling are to be kept separate

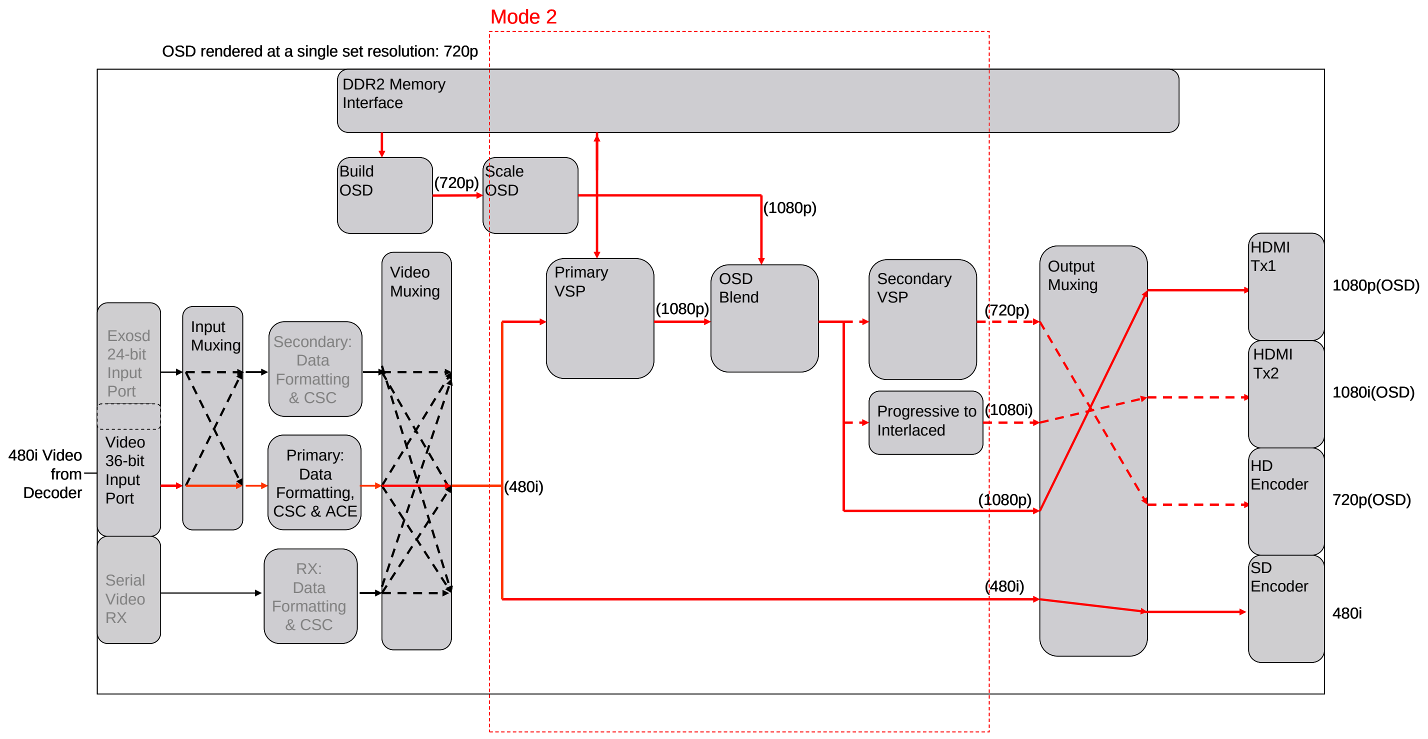


Figure 12: ADV8005 Mode 2 Configuration

Mode 2 places the PVSP after the input block. The output from this is sent to the OSD which is in turn sent to the SVSP or PtoI converter. This mode is very similar to mode 4, except that the OSD position has swapped with the PVSP. The primary reason is that, in this case, the OSD data is not overlaid on the incoming video data and then scaled, but rather scaled and then overlaid. Scaling the video and OSD separately may improve the quality of the video input to the SVSP. If it is possible, it is better to scale video and OSD separately and then blend rather than scaling both together.

The example in Figure 12 takes a 480i video signal and scales this to 1080p. This is then overlaid with OSD data scaled to 1080p. This example can generate three different output formats (720p, 1080i, and 1080p) as well as outputting the input SD standard of 480i.

2.1.4. *Mode 3*

Mode 3 should be used if:

- Two separate upscaled resolutions are required
- De-interlacing is not required
- OSD is required on one resolution only (preferably the higher resolution output)

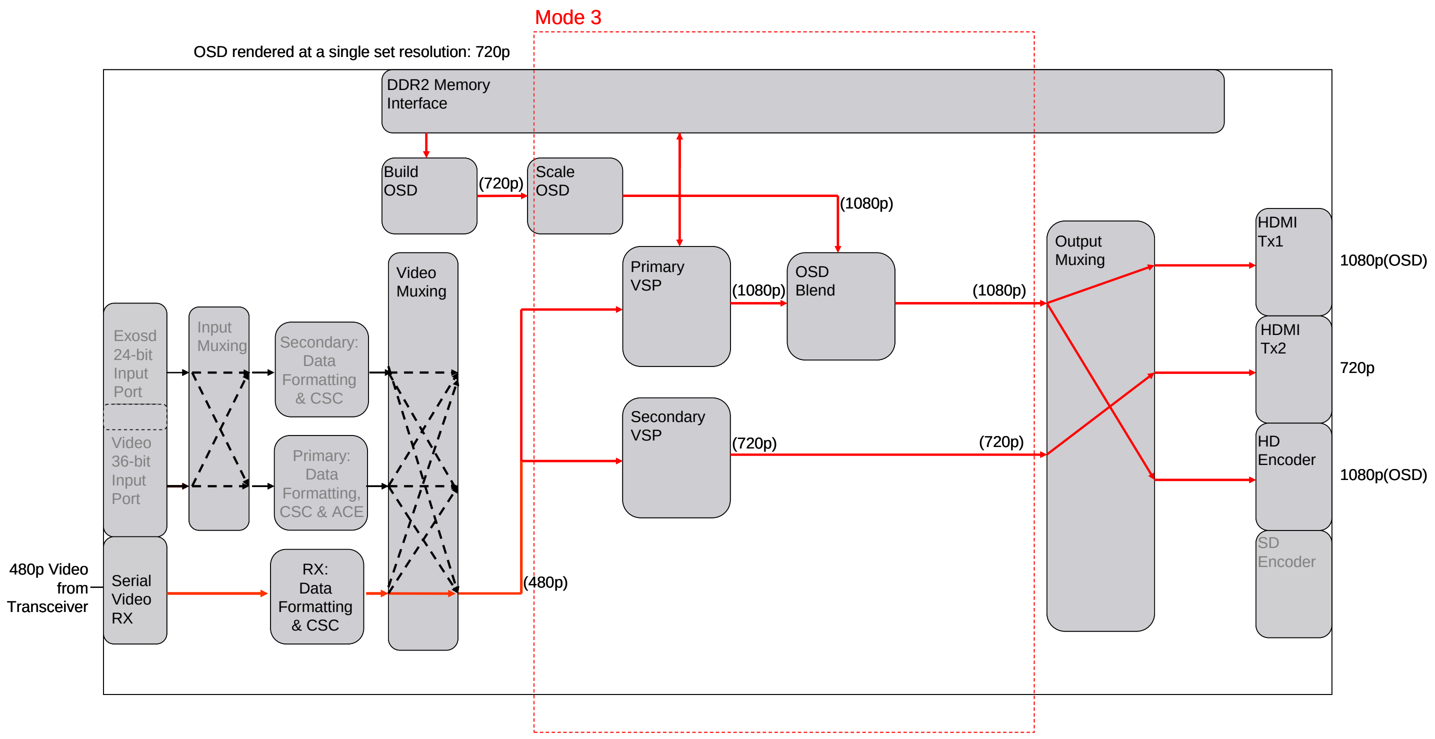


Figure 13: ADV8005 Mode 3 Configuration

Both the PVSP and SVSP work in parallel in this mode. As the OSD is only on one data path, it will only be displayed at a single resolution. As shown in the example in Figure 13, the input to the SVSP must be a progressive format. Therefore, this mode can only be used when the input is progressive. This mode allows the user to overlay the OSD on the higher resolution output(s).

It can be seen in Figure 13 that the same output formats can be generated in this mode. However, the OSD is now only generated on the higher resolution outputs. This mode allows the user to generate two different outputs resolutions and only display OSD on one output.

Note: De-interlaced inputs can be input to the device in this mode; however the SVSP can only accept progressive input formats. Therefore, the SVSP would be excluded from the processing in this case.

2.1.5. *Mode 4*

Mode 4 should be used if:

- Three possible separate output formats are required
- Additional processing is required on the new output formats
- OSD is required on multiple output formats

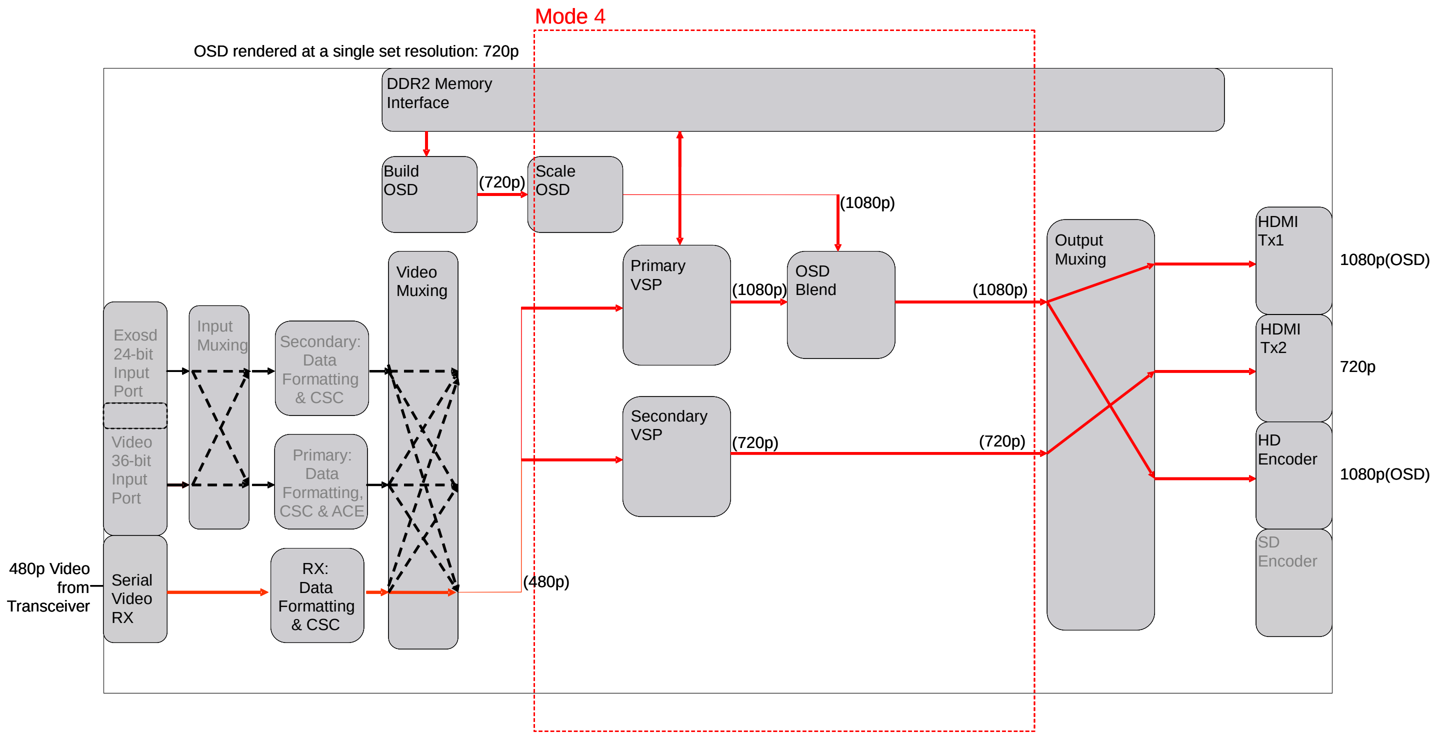


Figure 14: ADV8005 Mode 4 Configuration

Mode 4 places the OSD blend block before the PVSP. The output of the PVSP is then input to both the SVSP and the PtoI converter. The OSD is overlaid on all output formats. In addition, high performance PVSP processing is performed on all outputs which can improve video quality at all resolutions. While blending the OSD on the incoming video, the OSD can be scaled to the necessary resolution of the incoming video using the OSD scaler.

In the example in Figure 14, the input resolution is taken to be 480p. The OSD is downscaled, blended, and passed to the PVSP which scales to 1080p. The advantage of configuring the ADV8005 core in this way is that by including the PVSP on multiple data path, additional processing can be included on other outputs also. In Figure 14, three different formats (1080p, 1080i, and 720p) can be generated.

2.1.6. *Mode 5*

Mode 5 should be used if:

- Two separate upscaled resolutions are required
- De-interlacing is not required
- OSD is required on both output formats

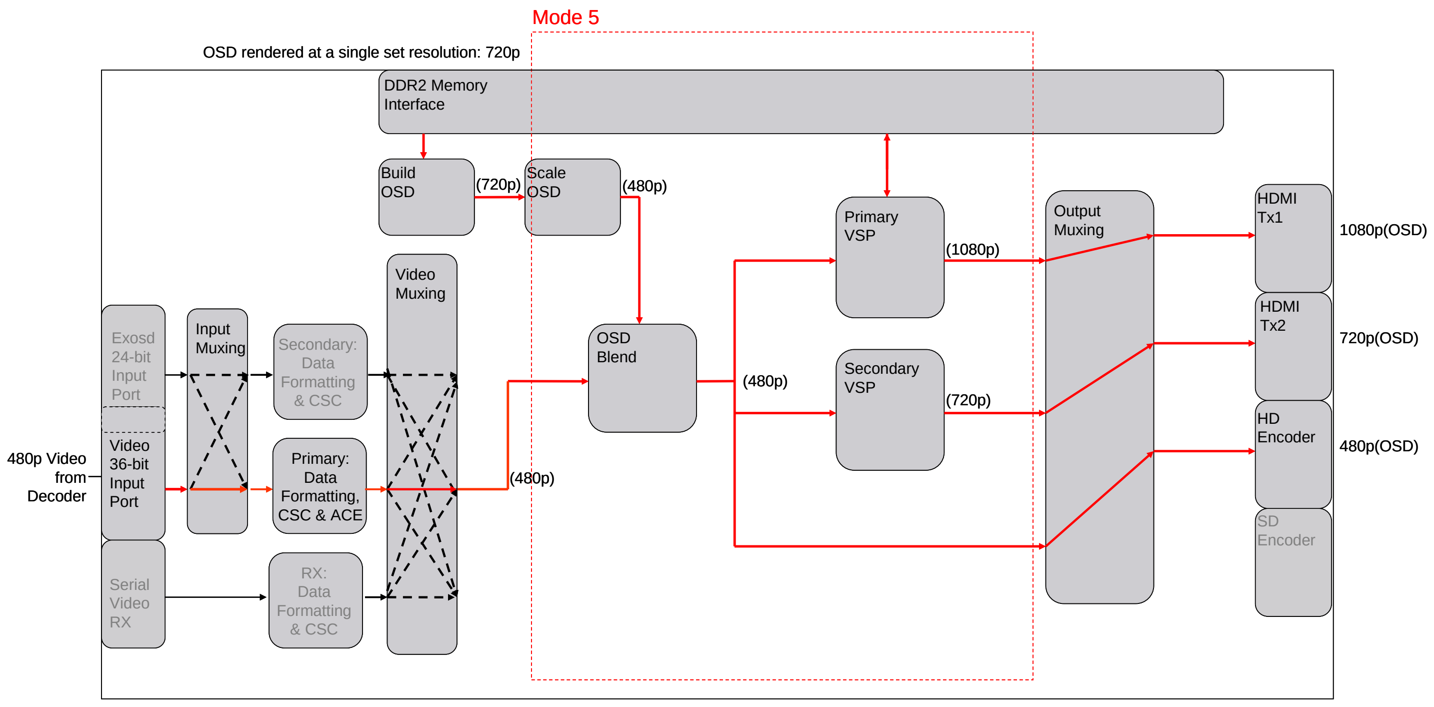


Figure 15: ADV8005 Mode 5 Configuration

Mode 5 places the OSD blend block before both the PVSP block and the SVSP block. Both the PVSP block and the SVSP work in parallel in this mode. As the OSD is before both scalers, the OSD will be available on all the outputs. As mentioned in Section 2.1.4, the input to the SVSP must be progressive, therefore, this mode can only be used when the input is progressive.

As can be seen in the example in Figure 15, the output formats required are 1080p and 720p. The 1080p format is converted through the PVSP block with the SVSP upscaling to 720p. It should be noted from Figure 15, that when large video scalings are required, these should be processed by the PVSP block for optimal performance.

Note: De-interlaced inputs can be input to the device in this mode; however the SVSP can only accept progressive input formats. Therefore, the SVSP would be excluded from the processing in this case.

2.1.7. *Mode 6*

Mode 6 should be used if:

- Two separate upscaled resolutions are required
- De-interlacing is not required
- OSD is required on one resolution only (preferably the lower upscaled resolution output)

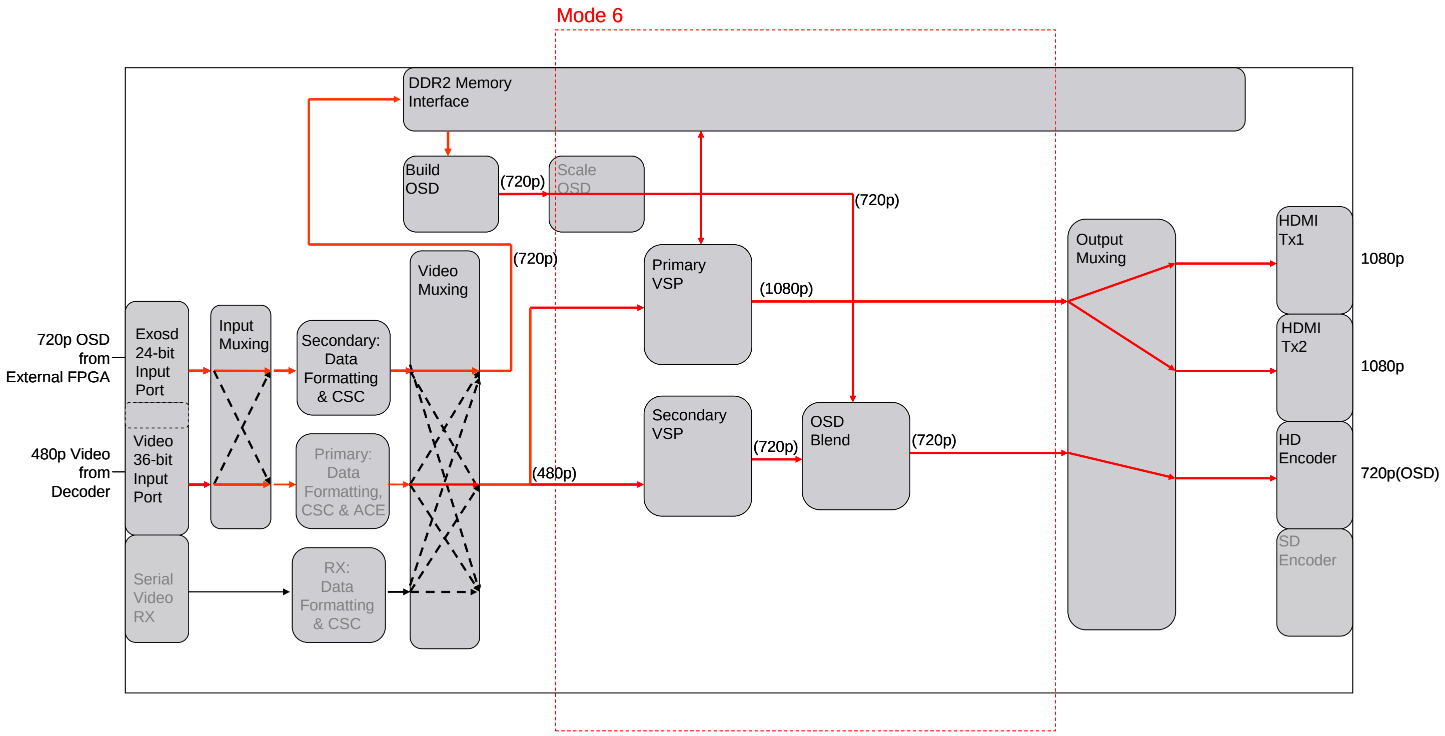


Figure 16: ADV8005 Mode 6 Configuration

Mode 6 places the OSD blend block after the SVSP. Both the PVSP block and SVSP work in parallel in this mode. As the OSD is only on one data path, it will only be displayed at a single resolution. As shown in the example in Figure 16, the input to the SVSP must be a progressive format. Therefore, this mode can only be used when the input is progressive.

It can be seen from Figure 16 that the same output formats can be generated in this mode. However, due to the change in location of the OSD blend block, the OSD can only be generated on a single output resolution.

Note: De-interlaced inputs can be input to the device in this mode. However, the SVSP can only accept progressive input formats. Therefore, the SVSP would be excluded from the processing in this case as would the OSD blend.

2.1.8. *Mode 7*

Mode 7 should be used if:

- HDMI input video is copy protected
- Additional processing is required on the new output formats
- OSD is required on multiple outputs
- OSD and video scaling are to be kept separate

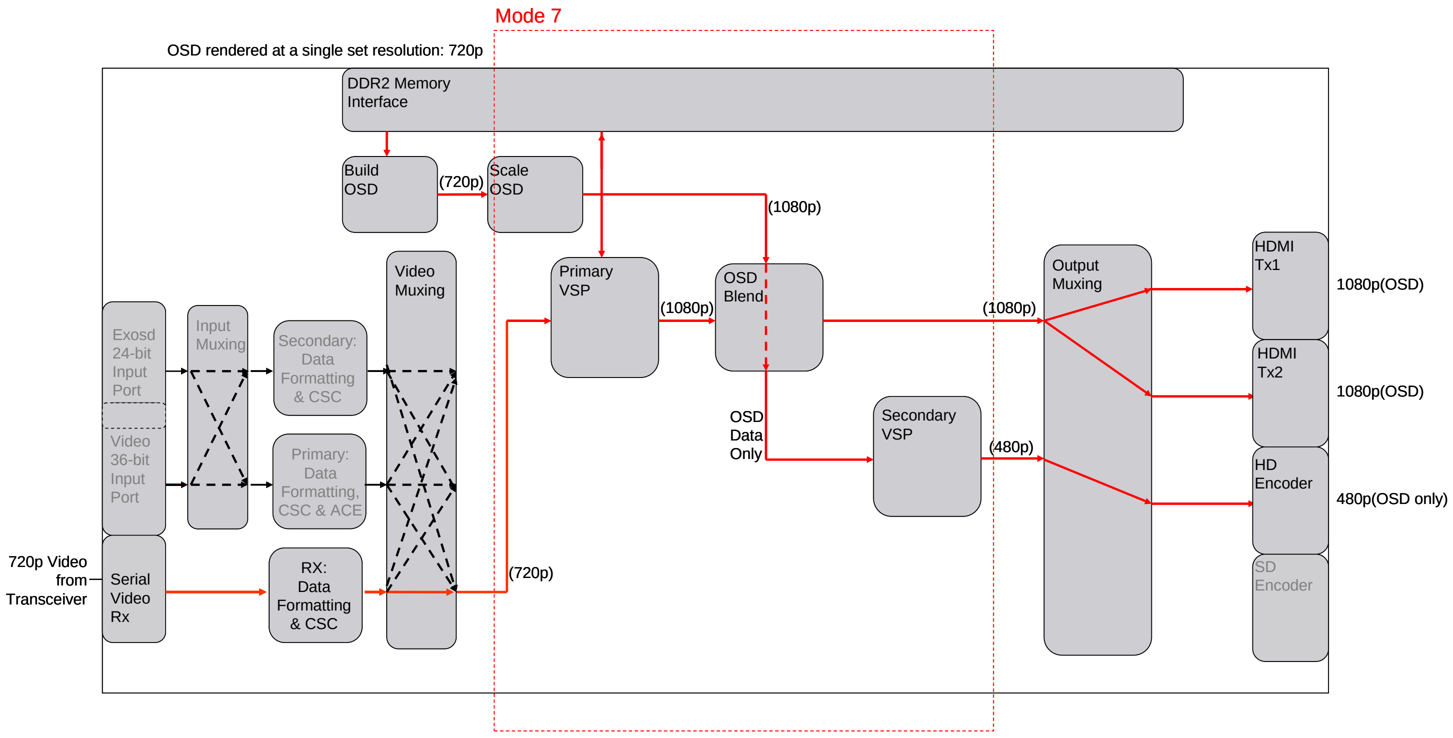


Figure 17: ADV8005 Mode 7 Configuration

Mode 7 is different to other modes in that OSD is not overlaid on video data on certain outputs but rather just output on its own. In certain cases where HDMI video from an upstream IC is copy protected, video data can be output on HDMI outputs but not analog outputs. However, OSD data can still be displayed on analog output, for example, to indicate system status or to recover the system from an error-like state.

In this mode, the input format is 720p from an external video transceiver (this could also come from the Video TTL input if video is from an upstream HDMI IC) and is passed to the PVSP. This is upscaled, blended with the 1080p OSD data and sent to both HDMI transmitters. Because this may be copy protected, this cannot be passed to the analog outputs. The OSD on its own, however, can be passed directly to these outputs. In the example in Figure 17, the OSD is scaled down to 480p and passed to the HD encoder.

2.1.9. *Mode 8*

Mode 8 should be used if:

- HDMI input video is copy protected
- Additional processing is required on the new output formats
- OSD is required on multiple outputs
- OSD and video scaling are to be kept separate

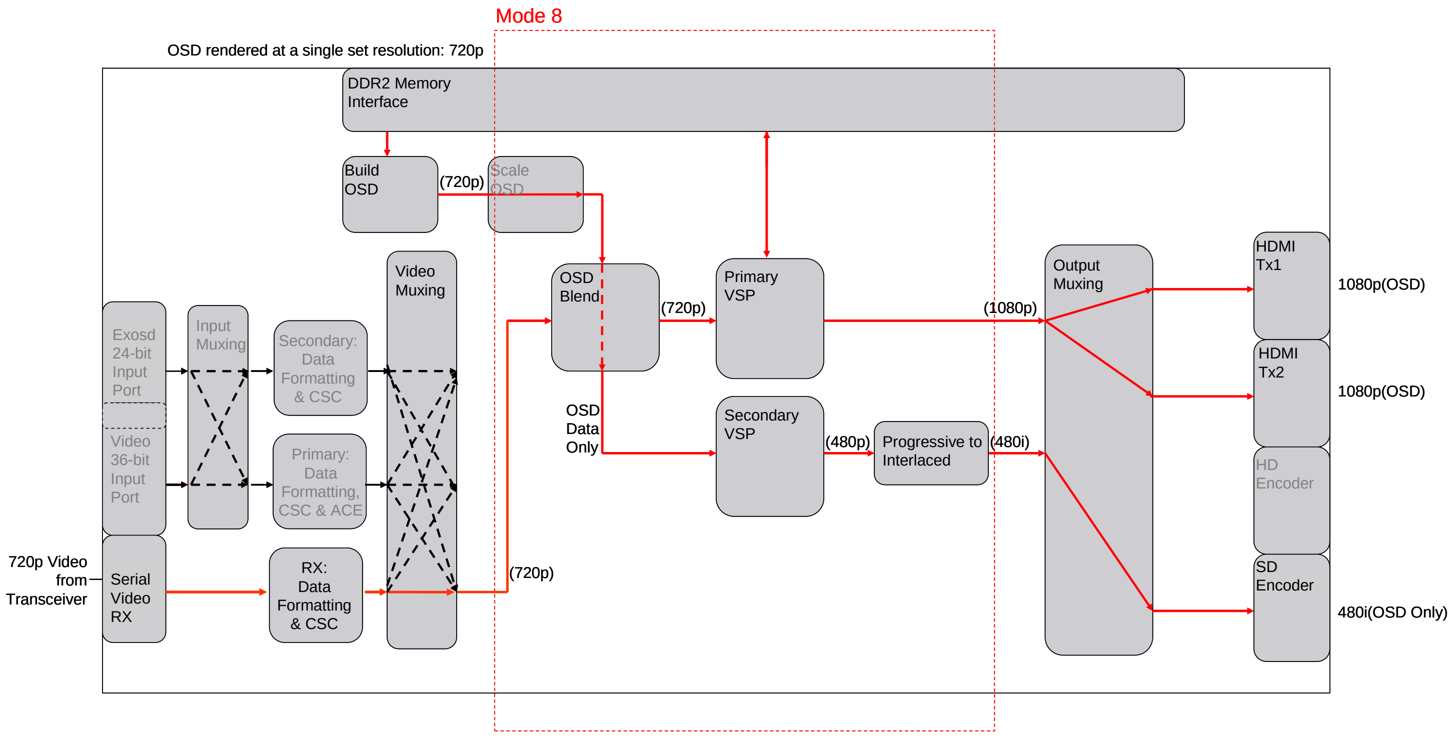


Figure 18: ADV8005 Mode 8 Configuration

Mode 8 is similar to mode 7 in that OSD is not overlaid on the input video but rather output as the OSD video on its own. This may be required when HDMI video from an upstream IC is copy protected (as in Figure 18), but the OSD is still required on the analog outputs.

In this mode, the input format is 720p from an external video transceiver and passed to the OSD blend. As the OSD is generated at the same resolution as the input video, they are just blended. This is then passed to the PVSP where it is upscaled and sent to both HDMI transmitters. If this data is copy protected, this cannot be passed to the analog outputs. The OSD on its own, however, can be passed directly to these outputs. In the example in Figure 18, the OSD is scaled down to 480p and passed through the PtoI block and sent out on the SD encoder. The difference between mode 7 and mode 8 is very similar to the difference between modes 2 and 4. Ideally the video and OSD should be scaled separately and then blended.

2.1.10. *Mode 9 - Bypass*

Mode 9 should be used if input video is to be passed straight to the output with no video processing.

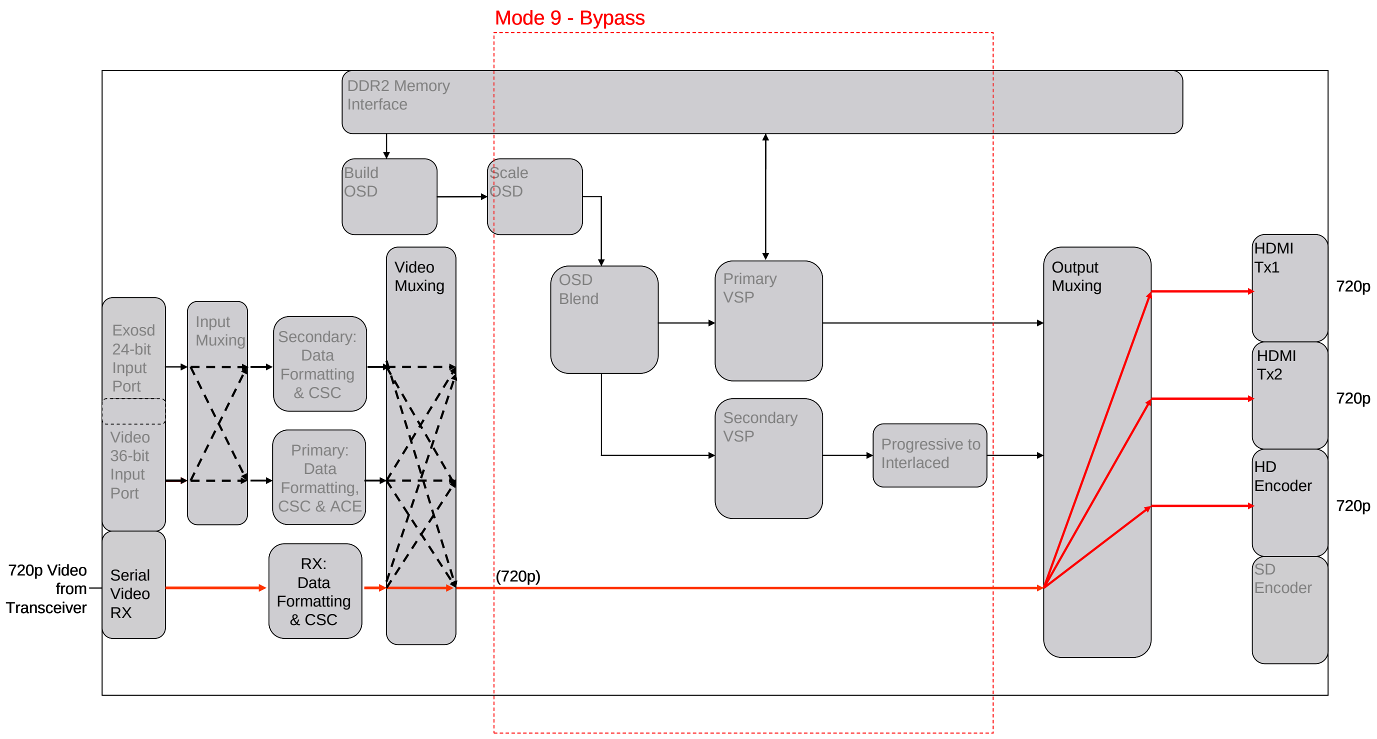


Figure 19: ADV8005 Mode 9 Configuration

Mode 9 is used in cases where no processing is required on the input video. This can be passed directly to the output. No access to external DDR2 memory is required in this case.

2.1.11. *Mode 10 – Picture in Picture (PiP) (External OSD Less Than 720p)*

Mode 10 should be used if:

- OSD data is input via the EXOSD TTL 24-bit input port
- OSD data input via the EXOSD TTL 24-bit input port is less than 720p

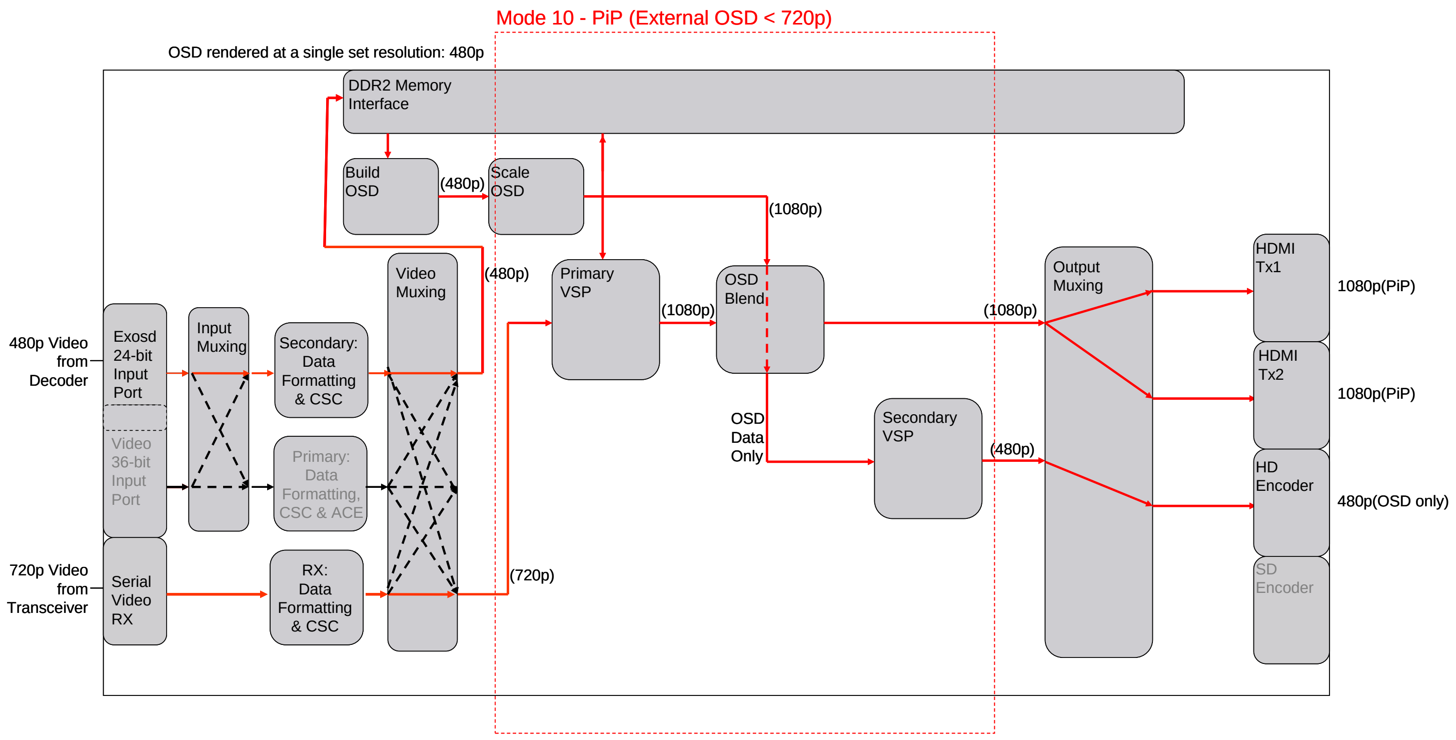


Figure 20: ADV8005 Mode 10 Configuration

Mode 10 is used to support the external input of either part of or the complete OSD from another device, for example, an MCU. With support for HS, VS, DE and CLK, the external OSD input can also be used to input video data. Using mode 10, the external OSD bus can be used to support picture in picture (PiP) with two video streams.

In this mode, the input from the EXOSD TTL 24-bit input port is written into DDR2 memory and read back by the OSD core as a region of the OSD. This region is then blended with input video.

2.1.12. *Mode 11 – PIP (External OSD Greater Than or Equal To 720p)*

Mode 11 should be used if:

- OSD data is input via the EXOSD TTL 24-bit input port
- OSD data input via the EXOSD TTL 24-bit input port is greater than or equal to 720p

Mode 11 is used to support the external input of either part of or the complete OSD from another device, for example, an MCU. With support for HS, VS, DE and CLK, the external OSD input can also be used to input video data. Using mode 10, the external OSD bus can be used to support picture in picture (PiP) with two video streams. The difference between mode 10 and mode 11 is the resolution of the incoming video – mode 11 can support incoming video of 720p or greater.

In this mode, the input from EXOSD TTL 24-bit input port is routed to the SVSP where it is scaled before being written into DDR2 memory. The OSD core then reads back the data as one OSD region and blends this region with input video.

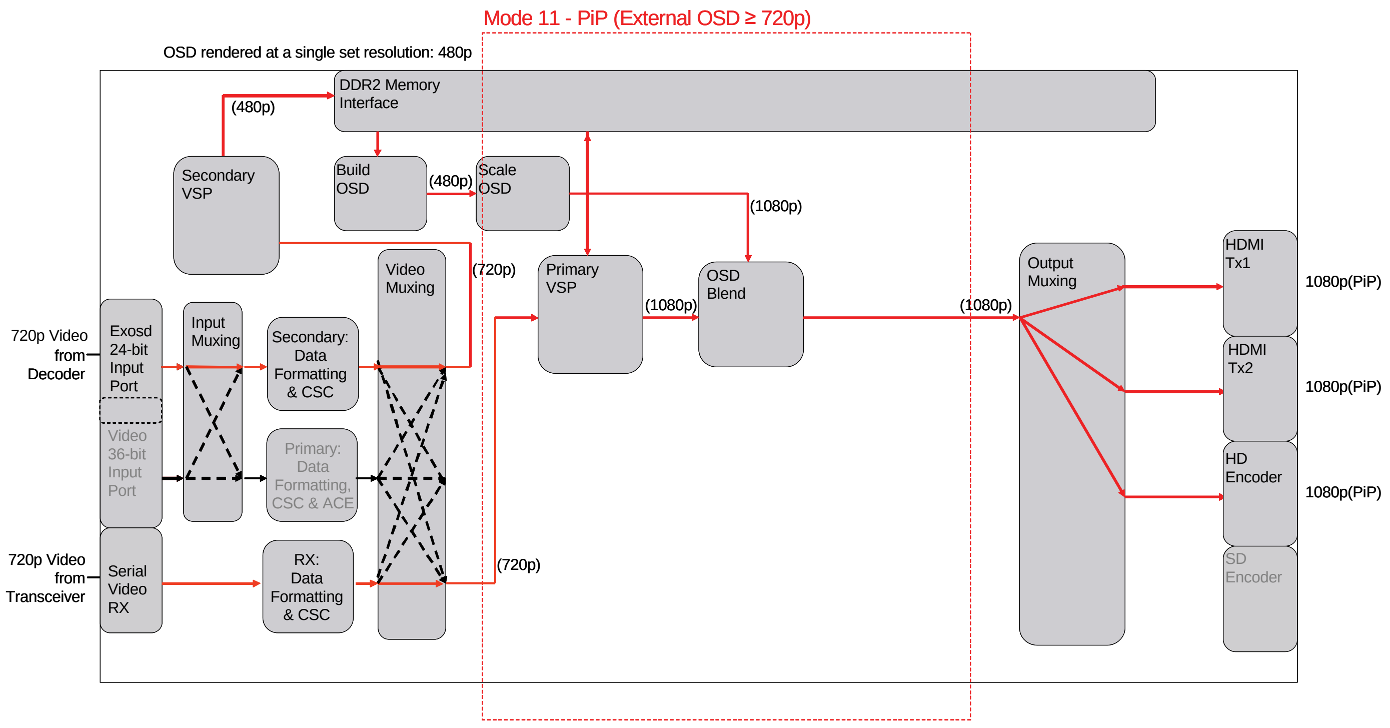


Figure 21: ADV8005 Mode 11 Configuration

2.1.13. *Mode 12 – Dual Zone OSD*

Mode 12 should be used if OSD output is required in dual zones.

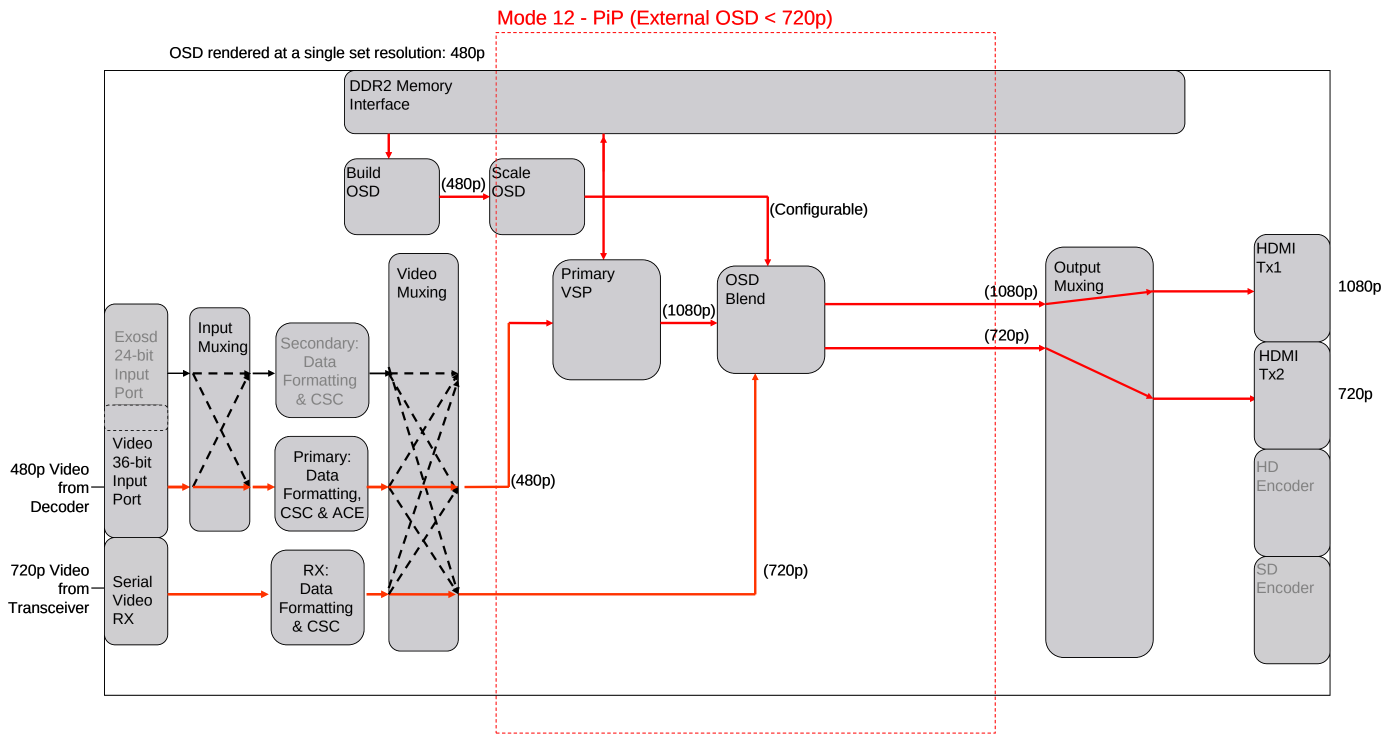


Figure 22: ADV8005 Mode 12 Configuration

Mode 12 is used to support dual zone OSD output without disturbing either video stream. Using this mode, two inputs (for example, 480p from the video TTL input port and 720p from the Serial Video Rx) can be applied to the part, processed and connected to the OSD core. The OSD can be blended onto one or other of the two video streams and switched between the two video streams without causing any disturbance to either.

2.1.14. *Mode 13 – OSD from HDMI RX*

Mode 13 should be used if the ADV8005 is being used in conjunction with a legacy standalone OSD generator with an HDMI interface.

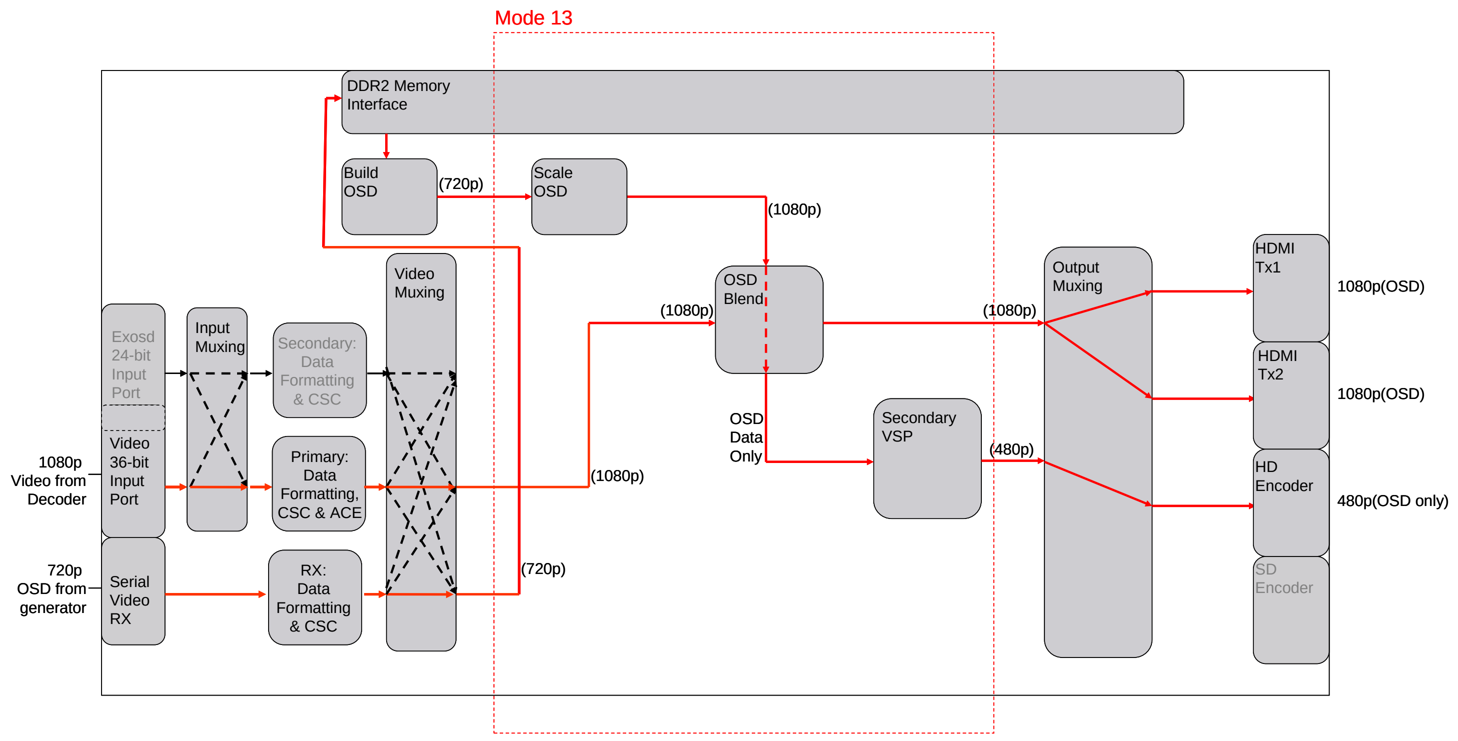


Figure 23: ADV8005 Mode 13 Configuration

Mode 13 is used to support OSD input from an OSD generator with an HDMI interface. Using this mode, the Serial Video Rx video is loaded into memory before being called out by the OSD core. This video can then be scaled and blended with the video on the primary video channel. It is possible to output the unblended video, the blended video or the raw OSD.

2.1.15. *Mode 14 – Handling Triple Inputs*

Mode 14 should be used if three independent video streams are required on the output of the ADV8005.

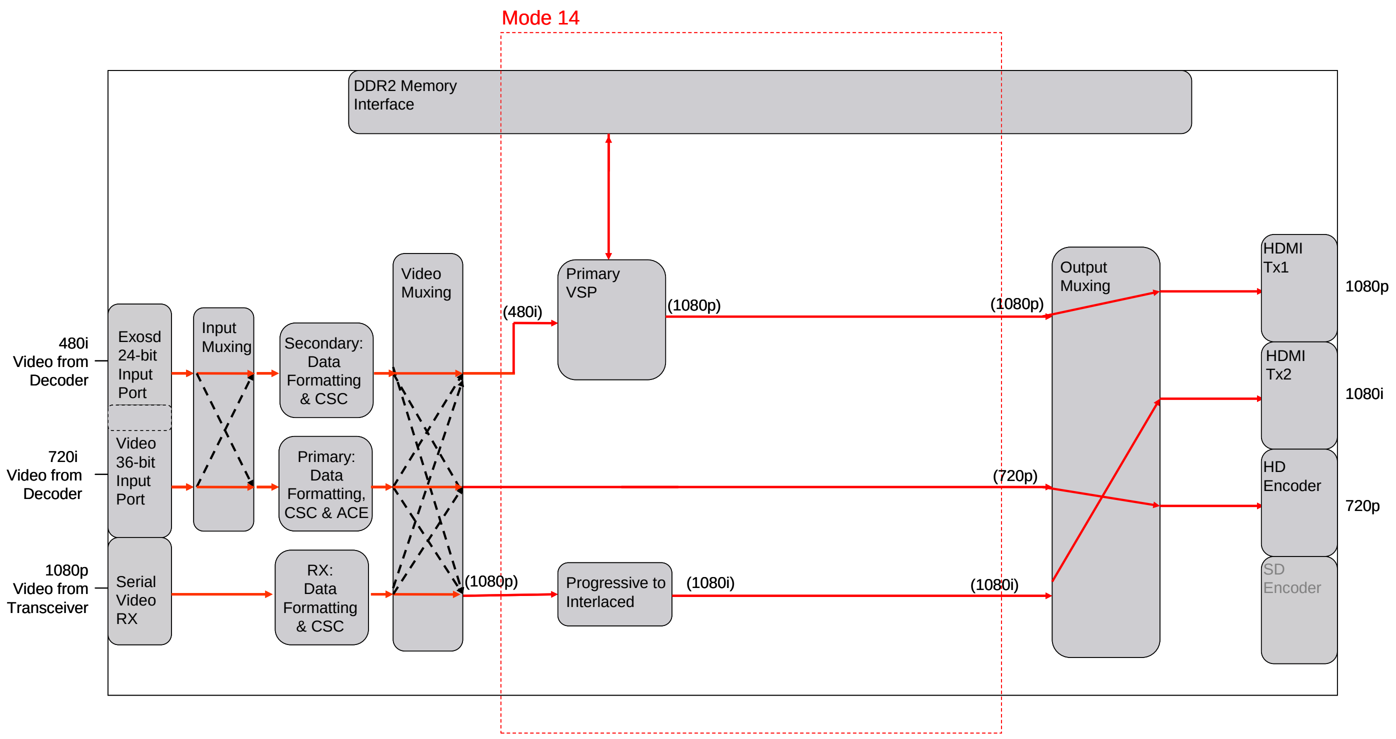


Figure 24: ADV8005 Mode 14 Configuration

Mode 14 is used to support three independent video streams. The independent video streams are input on the video TTL and EXOSD TTL inputs and the Serial Video Rx. These video streams can then be routed through internal processing blocks (for example, PVSP or progressive to interlaced converter) or connected directly to the backend transmission blocks, for example, HDMI transmitters and encoder.

2.2. ADV8005 TOP LEVEL OVERVIEW

This section documents the ADV8005 top level register descriptions, explaining some of the registers required to configure the part which are not section or hardware block specific. For more details on block specific settings, refer to their appropriate sections.

Note: This section details the ADV8005KBCZ-8A/8N. Other versions of the ADV8005 do not offer the same functionality, for example, single Tx or no encoder.

2.2.1. *Video Muxing*

There are several blocks which make up the ADV8005 VSP, as described in Section 2. The digital core of the ADV8005 offers flexible routing of video data, as shown in Figure 25.

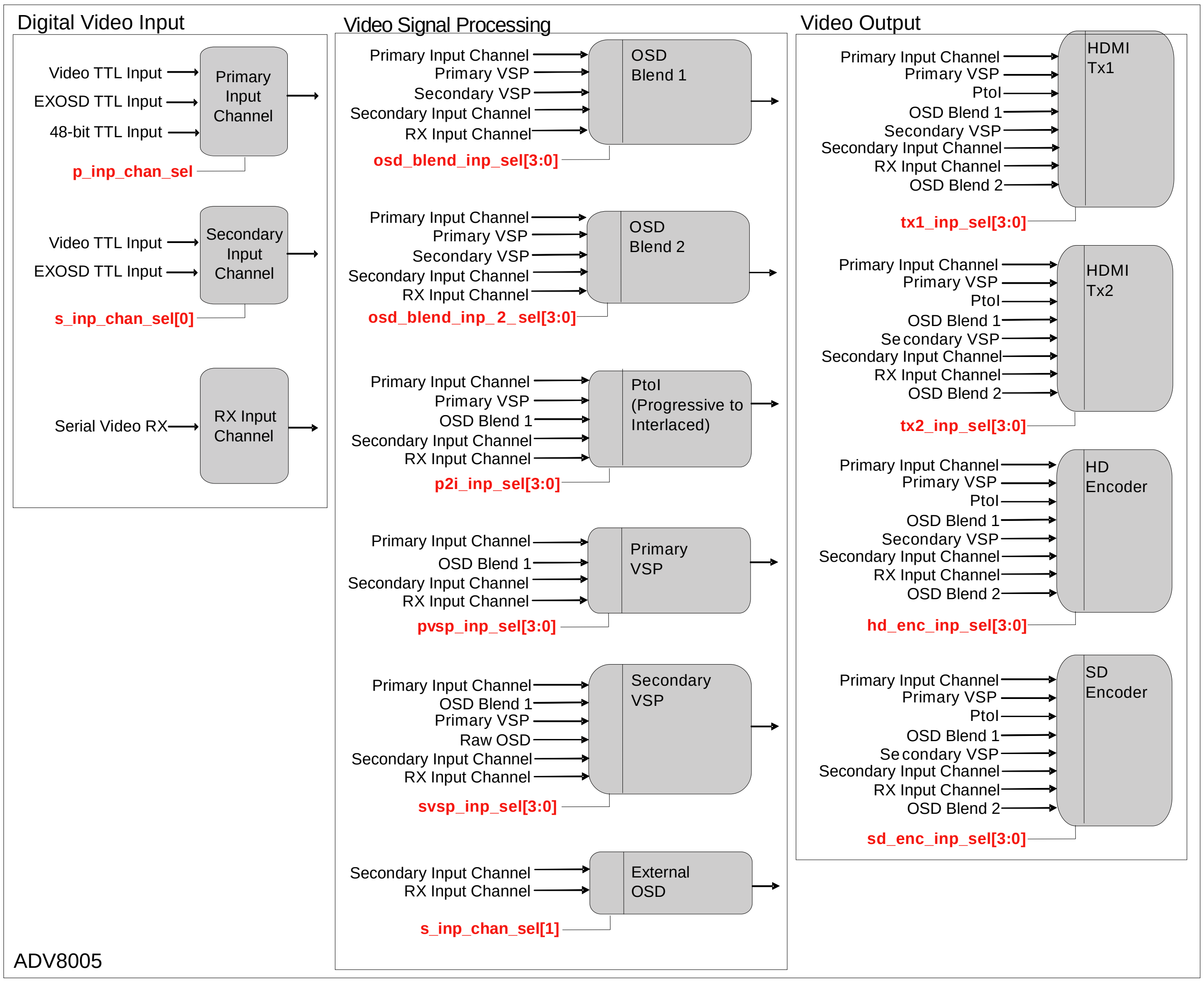


Figure 25: ADV8005 Digital Core Muxing

The following registers are used to configure the video routed through the ADV8005.

tx1_inp_sel[3:0], IO Map, *Address 0x1A03[7:4]*
This signal is used to select the video source for the HDMI Tx1.

Function

tx1_inp_sel[3:0]	Description
0x00 (default)	From Primary Input Channel
0x01	From Primary VSP
0x02	From PtoI Converter
0x03	From Internal OSD Blend 1
0x04	From Secondary VSP/PtoI Converter
0x05	From Secondary Input Channel
0x06	From RX Input
0x07	From Internal OSD Blend 2

tx2_inp_sel[3:0], IO Map, *Address 0x1A03[3:0]*
This signal is used to select the video source for the HDMI Tx2.

Function

tx2_inp_sel[3:0]	Description
0x00 (default)	From Primary Input Channel
0x01	From Primary VSP
0x02	From PtoI Converter
0x03	From Internal OSD Blend 1
0x04	From Secondary VSP/PtoI Converter
0x05	From Secondary Input Channel
0x06	From RX Input
0x07	From Internal OSD Blend 2

hd_enc_inp_sel[3:0], IO Map, *Address 0x1A04[7:4]*
This signal is used to select the video source for the HD encoder. When using the encoder in SD only mode, this signal must be set to the same value as sd_enc_inp_sel.

Function

hd_enc_inp_sel[3:0]	Description
0x00 (default)	From Primary Input Channel
0x01	From Primary VSP
0x02	From PtoI Converter
0x03	From Internal OSD Blend 1
0x04	From Secondary VSP/PtoI Converter
0x05	From Secondary Input Channel
0x06	From RX Input
0x07	From Internal OSD Blend 2

sd_enc_inp_sel[3:0], IO Map, *Address 0x1A04[3:0]*
This signal is used to select the video source for the SD encoder. When using the encoder in SD only mode, hd_enc_inp_sel must be set to the same value as this signal.

Function

sd_enc_inp_sel[3:0]	Description
0x00 (default)	From Primary Input Channel
0x01	From Primary VSP
0x02	From PtoI Converter
0x03	From Internal OSD Blend 1
0x04	From Secondary VSP/PtoI Converter
0x05	From Secondary Input Channel
0x06	From RX Input
0x07	From Internal OSD Blend 2

svsp_inp_sel[3:0], IO Map, *Address 0x1A05[7:4]*
This signal is used to select the video source for the Secondary VSP.

Function

svsp_inp_sel[3:0]	Description
0x00 (default)	From Primary Input Channel
0x01	From Internal OSD Blend 1
0x02	From Primary VSP
0x03	From Internal OSD (OSD only, no blend)
0x04	From Secondary Input Channel
0x05	From RX Input
0x06	From Horizontal Prescaler

pvsp_inp_sel[3:0], IO Map, *Address 0x1A05[3:0]*
This signal is used to select the video source for the Primary VSP.

Function

pvsp_inp_sel[3:0]	Description
0x00 (default)	From Primary Input Channel
0x01	From Internal OSD Blend 1
0x02	From Secondary Input Channel
0x03	From RX Input
0x04	From Secondary VSP
0x05	From Horizontal Pre-scaler

p2i_inp_sel[3:0], IO Map, *Address 0x1A06[7:4]*
This signal is used to select the video source for the Progressive to Interlaced converter.

Function

p2i_inp_sel[3:0]	Description
0x00 (default)	From Primary VSP
0x01	From Internal OSD Blend 1
0x02	From Secondary Input Channel
0x03	From RX Input
0x04	From Primary Input Channel

osd_blend_inp_sel[3:0], IO Map, *Address 0x1A06[3:0]*
This signal is used to select the video source to the OSD Blend block.

Function

osd_blend_inp_sel[3:0]	Description
0x00 (default)	From Primary Input Channel
0x01	From Secondary VSP/PtoI Converter
0x02	From Primary VSP
0x03	From Secondary Input Channel
0x04	From RX Input

osd_blend_inp_2_sel[3:0], IO Map, *Address 0x1A08[3:0]*
This signal is used to select the video to be blended on OSD channel 2.

Function

osd_blend_inp_2_sel[3:0]	Description
0x00 (default)	From Primary Input Channel
0x01	From Secondary VSP/PtoI Converter
0x02	From Primary VSP
0x03	From Secondary Input Channel
0x04	From RX Input

For example, when using the ADV8005 in Mode 3 (described in Section 2.1.4), the following register settings are needed to configure the video data path:

1A 1A03 34; Output of OSD blend to HDMI Tx1, Output of Secondary VSP to HDMI Tx2
1A 1A04 30; Output of OSD blend to HD encoder, SD encoder not used.
1A 1A05 00; Input to ADV8005 to both Primary and Secondary VSP.
1A 1A06 02; Progressive to Interlaced converter not used, output from Primary VSP to OSD blend.

These four register writes configure the hardware blocks in the ADV8005 in mode 3. More registers will need to be configured depending on the input and desired video standards.

2.2.2. *Digital Video Input*

The ADV8005 has three means of receiving video: the video TTL input and the EXOSD TTL input which constitute the flexible 60-bit TTL input port, and the Serial Video Rx. Each of the TTL inputs can be connected to one of the input channels – the primary input channel or the secondary input channel. The Serial Video Rx is always connected to the RX input channel. Each channel features a dedicated input formatter, color space converter (CSC) and dither block. The primary input channel also features an automatic contrast enhancement (ACE) control. The ADV8005 input channels are illustrated in Figure 26, Figure 27 and Figure 28.

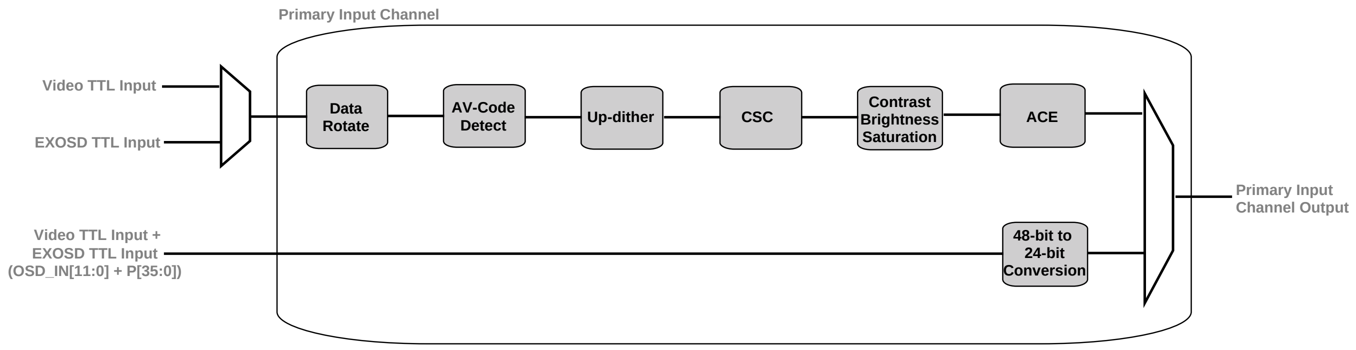

Primary Input Channel
Video TTL Input
EXOSD TTL Input
Data Rotate
AV-Code Detect
Up-dither
CSC
Contrast Brightness Saturation
ACE
Primary Input Channel Output
Video TTL Input + EXOSD TTL Input (OSD_IN[11:0] + P[35:0])
48-bit to 24-bit Conversion

Figure 26: Video TTL Input Channel

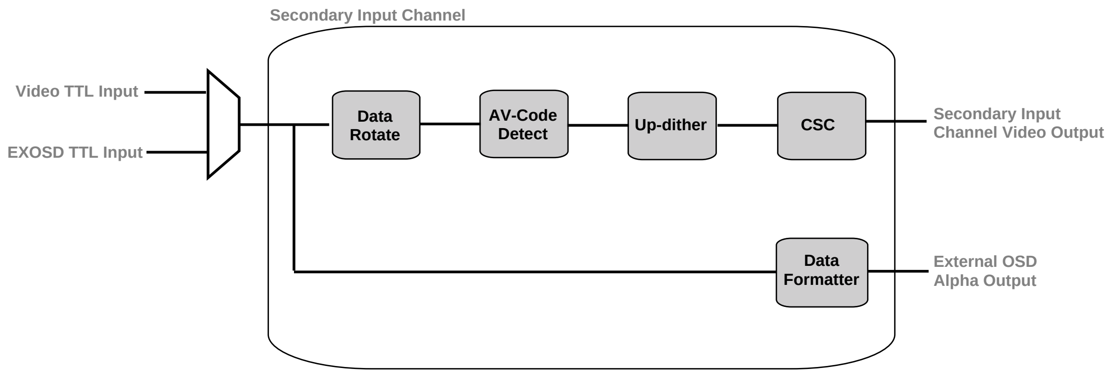


Figure 27: EXOSD TTL Input Channel

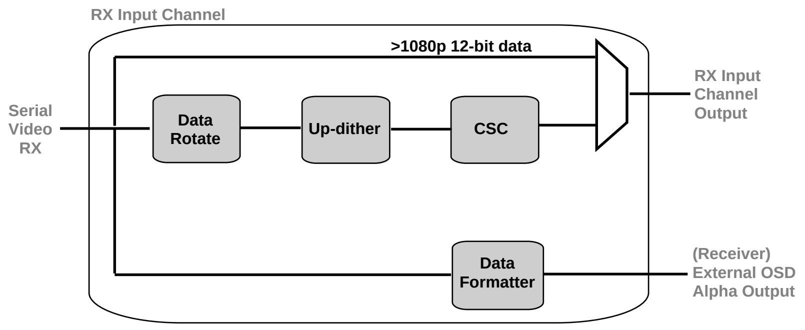


Figure 28: RX Input Channel

2.2.2.1. *Video TTL Input*

The video TTL input pins are defined as follows:

- P[47:0]
- HS
- VS
- DE
- PCLK

The video TTL input pins can be connected to either the primary input channel (refer to Section 2.2.2.6) or the secondary input channel (refer to Section 2.2.2.7).

2.2.2.2. *EXOSD TTL Input*

The EXOSD TTL input pins are defined as follows:

- OSD_IN[23:16]
- OSD_IN[15]/VBI_SCK
- OSD_IN[14]/VBI_MOSI
- OSD_IN[13]/VBI_SCK
- OSD_IN[12:0]
- OSD_HS
- OSD_VS
- OSD_DE
- OSD_CLK

The EXOSD TTL input pins can be connected to either the primary input channel (refer to Section 2.2.2.6) or the secondary input channel (refer to Section 2.2.2.7).

2.2.2.3. *TTL Output*

The ADV8005 includes a TTL output port, The external OSD TTL input pins (OSD_IN[23:0]) and 12 of the TTL input pins (P35:24) can function as TTL output pins (refer to Table 91 and Table 92. If all 36 TTL pins are used as outputs, this leaves only 24 pins for TTL inputs.

Appendix C describes the different pinout options available for the TTL input and output buses. HS, VS, DE and the TTL clock are output on the following pins:

- OSD_IN[23:0]
- P[35:24]
- OSD_HS
- OSD_VS
- OSD_DE
- OSD_CLK

The video data can be output at pixel frequencies up to 162 MHz. Only single data rate video is supported on the TTL output bus – it is not possible to clock video out on the rising and falling edge of the TTL output clock.

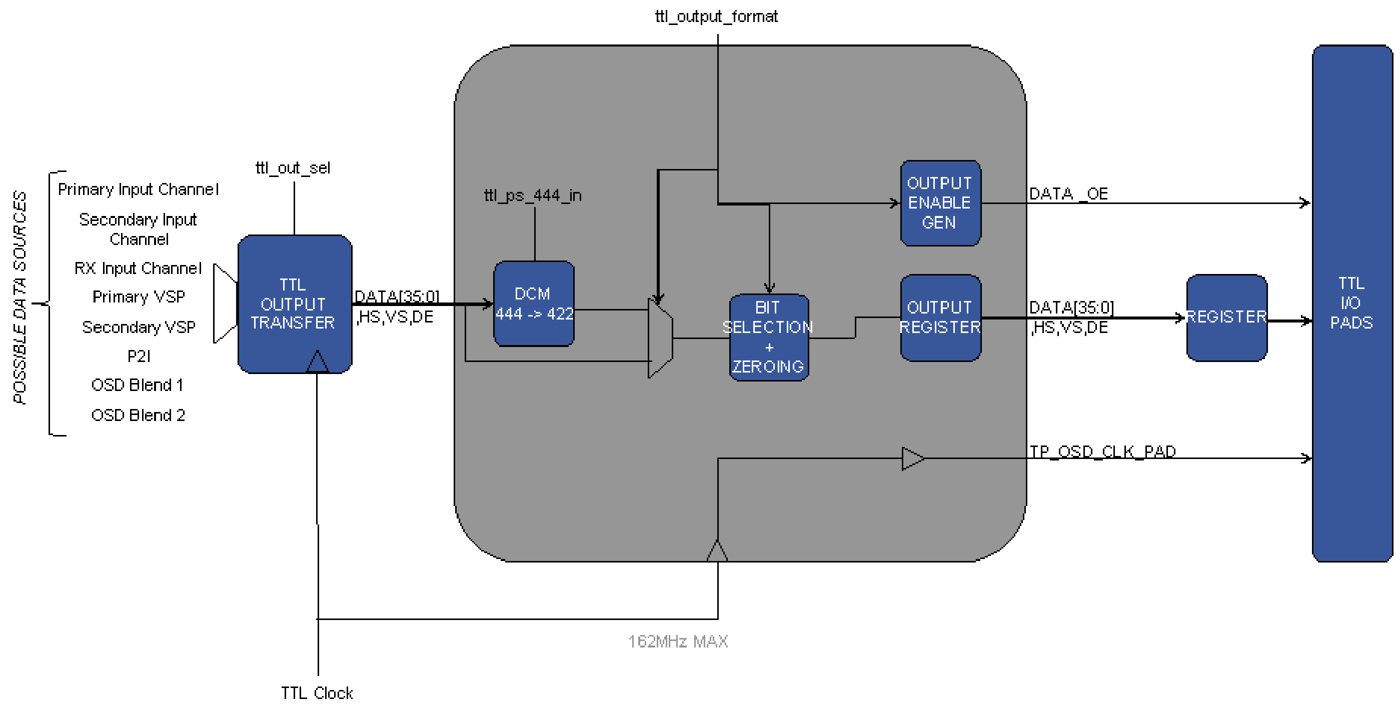


Figure 29: TTL Output Block Diagram

The following registers are used to control the TTL outputs.

ttl_ps444_in, IO Map, *Address 0x1A01[0]*
This bit is used to select the video type sent to the TTL output format block.

Function

ttl_ps444_in	Description
0 (default)	Input to TTL output block is real 4:4:4
1	Input to TTL output block is pseudo 4:4:4

ttl_op_format[3:0], IO Map, *Address 0x1A02[7:4]*
This signal is used to specify the TTL output format.

Function

ttl_op_format[3:0]	Description
0011	2 x 8-bit buses, SDR 4:2:2
0100	2 x 10-bit buses, SDR 4:2:2
0101	2 x 12-bit buses, SDR 4:2:2
0110	3 x 8-bit buses, SDR 4:4:4
0111 (default)	3 x 10-bit buses, SDR 4:4:4
1000	3 x 12-bit buses, SDR 4:4:4

ttl_vid_out_en, IO Map, *Address 0x1A02[3]*
This bit is used to enable the TTL video output.

Function

ttl_vid_out_en	Description
0 (default)	Disable TTL output
1	Enable TTL output

ttl_out_sel[2:0], IO Map, *Address 0x1A02[2:0]*
This signal is used to select the video source for the TTL video output.

Function

ttl_out_sel[2:0]	Description
0x00 (default)	From Primary Input Channel
0x01	From Primary VSP
0x02	From PtoI Converter
0x03	From Internal OSD Blend 1
0x04	From Secondary VSP/PtoI Converter
0x05	From Secondary Input Channel
0x06	From RX Input
0x07	From Internal OSD Blend 2

osd_clk_drv_str[1:0], IO Map, *Address 0x1BA7[1:0]*
This signal is used to control the drive strength for the video output clock signal.

Function

osd_clk_drv_str[1:0]	Description
00 (default)	Minimum
01	Medium low (x2)
10	Medium high (x3)
11	Maximum (x4)

osd_dout_drv_str[1:0], IO Map, *Address 0x1BA3[1:0]*
This signal is used to control the drive strength for the video output data and sync signals.

Function

osd_dout_drv_str[1:0]	Description
00 (default)	Minimum
01	Medium low (x2)
10	Medium high (x3)
11	Maximum (x4)

2.2.2.4. *Treatment o f Unused TTL Inputs*

ADV8005 allows the TTL pins to be powered down when unused, removing the need for external pulldowns on many unused I/O pins.

Note: The TTL pins are powered down by default, each of these pins must be powered up to use them. Unused pins should be left powered down.

vid_clk_ie, IO Map, *Address 0x1BC8[5]*
This bit is used to control the input path enable for the VID CLK pin.

Function

vid_clk_ie	Description
0 (default)	input path disable
1	input path enable

clk_osd_ie, IO Map, *Address 0x1BC8[4]*
This bit is used to control the input path enable for the osd clk pin.

Function

clk_osd_ie	Description
0 (default)	input path disable
1	input path enable

pix_pins_ie[31:0], IO Map, *Address 0x1BC9[7:0]; Address 0x1BCA[7:0]; Address 0x1BCB[7:0]; Address 0x1BCC[7:0]*
This bit is used to control the input path enable for the pixel pins.

Function

pix_pins_ie[31:0]	Description
0 (default)	input path disable
1	input path enable

osd_pins_ie[23:0], IO Map, *Address 0x1BCD[7:0]; Address 0x1BCE[7:0]; Address 0x1BCF[7:0]*
This bit is used to control the input path enable for the osd pins.

Function

osd_pins_ie[23:0]	Description
0 (default)	input path disable
1	input path enable

hs_ie, IO Map, *Address 0x1BD0[7]*
This bit is used to control the input path enable for the HS pin.

Function

hs_ie	Description
0 (default)	input path disable
1	input path enable

vs_ie, IO Map, *Address 0x1BD0[6]*
This bit is used to control the input path enable for the VS pin.

Function

vs_ie	Description
0 (default)	input path disable
1	input path enable

de_ie, IO Map, *Address 0x1BD0[5]*
This bit is used to control the input path enable for the DE pin.

Function

de_ie	Description
0 (default)	input path disable
1	input path enable

sfl_ie, IO Map, *Address 0x1BD0[4]*
This bit is used to control the input path enable for the SFL pin.

Function

sfl_ie	Description
0 (default)	input path disable
1	input path enable

hs_osd_ie, IO Map, *Address 0x1BD0[2]*
This bit is used to control the input path enable for the osd HS pin.

Function

hs_osd_ie	Description
0 (default)	input path disable
1	input path enable

de_osd_ie, IO Map, *Address 0x1BD0[0]*
This bit is used to control the input path enable for the osd DE pin.

Function

de_osd_ie	Description
0 (default)	input path disable
1	input path enable

audio_pins_ie[6:0], IO Map, *Address 0x1BD1[6:0]*
This bit is used to control the input path enable for the audio pins.

Function

audio_pins_ie[6:0]	Description
0 (default)	input path disable
1	input path enable

arc1_pin_ie, IO Map, *Address 0x1BD2[7]*
This bit is used to control the input path enable for the ARC 1 pin.

Function

arc1_pin_ie	Description
0 (default)	input path disable
1	input path enable

arc2_pin_ie, IO Map, *Address 0x1BD2[6]*
This bit is used to control the input path enable for the ARC 2 pin.

Function

arc2_pin_ie	Description
0 (default)	input path disable
1	input path enable

int_pin_ie[**2:0**], IO Map, *Address 0x1BD2[5:3]*
This bit is used to control the input path enable for the INT pins.

Function

int_pin_ie[2:0]	Description
0 (default)	input path disable
1	input path enable

sclk_ie, IO Map, *Address 0x1BD2[2]*
This bit is used to control the input path enable for the audio SCLK pin.

Function

sclk_ie	Description
0 (default)	input path disable
1	input path enable

mclk_ie, IO Map, *Address 0x1BD2[1]*
This bit is used to control the input path enable for the audio MCLK pin.

Function

mclk_ie	Description
0 (default)	input path disable
1	input path enable

dsd_clk_ie, IO Map, *Address 0x1BD2[0]*
This bit is used to control the input path enable for the audio DSD CLK pin.

Function

dsd_clk_ie	Description
0 (default)	input path disable
1	input path enable

spi1_cs_ie, IO Map, *Address 0x1BD3[7]*
This bit is used to control the input path enable for the spi1 CS pin.

Function

spi1_cs_ie	Description
0 (default)	input path disable
1	input path enable

spi1_miso_ie, IO Map, *Address 0x1BD3[6]*
This bit is used to control the input path enable for the spi1 MISO pin.

Function

spi1_miso_ie	Description
0 (default)	input path disable
1	input path enable

spi1_mosi_ie, IO Map, *Address 0x1BD3[5]*
This bit is used to control the input path enable for the spi1 MOSI pin.

Function

spi1_mosi_ie	Description
0 (default)	input path disable
1	input path enable

spi1_sclk_ie, IO Map, *Address 0x1BD3[4]*
This bit is used to control the input path enable for the spi1 SCLK pin.

Function

spi1_sclk_ie	Description
0 (default)	input path disable
1	input path enable

spi2_cs_ie, IO Map, *Address 0x1BD3[3]*
This bit is used to control the input path enable for the spi1 CS pin.

Function

spi2_cs_ie	Description
0 (default)	input path disable
1	input path enable

spi2_miso_ie, IO Map, *Address 0x1BD3[2]*
This bit is used to control the input path enable for the spi2 ,OSP pin.

Function

spi2_miso_ie	Description
0 (default)	input path disable
1	input path enable

spi2_mosi_ie, IO Map, *Address 0x1BD3[1]*
This bit is used to control the input path enable for the spi2 MOSI pin.

Function

spi2_mosi_ie	Description
0 (default)	input path disable
1	input path enable

spi2_sclk_ie, IO Map, *Address 0x1BD3[0]*
This bit is used to control the input path enable for the spi2 SCLK pin.

Function

spi2_sclk_ie	Description
0 (default)	input path disable
1	input path enable

mas_clk_ie, IO Map, *Address 0x1BD4[2]*
This bit is used to control the input path enable for the master CLK pin.

Function

mas_clk_ie	Description
0 (default)	input path disable
1	input path enable

mas_hs_ie, IO Map, *Address 0x1BD4[1]*
This bit is used to control the input path enable for the master HS pin.

Function

mas_hs_ie	Description
0 (default)	input path disable
1	input path enable

mas_vs_ie, IO Map, *Address 0x1BD4[0]*
This bit is used to control the input path enable for the master VS pin.

Function

mas_vs_ie	**Description**
0 (default)	input path disable
1	input path enable

2.2.2.5. Serial Video Rx

The Serial Video Rx can only be connected to the RX input channel (see Section 2.2.2.8).

2.2.2.6. Primary Input Channel

The ADV8005 primary input channel incorporates an input formatter, CSC, updither block and ACE control.

The input formatter provides a number of controls to configure what data the video TTL input channel is configured for. The video TTL input channel must be connected to either the video TTL input pins, the EXOSD TTL input pins or the high speed TTL input pins using p_inp_chan_sel[1:0]. If the primary input channel is connected to the video TTL input pins, the format and bit width of the data, for example, 2 x 8 bit buses of 4:2:2 data, must be specified using vid_format_sel[4:0]. vid_swap_bus_ctrl[2:0] can be used to indicate which input pins are used to carry the upper, middle and lower ranges of bits (for example, upper = D[35:25], middle = D[24:12], lower = D[11:0] or upper = D[11:0], middle = D[35:25], lower = D[24:12]).

p_inp_chan_sel[1:0], IO Map, *Address 0x1A07[1:0]*
This signal is used to select the input for the Primary Input Channel.

Function

p_inp_chan_sel[1:0]	**Description**
00 (default)	Video TTL input (P[35:0])
01	EXOSD TTL Input (OSD_IN[23:0])
10	48-bit TTL input (OSD_IN[11:0] and P[35:0]) for 3GHz interleaved TTL
11	Reserved

vid_format_sel[4:0], IO Map, *Address 0x1B48[4:0]*
This signal is used to select the input format for the video data.

Function

vid_format_sel[4:0]	**Description**
0x00	1 x 8-bit bus, SDR 4:2:2
0x01	1 x 10-bit bus, SDR 4:2:2
0x02	1 x 12-bit bus, SDR 4:2:2
0x03	2 x 8-bit buses, SDR 4:2:2
0x04	2 x 10-bit buses, SDR 4:2:2
0x05	2 x 12-bit buses, SDR 4:2:2
0x06	3 x 8-bit buses, SDR 4:4:4 (P[35:28], P[23:16], P[11:4])
0x07	3 x 10-bit buses, SDR 4:4:4 (P[35:26], P[23:14], P[11:2])
0x08 (default)	3 x 12-bit buses, SDR 4:4:4
0x09	1 x 8-bit bus, DDR 4:2:2
0x0A	1 x 10-bit bus DDR 4:2:2
0x0B	1 x 12 bit bus, DDR 4:2:2
0x0C	3 x 8 bit buses, SDR 4:4:4 (P[23:0])
0x0D	2 x 3 x 8-bit interleaved buses, SDR 4:4:4
0x0E	2 x 2 x 8-bit interleaved buses, SDR 4:2:2
0x0F	2 x 2 x 10-bit interleaved buses, SDR 4:2:2
0x10	2 x 2 x 12-bit interleaved buses, SDR 4:2:2
0x11	3 x 10-bit buses, SDR 4:4:4 (P[29:0])
0x12	3 x 7-bit buses, SDR 4:4:4 (for external alpha blend)
0x13	3 x 10-bit buses, SDR 4:4:4 (OSD_IN[23:0] and P[35:30])

vid_swap_bus_ctrl[2:0], IO Map, *Address 0x1B48[7:5]*
This signal is used to control the video input pixel bus. The input pixel bus is 36 bits wide and is divided into three data channels: Top = D[35:24], Middle = D[23:12] and Bottom = D[11:0]. This register allows the user to swap the order of these three data channels.

Function

vid_swap_bus_ctrl[2:0]	Description
000 (default)	D[35:24] D[23:12] D[11:0]
001	D[35:24] D[11:0] D[23:12]
010	D[35:24] D[23:12] D[11:0]
011	D[23:12] D[35:24] D[11:0]
100	D[11:0] D[35:24] D[23:12]
101	D[11:0] D[23:12] D[35:24]
110	D[23:12] D[11:0] D[35:24]
111	D[35:24] D[23:12] D[11:0]

The input formatter also has a number of controls which can be used to provide extra flexibility in terms of data processing.

Once a DDR mode is selected using vid_format_sel[4:0], the order of the luma and chroma data can be configured using vid_ddr_yc_swap. In DDR modes, the luma is expected on the rising edge of the pixel clock. Setting this bit to 1 swaps the luma and chroma samples and places the chroma sample (C) on the rising edge and the luma sample (Y) on the falling edge. Refer to Figure 30 for more information. The edge on which each sample of DDR data is latched into the part can be specified using vid_ddr_edge_sel.

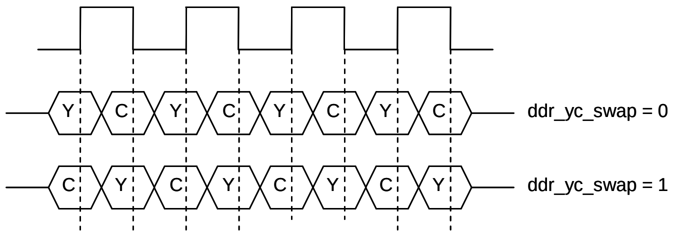


Figure 30: DDR Mode, Luma and Chroma Swap

vid_ddr_yc_swap, IO Map, *Address 0x1B4A[0]*
This bit is used to swap the Luma (Y) and Chroma (C) data in DDR modes. By default, Y is expected on the rising edge of the clock.

Function

vid_ddr_yc_swap	Description
0 (default)	Y on rising edge of clock
1	C on rising edge of clock

vid_ddr_edge_sel, IO Map, *Address 0x1B4A[3]*
This bit is used to select which edge the first sample of DDR data is latched on.

Function

vid_ddr_edge_sel	Description
0 (default)	Posedge data first
1	Negedge data first

Using the pixel clock as a reference, ADV8005 expects the Y sample on a rising edge and then a chroma sample on the falling edge. When vid_ddr_yc_swap is set, ADV8005 expects a chroma sample on the rising edge and the Y sample on the falling edge. vid_swap_cb_cr_422 can be used to swap the order of the chroma data. By default, ADV8005 expects a sequence of Cb, Cr, Cb, Cr… When vid_swap_cb_cr_422 is set, ADV8005 expects a sequence of Cr, Cb, Cr, Cb....

vid_swap_cb_cr_422, IO Map, *Address 0x1B49[7]*
This bit is used to swap the order of the C data when decoding 4:2:2 data.

Function

vid_swap_cb_cr_422	Description
0 (default)	Cb/Cr decoding
1	Cr/Cb decoding

vid_ps444_r444_conv is used to convert from pseudo 444 video data to real 444. All processing occurs in the ADV8005 in 4:4:4 mode. Therefore, if video input to the device is not in this format, this must be first converted to 4:4:4. Setting this bit to 1 converts video data to 4:4:4.

vid_ps444_r444_conv, IO Map, *Address 0x1B49[6]*
This bit is used to convert 4:2:2 data to pseudo 444 or to real 444.

Function

vid_ps444_r444_conv	**Description**
0 (default)	Nothing done
1	Pseudo 444 to Real 444 conversion

vid_hs_pol, vid_vs_pol, vid_de_pol and vid_fld_pol configure the polarity of the input video timing signals. These must be set depending on the polarity of the upstream IC. If active low, these register can be left at their default. If these signals from the upstream IC are active high, their polarity can be inverted.

vid_hs_pol, IO Map, *Address 0x1B49[3]*
This bit is used to set the polarity of the input HS timing signal.

Function

vid_hs_pol	**Description**
0 (default)	Input HS polarity does not change
1	Input HS polarity gets inverted

vid_vs_pol, IO Map, *Address 0x1B49[2]*
This bit is used to set the polarity of the input VS timing signal.

Function

vid_vs_pol	**Description**
0 (default)	Input VS polarity does not change
1	Input VS polarity gets inverted

vid_de_pol, IO Map, *Address 0x1B49[1]*
This bit is used to set the polarity of the input DE enable signal.

Function

vid_de_pol	**Description**
0 (default)	Input DE polarity does not change
1	Input DE polarity gets inverted

vid_fld_pol, IO Map, *Address 0x1B49[0]*
This bit is used to set the polarity of the input Field (FLD) timing signal.

Function

vid_fld_pol	**Description**
0 (default)	Input FLD polarity does not change
1	Input FLD polarity gets inverted

vid_hs_vs_mode is used to select the method by which the input video will be synchronized. This may be required when the ADV8005 is used in conjunction with an MPEG decoder. MPEG decoders use embedded timing codes rather than using external HS and VS signals. Similarly, other ADI decoders/HDMI Rxs can output video using embedded timing codes. This register should be programmed depending on the timing method of the upstream IC.
Refer to Section 2.2.11 for more information on AV-codes.

vid_hs_vs_mode, IO Map, *Address 0x1B4B[7]*
This bit is used to select the method of input timing.

Function

vid_hs_vs_mode	**Description**
0	Use embedded SAV/EAV codes
1 (default)	Use external HS/VS synchronization signals

vid_av_pos_sel, IO Map, *Address 0x1B4B[3]*
This bit is used to select if the HS generated is consistent with EIA 861 timing or dependant on the embedded timing codes.

Function

vid_av_pos_sel	**Description**
0 (default)	Generate HS coincident with EAV code
1	Generate HS/VS based on 861 timing

vid_av_split_code, IO Map, *Address 0x1B4B[2]*
This bit is used to control how AV codes are decoded - replicated on or split across all channels.

Function

vid_av_split_code	**Description**
0 (default)	Decodes AV codes which are replicated on all channels
1	Decodes AV codes which are split across all channels

vid_av_codes_rep_man_en, IO Map, *Address 0x1B4B[1]*
This bit is used to control the enable for AV source codes. AV_codes_rep_man is used instead of the auto based on the input video format.

Function

vid_av_codes_rep_man_en	**Description**
0 (default)	AV codes replicated based on internal flag
1	Use i2c bit

vid_av_codes_rep_man, IO Map, *Address 0x1B4B[0]*
This bit is used to specify if the AV_codes are replicated or not.
Codes replicated (4:4:4) = FF,FF,FF,00,00,00,00,00, 00,AV,AV,AV.
Codes not replicated = FF,00,00,AV.

Function

vid_av_codes_rep_man	**Description**
1	AV codes are replicated.
0 (default)	AV codes are not replicated.

The updither feature in the ADV8005 can be used to randomize quantization errors, preventing large scale patterns such as color banding in images. Refer to Section 2.2.3 for more information on the updither block.

The updither block on the video TTL input channel can be controlled via the vid_ud_bypass_man_en and vid_ud_bypass_man bits. By default, the manual bypass is disabled which means that the updither block cannot be bypassed. The updither block configuration is outlined in Section 2.2.3. The updither settings are shared for all channels (primary, secondary and RX).

vid_ud_bypass_man_en, IO Map, *Address 0x1B4A[2]*
This bit is used to enable the manual bypass for the up dither. Setting this bit enables the bypass to be used.

Function

vid_ud_bypass_man_en	**Description**
0 (default)	Manual bypass disable
1	Manual bypass enable

vid_ud_bypass_man, IO Map, *Address 0x1B4A[1]*
This bit is used to bypass the up dither block.

Function

vid_ud_bypass_man	**Description**
0 (default)	Disable bypass
1	Enable bypass

The primary input path features contrast, brightness and saturation controls. All contrast, brightness and saturation controls (contrast[9:0], brightness[7:0], saturation[7:0], blank_level_y[11:0], blank_level_u[11:0] and blank_level_v[11:0]) are doubled buffered on VSync.

The contrast[9:0] value has a range of 0 to 1.992. Refer to Figure 31 for more information on how the contrast controls influence the video signal.

The brightness[7:0] value has a range of -1024 to 1016. Refer to Figure 32 for more information on how the brightness controls influence the video signal.

The saturation[7:0] value has a range 0 to 1.992. Refer to Figure 33 for more information on how the saturation controls influence the video signal.

contrast[9:0], Encoder Map, *Address 0xE49D[7:0]; Address 0xE49C[1:0]*
This signal is used to set the SD Y scale value.

blank_level_y[11:0], IO Map, *Address 0x1A24[3:0]; Address 0x1A25[7:0]*
This signal is used to adjust the blank level of y input to the vid adjust block.

Function

blank_level_y[11:0]	**Description**
0x000	y blank level sits at code 0
0x100 (default)	y blank level sits at code 256 decimal

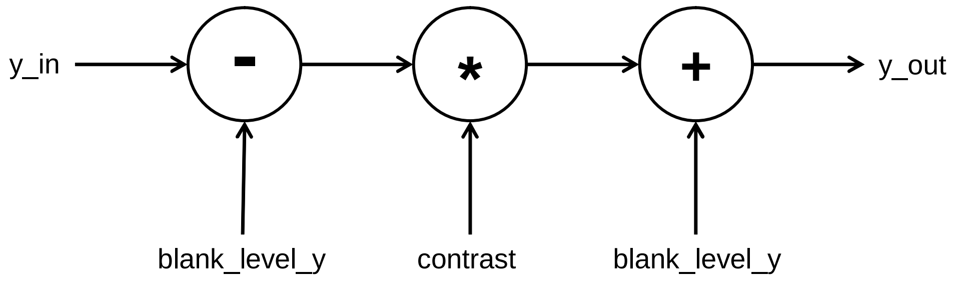


Figure 31: Contrast Processing

brightness[7:0], IO Map, *Address 0x1A2A[7:0]*
This register is used to adjust the brightness value for Y channel. The register uses s1.6 notation.

Function

brightness[7:0]	**Description**
0x7F	(+127) * 8
0x00 (default)	(No adjustment) * 8
0xFF	(-1) * 8

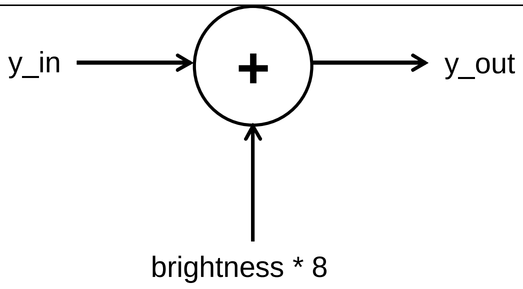


Figure 32: Brightness Processing

saturation[7:0], IO Map, *Address 0x1A29[7:0]*
This register is used to adjust the saturation value for U/V channels. The register uses 1.7 notation.

Function

saturation[7:0]	**Description**
0x00	Gain of 0
0x80 (default)	Unity Gain
0xFF	Gain of 2

blank_level_u[11:0], IO Map, *Address 0x1A26[7:0]; Address 0x1A27[7:4]*
This signal is used to adjust the blank level of u input to the vid adjust block.

Function

blank_level_u[11:0]	**Description**
0x000	u blank level sits at code 0
0x800 (default)	u blank level sits at code 2048 decimal

blank_level_v[11:0], IO Map, *Address 0x1A27[3:0]; Address 0x1A28[7:0]*
This signal is used to adjust the blank level of v input to the vid adjust block

Function

blank_level_v[11:0]	**Description**
0x000	v blank level sits at code 0
0x800 (default)	v blank level sits at code 2048 decimal

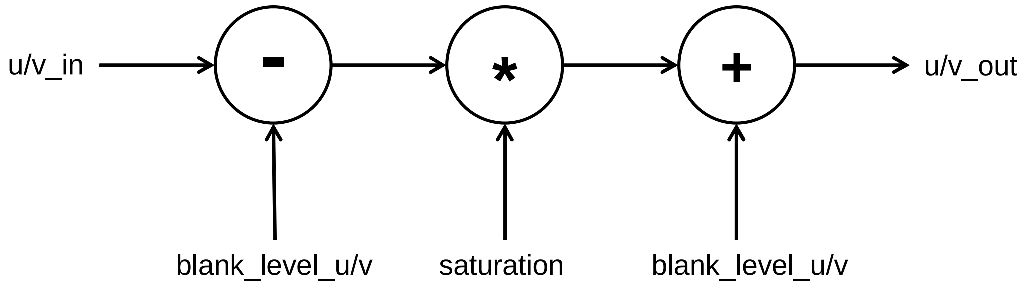


Figure 33: Saturation Processing

Refer to Section 2.2.12.1 for more information on the CSC controls for the primary input channel.

Refer to Section 3.2.3.16 for more information on the ACE controls for the primary input channel.

2.2.2.7. Secondary Input Channel

The ADV8005 secondary input channel incorporates an input formatter, CSC and updither block.

The input formatter provides a number of controls to configure what data the secondary input channel is configured for. The secondary input channel must be connected to either the video TTL input pins or the EXOSD TTL input pins using s_inp_chan_sel[1:0]. If the secondary input channel is connected to the video TTL input pins, the format and bit width of the data, for example, 2 x 8 bit buses of 4:2:2 data, must be specified using exosd_format_sel[4:0]. exosd_swap_bus_ctrl[2:0] can be used to indicate which input pins are used to carry the upper, middle and lower ranges of bits (for example, upper = D[35:25], middle = D[24:12], lower = D[11:0]; or upper = D[11:0], middle = D[35:25], lower = D[24:12]).

s_inp_chan_sel[1:0], IO Map, *Address 0x1A07[3:2]*
This signal is used to select the input for the Secondary Input Channel.

Function

s_inp_chan_sel[1:0]	**Description**
00	Video TTL input (P[35:0])
01 (default)	EXOSD TTL Input (OSD_IN[23:0])
10	RX video
11	N/A

exosd_format_sel[4:0], IO Map, *Address 0x1B68[4:0]*
This signal is used to select the input format for the video data.

Function

exosd_format_sel[4:0]	**Description**
0x00	1 x 8 bit bus 4:2:2
0x01	1 x 10 bit bus 4:2:2
0x02	1 x 12 bit bus 4:2:2
0x03	2 x 8 bit buses 4:2:2
0x04	2 x 10 bit buses 4:2:2
0x05	2 x 12 bit buses 4:2:2
0x06	3 x 8-bit buses, SDR 4:4:4
0x07	3 x 10-bit buses, SDR 4:4:4
0x08	3 x 12-bit buses, SDR 4:4:4
0x09	1 x 8 bit DDR bus 4:2:2
0x0A	1 x 10 bit DDR bus 4:2:2
0x0B	1 x 12 bit DDR bus 4:2:2
0x0C (default)	3 x 8 bit buses 4:4:4

exosd_swap_bus_ctrl[2:0], IO Map, *Address 0x1B68[7:5]*
This signal is used to control the external OSD input pixel bus. The input pixel bus is 24 bits wide and is divided into three data channels: Top = D[23:16], Middle = D[15:8] and Bottom = D[7:0]. This register allows the user to swap the order of these three data channels.

Function

exosd_swap_bus_ctrl[2:0]	**Description**
000 (default)	D[23:16] D[15:8] D[7:0]
001	D[23:16] D[7:0] D[15:8]
010	D[23:16] D[15:8] D[7:0]
011	D[15:8] D[23:16] D[7:0]
100	D[7:0] D[23:16] D[15:8]
101	D[7:0] D[15:8] D[23:16]
110	D[15:8] D[7:0] D[23:16]
111	D[23:16] D[15:8] D[7:0]

The input formatter also has a number of controls which can be used to provide extra flexibility in terms of data processing.

Once a DDR mode is selected using exosd_format_sel[4:0], the order of the luma and chroma data can be configured using exosd_ddr_yc_swap. In DDR modes, the luma is expected on the rising edge of the pixel clock. Setting this bit to 1 swaps the luma and chroma samples and places the chroma sample (C) on the rising edge and the luma sample (Y) on the falling edge. Refer to Figure 30 for more information. The edge on which each sample of DDR data is latched into the part can be specified using exosd_ddr_edge_sel.

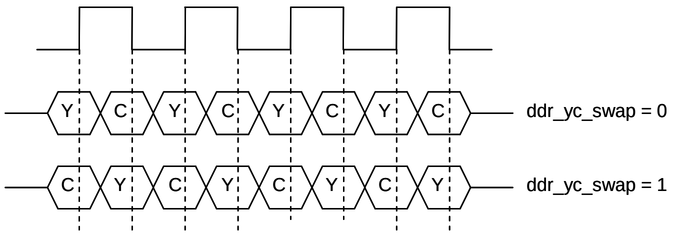


Figure 34: DDR Mode, Luma and Chroma Swap

exosd_ddr_yc_swap, IO Map, *Address 0x1B6A[0]*
This bit is used to swap the Luma (Y) and Chroma (C) data in DDR modes. By default, Y is expected on the rising edge of the clock.

Function

exosd_ddr_yc_swap	**Description**
0 (default)	Y on rising edge of clock
1	C on rising edge of clock

exosd_ddr_edge_sel, IO Map, *Address 0x1B6A[3]*
This bit is used to select which edge the first sample of DDR data is latched on.

Function

exosd_ddr_edge_sel	Description
0 (default)	Posedge data first
1	Negedge data first

Using the pixel clock as a reference, ADV8005 expects the Y sample on a rising edge and then a chroma sample on the falling edge. When exosd_ddr_yc_swap is set, ADV8005 expects a chroma sample on the rising edge and the Y sample on the falling edge. exosd_swap_cb_cr_422 can be used to swap the order of the chroma data. By default, ADV8005 expects a sequence of Cb, Cr, Cb, Cr… When exosd_swap_cb_cr_422 is set, ADV8005 expects a sequence of Cr, Cb, Cr, Cb....

exosd_swap_cb_cr_422, IO Map, *Address 0x1B69[7]*
This bit is used to swap the order of the C data when decoding 4:2:2 data.

Function

exosd_swap_cb_cr_422	Description
0 (default)	Cb/Cr decoding
1	Cr/Cb decoding

exosd_ps444_r444_conv is used to convert from pseudo 444 video data to real 444. All processing occurs in the ADV8005 in 4:4:4 mode. Therefore, if video input to the device is not in this format, it must be first converted to 4:4:4. Setting this bit to 1 converts video data to 4:4:4.

exosd_ps444_r444_conv, IO Map, *Address 0x1B69[6]*
This bit is used to convert 4:2:2 data to pseudo 444 or to real 444.

Function

exosd_ps444_r444_conv	Description
0 (default)	Nothing done.
1	Pseudo444 to Real 444 conversion.

exosd_rev_bus is used to reverse the order of the video TTL input. By default, this is set to non reversed.

exosd_rev_bus, IO Map, *Address 0x1B6B[4]*
This bit is used to reverse the input video bus, i.e. D[23:0] -> D[0:23].

Function

exosd_rev_bus	Description
0 (default)	Reverse the pin mapping on the OSD bus
1	Use the OSD bus as it comes from the pins

exosd_hs_pol, exosd_vs_pol and exosd_de_pol configure the polarity of the input video timing signals. These must be set depending on the polarity of the upstream IC. If active low, these register can be left at their default. If these signals from the upstream IC are active high, their polarity can be inverted.

exosd_hs_pol, IO Map, *Address 0x1B69[0]*
This bit is used to set the polarity of the input External OSD HS timing signal.

Function

exosd_hs_pol	Description
0 (default)	Input HS polarity doesn't change.
1	Input HS polarity gets inverted.

exosd_vs_pol, IO Map, *Address 0x1B69[1]*
This bit is used to set the polarity of the input External OSD VS timing signal.

Function

exosd_vs_pol	Description
0 (default)	Input VS polarity doesn't change.
1	Input VS polarity gets inverted.

exosd_de_pol, IO Map, *Address 0x1B69[2]*
This bit is used to set the polarity of the input External OSD DE timing signal.

Function

exosd_de_pol	**Description**
0 (default)	Input DE polarity doesn't change.
1	Input DE polarity gets inverted.

exosd_hs_vs_mode is used to select the method by which the input video will be synchronized. This may be required when the ADV8005 is used in conjunction with an MPEG decoder. MPEG decoders use embedded timing codes rather than using external HS and VS signals. Similarly, other ADI decoders/HDMI Rxs can output video using embedded timing codes. This register should be programmed depending on the timing method of the upstream IC.

Refer to Section 2.2.11 for more information on AV-codes.

exosd_hs_vs_mode, IO Map, *Address 0x1B6B[7]*
This bit is used to select the method of input timing.

Function

exosd_hs_vs_mode	**Description**
0	Embedded timing codes
1 (default)	VS/DE mode

exosd_av_pos_sel, IO Map, *Address 0x1B6B[3]*
This bit is used to select if the HS generated is consistent with EIA 861 timing or dependant on the embedded timing codes.

Function

exosd_av_pos_sel	**Description**
0 (default)	Generate hs coincident with eav code
1	Generate hs/vs based on 861 timing

exosd_av_split_code, IO Map, *Address 0x1B6B[2]*
This bit is used to control how AV codes are decoded - replicated on or split across all channels.

Function

exosd_av_split_code	**Description**
0 (default)	Replicated av codes on all channels
1	AV codes split across all buses

exosd_av_codes_rep_man_en, IO Map, *Address 0x1B6B[1]*
This bit is used to control the enable for AV source codes. AV_codes_rep_man is used instead of the auto based on the input video format.

Function

exosd_av_codes_rep_man_en	**Description**
0 (default)	AV codes replicated based on internal flag
1	Use i2c bit

exosd_av_codes_rep_man, IO Map, *Address 0x1B6B[0]*
This bit is used to specify if the AV_codes are replicated or not.
Codes replicated (4:4:4) = FF,FF,FF,00,00,00,00,00, 00,AV,AV,AV.
Codes not replicated = FF,00,00,AV.

Function

exosd_av_codes_rep_man	**Description**
1	AV codes are replicated.
0 (default)	AV codes are not replicated.

The updither feature in the ADV8005 can be used to randomize quantization error preventing large scale patterns such as color banding in images. Refer to Section 2.2.3 for more information on the updither block.

The updither block on the secondary input channel can be controlled via the exosd_ud_bypass_man and exosd_ud_bypass_man_en bits. By default, the manual bypass is disabled which means that the updither block cannot be bypassed. The updither block configuration is outlined in Section 2.2.3. The updither settings are shared for all channels (primary, secondary and RX).

exosd_ud_bypass_man_en, IO Map, *Address 0x1B6A[2]*
This bit is used to enable the manual bypass for the up dither. Setting this bit enables the bypass to be used.

Function

exosd_ud_bypass_man_en	Description
0 (default)	Manual bypass disable
1	Manual bypass enable

exosd_ud_bypass_man, IO Map, *Address 0x1B6A[1]*
This bit is used to bypass the up dither block.

Function

exosd_ud_bypass_man	Description
0 (default)	Disable bypass
1	Enable bypass

Refer to Section 2.2.12.2 for more information on the CSC controls for the secondary input channel.

2.2.2.8. RX Input Channel

The ADV8005 RX input channel incorporates an input formatter, CSC and updither block.

The updither feature in the ADV8005 can be used to randomize quantization error preventing large scale patterns such as color banding in images. Refer to Section 2.2.3 for more information on the updither block.

The updither block on the RX input channel can be controlled via the rx_ud_bypass_man_en and rx_ud_bypass_man bits. By default, the manual bypass is disabled which means that the updither block cannot be bypassed. The updither block configuration is outlined in Section 2.2.3. The updither settings are shared for all channels (primary, secondary and RX).

rx_ud_bypass_man_en, IO Map, *Address 0x1B8A[2]*
This bit is used to enable the manual bypass for the up dither. Setting this bit enables the bypass to be used.

Function

rx_ud_bypass_man_en	Description
0 (default)	Manual bypass disable
1	Manual bypass enable

rx_ud_bypass_man, IO Map, *Address 0x1B8A[1]*
This bit is used to bypass the up dither block.

Function

rx_ud_bypass_man	Description
0 (default)	Disable bypass
1	Enable bypass

rx_swap_bus_ctrl[2:0], IO Map, *Address 0x1B88[7:5]*
This signal is used to configure the order of the input video bus.

Function

rx_swap_bus_ctrl[2:0]	Description
000 (default)	D[35:24] D[23:12] D[11:0]
001	D[35:24] D[11:0] D[23:12]
010	D[35:24] D[23:12] D[11:0]
011	D[23:12] D[35:24] D[11:0]
100	D[11:0] D[35:24] D[23:12]
101	D[11:0] D[23:12] D[35:24]
110	D[23:12] D[11:0] D[35:24]
111	D[35:24] D[23:12] D[11:0]

Refer to Section 2.2.12.3 for more information on the CSC controls for the RX input channel.

2.2.3. *Updither Configuration*

The updither block on each of the input channels can be used to increase the bit width of the incoming video. This is useful if the output video must be a certain bit depth and the input video is below this level. Updither can increase color richness and reduce the effects of quantization, rounding and truncation which may have been induced on the video data. The updither block can be used in a situation where the video input to the ADV8005 is in 8-bit form and must be converted to 10-bit or 12-bit for output.

The operation of the updither block can be seen in Figure 35. When converting to a higher bit width, the ADV8005 updither block first converts to a bit width of 14 and then down converts to 12- and 10-bit width.

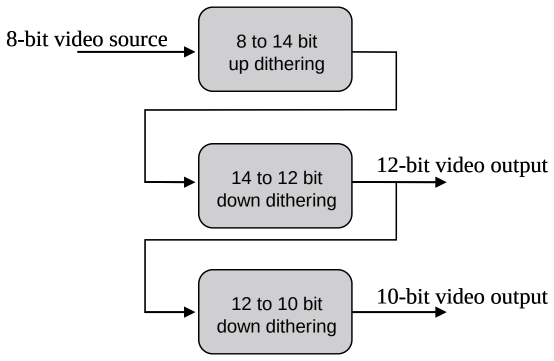


Figure 35: Updither Operation

updither_level[1:0] is used to configure the updither algorithm level. This should be configured depending on the input and output from the block. For example, if the input video is 8-bit data and the output is 12-bit data, this should be set to the highest level.

updither_level[1:0], IO Map, *Address 0x1A0D[5:4]*
This signal is used to set the sharpness of the updither block's HPF processing of the video data. When this signal is set to low the characteristic of the dither block's HPF gives smoother output video. When this signal is set to high, the characteristic of the dither block's HPF gives sharper output video.

Function

updither_level[1:0]	**Description**
00	Low updither
11	High updither

2.2.4. *Clock Configuration*

This section describes the method of configuring the various clocks of the ADV8005 using the automatic controls video_in_id[7:0], exosd_in_id[7:0] and rx_in_id[7:0]. These controls can be employed to automatically configure the internal clocks for the following:

- Pixel de-repetition/front-end formatter clock configuration (main and secondary TTL channels and Serial Video Rx channel)
- Timing generation for inputs with AV-codes (main and secondary TTL channels only)
- ACE configuration (main TTL channel only)
- VBI ancillary data (main TTL channel only)

In any of these modes, the video_in_id[7:0], exosd_in_id[7:0] and rx_in_id[7:0] controls must be configured.

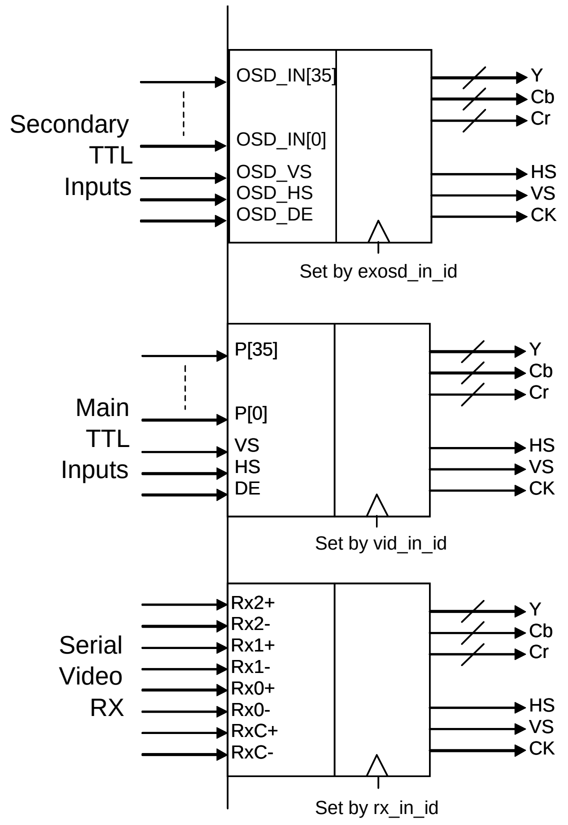


Figure 36: Configuring Input Port Clock

video_in_id[**7:0**], IO Map, *Address 0x1A00[7:0]*
This register is used to set the output clock frequencies from the input video formatting block used by both the Serial Video RX and Video TTL input ports.

Function

video_in_id[7:0]	**Description**
0x01	640x480p@60Hz
0x03	720x480p@60Hz
0x04	1280x720p@60Hz
0x05	1920x1080i@60Hz
0x07	720(1440)x480i@60Hz
0x09	720(1440)x240p@60Hz
0x0B	(2880)x480i@60Hz
0x0D	(2880)x240p@60Hz
0x0F	1440x480p@60Hz
0x10	1920x1080p@60Hz
0x12	720x576p@50Hz
0x13	1280x720p@50Hz
0x14	1920x1080i@50Hz
0x16	720(1440)x576i@50Hz
0x18	720(1440)x288p@50Hz
0x1A	(2880)x576i@50Hz
0x1C	(2880)x288p@50Hz
0x1E	1440x576p@50Hz
0x1F	1920x1080p@50Hz
0x20	1920x1080p@24Hz
0x21	1920x1080p@25Hz
0x22	1920x1080p@30Hz
0x24	2880x480p@60Hz
0x26	2880x576p@50Hz
0x80	640x350@85hz
0x81	640x400@85hz
0x82	720x400@85hz
0x83	640x480@60hz
0x84	640x480@72hz
0x85	640x480@75hz
0x86	640x480@85hz
0x87	800x600@56hz
0x88	800x600@60hz
0x89	800x600@72hz
0x8A	800x600@75hz
0x8B	800x600@85hz
0x8D	1024x768@60hz
0x8E	1024x768@70hz
0x8F	1024x768@75hz
0x90	1024x768@85hz
0xFC	720x288p@50Hz
0xFD	720x240p@60Hz
0xFE (default)	720x480i@60Hz
0xFF	720x576i@50Hz

exosd_in_id[7:0], IO Map, *Address 0x1B6C[7:0]*

This register is used to specify the video_id relative to CEA 861.

Function

exosd_in_id[7:0]	**Description**
0x01	CEA 861 VIC 1 (480p_60 640)
0x02	CEA 861 VIC 2 (480p_60)
0x03	CEA 861 VIC 3 (480p_60)
0x04	CEA 861 VIC 4 (720p_60)
0x05	CEA 861 VIC 5 (1080i_60)
0x06	CEA 861 VIC 6 (480i_60)
0x07	CEA 861 VIC 7 (480i_60)
0x08	CEA 861 VIC 8 (240p_60)
0x09	CEA 861 VIC 9 (240p_60)
0x10	CEA 861 VIC 16 (1080p_60)
0x11	CEA 861 VIC 17 (576p_50)
0x12	CEA 861 VIC 18 (576p_50)
0x13	CEA 861 VIC 19 (720p_50)
0x14	CEA 861 VIC 20 (1080i_50)
0x15	CEA 861 VIC 21 (576i_50)
0x16	CEA 861 VIC 22 (576i_50)
0x17	CEA 861 VIC 23 (288p_50)
0x18	CEA 861 VIC 24 (288p_50)
0x1F	CEA 861 VIC 31 (1080p_50)
0xFC	CEA 861 VIC 252 (288p_50)
0xFD	CEA 861 VIC 253 (240p_60)
0xFE (default)	CEA 861 VIC 254 (480i_60)
0xFF	CEA 861 VIC 255 (576i_50)

rx_in_id[7:0], IO Map, *Address 0x1B96[7:0]*

This register is used to specify the VIC relative to CEA 861.

Function

rx_in_id[7:0]	**Description**
0x06	CEA861 VIC 6 (480i60 2x)
0x07	CEA861 VIC 7 (480i60 2x)
0x08	CEA861 VIC 8 (240p60 2x)
0x09	CEA861 VIC 9 (240p60 2x)
0x15	CEA861 VIC 21 (576i50 2x)
0x16	CEA861 VIC 22 (576i50 2x)
0x17	CEA861 VIC 23 (288p50 2x)
0x18	CEA861 VIC 24 (288p50 2x)

The ADV8005 can output a large number of video formats including many common graphics resolutions. To enable the PVSP and SVSP cores to output these frequencies, the output timing clocks must first be programmed. The output clocks for both the PVSP and SVSP are shown in Figure 37.

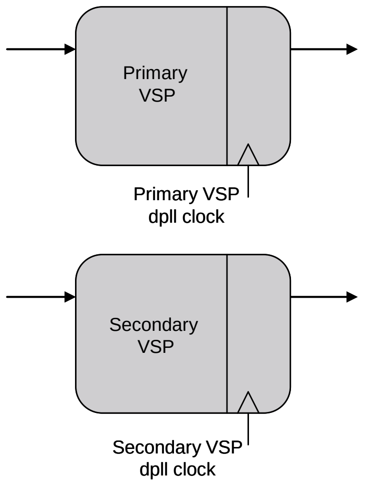


Figure 37: PVSP/SVSP Output Clock Configure

For the PVSP and SVSP, the correct clocks must be configured manually. This can be done using the DPLL period registers, which allows the user to program the sampling rate for the appropriate output format by I^2C. The equation for calculating this I^2C value is provided in Equation 1.

$$dpll\_phase\_period \equiv \frac{1}{64 \times 12 \times 27MHz}$$

Equation 1: Calculating DPLL Phase Period

Once the dpll_phase_period is calculated, Equation 2 is used to calculate the dpll_clock_period.

$$dpll\_clock\_period \equiv \frac{output\_clock\_period \times 2^{22}}{dpll\_phase\_period}$$

Equation 2: Calculating DPLL Clock Period

where output_clock_period is the period of the desired output sampling frequency.

For example, for HD video, the output clock sampling frequency would be 148.5 MHz. This equation returns a decimal value. Once calculated, this should be converted to hex and written to pvsp_vid_clk_period[33:0] and svsp_vid_clk_period[33:0]. Table 4 outlines some common resolutions and their associated dpll_clock_period values.

Table 4: Example Values for dpll_clock_period

Active Resolution	Frame Rate (Hz)	Sampling Frequency (MHz)	dpll_clock_period (Hex)
720 x 480i	29.97	13.5	0x180000000
720 x 480p	59.94	27	0x0C0000000
720 x 576i	25	13.5	0x180000000
720 x 576p	50	27	0x0C0000000
960 x 480i	29.97	18	0x120000000
960 x 576i	25	18	0x120000000
1280 x 720p	59.94	74.175	0x045E386DC
1280 x 720p	60	74.25	0x045D1745D
1920 x 1080i	29.97	74.175	0x045E386DC
1920 x 1080i	30	74.25	0x045D1745D
1920 x 1080p	59.94	148.35	0x022F1C36E
1920 x 1080p	60	148.5	0x022E8BA2F
1920 x 1080i	25	74.25	0x045D1745D
1920 x 1080p	50	148.5	0x022E8BA2F

Depending on the sampling frequency required, the following registers need to be programmed with this DPLL clock period.

Note: To enable the DPLL to configure the correct clocks for the ADV8005, register 0x0039 must be set to 0x0A. This register must always be configured before the following registers are set. This configures the ADV8005 clock generators to generate the clocks for the ADV8005.

2.2.4.1. PVSP Output Timing

The following registers are programmed for the PVSP.

pvsp_vid_clk_period[33:0], IO Map, *Address 0x1A3A[1:0]; Address 0x1A3B[7:0]; Address 0x1A3C[7:0]; Address 0x1A3D[7:0]; Address 0x1A3E[7:0]*
This register is used to set the open_loop_period of the DPLL section. This should be programmed based on the value calculated from the given equations.

pvsp_vid_clk_update, IO Map, *Address 0x1A3A[4]*
This bit is used to trigger the open loop period to be captured in the DPLL. A low to high transition triggers the action.

Function

pvsp_vid_clk_update	Description
0 (default)	Do not update open_loop_period in DPLL
1	Update open_loop_period in DPLL

For example, the following procedure updates the PVSP DPLL clock period:

1A 1A39 0A – Put the DPLL into ADV8005 (scaler) mode
1A 1A3B XX – Configure DPLL clock period setting
1A 1A3C XX – Configure DPLL clock period setting
1A 1A3D XX – Configure DPLL clock period setting
1A 1A3E XX – Configure DPLL clock period setting
1A 1A3A 80 – Recommended setting
1A 1A3A 90 – Recommended setting

Once configured, the clock in Figure 37 is programmed for operation.

2.2.4.2. SVSP Output Timing

The following registers are programmed for the SVSP.

svsp_vid_clk_period[33:0], IO Map, *Address 0x1A3F[1:0]; Address 0x1A40[7:0]; Address 0x1A41[7:0]; Address 0x1A42[7:0]; Address 0x1A43[7:0]*
This signal is used to set the open_loop_period of the DPLL section. This should be programmed based on the value calculated from the given equations.

svsp_vid_clk_update, IO Map, *Address 0x1A3F[4]*
This bit is used to trigger the open loop period to be captured in the DPLL. A low to high transition triggers the action.

Function

svsp_vid_clk_update	Description
0 (default)	Do not update open_loop_period in DPLL
1	Update open_loop_period in DPLL

For example, the following procedure for updating the SVSP DPLL clock period is very similar to that of the PVSP:

1A 1A39 0A – Put the DPLL into ADV8005 mode
1A 1A40 XX – Configure DPLL clock period setting
1A 1A41 XX – Configure DPLL clock period setting
1A 1A42 XX – Configure DPLL clock period setting
1A 1A43 XX – Configure DPLL clock period setting
1A 1A3F 80 – Recommended setting
1A 1A3F 90 – Recommended setting

Once configured, the clock in Figure 37 is programmed for operation.

2.2.4.3. Frame Tracking

The ADV8005 employs frame tracking on its scaler outputs. There will always be some error in the input frame rate versus the ideal frame rate. This could cause frame drops or repeats at the output. Frame tracking allows the output timing to track the input timing in such a way that eliminates frame drops and repeats while also remaining immune to discontinuities in the input. The system can be fully frequency and phase locked using . If phase locked is selected, there will be an integer frame latency from input to output. If frequency locked is selected, there could be a non integer frame latency number from input to the output. Selecting phase error latency is the recommended setting.

Frame tracking results in an integer ratio relationship between the input and output frame rates of 1:1, 2:1, 1:2, 5:2 or 2:5. For example, if scaling from 1080p30 to 720p59.94 with frame tracking enabled, the resulting output may be 720p60 due to the 1:2 relationship.

Frame rate tracking is primarily intended for cases where the input frame rate and output frame rate have a 1:1 relationship or are close to this target, that is, 59.94 Hz to 60 Hz. However, it can also be used for some standard frame rate conversion modes such as 24 Hz to 60 Hz, 25 Hz to 50 Hz, and 30 Hz to 60 Hz. The list of scaling conversions where frame tracking can be enabled is covered in Table 5.

Table 5: Frame Tracking

		Output Frame Rate							
Input Frame Rate		23.97 Hz	24 Hz	25 Hz	29.97 Hz	30 Hz	50 Hz	59.94 Hz	60 Hz
	23.97 Hz	Yes	Yes	No	No	No	No	Yes	Yes
	24 Hz	Yes	Yes	No	No	No	No	Yes	Yes
	25 Hz	No	No	Yes	No	No	Yes	No	No
	29.97 Hz	No	No	No	Yes	Yes	No	Yes	Yes
	30 Hz	No	No	No	Yes	Yes	No	Yes	Yes
	50 Hz	No	No	Yes	No	No	Yes	No	No
	59.94 Hz	Yes	Yes	No	Yes	Yes	No	Yes	Yes
	60 Hz	Yes	Yes	No	Yes	Yes	No	Yes	Yes

pvsp_track_en is set to enable frame tracking for the PVSP. svsp_track_en is set to enable frame tracking for the SVSP. If tracking is to be used in frame rate conversion mode, video_in_id[7:0], pvsp_autocfg_output_vid[7:0] (PVSP) and svsp_autocfg_output_vid[7:0] (SVSP) should also be set.

pvsp_track_en, IO Map, *Address 0x1A44[6]*
This bit is used to enable tracking of the frequency error to reduce the number of dropped/repeated frames for the Primary VSP.

Function

pvsp_track_en	**Description**
0 (default)	Do not adjust for frequency difference between input and output vertical sync
1	Adjust for frequency difference between input and output vertical sync

svsp_track_en, IO Map, *Address 0x1A44[2]*
This bit is used to enable tracking of the frequency error to reduce the number of dropped/repeated frames for the Secondary VSP.

Function

svsp_track_en	**Description**
0 (default)	Do not adjust for frequency difference between input and output vertical sync
1	Adjust for frequency difference between input and output vertical sync

pvsp_err_sel, IO Map, *Address 0x1A4E[3]*
This bit is used to choose between phase locked loop and frequency locked loop for the Primary VSP frame tracking mode.

Function

pvsp_err_sel	**Description**
0 (default)	Phase error
1	Frequency error

svsp_err_sel, IO Map, *Address 0x1A4F[3]*
This bit is used to choose between phase locked loop and frequency locked loop for the Secondary VSP frame tracking mode.

Function

svsp_err_sel	**Description**
0 (default)	Phase error
1	Frequency error

2.2.5. *DDR2 Interface*

The ADV8005 uses DDR2 memory to enable the de-interlacer, scaler and OSD features. The DDR2 interface on ADV8005 is designed to meet the JESD79-2F standard.

2.2.5.1. DDR2 Configuration

The controls described in this section are used to configure the ADV8005 DDR2 memory interface.

The first three bits configure the DDR2 memory interface for the external memory configuration. The sdram**_size[3:0]** sets the memory size of the attached memory or memories. For example, if using 256 Mb memory, sdram**_size[3:0]** should be set to 0001. If using 2 Gb memory, sdram**_size[3:0]** should be set to 0100.

The word_size[3:0] and
burst_length[2:0] fields must also be configured depending on whether there are single or multiple memories connected to the ADV8005. If there is a single DDR2 memory, word_size[3:0] and
burst_length[2:0] should be set for a 32-bit word size and bursts of 8. If there are dual DDR2 memories, word_size[3:0] and
burst_length[2:0] should be set for a 64-bit word size and bursts of 4.

ADV8005 is configured for dual 512 Mb memories with a 64-bit word size and bursts of 4.
sdram_size[3:0], IO Map, *Address 0x1A5B[7:4]*
This signal is used to specify the SDRAM size. All values other than those specified here are reserved.

Function

sdram_size[3:0]	Description
0001	individual SDRAM is 256Mbit
0010 (default)	individual SDRAM is 512Mbit
0011	individual SDRAM is 1Gbit
0100	individual SDRAM is 2Gbit

word_size[3:0], IO Map, *Address 0x1A5C[7:4]*
This signal is used to specify the word size on the user interface. The data width to the SDRAM is half of this value. All other values are reserved

Function

word_size[3:0]	Description
0010	32 bits
0011 (default)	64 bits

burst_length[2:0], IO Map, *Address 0x1A5D[1:0]; Address 0x1A5E[7]*
This signal is used to indicate the burst length of the read/write transaction.

Function

burst_length[2:0]	Description
010 (default)	Burst of 4
011	Burst of 8.

rw_ctrl_oe sets the direction for several of the pins on the DDR2 memory interface. By default, these pins are set to input. However, when set to 1, this bit enables these pins to be outputs. Likewise, when ddr2_ck_oe is set to 1, the DDR2 clock pin becomes an output.

rw_ctrl_oe, IO Map, *Address 0x1AA8[7]*
This bit is used to control the output enable for external memory read/write signals (ras, cas, clock, address…).

Function

rw_ctrl_oe	Description
0 (default)	Input
1	Output

ddr2_ck_oe, IO Map, *Address 0x1AA8[6]*
This bit is used to control the output enable for external memory clock signal.

Function

ddr2_ck_oe	Description
0 (default)	Input
1	Output

The PLL clock generator for the DDR2 memory interface can be set to a user defined frequency over the range of 200 to 250MHz by setting the plldll_sel_div[5:0] and the plldll_pre_div[1:0] I^2C controls.

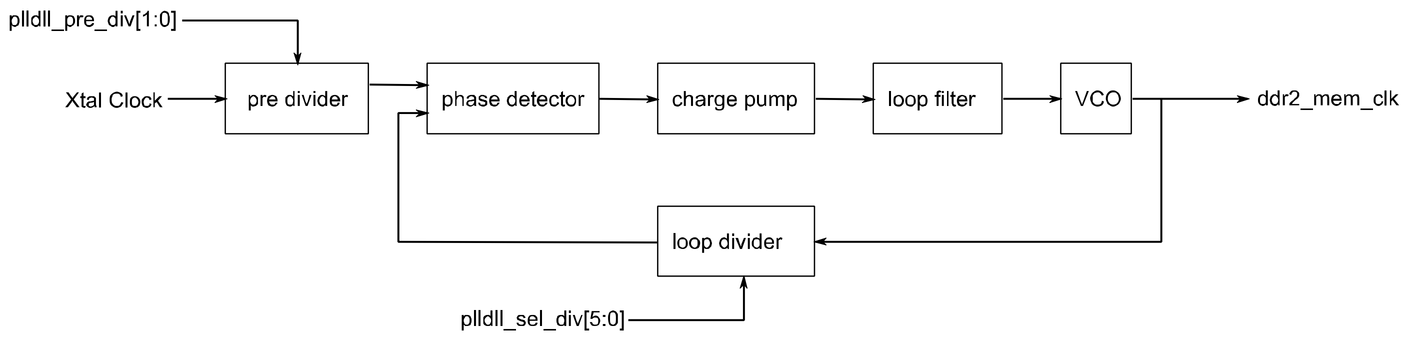


Figure 38: DDR2 PLL Architecture

Figure 38 shows the block diagram of the PLL with the relevant I²C controls. The formula used to determine the frequency of the DDR2 memory interface clock is given in Equation 3.

$$F_{ddr2\_clk} \equiv \frac{(F_{xtal\_clk})(plldll\_sel\_div)}{plldll\_pre\_div+1}$$

Equation 3: DDR2 Memory Interface Clock Frequency

The DDR2 clock frequency must not be changed during operation and should only be set prior to initialization of the memory interface.

plldll_sel_div[5:0], IO Map, *Address 0x1AA2[5:0]*
This signal is used to control the DDR2 PLL loop divider. The DDR2 clock frequency is given by: fxtal * i2c_plldll_sel_div / i2c_plldll_pre_div.

plldll_pre_div[1:0], IO Map, *Address 0x1AA3[3:2]*
This signal is used to control the DDR2 PLL pre divider.

2.2.5.2. DDR2 Bandwidth and Memory Selection

The DDR2 interface on ADV8005 can be configured to work with one or two (default) DDR2 memories. Using a single DDR2 memory limits the amount of functionality. Different capabilities are possible with different memory sizes. An outline of expected limitations are outlined in Table 7, Table 6 and Table 8.

Table 6: Indication of ADV8005 Capabilities with One DDR2 Memory

Features	**FRC**	**Motion Adaptive De-interlacing**	**Random Noise Reduction**	**OSD**	**Dual Output (ADV8005-8A/8N/8C only)**
SD input	Supported	Supported	Supported	Total area of all OSD regions (on screen at same time) must be < 2 * 720 * 480 pixels. (Entire OSD can be up-scaled to desired output resolution)	Supported
HD input (720p)	Supported	N/A	Not supported		Supported
HD input (1080i)	Supported	Intra-field interpolation not supported	Not supported		Supported
HD input (1080p)	Not supported - VSP_3D works in bypass mode	N/A	Not supported		Support only for: TX1 ->1080p; TX2 -> 480p/720p/1080p, as VSP_3D works in bypass mode

Table 7: Indication of ADV8005 Capabilities with Two DDR2 Memories

Features	**FRC**	**Motion Adaptive De-interlacing**	**Random Noise Reduction**	**OSD**	**Dual Output (ADV8005-8A/8N/8C only)**
SD input	Supported	Supported	Supported	Total area of all OSD regions (on screen at same time) must be < 3 * 720 * 480 pixels. (Entire OSD can be up-scaled to desired output resolution)	Supported
HD input (720p)	Supported	N/A	Supported		Supported
HD input (1080i)	Supported	Supported	Supported		Supported
HD input (1080p)	Supported	N/A	Only supported for 8- bit processing. Cannot be supported when OSD enabled.		Supported

Table 8: Indication of ADV8005 Capabilities with Different Memory Sizes

	1Gbx2	**512 Mbx2**	**1Gbx1**	**512 Mbx1**
FRC				
SD/ED input	Supported	Supported	Supported	Supported
HD input				
720P60/50	Supported	Supported	Supported	Supported
1080P60/50->1080P50/60@32/24/16bit	Supported	Supported	Not Supported	Not Supported
1080P60->1080P24@32/24bit	Supported	Supported	Not Supported	Not Supported
1080P60->1080P24@16bit	Supported	Supported	Supported	Supported
Motion Adaptive De-interlacing				
SD/ED input	Supported	Supported	Supported	Supported
HD input				
1080i60/50@32/24bit	Supported	Supported	Not Supported	Not Supported
1080i60/50@16bit*	Supported	Supported	Supported	Supported
Intra-field De-interlacing				
SD/ED input	Supported	Supported	Supported	Supported
HD input				
1080i60/50@32bit	Supported	Supported	Not Supported	Not Supported
1080i60/50@24/16bit	Supported	Supported	Supported	Supported
RNR				
SD/ED input	Supported	Supported	Supported	Supported
HD input				
720P60/50@32/24bit	Supported	Supported	Not Supported	Not Supported
720P60/50@16bit	Supported	Supported	Supported	Supported
1080i60/50@32/24/16bit	Supported	Supported	Not Supported	Not Supported
1080P60/50@32/24bit	Not Supported	Not Supported	Not Supported	Not Supported
1080P60/50@16bit	Supported	Supported	Not Supported	Not Supported
Game Mode				
SD/ED input	Supported	Supported	Supported	Supported
HD input	Supported	Supported	Supported	Supported
Memory left for OSD (Mbytes)	198.25	70.25	70.25	6.25

2.2.5.3. *Single DDR2 Memory Configuration*

If using a single DDR2 memory, the number of field buffers must be reduced from seven (default) to four when performing de-interlacing and scaling on 720p, 1080i and 1080p inputs. This is achieved by enabling intra field interpolation and setting (pvsp_ex_mem_data_format[1:0]) to indicate 16-bit 4:2:2. Next pvsp_frc_low_latency_mode must be enabled. Finally, the field buffers addresses in DDR2 must be reassigned as follows:

0xE800[31:0] (pvsp_fieldbuffer0_addr[31:0]) = 5184000
0xE804[31:0] (pvsp_fieldbuffer1_addr[31:0]) = 9331200
0xE808[31:0] (pvsp_fieldbuffer2_addr[31:0]) = 13478400
0xE80C[31:0] (pvsp_fieldbuffer3_addr[31:0]) = 17625600
0xE810[31:0] (pvsp_fieldbuffer4_addr[31:0]) = 21772800
0xE814[31:0] (pvsp_fieldbuffer5_addr[31:0]) = 25920000
0xE889[31:0] (pvsp_fieldbuffer6_addr[31:0]) = 27578880

2.2.5.4. *DDR2 Loopback Test*

The ADV8005 features a DDR2 loopback test block to allow testing of the ADV8005 DDR2 interface. When the loopback test block is enabled, it controls the commands sent to the DDR2 controller of the ADV8005 and generates pseudo random data and addresses using a defined protocol.

The controller first writes a programmable number of random 32-bit words to the external memory. The same number of reads are then performed from the written addresses. The readback is compared with the pseudo random data generated to check if there are any errors.

The results are available via I^{2}C readback.

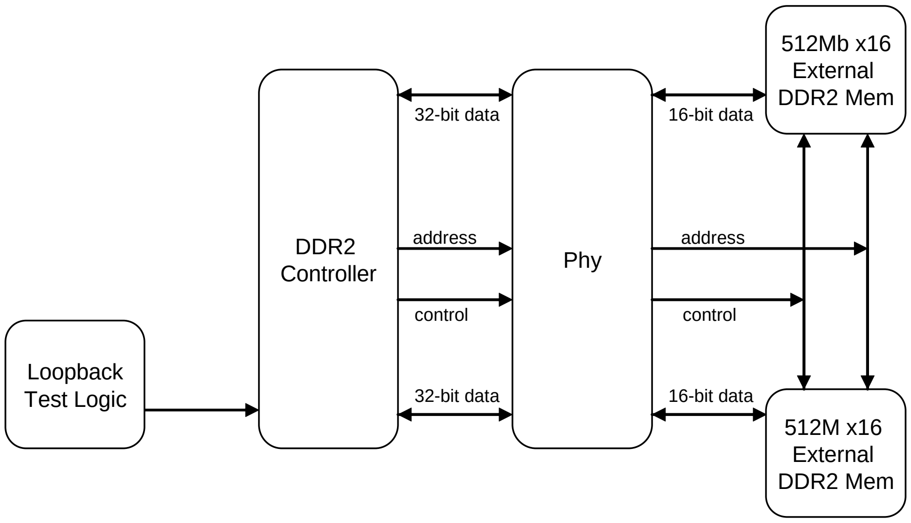


Figure 39: DDR2 Loopback Test Architecture

A two memory DDR2 loopback test is initialized and started via the following writes:

1A 1A5B 22 ; Recommended Write
1A 1A5F 00 ; Recommended Write
1A 1A61 06 ; Recommended Write
1A 1AA0 13 ; Recommended Write
1A 1AA1 01 ; Recommended Write
1A 1AA2 25 ; Recommended Write
1A 1AA3 1D ; Recommended Write
1A 1AA4 81 ; Recommended Write
1A 1AA5 81 ; Recommended Write
1A 1AA7 53 ; Recommended Write
1A 1AA8 B4 ; Recommended Write
1A 1AFE 08 ; Recommended Write
1A 1A0B 10 ; Recommended Write
1A E649 40 ; Recommended Write

A single memory DD2 loopback test is initialized and started via the following writes:

```
1A 1A5B 22 ; Recommended Write
1A 1A5C 20 ; Recommended Write
1A 1A5E 80 ; Recommended Write
1A 1A5F 00 ; Recommended Write
1A 1A61 06 ; Recommended Write
1A 1AA0 13 ; Recommended Write
1A 1AA1 01 ; Recommended Write
1A 1AA2 25 ; Recommended Write
1A 1AA3 1D ; Recommended Write
1A 1AA4 81 ; Recommended Write
1A 1AA5 81 ; Recommended Write
1A 1AA7 53 ; Recommended Write
1A 1AA8 B4 ; Recommended Write
1A 1AB2 02 ; Recommended Write
1A 1AFE 08 ; Recommended Write
```

The result of the DDR2 loopback test is given by the lbk_test_done and lbk_test_result bits.

lbk_test_done, IO Map, *Address 0x1AE1[0] (Read Only)*
This bit is used to readback the DDR2 loopback test has completed.

Function

lbk_test_done	Description
0 (default)	Test not complete
1	Loopback test finished

lbk_test_result, IO Map, *Address 0x1AE1[1] (Read Only)*
This bit is used to readback the DDR2 loopback test error result.

Function

lbk_test_result	Description
0 (default)	No error detected
1	Errors detected

The following are possible failures that could cause the DDR2 loopback test to fail:

- Address or control or clock open or short: all bit lines failing
- Single DQ open or short to ground or supply: single bit line failing on both positive and negative edges
- Short between DQ lines: two bit lines failing, routing in adjacent resistors of resistor pack
- DQS or DM open or short: eight DQ lines failing
- Timing transfer problem: one or more bit lines failing

2.2.6. *I^{2}C Auto Increment*

read_auto_inc_en is used to auto increment register addresses to allow the user to do consecutive reads from the registers on the ADV8005. By default, this is set to 1 which means that a read from a particular address in the ADV8005 increments the read pointer to the next register map address.

read_auto_inc_en, IO Map, *Address 0x1AFC[0]*
This register is used to auto increment I2C addresses in the device for consecutive reads.

Function

read_auto_inc_en	Description
0	No auto increment of I2C address for consecutive reads
1 (default)	Auto increment of I2C address for consecutive reads

2.2.7. *SPI Loop Through*

The ADV8005 SPI ports can be put in loop through mode for programming the external SPI flash that may be connected to the ADV8005 master SPI port (if an OSD design is to be used). Refer to Section 4.2.8 for more information.

spi_loop_through, IO Map, *Address 0x1AB6[5]*
This bit is used to enable SPI loop through mode. In loop through mode, Serial Port 1 (SCK1, MOSI1, MISO1 and CS1) is connected to the Serial Port 2 (SCK2, MOSI2, MISO2, CS2).

Function

spi_loop_through	Description
0 (default)	Regular SPI mode
1	SPI slave clock routed to SPI master clock output

2.2.8. *VBI Data Insertion*

ADV8005 supports VBI data (such as CGMS, WSS, and CCAP) insertion into the video stream through either the ancillary data input (Y channel input of 36-bit data bus) or the SPI-compatible slave input (VBI_SCK, VBI_MOSI and VBI_CS). When using the SPI-compatible slave input for VBI insertion, a reduced set of video input formats are supported on the EXOSD TTL input due to the shared pins. The VBI data is decoded and supplied to the encoder for output in the video data stream.

The supported VBI standards are the following:

- WSS (625i)
- CCAP (525i and 625i)
- CGMS (525i)
- CGMS (525p)
- CGMS (625p)

2.2.8.1. *Extraction Overview*

VBI data can be supplied to the ADV8005 through two separate interfaces. If there is a pixel bus input from the front end decoder then the VBI data may be provided via an ancillary data stream encoded into the video data. If a pixel bus is not available, the VBI data can be sent via the dedicated SPI interface. Refer to Figure 40 for an overview of this architecture.

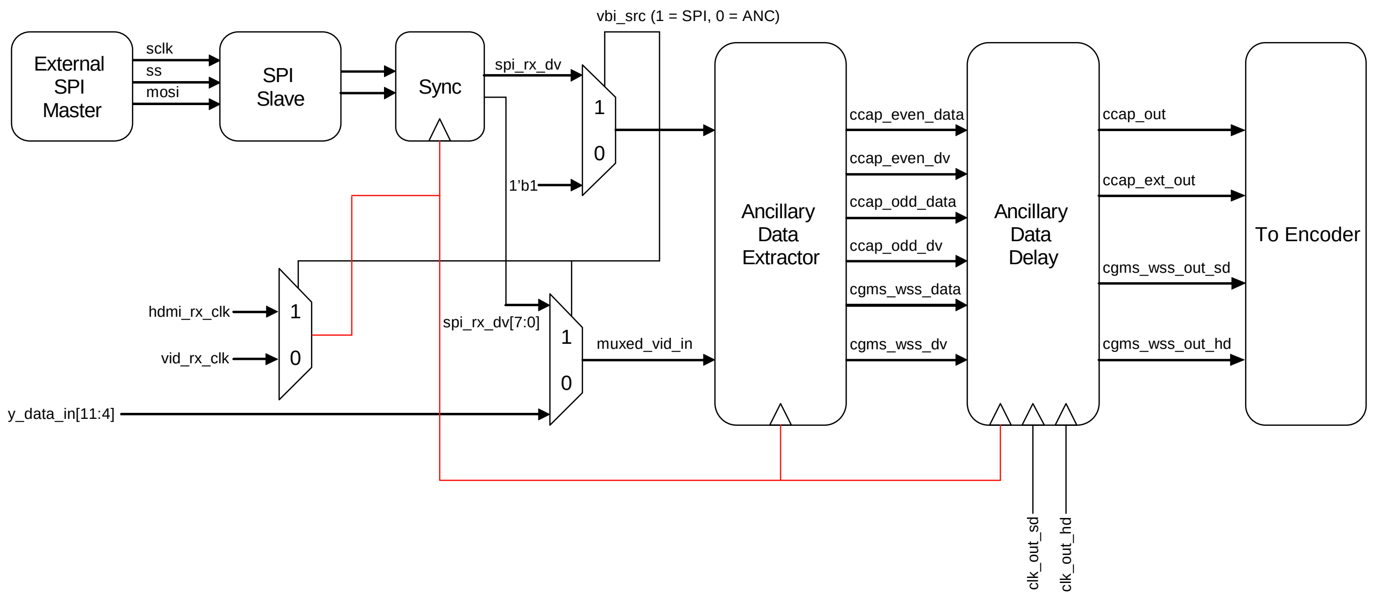


Figure 40: VBI Data Extraction Block Diagram

2.2.8.2. *Ancillary Data Extraction*

The ancillary data which is encoded in either nibble mode or byte mode is extracted from the input data stream on the Y channel and the VBI data is retrieved. The DID and SDID from the sending device must match the value programmed in 1A 1A4A[7:0] and 1A 1A4B[7:0]. The format of the ancillary data packet is shown in Table 9.

Table 9: Output Mode Outline

Byte	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0	Description
0	0	0	0	0	0	0	0	0	0	0	Ancillary Data Preamble
1	1	1	1	1	1	1	1	1	1	1	
2	1	1	1	1	1	1	1	1	1	1	
3	EP	EP		I2C_DID6[4:0]					0	0	DID Data Identification Word
4	EP	EP	I2C_SDID7_2[5:0]						0	0	SDID Secondary Data Identification Word
5	EP	EP	0	DC[4:0]					0	0	ID1 User Data Word 1
6	EP	EP	Padding[1:0]		VBI_DATA_STD[3:0]				0	0	ID2 User Data Word 2
7	EP	EP	LCOUNT[11:6]						0	0	ID3 User Data Word 3
8	EP	EP	LCOUNT[5:0]						0	0	ID4 User Data Word 4
9	EP	EP	0	0	0	EF	VDP_TTXT TYPE[1:0]		0	0	ID5 User Data Word 5
10	EP	EP	0	0	VBI_WORD_1[7:4]				0	0	ID6 User Data Word 6
11	EP	EP	0	0	VBI_WORD_1[3:0]				0	0	ID7 User Data Word 7
12	EP	EP	0	0	VBI_WORD_2[7:4]				0	0	ID8 User Data Word 8
13	EP	EP	0	0	VBI_WORD_2[3:0]				0	0	ID9 User Data Word 9
	1	0	0	0	0	0	0	0	0	0	Pad, May or may not be present
N-1	B8	Checksum							0	0	

2.2.8.3. *SPI Data Extraction*

If there is not an input video data bus which can provide the ancillary data, it may be serialized and sent to the part via a SPI master. The ADV8005 contains a dedicated SPI slave for receiving VBI data. The SPI interface receives serialized ancillary data bytes. All of the ancillary data packets must be encoded, including the preamble. A high to low transition on the VBI_CS line indicates the start of a new byte. As the bytes are directly encoded ancillary data, the same decoder described in Section 2.2.8.2 for ancillary data can be used to extract the VBI data. Only modes 0 and 3 are supported by the SPI slave and, therefore, the SPI master must use one of these modes.

2.2.8.4. *VBI Data Delay*

Once the VBI data has been decoded for each of the supported standards, it is latched and delayed by the desired amount. The delay on the VBI data is measured in frames and is controllable in the range $0 \leq \text{delay} \leq 3$ frames. The data can be delayed on either the rising or falling edge of the input VSync. The output VBI data is muxed directly with the VBI data from the encoder register map before being output by the encoder.

vbi_src, IO Map, *Address 0x1A4C[7]*
This bit is used to choose the source of the VBI data.

Function

vbi_src	**Description**
0 (default)	VBI data from ancillary input
1	VBI data from SPI input

ccap_odd_en, IO Map, *Address 0x1A4C[3]*
This bit is used to enable/disable closed caption data extraction on the odd field.

Function

ccap_odd_en	**Description**
0 (default)	Disable closed caption data extraction on odd field
1	Enable closed caption data extraction on odd field

ccap_even_en, IO Map, *Address 0x1A4C[2]*
This bit is used to enable/disable closed caption data extraction on the even field.

Function

ccap_even_en	**Description**
0 (default)	Disable closed caption data extraction on even field
1	Enable closed caption data extraction on even field

cgms_anc_en, IO Map, *Address 0x1A4C[1]*
This bit is used to enable/disable CGMS data extraction on the even field.

Function

cgms_anc_en	**Description**
0 (default)	Disable CGMS data extraction on even field
1	Enable CGMS data extraction on even field

wss_anc_en, IO Map, *Address 0x1A4C[0]*
This bit is used to enable/disable WSS data extraction on the even field.

Function

wss_anc_en	**Description**
0 (default)	Disable WSS data extraction on even field
1	Enable WSS data extraction on even field

anc_delay[1:0], IO Map, *Address 0x1A4D[1:0]*
This bit is used to set the delay on ancillary data in vsyncs. The interlaced input delay will be in fields and the progressive delay will be in frames. Decoded data is firstly transferred onto input vsync and then output vsync, this will be the base delay with a setting of 0. Every increment above this adds one input vsync delay.

did_a[7:0], IO Map, *Address 0x1A4A[7:0]*
This register is used to specify the value of the DID sent in the ancillary stream with VBI decoded data.

sdid_a[7:0], IO Map, *Address 0x1A4B[7:0]*
This register is used to specify the value of the SDID sent in the ancillary stream with VBI decoded data.

2.2.9. *Resets*

This section documents the register bits used for resetting various sections of the ADV8005. These resets can be used by the system controller to reset individual sections of the device without having to reset the whole part. If the whole device needs to be reset, this can be implemented by setting the global reset, main_reset. All these register bits are self clearing, which means that when set to 1, they are set back to 0 after the appropriate section has been reset.

Refer to Section 6.2 for more information on the reset strategy for the HDMI Tx.

svsp_reset, IO Map, *Address 0x1AFD[7] (Self-Clearing)*
This bit is used to reset the Secondary VSP.

Function

svsp_reset	**Description**
0 (default)	Default
1	Reset

pvsp_reset, IO Map, *Address 0x1AFD[6] (Self-Clearing)*
This bit is used to reset the Primary VSP.

Function

pvsp_reset	Description
0 (default)	Default
1	Reset

p2i_reset, IO Map, *Address 0x1AFD[5] (Self-Clearing)*
This bit is used to reset the Progressive to Interlaced core.

Function

p2i_reset	Description
0 (default)	Default
1	Reset

ddr2_intf_reset, IO Map, *Address 0x1AFD[4] (Self-Clearing)*
This bit is used to reset the external DDR memory interface core.

Function

ddr2_intf_reset	Description
0 (default)	Default
1	Reset

spi_reset, IO Map, *Address 0x1AFD[3] (Self-Clearing)*
This bit is used to reset the SPI hardware, both master and slave.

Function

spi_reset	Description
0 (default)	Default
1	Reset

sys_clk_reset, IO Map, *Address 0x1AFD[2] (Self-Clearing)*
This register bit resets the clock for the digital core.

Function

sys_clk_reset	Description
0 «	Default
1	Reset

osd_reset, IO Map, *Address 0x1AFD[1] (Self-Clearing)*
This bit is used to reset the OSD core and the secondary input channel.

Function

osd_reset	Description
0 (default)	Default
1	Reset

inp_sdr_reset, IO Map, *Address 0x1AFD[0] (Self-Clearing)*
This bit is used to reset the input capture and formatting logic for the primary input channel.

Function

inp_sdr_reset	Description
0 (default)	Default
1	Reset

rx_reset, IO Map, *Address 0x1AFE[7] (Self-Clearing)*
This bit is used to reset the Serial Video RX core and the RX input channel.

Function

rx_reset	Description
0 (default)	Default
1	Reset

enc_reset, IO Map, *Address 0x1AFE[6] (Self-Clearing)*
This bit is used to reset the HD and SD encoders.

Function

enc_reset	**Description**
0 (default)	Default
1	Reset

tx2_reset, IO Map, *Address 0x1AFE[5] (Self-Clearing)*
This bit is used to reset the HDMI TX2.

Function

tx2_reset	**Description**
0 (default)	Default
1	Reset

tx1_reset, IO Map, *Address 0x1AFE[4] (Self-Clearing)*
This bit is used to reset the HDMI TX1.

Function

tx1_reset	**Description**
0 (default)	Default
1	Reset

dpll_reset, IO Map, *Address 0x1AFE[2] (Self-Clearing)*
This bit is used to reset the DPLL clock generator.

Function

dpll_reset	**Description**
0 (default)	Default
1	Reset

xtal_reset, IO Map, *Address 0x1AFE[0] (Self-Clearing)*
This bit is used to reset all the clocks in the device and peripheral logic in the core including the interrupt generator and the automatic clock selection.

Function

xtal_reset	**Description**
0 (default)	Default
1	Reset

main_reset, IO Map, *Address 0x1BFF[7] (Self-Clearing)*
This bit is used to initiate a global reset for the device.

Function

main_reset	**Description**
0 (default)	Default
1	Reset

2.2.10. *Image Processing Colorimetry Breakdown*

The ADV8005 performs its image processing in the YUV format except for the internal OSD which is generated in RGB. The internally generated OSD is muxed with the external OSD (which can be in either YUV or RGB) before being input into a CSC. The CSC converts all input signals into YUV format for input into the OSD video blend block.

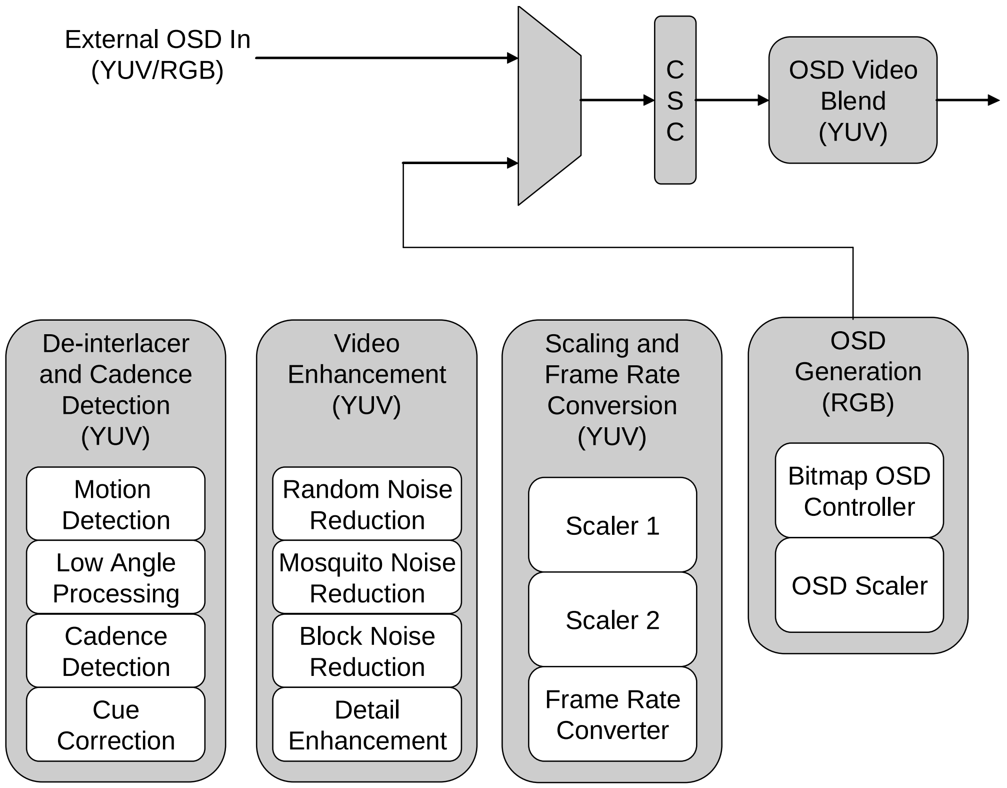


Figure 41: ADV8005 Image Processing Colorimetry Breakdown

2.2.11. *AV-Codes*

Embedded end of active video (EAV) and start of active video (SAV) timing codes are supported on the TTL inputs of the ADV8005. AV-code information is embedded into the pixel data and is transmitted using a standard 4-byte synchronization pattern. A synchronization pattern is sent immediately before and after each line during active picture and retrace.

The following video formats are supported automatically for AV-code insertion.

- 480i60
- 576i50
- 240p60
- 288p50
- 480p60
- 576p50
- 720p60
- 720p50
- 1080i60
- 1080i50
- 1080p60
- 1080p50
- VGA (640x480)
- SVGA (800x600)
- XGA (1027x768
- WXGA (1280x768)
- SXGA (1280x1024)
- WXGA (1360x768)

- UXGA (1600x1200)
- WXGA(1366x768
- WUXGA (1900x1200)

A number of CEA formats are not supported automatically for AV-codes

1. 1920x1080p @ 23.97/24 Hz (CEA VIC 32)
2. 1920x1080p @ 25 Hz (CEA VIC 33)
3. 1920x1080p @ 29.97/30 Hz (CEA VIC 34)
4. 1280x720p @ 23.97/24 Hz (CEA VIC 60)
5. 1280x720p @ 25 Hz (CEA VIC 61)
6. 1280x720p @ 29.97/30 Hz (CEA VIC 62)

These formats can be supported following the manual configuration mode outlined in this section.

de_v_beg_e_pos[6:0], IO Map, *Address 0x1B8C[7:1]*
This signal is used to specify the DE vertical beginning position for even fields, if CEA 861 timing generation is enable and manual values selected.

Function

de_v_beg_e_pos[6:0]	Description
0xXX	assert de when lcount reaches 0xXX on even fields

de_v_beg_o_pos[6:0], IO Map, *Address 0x1B8C[0]; Address 0x1B8D[7:2]*
This signal is used to specify the DE vertical beginning position for odd fields, if CEA 861 timing generation is enable and manual values selected.

Function

de_v_beg_o_pos[6:0]	Description
0xXX	assert de when lcount reaches 0xXX on even fields

de_h_beg_pos[9:0], IO Map, *Address 0x1B8D[1:0]; Address 0x1B8E[7:0]*
This signal is used to specify the DE horizontal beginning position, counting from the EAV, if CEA 861 timing generation is enable and manual values selected.

Function

de_h_beg_pos[9:0]	Description
0xXX	assert de when hcount reaches 0xXX

hs_beg_pos[9:0], IO Map, *Address 0x1B8F[7:0]; Address 0x1B90[7:6]*
This signal is used to specify the HS beginning position, counting from the EAV, if CEA 861 timing generation is enable and manual values selected.

Function

hs_beg_pos[9:0]	Description
0xXX	assert hs when hcount reaches 0xXX

hs_end_pos[9:0], IO Map, *Address 0x1B90[5:0]; Address 0x1B91[7:4]*
This signal is used to specify the HS ending position, counting from the EAV, if CEA 861 timing generation is enable and manual values selected.

Function

hs_end_pos[9:0]	Description
0xXX	release hs when hcount reaches 0xXX

vs_h_beg_o_pos[10:0], IO Map, *Address 0x1B91[2:0]; Address 0x1B92[7:0]*
This signal is used to specify the horizontal beginning position of VS for odd fields (counting from the EAV), if CEA 861 timing generation is enable and manual values selected.

Function

vs_h_beg_o_pos[10:0]	Description
0xXX	assert vs when hcount reaches 0xXX on odd fields

vs_h_beg_e_pos[10:0], IO Map, *Address 0x1B93[7:0]; Address 0x1B94[7:5]*
This signal is used to specify the horizontal beginning position of VS for even fields (counting from the EAV), if CEA 861 timing generation is enable and manual values selected.

Function

vs_h_beg_e_pos[10:0]	Description
0xXX	assert vs when hcount reaches 0xXX on even fields

vs_v_beg_pos[5:0], IO Map, *Address 0x1B94[3:0]; Address 0x1B95[7:6]*
This signal is used to specify the vertical beginning position of VS, if CEA 861 timing generation is enable and manual values selected.

Function

vs_v_beg_pos[5:0]	Description
0xXX	assert vs when lcount reaches 0xXX

vs_v_end_pos[5:0], IO Map, *Address 0x1B95[5:0]*
This signal is used to specify the vertical ending position of VS, if CEA 861 timing generation is enable and manual values selected.

Function

vs_v_end_pos[5:0]	Description
0xXX	release vs when lcount reaches 0xXX

For the secondary input channel, the controls are as follows:

de_v_beg_e_pos[6:0], IO Map, *Address 0x1B8C[7:1]*
This signal is used to specify the DE vertical beginning position for even fields, if CEA 861 timing generation is enable and manual values selected.

Function

de_v_beg_e_pos[6:0]	Description
0xXX	assert de when lcount reaches 0xXX on even fields

de_v_beg_o_pos[6:0], IO Map, *Address 0x1B8C[0]; Address 0x1B8D[7:2]*
This signal is used to specify the DE vertical beginning position for odd fields, if CEA 861 timing generation is enable and manual values selected.

Function

de_v_beg_o_pos[6:0]	Description
0xXX	assert de when lcount reaches 0xXX on even fields

de_h_beg_pos[9:0], IO Map, *Address 0x1B8D[1:0]; Address 0x1B8E[7:0]*
This signal is used to specify the DE horizontal beginning position, counting from the EAV, if CEA 861 timing generation is enable and manual values selected.

Function

de_h_beg_pos[9:0]	Description
0xXX	assert de when hcount reaches 0xXX

hs_beg_pos[9:0], IO Map, *Address 0x1B8F[7:0]; Address 0x1B90[7:6]*
This signal is used to specify the HS beginning position, counting from the EAV, if CEA 861 timing generation is enable and manual values selected.

Function

hs_beg_pos[9:0]	Description
0xXX	assert hs when hcount reaches 0xXX

hs_end_pos[9:0], IO Map, *Address 0x1B90[5:0]; Address 0x1B91[7:4]*
This signal is used to specify the HS ending position, counting from the EAV, if CEA 861 timing generation is enable and manual values selected.

Function

hs_end_pos[9:0]	**Description**
0xXX	release hs when hcount reaches 0xXX

vs_h_beg_o_pos[10:0], IO Map, *Address 0x1B91[2:0]; Address 0x1B92[7:0]*
This signal is used to specify the horizontal beginning position of VS for odd fields (counting from the EAV), if CEA 861 timing generation is enable and manual values selected.

Function

vs_h_beg_o_pos[10:0]	**Description**
0xXX	assert vs when hcount reaches 0xXX on odd fields

vs_h_beg_e_pos[10:0], IO Map, *Address 0x1B93[7:0]; Address 0x1B94[7:5]*
This signal is used to specify the horizontal beginning position of VS for even fields (counting from the EAV), if CEA 861 timing generation is enable and manual values selected.

Function

vs_h_beg_e_pos[10:0]	**Description**
0xXX	assert vs when hcount reaches 0xXX on even fields

vs_v_beg_pos[5:0], IO Map, *Address 0x1B94[3:0]; Address 0x1B95[7:6]*
This signal is used to specify the vertical beginning position of VS, if CEA 861 timing generation is enable and manual values selected.

Function

vs_v_beg_pos[5:0]	**Description**
0xXX	assert vs when lcount reaches 0xXX

vs_v_end_pos[5:0], IO Map, *Address 0x1B95[5:0]*
This signal is used to specify the vertical ending position of VS, if CEA 861 timing generation is enable and manual values selected.

Function

vs_v_end_pos[5:0]	**Description**
0xXX	release vs when lcount reaches 0xXX

A worked example showing how 720 (1440) x 240p can be supported using manual AV-code configuration is shown in Figure 42. The horizontal measurements must be the following:

de_h_beg_pos[9:0] = 276/2 = 138 = 0010001010
de_v_beg_o_pos[6:0] = 22 = 0010110
de_v_beg_e_pos[6:0] = 22 = 0010110
hs_beg_pos[9:0] = 38/2 = 19 = 0000010011
hs_end_pos[9:0] = 162/2 = 81 = 0001010001
vs_v_beg_pos[5:0] = 4 = 000100
vs_h_beg_o_pos[10:0] = 38/2 = 19 = 00000010011
vs_h_beg_e_pos[10:0] = 38/2 = 19 = 00000010011
vs_v_end_pos[5:0] = 7 = 000111

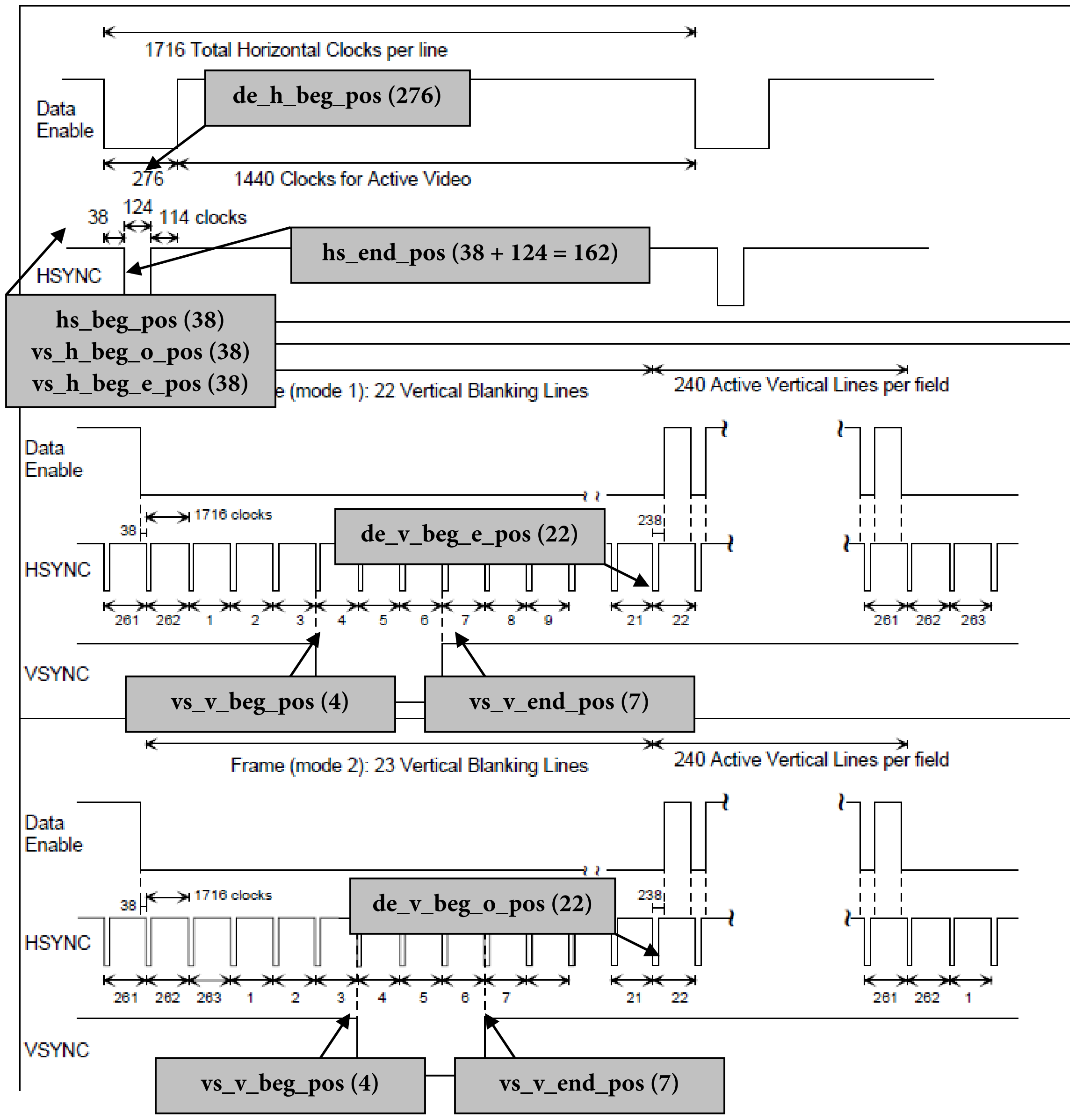


Figure 42: 720(1440) x 240p @ 59.94/60Hz, CEA Formats 8 and 9

2.2.12. *Color Space Conversion*

Although all processing in the ADV8005 is performed in the YCbCr color space, the part is capable of receiving video in the RGB, YUV and YCbCr color spaces. The ADV8005 provides any-to-any CSC on each of the inputs and on both of the outputs (five color space converters in all). All CSCs support formats such as RGB, YUV and YCbCr. The front end CSCs on the primary input channel, secondary input channel and RX input channel run at a maximum clock rate of 162 MHz. The back end CSCs in HDMI Tx1 and HDMI Tx2 operate at a maximum input clock rate of 300 MHz.

2.2.12.1. *Primary Input Channel CSC*

The CSC must be manually configured for each color space conversion. The CSC on the primary input channel can be enabled using the vid_csc_enable control. This CSC can run at 297 MHz and provides color space conversion for UHD video formats. The CSC mode on the primary input channel can be configured using vid_csc_mode[1:0]. The CSC mode is used to define the fixed point position of the CSC coefficients which are located after vid_csc_mode[1:0] in the IO Map for the primary input channel.

Reference configuration scripts to configure the primary input channel CSC are provided with the evaluation software.

vid_csc_enable, IO Map, *Address 0x1B30[7]*
This bit is used to control the Primary Input Channel CSC.

Function

vid_csc_enable	**Description**
0 (default)	CSC disable
1	CSC enable

vid_csc_mode[1:0], IO Map, *Address 0x1B30[6:5]*
This signal is used to specify the CSC mode for the Primary Input Channel CSC. The CSC mode sets the fixed point position of the CSC coefficients, including a4, b4, c4 and offsets.

Function

vid_csc_mode[1:0]	**Description**
00 (default)	+/- 1.0, -4096 to 4095
01	+/-2.0, -8192 to 8190
10	+/- 4.0, -16384 to 16380
11	+/- 4.0, -16384 to 16380

The characteristic equations for the primary input CSC are provided in Equation 4, Equation 5 and Equation 6.

$$Out\_A = \left[In\_A * \frac{A1[12:0]}{4096} + In\_B * \frac{A2[12:0]}{4096} + In\_C * \frac{A3[12:0]}{4096} + A4[12:0] \right] * 2^{CSC\_scale}$$

Equation 4: Primary Input CSC Channel A Output

$$Out\_B = \left[In\_A * \frac{B1[12:0]}{4096} + In\_B * \frac{B2[12:0]}{4096} + In\_C * \frac{B3[12:0]}{4096} + B4[12:0] \right] * 2^{CSC\_scale}$$

Equation 5: Primary Input CSC Channel B Output

$$Out\_C = \left[In\_A * \frac{C1[12:0]}{4096} + In\_B * \frac{C2[12:0]}{4096} + In\_C * \frac{C3[12:0]}{4096} + C4[12:0] \right] * 2^{CSC\_scale}$$

Equation 6: Primary Input CSC Channel C Output

The CSC on the primary input channel is illustrated in Figure 43.

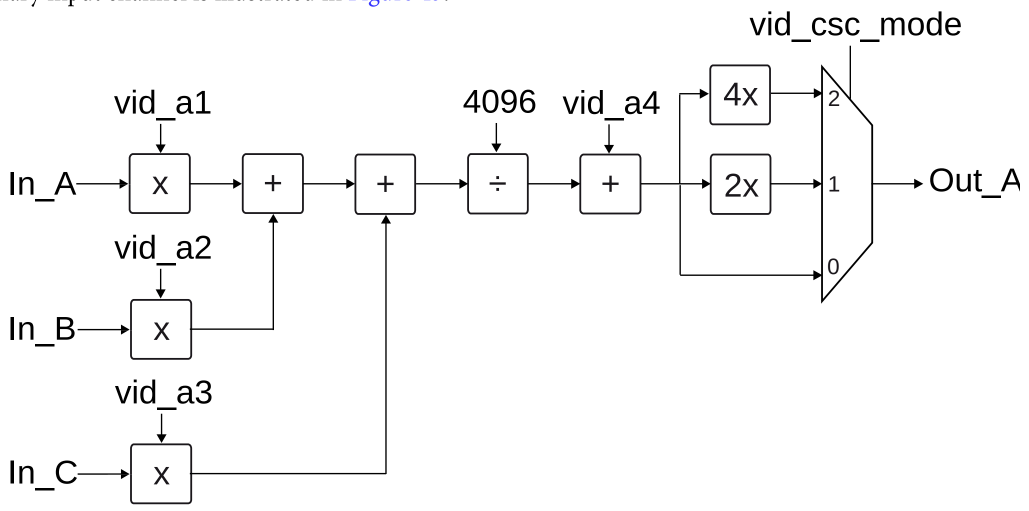


Figure 43: Primary Input Channel CSC

The video inputs In_A, In_B and In_C are connected by default to R, G and B. Refer to Table 10 for more information. The default routing can be changed by adjusting the value of vid_swap_bus_ctrl[2:0].

Table 10: Default Primary Input Channel CSC Signal Routing

Input Channel	Default RGB Routing	Default YCbCr Routing
In_A	R	Cr
In_B	G	Y
In_C	B	Cb

The A1 to A3, B1 to B3, and C1 to C3 coefficients are used to scale the primary inputs. A4, B4 and C4 are added as offsets. Floating point coefficients must be converted into 120-bit fixed decimal format then converted into binary format using twos complement for negative values and can only be programmed in the range [-1….+1] or [-4096….+4095].

The dynamic range of the CSC is [0…..1] for unipolar signals (Y, R, G, B) or [-0.5…….+0.5] for bipolar signals. Bipolar signals (Pr/Pb) must be offset to mid range. Equations with a dynamic range larger than 1 need to be scaled appropriately using the vid_csc_mode[1:0] control. To achieve a coefficient value of 1.0 for any given coefficient, vid_csc_mode[1:0] should be set high and the coefficient should be programmed to a value of 0.5. Otherwise, the largest value would be 4095/4096 = 0.9997. While this value could be interpreted as 1, it is recommended to use the value of 0.5 and set the vid_csc_mode[1:0] bits for maximum accuracy.

The CSC configurations for common modes are provided in Table 11.

Table 11: Primary Input Channel CSC Common Configuration Coefficients

Color Space Conversion	csc_mode[1:0]	A1	A2	A3	A4	B1	B2	B3	B4	C1	C2	C3	C4
HDTV YCbCr (limited) to RGB (limited)	0x1	0x0C53	0x0800	0x0000	0x19D6	0x1C56	0x0800	0x1E88	0x0291	0x1FFF	0x0800	0x0E85	0x18BE
HDTV YCbCr (limited) to RGB (full)	0x2	0x0734	0x04AD	0x0000	0x1C1B	0x1DDC	0x04AD	0x1F24	0x0135	0x0000	0x04AD	0x087C	0x1B77
HDTV YCbCr (limited) to SDTV YCbCr (limited)	0x1	0x07DD	0x0000	0x1F6C	0x005B	0x0188	0x0800	0x00CB	0x1ED6	0x1F1D	0x0000	0x07EB	0x007B
HDTV YCbCr (limited) to SDTV YCbCr (full)	0x1	0x08EB	0x0000	0x1F58	0x1FDE	0x01C9	0x0950	0x00EC	0x1F25	0x1EFF	0x0000	0x08FA	0x031F
HDTV YCbCr (full) to SDTV YCbCr (limited)	0x0	0x0E0D	0x0000	0x0000	0x0100	0x0000	0x0DBC	0x0000	0x0100	0x0000	0x0000	0x0E0D	0x0100
SDTV YCbCr (limited) to RGB (limited)	0x1	0x0AF8	0x0800	0x0000	0x1A84	0x1A6A	0x0800	0x1D50	0x0423	0x1FFC	0x0800	0x0DDE	0x1913
SDTV YCbCr (limited) to RGB (full)	0x2	0x0669	0x04AC	0x0000	0x1C81	0x1CBC	0x04AD	0x1E6E	0x0220	0x1FFE	0x04AD	0x081A	0x1BA9
SDTV YCbCr (limited) to HDTV YCbCr (limited)	0x1	0x0833	0x0000	0x0099	0x1F99	0x1E56	0x0800	0x1F13	0x014B	0x00EA	0x0000	0x0826	0x1F78
SDTV YCbCr (limited) to SDTV YCbCr (full)	0x1	0x091B	0x0000	0x0000	0x1F6E	0x0000	0x0950	0x0000	0x1F6B	0x0000	0x0000	0x091B	0x1F6E
SDTV YCbCr (full) to HDTV YCbCr (limited)	0x2	0x039D	0x0000	0x0043	0x0F26	0x1F44	0x036F	0x1F97	0x00D2	0x0067	0x0000	0x0397	0x004D

Color Space Conversion	csc_mode[1:0]	A1	A2	A3	A4	B1	B2	B3	B4	C1	C2	C3	C4
RGB (limited) to HDTV YCbCr (limited)	0x0	0x082E	0x1893	0x1F3F	0x0800	0x0367	0x0B71	0x0128	0x0000	0x1E21	0x19B2	0x082D	0x0800
RGB (limited) to SDTV YCbCr (limited)	0x0	0x082E	0x1926	0x1EAC	0x0800	0x04C9	0x0965	0x01D2	0x0000	0x1D3F	0x1A93	0x082E	0x0800
RGB (limited) to RGB (full)	0x0	0x0DBC	0x0000	0x0000	0x0100	0x0000	0x0DBC	0x0000	0x0100	0x0000	0x0000	0x0DBC	0x0100
RGB (full) to HDTV YCbCr (limited)	0x0	0x06FF	0x19A6	0x1F5B	0x0800	0x02E9	0x09CB	0x00FD	0x0100	0x1E66	0x1A9B	0x06FF	0x0800
RGB (Full) to SDTV YCbCr (limited)	0x0	0x06FF	0x1A24	0x1EDD	0x0800	0x0418	0x080A	0x018F	0x0100	0x1DA5	0x1B5C	0x06FF	0x0800
RGB (Full) to RGB (limited)	0x1	0x0950	0x0000	0x0000	0x1F6B	0x0000	0x0950	0x0000	0x1F6B	0x0000	0x0000	0x0950	0x1F6B
Identity matrix (output = input)	0x1	0x0800	0x0000	0x0000	0x0000	0x0000	0x0800	0x0000	0x0000	0x0000	0x0000	0x0800	0x0000

2.2.12.2. *Secondary Input Channel CSC*

The CSC must be manually configured for each color space conversion. The CSC on the secondary input channel can be enabled using the exosd_csc_enable control. This CSC can run at pixel clock frequencies up to 162MHz. The CSC mode on the secondary input channel can be configured using exosd_csc_mode[1:0]. The CSC mode is used to define the fixed point position of the CSC coefficients which are located after exosd_csc_mode[1:0] in the IO Map for the secondary input channel.

Reference configuration scripts to configure the secondary input channel CSC are provided with the evaluation software.

exosd_csc_enable, IO Map, *Address 0x1B50[7]*
This bit is used to enable the Secondary Input Channel CSC.

Function

exosd_csc_enable	Description
0 (default)	CSC disable
1	CSC enable

exosd_csc_mode[1:0], IO Map, *Address 0x1B50[6:5]*
This signal is used to specify the CSC mode for the Secondary Input Channel CSC. The CSC mode sets the fixed point position of the CSC coefficients, including a4, b4, c4 and offsets.

Function

exosd_csc_mode[1:0]	Description
00 (default)	+/- 1.0, -4096 to 4095
01	+/-2.0, -8192 to 8190
10	+/- 4.0, -16384 to 16380
11	+/- 4.0, -16384 to 16380

The characteristic equations for the secondary input CSC are provided in Equation 7, Equation 8 and Equation 9.

$$Out\_A = \left[In\_A * \frac{A1[12:0]}{4096} + In\_B * \frac{A2[12:0]}{4096} + In\_C * \frac{A3[12:0]}{4096} + A4[12:0] \right] * 2^{CSC\_scale}$$

Equation 7: Secondary Input CSC Channel A Output

$$Out\_B = \left[In\_A * \frac{B1[12:0]}{4096} + In\_B * \frac{B2[12:0]}{4096} + In\_C * \frac{B3[12:0]}{4096} + B4[12:0] \right] * 2^{CSC\_scale}$$

Equation 8: Secondary Input CSC Channel B Output

$$Out\_C = \left[In\_A * \frac{C1[12:0]}{4096} + In\_B * \frac{C2[12:0]}{4096} + In\_C * \frac{C3[12:0]}{4096} + C4[12:0] \right] * 2^{CSC\_scale}$$

Equation 9: Secondary Input CSC Channel C Output

The CSC on the secondary input channel is illustrated in Figure 44.

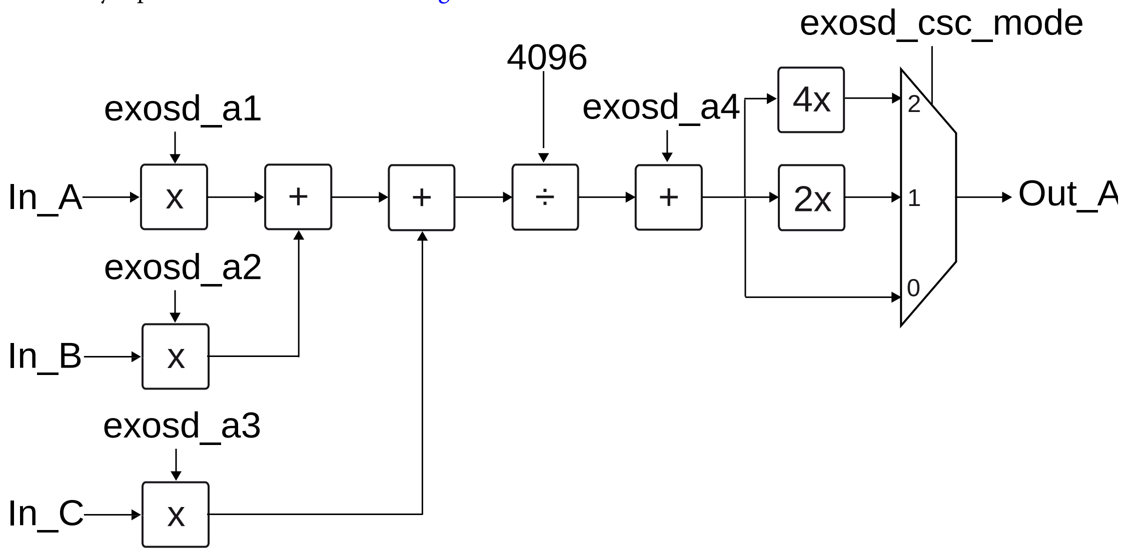


Figure 44: Secondary Input Channel CSC

The video inputs In_A, In_B and In_C are connected by default to R, G and B. Refer to Table 12 for more information. The default routing can be changed by adjusting the value of exosd_swap_bus_ctrl[2:0].

Table 12: Default Secondary Input Channel CSC Signal Routing

Input Channel	Default RGB Routing	Default YCbCr Routing
In_A	R	Cr
In_B	G	Y
In_C	B	Cb

The A1 to A3, B1 to B3, and C1 to C3 coefficients are used to scale the primary inputs. A4, B4 and C4 are added as offsets. Floating point coefficients must be converted into 120-bit fixed decimal format then converted into binary format using twos complement for negative values and can only be programmed in the range [-1….+1] or [-4096….+4095].

The dynamic range of the CSC is [0…..1] for unipolar signals (Y, R, G, B) or [-0.5…….+0.5] for bipolar signals. Bipolar signals (Pr/Pb) must be offset to mid range. Equations with a dynamic range larger than 1 need to be scaled appropriately using the exosd_csc_mode[1:0] control. To achieve a coefficient value of 1.0 for any given coefficient, exosd_csc_mode[1:0] should be set high and the coefficient should be programmed to a value of 0.5. Otherwise, the largest value would be 4095/4096 = 0.9997. While this value could be interpreted as 1, it is recommended to use the value of 0.5 and set the exosd_csc_mode[1:0] bits for maximum accuracy.

The CSC configurations for common modes are provided in Table 13.

Table 13: Secondary Input Channel CSC Common Configuration Coefficients

Color Space Conversion	csc_mode[1:0]	A1	A2	A3	A4	B1	B2	B3	B4	C1	C2	C3	C4
HDTV YCbCr (limited) to RGB (limited)	0x1	0x0C53	0x0800	0x0000	0x19D6	0x1C56	0x0800	0x1E88	0x0291	0x1FFF	0x0800	0x0E85	0x18BE
HDTV YCbCr (limited) to RGB (full)	0x2	0x0734	0x04AD	0x0000	0x1C1B	0x1DDC	0x04AD	0x1F24	0x0135	0x0000	0x04AD	0x087C	0x1B77
HDTV YCbCr (limited) to SDTV YCbCr (limited)	0x1	0x07DD	0x0000	0x1F6C	0x005B	0x0188	0x0800	0x00CB	0x1ED6	0x1F1D	0x0000	0x07EB	0x007B
HDTV YCbCr (limited) to SDTV YCbCr (full)	0x1	0x08EB	0x0000	0x1F58	0x1FDE	0x01C9	0x0950	0x00EC	0x1F25	0x1EFF	0x0000	0x08FA	0x031F
HDTV YCbCr (full) to SDTV YCbCr (limited)	0x0	0x0E0D	0x0000	0x0000	0x0100	0x0000	0x0DBC	0x0000	0x0100	0x0000	0x0000	0x0E0D	0x0100
SDTV YCbCr (limited) to RGB (limited)	0x1	0x0AF8	0x0800	0x0000	0x1A84	0x1A6A	0x0800	0x1D50	0x0423	0x1FFC	0x0800	0x0DDE	0x1913
SDTV YCbCr (limited) to RGB (full)	0x2	0x0669	0x04AC	0x0000	0x1C81	0x1CBC	0x04AD	0x1E6E	0x0220	0x1FFE	0x04AD	0x081A	0x1BA9
SDTV YCbCr (limited) to HDTV YCbCr (limited)	0x1	0x0833	0x0000	0x0099	0x1F99	0x1E56	0x0800	0x1F13	0x014B	0x00EA	0x0000	0x0826	0x1F78
SDTV YCbCr (limited) to SDTV YCbCr (full)	0x1	0x091B	0x0000	0x0000	0x1F6E	0x0000	0x0950	0x0000	0x1F6B	0x0000	0x0000	0x091B	0x1F6E
SDTV YCbCr (full) to HDTV YCbCr (limited)	0x2	0x039D	0x0000	0x0043	0x0F26	0x1F44	0x036F	0x1F97	0x00D2	0x0067	0x0000	0x0397	0x004D
RGB (limited) to HDTV YCbCr (limited)	0x0	0x082E	0x1893	0x1F3F	0x0800	0x0367	0x0B71	0x0128	0x0000	0x1E21	0x19B2	0x082D	0x0800
RGB (limited) to SDTV YCbCr (limited)	0x0	0x082E	0x1926	0x1EAC	0x0800	0x04C9	0x0965	0x01D2	0x0000	0x1D3F	0x1A93	0x082E	0x0800
RGB (limited) to RGB (full)	0x0	0x0DBC	0x0000	0x0000	0x0100	0x0000	0x0DBC	0x0000	0x0100	0x0000	0x0000	0x0DBC	0x0100
RGB (full) to HDTV YCbCr (limited)	0x0	0x06FF	0x19A6	0x1F5B	0x0800	0x02E9	0x09CB	0x00FD	0x0100	0x1E66	0x1A9B	0x06FF	0x0800
RGB (full) to SDTV YCbCr (limited)	0x0	0x06FF	0x1A24	0x1EDD	0x0800	0x0418	0x080A	0x018F	0x0100	0x1DA5	0x1B5C	0x06FF	0x0800
RGB (full) to	0x1	0x0950	0x0000	0x0000	0x1F6B	0x0000	0x0950	0x0000	0x1F6B	0x0000	0x0000	0x0950	0x1F6B

Color Space Conversion	csc_mode[1:0]	A1	A2	A3	A4	B1	B2	B3	B4	C1	C2	C3	C4
RGB (limited)													
Identity Matrix (Output = Input)	0x1	0x0800	0x0000	0x0000	0x0000	0x0000	0x0800	0x0000	0x0000	0x0000	0x0000	0x0800	0x0000

2.2.12.3. RX Input Channel CSC

The CSC must be manually configured for each color space conversion. The CSC on the RX input channel can be enabled using the rx_csc_enable control. This CSC can run at 297MHz and provides color space conversion for UHD video formats. The CSC mode on the RX input channel can be configured using rx_csc_mode[1:0]. The CSC mode is used to define the fixed point position of the CSC coefficients which are located after rx_csc_mode[1:0] in the IO Map for the RX input channel.

Reference configuration scripts to configure the RX input channel CSC are provided with the evaluation software.

rx_csc_enable, IO Map, *Address 0x1B70[7]*
This bit is used to enable the RX input channel CSC.

Function

rx_csc_enable	Description
0 (default)	CSC disable
1	CSC enable

rx_csc_mode[1:0], IO Map, *Address 0x1B70[6:5]*
This signal is used to specify the CSC mode for the RX input channel CSC. The CSC mode sets the fixed point position of the CSC coefficients, including a4, b4, c4 and offsets.

Function

rx_csc_mode[1:0]	Description
00 (default)	+/- 1.0, -4096 to 4095
01	+/-2.0, -8192 to 8190
10	+/- 4.0, -16384 to 16380
11	+/- 4.0, -16384 to 16380

The characteristic equations for the secondary input CSC are provided in Equation 10, Equation 11 and Equation 12.

$$Out\_A = \left[In\_A * \frac{A1[12:0]}{4096} + In\_B * \frac{A2[12:0]}{4096} + In\_C * \frac{A3[12:0]}{4096} + A4[12:0] \right] * 2^{CSC\_scale}$$

Equation 10: RX Input CSC Channel A Output

$$Out\_B = \left[In\_A * \frac{B1[12:0]}{4096} + In\_B * \frac{B2[12:0]}{4096} + In\_C * \frac{B3[12:0]}{4096} + B4[12:0] \right] * 2^{CSC\_scale}$$

Equation 11: RX Input CSC Channel B Output

$$Out\_C = \left[In\_A * \frac{C1[12:0]}{4096} + In\_B * \frac{C2[12:0]}{4096} + In\_C * \frac{C3[12:0]}{4096} + C4[12:0] \right] * 2^{CSC\_scale}$$

Equation 12: RX Input CSC Channel C Output

The CSC on the RX input channel is illustrated in Figure 44.

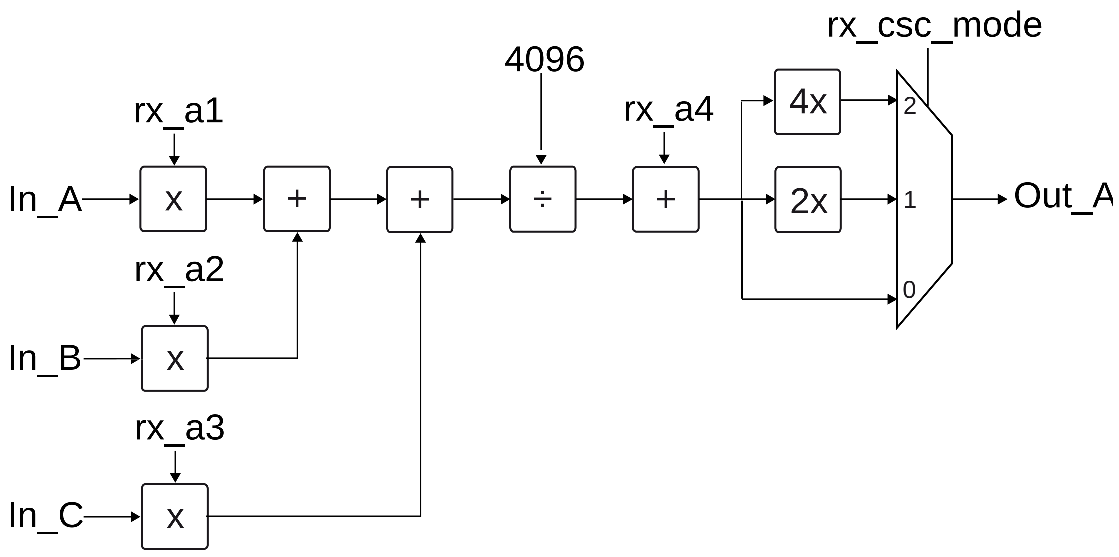


Figure 45: RX Input Channel CSC

The video inputs In_A, In_B and In_C are connected by default to R, G and B. For more information, please see Table 14. The default routing can be changed by adjusting the value of rx_swap_bus_ctrl[2:0].

Table 14: Default RX Input Channel CSC Signal Routing

Input Channel	Default RGB Routing	Default YCbCr Routing
In_A	R	Cr
In_B	G	Y
In_C	B	Cb

The A1 to A3, B1 to B3, and C1 to C3 coefficients are used to scale the primary inputs. A4, B4 and C4 are added as offsets. Floating point coefficients must be converted into 120-bit fixed decimal format then converted into binary format using twos complement for negative values and can only be programmed in the range [-1….+1] or [-4096….+4095].

The dynamic range of the CSC is [0…..1] for unipolar signals (Y, R, G, B) or [-0.5…….+0.5] for bipolar signals. Bipolar signals (Pr/Pb) must be offset to mid range. Equations with a dynamic range larger than 1 need to be scaled appropriately using the rx_csc_mode[1:0] control. To achieve a coefficient value of 1.0 for any given coefficient, rx_csc_mode[1:0] should be set high and the coefficient should be programmed to a value of 0.5. Otherwise, the largest value would be 4095/4096 = 0.9997. While this value could be interpreted as 1, it is recommended to use the value of 0.5 and set the rx_csc_mode[1:0] bits for maximum accuracy.
The CSC configurations for common modes are provided in Table 15.

Table 15: RX Input Channel CSC Common Configuration Coefficients

Color Space Conversion	csc_mode[1:0]	A1	A2	A3	A4	B1	B2	B3	B4	C1	C2	C3	C4
HDTV YCbCr (limited) to RGB (limited)	0x1	0x0C53	0x0800	0x0000	0x19D6	0x1C56	0x0800	0x1E88	0x0291	0x1FFF	0x0800	0x0E85	0x18BE
HDTV YCbCr (limited) to RGB (full)	0x2	0x0734	0x04AD	0x0000	0x1C1B	0x1DDC	0x04AD	0x1F24	0x0135	0x0000	0x04AD	0x087C	0x1B77
HDTV YCbCr (limited) to SDTV YCbCr (limited)	0x1	0x07DD	0x0000	0x1F6C	0x005B	0x0188	0x0800	0x00CB	0x1ED6	0x1F1D	0x0000	0x07EB	0x007B
HDTV YCbCr (limited) to SDTV YCbCr (full)	0x1	0x08EB	0x0000	0x1F58	0x1FDE	0x01C9	0x0950	0x00EC	0x1F25	0x1EFF	0x0000	0x08FA	0x031F
HDTV YCbCr (full) to SDTV YCbCr (limited)	0x0	0x0E0D	0x0000	0x0000	0x0100	0x0000	0x0DBC	0x0000	0x0100	0x0000	0x0000	0x0E0D	0x0100
SDTV YCbCr (limited) to RGB (limited)	0x1	0x0AF8	0x0800	0x0000	0x1A84	0x1A6A	0x0800	0x1D50	0x0423	0x1FFC	0x0800	0x0DDE	0x1913
SDTV YCbCr (limited) to RGB (full)	0x2	0x0669	0x04AC	0x0000	0x1C81	0x1CBC	0x04AD	0x1E6E	0x0220	0x1FFE	0x04AD	0x081A	0x1BA9
SDTV YCbCr (limited) to HDTV YCbCr (limited)	0x1	0x0833	0x0000	0x0099	0x1F99	0x1E56	0x0800	0x1F13	0x014B	0x00EA	0x0000	0x0826	0x1F78
SDTV YCbCr (limited) to SDTV YCbCr (full)	0x1	0x091B	0x0000	0x0000	0x1F6E	0x0000	0x0950	0x0000	0x1F6B	0x0000	0x0000	0x091B	0x1F6E
SDTV YCbCr (full) to HDTV YCbCr (limited)	0x2	0x039D	0x0000	0x0043	0x0F26	0x1F44	0x036F	0x1F97	0x00D2	0x0067	0x0000	0x0397	0x004D
RGB (limited) to HDTV YCbCr (limited)	0x0	0x082E	0x1893	0x1F3F	0x0800	0x0367	0x0B71	0x0128	0x0000	0x1E21	0x19B2	0x082D	0x0800
RGB (limited) to SDTV YCbCr (limited)	0x0	0x082E	0x1926	0x1EAC	0x0800	0x04C9	0x0965	0x01D2	0x0000	0x1D3F	0x1A93	0x082E	0x0800
RGB (limited) to RGB (full)	0x0	0x0DBC	0x0000	0x0000	0x0100	0x0000	0x0DBC	0x0000	0x0100	0x0000	0x0000	0x0DBC	0x0100
RGB (full) to HDTV YCbCr (limited)	0x0	0x06FF	0x19A6	0x1F5B	0x0800	0x02E9	0x09CB	0x00FD	0x0100	0x1E66	0x1A9B	0x06FF	0x0800
RGB (full) to SDTV YCbCr (limited)	0x0	0x06FF	0x1A24	0x1EDD	0x0800	0x0418	0x080A	0x018F	0x0100	0x1DA5	0x1B5C	0x06FF	0x0800
RGB (full) to RGB (limited)	0x1	0x0950	0x0000	0x0000	0x1F6B	0x0000	0x0950	0x0000	0x1F6B	0x0000	0x0000	0x0950	0x1F6B
Identity matrix (output = input)	0x1	0x0800	0x0000	0x0000	0x0000	0x0000	0x0800	0x0000	0x0000	0x0000	0x0000	0x0800	0x0000

2.2.12.4. TTL Output CSC

Models of ADV8005 which provide TTL output now have a CSC in that path, allowing, for example, theTTL output video to be converted to RGB. The TTL output CSC has the same structure as the primary input CSC but it is limited to a maximum pixel clock frequency of 162MHz. For higher pixels rates the HDMI TX should be used. The CSC must be manually configured for each color space conversion. The CSC on the TTL output channel can be enabled using the ttl_out_csc_enable control. The CSC mode on the TTL output channel can be configured using ttl_out_csc_mode[1:0]. The CSC mode is used to define the fixed point position of the CSC coefficients which are located after ttl_out_csc_mode[1:0] in the IO Map for the TTL output channel.

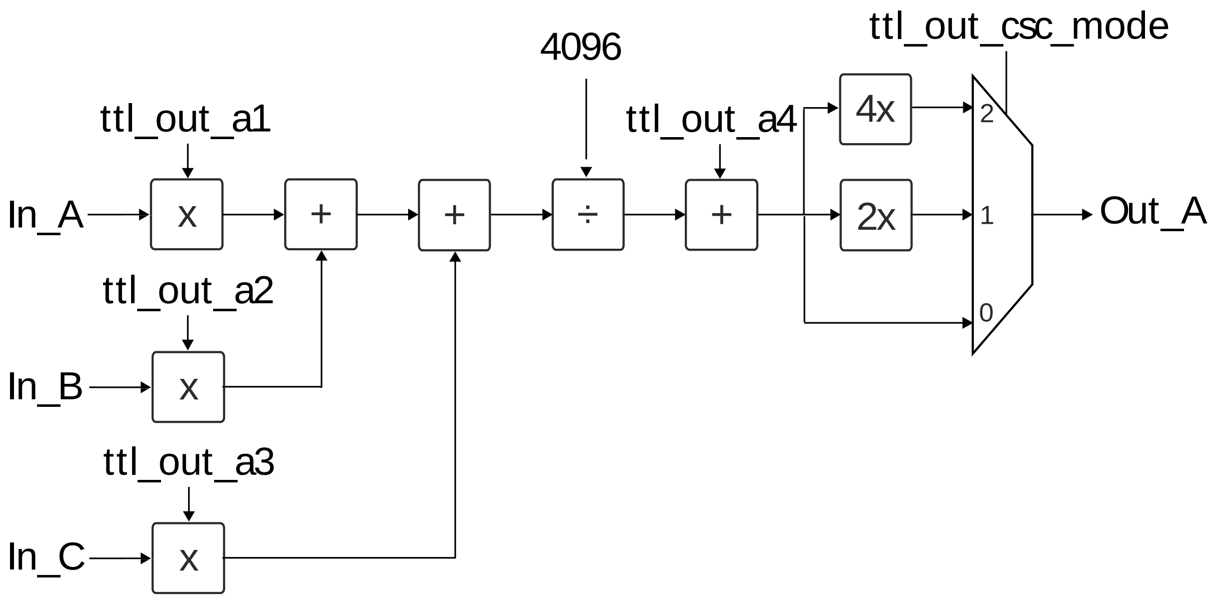


Figure 46 TTL Output Channel CSC

ttl_out_csc_enable, IO Map, *Address 0x1BB0[7]*
This bit is used to enable the ttl output channel CSC.

Function

ttl_out_csc_enable	**Description**
0 (default)	CSC disable
1	CSC enable

ttl_out_csc_mode[1:0], IO Map, *Address 0x1BB0[6:5]*
This signal is used to specify the CSC mode for the ttl output channel CSC. The CSC mode sets the fixed point position of the CSC coefficients, including a4, b4, c4 and offsets.

Function

ttl_out_csc_mode[1:0]	**Description**
00 (default)	+/- 1.0, -4096 to 4095
01	+/-2.0, -8192 to 8190
10	+/- 4.0, -16384 to 16380
11	+/- 4.0, -16384 to 16380

ttl_out_a1[12:0], IO Map, *Address 0x1BB0[4:0]; Address 0x1BB1[7:0]*
This signal is used to specify the ttl out channel CSC coefficient A1.

ttl_out_a2[12:0], IO Map, *Address 0x1BB2[4:0]; Address 0x1BB3[7:0]*
This signal is used to specify the ttl out channel CSC coefficient A2.

ttl_out_a3[12:0], IO Map, *Address 0x1BB4[4:0]; Address 0x1BB5[7:0]*
This signal is used to specify the ttl out channel CSC coefficient A3.

ttl_out_a4[12:0], IO Map, *Address 0x1BB6[4:0]; Address 0x1BB7[7:0]*
This signal is used to specify the ttl out channel CSC coefficient A4.

ttl_out_b1[12:0], IO Map, *Address 0x1BB8[4:0]; Address 0x1BB9[7:0]*
This signal is used to specify the ttl out channel CSC coefficient B1.

ttl_out_b2[12:0], IO Map, *Address 0x1BBA[4:0]; Address 0x1BBB[7:0]*
This signal is used to specify the ttl out channel CSC coefficient B2.

ttl_out_b3[12:0], IO Map, *Address 0x1BBC[4:0]; Address 0x1BBD[7:0]*
This signal is used to specify the ttl out channel CSC coefficient B3.

ttl_out_b4[12:0], IO Map, *Address 0x1BBE[4:0]; Address 0x1BBF[7:0]*
This signal is used to specify the ttl out channel CSC coefficient B4.

ttl_out_c1[12:0], IO Map, *Address 0x1BC0[4:0]; Address 0x1BC1[7:0]*
This signal is used to specify the ttl out channel CSC coefficient C1.

ttl_out_c2[12:0], IO Map, *Address 0x1BC2[4:0]; Address 0x1BC3[7:0]*
This signal is used to specify the ttl out channel CSC coefficient C2.

ttl_out_c3[12:0], IO Map, *Address 0x1BC4[4:0]; Address 0x1BC5[7:0]*
This signal is used to specify the ttl out channel CSC coefficient C3.

ttl_out_c4[12:0], IO Map, *Address 0x1BC6[4:0]; Address 0x1BC7[7:0]*
This signal is used to specify the ttl out channel CSC coefficient C4.

The characteristic equations for the secondary input CSC are provided in Equation 10, Equation 11 and Equation 12.

$$Out\_A = \left[In\_A * \frac{A1[12:0]}{4096} + In\_B * \frac{A2[12:0]}{4096} + In\_C * \frac{A3[12:0]}{4096} + A4[12:0] \right] * 2^{CSC\_scale}$$

Equation 13: TTL Output CSC Channel A Output

$$Out\_B = \left[In\_A * \frac{B1[12:0]}{4096} + In\_B * \frac{B2[12:0]}{4096} + In\_C * \frac{B3[12:0]}{4096} + B4[12:0] \right] * 2^{CSC\_scale}$$

Equation 14: TTL Output CSC Channel B Output

$$Out\_C = \left[In\_A * \frac{C1[12:0]}{4096} + In\_B * \frac{C2[12:0]}{4096} + In\_C * \frac{C3[12:0]}{4096} + C4[12:0] \right] * 2^{CSC\_scale}$$

Equation 15: TTL Output CSC Channel C Output

Table 16 TTL Output Channel CSC Common Configuration Coefficients

Color Space Conversion	csc_mode[1:0]	A1	A2	A3	A4	B1	B2	B3	B4	C1	C2	C3	C4
HDTV YCbCr (limited) to RGB (limited)	0x1	0x0C53	0x0800	0x0000	0x19D6	0x1C56	0x0800	0x1E88	0x0291	0x1FFF	0x0800	0x0E85	0x18BE
HDTV YCbCr (limited) to RGB (full)	0x2	0x0734	0x04AD	0x0000	0x1C1B	0x1DDC	0x04AD	0x1F24	0x0135	0x0000	0x04AD	0x087C	0x1B77
HDTV YCbCr (limited) to SDTV YCbCr (limited)	0x1	0x07DD	0x0000	0x1F6C	0x005B	0x0188	0x0800	0x00CB	0x1ED6	0x1F1D	0x0000	0x07EB	0x007B
HDTV YCbCr (limited) to SDTV YCbCr (full)	0x1	0x08EB	0x0000	0x1F58	0x1FDE	0x01C9	0x0950	0x00EC	0x1F25	0x1EFF	0x0000	0x08FA	0x031F
HDTV YCbCr (full) to SDTV YCbCr (limited)	0x0	0x0E0D	0x0000	0x0000	0x0100	0x0000	0x0DBC	0x0000	0x0100	0x0000	0x0000	0x0E0D	0x0100
SDTV YCbCr (limited) to RGB (limited)	0x1	0x0AF8	0x0800	0x0000	0x1A84	0x1A6A	0x0800	0x1D50	0x0423	0x1FFC	0x0800	0x0DDE	0x1913
SDTV YCbCr (limited) to RGB (full)	0x2	0x0669	0x04AC	0x0000	0x1C81	0x1CBC	0x04AD	0x1E6E	0x0220	0x1FFE	0x04AD	0x081A	0x1BA9
SDTV YCbCr (limited) to HDTV YCbCr (limited)	0x1	0x0833	0x0000	0x0099	0x1F99	0x1E56	0x0800	0x1F13	0x014B	0x00EA	0x0000	0x0826	0x1F78
SDTV YCbCr (limited) to SDTV YCbCr (full)	0x1	0x091B	0x0000	0x0000	0x1F6E	0x0000	0x0950	0x0000	0x1F6B	0x0000	0x0000	0x091B	0x1F6E
SDTV YCbCr (full) to HDTV YCbCr (limited)	0x2	0x039D	0x0000	0x0043	0x0F26	0x1F44	0x036F	0x1F97	0x00D2	0x0067	0x0000	0x0397	0x004D
RGB (limited) to HDTV YCbCr (limited)	0x0	0x082E	0x1893	0x1F3F	0x0800	0x0367	0x0B71	0x0128	0x0000	0x1E21	0x19B2	0x082D	0x0800
RGB (limited) to SDTV YCbCr (limited)	0x0	0x082E	0x1926	0x1EAC	0x0800	0x04C9	0x0965	0x01D2	0x0000	0x1D3F	0x1A93	0x082E	0x0800
RGB (limited) to RGB (full)	0x0	0x0DBC	0x0000	0x0000	0x0100	0x0000	0x0DBC	0x0000	0x0100	0x0000	0x0000	0x0DBC	0x0100
RGB (full) to HDTV YCbCr (limited)	0x0	0x06FF	0x19A6	0x1F5B	0x0800	0x02E9	0x09CB	0x00FD	0x0100	0x1E66	0x1A9B	0x06FF	0x0800
RGB (full) to SDTV YCbCr (limited)	0x0	0x06FF	0x1A24	0x1EDD	0x0800	0x0418	0x080A	0x018F	0x0100	0x1DA5	0x1B5C	0x06FF	0x0800
RGB (full) to RGB (limited)	0x1	0x0950	0x0000	0x0000	0x1F6B	0x0000	0x0950	0x0000	0x1F6B	0x0000	0x0000	0x0950	0x1F6B
Identity Matrix (Output = Input)	0x1	0x0800	0x0000	0x0000	0x0000	0x0000	0x0800	0x0000	0x0000	0x0000	0x0000	0x0800	0x0000

2.2.12.5. *HDMI Transmitter CSCs*

Both of the HDMI transmitters feature an any-to-any CSC. The CSC register controls for HDMI Tx1 are described here; the same controls co-exist in the HDMI Tx2 Main Map for the HDMI Tx2 CSC.

The CSC must be manually configured for each color space conversion. The HDMI Tx CSC output can be enabled using the csc_en control. The HDMI Tx CSC mode can be configured using csc_scaling_factor[1:0]. The CSC mode is used to define the fixed point position of the CSC coefficients which are located after csc_scaling_factor[1:0] in the TX Main Map.

Reference configuration scripts to configure the HDMI Tx CSCs are provided with the evaluation software.

csc_en, TX2 Main Map, *Address 0xF418[7]*
This bit is used to enable the colour space converter.

Function

csc_en	Description
0 (default)	CSC Disabled
1	CSC Enabled

csc_scaling_factor[1:0], TX2 Main Map, *Address 0xF418[6:5]*
This signal is used to specify the CSC scaling factor. The CSC scaling factor sets the fixed point position of the CSC coefficients, including a4, b4, c4 and offsets.

Function

csc_scaling_factor[1:0]	Description
00	+/- 1.0, -4096 to 4095
01	+/-2.0, -8192 to 8190
10 (default)	+/- 4.0, -16384 to 16380
11	+/- 4.0, -16384 to 16380

The characteristic equations for the HDMI Tx CSCs are captured in Equation 16, Equation 17 and Equation 18.

$$Out\_A = \left[In\_A * \frac{A1[12:0]}{4096} + In\_B * \frac{A2[12:0]}{4096} + In\_C * \frac{A3[12:0]}{4096} + A4[12:0] \right] * 2^{CSC\_scale}$$

Equation 16: HDMI Tx CSC Channel A Output

$$Out\_B = \left[In\_A * \frac{B1[12:0]}{4096} + In\_B * \frac{B2[12:0]}{4096} + In\_C * \frac{B3[12:0]}{4096} + B4[12:0] \right] * 2^{CSC\_scale}$$

Equation 17: HDMI Tx CSC Channel B Output

$$Out\_C = \left[In\_A * \frac{C1[12:0]}{4096} + In\_B * \frac{C2[12:0]}{4096} + In\_C * \frac{C3[12:0]}{4096} + C4[12:0] \right] * 2^{CSC\_scale}$$

Equation 18: HDMI Tx CSC Channel C Output

The CSC in each of the HDMI Txs is illustrated in Figure 44.

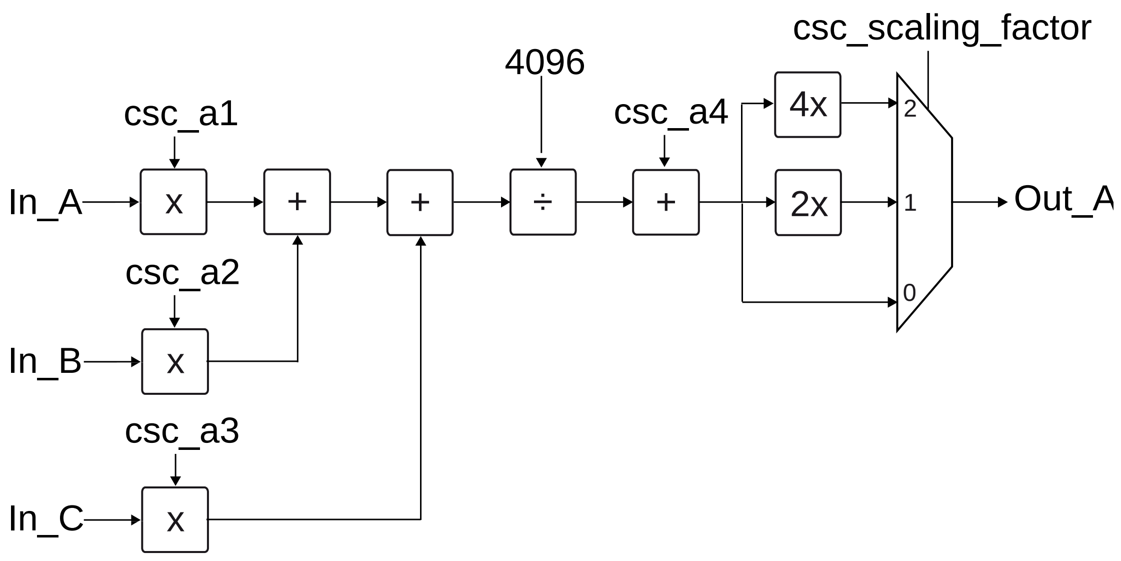


Figure 47: HDMI Tx CSC

The video inputs In_A, In_B and In_C are connected by default to R, G and B. Refer to Table 17 for more information. The default routing cannot be changed for the HDMI Tx CSCs.

Table 17: Default HDMI Tx Channel CSC Signal Routing

Input Channel	**Default RGB Routing**	**Default YCbCr Routing**
In_A	R	Cr
In_B	G	Y
In_C	B	Cb

The A1 to A3, B1 to B3, and C1 to C3 coefficients are used to scale the primary inputs. A4, B4 and C4 are added as offsets. Floating point coefficients must be converted into 120-bit fixed decimal format then converted into binary format using twos complement for negative values and can only be programmed in the range [-1….+1] or [-4096….+4095].

The dynamic range of the CSC is [0…..1] for unipolar signals (Y, R, G, B) or [-0.5…….+0.5] for bipolar signals. Bipolar signals (Pr/Pb) must be offset to mid-range. Equations with a dynamic range larger than 1 need to be scaled appropriately using the csc_scaling_factor[1:0] control. To achieve a coefficient value of 1.0 for any given coefficient, csc_scaling_factor[1:0] should be set high and the coefficient should be programmed to a value of 0.5. Otherwise, the largest value would be 4095/4096 = 0.9997. While this value could be interpreted as 1, it is recommended to use the value of 0.5 and set the csc_scaling_factor[1:0] bits for maximum accuracy.

The CSC configurations for common modes are provided in Table 18.

Table 18: HDMI Tx CSC Common Configuration Coefficients

Color Space Conversion	csc_mode[1:0]	A1	A2	A3	A4	B1	B2	B3	B4	C1	C2	C3	C4
HDTV YCbCr (limited) to RGB (limited)	0x1	0x0C53	0x0800	0x0000	0x19D6	0x1C56	0x0800	0x1E88	0x0291	0x1FFF	0x0800	0x0E85	0x18BE
HDTV YCbCr (limited) to RGB (full)	0x2	0x0734	0x04AD	0x0000	0x1C1B	0x1DDC	0x04AD	0x1F24	0x0135	0x0000	0x04AD	0x087C	0x1B77
HDTV YCbCr (limited) to SDTV YCbCr (limited)	0x1	0x07DD	0x0000	0x1F6C	0x005B	0x0188	0x0800	0x00CB	0x1ED6	0x1F1D	0x0000	0x07EB	0x007B
HDTV YCbCr (limited) to SDTV YCbCr (full)	0x1	0x08EB	0x0000	0x1F58	0x1FDE	0x01C9	0x0950	0x00EC	0x1F25	0x1EFF	0x0000	0x08FA	0x031F
HDTV YCbCr (full) to SDTV YCbCr (limited)	0x0	0x0E0D	0x0000	0x0000	0x0100	0x0000	0x0DBC	0x0000	0x0100	0x0000	0x0000	0x0E0D	0x0100
SDTV YCbCr (limited) to RGB (limited)	0x1	0x0AF8	0x0800	0x0000	0x1A84	0x1A6A	0x0800	0x1D50	0x0423	0x1FFC	0x0800	0x0DDE	0x1913
SDTV YCbCr (limited) to RGB (Full)	0x2	0x0669	0x04AC	0x0000	0x1C81	0x1CBC	0x04AD	0x1E6E	0x0220	0x1FFE	0x04AD	0x081A	0x1BA9
SDTV YCbCr (limited) to HDTV YCbCr (limited)	0x1	0x0833	0x0000	0x0099	0x1F99	0x1E56	0x0800	0x1F13	0x014B	0x00EA	0x0000	0x0826	0x1F78
SDTV YCbCr (limited) to SDTV YCbCr (full)	0x1	0x091B	0x0000	0x0000	0x1F6E	0x0000	0x0950	0x0000	0x1F6B	0x0000	0x0000	0x091B	0x1F6E
SDTV YCbCr (full) to HDTV YCbCr (limited)	0x2	0x039D	0x0000	0x0043	0x0F26	0x1F44	0x036F	0x1F97	0x00D2	0x0067	0x0000	0x0397	0x004D
RGB (limited) to HDTV YCbCr (limited)	0x0	0x082E	0x1893	0x1F3F	0x0800	0x0367	0x0B71	0x0128	0x0000	0x1E21	0x19B2	0x082D	0x0800
RGB (limited) to SDTV YCbCr (limited)	0x0	0x082E	0x1926	0x1EAC	0x0800	0x04C9	0x0965	0x01D2	0x0000	0x1D3F	0x1A93	0x082E	0x0800
RGB (limited) to RGB (full)	0x0	0x0DBC	0x0000	0x0000	0x0100	0x0000	0x0DBC	0x0000	0x0100	0x0000	0x0000	0x0DBC	0x0100
RGB (full) to HDTV YCbCr (limited)	0x0	0x06FF	0x19A6	0x1F5B	0x0800	0x02E9	0x09CB	0x00FD	0x0100	0x1E66	0x1A9B	0x06FF	0x0800
RGB (full) to SDTV YCbCr (limited)	0x0	0x06FF	0x1A24	0x1EDD	0x0800	0x0418	0x080A	0x018F	0x0100	0x1DA5	0x1B5C	0x06FF	0x0800
RGB (full) to RGB (limited)	0x1	0x0950	0x0000	0x0000	0x1F6B	0x0000	0x0950	0x0000	0x1F6B	0x0000	0x0000	0x0950	0x1F6B
Identity Matrix (Output = Input)	0x1	0x0800	0x0000	0x0000	0x0000	0x0000	0x0800	0x0000	0x0000	0x0000	0x0000	0x0800	0x0000

2.2.13. *VGA Position and Phase Information*

The ADV8005 can measure picture position and sample quality information and record these in on-chip registers. This information can be read and used by the software on an external MCU to program the optimum sampling clock frequency and phase for an external Video AFE when it is sampling a VGA-type input signal.

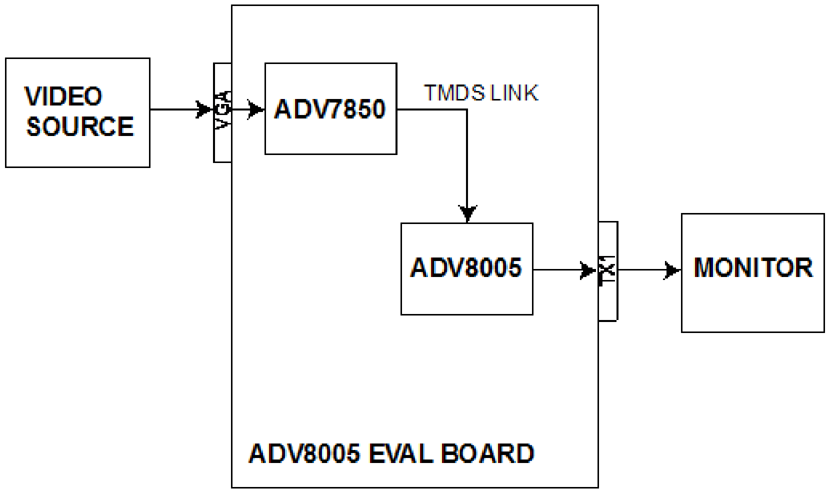


Figure 48: Autoposition and Phase Block Diagram

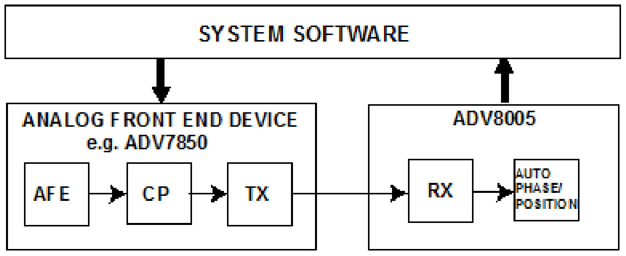


Figure 49: Autopostion and Phase Implementation Using System Firmware

The autophase block within the ADV8005 is designed to tune the ADC sampling phase in a device with an analog front end such as the ADV7850, ADV7844, or ADV7842. Figure 48 shows a system view of how the ADV8005 implements autoposition and phase. Figure 49 shows a block diagram of how the ADV8005 interfaces with the system software to tune the analog front end device, the ADV7850. For the autophase the software driver cycles through each of the ADV7850 ADC sampling phases. The ADV8005 analyzes the input video timing and then rb_auto_ph_right_phase[5:0] indicates the best sampling phase to use. The software routine required to implement this routing is decribed in Figure 50.

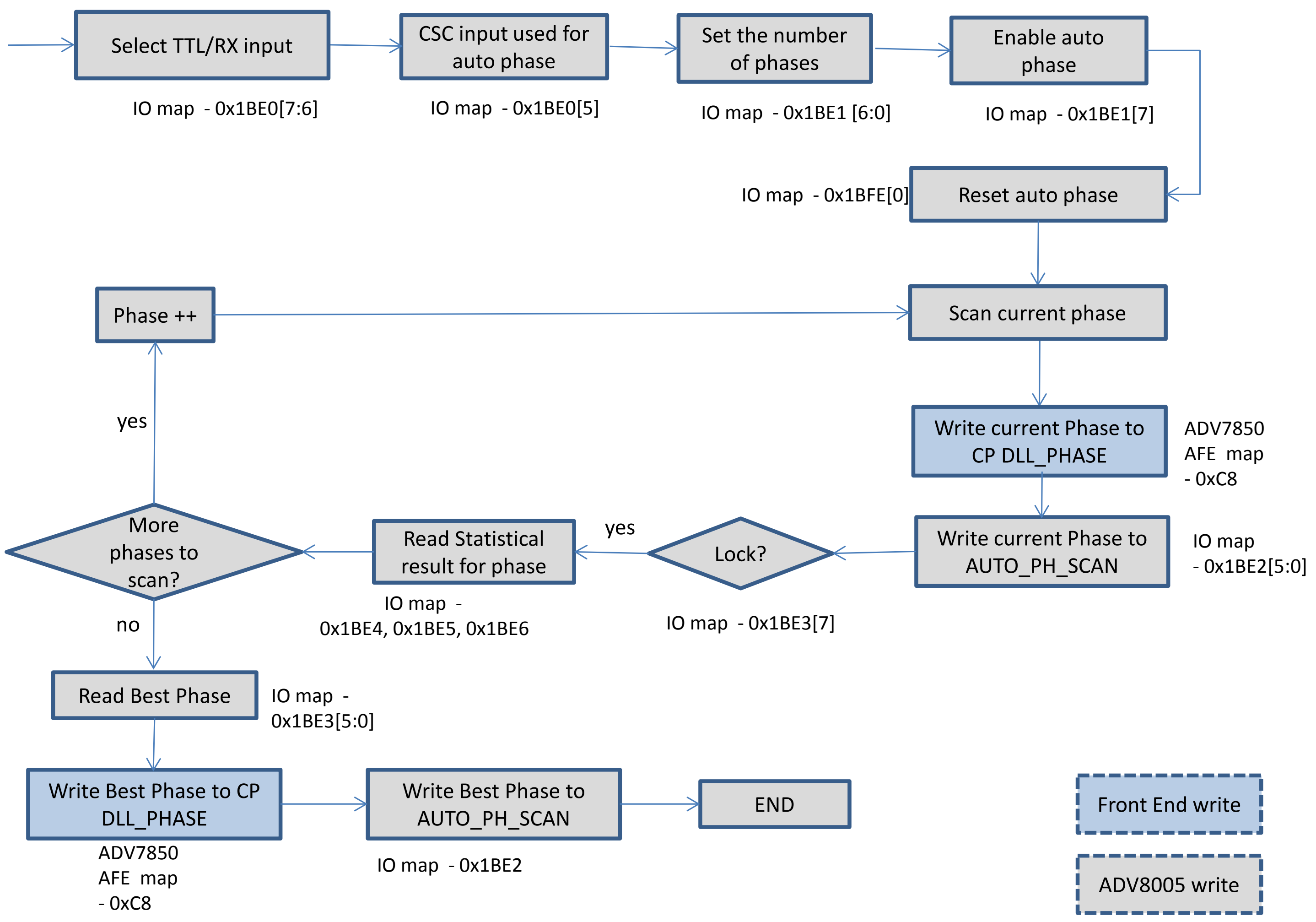


Figure 50 ADV8005 Auto-Phase Software Flow Chart

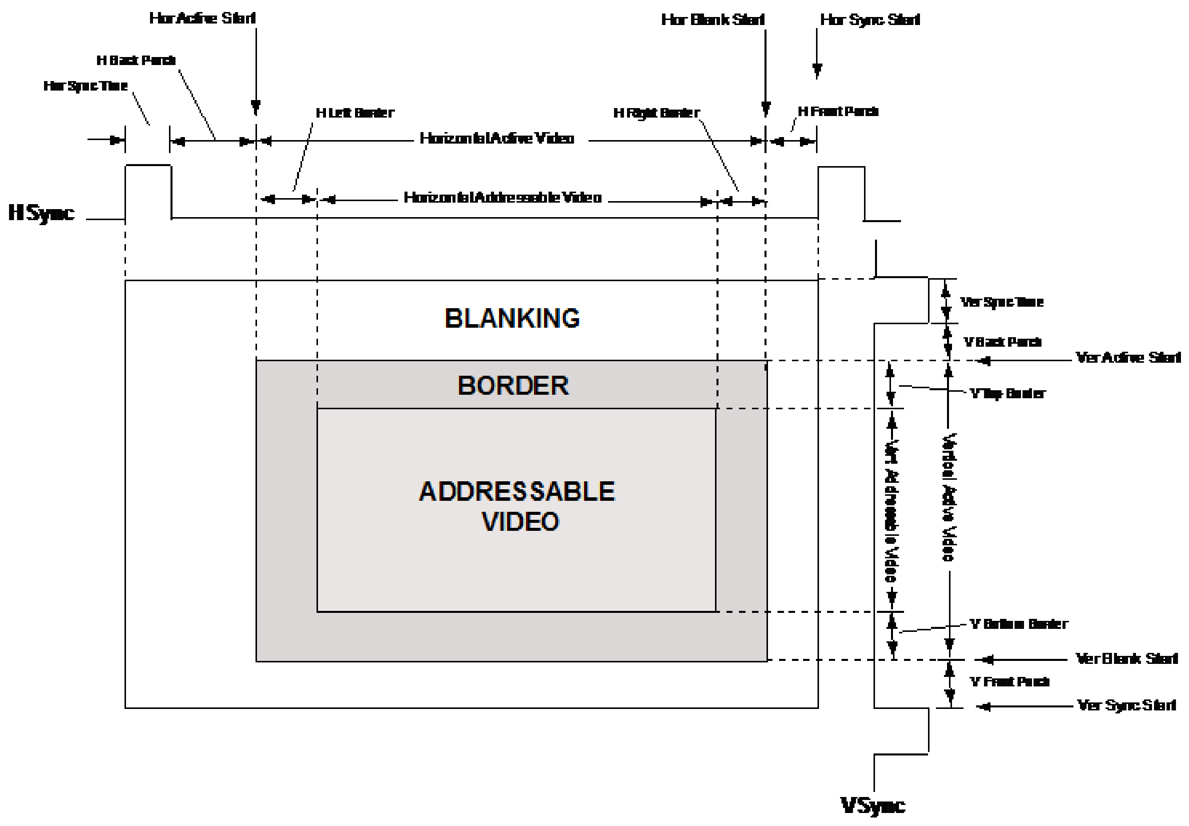


Figure 51: Graphics Video Timing Parameters

Similar to the autophase, the auto position block is designed to tune the ADC sampling clock frequency in a device with an analog front end. To carry it out, the block will analyse the graphics input and return the top, bottom, left and right pixel vacancy numbers. This information, along with the input standard format, can then be used to adjust the ADC sampling clock frequency.
Figure 51 shows the timing parameters for graphics inputs.

It returns the number of pixels from:

- End of HSync to the start of active video
- End of Active video to start of HSync

The autoposition also returns the number of lines from:

- End of VSync to start of active video
- End of active video to start of VSync

The readbacks are updated with every VSync period. It is required that the input to the autoposition and autophase blocks is in RGB format. If the input is in YCrCb format, the auto_phpo_byp_csc must be cleared, enabling the ADV8005 to perform the color space conversion to RGB. If the input is RGB, then the CSC should be bypassed.

Before running the auto-position software routine, ensure a bright test pattern is used. For example a white RGB flat field which will have valid video on the first and last pixel in each line. The bright color makes it easier for the algorithm to detect the blank area.

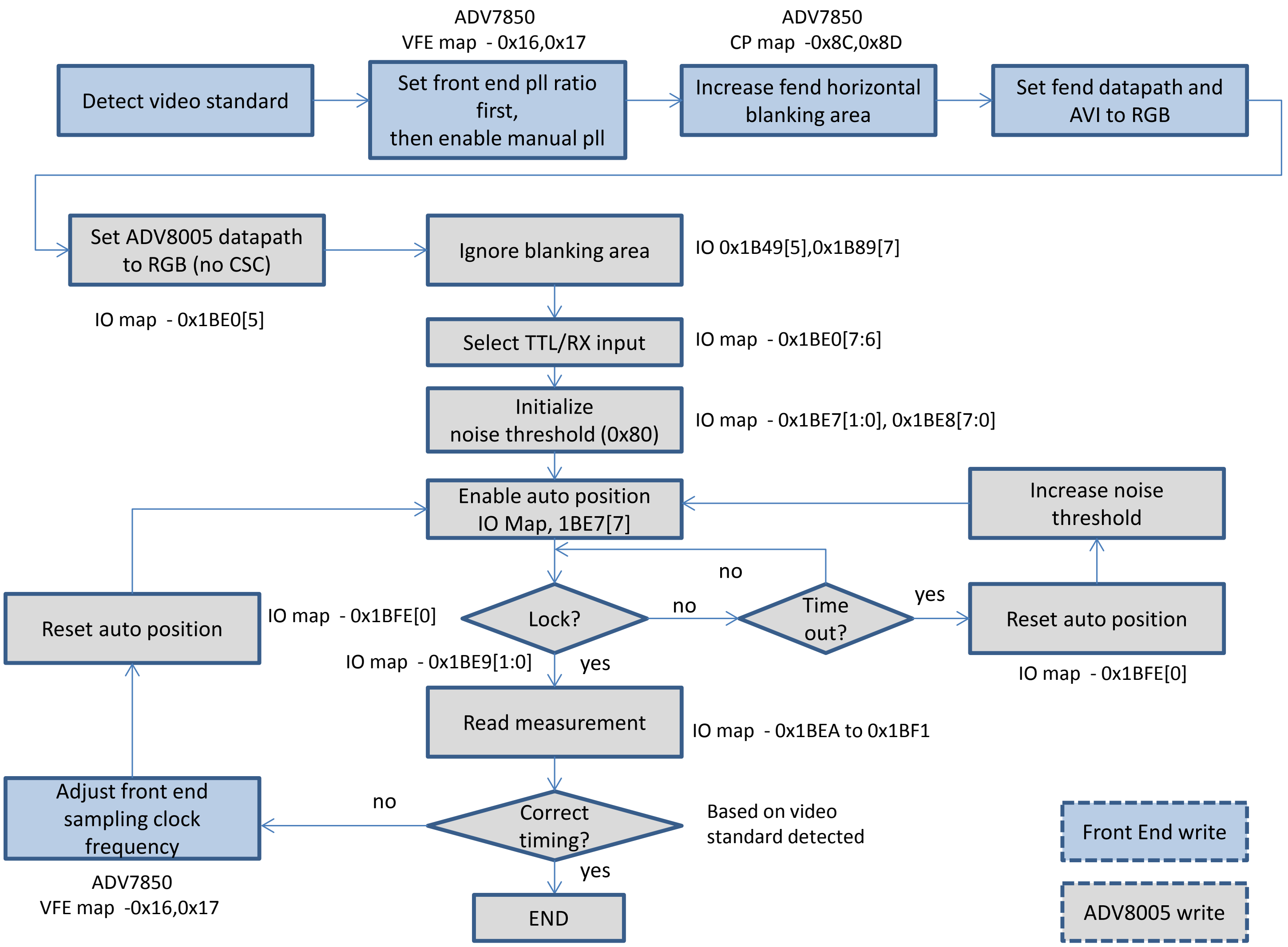


Figure 52 ADV8005 Auto-Position Software Flow Chart

auto_phpo_inp_sel[1:0], IO Map, *Address 0x1BE0[7:6]*
This control signal is used to select which input is routed to the auto position and auto phase blocks

Function

auto_phpo_inp_sel[1:0]	**Description**
00 (default)	VID TTL
01	OSD TTL
10	RX
11	N/A

auto_phpo_byp_csc, IO Map, *Address 0x1BE0[5]*
This bit is used to bypass the CSC or not before routing to the auto Phase and auto Position detection blocks

Function

auto_phpo_byp_csc	**Description**
0	CSC output used for auto PHPO
1 (default)	CSC input used for auto PHPO

auto_ph_en, IO Map, *Address 0x1BE1[7]*
This bit is used to enable auto phase detection block

Function

auto_ph_en	Description
0 (default)	Disabled
1	Enabled

auto_ph_num[6:0], IO Map, *Address 0x1BE1[6:0]*
This control signal sets the total number of phases available on the front end part, e.g. 8, 16, 32, etc

auto_ph_scan[5:0], IO Map, *Address 0x1BE2[5:0]*
This control signal sets the scan phase number being tested. When the scan value changes, a new scan is triggered to start.

rb_auto_ph_read_ready, IO Map, *Address 0x1BE3[7] (Read Only)*
This bit is used to indicate rb_auto_ph_diff_sum_lock is valid, a HIGH means it is valid to read the value in auto_ph_diff_sum_lock.

rb_auto_ph_right_phase[5:0], IO Map, *Address 0x1BE3[5:0] (Read Only)*
This signal is used to indicate the correct phase after i2c_auto_ph_scan has been indexed through all of the phases.

rb_auto_ph_diff_sum_lock[23:0], IO Map, *Address 0x1BE4[7:0]; Address 0x1BE5[7:0]; Address 0x1BE6[7:0] (Read Only)*
This signal is used to indicate the statistical result for the phase in AUTO_PH_SCAN (Auto Phase Scan Number). This signal is valid when AUTO_PH_READ_READY is HIGH.

auto_po_en, IO Map, *Address 0x1BE7[7]*
This bit is used to enable the auto position detection block

Function

auto_po_en	Description
0 (default)	Disabled
1	Enabled

vid_blank_blanking_area, IO Map, *Address 0x1B49[5]*
This bit is used to specify the blanking area that is blanked to avoid the filters mistakenly interpreting data in the blanking area.

Function

vid_blank_blanking_area	Description
1 (default)	Blanking area is blanked.
0	Blanking area data passes through.

rx_blank_blanking_area, IO Map, *Address 0x1B89[7]*
This bit is used to specify the blanking area that is blanked to avoid the filters mistakenly interpreting data in the blanking area.

Function

rx_blank_blanking_area	Description
1	Blanking area is blanked.
0 (default)	Blanking area data passes through.

auto_po_noise_thr[9:0], IO Map, *Address 0x1BE7[1:0]; Address 0x1BE8[7:0]*
This signal sets the noise threshold (minimum value) for the sum of the three channels R, G and B to differentiate the active pixels from the blank pixels. For example, if blank value for RGB is 16, the noise threshold should be larger than 48.

rb_auto_po_l_edg_lock_flag, IO Map, *Address 0x1BE9[1] (Read Only)*
This bit indicates if the algorithm has locked to the left edge of the input video. If this bit is high, it has locked to the left edge, a low indicates it has not locked to it.
rb_auto_po_r_edg_lock_flag, IO Map, *Address 0x1BE9[0] (Read Only)*
This bit indicates if the algorithm has locked to the right edge of the input video. If this bit is high, it has locked to the right edge, a low indicates it has not locked to it.

rb_auto_po_t_offset[15:0], IO Map, *Address 0x1BEA[7:0]; Address 0x1BEB[7:0] (Read Only)*
This readback signal returns the top offset, the number of blank lines before the start of active video. This offset excludes the vertical blanking area.

rb_auto_po_b_offset[15:0], IO Map, *Address 0x1BEC[7:0]; Address 0x1BED[7:0] (Read Only)*
This readback signal returns the bottom offset, the number of blank lines after active video. This offset excludes the vertical blanking area.

rb_auto_po_l_offset[15:0], IO Map, *Address 0x1BEE[7:0]; Address 0x1BEF[7:0] (Read Only)*
This readback signal returns the left offset, the number of blank Pixels before the start of active video. This offset excludes the horizontal blanking area.

rb_auto_po_r_offset[15:0], IO Map, *Address 0x1BF0[7:0]; Address 0x1BF1[7:0] (Read Only)*
This readback signal returns the right offset, the number of blank Pixels after active video. This offset excludes the horizontal blanking area.

2.2.14. *ADV8005 Silicon Revision*

The ADV8005 silicon revision can be determined using rb_chip_id[16].

rb_chip_id[16], IO Map, *Address 0x1AD3[0] (Read Only)*
Readback of Macrovision enabled / disabled

Function

rb_chip_id[16]	Description
0 (default)	Rovi Enabled
1	Rovi Disabled

2.2.15. *System Configuration*

When configuring a system featuring an HDMI Rx and ADV8005, the following sequences for HDMI Tx and encoder are recommended.

For HDMI Tx:

1. Configure the HDMI Rx (ADV7850).
2. Wait until the ADV8005 Serial Video Rx achieves lock.
3. Wait 100 ms.
4. Configure the VSP.
5. Wait 1 field/frame.
6. Configure the HDMI Tx.

For the encoder:

1. Configure the HDMI Rx (ADV7850).
2. Wait until the ADV8005 Serial Video Rx achieves lock.
3. Wait 100 ms.
4. Configure the VSP.
5. Wait 250 ms.
6. Configure the encoder.

3. VIDEO SIGNAL PROCESSING

3.1. INTRODUCTION

The primary function of the ADV8005 is high performance video processing, such as motion adaptive de-interlacing, flexible scaling and frame rate conversion, as well as additional video processing such as noise reduction, CUE correction, and aspect ratio/panorama scaling.

This section details the registers used to control the Video Signal Processing (VSP) hardware.

The three constituent sections of the ADV8005 video processor are the PVSP, SVSP, and the PtoI converter. These hardware blocks are completely independent of each other and can be placed in various configurations within the ADV8005.

Access to an external DDR2 memory can be required for the PVSP and SVSP to operate correctly. The PVSP needs access to external DDR2 memory in every mode except game mode. While the SVSP uses external DDR2 memory for the majority of operations, in the case of down converting from 1080p to 720p (with the same frame rate), no external memory is required and all conversions can take place in internal line memories. The PtoI converter does not need access to external DDR2 memory.

3.2. PRIMARY VSP

3.2.1. *Introduction to PVSP*

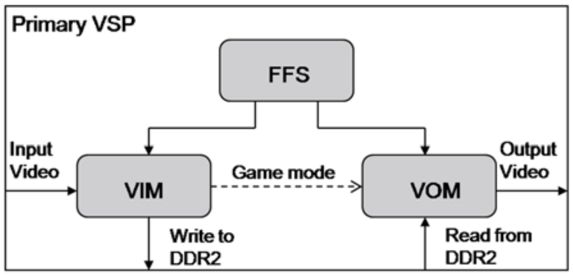


Figure 53: ADV8005 PVSP

Figure 53 shows the structure of the PVSP which comprises three sections: the Video Input Module (VIM), the Video Output Module (VOM), and a controller referred to as the Field Frame Scheduler (FFS).

The VIM is used to capture input video data which it then writes to external DDR2 memory. The VIM is also capable of cropping input video data and performing horizontal downscaling. Before the VIM writes video data to external memory, it first packs the video into the appropriate data formats. In game mode, VIM will send packed 128-bit words to VOM directly instead of writing them into external memory.

The VOM is used to read data from external memory, format this data into 12-bit pixels, perform various functions on this data (scaling, de-interlacing, and so on), and then output this video from the PVSP. Many of the PVSP video processing functions are implemented in the VOM. In game mode, the VOM will use data from the VIM instead of reading data from external memory.

The FFS is used to schedule and control the interaction between the VIM, external DDR2 memory, and the VOM. Field/frame buffer scheduling, field polarity management, and FRC management are all implemented in the FFS.

The PVSP can be bypassed by setting pvsp_bypass.

pvsp_bypass, Primary VSP Map, *Address 0xE829[7]*
This bit is used to bypass the Primary VSP. If this bit is set to 1, the input video to the Primary VSP will be directly bypassed to the output port.

Function

pvsp_bypass	**Description**
0 (default)	Not bypass Primary VSP
1	Bypass Primary VSP

The VIM and VOM must be enabled if using the PVSP. This can be done by enabling the pvsp_enable_vim and pvsp_enable_vom bits. This must be done regardless of the video conversions being performed.

pvsp_enable_vim, Primary VSP Map, *Address 0xE828[1]*
This bit is used to control the Video Input Module (VIM). If this bit is set to 1, the VIM is enabled to write packed input video data into a defined external field/frame buffer. While the Primary VSP is running, if this bit is set to 0, the output video stream will be frozen.

Function

pvsp_enable_vim	**Description**
0 (default)	Disable VIM
1	Enable VIM

pvsp_enable_vom, Primary VSP Map, *Address 0xE828[2]*
This bit is used to control the Video Output Module (VOM). If this bit is set to 1, the VOM is enabled to read video data from external memory, process it and then output it.

Function

pvsp_enable_vom	**Description**
0 (default)	Disable VOM
1	Enable VOM

Also, if using the PVSP, the FFS must be enabled using pvsp_enable_ffs. This informs the hardware of the various conversions that must be performed. Field/frame buffers in external memory are managed by the FFS which decides which field/frame buffer should be used by the VIM to store input video data. The FFS also decides which field/frame buffer should be read back by VOM to process. In the case of interlaced video, the FFS informs the VOM if the input video is the even field or the odd field. The PVSP utilizes a frame repeat/drop mechanism to implement FRC, which is also managed by the FFS.

pvsp_enable_ffs, Primary VSP Map, *Address 0xE828[0]*
This bit is used to control the Field Frame Scheduler (FFS). If this bit is set to 1, the FFS is enabled and the VIM and VOM are scheduled by the FFS, which means the Primary VSP is in operating mode. If this bit is set to 0, the Primary VSP is in idle mode.

Function

pvsp_enable_ffs	**Description**
0 (default)	Disable FFS/FRC
1	Enable FFS/FRC

3.2.1.1. *Autoconfiguration*

Each block inside VIM and VOM can be automatically configured to reduce the configuration complexity. Two registers, pvsp_autocfg_input_vid[7:0] and pvsp_autocfg_output_vid[7:0] should be set to make the auto configuration work.

The 59.94/23.97 Hz timings have the same VID as the corresponding 60/24Hz timing in Figure 20.

pvsp_autocfg_input_vid[7:0], Primary VSP Map, *Address 0xE881[7:0]*
This register is used to set the input timing VIC. If this register is 0, PVSP will use values in registers of pvsp_vin_h, pvsp_vin_v, pvsp_is_i_to_p and pvsp_vin_fr to set input video.

Function

pvsp_autocfg_input_vid[7:0]	**Description**
0x06 (default)	Default: 480i@60
0xXX	Input timing VID

Table 19: PVSP Supported Input Video Timing and VID

Video Timing		VID
CEA	640x480p60	1
	720x480p60	2 or 3 or 14 or 15 or 35 or 36
	720x240p60	8 or 9 or 12 or 13
	1280x720p60	4
	1920x1080i60	5
	720x480i60	6 or 7 or 10 or 11
	1920x1080p	16
	720x576p50	17 or 18 or 29 or 30 or 37 or 38
	1280x720p50	19
	1920x1080i50	20
	720x576i50	21 or 22 or 25 or 26
	720x288p50	23 or 24 or 27 or 28
	1920x1080p50	31
	1920x1080p24	32
	1920x1080p25	33
	1920x1080p30	34
	1080i50-even	39
	1080i100	40
	720p100	41
	576p100	42 or 43
	576i100	44 or 45
	1080i120	46
	720p120	47
	480p120	48 or 49
	480i120	50 or 51
	576p200	52 or 53
	576i200	54 or 55
	480p240	56 or 57
	480i240	58 or 59
VESA timing	VGA	200
	SVGA	201
	XGA	202
	WXGA	203
	SXGA	204
	WXGA-2	205
	UXGA	206
	WXGA-3	207
	WUXGA	208

pvsp_autocfg_output_vid[7:0], Primary VSP Map, *Address 0xE882[7:0]*
This register is used to set the output timing VIC. If this register is 0, PVSP will use values in registers of pvsp_dp_decount, pvsp_dp_hfrontporch, pvsp_dp_hsynctime, pvsp_dp_hbackporch, pvsp_dp_activeline, pvsp_dp_vfrontporch, pvsp_dp_vsynctime, pvsp_dp_vbackporch, pvsp_dp_hpolarity, pvsp_dp_vpolarity, pvsp_vout_fr and pvsp_dp_4kx2k_mode_en to set output video.

Function

pvsp_autocfg_output_vid[7:0]	**Description**
0x10 (default)	Default: 1080p@60
0xXX	Output timing VID

Table 20 lists the supported output video timings and the corresponding VID. 59.94/23.97 Hz timings have the same VID as the corresponding

60/24 Hz timing in the table.

Table 20: PVSP Supported Output Video Timing and VID

Video Timing		VID
CEA	640x480p60	1
	720x480p60	2 or 3 or 14 or 15 or 35 or 36
	720(1440)x240p60	8 or 9
	720(2880)x240p60	12 or 13
	1280x720p60	4
	1920x1080p	16
	720x576p50	17 or 18 or 29 or 30 or 37 or 38
	1280x720p50	19
	720x288p50	23 or 24 or 27 or 28
	1920x1080p50	31
	1920x1080p24	32
	1920x1080p25	33
	1920x1080p30	34
	720p100	41
	576p100	42 or 43
	720p120	47
	480p120	48 or 49
	576p200	52 or 53
	480p240	56 or 57
	4kx2k 30 Hz	112
	4kx2k 25 Hz	113
	4kx2k 24 Hz	114
	4kx2k 24 Hz SMPTE	115
VESA timing	VGA	200
	SVGA	201
	XGA	202
	WXGA	203
	SXGA	204
	WXGA-2	205
	UXGA	206
	WXGA-3	207
	WUXGA	208

If overscan, crop or album mode is being used, the required blocks must be configured manually by enabling the corresponding enable bits, such as pvsp_vim_crop_enable, to enable the VIM crop block.

3.2.1.2. Customized Input/Output Video Format Configuration

If the input timing is not in the PVSP input format table, customized input format needs to be set manually.

If the input resolution has a variation with regard to standard timing (for example, if pvsp_autocfg_input_vid[7:0] is set to 2, which indicates the input resolution is 720x480, but the actual resolution is 718x478), the user can manually set pvsp_autocfg_input_vid[7:0] to 0 and set the input resolution through the following three registers.
pvsp_man_input_res, Primary VSP Map, *Address 0xE884[5]*
This bit is used to enable the manual configuration of the input resolution.

Function

pvsp_man_input_res	**Description**
0 (default)	Disable manual configuration of input resolution
1	Enable manual configuration of input resolution

pvsp_vin_h[10:0], Primary VSP Map, *Address 0xE82E[2:0]; Address 0xE82F[7:0]*
This signal is used to set the horizontal resolution of the input video. This register's value will be used while pvsp_man_input_res is 1 or pvsp_autocfg_input_vid is 0.

Function

pvsp_vin_h[10:0]	Description
0x000 (default)	Default
0xXXX	Horizontal resolution of input video

pvsp_vin_v[10:0], Primary VSP Map, *Address 0xE830[2:0]; Address 0xE831[7:0]*
This signal is used to set the vertical resolution of the input video. This register's value will be used while pvsp_man_input_res is 1 or pvsp_autocfg_input_vid is 0.

Function

pvsp_vin_v[10:0]	Description
0x000 (default)	Default
0xXXX	Vertical resolution of input video

Similarly, if the output timing is not in the PVSP output format table, customized output format needs to be set manually. The detailed configuration instructions are given in the PVSP VOM output port description.

3.2.1.3. *Field/Frame Buffer Number*

Depending on the type of conversion that is to take place, a number of buffers must be allocated for the input/output video data. Depending on the conversion required, this should be set in the pvsp_fieldbuf_num register. pvsp_fieldbuf_num can be automatically set by pvsp_autocfg_input_vid[7:0] and pvsp_autocfg_output_vid[7:0]. The pvsp_fieldbuf_num register does not change when crop or album mode is enabled.

pvsp_fieldbuf_num[2:0], Primary VSP Map, *Address 0xE829[2:0]*
Sets the number of field/frame buffers.

Function

pvsp_fieldbuf_num[2:0]	Description
000 «	Default
XXX	Number of field/frame buffers

3.2.1.4. *Field/Frame Buffer Address and Size*

In order to store video data in external memory in the correct size fields, the buffer size of the external DDR2 memory must be programmed by the user. Configuring this manually allows the user to have very flexible control over the external DDR memory.

These programmed field buffers or frame buffers are allocated by setting the following registers:
pvsp_fieldbuffer0_addr[31:0]
pvsp_fieldbuffer1_addr[31:0]
pvsp_fieldbuffer2_addr[31:0]
pvsp_fieldbuffer3_addr[31:0]
pvsp_fieldbuffer4_addr[31:0]
pvsp_fieldbuffer5_addr[31:0]
pvsp_fieldbuffer6_addr[31:0].

The value programmed into each of these registers is determined by Equation 19.

$$field\_size \equiv \frac{(active\_video\_width \times active\_video\_height)}{1+\mathrm{PVSP\_IS\_I\_TO\_P}} xbytes\_per\_pixel$$

Equation 19: Calculating External Memory Field Buffers

where:

- PVSP_IS_I_TO_P indicates whether the input timing is interlaced or progressive (interlaced = 1, progressive = 0)
- bytes_per_pixel indicates the number of bytes required to store each pixel (refer to Table 23 for more details on the number of bytes required per pixel)

For example, for an input video resolution of 720p, Equation 19 would yield the following field size:

Field_size = ((720)x(1280))x4 = 3686400

The following values would then need to be programmed to the above registers:

pvsp_fieldbuffer0_addr[31:0] = 0pvsp_fieldbuffer1_addr[31:0] = 38400 (3686400 in hex)

pvsp_fieldbuffer2_addr[31:**0**] = 70800 (7372800 in hex)

Note: The default value of the field/frame buffer is set for a 1080p input. If the maximum supported video is 1080p, there is no need to change the setting of the field/frame buffer. It is recommended to leave the setting of the buffer number and the buffer size unchanged.

pvsp_fieldbuffer0_addr[**31:0**], Primary VSP Map, *Address 0xE800[7:0]; Address 0xE801[7:0]; Address 0xE802[7:0]; Address 0xE803[7:0]*
This signal is used to set the start address of field/frame buffer 0. Software should arrange memory space properly, avoiding conflict between different buffers.

Function

pvsp_fieldbuffer0_addr[31:0]	Description
0x004F1A00	Default
0xXXXXXXXX	Start address of field/frame buffer 0

pvsp_fieldbuffer1_addr[**31:0**], Primary VSP Map, *Address 0xE804[7:0]; Address 0xE805[7:0]; Address 0xE806[7:0]; Address 0xE807[7:0]*
This signal is used to set the start address of field/frame buffer 1. Software should arrange memory space properly, avoiding conflict between different buffers.

Function

pvsp_fieldbuffer1_addr[31:0]	Description
0x00CDAA00	Default
0xXXXXXXXX	Start address of field/frame buffer 1

pvsp_fieldbuffer2_addr[**31:0**], Primary VSP Map, *Address 0xE808[7:0]; Address 0xE809[7:0]; Address 0xE80A[7:0]; Address 0xE80B[7:0]*
This signal is used to set the start address of field/frame buffer 2. Software should arrange memory space properly, avoiding conflict between different buffers.

Function

pvsp_fieldbuffer2_addr[31:0]	Description
0x014C3A00	Default
0xXXXXXXXX	Start address of field/frame buffer 2

pvsp_fieldbuffer3_addr[**31:0**], Primary VSP Map, *Address 0xE80C[7:0]; Address 0xE80D[7:0]; Address 0xE80E[7:0]; Address 0xE80F[7:0]*
This signal is used to set the start address of field/frame buffer 3. Software should arrange memory space properly, avoiding conflict between different buffers.

Function

pvsp_fieldbuffer3_addr[31:0]	Description
0x01CACA00	Default
0xXXXXXXXX	Start address of field/frame buffer 3

pvsp_fieldbuffer4_addr[31:0], Primary VSP Map, *Address 0xE810[7:0]; Address 0xE811[7:0]; Address 0xE812[7:0]; Address 0xE813[7:0]*
This signal is used to set the start address of field/frame buffer 4. Software should arrange memory space properly, avoiding conflict between different buffers.

Function

pvsp_fieldbuffer4_addr[31:0]	**Description**
0x02495A00	Default
0xXXXXXXXX	Start address of field/frame buffer 4

pvsp_fieldbuffer5_addr[31:0], Primary VSP Map, *Address 0xE814[7:0]; Address 0xE815[7:0]; Address 0xE816[7:0]; Address 0xE817[7:0]*
This signal is used to set the start address of field/frame buffer 5. Software should arrange memory space properly, avoiding conflict between different buffers.

Function

pvsp_fieldbuffer5_addr[31:0]	**Description**
0x02C7EA00	Default
0xXXXXXXXX	Start address of field/frame buffer 5

pvsp_fieldbuffer6_addr[31:0], Primary VSP Map, *Address 0xE889[7:0]; Address 0xE88A[7:0]; Address 0xE88B[7:0]; Address 0xE88C[7:0]*
This signal is used to set the start address of field/frame buffer 6. Software should arrange memory space properly, avoiding conflict between different buffers.

Function

pvsp_fieldbuffer6_addr[31:0]	**Description**
0x03073200	Default
0xXXXXXXXX	Start address of field/frame buffer 6

3.2.1.5. Frame Latency

Different resolutions have different frame latencies, depending on the timing combination to and from the PVSP. This is due to the increased processing required in converting and scaling video data. Table 21 lists the frame latencies in normal mode for various resolutions.

Table 21: Frame Latency in Normal Mode

Output / **Input**	**Frame Rate**	**50 Hz**	**59.94/60 Hz**	**23.97/24 Hz**	**25/30 Hz**
Frame rate	**Timing**	**576p/720p/1080p**	**480p/720p/1080p**	**720p/1080p /4kx2k**	**720p/1080p /4kx2k**
50 Hz	576i 1080i	1.1~2.3[1, 2]	1.1~2.3	1.1~2.4	1.1~2.4
	576p 720p 1080p	0.1~1.3	0.1~1.3	0.1~1.4	0.1~1.4
59.94/60 Hz	480i 1080i	1.1~2.3	1.1~2.3	0.1~3.4[3]	0.1~1.4
	480p 720p 1080p	0.1~1.3	0.1~1.3	0.1~3.4[4]	0.1~1.4
23.97/24/25/30 Hz	720/1080p	0.1~0.8	0.1~0.8	0.1~1.3	0.1~1.3

1. x.x means x.x times the input video field/frame
2. A~B means frame latency is not a fixed value, it varies between A and B
3. If cadence detection is disabled, this value should be 0.3~1.4 with setting pvsp_frc_change_phase_en to 0, otherwise is 0.3~3.4
4. If progressive cadence detection is disabled, this value should be 0.3~1.4 with setting pvsp_frc_change_phase_en to 0, otherwise, it is 0.3~3.4

When crop or album mode is enabled, frame latency will be different from what is listed in Table 21. In this case, the user can use the following methods to measure frame latency:

pvsp_rb_frame_latency[2:0] and
pvsp_rb_hsync_latency[11:**0**] are read only registers. Their values are real-time frame and HSync latency between input and output video.

Frame latency may vary within a range; the pvsp_rb_max_latency[14:0] readback register indicates the maximum frame latency, while pvsp_rb_min_latency[14:0] indicates the minimum frame latency. If pvsp_frc_latency_measure_en is set to 0, pvsp_rb_max_latency[14:0] and pvsp_rb_min_latency[14:0] are cleared.

If asserting pvsp_frc_latency_measure_en, the PVSP will monitor the values in pvsp_rb_frame_latency[2:0] and pvsp_rb_hsync_latency[11:**0**], then record their maximum and minimum values in pvsp_rb_max_latency[14:0] and pvsp_rb_min_latency[14:0]. Both of these signals are 15 bits wide – the highest 3 bits are the frame latency and the lowest 12 bits are the HSync latency. Note that it will take several seconds for PVSP to find the maximum and minimum frame/HSync latency.

In a normal case (not game mode), the PVSP's input video and output video latency are consistent.

pvsp_frc_latency_measure_en, Primary VSP Map, *Address 0xE8F0[6]*
This bit is used to enable frame latency measuring. The results are recorded in pvsp_rb_max_latency and pvsp_rb_min_latency.

Function

pvsp_frc_latency_measure_en	Description
0 (default)	Disable frame latency measuring
1	Enable frame latency measuring

pvsp_rb_frame_latency[2:0], Primary VSP Map, *Address 0xE870[6:4] (Read Only)*
This signal is used to indicate the real time vsync latency.

Function

pvsp_rb_frame_latency[2:0]	Description
0xXXX	number of frame latency

pvsp_rb_hsync_latency[11:0], Primary VSP Map, *Address 0xE875[7:0]; Address 0xE876[7:4] (Read Only)*
This signal is used to indicate the real time Hsync latency.

Function

pvsp_rb_hsync_latency[11:0]	Description
0xXXX	number of hsync latency

pvsp_rb_max_latency[14:0], Primary VSP Map, *Address 0xE8F2[7:0]; Address 0xE8F3[7:1] (Read Only)*
This signal is used to record the maximum frame latency.

Function

pvsp_rb_max_latency[14:0]	Description
0xXXX	Maximum of frame latency

pvsp_rb_min_latency[14:0], Primary VSP Map, *Address 0xE8F4[7:0]; Address 0xE8F5[7:1] (Read Only)*
This is signal is used to record the minimum frame latency.

Function

pvsp_rb_min_latency[14:0]	Description
0xXXX	Minimum of frame latency

3.2.1.6. Game Mode

Frame latency should be as small as possible for gaming applications. PVSP supports a game mode, which has nearly zero frame latency (latency less than 5 lines).

To enable the game mode of PVSP, pvsp_bypass_ddr_mode should be asserted.

pvsp_bypass_ddr_mode, Primary VSP Map, *Address 0xE84D[5]*
This bit is used to enable game mode for the Primary VSP.

Function

pvsp_bypass_ddr_mode	**Description**
0 (default)	Normal mode
1	Game mode

External memory is not used in game mode. Intra-field interpolation is used for interlaced input. Mosquito/block noise reduction and sharpness are supported in game mode, both for interlaced input and progressive input.

In game mode, the following functions are not supported:

- Frame rate change
- Motion adaptive de-interlacing (autodisabled)
- Cadence detection (autodisabled)
- Random noise reduction (autodisabled)
- CUE correction (autodisabled)
- Crop
- Album mode

The functions listed as autodisabled do not need to be manually disabled in game mode – ADV8005 will automatically disable them when game mode is enabled. Functions which are not listed as autodisabled must be manually disabled before game mode is enabled.

3.2.1.7. *Low Latency Mode*

Game mode has a very small frame latency but some processing functions cannot be supported in this mode. ADV8005 provides another mode, low latency mode, which can support frame rate change, scaling, crop and album mode.

To enable low latency mode, pvsp_frc_low_latency_mode should be set to 1.

Frame latency in low latency mode is listed in Table 22, which shows the maximum frame latency is 1.4 x frame.

Table 22: Frame Latency in Low Latency Mode

Output **Input**	**Frame Rate**	**50 Hz**	**59.94/60 Hz**	**23.97/24 Hz**	**25/30 Hz**
Frame rate	**Timing**	**576p/720p/1080p**	**480p/720p/1080p**	**720p/1080p /4kx2k**	**720p/1080p /4kx2k**
50 Hz	576i 1080i	0.3~1.3	0.3~1.3	0.3~1.4	0.3~1.4
	576p 720p 1080p	0.3~1.3	0.3~1.3	0.3~1.4	0.3~1.4
59.94/60 Hz	480i 1080i	0.3~1.3	0.3~1.3	0.3~1.4	0.3~1.4
	480p 720p 1080p	0.3~1.3	0.3~1.3	0.3~1.4	0.3~1.4
23.97/24/25/30 Hz	720/1080p	0.3~0.8	0.3~0.8	0.3~1.3	0.3~1.3

The following functions are not supported In low latency mode:

- Motion adaptive de-interlacing (autodisabled)
- Cadence detection (autodisabled)
- Random noise reduction (autodisabled)
- CUE correction (autodisabled)

pvsp_frc_low_latency_mode, Primary VSP Map, *Address 0xE84D[2]*
This bit is used to enable low latency mode.

Function

pvsp_frc_low_latency_mode	**Description**
0 (default)	Disable low latency mode
1	Enable low latency mode

3.2.1.8. *Freezing Output Video*

It is possible to freeze the output video from the PVSP by disabling the VIM. This can be achieved by setting pvsp_enable_vim to 0.

3.2.1.9. *Progressive Cadence Detection*

The ADV8005 PVSP supports multiple different types of cadence detection. Progressive cadence detection is another feature supported by ADV8005 when the input video is 60 Hz and the output video is 24 Hz. An example of progressive cadence detection would involve the ADV8005 detecting a pull-down ratio of 3:2 for 60 Hz video and reconverting this to its original film content at 24 Hz. This would allow the video to be output at 24 Hz and, therefore, be displayed at the highest image quality possible.

Conversions from slower to higher frame rates are achieved by repeating certain frames. Similarly, conversions from higher to lower frame rates are achieved by dropping some frames. Care has to be taken with repeating and dropping frames so that the quality of the video is not impacted. A simple example of frame rate conversion is outlined in Figure 54. This example involves converting the input video at a rate of 24 fps to 30 fps. These two frame rates have a ratio of 4:5; for every 4 frames of input video, there must be 5 frames of output video. This example uses a cadence detection of 3:2 pull-down which means that for every second frame of video data, an extra field of video information will be displayed.

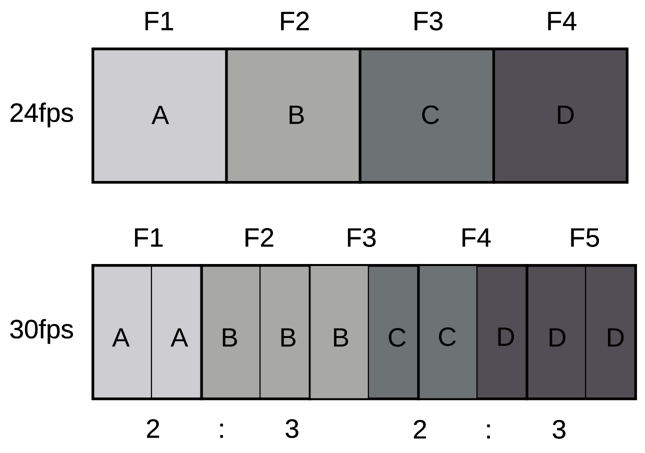


Figure 54: 2:3 Frame Rate Conversion

Progressive cadence detection can be enabled by setting register pcadence_enable to 1.

pcadence_enable, Primary VSP Map, *Address 0xE84D[1]*
This bit is used to enable progressive cadence detection.

Function

pcadence_enable	**Description**
0	Disable progressive cadence detection
1 (default)	Enable progressive cadence detection

3.2.2. *PVSP Video Input Module*

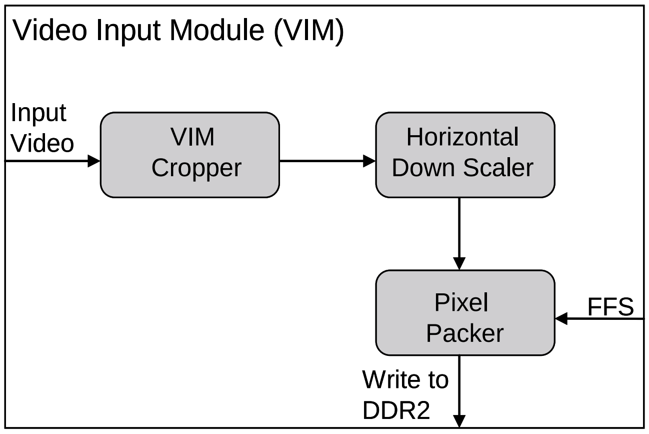


Figure 55: PVSP Video Input Module

3.2.2.1. *VIM Cropper*

The VIM cropper block is used to define a sub window within the given input resolution. This cropped image becomes the video which is processed by the PVSP. The following registers are used to define this sub window:

- pvsp_vim_crop_enable
- pvsp_vim_crop_h_start[10:0]
- pvsp_vim_crop_v_start[10:0]
- pvsp_vim_crop_width[10:0]
- pvsp_vim_crop_height[10:0]

To enable cropper block in VIM, pvsp_vim_crop_enable should be asserted.

pvsp_vim_crop_enable, Primary VSP Map, *Address 0xE883[6]*
This bit is used to enable the VIM crop.

Function

pvsp_vim_crop_enable	**Description**
0 (default)	Disable VIM Crop
1	Enable VIM Crop

Figure 56 shows the correlation between this cropped image and the input video.

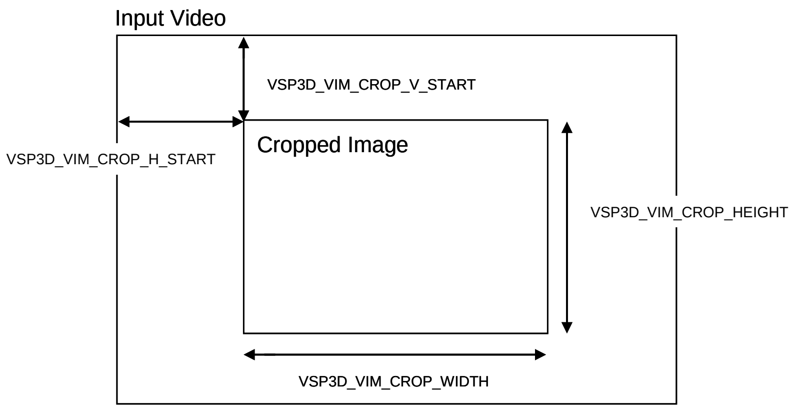


Figure 56: VIM Crop Dimensions

pvsp_vim_crop_h_start[10:0], Primary VSP Map, *Address 0xE832[2:0]; Address 0xE833[7:0]*
This signal is used to set the horizontal start position of the VIM cropper.

Function

pvsp_vim_crop_h_start[10:0]	Description
0x000 (default)	Default
0xXXX	Horizontal start position of VIM cropper input

pvsp_vim_crop_v_start[10:0], Primary VSP Map, *Address 0xE834[2:0]; Address 0xE835[7:0]*
This signal is used to set the vertical start position of the VIM cropper.

Function

pvsp_vim_crop_v_start[10:0]	Description
0x000 (default)	Default
0xXXX	Vertical start position of VIM cropper input

pvsp_vim_crop_width[10:0], Primary VSP Map, *Address 0xE836[2:0]; Address 0xE837[7:0]*
This signal is used to set the input width of the VIM cropper.

Function

pvsp_vim_crop_width[10:0]	Description
0x000 (default)	Default
0xXXX	Width of VIM cropper input

pvsp_vim_crop_height[10:0], Primary VSP Map, *Address 0xE838[2:0]; Address 0xE839[7:0]*
This signal is used to set the input height of the VIM cropper.

Function

pvsp_vim_crop_height[10:0]	Description
0x000 (default)	Default
0xXXX	Height of VIM cropper input

Note: The following limitations apply to the values that can be programmed in these registers:

- 0 <= pvsp_vim_crop_h_start[10:0] <= (INPUT VIDEO HORIZONTAL RESOLUTION – 1)
- 0 <= pvsp_vim_crop_v_start[10:0] <= (INPUT VIDEO VERTICAL RESOLUTION – 1)
- (pvsp_vim_crop_h_start[10:0] + pvsp_vim_crop_width[10:0]) <= INPUT VIDEO HORIZONTAL ACTIVE PIXELS
- (pvsp_vim_crop_v_start[10:0] + pvsp_vim_crop_height[10:0]) <= INPUT VIDEO VERTICAL ACTIVE PIXELS

3.2.2.2. *Horizontal Down Scaler*

Although the VOM has both horizontal and vertical scalers, there is also a horizontal down scaler in the VIM. The purpose of the VIM down scaler is to save external memory bandwidth by doing horizontal downscaling before writing video data into the external memory to save memory bandwidth.

The down scaler in the VIM should only be enabled when horizontal downscaling is needed, which means that the number of horizontal output active pixels should be less than the number of horizontal input active pixels. When album mode is enabled, the specified active output video width should be the album width.

If the horizontal resolution of the PVSP output timing is less than the input timing, the horizontal down scaler can be enabled to reduce the load on the external DDR2 memory. This horizontal down scaler input resolution is defined by the pvsp_vim_crop_width[10:0] register and the output resolution is defined by the pvsp_vim_d_scal_out_width[10:0] register. To enable the horizontal down scaler, pvsp_vim_d_scal_enable should be set to 1.

pvsp_vim_d_scal_enable, Primary VSP Map, *Address 0xE883[5]*
This bit is used to enable the VIM down scaler.

Function

pvsp_vim_d_scal_enable	**Description**
0 (default)	Disable VIM down scaler
1	Enable VIM down scaler

pvsp_vim_d_scal_out_width[10:0], Primary VSP Map, *Address 0xE83A[2:0]; Address 0xE83B[7:0]*
This signal is used to set the output video width of the down-scaling scaler in the VIM. The input video width is set by register pvsp_vim_crop_width. If VIM crop is not enabled, pvsp_vim_crop_width is auto configured by pvsp_autocfg_input_vid, which is the same with input video's horizontal resolution.

Function

pvsp_vim_d_scal_out_width[10:0]	**Description**
0x000 (default)	Default
0xXXX	Output width of VIM scalar

3.2.2.3. *Scaler Interpolation Mode*

This section describes the method for scaling the input video data. The purpose of the scaler is to allow different input formats to be displayed on a screen with a fixed resolution. The VIM scaler is usually used for downscaling, for example, 1080p to be downscaled to a lower definition format such as 480p. Different scaling interpolation modes will affect scaler performance. The options for video scaling modes are described below and are chosen using pvsp_vim_scal_type[1:0].

Proprietary ADI Algorithm

This is a custom algorithm developed by ADI which allows improved performance in the scaling of the input video. This can reduce many common artifacts when scaling video data such as:

- **Saw tooth** – otherwise known as 'jaggies', this is an artifact that occurs when an image is zoomed in and is one of the most important criteria when evaluating scaling performance.
- **Edge blurring** – when zooming in, most high frequency information is lost, resulting in edges becoming blurred. The proprietary ADI algorithm keeps the edge region sharp by retaining the high frequency information.
- **Ringing** – also known as the Gibbs phenomenon, can be found on video due to a reduction in high frequency information. The proprietary ADI algorithm helps with the reduction of such artifacts.

Sharp/Smooth

Both the sharp/smooth options for scaler interpolation are versions of the proprietary ADI algorithm. The sharp and smooth versions allow for limited customization of the scaler function. This function can be set depending on the user preference.

Bilinear

The bilinear option uses an averaging method within a 2x2 pixel array to increase the size of the input frame. This is a cruder method of scaling than the default proprietary ADI Algorithm. In most cases, the scaler should be left at the default setting.

pvsp_vim_scal_type[1:0], Primary VSP Map, *Address 0xE8E5[7:6]*
This signal is used to set the VIM scaling algorithm. For up-scaling, the proprietary ADI algorithm is recommended; whereas for down-scaling, the sharp setting is recommended.

Function

pvsp_vim_scal_type[1:0]	Description
00	Proprietary ADI Algorithm
01	Sharp
10	Smooth
11 (default)	Bilinear

3.2.2.4. *Scaler Controls*

The following register is used in the control of the VIM scaling function and should be tailored according to user requirements.

pvsp_vim_scal_overshoot_ctrl[11:0], Primary VSP Map, *Address 0xE8E9[7:0]; Address 0xE8EA[7:4]*
This bit is used to control the overshoot in the scaling of input video. If set to a value larger than the default setting, more overshoot is allowed.

Function

pvsp_vim_scal_overshoot_ctrl[11:0]	Description
0x080 (default)	Default

3.2.2.5. *Pixel Packer*

At the back end of the VIM, the pixel packer converts input video to word packets suitable for writing to external memory. Refer to Figure 55 for more details on where the pixel packer is located in the hardware. Depending on the format of the input video, there are four different packing formats:

- 12-bit 4:4:4 YCbCr
- 10-bit 4:4:4 YCbCr
- 12-bit 4:2:2 YCbCr
- 8-bit 4:2:2 YCbCr

There is a trade off in the number of bits that can be stored. A higher number of bits means the video stored will be stored at a higher quality, however, this will reduce the available DDR2 memory bandwidth for other functions such as OSD read/write.

The data format can be set by the pvsp_ex_mem_data_format[1:0] register. This register can be set at any time, but it may take some time (not more than 300 ms) to become valid. This delay is related to the ADV8005 taking control of the memory format change to avoid the display of garbage information. This information is important when calculating the field/frame buffer sizes, as explained in Section 3.2.1.

pvsp_ex_mem_data_format[1:0], Primary VSP Map, *Address 0xE829[4:3]*
This signal is used to set the data format in external memory.

Function

pvsp_ex_mem_data_format[1:0]	Description
00 (default)	YCbCr-12b-10b-10b
01	YCbCr-8b-8b-8b
10	YCbCr-4:2:4-12b
11	YCbCr-4:2:2-8b

Table 23 indicates the number of bytes required when storing a particular type of video data.

Table 23: Bytes per Pixel

pvsp_ex_mem_data_format	Format in Memory	Bytes per Pixel
0	12 bit 4:4:4 YCbCr	4
1	8 bit 4:4:4 YCbCr	3
2	12 bit 4:2:2 YCbCr	4
3	8 bit 4:2:2 YCbCr	2

3.2.3. *PVSP Video Output Module*

Figure 57 shows the structure of the PVSP VOM. The direction arrow inside Figure 57 does not capture the real processing order inside the VOM but gives a clear overview of each processing block.

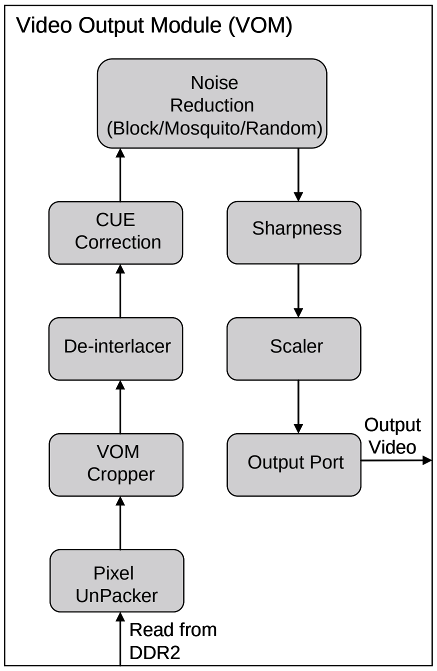


Figure 57: PVSP Video Output Module

The VOM has the following main features:

- Pixel unpacker: this module reads the field/frame from external memory and unpacks memory words to video pixel information
- VOM cropper: crops the image read from external memory
- De-interlacer: converts interlaced video to progressive video
- CUE correction: filtering for Color Upsampling Error
- Noise reduction: removes random, mosquito, and block noise
- Detail and edge sharpness enhancement
- Scaler: scales video to target resolution
- Output port: generates output timing and output video

Register update protection is provided in the ADV8005. Refer to Section 3.4 for more details regarding how to update the various VSP registers.

pvsp_lock_vom, Primary VSP Map, *Address 0xE828[3]*
This bit is used to lock the Video Output Module (VOM). If the Primary VSP is running and this bit is set to 1, the VOM will be locked to a current register setting to display the last frame. The Primary VSP registers can be configured safely in this state. All new register settings will be updated after this bit is set back to 0.

Function

pvsp_lock_vom	**Description**
0 (default)	Unlock VOM
1	Lock VOM

Note: This register should be used only as part of the gentle reboot protocol. Refer to Section 3.4.3 for more details.

pvsp_update_vom, Primary VSP Map, *Address 0xE828[4]*
This bit is used to control the updating of the VOM. Registers in the VOM can be updated only when pvsp_update_vom is asserted. To modify registers in the VOM, pvsp_update_vom should be de-asserted. The registers can then be modified. pvsp_update_vom should then be asserted to let the VOM use the updated register value in the next frame. This procedure will guarantee the correctness of the VOM configuration.

Function

pvsp_update_vom	**Description**
0	Do not update VOM
1 (default)	Update VOM

Note: Refer to Section 3.4 for more details on configuring the PVSP registers.

3.2.3.1. *Pixel Unpacker*

The pixel unpacker in the VOM is very similar to that in the VIM. It is used to convert external memory words into video pixel (YCbCr) data. Pixels in external memory have the following four different data formats (the same as those set by the VIM). The VOM pixel unpacker is configured in the same way as the VIM pixel unpacker.

- 12-bit 4:4:4 YCbCr
- 10-bit 4:4:4 YCbCr
- 12-bit 4:2:2 YCbCr
- 8-bit 4:2:2 YCbCr

Data format details are described in pvsp_ex_mem_data_format[1:0].

3.2.3.2. *VOM Cropper*

The VOM cropper is similar to the VIM cropper with the exception that it uses the VOM set protocol while the VIM cropper uses the gentle reboot protocol (refer to Section 3.4). Using the VIM cropper can reduce the external memory bandwidth required for scaling in cases where bandwidth is a concern. If not, the VOM cropper should be used. The following registers are used to configure the VOM cropper:

- pvsp_di_crop_enable
- pvsp_di_crop_h_start[10:0]
- pvsp_di_crop_v_start[10:0]
- pvsp_di_crop_width[10:0]
- pvsp_di_crop_height[10:0]

To enable cropper in VOM, pvsp_di_crop_enable should be asserted.

pvsp_di_crop_enable, Primary VSP Map, *Address 0xE883[4]*
This bit is used to enable the VOM crop.

Function

pvsp_di_crop_enable	**Description**
0 (default)	Disable VOM Crop
1	Enable VOM Crop

Figure 58 shows the function of the VOM cropper.

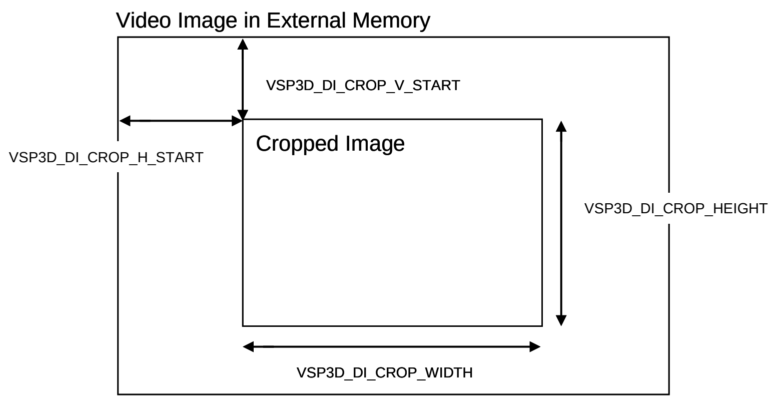


Figure 58: VOM Crop Dimensions

pvsp_di_crop_h_start[10:0], Primary VSP Map, *Address 0xE83C[2:0]; Address 0xE83D[7:0]*
This signal is used to set the horizontal start position of the VOM cropper.

Function

pvsp_di_crop_h_start[10:0]	Description
0x000 (default)	Default
0xXXX	Horizontal start position of VOM cropper input

pvsp_di_crop_v_start[10:0], Primary VSP Map, *Address 0xE83E[2:0]; Address 0xE83F[7:0]*
This signal is used to set the vertical start position of the VOM cropper.

Function

pvsp_di_crop_v_start[10:0]	Description
0x000 (default)	Default
0xXXX	Vertical start position of VOM cropper input

pvsp_di_crop_width[10:0], Primary VSP Map, *Address 0xE840[2:0]; Address 0xE841[7:0]*
This signal is used to set the width of the VOM cropper.

Function

pvsp_di_crop_width[10:0]	Description
0x000 (default)	Default
0xXXX	Width of VOM cropper input

pvsp_di_crop_height[10:0], Primary VSP Map, *Address 0xE842[2:0]; Address 0xE843[7:0]*
This signal is used to set the height of the VOM cropper.

Function

pvsp_di_crop_height[10:0]	Description
0x000 (default)	Default
0xXXX	Height of VOM cropper input

Note: The following restrictions apply to the values to which these registers can be set:
0 <= pvsp_di_crop_h_start[10:0] <= (HORIZONTAL RESOLUTION OUTPUT BY VIM – 1)
0 <= pvsp_di_crop_v_start[10:0] <= (VERTICAL RESOLUTION OUTPUT BY VIM – 1)
(pvsp_di_crop_h_start[10:0] + pvsp_di_crop_width[10:0]) <= HORIZONTAL RESOLUTION OUTPUT BY VIM
(pvsp_di_crop_v_start[10:0] + pvsp_di_crop_height[10:0]) <= VERTICAL RESOLUTION OUTPUT BY VIM

3.2.3.3. Motion Detection

The ADV8005 de-interlacer is used to convert interlaced video to progressive video. The PVSP has an extremely high quality de-interlacer algorithm which achieves excellent quality interlaced to progressive conversion. The algorithm uses motion adaptive de-interlacing technology, which includes motion detection, cadence detection, low angle detection and interpolation.

Motion detection extracts the motion information of each pixel. Based on this information, the ADV8005 chooses the most suitable form of de-interlacing. For static pixels (that is, pixels where no motion is deemed to have occurred), inter field interpolation is performed. For pixels where motion is detected, intra-field interpolation is performed. Motion detection technology is the essence of de-interlacing, so if a static pixel is detected as motion by mistake, vertical detail is lost. In contrast, if motion is detected as static by mistake, combing artifact occurs.

In order to support motion detection for interlaced inputs, two buffers in external memory are needed to store motion information. Their addresses are defined in the pvsp_motionbuf0_addr[31:0] and pvsp_motionbuf1_addr[31:0] registers. The size of each buffer should be equal to the MOTION_BUF_SIZE, which can be calculated from Equation 20.

$$MOTION\_BUF\_SIZE(byte) \equiv \frac{(active\_input\_video\_width \times active\_input\_video\_height)}{4}$$

Equation 20: Calculating Interlaced Buffers

pvsp_motionbuf0_addr[31:0], Primary VSP Map, *Address 0xE818[7:0]; Address 0xE819[7:0]; Address 0xE81A[7:0]; Address 0xE81B[7:0]*
This signal is used to set the start address of motion information buffer 0. Motion buffers are needed only when motion adaptive deinterlacing is enabled for interlaced input.

Function

pvsp_motionbuf0_addr[31:0]	Description
0x00000000 (default)	Default
0xXXXXXXXX	Start address of motion buffer 0

pvsp_motionbuf1_addr[31:0], Primary VSP Map, *Address 0xE81C[7:0]; Address 0xE81D[7:0]; Address 0xE81E[7:0]; Address 0xE81F[7:0]*
This signal is used to set the start address of motion information buffer 1. Motion buffers are needed only when motion adaptive deinterlacing is enabled for interlaced input.

Function

pvsp_motionbuf1_addr[31:0]	Description
0x0007E900 (default)	Default
0xXXXXXXXX	Start address of motion buffer 1

3.2.3.4. *Low Angle De-interlacing*

The ultra low angle de-interlacing interpolation algorithm (ULAI) developed by ADI performs intra field interpolation for the de-interlacing function. It is capable of determining the correct direction by examining several different directions and interpolating missing pixels based on this information. This results in higher quality low angle interpolation and reduces the effect of jaggies.

The ultra low angle interpolation function is only used for converting from interlaced to progressive formats. It can be enabled or disabled by asserting or de-asserting register di_ulai_enable.

di_ulai_enable, Primary VSP Map, *Address 0xE84C[3]*
This bit is used to enable the ultra low angle de-interlacing algorithm (ULAI).

Function

di_ulai_enable	Description
0	Disable ULAI
1 (default)	Enable ULAI

3.2.3.5. *Cadence Detection*

The ADV8005 cadence detection can handle multiple different types of cadences, typically introduced when content originated as film format but was converted into interlaced format for broadcast. Examples of such conversion can be seen in Figure 54. The PVSP is able to detect arbitrary cadences and even unknown cadence modes, with per pixel correction for combing artifacts.

There are several features of cadence detection, including the reliable detection of 2:2 cadences for PAL video and the detection of poor editing techniques often found in films converted to video standards (this may introduce artifacts). These artifacts are caused by multiple cadences in the same source as well as fast switching from film to video or between different cadences.

For an interlaced video input, cadence detection can be enabled or disabled by asserting or de-asserting di_cadence_enable. For progressive video input, cadence detection can be enabled or disabled by asserting or de-asserting pcadence_enable.

di_cadence_enable, Primary VSP Map, *Address 0xE84C[2]*
This bit is used to enable cadence detection.

Function

di_cadence_enable	Description
0	Disable cadence detection
1 (default)	Enable cadence detection

The PVSP supports the following cadence types:

- 2:2
- 2:2:2:4
- 3:2
- 2:3:3:2:2
- 2:3:3:2
- 3:2:3:2:2
- 3:3
- 4:4
- 5:5
- 6:4
- 8:7

Each of these cadence types can be disabled by setting the corresponding bit in di_fd_disabled_cadence[10:0] to 1.

For conversion of 60 Hz interlaced and progressive input timing to 24 Hz progressive timing, pvsp_frc_change_phase_en should be asserted. For all other cases, pvsp_frc_change_phase_en should be disabled when using 1 external DDR2 memory.

pvsp_frc_change_phase_en, Primary VSP Map, *Address 0xE84E[4]*
This bit is used to lock the phase change for cadence detection.

Function

pvsp_frc_change_phase_en	Description
0	Disable
1 (default)	Enable

di_fd_disabled_cadence[10:0], Primary VSP Map, *Address 0xE8FA[7:0]; Address 0xE8FB[7:5]*
This signal is used to disable corresponding cadence detection.

Function

di_fd_disabled_cadence[10:0]	Description
0x000 (default)	Default

Table 24: Corresponding Bit for Each Cadence Type

Bit	Disabled Cadence
0xE8FB[5]	2:2
0xE8FB[6]	2:2:2:4
0xE8FB[7]	3:2
0xE8FA[0]	2:3:3:2:2
0xE8FA[1]	2:3:3:2
0xE8FA[2]	3:2:3:2:2
0xE8FA[3]	3:3
0xE8FA[4]	4:4
0xE8FA[5]	5:5
0xE8FA[6]	6:4
0xE8FA[7]	8:7

3.2.3.6. CUE Correction

Color Upsampling Error (CUE) correction is implemented using a filter which removes the jagged edges caused by the artifacts introduced by the incorrect upsampling of MPEG 2 video data in 4:2:0 format to the 4:2:2/4:4:4 formats supported by DVD players.

The CUE correction function can be enabled or disabled by di_cue_enable.

di_cue_enable, Primary VSP Map, *Address 0xE84D[0]*
This bit is used to enable CUE correction.

Function

di_cue_enable	Description
0	Disable CUE correction
1 (default)	Enable CUE correction

3.2.3.7. Random Noise Reduction

There are several noise reduction algorithms in the ADV8005 that help with the reduction of common sources of video noise. The random noise reduction (RNR) block reduces the random noise which may be introduced in analog broadcasting or capturing. It employs a temporal recursive algorithm to stabilize the static regions while just processing the luma channel. Users can configure the register parameters to adjust the algorithm according to their preference. The amount level of RNR can be configured using di_rnr_level[1:0].

RNR supports both interlaced and progressive input. It can be enabled or disabled using di_rnr_enable.

di_rnr_enable, Primary VSP Map, *Address 0xE84C[4]*
This bit is used to enable random noise reduction (RNR).

Function

di_rnr_enable	Description
0 (default)	Disable RNR
1	Enable RNR

di_rnr_level[1:0], Primary VSP Map, *Address 0xE84F[1:0]*
This signal sets the RNR level.

Function

di_rnr_level[1:0]	Description
00	N/A
01	Low
10 (default)	Middle
11	High

For the RNR feature to operate, two buffers in external memory must be allocated to store video information which will be used for noise reduction purposes. The addresses of these two buffers can be set in the pvsp_rnrbuf0_addr[31:0] and pvsp_rnrbuf1_addr[31:0] registers. The size of each buffer should be larger than RNR_BUF_SIZE, which can be calculated as shown in Equation 21.

$$RNR\_BUF\_SIZE(byte) \equiv active\_input\_video\_width \times active\_input\_video\_height$$

Equation 21: Calculating RNR Buffers

Note: Using RNR will use external memory bandwidth which may impact on other features such as OSD image storage as well as de-interlacing.

pvsp_rnrbuf0_addr[31:0], Primary VSP Map, *Address 0xE820[7:0]; Address 0xE821[7:0]; Address 0xE822[7:0]; Address 0xE823[7:0]*
Sets the start address of random noise reduction information buffer 0. RNR buffers are needed only when random noise reduction is enabled.

Function

pvsp_rnrbuf0_addr[31:0]	Description
0x000FD200 «	Default
0xXXXXXXXX	Start address of RNR buffer 0

pvsp_rnrbuf1_addr[31:0], Primary VSP Map, *Address 0xE824[7:0]; Address 0xE825[7:0]; Address 0xE826[7:0]; Address 0xE827[7:0]*
Sets the start address of random noise reduction information buffer 1. RNR buffers are needed only when random noise reduction is enabled.

Function

pvsp_rnrbuf1_addr[31:0]	**Description**
0x002F7600 «	Default
0xXXXXXXXX	Start address of RNR buffer 1

3.2.3.8. *Mosquito Noise Reduction*

The second type of noise reduction algorithm implemented in the ADV8005 is the mosquito noise reduction (MNR). The MNR block selectively removes ringing artifacts introduced into highly compressed (MPEG) video data. For the best results, this block should be enabled when the input video is not being scaled, due to the fact that it is easier to identify and remove compressed artifacts at lower resolutions.

MNR can support both interlaced and progressive input video. It can be enabled or disabled by di_mnr_enable. As with the RNR block, a certain amount of control is provided to the user. This can be controlled using di_mnr_level[1:0].

di_mnr_enable, Primary VSP Map, *Address 0xE84C[5]*
This bit is used to enable mosquito noise reduction (MNR).

Function

di_mnr_enable	**Description**
0 (default)	Disable MNR
1	Enable MNR

di_mnr_level[1:0], Primary VSP Map, *Address 0xE84F[3:2]*
This signal sets the MNR level.

Function

di_mnr_level[1:0]	**Description**
00	N/A
01	Low
10 (default)	Middle
11	High

To get better image performance, register di_mnr_th_min[3:0] can be used to set the MNR level.

di_mnr_th_min[3:0], Primary VSP 2 Map, *Address 0xE917[7:4]*
This signal is used to set the strength of the mosquito noise reduction (MNR). The larger the value, the stronger the MNR noise reduction.

Function

di_mnr_th_min[3:0]	**Description**
0010 (default)	Normal strength MNR
0110	High strength MNR

3.2.3.9. *Block Noise Reduction*

The block noise reduction (BNR) algorithm removes 'blocky' artifacts introduced into highly compressed video such as MPEG2 encoded video. For the best results, this function should be enabled when the input video is not scaled. The BNR has excellent performance for high level block artifact patterns, and it has smart block position detection.

BNR supports both interlaced and progressive input. It can be enabled or disabled using di_bnr_enable. The BNR level can be controlled by setting di_bnr_detect_scale_line[3:0], di_bnr_disable_local_detect, di_bnr_edge_offset[7:0], di_bnr_global_strength_gain[3:0], di_bnr_scale_global_hori[2:0] and di_bnr_scale_global_vert[2:0]. The corresponding value for different reduction level is given in Table 25.

Table 25: Corresponding Value for Block Noise Reduction Level

Register Name	High	Middle	Low
di_bnr_detect_scale_line[3:0]	9	7	7
di_bnr_disable_local_detect	0	1	1
di_bnr_edge_offset[7:0]	96	64	32
di_bnr_global_strength_gain[3:0]	12	8	7
di_bnr_scale_global_hori[2:0]	6	5	5
di_bnr_scale_global_vert[2:0]	6	5	5

di_bnr_enable, Primary VSP Map, *Address 0xE84C[6]*
This bit is used to enable block noise reduction (BNR).

Function

di_bnr_enable	Description
0 (default)	Disable BNR
1	Enable BNR

di_bnr_edge_offset[7:0], Primary VSP 2 Map, *Address 0xE98D[7:0]*
This signal is used to configure the BNR processing ability.

Function

di_bnr_edge_offset[7:0]	Description
0x32	Recommended setting for low level BNR
0x64	Recommended value for mid level BNR
0x96	Recommended value for high level BNR

di_bnr_disable_local_detect, Primary VSP 2 Map, *Address 0xE987[3]*
This signal is used to configure the BNR processing ability.

Function

di_bnr_disable_local_detect	Description
0	Recommended setting for high level BNR
1 (default)	Recommended setting for low/mid level BNR

di_bnr_scale_global_vert[2:0], Primary VSP 2 Map, *Address 0xE98B[7:5]*
This signal is used to configure the BNR processing ability.

Function

di_bnr_scale_global_vert[2:0]	Description
0101 (default)	Recommended setting for low/mid level BNR
0110	Recommended setting for high level BNR

di_bnr_scale_global_hori[2:0], Primary VSP 2 Map, *Address 0xE98B[4:2]*
This signal is used to configure the BNR processing ability.

Function

di_bnr_scale_global_hori[2:0]	Description
0101 (default)	Recommended setting for low/mid level BNR
0110	Recommended setting for high level BNR

di_bnr_global_strength_gain[3:0], Primary VSP 2 Map, *Address 0xE988[7:4]*
This signal is used to configure the BNR processing ability.

Function

di_bnr_global_strength_gain [3:0]	Description
1000 (default)	Recommended setting for low/mid level BNR
1100	Recommended setting for high level BNR

di_bnr_detect_scale_line[3:0], Primary VSP 2 Map, *Address 0xE987[7:4]*
This signal is used to configure the BNR processing ability.

Function

di_bnr_detect_scale_line[3:0]	Description
0111 (default)	Recommended setting for low/mid level BNR
1001	Recommended setting for high level BNR

3.2.3.10. Sharpness Enhancement

The sharpness enhancement block extracts high frequency data from the de-interlaced and de-noised video frame to help simultaneously sharpen the appearance of edges and other video details, recover high frequency components and provide pictures with a natural look without adding a halo or ringing artifact. Since the sharpness works on a two dimensional pixel array before the scaler, noise will not be scaled during the scaling operation.

Detail and edge sharpness enhancement supports both interlaced and progressive inputs. It can be enabled or disabled using di_sharpness_enable. The sharpness level can be adjusted using the signed, twos complement value in pvsp_srscal_scale_gain[11:0]. To increase the sharpness setting, the value in pvsp_srscal_scale_gain[11:0] should be increased. To decrease the sharpness setting, the value in pvsp_srscal_scale_gain[11:0] should be decreased.

di_sharpness_enable, Primary VSP Map, *Address 0xE84C[7]*
This bit is used to enable sharpness control.

Function

di_sharpness_enable	Description
0 (default)	Disable sharpness
1	Enable sharpness

pvsp_srscal_scale_gain[11:0], Primary VSP Map, *Address 0xE891[7:0]; Address 0xE892[7:4]*
This signal is used to control the sharpness level.

Function

pvsp_srscal_scale_gain[11:0]	Description
0x000 (default)	Sharpness level

3.2.3.11. Scaler

The last block before the VOM output is the scaler which is used to scale the input video to the desired resolution. This is very flexible and can support arbitrary resolution conversion and independently scale the input video horizontally and vertically. The ADI proprietary scaler algorithms also allow improved performance in the scaling of the input video which improves many common issues associated with scaling video data such as saw tooth, edge blurring, and ringing.

The ADV8005 scaler employs contour-based interpolation techniques to provide sharp edges and crisp details on high resolution content. The embedded compression noise reduction will eliminate mosquito noise and block artifacts. The contour-based interpolation scaler is capable of upscaling input video formats from 480i to 4k x 2k formats (these include 4k x 2k 30 Hz/4k x 2k 25 Hz/4k x 2k 24 Hz and 4k x 2k 24 Hz SMPTE).

When the automatic scaler algorithm selection is enabled, the contour-based interpolation scaler is used for upscaling and downscaling is performed using the frequency-adaptive scaler which implements the same algorithm as the VIM down scaler. A manual selection between the contour-based interpolation scaler and the frequency-adaptive scaler is provided by pvsp_srscal_interp_mode[1:0].

Also, because ADV8005 provides 4kx2k timing in 8-bit precision, pvsp_srscal_8bit_en is provided to make the entire scaler operate in 8-bit mode. This lowers the power consumed by the scaler.

pvsp_srscal_interp_mode[1:0], Primary VSP Map, *Address 0xE894[7:6]*
This signal is used to select the scaler algorithm employed.

Function

pvsp_srscal_interp_mode[1:0]	**Description**
00 (default)	Automatic scaler algorithm selection
01	Contour-based interpolation scaler (2nd gen scaling algorithm with 4k x 2k support)
10	Frequency-adaptive scaler (1st gen scaling algorithm)
11	Bilinear scaler

pvsp_srscal_8bit_en, Primary VSP Map, *Address 0xE890[3]*
This bit is used to set the scaler into 8-bit mode. This bit should be set when output 4K x 2K timing.

Function

pvsp_srscal_8bit_en	**Description**
0 (default)	Scaler not in 8 bit mode
1	Scaler in 8 bit mode

The size of the active image sent to the scaler is set by pvsp_di_crop_height[10:0] and pvsp_di_crop_width[10:0]**.** The scaler output can then be set using pvsp_scal_out_height[12:0] and pvsp_scal_out_width[12:0] by setting pvsp_man_scal_out_enable to 1, or it can set automatically using pvsp_autocfg_input_vid[7:0]. These registers should be set to the resolution of the output video. Refer to Figure 59 for more details.

pvsp_man_scal_out_enable, Primary VSP Map, *Address 0xE883[3]*
This bit is used to enable the manual setting of pvsp_scal_out_width and pvsp_scal_out_height.

Function

pvsp_man_scal_out_enable	**Description**
0 (default)	Disable manually setting M_Scaler output resolution
1	Enable manually setting M_Scaler output resolution

pvsp_scal_out_height[12:0], Primary VSP Map, *Address 0xE846[4:0]; Address 0xE847[7:0]*
This signal is used to set the output vertical resolution of scaler in the VOM.

Function

pvsp_scal_out_height[12:0]	**Description**
0x000 (default)	Default
0xXXX	Output height of VOM scaler

pvsp_scal_out_width[12:0], Primary VSP Map, *Address 0xE844[4:0]; Address 0xE845[7:0]*
This signal is used to set the output horizontal resolution of scaler in the VOM.

Function

pvsp_scal_out_width[12:0]	**Description**
0x000 (default)	Default
0xXXX	Output width of VOM scaler

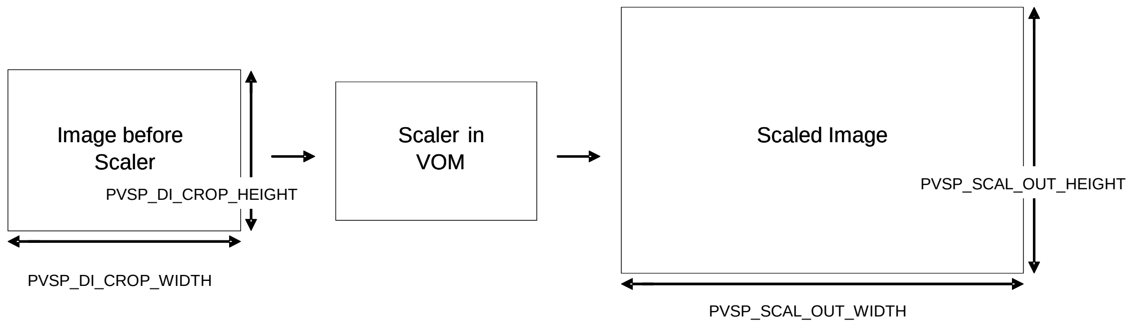


Figure 59: VOM Scaler Dimensions

3.2.3.12. Panorama Mode

If the scaled video has a different aspect ratio to the original and the horizontal scaling factor is larger than the vertical, the panorama function can be enabled using m_scaler_panorama_en. In effect, this stretches the left- and right-most sides of the input video to fill the output resolution. This method keeps the original ratio in the centre of the screen. Figure 60 explains the panorama mode scaling feature.

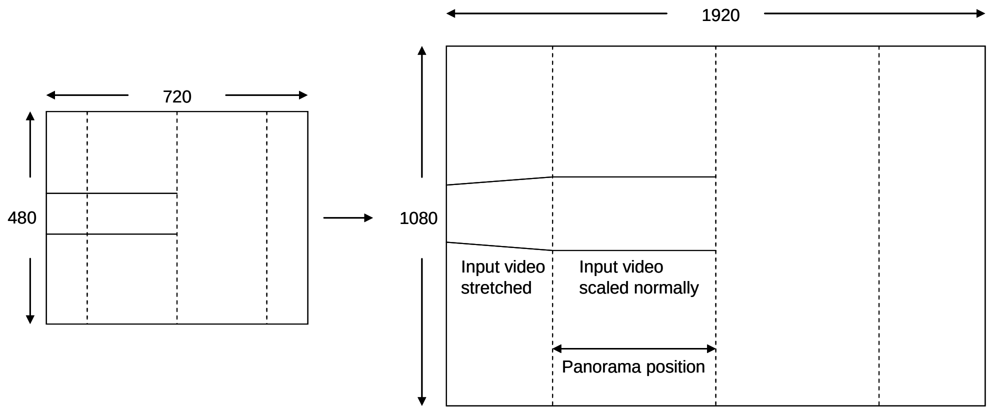


Figure 60: Panorama Scaling Feature

m_scaler_panorama_en, Primary VSP Map, *Address 0xE850[0]*
This bit enables panorama scaling for the VOM scaler.

Function

m_scaler_panorama_en	**Description**
0 (default)	Disable VOM panorama
1	Enable VOM panorama

The position from which the output video becomes stretched is controlled using m_scaler_panorama_pos[11:0]. This allows the user to control the width of the sides of the output image. Refer to Figure 60 for more details.

m_scaler_panorama_pos[11:0], Primary VSP Map, *Address 0xE851[3:0]; Address 0xE852[7:0]*
This signal is used to define the width of the output video frame which is not stretched when panorama mode is enabled but, rather, is scaled properly. The maximum value of this register is set by: pvsp_di_crop_width * (pvsp_scal_out_width/pvsp_di_crop_height) - pvsp_scal_out_width/2.

This register sets half the width of the output frame which is to be scaled normally. By default, this register is set to 0 which means that all the input frame will be stretched. It is, therefore, recommended that this register is set by the user before enabling the panorama function.

Function

m_scaler_panorama_pos[11:0]	**Description**
0x000 (default)	Default
0xXXX	Width of not-stretched image

3.2.3.13. Output Port

This section details the configuration registers for the final block of the PVSP VOM. The primary purpose of the output port is to generate the output video timing and output the video data. Refer to Table 26 for the register settings for the common CEA video formats that are supported by the ADV8005. The output setting can be automatically configured using the setting of pvsp_autocfg_output_vid[7:0]. If the output configuration needs to be set manually, pvsp_man_dp_timing_enable must be set to 1 and pvsp_autocfg_output_vid[7:0] must be set to 0. Refer to Figure 61 for more information.

When using manual configuration of the output timing format, pvsp_dp_4kx2k_mode_en needs to be manually enabled when outputting 4k x 2k series timings and should be disabled for other timing formats.

If a limited range of output must be provided, pvsp_data_clipping_en should be enabled. Otherwise, this register should be disabled. A limited range indicates the output is clipped to 16-235 range for each data channel of pixel.

pvsp_dp_4kx2k_mode_en, Primary VSP Map, *Address 0xE869[4]*

This bit is used to make the VOM display module work in 4K x 2K mode. This register's value will be used while pvsp_autocfg_output_vid is 0.

Function

pvsp_dp_4kx2k_mode_en	Description
0 (default)	Not in 4K x 2K mode
1	In 4K x 2K mode

pvsp_data_clipping_en, Primary VSP Map, *Address 0xE84E[3]*

This bit is used to limit the output data within range of 16~235.

Function

pvsp_data_clipping_en	Description
0 (default)	Not limit output data.
1	Limit output data

pvsp_man_dp_timing_enable, Primary VSP Map, *Address 0xE883[0]*

This bit is used to enable the manual setting of the display port's timing.

Function

pvsp_man_dp_timing_enabl e	Description
0 (default)	Disable manually setting output timing
1	Enable manually setting output timing

pvsp_dp_decount[12:0], Primary VSP Map, *Address 0xE856[4:0]; Address 0xE857[7:0]*

This signal is used to set the DE duration of output timing. This register's value will be used while pvsp_autocfg_output_vid is 0.

Function

pvsp_dp_decount[12:0]	Description
0x000 (default)	Default
0xXXX	Data enable count of output timing

pvsp_dp_hfrontporch[11:0], Primary VSP Map, *Address 0xE858[3:0]; Address 0xE859[7:0]*

This signal is used to set the horizontal front porch duration of output timing. This register's value will be used while pvsp_autocfg_output_vid is 0.

Function

pvsp_dp_hfrontporch[11:0]	Description
0x000 (default)	Default
0xXXX	Horizontal front porch of output timing

pvsp_dp_hsynctime[11:0], Primary VSP Map, *Address 0xE85A[3:0]; Address 0xE85B[7:0]*

This signal sets the Hsync duration of output timing. This register's value will be used while pvsp_autocfg_output_vid is 0.

Function

pvsp_dp_hsynctime[11:0]	Description
0x000 (default)	Default
0xXXX	Hsync width of output timing

pvsp_dp_hbackporch[11:0], Primary VSP Map, *Address 0xE85C[3:0]; Address 0xE85D[7:0]*

This signal is used to set the horizontal back porch duration of output timing. This register's value will be used while pvsp_autocfg_output_vid is 0.

Function

pvsp_dp_hbackporch[11:0]	Description
0x000 (default)	Default
0xXXX	Horizontal back porch of output timing

pvsp_dp_activeline[11:0], Primary VSP Map, *Address 0xE85E[3:0]; Address 0xE85F[7:0]*

This signal is used to set the active line number of output timing. This register's value will be used while pvsp_autocfg_output_vid is 0.

Function

pvsp_dp_activeline[11:0]	Description
0x000 (default)	Default
0xXXX	Active lines of output timing

pvsp_dp_vfrontporch[9:0], Primary VSP Map, *Address 0xE860[1:0]; Address 0xE861[7:0]*
This signal is used to set the vertical front porch duration of output timing. This register's value will be used while pvsp_autocfg_output_vid is 0.

Function

pvsp_dp_vfrontporch[9:0]	Description
0x000 (default)	Default
0xXXX	Vertical front porch of output timing

pvsp_dp_vsynctime[9:0], Primary VSP Map, *Address 0xE862[1:0]; Address 0xE863[7:0]*
This signal is used to set the vertical synchronous time duration of output timing. This register's value will be used while pvsp_autocfg_output_vid is 0.

Function

pvsp_dp_vsynctime[9:0]	Description
0x000 (default)	Default
0xXXX	Vsync width of output timing

pvsp_dp_vbackporch[9:0], Primary VSP Map, *Address 0xE864[1:0]; Address 0xE865[7:0]*
This signal is used to set the vertical back porch duration of output timing. This register's value will be used while pvsp_autocfg_output_vid is 0.

Function

pvsp_dp_vbackporch[9:0]	Description
0x000 (default)	Default
0xXXX	Vertical back porch of output timing

pvsp_dp_vpolarity, Primary VSP Map, *Address 0xE869[0]*
This bit is used to set the polarity of output Vsync. This register's value will be used while pvsp_autocfg_output_vid is 0.

Function

pvsp_dp_vpolarity	Description
0 (default)	Low
1	High

pvsp_dp_hpolarity, Primary VSP Map, *Address 0xE869[1]*
This bit is used to set the polarity of output Hsync. This register's value will be used while pvsp_autocfg_output_vid is 0.

Function

pvsp_dp_hpolarity	Description
0 (default)	Low
1	High

Table 26: Output Port Configuration Settings for Example Output Resolutions

Output Timing	decount		hfrontporch		HSync		hbackporch		activeline		Vfrontporch		VSync		vbackporch		hpol	vpol
	0xE8 56	0xE8 57	0xE8 58	0xE8 59	0xE8 5A	0xE8 5B	0xE8 5C	0xE8 5D	0xE8 5E	0xE8 5F	0xE8 60	0xE8 61	0xE8 62	0xE8 63	0xE8 64	0xE8 65	0xE8 69[1]	0xE8 69[0]
576p	0x02	0xD0	0x00	0x0C	0x00	0x40	0x00	0x44	0x02	0x40	0x00	0x05	0x00	0x05	0x00	0x27	0	0
720p50	0x05	0x00	0x01	0xB8	0x00	0x28	0x00	0xDC	0x02	0xD0	0x00	0x05	0x00	0x05	0x00	0x14	1	1
1080p50	0x07	0x80	0x02	0x10	0x00	0x2C	0x00	0x94	0x04	0x38	0x00	0x04	0x00	0x05	0x00	0x24	1	1
vga	0x02	0x80	0x00	0x10	0x00	0x60	0x00	0x30	0x01	0xE0	0x00	0x0A	0x00	0x02	0x00	0x21	0	0
480p	0x02	0xD0	0x00	0x10	0x00	0x3E	0x00	0x3C	0x01	0xE0	0x00	0x09	0x00	0x06	0x00	0x1E	0	0
720p60	0x05	0x00	0x00	0x6E	0x00	0x28	0x00	0xDC	0x02	0xD0	0x00	0x05	0x00	0x05	0x00	0x14	1	1
1080p60	0x07	0x80	0x00	0x58	0x00	0x2C	0x00	0x94	0x04	0x38	0x00	0x04	0x00	0x05	0x00	0x24	1	1
1080p24	0x07	0x80	0x02	0x7E	0x00	0x2C	0x00	0x94	0x04	0x38	0x00	0x04	0x00	0x05	0x00	0x24	1	1

The size of output images of the VOM scaler can be smaller than that defined by the parameters of the output port (that is, album mode). The starting position for the PVSP output video can be set using pvsp_dp_video_h_start[12:0] and pvsp_dp_video_v_start[12:0]. Figure 61 shows the relationship of the VOM scaler image and the output video. In this case, the blank area around the output image is filled with color defined by pvsp_dp_margin_color[23:0] in the YCbCr color space.

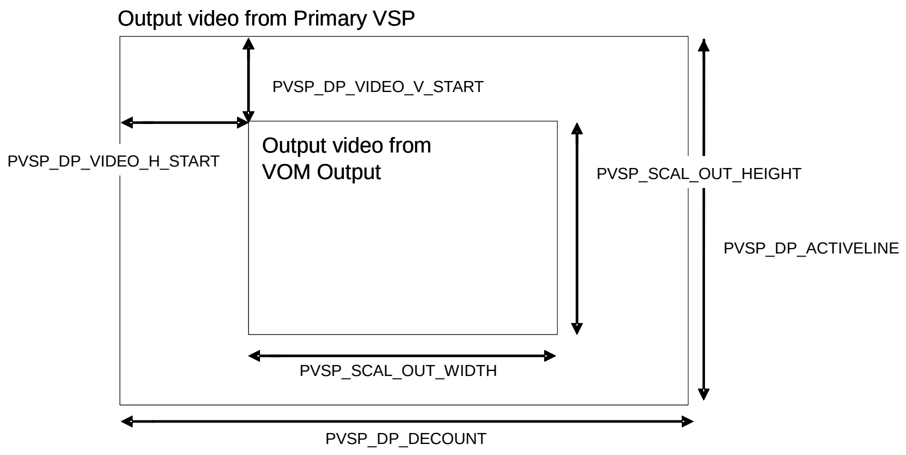


Figure 61: VOM Output Dimensions

pvsp_dp_video_h_start[12:0], Primary VSP Map, *Address 0xE848[4:0]; Address 0xE849[7:0]*
This signal is used to set the horizontal start position where the output video of the scaler is placed.

Function

pvsp_dp_video_h_start[12:0]	Description
0x000 (default)	Default
0xXXX	Horizontal start position of VOM output

pvsp_dp_video_v_start[12:0], Primary VSP Map, *Address 0xE84A[4:0]; Address 0xE84B[7:0]*
This signal is used to set the vertical start position where the output video of scaler is placed.

Function

pvsp_dp_video_v_start[12:0]	Description
0x000 (default)	Default
0xXXX	Vertical start position of VOM output

pvsp_dp_margin_color[23:0], Primary VSP Map, *Address 0xE866[7:0]; Address 0xE867[7:0]; Address 0xE868[7:0]*
This signal is used to set the default color in output video in YUV colorspace.

Function

pvsp_dp_margin_color[23:0]	Description
0x000000	Default
0xXXXXXX	Default color in YUV colorspace

pvsp_dp_output_blank, Primary VSP Map, *Address 0xE869[2]*
This bit is used to force the colour output of the Primary VSP. This If this bit is set to 1, the output of Primary VSP is forced to the user defined color in pvsp_dp_margin_color.

Function

pvsp_dp_output_blank	Description
0 (default)	Not output default color
1	Output default Color

3.2.3.14. Demo Function

ADV8005 supports automatically splitting the display window to demo several processing functions of ADV8005. pvsp_demo_window_enable can be used to enable the demo function.

pvsp_demo_window_enable, Primary VSP Map, *Address 0xE87E[7]*
Enables demo window.

Function

pvsp_demo_window_enable	Description
0 (default)	Disable demo window
1	Enable demo window

pvsp_demo_window_use_lower_screen can be used to set the position of the demo window. If this bit is set to 1, the lower half display window is used for certain processing function, otherwise the upper half display window is used.

pvsp_demo_window_use_lower_screen, Primary VSP Map, *Address 0xE87E[6]*
This bit is used to enable a demo mode on the lower half of the screen. If this bit is set to 1, the lower half display window will be used for certain processing functions, otherwise the upper half display window will be used.

Function

pvsp_demo_window_use_lower_screen	Description
0 (default)	Use upper half screen as demo window
1	Use lower half screen as demo window

The following registers can be used to enable each corresponding demo function.

pvsp_demo_window_rnr_enable, Primary VSP Map, *Address 0xE87E[4]*
This bit is used to enable the RNR in the demo window.

Function

pvsp_demo_window_rnr_enable	Description
0 (default)	Disable RNR in demo window
1	Enable RNR in demo window

pvsp_demo_window_mnr_enable, Primary VSP Map, *Address 0xE87E[3]*
This bit is used to enable the MNR in the demo window.

Function

pvsp_demo_window_mnr_enable	Description
0 (default)	Disable MNR in demo window
1	Enable MNR in demo window

pvsp_demo_window_bnr_enable, Primary VSP Map, *Address 0xE87E[2]*
This bit is used to enable the BNR in the demo window.

Function

pvsp_demo_window_bnr_enable	Description
0 (default)	Disable BNR in demo window
1	Enable BNR in demo window

pvsp_demo_window_cadence_enable, Primary VSP Map, *Address 0xE87E[1]*
This bit is used to enable the cadence detection in the demo window.

Function

pvsp_demo_window_cadence_enable	Description
0 (default)	Disable Cadence detection in demo window
1	Enable Cadence detection in demo window

pvsp_demo_window_ulai_enable, Primary VSP Map, *Address 0xE87E[0]*
This bit is used to enable the ULAI in the demo window.

Function

pvsp_demo_window_ulai_enable	Description
0 (default)	Disable ULAI in demo window
1	Enable ULAI in demo window

pvsp_demo_window_cue_enable, Primary VSP Map, *Address 0xE87F[5]*
This bit is used to enable CUE correction in the demo window.

Function

pvsp_demo_window_cue_enable	**Description**
0 (default)	Disable CUE in demo window
1	Enable CUE in demo window'

pvsp_demo_window_intra_field_enable, Primary VSP Map, *Address 0xE87F[4]*
This bit is used to enable the intra field interpolation in the demo window.

Function

pvsp_demo_window_intra_field_enable	**Description**
0 (default)	Disable intra field interpolation in demo window
1	Enable intra field interpolation in demo window

The contour-based interpolation scaler (2nd generation with 4k x 2k support) demo can be enabled by setting pvsp_srscal_demo_mode_en to compare the contour-based interpolation scaler and the frequency-adaptive scaler (1st generation) performance side by side.

pvsp_srscal_demo_mode_en, Primary VSP Map, *Address 0xE890[4]*
This bit is used to enable scaler demo mode.

Function

pvsp_srscal_demo_mode_en	**Description**
0 (default)	Scaler not in demo mode
1	Scaler in demo mode

3.2.3.15. Progressive to Interlaced Converter

The main progressive to interlaced (PtoI) converter can be connected to many blocks, for example, Video TTL input channel, EXOSD TTL input channel, PVSP, and so on. The block can be used for video conversion, for example, conversion of 1080p to 1080i. It drops the progressive video odd or even lines based on the field signal of the output interlaced video. It can only support 480p, 576p, and 1080p input. The associated interlaced timing signals can be generated in the independent PtoI hardware block.

By enabling m_p2i_drop_line_as_pvsp_flag, the PtoI module can drop interpolated lines to get optimal output performance.

The PtoI hardware can be enabled using m_p2i_enable.

m_p2i_enable, Secondary VSP Map, *Address 0xE649[4]*
This bit is used to enable the PtoI In VSP_top.

Function

m_p2i_enable	**Description**
0 (default)	Disable
1	Enable

m_p2i_drop_line_as_pvsp_flag, Secondary VSP Map, *Address 0xE65B[7]*
In Game Mode, this bit is used to select an interlaced mode. If the PVSP works in game mode and the PVSP's input is interlaced, this bit should be set to 1 for the P2I block to drop interpolated lines. Otherwise, this bit should be set to 0.

In external sync mode, this bit enables field tracking. When this bit is set low, it uses the internally generated field instead of the master one provided.
The input video to the PtoI block is defined using m_p2i_vid[7:0]. Refer to Table 27 for more details on the value of this register.

m_p2i_vid[7:0], Secondary VSP Map, *Address 0xE64B[7:0]*
This register is used to set the VIC of the PtoI in VSP_top.

Function

m_p2i_vid[7:0]	**Description**
0x00 (default)	Default

Table 27: VID Set to PtoI

Input Timing Format to P2I	576p	1080p50	480p	1080p60
svsp_m_p2i_vid	17	31	2	16

The PVSP PtoI does not have direct access to the data from the input pins but it can be utilized to convert a progressive input format to interlaced using the PVSP core bypass path by setting the pvsp_bypass bit.

3.2.3.16. *Automatic Contrast Enhancement*

The Automatic Contrast Enhancement (ACE) block is used to intelligently enhance the contrast of the whole picture by making dark regions darker and bright regions brighter. It is stable under scene changes as well as being robust in the presence of noise. ACE supports both interlaced and progressive inputs and can be enabled/disabled using ace_enable.

ace_enable, IO Map, *Address 0x1A30[7]*
This bit is used to enable the automatic contrast enhancement (ACE) block.

Function

ace_enable	Description
0 (default)	Bypass A.C.E.
1	Enable A.C.E.

3.3. SECONDARY VSP

3.3.1. *Introduction to SVSP*

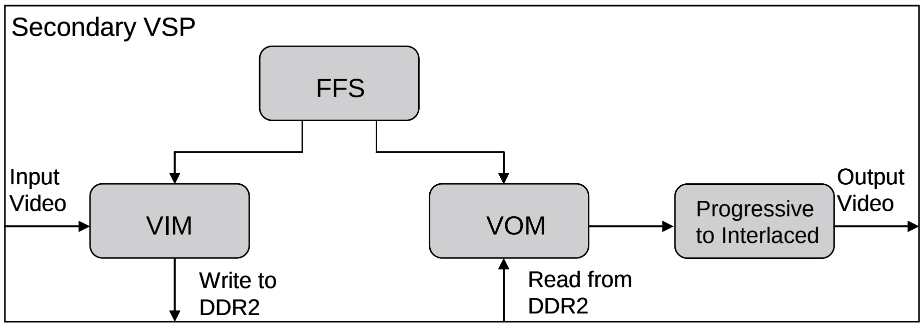


Figure 62: ADV8005 SVSP

Figure 62 shows the structure of the SVSP. The SVSP comprises of four sections; the VIM, the VOM, a controller which is the FFS, and a PtoI converter.

The SVSP can be used to offer the option of a second output resolution to the user. The structure of the SVSP is similar to the PVSP but it is much simpler in design and does not contain all the processing elements of the PVSP. The structure of the SVSP comprises FFS, VIM, and VOM blocks.

Input to the SVSP can only be in progressive format.

The SVSP has the following features:

- Image cropping
- Scaling
- FRC
- PtoI conversion

The image cropping function is the same as that provided in the PVSP and, like the PVSP, there is an image cropper in both the VIM and the

VOM of the SVSP. In the SVSP only, the VIM is capable of scaling video data. This means that the VIM of the SVSP can support vertical resolution scaling as well as horizontal resolution scaling.

The SVSP is also capable of performing FRC, which is controlled by the FFS of the SVSP. The FFS in the SVSP provides the same functionality as the FFS in the PVSP. A PtoI converter which can be used to convert the incoming video standard from progressive to interlaced is also included as part of the SVSP.

Like game mode in PVSP, SVSP can also support bypass DDR mode. Using this mode, the SVSP can convert between 1080p and 720p without using external memory. This allows the user to perform a simple conversion which does not use external memory bandwidth. However, FRC is not supported in this case.

The SVSP can be simply bypassed by setting svsp_bypass to 1.

Note: The input to the SVSP can only be progressive video. Therefore, interlaced video must be routed through the de-interlacer in the PVSP before being routed to the SVSP. The PVSP output can also be sent to the SVSP as a progressive input.

svsp_bypass, Secondary VSP Map, *Address 0xE649[6]*
This bit is used to bypass the Secondary VSP.

Function

svsp_bypass	**Description**
0 (default)	Not bypass Secondary VSP
1	Bypass Secondary VSP

Similarly, if using the SVSP, the VIM and VOM must be enabled. This can be done by enabling svsp_enable_vim and svsp_enable_vom.

svsp_enable_vim, Secondary VSP Map, *Address 0xE610[6]*
This bit is used to control the Video Input Module (VIM). If this bit is set to 1, the VIM is enabled to write packed input video data into the defined external frame buffer. While the Secondary VSP is running, if this bit is set to 0, the output video stream will be frozen.

Function

svsp_enable_vim	**Description**
0 (default)	Disable VIM
1	Enable VIM

svsp_enable_vom, Secondary VSP Map, *Address 0xE610[5]*
This bit is used to control the Video Output Module (VOM). If this bit is set to 1, the VOM is enabled to read video data from external memory, process it and then output it.

Function

svsp_enable_vom	**Description**
0 (default)	Disable VOM
1	Enable VOM

If using the SVSP, the FFS must be enabled (using svsp_enable_ffs) so that the hardware knows the various conversions that must be performed. The use of field/frame buffers in external memory is managed by the FFS which decides which frame buffer should be used by the VIM to store input video data. The FFS also decides which frame buffer should be read back by the VOM. The SVSP utilizes a frame repeat/drop mechanism to implement FRC, which is also managed by the FFS.

svsp_enable_ffs, Secondary VSP Map, *Address 0xE610[7]*
This bit is used to control the Field Frame Scheduler (FFS). If this bit is set to 1, the FFS is enabled and the VIM and VOM are scheduled by the FFS, which means the Secondary VSP is in work mode. If this bit is set to 0, the Secondary VSP is in idle mode.

Function

svsp_enable_ffs	**Description**
0 (default)	Disable FFS/FRC
1	Enable FFS/FRC

3.3.1.1. Autoconfiguration

Each block inside the VIM and the VOM can be automatically configured to decrease the configuration complexity. The svsp_autocfg_input_vid[7:0] and svsp_autocfg_output_vid[7:0] registers should be set to make the autoconfiguration work. The 59.94/23.97 Hz timings have the same VID as the corresponding 60/24 Hz timing in Table 28.

svsp_autocfg_input_vid[7:0], Secondary VSP Map, *Address 0xE660[7:0]*
This register is used to set the input timing VIC. If this register is 0, SVSP will use values in registers of svsp_vin_h, svsp_vin_v and svsp_vin_fr to set input video.

Function

svsp_autocfg_input_vid[7:0]	**Description**
0x00 (default)	Custom input video;
0xXX	Input timing VIC

Table 28: SVSP Supported Input Video Timing and VID

Video Timing		**VID**
CEA	640x480p60	1
	720x480p60	2 or 3 or 14 or 15 or 35 or 36
	720x240p60	8 or 9 or 12 or 13
	1280x720p60	4
	1920x1080p	16
	720x576p50	17 or 18 or 29 or 30 or 37 or 38
	1280x720p50	19
	720x288p50	23 or 24 or 27 or 28
	1920x1080p50	31
	1920x1080p24	32
	1920x1080p25	33
	1920x1080p30	34
	720p100	41
	576p100	42 or 43
	720p120	47
	480p120	48 or 49
	576p200	52 or 53
	480p240	56 or 57
VESA timing	VGA	200
	SVGA	201
	XGA	202
	WXGA	203
	SXGA	204
	WXGA-2	205
	UXGA	206
	WXGA-3	207
	WUXGA	208

Note: The SVSP does not support the following formats:

7. 1280x720p @ 23.97/24 Hz (CEA VIC 60)
8. 1280x720p @ 25 Hz (CEA VIC 61)
9. 1280x720p @ 29.97/30 Hz (CEA VIC 62)

svsp_autocfg_output_vid[7:0], Secondary VSP Map, *Address 0xE661[7:0]*
This register is used to set the output timing VIC. If this register is 0, SVSP will use values in registers of svsp_dp_decount, svsp_dp_hfrontporch, svsp_dp_hsynctime, svsp_dp_hbackporch, svsp_dp_activeline, svsp_dp_vfrontporch, svsp_dp_vsynctime, svsp_dp_vbackporch, svsp_dp_hpolarity, svsp_dp_vpolarity and svsp_vout_fr to set output video.

Function

svsp_autocfg_output_vid[7:0]	**Description**
0x00 (default)	Custom output video;
0xXX	Output timing VIC

Table 29 lists all the supported video timings and their VID. The 59.94/23.97 Hz timings have the same VID as the corresponding 60/24 Hz timing in the table.

Table 29: SVSP Supported Output Video Timing and VID

Video Timing		**VID**
CEA	640x480p60	1
	720x480p60	2 or 3 or 14 or 15 or 35 or 36
	720(1440)x240p60	8 or 9
	720(2880)x240p60	12 or 13
	1280x720p60	4
	1920x1080i60	5
	720x480i60	6 or 7 or 10 or 11
	1920x1080p	16
	720x576p50	17 or 18 or 29 or 30 or 37 or 38
	1280x720p50	19
	1920x1080i50	20
	720x576i50	21 or 22 or 25 or 26
	720x288p50	23 or 24 or 27 or 28
	1920x1080p50	31
	1920x1080p24	32
	1920x1080p25	33
	1920x1080p30	34
	720p100	41
	576p100	42 or 43
	720p120	47
	480p120	48 or 49
	576p200	52 or 53
	480p240	56 or 57
VESA timing	VGA	200
	SVGA	201
	XGA	202
	WXGA	203
	SXGA	204
	WXGA-2	205
	UXGA	206
	WXGA-3	207
	WUXGA	208

Note: The SVSP does not support the following formats;

10. 1280x720p @ 23.97/24 Hz (CEA VIC 60)
11. 1280x720p @ 25 Hz (CEA VIC 61)
12. 1280x720p @ 29.97/30 Hz (CEA VIC 62)

If overscan, crop or album mode is employed, the required blocks should be configured manually by enabling the corresponding enable bits, such as svsp_vim_crop_enable, to enable the VIM crop block.

3.3.1.2. *Customized Input/Output Video Format Configuration*

If the input timing is not in the SVSP input format table, the input format needs to be set manually. If the input resolution has a variation in regard to standard timing (for example, if svsp_autocfg_input_vid[7:0] is set to 2, which indicates the input resolution is 720x480, but the actual resolution is 718x478), the user can manually set svsp_autocfg_input_vid[7:0] to be 0 and set the input resolution through the following three registers.

svsp_man_input_res, Secondary VSP Map, *Address 0xE663[4]*
This bit is used to enable manual configuration of input resolution.

Function

svsp_man_input_res	**Description**
0 (default)	Disable manual configuration of input resolution
1	Enable manual configuration of input resolution

svsp_vin_h[12:0], Secondary VSP Map, *Address 0xE616[7:0]; Address 0xE617[7:3]*
This signal is used to set the horizontal resolution of the input video. This register's value will be used while svsp_man_input_res is 1 or svsp_autocfg_input_vid is 1.

Function

svsp_vin_h[12:0]	**Description**
0x000 (default)	Default
0xXXX	Horizontal resolution of input video

svsp_vin_v[12:0], Secondary VSP Map, *Address 0xE618[7:0]; Address 0xE619[7:3]*
This signal is used to set the vertical resolution of the input video. This register's value will be used while svsp_man_input_res is 1 or svsp_autocfg_input_vid is 1.

Function

svsp_vin_v[12:0]	**Description**
0x000 (default)	Default
0xXXX	Vertical resolution of input video

Similarly, if the output timing is not in the SVSP output format table, the output format needs to be set manually. The detailed configuration instruction is given in the SVSP VOM output port description.

3.3.1.3. *Frame Buffer Number*

Depending on the type of conversion that is to take place, a certain number of buffers must be allocated for the input/output video data. Depending on the conversion required, this should be set in the svsp_fieldbuf_num[2:0] register. svsp_fieldbuf_num[2:0] can be automatically set per svsp_autocfg_input_vid[7:0] and svsp_autocfg_output_vid[7:0]. The svsp_fieldbuf_num[2:0] register will not change when crop or album mode is enabled.

svsp_fieldbuf_num[2:0], Secondary VSP Map, *Address 0xE610[2:0]*
This signal is used to set the number of field/frame buffers. This signal needs to be configured while svsp_osd_mode_en is 1.

Function

svsp_fieldbuf_num[2:0]	**Description**
000 (default)	Default
XXX	Number of field/frame buffers

3.3.1.4. *Frame Buffer Address and Size*

In order to store video data in external memory in the correct size frames, the buffer size of the external DDR2 memory must be programmed by the user. These programmed field buffers or frame buffers are allocated by setting the svsp_fieldbuffer0_addr[31:0], svsp_fieldbuffer1_addr[31:0], svsp_fieldbuffer2_addr[31:0] and
svsp_fieldbuffer3_addr[31:0] registers.

The value programmed into each of these registers is determined in Equation 22.

$$frame\_size = active\_video\_width \times active\_video\_height \times no\_bytes\_per\_pixel$$

Equation 22: Calculating External Memory Field Buffers

For example, for an output video resolution of 720p, Equation 22 would yield the following field size:

Field_size = ((1280)x(720))x2 = 1843200

where no_bytes_per_pixel indicates the number of bytes required to store each pixel. Refer to Table 23 for more details of the number of bytes required to store each pixel of data.

svsp_fieldbuffer0_addr[31:0], Secondary VSP Map, *Address 0xE600[7:0]; Address 0xE601[7:0]; Address 0xE602[7:0]; Address 0xE603[7:0]*
This signal is used to set the start address of frame buffer 0. Software should arrange memory space properly, avoiding conflict between different buffers.

Function

svsp_fieldbuffer0_addr[31:0]	Description
0x00000000	Default
0xXXXXXXXX	Start address of frame buffer 0

svsp_fieldbuffer1_addr[31:0], Secondary VSP Map, *Address 0xE604[7:0]; Address 0xE605[7:0]; Address 0xE606[7:0]; Address 0xE607[7:0]*
This signal is used to set the start address of frame buffer 1. Software should arrange memory space properly, avoiding conflict between different buffers.

Function

svsp_fieldbuffer1_addr[31:0]	Description
0x00000000	Default
0xXXXXXXXX	Start address of frame buffer 1

svsp_fieldbuffer2_addr[31:0], Secondary VSP Map, *Address 0xE608[7:0]; Address 0xE609[7:0]; Address 0xE60A[7:0]; Address 0xE60B[7:0]*
This signal is used to set the start address of frame buffer 2. Software should arrange memory space properly, avoiding conflict between different buffers.

Function

svsp_fieldbuffer2_addr[31:0]	Description
0x00000000	Default
0xXXXXXXXX	Start address of frame buffer 2

svsp_fieldbuffer3_addr[31:0], Secondary VSP Map, *Address 0xE60C[7:0]; Address 0xE60D[7:0]; Address 0xE60E[7:0]; Address 0xE60F[7:0]*
This signal is used to set the start address of frame buffer 3. Software should arrange memory space properly, avoiding conflict between different buffers.

Function

svsp_fieldbuffer3_addr[31:0]	Description
0x00000000	Default
0xXXXXXXXX	Start address of frame buffer 3

svsp_fieldbuffer4_addr[31:0], Secondary VSP Map, *Address 0xE664[7:0]; Address 0xE665[7:0]; Address 0xE666[7:0]; Address 0xE667[7:0]*
This signal is used to set the start address of field/frame buffer 4. Software should arrange memory space properly, avoiding conflict between different buffers.'

Function

svsp_fieldbuffer4_addr[31:0]	Description
0x00000000	Default
0xXXXXXXXX	Start address of frame buffer 4

svsp_fieldbuffer5_addr[31:0], Secondary VSP Map, *Address 0xE668[7:0]; Address 0xE669[7:0]; Address 0xE66A[7:0]; Address 0xE66B[7:0]*
This signal is used to set the start address of field/frame buffer 5. Software should arrange memory space properly, avoiding conflict between different buffers.'

Function

svsp_fieldbuffer5_addr[31:0]	Description
'0x00000000	Default
0xXXXXXXXX	Start address of frame buffer 5

svsp_fieldbuffer6_addr[31:0], Secondary VSP Map, *Address 0xE66C[7:0]; Address 0xE66D[7:0]; Address 0xE66E[7:0]; Address 0xE66F[7:0]*
This signal is used to set the start address of field/frame buffer 6. Software should arrange memory space properly, avoiding conflict between different buffers.'

Function

svsp_fieldbuffer6_addr[31:0]	**Description**
0x00000000	Default
0xXXXXXXXX	Start address of frame buffer 6

3.3.1.5. *Frame Latency*

Depending on the format being input to the ADV8005 and the output required from the SVSP, different resolutions will have different frame latencies. This is due to the increased processing required on scaling different types of video data. This has a certain impact in that the audio will have to be delayed by the same amount. Table 30 lists frame latencies in different cases for various resolutions.

Table 30: Frame Latency in Normal Mode

Output / Input	**Frame Rate**	**50 Hz**	**59.94/60 Hz**	**23.97/24 Hz**	**25/30 Hz**
Frame Rate	**Timing**	**576p/720p/1080p**	**480p/720p/1080p**	**720p/1080p**	**720p/1080p**
50 Hz	576p 720p 1080p	0.1~1.3[1, 2]	0.1~1.3	0.1~1.4	0.1~1.4
59.94/60 Hz	480p 720p 1080p	0.1~1.3	0.1~1.3	0.1~1.4	0.1~1.4
23.97/24/25/30Hz	720/1080p	0.1~0.8	0.1~0.8	0.1~1.3	0.1~1.3

1. x.x means x.x times the input video frame
2. A~B means frame latency is not a fixed value, it varies between A and B

When crop or album mode is enabled, frame latency will be different from that listed in Table 30. In this case, the user can use the following controls to measure frame latency: svsp_rb_frame_latency[2:0] and svsp_rb_hsync_latency[11:0] are read only registers. Their values are real-time frame and HSync latency between input and output video.

Frame latency may vary within a range; the svsp_rb_max_latency[14:0] readback register indicates the maximum frame latency, while svsp_rb_min_latency[14:0] indicates the minimum frame latency. If svsp_frc_latency_measure_en is set to 0, svsp_rb_max_latency[14:0] and svsp_rb_min_latency[14:0] are cleared. If asserting svsp_frc_latency_measure_en, SVSP monitors values in svsp_rb_max_latency[14:0] and svsp_rb_min_latency[14:0] and then records the maximum and minimum values of them in the svsp_rb_max_latency[14:0] and svsp_rb_min_latency[14:0] registers which are both 15 bits wide. The highest three bits are the frame latency and the lower 12 bits are the HSync latency. Users should note that it will take several seconds for the SVSP to find the maximum and minimum frame/HSync latency.

In a normal case (not game mode), the SVSP's input video and output video latency is consistent.

svsp_frc_latency_measure_en, Secondary VSP Map, *Address 0xE662[2]*
This bit is used to enable measuring frame/Hsync latency.

Function

svsp_frc_latency_measure_en	Description
0 (default)	Disable
1	Enable

svsp_rb_frame_latency[2:0], Secondary VSP Map, *Address 0xE6F2[7:5] (Read Only)*
This signal is used to readback the realtime frame latency.

Function

svsp_rb_frame_latency[2:0]	Description
0xXXX	Frame latency

svsp_rb_hsync_latency[11:0], Secondary VSP Map, *Address 0xE6F3[7:0]; Address 0xE6F4[7:4] (Read Only)*
This signal is used to readback the realtime Hsync latency.

Function

svsp_rb_hsync_latency[11:0]	Description
0xXXX	HSync latency

svsp_rb_max_latency[14:0], Secondary VSP Map, *Address 0xE6F5[7:0]; Address 0xE6F6[7:1] (Read Only)*
This signal is used to readback the maximum frame/Hsync latency. Upper 3 bit is VS latency, Lower 12 bit HS latency.

Function

svsp_rb_max_latency[14:0]	Description
0xXXX	Maximum of frame latency

svsp_rb_min_latency[14:0], Secondary VSP Map, *Address 0xE6F7[7:0]; Address 0xE6F8[7:1] (Read Only)*
This signal is used to readback the minimum frame/Hsync latency. Upper 3 bit is VS latency, Lower 12 bit HS latency.

Function

svsp_rb_min_latency[14:0]	Description
0xXXX	Minimum of frame latency

3.3.1.6. *Freezing Output Video*

Output video can be frozen by disabling the VIM by setting svsp_enable_vim to 0.

3.3.2. *SVSP Video Input Module (VIM)*

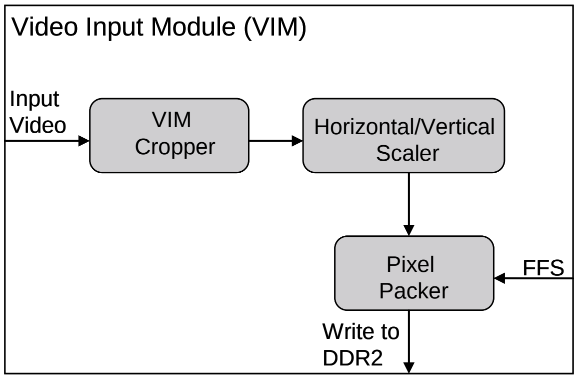


Figure 63: SVSP Video Input Module

Figure 63 shows the structure of the SVSP VIM. This can be broken up into three hardware blocks. The VIM cropper can be used to crop an input video image to a given image size. The scaler can be used to scale a video resolution to any target resolution. The pixel packer is used to pack pixels data into memory words and write them into external memory. The starting address in external memory is provided by FFS and is configured by the user using the frame buffer registers. As indicated at the start of Section 3.3, in order for the VIM module to operate, it must first be enabled. This can be done using the svsp_enable_vim bit. If the VIM is disabled by setting this register to 0, the output video will be frozen.

3.3.2.1. *VIM Cropper*

The VIM cropper block is used to define a sub window within the given input resolution. This cropped image will then become the video which will be processed by the SVSP. The following registers are used to define this sub window.

- svsp_vim_crop_enable
- svsp_vim_crop_h_start[12:0]
- svsp_vim_crop_v_start[12:0]
- svsp_vim_crop_width[12:0]
- svsp_vim_crop_height[12:0]

To enable cropper block in VIM, svsp_vim_crop_enable must be set to 1.

svsp_vim_crop_enable, Secondary VSP Map, *Address 0xE662[6]*
This bit is used to enables the VIM crop.

Function

svsp_vim_crop_enable	Description
0 (default)	Disable
1	Enable

Figure 64 shows the correlation between the cropped image and the input video resolution.

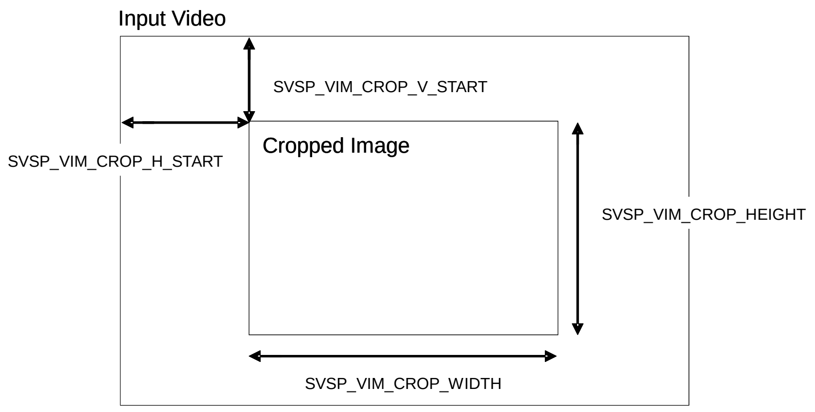


Figure 64: VIM Crop Dimensions

svsp_vim_crop_h_start[12:0], Secondary VSP Map, *Address 0xE61A[7:0]; Address 0xE61B[7:3]*
Sets the horizontal start position of the VIM cropper.

Function

svsp_vim_crop_h_start[12:0]	Description
0x000 (default)	Default
0xXXX	Horizontal start position of VIM cropper input

svsp_vim_crop_v_start[12:0], Secondary VSP Map, *Address 0xE61C[7:0]; Address 0xE61D[7:3]*
This signal is used to set the horizontal start position of the VIM cropper.

Function

svsp_vim_crop_v_start[12:0]	Description
0x000 (default)	Default
0xXXX	Vertical start position of VIM cropper input

svsp_vim_crop_width[12:0], Secondary VSP Map, *Address 0xE61E[7:0]; Address 0xE61F[7:3]*
This signal is used to set the input width of the VIM cropper.

Function

svsp_vim_crop_width[12:0]	**Description**
0x000 (default)	Default
0xXXX	Width of VIM cropper input

svsp_vim_crop_height[12:0], Secondary VSP Map, *Address 0xE620[7:0]; Address 0xE621[7:3]*
This signal is used to set the input height of the VIM cropper.

Function

svsp_vim_crop_height[12:0]	**Description**
0x000 (default)	Default
0xXXX	Height of VIM cropper input

Note: The following limitations apply to the values that can be programmed in these registers:

- Register values programmed must be even numbers
- 0 <= svsp_vim_crop_h_start[12:0] <= (INPUT VIDEO HORIZONTAL RESOLUTION – 1)
- 0 <= svsp_vim_crop_v_start[12:0] <= (INPUT VIDEO VERTICAL RESOLUTION – 1)
- (svsp_vim_crop_h_start[12:0] + svsp_vim_crop_width[12:0]) <= INPUT VIDEO HORIZONTAL ACTIVE PIXELS
- (svsp_vim_crop_v_start[12:0] + svsp_vim_crop_height[12:0]) <= INPUT VIDEO VERTICAL ACTIVE PIXELS

3.3.2.2. *Scaler*

The size of the active image being sent to the SVSP is configured using svsp_vim_crop_height[12:0] and svsp_vim_crop_width[12:0] as mentioned in Section 3.3.2. The output of the SVSP scaler can be set using svsp_vim_scal_out_height[10:0] and svsp_vim_scal_out_width[10:0], or it can be automatically set per svsp_autocfg_output_vid[7:0]. These registers should be set to the resolution of the output video.

svsp_man_scal_out_enable, Secondary VSP Map, *Address 0xE662[5]*
This bit is used to enable manually setting scaler output resolution.

Function

svsp_man_scal_out_enable	**Description**
0 (default)	Disable
1	Enable

svsp_vim_scal_out_height[10:0], Secondary VSP Map, *Address 0xE624[7:0]; Address 0xE625[7:5]*
This signal is used to set the output vertical resolution of scaler in the VIM.

Function

svsp_vim_scal_out_height[10:0]	**Description**
0x000 (default)	Default
0xXXX	Output height of VIM scaler

svsp_vim_scal_out_width[10:0], Secondary VSP Map, *Address 0xE622[7:0]; Address 0xE623[7:5]*
This signal is used to set the output horizontal resolution of scaler in the VIM.

Function

svsp_vim_scal_out_width[10:0]	**Description**
0x000 (default)	Default
0xXXX	Output width of VIM scaler

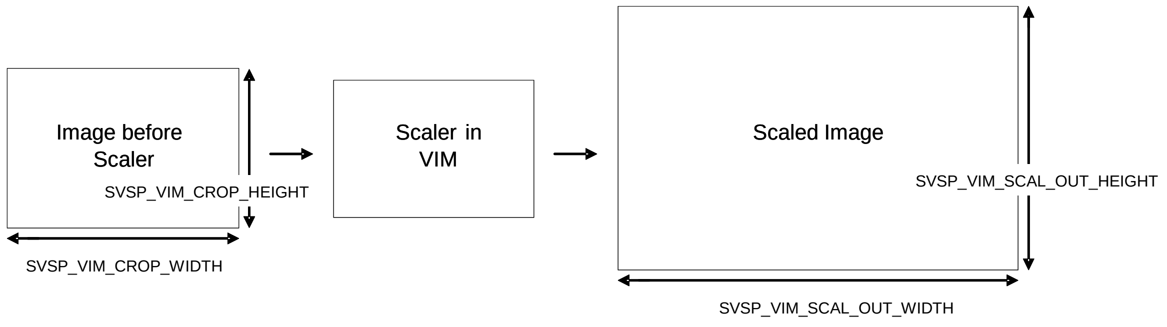


Figure 65: VIM Scaler Dimensions

3.3.2.3. Scaler Interpolation Mode

This section describes the method for scaling the input video data. The purpose of the scaler is to allow different input formats to be displayed on a screen with a fixed resolution. This can allow lower resolution video, for example, 480p, to be upscaled to a high definition format such as 1080p. This can improve the overall quality of a video signal when displayed on a high definition television. The four options of video scaling are listed below and are chosen using svsp_vim_scal_type[1:0].

Refer to Section 3 for more information on the types of scaler algorithm.

svsp_vim_scal_type[1:0], Secondary VSP Map, *Address 0xE646[7:6]*
This signal is used to set the VIM scaling algorithm. In most cases, the scaler type should be left at the default setting.

Function

svsp_vim_scal_type[1:0]	**Description**
00 (default)	Proprietary ADI Algorithm
01	Sharp
10	Smooth
11	Bilinear

3.3.2.4. VIM Miscellaneous Control

The following registers are used in the control of the VIM scaling function and should be tailored according to user requirements.

Anti-alias filters are provided to improve the performance of the SVSP downscaling and can be enabled using svsp_vim_scal_anti_alias_h_en and svsp_vim_scal_anti_alias_v_en.

svsp_vim_scal_anti_alias_h_en, Secondary VSP Map, *Address 0xE650[5]*
This bit is used to enable the anti-aliasing filter for horizontal direction.

Function

svsp_vim_scal_anti_alias_h_en	**Description**
0	Disable
1 (default)	Enable

svsp_vim_scal_anti_alias_v_en, Secondary VSP Map, *Address 0xE650[6]*
This bit is used to enable anti-aliasing filter for vertical direction.

Function

svsp_vim_scal_anti_alias_v_en	**Description**
0	Disable
1 (default)	Enable

svsp_vim_scal_type[1:0], svsp_vim_scal_anti_alias_h_en and svsp_vim_scal_anti_alias_v_en can be manually set. These settings take effect only when svsp_man_scaler_para_enable is set to 1, otherwise they can be automatically configured by the SVSP using svsp_autocfg_input_vid[7:0] and svsp_autocfg_output_vid[7:0].

svsp_man_scaler_para_enable, Secondary VSP Map, *Address 0xE662[4]*
This bit is used to enable manually setting scaler parameters.

Function

svsp_man_scaler_para_enable	Description
0 (default)	Disable
1	Enable

When a picture is zoomed in, it is possible to maintain the original high frequency content. However, maintaining this content can sometimes introduce ringing artifacts. This overshoot can be controlled by adjusting svsp_vim_scal_overshoot_ctrl[11:0] according to user preference.

svsp_vim_scal_overshoot_ctrl[11:0], Secondary VSP Map, *Address 0xE647[7:0]; Address 0xE648[7:4]*
This signal is used to control the overshoot in the scaling of input video. If set to a value larger than the default setting, more overshoot is allowed.

Function

svsp_vim_scal_overshoot_ctrl[11:0]	Description
0x080 (default)	Default

3.3.2.5. *Panorama Mode*

This feature is the same as for the PVSP. If the scaled video has a different aspect ratio to the original and the horizontal scaling factor is larger than the vertical one, the panorama function can be enabled using svsp_vim_scal_pano_en. In effect, this stretches the left- and right-most sides of the input video to fill the output resolution. This method keeps the original ratio in the centre of the screen. Figure 60 explains the panorama mode scaling feature.

svsp_vim_scal_pano_en, Secondary VSP Map, *Address 0xE650[7]*
This bit is used to enable panorama scaling for the Secondary VSP.

Function

svsp_vim_scal_pano_en	Description
0 (default)	Disable panorama
1	Enable panorama

The position from which the output video becomes stretched is controlled using
svsp_vim_scal_pano_pos[10:0]. This allows the user to control the width of the sides of the output image. Refer to Figure 60 for more details.

svsp_vim_scal_pano_pos[10:0], Secondary VSP Map, *Address 0xE651[7:0]; Address 0xE652[7:5]*
This signal is used to define the width of the output video frame which is not stretched when panorama mode is enabled but rather scaled properly. The maximum value of this register is set by: svsp_vim_crop_width * (svsp_vim_scal_out_height/svsp_vim_crop_height) - svsp_vim_scal_out_width/2.

This register sets half the width of the output frame which is to be scaled normally. By default, this register is set to 0 which means that all the input frame will be stretched. It is, therefore, recommended that this register is set by the user before enabling the panorama function.

Function

svsp_vim_scal_pano_pos[10:0]	Description
0x000 (default)	Default
0xXXX	Width of not-stretched image

3.3.2.6. *Pixel Packer*

At the back end of the VIM, the pixel packer converts input video to word packets suitable for writing to external memory. The operation of this hardware block is similar to the pixel packer in the PVSP. The SVSP manages pixels in 8-bit precision. Pixels in external memory have two different data formats which can be selected using svsp_ex_mem_data_format[1:0]:

- 24-bit YCbCr
- 16-bit YCbCr-4:2:2

svsp_ex_mem_data_format[1:0], Secondary VSP Map, *Address 0xE611[7:6]*
This signal is used to set the data format in external memory.

Function

svsp_ex_mem_data_format[1:0]	**Description**
01	YCbCr-8b-8b-8b
11	YCbCr-4:2:2-8b

3.3.3. *SVSP Video Output Module*

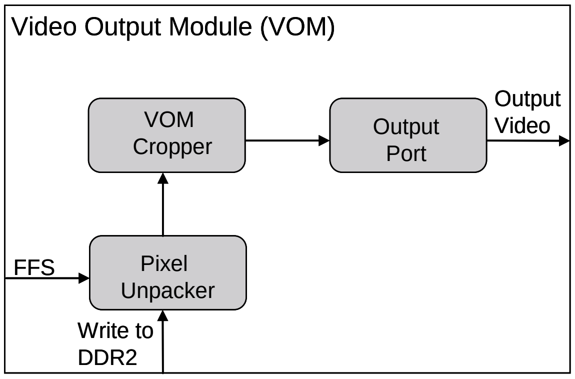


Figure 66: SVSP Video Output Module (VOM)

Figure 66 shows the structure of the VOM in the SVSP. This is a much simpler structure than that of the VOM in the PVSP.

The SVSP VOM offers the following functions:

- Pixel unpacker: reads field/frame from external memory and unpacks memory word to video pixel information
- VOM cropper: reads cropped images from external memory
- Output port: generates output timing and output video

Register update protection is provided in the ADV8005. Refer to Section 3.4 for more details regarding the update of the various VSP registers.

svsp_lock_vom, Secondary VSP Map, *Address 0xE610[4]*
'This bit is used to lock the Video Output Module (VOM). If the Secondary VSP is running and this bit is set to 1, the VOM will be locked to the current register setting to display the last frame. The Secondary VSP registers can be configured safely in this state. All new register settings will be updated after this bit is set back to 0.

Function

svsp_lock_vom	**Description**
0 (default)	Unlock VOM
1	Lock VOM

Note: This register should be used only as part of the gentle reboot protocol. Refer to Section 3.4.3 for more details.

svsp_update_vom, Secondary VSP Map, *Address 0xE610[3]*
Registers related to the VOM can be updated only when this bit is set to 0. All new register settings will be updated by VOM in next frame after this bit is set back to 1.

Function

svsp_update_vom	**Description**
0 (default)	Do not update VOM
1	Update VOM

3.3.3.1. *Pixel Unpacker*

The pixel unpacker in the VOM of the SVSP is similar to that in the VOM of the PVSP. The pixel unpacker is used to convert external memory words (128 bits) into video pixel (YCbCr-8-8-8-bit) data. Pixels in external memory can have the following two different data formats which are the same as those set by the VIM. This is configured in the same way as the VIM.

- 24-bit YCbCr
- 16-bit YCbCr 4:2:2

Data format details are described in svsp_ex_mem_data_format[1:0].

3.3.3.2. *VOM Cropper*

The VOM cropper is also very similar to the cropper in the VOM of the PVSP. The following registers are used to configure the VOM cropper.

- svsp_vom_crop_enable
- svsp_vom_crop_h_start[10:0]
- svsp_vom_crop_v_start[10:0]
- svsp_vom_crop_width[10:0]
- svsp_vom_crop_height[10:0]

The function of the VOM cropper can be seen in Figure 67. To enable the cropper in the SVSP VOM, svsp_vom_crop_enable should be asserted.

svsp_vom_crop_enable, Secondary VSP Map, *Address 0xE662[1]*

This bit is used to enable the VOM crop.

Function

svsp_vom_crop_enable	**Description**
0 (default)	Disable
1	Enable

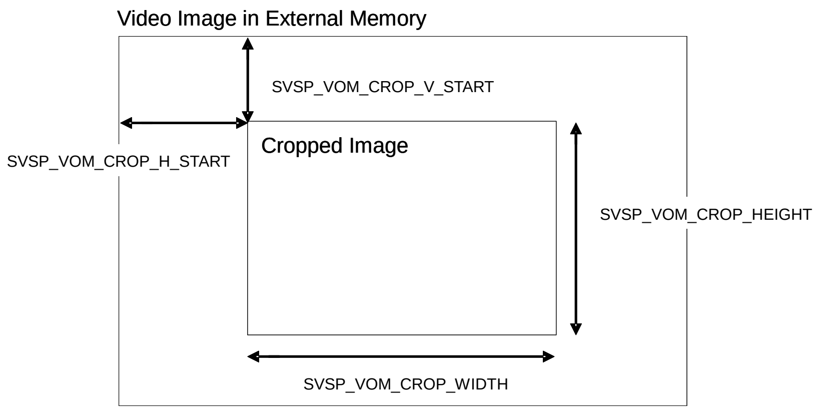


Figure 67: VOM Crop Dimensions

svsp_vom_crop_h_start[10:0], Secondary VSP Map, *Address 0xE626[7:0]; Address 0xE627[7:5]*

This signal is used to set the horizontal start position of the VOM cropper.

Function

svsp_vom_crop_h_start[10:0]	**Description**
0x000 (default)	Default
0xXXX	Horizontal start position of VOM cropper

svsp_vom_crop_v_start[10:0], Secondary VSP Map, *Address 0xE628[7:0]; Address 0xE629[7:5]*
This signal is used to set the vertical start position of the VOM cropper.

Function

svsp_vom_crop_v_start[10:0]	Description
0x000 (default)	Default
0xXXX	Vertical start position of VOM cropper

svsp_vom_crop_width[10:0], Secondary VSP Map, *Address 0xE62A[7:0]; Address 0xE62B[7:5]*
This signal is used to set the width of the VOM cropper.

Function

svsp_vom_crop_width[10:0]	Description
0x000 (default)	Default
0xXXX	Width of VOM cropper input

svsp_vom_crop_height[10:0], Secondary VSP Map, *Address 0xE62C[7:0]; Address 0xE62D[7:5]*
This signal is used to set the height of the VOM cropper.

Function

svsp_vom_crop_height[10:0]	Description
0x000 (default)	Default
0xXXX	Height of VOM cropper input

Note: The following restrictions apply to the values at which these registers can be set:

- All registers should contain even values
- 0 <= svsp_vom_crop_h_start[10:0] <= (HORIZONTAL RESOLUTION OUTPUT BY VIM – 1)
- 0 <= svsp_vom_crop_v_start[10:0] <= (RESOLUTION OUTPUT BY VIM – 1)
- (svsp_vom_crop_h_start[10:0] + svsp_vom_crop_width[10:0]) <= HORIZONTAL RESOLUTION OUTPUT BY VIM
- (svsp_vom_crop_v_start[10:0] + svsp_vom_crop_height[10:0]) <= VERTICAL RESOLUTION OUTPUT BY VIM

3.3.3.3. Output Port

This section describes the configuration registers for the final block of the VOM of the SVSP. The main purpose of the output port is to generate the output video timing and output the video data. For more details regarding the various register settings for the output port for various common video formats, refer to Table 31. The output setting can be automatically configured using svsp_autocfg_output_vid[7:0]. If the output configuration is to be set manually, svsp_man_dp_timing_enable should be set to 1. Refer to Figure 68 for more information.

svsp_man_dp_timing_enable, Secondary VSP Map, *Address 0xE663[7]*
This bit is used to enable manually setting output timing.

Function

svsp_man_dp_timing_enable	Description
0 (default)	Disable
1	Enable

svsp_dp_decount[10:0], Secondary VSP Map, *Address 0xE632[7:0]; Address 0xE633[7:5]*
This signal is used to set the DE duration of output timing. This register's value will be used while svsp_autocfg_output_vid is 0.

Function

svsp_dp_decount[10:0]	Description
0x000 (default)	Default
0xXXX	Data enable count of output timing

svsp_dp_hfrontporch[11:0], Secondary VSP Map, *Address 0xE634[7:0]; Address 0xE635[7:4]*
This signal is used to set the horizontal front porch duration of output timing. This register's value will be used while svsp_autocfg_output_vid is 0.

Function

svsp_dp_hfrontporch[11:0]	**Description**
0x000 (default)	Default
0xXXX	Horizontal front porch of output timing

svsp_dp_hsynctime[9:0], Secondary VSP Map, *Address 0xE636[7:0]; Address 0xE637[7:6]*
This signal is used to set the Hsync duration of output timing. This register's value will be used while svsp_autocfg_output_vid is 0.

Function

svsp_dp_hsynctime[9:0]	**Description**
0x000 (default)	Default
0xXXX	Hsync width of output timing

svsp_dp_hbackporch[9:0], Secondary VSP Map, *Address 0xE638[7:0]; Address 0xE639[7:6]*
This signal is used to set the horizontal back porch duration of output timing. This register's value will be used while svsp_autocfg_output_vid is 0.

Function

svsp_dp_hbackporch[9:0]	**Description**
0x000 (default)	Default
0xXXX	Horizontal back porch of output timing

svsp_dp_activeline[10:0], Secondary VSP Map, *Address 0xE63A[7:0]; Address 0xE63B[7:5]*
This signal is used to set the active line number of output timing. This register's value will be used while svsp_autocfg_output_vid is 0.

Function

svsp_dp_activeline[10:0]	**Description**
0x000 (default)	Default
0xXXX	Active lines of output timing

svsp_dp_vfrontporch[9:0], Secondary VSP Map, *Address 0xE63C[7:0]; Address 0xE63D[7:6]*
This signal is used to set the vertical front porch duration of output timing. This register's value will be used while svsp_autocfg_output_vid is 0.

Function

svsp_dp_vfrontporch[9:0]	**Description**
0x000 (default)	Default
0xXXX	Vertical front porch of output timing

svsp_dp_vsynctime[9:0], Secondary VSP Map, *Address 0xE63E[7:0]; Address 0xE63F[7:6]*
This signal is used to set the vertical synchronous time of output timing. This register's value will be used while svsp_autocfg_output_vid is 0.

Function

svsp_dp_vsynctime[9:0]	**Description**
0x000 (default)	Default
0xXXX	Vsync width of output timing

svsp_dp_vbackporch[9:0], Secondary VSP Map, *Address 0xE640[7:0]; Address 0xE641[7:6]*
This signal is used to set the vertical back porch duration of output timing. This register's value will be used while svsp_autocfg_output_vid is 0.

Function

svsp_dp_vbackporch[9:0]	**Description**
0x000 (default)	Default
0xXXX	Vertical back porch of output timing

svsp_dp_vpolarity, Secondary VSP Map, *Address 0xE642[7]*
This signal is used to set the polarity of output Vsync. This register's value will be used while svsp_autocfg_output_vid is 0.

Function

svsp_dp_vpolarity	Description
0 (default)	Low
1	High

svsp_dp_hpolarity, Secondary VSP Map, *Address 0xE642[6]*
This signal is used to set the polarity of output Hsync. This register's value will be used while svsp_autocfg_output_vid is 0.

Function

svsp_dp_hpolarity	Description
0 (default)	Low
1	High

Table 31: Output Port Configuration Settings for Example Output Formats

Output Timing	decount		hfrontporch		HSync		hbackporch		activeline		Vfrontporch		VSync		vbackporch		vpol	hpol
	0xE 632	0xE 633	0xE 634	0xE 635	0xE 636	0xE 637	0xE 638	0xE 639	0xE 63A	0xE 63B	0xE 63C	0xE 63D	0xE 63E	0xE 63F	0xE 640	0xE 641	0xE6 42[7]	0xE6 42[6]
576i	0x5A	0x00	0x03	0x00	0x10	0x00	0x11	0x00	0x48	0x00	0x01	0x40	0x01	0x40	0x09	0xC0	0	0
576p	0x5A	0x00	0x03	0x00	0x10	0x00	0x11	0x00	0x48	0x00	0x01	0x40	0x01	0x40	0x09	0xC0	0	0
720p50	0xA0	0x00	0x6E	0x00	0x0A	0x00	0x37	0x00	0x5A	0x00	0x01	0x40	0x01	0x40	0x05	0x00	1	1
1080i50	0xF0	0x00	0x84	0x00	0x0B	0x00	0x25	0x00	0x87	0x00	0x01	0x00	0x01	0x40	0x09	0x00	1	1
1080p50	0xF0	0x00	0x84	0x00	0x0B	0x00	0x25	0x00	0x87	0x00	0x01	0x00	0x01	0x40	0x09	0x00	1	1
vga	0x50	0x00	0x04	0x00	0x18	0x00	0x0C	0x00	0x3C	0x00	0x02	0x80	0x00	0x80	0x08	0x40	0	0
480i	0x5A	0x00	0x04	0x00	0x0F	0x80	0x0F	0x00	0x3C	0x00	0x02	0x40	0x01	0x80	0x07	0x80	0	0
480p	0x5A	0x00	0x04	0x00	0x0F	0x80	0x0F	0x00	0x3C	0x00	0x02	0x40	0x01	0x80	0x07	0x80	0	0
720p60	0xA0	0x00	0x1B	0x80	0x0A	0x00	0x37	0x00	0x5A	0x00	0x01	0x40	0x01	0x40	0x05	0x00	1	1
1080i60	0xF0	0x00	0x16	0x00	0x0B	0x00	0x25	0x00	0x87	0x00	0x01	0x00	0x01	0x40	0x09	0x00	1	1
1080p60	0xF0	0x00	0x16	0x00	0x0B	0x00	0x25	0x00	0x87	0x00	0x01	0x00	0x01	0x40	0x09	0x00	1	1
1080p24	0xF0	0x00	0x9F	0x80	0x0B	0x00	0x25	0x00	0x87	0x00	0x01	0x00	0x01	0x40	0x09	0x00	1	1

The size of the output images of the VOM scaler can be smaller than that defined by the parameters of the output port. The starting position for the SVSP output video can be set using svsp_dp_video_h_start[10:0] and svsp_dp_video_v_start[10:0]. Figure 68 shows the relationship of the VOM scaler image and output video. In this case, the blank area around the output image is filled with color defined by the svsp_dp_margin_color[23:0] register in the YUV color space. This feature can be enabled using svsp_dp_output_blank.

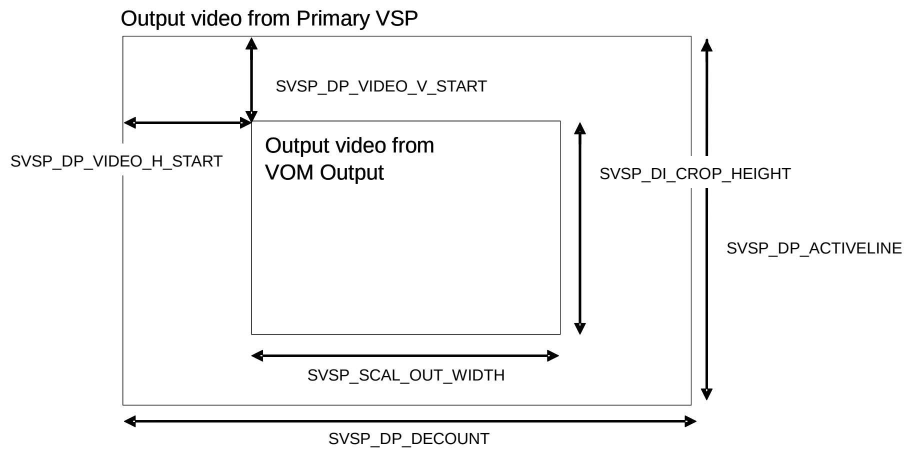


Figure 68: VOM Output Dimensions

svsp_dp_video_h_start[10:0], Secondary VSP Map, *Address 0xE62E[7:0]; Address 0xE62F[7:5]*
This signal is used to set the horizontal start position where the output video of scaler is placed.

Function

svsp_dp_video_h_start[10:0]	Description
0x000 (default)	Default
0xXXX	Horizontal start position of output port

svsp_dp_video_v_start[10:0], Secondary VSP Map, *Address 0xE630[7:0]; Address 0xE631[7:5]*
This signal is used to set the vertical start position where the output video of scaler is placed.

Function

svsp_dp_video_v_start[10:0]	Description
0x000 (default)	Default
0xXXX	Vertical start position of output port

svsp_dp_margin_color[23:0], Secondary VSP Map, *Address 0xE643[7:0]; Address 0xE644[7:0]; Address 0xE645[7:0]*
This signal is used to set the default color in output video in YUV colorspace.

Function

svsp_dp_margin_color[23:0]	Description
0x000000	Default
0xXXXXXX	Default color in YUV colorspace

svsp_dp_output_blank, Secondary VSP Map, *Address 0xE642[5]*
This bit is used to force the colour output of the Secondary VSP. If this register is set to 1, the output of the Secondary VSP is forced to the used defined color in svsp_dp_margin_color.

Function

svsp_dp_output_blank	Description
0 (default)	Not Output default Color
1	Output default Color

3.3.3.4. DDR Bypass Mode

In the case where the SVSP is being used to upscale or downscale between 1080p and 720p, external DDR2 memory is not required. Internal line buffers allow the user to convert between these two resolutions while maintaining the full external memory bandwidth for both the PVSP and OSD. The DDR bypass mode provided in the SVSP can be manually enabled/disabled using svsp_ddr_bypass. DDR2 bypass mode can be automatically configured using svsp_autocfg_input_vid[7:0] and svsp_autocfg_output_vid[7:0]. If the DDR bypass mode is to be set manually, svsp_man_set_ddr_bypass must be set to 1.

Note: This option is only available to the user when scaling between two resolutions which have the same frame rate.

svsp_man_set_ddr_bypass, Secondary VSP Map, *Address 0xE662[0]*
This bit is used to enable manually setting DDR bypass. If this bit is set to 1, SVSP will bypass DDR while svsp_ddr_bypass is 1, or not bypass DDR while svsp_ddr_bypass is 0.

Function

svsp_man_set_ddr_bypass	Description
0 (default)	Disable
1	Enable

svsp_ddr_bypass, Secondary VSP Map, *Address 0xE649[7]*
This bit is used to bypass external memory. This register's value will be used while svsp_man_set_ddr_bypass is 1.

Function

svsp_ddr_bypass	Description
0 (default)	Not bypass external memory
1	Bypass external memory

3.3.3.5. *Progressive to Interlaced Converter in SVSP*

The PtoI converter block in the SVSP is used to convert progressive video to interlaced video. It drops odd or even lines of the progressive video based on the output interlaced video field signal. Support is limited to 480p and 576p. The associated interlaced timing signals can be generated in the PtoI hardware block.

The PtoI converter in the SVSP cannot operate in standalone mode – it must be connected to the SVSP.
The PtoI hardware can be enabled using svsp_p2i_enable.

svsp_p2i_enable, Secondary VSP Map, *Address 0xE649[5]*
This bit is used to enable the PtoI in Secondary VSP.

Function

svsp_p2i_enable	**Description**
0 (default)	Disable
1	Enable

The input video to the PtoI block is defined using svsp_p2i_vid**[7:0]**. For more details on the values which must be programmed into this register, refer to Table 32.
svsp_p2i_vid[7:**0**], Secondary VSP Map, *Address 0xE64A[7:0]*
'This register is used to set the VIC of the PtoI in Secondary VSP.

Function

svsp_p2i_vid[7:0]	**Description**
0x00 (default)	Default

Table 32: VID for PtoI

Input Timing Format to P2I	**576p**	**480p**
svsp_s_p2i_vid	17	2

3.4. VSP REGISTER ACCESS PROTOCOLS

This section is used to describe the methods available to the user to update the VSP registers. The following types of register access protocols are available:

- Bootup protocol
- Reboot protocol
- Gentle reboot protocol
- VOM set protocol
- Free access protocol

These protocols are recommended to the user as best practice for updating VSP registers. The appropriate protocol should be used depending on the current status of the device. The seamless transfer of the VSP between standards can be achieved by using the bootup protocol, reboot protocol, gentle reboot protocol and VOM set protocols. If not changing VSP registers in real time, the free access protocol can be used.

3.4.1. *Bootup Protocol*

The bootup protocol is used to configure the PVSP or SVSP from a reset state. All registers can be accessed using this protocol.

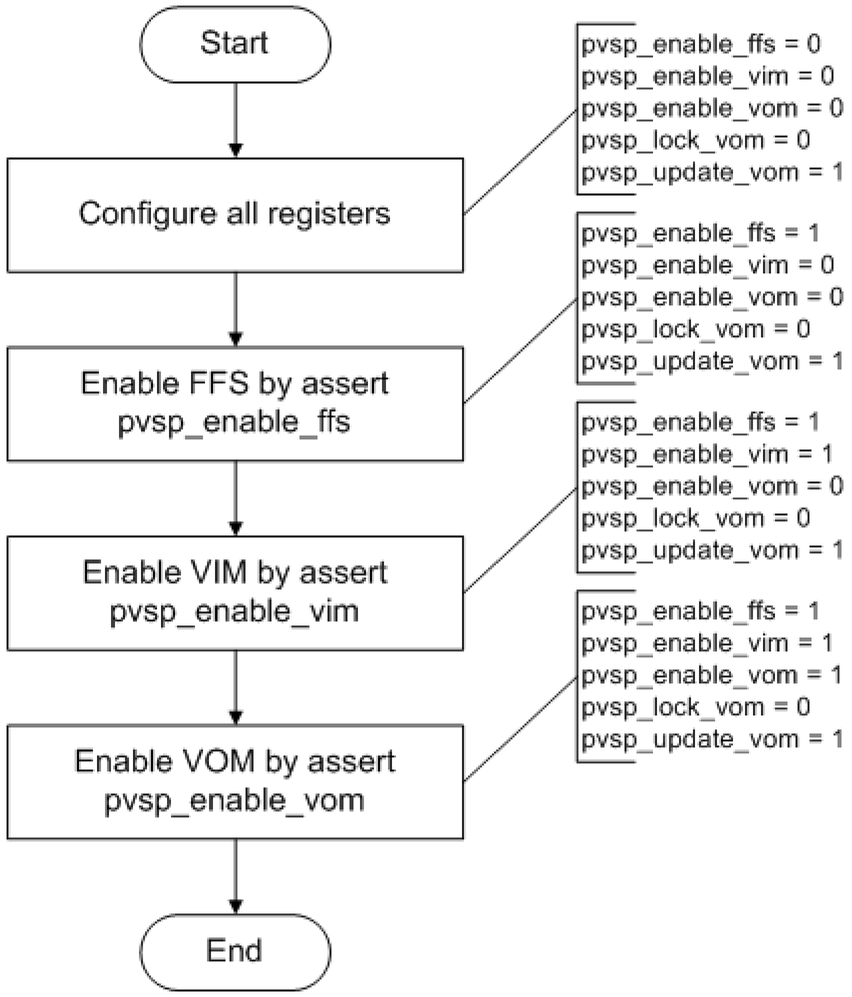


Figure 69: Bootup Protocol Flowchart

Figure 69 shows the process for the bootup protocol for the PVSP. This is exactly the same for the SVSP with the appropriate registers replaced.

3.4.2. *Reboot Protocol*

The reboot protocol is used to reset the PVSP and configure it again using different settings, especially different input timing or output timing. All registers can be accessed using this protocol. It should be noted that the output video will be interrupted using this protocol.

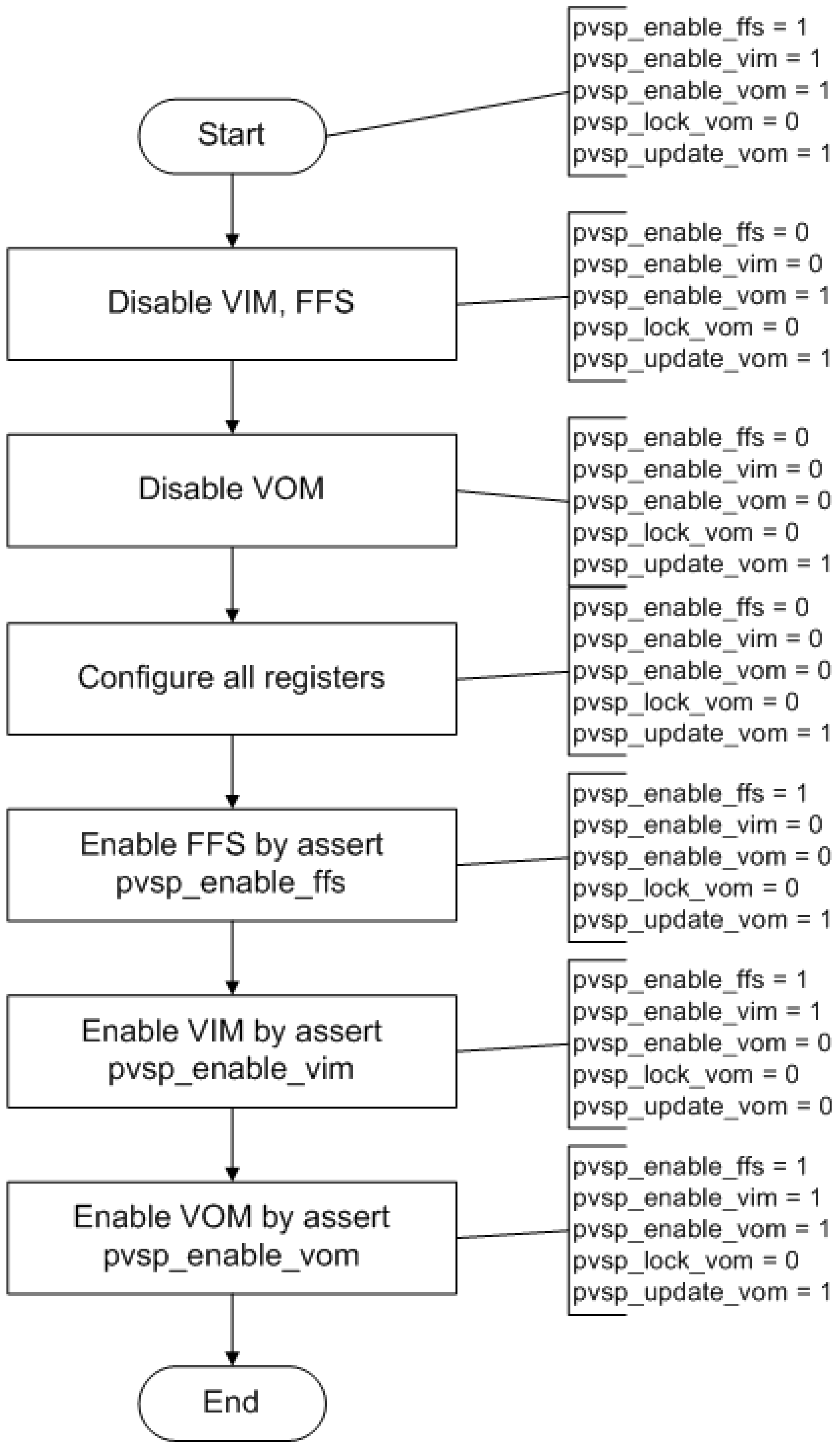


Figure 70: Reboot Protocol Flowchart

Figure 70 shows the process for the reboot protocol for the PVSP. This is exactly the same for the SVSP with the appropriate registers replaced.

3.4.3. *Gentle Reboot Protocol*

The gentle reboot is used to reboot the PVSP with different configuration settings but does not interrupt the output timing. The output video is frozen during this protocol. All registers except output video timing registers can be accessed.

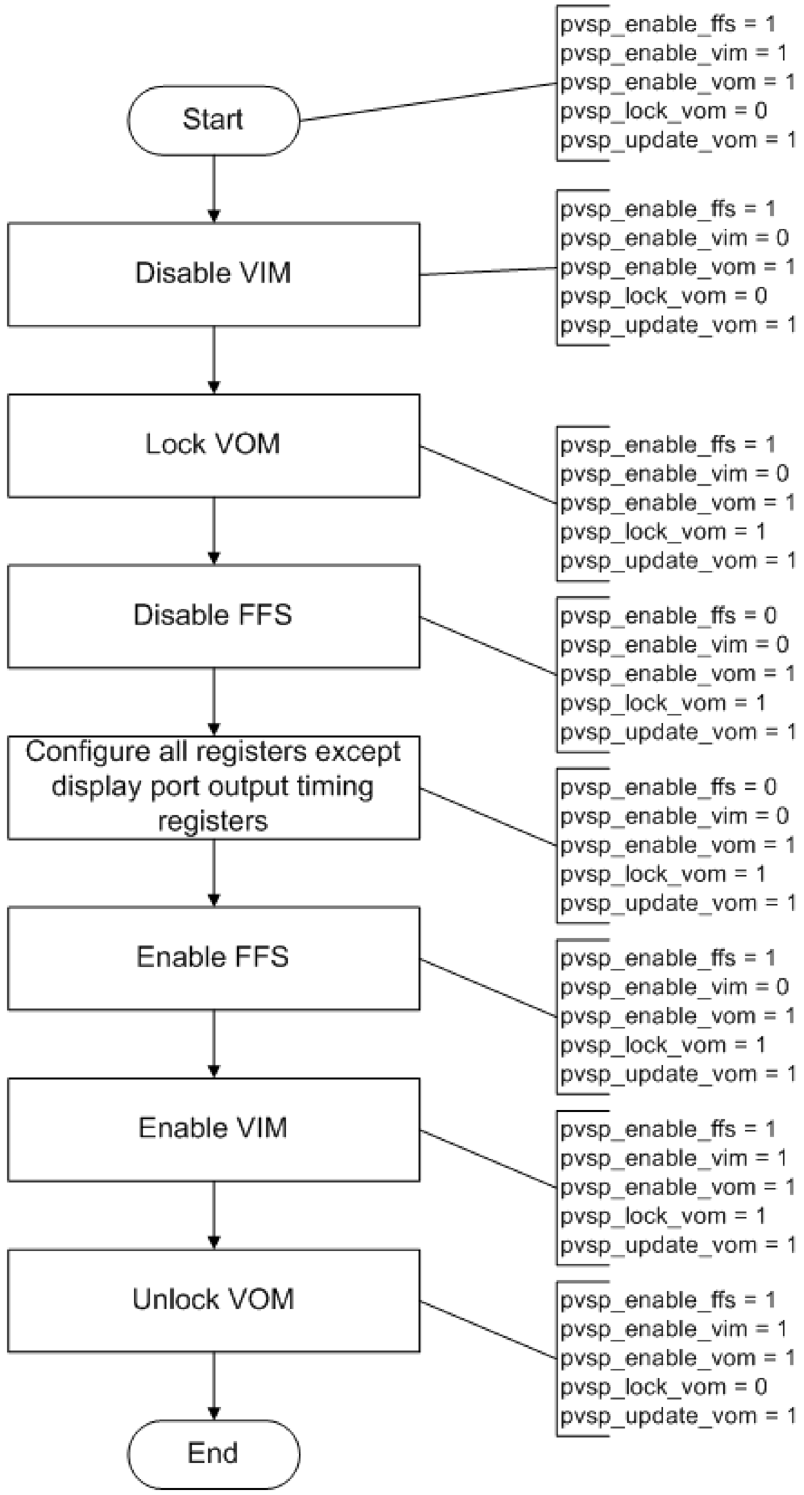


Figure 71: Gentle Reboot Protocol Flowchart

Figure 71 shows the process for the gentle reboot protocol for the PVSP. This is exactly the same for the SVSP with the appropriate registers replaced.

3.4.4. *VOM Set Protocol*

The VOM set protocol is used to configure the VOM. The registers in the VOM can be accessed without affecting the output video timing.

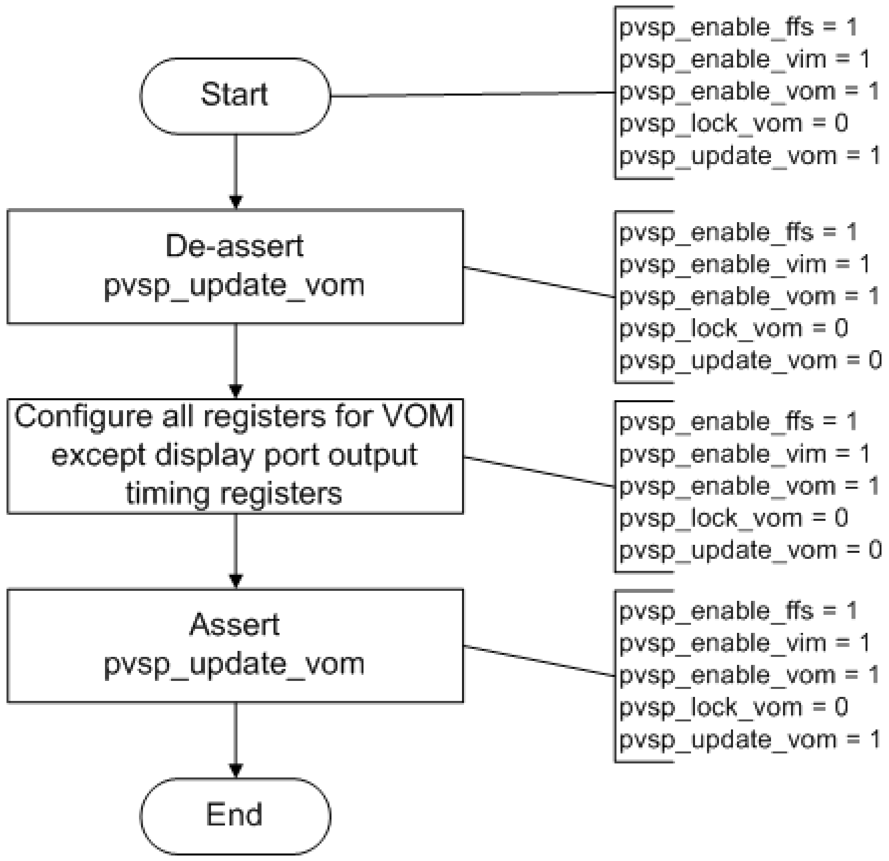


Figure 72: VOM Set Protocol Flowchart

Figure 72 shows the process for the VOM set protocol for the PVSP. This is exactly the same for the SVSP with the appropriate registers replaced.

3.4.5. *Free Access Protocol*

The free access protocol allows the user to configure all VSP registers regardless of the current configuration of the device. This can be seen in Figure 73.

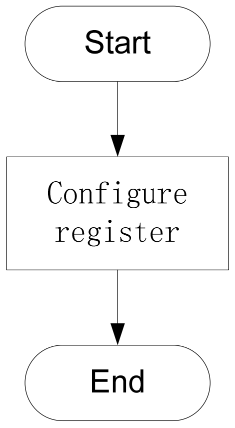


Figure 73: Free Access Protocol

3.5. HORIZONTAL PRE-SCALER

A Horizontal Pre-Scaler (HPS) has been implemented on the ADV8005 to extend the scaling functions of the ADV8005. The PVSP and SVSP are limited in the pixel clock frequencies and line lengths which they can handle. The HPS block has been designed for scaling between determined video formats as follows:

1. Down-conversion of video standards with pixel clocks greater than 162MHz and/or more than 2048 pixels/line. Typical use would be downscaling to video modes with pixel clocks of less than 162MHz, e.g. 4K@30 to 1080p@60.
2. Up-conversion of video standards with pixel clocks greater than 162MHz and/or more than 1920 pixels/line (but less than 3840) to video modes with pixel clocks greater than 162MHz, e.g. VESA 2048x1152 (162MHz) to 4K, VESA 1920x1440 (234MHz) to 4K.
3. Conversion of video standards with pixel clocks greater than 162MHz and more than 3840 pixels/lines. Typical use would be converting between different 4K timings, e.g. 4K@24 to 4K@24 SMPTE.
4. 3D to 2D conversion of some video modes.
5. Bypassing the downsampling block within the HPS, can be used just as an additional high-frequency filter to the one provided by the P/SVSP.

Video may be routed in to the HPS from any of the ADV8005 inputs using hps_inp_sel. The output from the HPS can be routed to either the PVSP or to the SVSP using **pvsp_inp_sel**[3:0] and **svsp_inp_sel**[3:0].

hps_inp_sel[3:0], IO Map, *Address 0x1A09[7:4]*
This signal is used to select the video source for the Horizontal pre-scaler (HPS) block

Function

hps_inp_sel[3:0]	**Description**
0x00 (default)	From Primary Input Channel
0x01	From Secondary Input Channel
0x02	From RX Input
0x03	From Internal OSD Blend 1

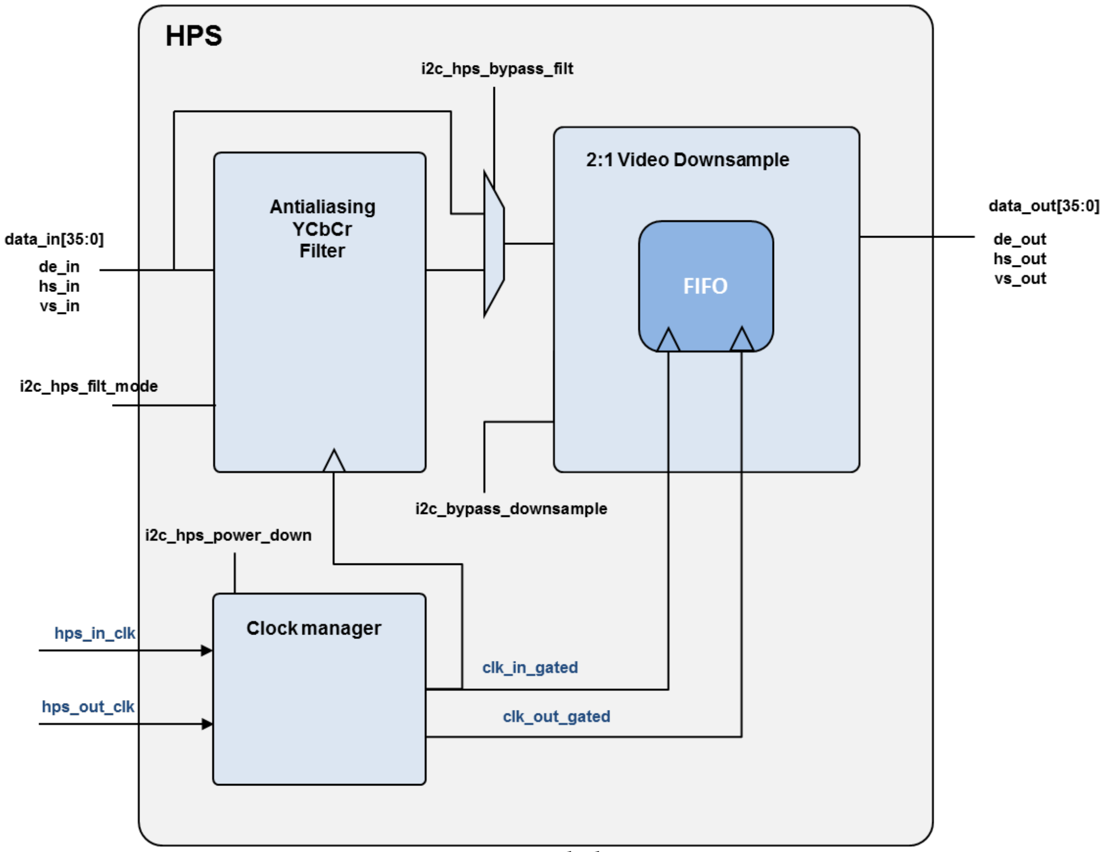


Figure 74 HPS Block Diagram

The HPS block provides two separate low pass filters which can be selected using hps_filt_bypass. The HPS filter can be powered down using hps_power_down.

hps_power_down, IO Map, *Address 0x1A85[7]*
Powers down the horizontal pre-scaler block (HPS). Powered down by default to save power

Function

hps_power_down	**Description**
0	HPS is active
1 (default)	HPS Block is powered down

hps_filt_bypass, IO Map, *Address 0x1A85[4]*
This bit bypasses filtering done before downsampling. Aliasing may occur if this filtering is not done

Function

hps_filt_bypass	**Description**
0 (default)	Do not bypass
1	Bypass

hps_bypass_downsample, IO Map, *Address 0x1A85[3]*
This bit bypasses data downsampling. Use this control to just filter but not downsample video data

Function

hps_bypass_downsample	**Description**
0 (default)	Do not bypass
1	Bypass

hps_phase_sel_downsample, IO Map, *Address 0x1A85[2]*
This bit selects whether the downsampling should start by keeping or dropping the first pixel when in a 2 - 1 downsampling. '

Function

hps_phase_sel_downsample	**Description**
0	Start by keeping the first Pixel
1 (default)	Start by dropping the first Pixel

hps_filt_mode[1:0], IO Map, *Address 0x1A85[1:0]*
The filter has 2 operating modes. Mode 0 has higher bandpass but less aliasing rejection.

Function

hps_filt_mode[1:0]	**Description**
0	Filter mode 0
1 (default)	Filer mode 1
2	Unused
3	Unused

3.5.1. *HPS Downscaling*

The video downsampling block provides a 2:1 reduction on the horizontal resolution of the video stream, required to route high-resolution, high-speed data to the PVSP/SVSP. If only the filter of the HPS is to be used, this downsampling block can be disabled with hps_bypass_downsample.

It is possible to select whether to keep/drop the first pixel of the line when downsampling. This is done through hps_phase_sel_downsample. Below image illustrates how this control affects a one pixel wide, black-white column, video pattern, sent through the HPS. Note that the filtering has been disabled in order to preserve the one pixel wide pure black/white pattern. Changing the phase of the downsampling results in a completely white or black pattern at the output of the HPS block.

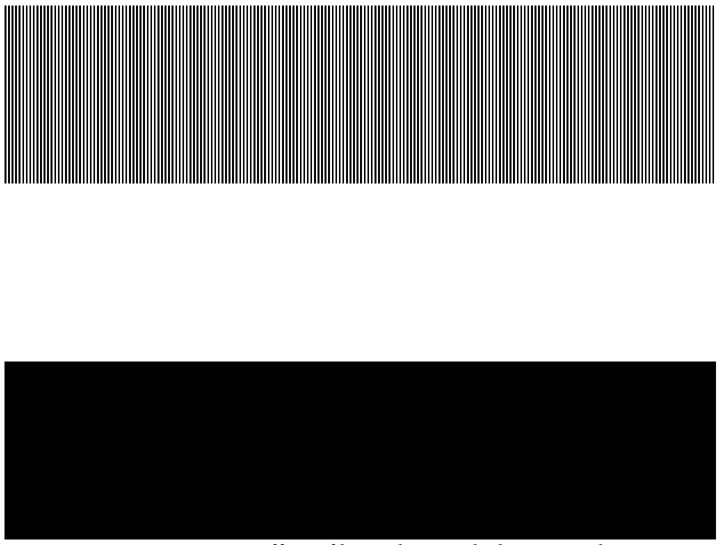


Figure 75 HPS effect of hps_phase_sel_downsample

In order to perform downscaling of video standards with pixel clocks greater than 162MHz, it will be necessary to go through the HPS before routing the video to either the PVSP or SVSP.

For video standards in which after the HPS block the horizontal resolution is bigger than 1920 pixels/line, it is mandatory to go through the SVSP.

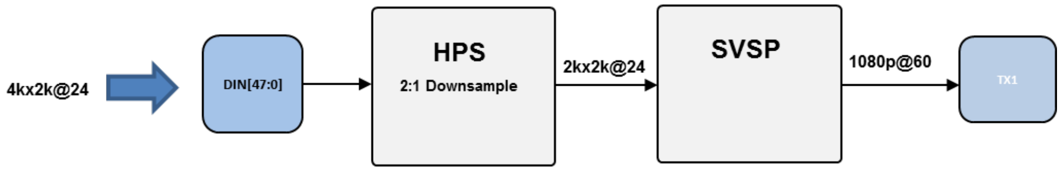


Figure 76 Using the HPS to Downscale the 4K2K video

If the horizontal resolution of the video stream after the HPS is 1920 or less pixels, it is also possible to route the video through the PVSP.

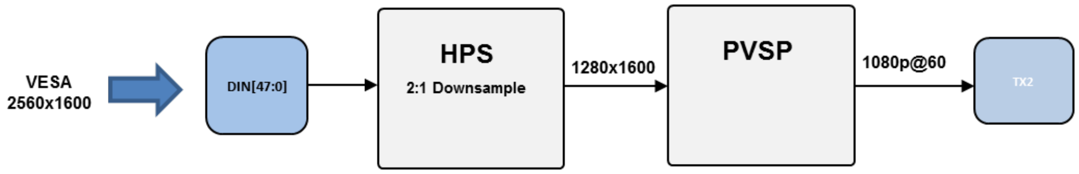


Figure 77 Using the HPS to downscale to downscale to less than 1920 horizontal pixels

3.5.2. *HPS Upscaling*

When upscaling video streams with pixel clocks greater than 162MHz or with horizontal resolutions larger than 1920 pixels (but smaller than 3840) to video modes with pixel clocks greater than 162MHz, the input video has to be routed to the HPS followed by the PVSP. E.g. VESA 2048x1152 (162MHz) to 4K, VESA 2560x1600 (Reduced Blanking, 268MHz) to 4K.

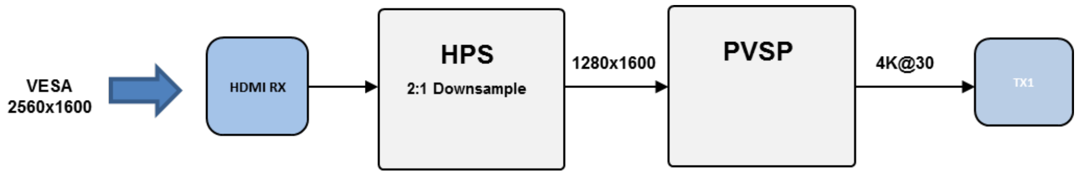


Figure 78 HPS Upscaling

When processing video standards with pixel clocks greater than 162MHz and more than 3840 pixels/lines, a combination of HPS, SVSP and

VIC	Format	Int/Prog	Field Rate [Hz]	Pixel Freq [MHz]	H Freq [kHz]	V Freq [Hz]	H total [dots]	H active [dots]	H blank [dots]	H Front Porch [dots]	Hsync [dots]	H Back Porch [dots]	V total [lines]	V active [lines]	V blank [lines]	V Front Porch [lines]	Vsync [lines]	V Back Porch [lines]
16	1920x1080p	Prog	59.94/60	297	134.866	59.94	2200	1920	280	88	44	148	2250	2205	45	4	5	36
31	1920x1080p	Prog	50	297	112.5	50	2640	1920	720	528	44	148	2250	2205	45	4	5	36
35	2880x480p	Prog	59.94/60	216	62.938	59.94	3432	2880	552	64	248	240	1050	1005	45	9	6	30
36	2880x480p	Prog	59.94/60	216	62.938	59.94	3432	2880	552	64	248	240	1050	1005	45	9	6	30
37	2880x576p	Prog	50	216	62.5	50	3456	2880	576	48	256	272	1250	1201	49	5	5	39
38	2880x576p	Prog	50	216	62.5	50	3456	2880	576	48	256	272	1250	1201	49	5	5	39
40	1920x1080i	Int	100	297	112.5	50	2640	1920	720	528	44	148	2250	2228	22	2	5	15
41	1280x720p	Prog	100	297	150	100	1980	1280	700	440	40	220	1500	1470	30	5	5	20
46	1920x1080i	Int	119.88/120	296.704	134.864	59.94	2200	1920	280	88	44	148	2250	1226	22	2	5	15
47	1280x720p	Prog	119.88/120	296.704	179.818	119.88	1650	1280	370	110	40	220	1500	1470	30	5	5	20
52	720x576p	Prog	200	216	250	200	864	720	144	12	64	68	1250	1201	49	5	5	39
53	720x576p	Prog	200	216	250	200	864	720	144	12	64	68	1250	1201	49	5	5	39
54	720(1440)x576i	Int	200	216	125	100	1728	1440	288	24	126	138	1250	1226	24	2	3	19
55	720(1440)x576i	Int	200	216	125	100	1728	1440	288	24	126	138	1250	1226	24	2	3	19
56	720x480p	Prog	239.76/240	216	251.748	239.76	858	720	138	16	62	60	1050	1005	45	9	6	30
57	720x480p	Prog	239.76/240	216	251.748	239.76	858	720	138	16	62	60	1050	1005	45	9	6	30
58	720(1440)x480i	Int	239.76/240	216	125.874	119.88	1716	1440	276	38	124	114	1050	1028	22	4	3	15
59	720(1440)x480i	Int	239.76/240	216	125.874	119.88	1716	1440	276	38	124	114	1050	1028	22	4	3	15

PVSP is required. Typical use would be converting between different 4K timings, e.g. 4K@24 to 4K@24 SMPTE. It is not possible however to do frame rate conversions between 4K modes.

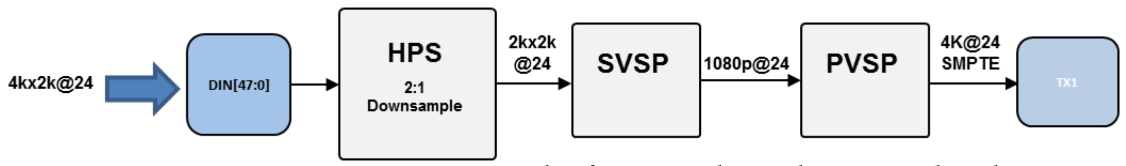


Figure 79 HPS scaling for inputs with more than 3840 pixels per line.

3.5.3. *Using the HPS for converting between 3D to 2D Video formats*

Other usage scenario of the HPS is the conversion from certain 3D modes to its 2D equivalent. The 3D modes which require making use of the HPS are the ones with pixels clocks greater than 162MHz and/or horizontal resolutions larger than 1920 pixels/line.

The 3D->2D conversions for Frame Packing, Side-by-Side Full and Side-by-Side Half packing modes which need to make use of the HPS are shown on the following tables.

The following formats are not supported

VIC	Format	Int/Prog	Field Rate [Hz]	Pixel Freq [MHz]	H Freq [kHz]	V Freq [Hz]	H total [dots]	H active [dots]	H blank [dots]	H Front Porch [dots]	Hsync [dots]	H Back Porch [dots]	V total [lines]	V active [lines]	V blank [lines]	V Front Porch [lines]	Vsync [lines]	V Back Porch [lines]
63	1920x1080p	Prog	119,88/120	594	270	120	2200	1920	280	88	44	148	2250	2205	45	4	5	36
64	1920x1080p	Prog	100	594	225	100	2640	1920	720	528	44	148	2250	2205	45	4	5	36

3.5.4. *3D Side by Side Full*

The following 3D standards need to go through the HPS before being converted to a 2D mode.

VIC	Format	Int/Prog	Field Rate [Hz]	Pixel Freq [MHz]	H Freq [kHz]	V Freq [Hz]	H total [dots]	H active [dots]	H blank [dots]	H Front Porch [dots]	Hsync [dots]	H Back Porch [dots]	V total [lines]	V active [lines]	V blank [lines]	V Front Porch [lines]	Vsync [lines]	V Back Porch [lines]
16	1920x1080p	Prog	59,94/60	297	67.433	59.94	4400	3840	560	176	88	296	1125	1080	45	4	5	36
31	1920x1080p	Prog	50	297	56.25	50	5280	3840	1440	1056	88	296	1125	1080	45	4	5	36
35	2880x480p	Prog	59,94/60	216	31.469	59.94	6864	5760	1104	128	496	480	525	480	45	9	6	30
36	2880x480p	Prog	59,94/60	216	31.469	59.94	6864	5760	1104	128	496	480	525	480	45	9	6	30
37	2880x576p	Prog	50	216	31.25	50	6912	5760	1152	96	512	544	625	576	49	5	5	39
38	2880x576p	Prog	50	216	31.25	50	6912	5760	1152	96	512	544	625	576	49	5	5	39
40	1920x1080i	Int	100	297	56.25	100	5280	3840	1440	1056	88	296	1125	1080	22/23	2/2,5	5	15/15,5
41	1280x720p	Prog	100	297	75	100	3960	2560	1400	880	80	440	750	720	30	5	5	20
46	1920x1080i	Int	119,88/120	296.704	67.432	119.88	4400	3840	560	176	88	296	1125	1080	22/23	2/2,5	5	15/15,5
47	1280x720p	Prog	119,88/120	296.704	89.909	119.88	3300	2560	740	220	80	440	750	720	30	5	5	20
52	720x576p	Prog	200	216	125	200	1728	1440	288	24	128	136	625	576	49	5	5	39
53	720x576p	Prog	200	216	125	200	1728	1440	288	24	128	136	625	576	49	5	5	39
54	720(1440)x576i	Int	200	216	62.5	200	3456	2880	576	48	252	276	625	576	24/25	2/2,5	3	19/19,5
55	720(1440)x576i	Int	200	216	62.5	200	3456	2880	576	48	252	276	625	576	24/25	2/2,5	3	19/19,5
56	720x480p	Prog	239,76/240	216	125.874	239.76	1716	1440	276	32	124	120	525	480	45	9	6	30
57	720x480p	Prog	239,76/240	216	125.874	239.76	1716	1440	276	32	124	120	525	480	45	9	6	30
58	720(1440)x480i	Int	239,76/240	216	62.937	239.76	3432	2880	552	76	248	228	525	480	22/23	4/4,5	3	15/15,5
59	720(1440)x480i	Int	239,76/240	216	62.937	239.76	3432	2880	552	76	248	228	525	480	22/23	4/4,5	3	15/15,5

The following standards are NOT supported.

VIC	Format	Int/Prog	Field Rate [Hz]	Pixel Freq [MHz]	H Freq [kHz]	V Freq [Hz]	H total [dots]	H active [dots]	H blank [dots]	H Front Porch [dots]	Hsync [dots]	H Back Porch [dots]	V total [lines]	V active [lines]	V blank [lines]	V Front Porch [lines]	Vsync [lines]	V Back Porch [lines]
63	1920x1080p	Prog	119,88/120	594	270	120	2200	1920	280	88	44	148	2250	2205	45	4	5	36
64	1920x1080p	Prog	100	594	225	100	2640	1920	720	528	44	148	2250	2205	45	4	5	36

3.5.5. *3D Side by Side Full*

The following 3D standards need to go through the HPS before being converted to a 2D mode.

VIC	Format	Int/Prog	Field Rate [Hz]	Pixel Freq [MHz]	H Freq [kHz]	V Freq [Hz]	H total [dots]	H active [dots]	H blank [dots]	H Front Porch [dots]	Hsync [dots]	H Back Porch [dots]	V total [lines]	V active [lines]	V blank [lines]	V Front Porch [lines]	Vsync [lines]	V Back Porch [lines]
63	1920x1080p	Prog	119,88/120	297	135	120	2200	1920	280	88	44	148	1125	1080	45	4	5	36
64	1920x1080p	Prog	100	297	112.5	100	2640	1920	720	528	44	148	1125	1080	45	4	5	36

3.6. EXTERNAL SYNC MODE

Using the ADV8005 external sync mode, it is possible to resynchronise multiple ADV8005 output video streams to an external sync input. The outputs from multiple ADV8005 devices will be locked to +/- 3 Xtal clock cycles, where the Xtal clock will be 27 MHz.

When the ADV8005 is in external sync mode, the output video timing will be locked to an externally provided master sync signal (MAS_VS). This master signal must be provided to the MAS_VS ball. The polarity of this sync signal is assumed to be active high and will default to this operation. mas_vs_ie, mas_hs_ie**,** and mas_clk_ie are used to enable the respective external sync pins.

Assumptions for operating in this mode:

- The external sync provided to the ADV8005 will be a CEA-861 or VESA compliant VSync. Non standard timing will NOT be supported, that is, extra or fewer pixels, lines or frames than specified in the standard. Note that the VS and HS are assumed to be active high.
- The sync signals supported will be VS and HS. Note that HS is optional and only required if interlaced output is required. In this case the HS position with respect to the VS will be used to determine the output field required. If only progressive outputs are required then the HS may be omitted and VS alone will suffice to lock the output.
- The external timing provided should match the output video standard programmed. For example if 1080i60Hz is to be output from the ADV8005 PVSP and locked to external timing then a 60 Hz Vsync signal should be provided on the MAS_VS pin and a 33.750 kHz HSync should be provided on the MAS_HS pin. In this case the pvsp_autocfg_output_vid[7:0] should be set to 5.

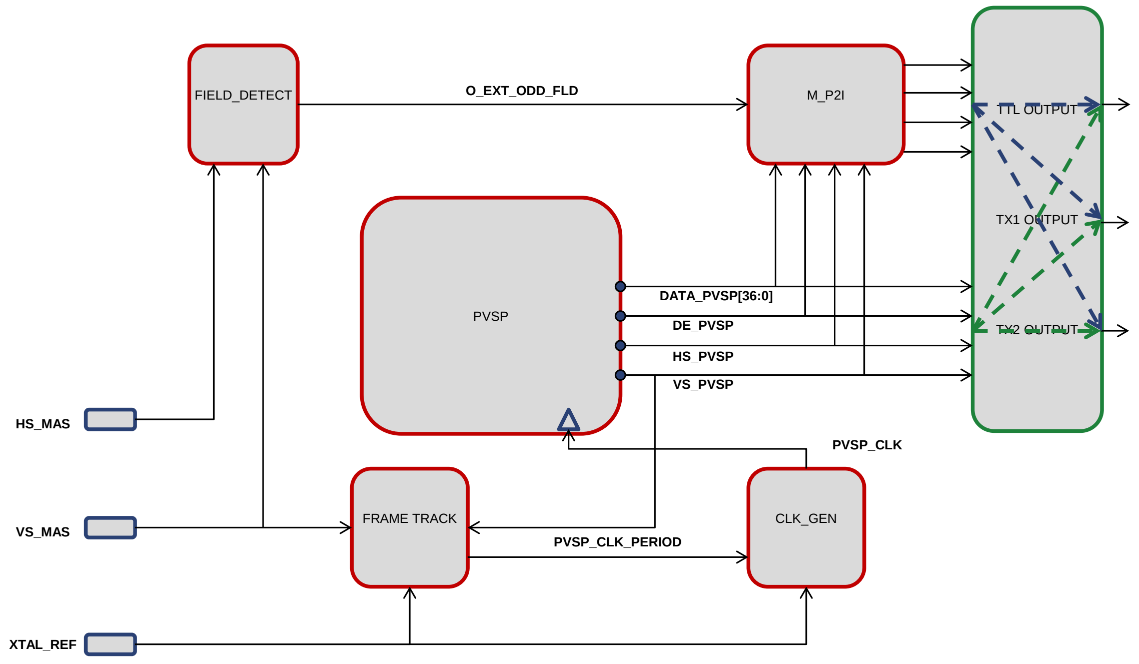


Figure 80: ADV8005 External Sync Mode Block Diagram

3.6.1. *Functional Description*

The ADV8005 compares the phase difference between the MAS_VS and the internally generated VS out as shown in Figure 80. The phase difference is measured using a fixed crystal clock running at 27MHz. The phase difference between the input and output VS signals constitutes an error which must be reduced to zero in order for the outputs to be locked together. This is achieved by varying the output clock in order to change the period of the output VS. Once the error is reduced to 0 the output video and timing will be locked to the external master. This locking process can take from 0 – 5 seconds. As the external master sync will always be present and stable this will constitute a start-up condition and once locked will remain locked. If the input video source is changed at a future time this will not disrupt the relationship between the external master sync and output timing.

The video output will be locked to within +/-2 Xtal clock cycles of the externally provided master sync. In the worst case scenario where 4k2k is being output on a 297MHz clock the potential pixel difference is +/-22pixels. For 1080p outputs this variation drops to +/-11 pixels. This difference between outputs can be eliminated using a small FIFO. Note that this resynchronisation block will also eliminate any cable delay differences between different ADV8005 systems.

It is important to note that if the output timing is being locked to the external MAS_VS reference it cannot be locked to the input timing at the same time. This means that if there are frequency differences between the external timing and input timing provided to the ADV8005, input frames of video will be either dropped or repeated to account for these differences and keep the output timing locked to the external master reference (MAS_VS).

It is also possible to add a track_offset via pvsp_track_offset[20:0] to the phase error that is eliminated. This allows the ADV8005 to either advance or delay the output timing versus the reference timing, which is externally provided on the MAS_VS ball in this case. If there is not the possibility of providing an advanced external sync versus the desired output timing then an advance can be programmed to individual ADV8005 parts in order to achieve the same effect.

pvsp_track_offset[**20:0**], IO Map, *Address 0x1A94[4:0]; Address 0x1A95[7:0]; Address 0x1A96[7:0]*
This signal is used to program the delay on the output timing of VSyncs from the Primary VSP.

Function

pvsp_track_offset[20:0]	**Description**
0 «	input and output VSync coincident
1	1 Xtal clk between input and output VSync

MAS sync mode using frame track can be enabled using pvsp_frtrk_mas_mode_en.

pvsp_frtrk_mas_mode_en, IO Map, *Address 0x1B97[0]*
This bit enables the use of external master hs and vs for frame tracking

Function

pvsp_frtrk_mas_mode_en	**Description**
0 (default)	Frame track input
1	Frame track external master hs/vs

External Sync Mode Summary

External sync locking mode is only needed for applications where the ADV8005 is required to lock its output timing to an externally provided source. Applications where the ADV8005/ADV8003 output is required to be locked to the input timing do not require this functionality. These applications (e.g. video wall) can use 'phase locked frame track mode' to achieve this functionality. The output will be locked within +/- 2xtal clocks after an initial lock time of 0 – 5 seconds.

svsp_frtrk_mas_mode_en, IO Map, *Address 0x1B99[0]*
This bit enables the use of external master hs and vs for frame tracking

Function

svsp_frtrk_mas_mode_en	**Description**
0 (default)	Frame track input
1	Frame track external master hs/vs

mp2i_frtrk_mas_fld, IO Map, *Address 0x1B97[1]*
This bit select whether the input field information from the mas_vs and mas_hs is tracked by the mp2i block or not. The control signal pvsp_frtrk_mas_mode_en, must also be enabled for this bit to take effect.

Function

mp2i_frtrk_mas_fld	**Description**
0 (default)	Disable tracking of input master field
1	Enable tracking of input master field

sp2i_frtrk_mas_fld, IO Map, *Address 0x1B99[1]*
This bit selects whether the input field information from the mas_vs and mas_hs is tracked by the mp2i block or not. The control signal svsp_frtrk_mas_mode_en, must also be enabled for this bit to take effect.

Function

sp2i_frtrk_mas_fld	**Description**
0 (default)	Disable tracking of input master field
1	Enable tracking of input master field

s_p2i_invert_vsp2d_flag, Secondary VSP Map, *Address 0xE65E[7]*
This bit is used to invert the field information being sent to the secondary P2I block.

The following I2C controls are used for external sync mode 3 only.

pvsp_mas_resync_en, Primary VSP Map, *Address 0xE8A1[7]*
This bit enables direct timing generation reset via external sync for the PVSP. This is for modes 2 and 3 only.

pvsp_freq_sel, IO Map, *Address 0x1A44[7]*
This bit is used to manually configure the vertical frequency for the Primary VSP.

Function

pvsp_freq_sel	**Description**
0 (default)	59.94Hz or 23.9Hz
1	60Hz or 24Hz

pvsp_track_offset[20:0], IO Map, *Address 0x1A94[4:0]; Address 0x1A95[7:0]; Address 0x1A96[7:0]*
This signal is used to program the delay on the output timing of vsyncs from the Primary VSP.

Function

pvsp_track_offset[20:0]	**Description**
0 (default)	input and output vsyncs are coincident
1	1 xltal clk between input and output vsync

svsp_track_offset[**20:0**], IO Map, *Address 0x1A97[4:0]; Address 0x1A98[7:0]; Address 0x1A99[7:0]*
This signal is used to program the delay on the output timing of vsyncs from the Secondary VSP.

Function

svsp_track_offset[20:0]	**Description**
0 (default)	input and output vsync coincident
1	1 xltal clk between input nad output vsync

3.7. PROGRESSIVE TO INTERLACED CONVERSION

ADV8005 has two progressive to interlaced converters (P2I).

The primary P2I converter is an independent block to which the PVSP, OSD and inputs can be connected. The primary P2I converter can convert from any progressive format to its interlaced equivalent. The input to the primary P2I converter is selected by p2i_inp_sel[3:0].

The secondary P2I converter is connected directly to the SVSP. The secondary P2I converter cannot convert from 1080p to 1080i but can handle all other progressive to interlaced conversions.

p2i_inp_sel[**3:0**], IO Map, *Address 0x1A06[7:4]*
This signal is used to select the video source for the Progressive to Interlaced converter.

Function

p2i_inp_sel[3:0]	**Description**
0x00 «	From Primary VSP
0x01	From Internal OSD Blend 1
0x02	From EXOSD TTL Input
0x03	From RX Input
0x04	From Video TTL Input

4. ON SCREEN DISPLAY

4.1. INTRODUCTION

The On Screen Display (OSD) core in the ADV8005 allows the user to overlay a bitmap-based OSD onto one of the input video streams. The OSD blend is capable of being performed at data rates up to 3 GHz. The OSD can be designed using the *ADI Blimp* software tool. This code generating tool may be used to design, simulate and compile the OSD which will be used in the end system application.

The *Blimp OSD* software tool covers the full design flow involved in delivering a complex bitmap-based OSD – from initial graphics design through to outputting the files required for integration into the system application. *Blimp OSD* abstracts the user from the OSD hardware so a detailed description of the OSD hardware is not provided. For more information on the OSD design flow and *Blimp OSD* software, refer to the *Blimp OSD* software tool user manual.

4.1.1. *Features*

- Full design-flow covered by *Blimp OSD* software, user does not need to worry about the OSD hardware
- OSD maximum resolution of 4096 x 3840
- Pixel-by-pixel alpha blending
- Dual video paths through the OSD blend block to support dual zone OSD display
- Eight hardware timers which provide added functionality for OSD or system tasks
- Programmable blending effect of OSD and background video
- Programmable priority of regions
- Uniform programmable transparent color in the OSD
- OSD video input and output format: 36-bit RGB
- Support for main 3D video format timings
- High-performance scaling quality with 8-bit horizontal and vertical video scaler
- Arbitrary resolution conversion
- Support vertical/horizontal scaling order change
- Support progressive to interlaced conversion
- Anti-alias mode for downscaling
- OSD data range control

4.1.2. *OSD System Application Diagram*

Figure 81 provides a typical application diagram for using the bitmap OSD. The external MCU uses the ADV8005 SPI slave (serial port 1) interface to configure the registers in the bitmap OSD module. The ADV8005 uses its SPI master (serial port 2) interface to obtain the OSD data (fonts, icons, and images) from an external flash memory and store it into the DDR2 memory. The OSD can then be blended onto either of the video paths through the OSD core.

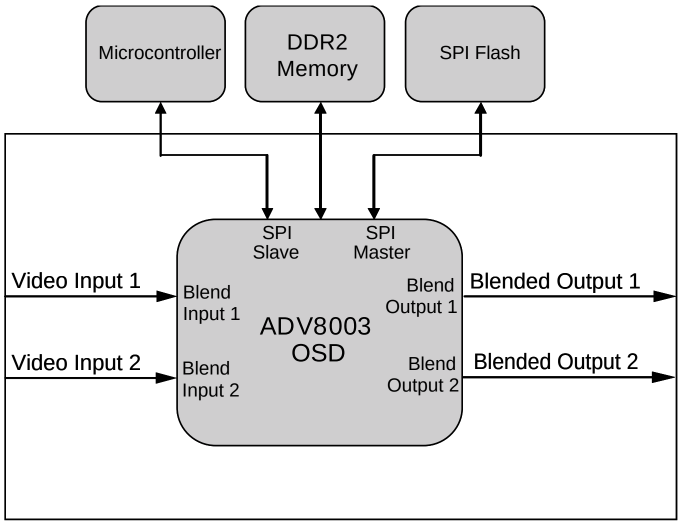


Figure 81: Typical Application Diagram

4.1.3. *Typical OSD Component Sizes*

An indication of typical OSD component sizes in provided in Table 33. This can be used to gain an approximation of the size of an OSD.

Table 33: Output Port Configuration Settings for Example Output Formats

Component	Color Mode (per pixel)	DDR2 Size (bytes, W – width, H – height)
Label	8 bits	W*H*2
Image	8 bits/16 bits/32 bits	W*H*2/4/8
Listbox	32 bits	W*H*8
Textbox	16 bits	W*H*4
Iptextbox	16 bits	W*H*4
Histogram	32 bits	W*H*8
Menubar	32 bits	W*H*8
Keyboard	32 bits	W*H*8
Progressbar	32 bits	W*H*8
Timer	0	0

4.2. ARCHITECTURE OVERVIEW

4.2.1. *Introduction*

As outlined in Section 4.1.2, the OSD core in the ADV8005 is controlled mainly via a SPI slave interface and loads images and OSD data into the part via a SPI master interface. Consequently, a number of the configuration registers for the OSD core are SPI registers and the code required to control these registers is automatically generated by the *Blimp OSD* software tool – abstracting the user away from having to understand them. For this reason, many of the SPI registers are not described in this section. For more information, refer to the *Blimp OSD* software tool user manual.

4.2.2. *Top Level Diagram*

Figure 82 provides a diagram of the ADV8005 OSD top level.

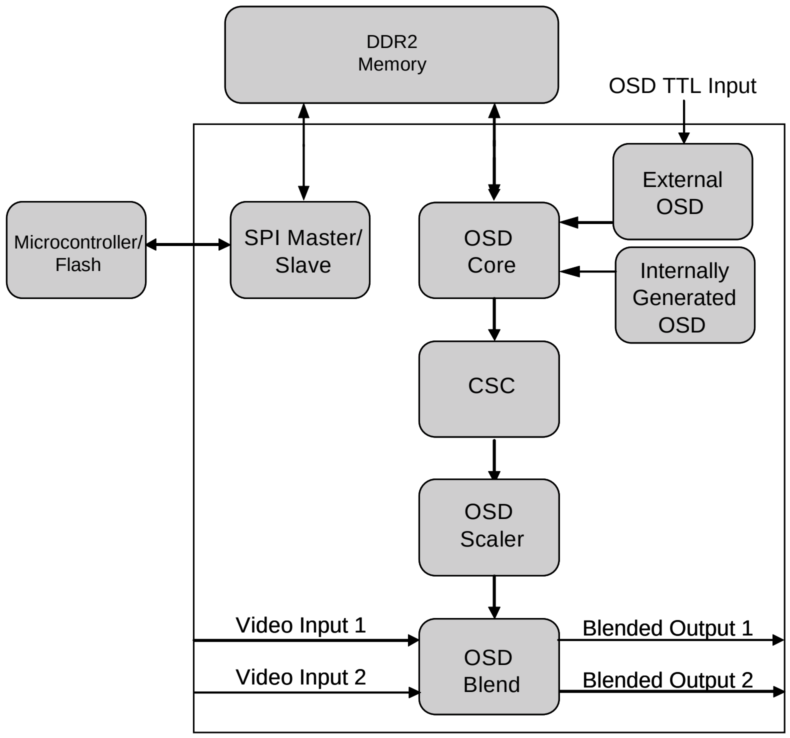


Figure 82: Bitmap OSD Top Level Diagram

OSD Blend: Used to overlay the OSD data with the input video.
OSD Scaler: Used to scale the OSD to the target resolution.
CSC: Used to convert the OSD core data color to the same color space as that of the input video.
OSD Core: Used to generate internal OSD data. Reads data from DDR2 memory and outputs data to FIFO.
SPI Master and SPI Slave: SPI master used to copy flash data into DDR2 memory. SPI slave used as the only means to control OSD configuration registers and memories.

4.2.3. *OSD Blending*

The OSD core in the ADV8005 has two video inputs and two video outputs and is capable of blending at data rates of up to 3 GHz.

The two video inputs allow two different video streams to be connected to the OSD core, for example, video TTL input channel and SVSP output. The inputs connected to the OSD core can be selected using osd_blend_inp_sel[3:0] and osd_blend_inp_2_sel[3:0]. Refer to Figure 25 for further details. The video stream connected to OSD input 1 is output to the OSD blend 1 output and the video stream connected to OSD input 2 is output to the OSD blend 2 output. It is only possible to blend video on OSD blend 1 output or on OSD blend 2 output. It is not possible to OSD blend on both at the same time.

The OSD can be blended onto either one of the two video streams connected to the OSD core, that is, there is only one source of OSD data and it must be configured to match one video stream's format and timing at a time. The OSD can be switched between the two video streams without causing any disturbance on either output video stream. The OSD core outputs can be connected to one or more of the output blocks, for example, HDMI TX1, HDMI TX2, SD encoder and HD encoder.

The OSD is blended with the selected video stream using alpha blending. This means that each pixel of OSD has its own blending parameter which is used to blend this pixel with its corresponding background video. If the OSD data is transparent, the background video will be passed through and unadjusted.

As shown in Figure 82, the OSD data needs to be scaled to the target resolution before getting into the blending block (refer to Section 4.2.7). The clock and DE of the selected video stream are used to read the scaler output data. Delay is added to DATA, DE, HS and VS for matching the delay of the OSD processing, so the OSD scaler can ensure the correct synchronization of OSD data and input video data.

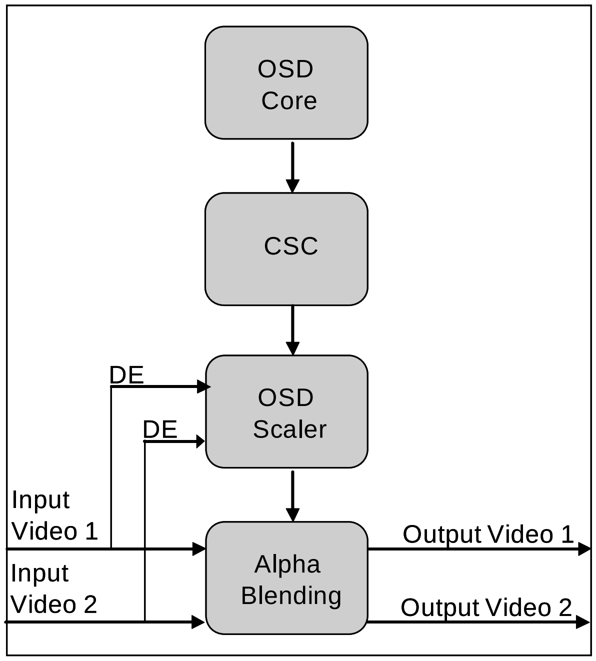


Figure 83: OSD Scaler and Blending Top Level Diagram

4.2.4. *External Alpha Blending*

The ADV8005 features an external alpha blend input which is shared with the input pixel port. The external alpha blend can only be used in conjunction with the EXOSD TTL input. This allows the option to specify an external alpha blend value for the EXOSD TTL input channel. The options for routing the external alpha blend value are outlined in Table 90. The external alpha blend function is enabled via SPI.

4.2.5. *OSD Core*

The OSD core generates the internal data for the OSD display. It accesses the DDR2 memory (through a DMA controller) to load the required resources.

reg_osd_enable is used to enable the OSD core on the ADV8005.

reg_osd_enable, OSD, *Address 0xEE00[0]*
The enable bit of OSD core.

Function

reg_osd_enable	**Description**
0 «	Disables OSD core
1	Enables OSD core

osd_reset is used to reset the whole OSD core.
osd_reset, IO, *Address 0x1AFD[1]*
This register bit resets the OSD core.

Function

osd_reset	**Description**
0 «	Default
1	Resets OSD core

4.2.5.1. OSD Core Region Definition

A region defines an area on the plane, as shown in Figure 84. The regions are derived from the OSD components defined in the *Blimp OSD* software and, therefore, contain the different elements of the OSD, for example, the text, images, icons, and so on. In other words, the regions define how the OSD pixels to be displayed are stored in DDR2 memory. The equivalence between OSD components and regions can be found in Table 34. A maximum of 256 regions can be displayed simultaneously on the screen.

Note: Only the regions being displayed at a given time count (and not the total on the whole OSD), so this number should be more than enough for even the most complex OSD.

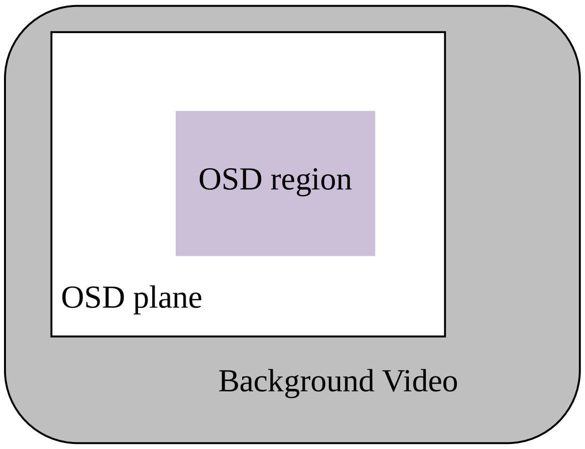


Figure 84: Definition of OSD Region

Table 34: Regions Used for OSD Components

Component	Number of Regions Needed in Hardware
OSDLabel	1
OSDImage	1
OSDHistogram	1
OSDKeyboard	2
OSDProgressbar	2
OSDTextbox	1
OSDMenubar	One region per item on each level
OSDListbox	One region per item
OSDTimer	0
OSDIptextbox	1

For example, if the designed OSD uses the OSD Menu bar component shown in Figure 85, and the user is moving through the icon menu, there will be three regions in use at the time when the selected icon is Node1 (that is, the elements from the same level, Node1, Node5 and Node6). When the selected icon is Node3, there will be three regions in use, that is, Node2, Node3, and Node4. When the selected icon is Node7, there will be two regions in use, that is, Node7 and Node8.

Note how the efficient translation of components to regions means that it is almost impossible to run out of regions while designing even the most complex OSD.

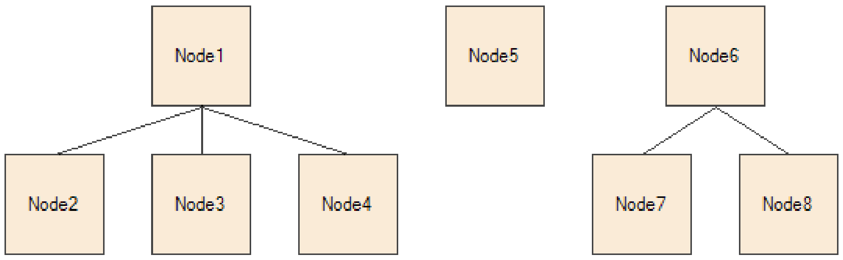


Figure 85: OSD Menu Bar Component

4.2.5.2. *OSD Color Space*

Bitmap images as well as external OSDs are passed to the OSD core in 8-bit RGB format. However, all video processing in the ADV8005 takes place in YCbCr. The OSD core features a CSC to enable conversion of the OSD data from RGB to YCbCr. The OSD core CSC can convert into either full of limited range YCbCr.

4.2.6. *OSD Timers*

ADV8005 OSD supports up to eight hardware timers. One of these timers (user-selectable in the OSD firmware) is used by the *OSDTimer* component of *Blimp OSD*, which can be inserted within any OSD design (consult the *Blimp OSD* manual for a detailed description of how to do this). *Blimp OSD* will automatically handle a number of OSD timers and will map all of them to one hardware timer. If the OSD design flow with the *Blimp OSD* tool is followed, the user does not need to know any low-level details about the timers. However, since they can also be used as general purpose system timers, its low-level functionality will be described in this section. Note that the HW timer being used by *Blimp OSD* (user-selectable as mentioned) will not be available to be used as general purpose timer.

Any of these eight timers can trigger an interrupt on the INT0 pin. This interrupt can then be handled by the MCU, and the timer which generated it can be found out by polling the timer registers.

These timers can be configured through the Timer register map. This map is only accessible through the SPI slave interface (address 0x0B). For more information on the SPI slave interface, refer to Section 4.2.8.2. The registers used to configure the timers are described below.

sys_clock_freq[23:0], SPI Device Address 0x0B (TIMER), *Address 0x00[7:0]; Address 0x01[7:0]; Address 0x02[7:0]*
System clock frequency, unit is KHz, the default value is 157.5 MHz.

Function

sys_clock_freq[23:0]	Description
0x02673C «	Default
0xXXXXXX	System Clock Frequency

This register is used to generate a 1 KHz pulse, which all eight timers are based on to measure a 1 ms interval. If the system clock frequency is changed, this register can be changed to guarantee the 1 KHz accuracy. It is also possible to modify this register if a smaller time interval than 1ms needs to be measured.

For example:

The default value of sys_clock_freq is 0x0278D0, that is, 162000 (162 MHz).
If it is changed to 16200, the minimum interval will be 0.1 ms.

If it is changed to 1620, the minimum interval will be 0.01ms.
timer1_enable, SPI Device Address 0x0B (TIMER), *Address 0x03[0]*
Timer 1 Enable

Function

timer1_enable	Description
0 «	Disables
1	Enables

Once the timer is enabled, disabling this bit will stop the counting, and it will be resumed when enabling back this bit.

Note that the rest of the bits within this register perform the same operation as for timer1 but for the other seven timers (i.e. bit[1] controls timer2, bit[2] controls timer3, etc.); they are not included here for readability reasons.

timer1_reset, SPI Device Address 0x0B (TIMER), *Address 0x04[0]*
Timer 1 Reset

Function

timer1_reset	**Description**
0 «	Not reset
1	Reset

Enabling this reset will clear the timer_cnt and timer_irq_cnt registers.

Note that the rest of the bits within this register perform the same operation as for timer1 but for the other seven timers (that is, bit[1] controls timer2, bit[2] controls timer3, and so on); they are not included here for readability reasons.

timer1_loop_mode, SPI Device Address 0x0B (TIMER), *Address 0x05[0]*
Timer 1 Mode Control

Function

timer1_loop_mode	**Description**
0 «	One time mode
1	Infinite mode

When working in one time mode, after the interval is reached, the timer will stop by itself, that is, there is no need to set timer_enable to disabled).

When working in infinite mode, timer_keep_result should be set to 0.

Note that the rest of the bits within this register perform the same operation as for timer1 but for the other seven timers (that is, bit[1] controls timer2, bit[2] controls timer3, and so on); they are not included here for readability reasons.

timer1_keep_result, SPI Device Address 0x0B (TIMER), *Address 0x06[0]*
Timer 1 result control.

Function

timer1_keep_result	**Description**
0 «	Does not keep timer counter value after timer done
1	Keep timer counter value after timer done

Note that the rest of the bits within this register perform the same operation as for timer1 but for the other seven timers (that is, bit[1] controls timer2, bit[2] controls timer3, and so on); they are not included here for readability reasons.

timer1_irq_en, SPI Device Address 0x0B (TIMER), *Address 0x07[0]*
Timer 1 interrupt enable.

Function

timer1_irq_en	**Description**
0 «	Disable
1	Enable

Note that the rest of the bits within this register perform the same operation as for timer1 but for the other seven timers (that is, bit[1] controls timer2, bit[2] controls timer3, and so on); they are not included here for readability reasons.

timer1_clr_irq, SPI Device Address 0x0B (TIMER), *Address 0x08[0]*
Clears the timer 1 interrupt after writing 1 to this bit. Note these are not self clearing bits, the user just needs to write 1 to this bit and it will clear the timer_flag and timer_irq_cnt registers. Even if timer_clr_irq is already set at 1, it will not clear the timer interrupt and flag until the user writes 1 to it.

Note that the rest of the bits within this register perform the same operation as for timer1 but for the other seven timers (i.e. bit[1] controls timer2, bit[2] controls timer3, etc.); they are not included here for readability reasons.

timer1_flag, SPI Device Address 0x0B (TIMER), *Address 0x09[0] (Read Only)*
Timer 1 flag.

Function

timer1_flag	**Description**
0 «	Timer 1 is running
1	Timer 1 is done

Note that the rest of the bits within this register perform the same operation as for timer1 but for the other seven timers (that is, bit[1] controls timer2, bit[2] controls timer3, and so on); they are not included here for readability reasons.

timer1_interval[31:0], SPI Device Address 0x0B (TIMER), *Address 0x0A[7:0]; Address 0x0B[7:0]; Address 0x0C[7:0]; Address 0x0D[7:0]*
Timer 1 interval, unit is ms.
timer2_interval[31:0], SPI Device Address 0x0B (TIMER), *Address 0x0E[7:0]; Address 0x0F[7:0]; Address 0x10[7:0]; Address 0x11[7:0]*
Timer 2 interval, unit is ms.
timer3_interval[31:0], SPI Device Address 0x0B (TIMER), *Address 0x12[7:0]; Address 0x13[7:0]; Address 0x14[7:0]; Address 0x15[7:0]*
Timer 3 interval, unit is ms.
timer4_interval[31:0], SPI Device Address 0x0B (TIMER), *Address 0x16[7:0]; Address 0x17[7:0]; Address 0x18[7:0]; Address 0x19[7:0]*
Timer 4 interval, unit is ms.
timer5_interval[31:0], SPI Device Address 0x0B (TIMER), *Address 0x1A[7:0]; Address 0x1B[7:0]; Address 0x1C[7:0]; Address 0x1D[7:0]*
Timer 5 interval, unit is ms.
timer6_interval[31:0], SPI Device Address 0x0B (TIMER), *Address 0x1E[7:0]; Address 0x1F[7:0]; Address 0x20[7:0]; Address 0x21[7:0]*
Timer 6 interval, unit is ms.
timer7_interval[31:0], SPI Device Address 0x0B (TIMER), *Address 0x22[7:0]; Address 0x23[7:0]; Address 0x24[7:0]; Address 0x25[7:0]*
Timer 7 interval, unit is ms.
timer8_interval[31:0], SPI Device Address 0x0B (TIMER), *Address 0x26[7:0]; Address 0x27[7:0]; Address 0x28[7:0]; Address 0x29[7:0]*
Timer 8 interval, unit is ms.

timer1_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x2A[7:0]; Address 0x2B[7:0]; Address 0x2C[7:0]; Address 0x2D[7:0] (Read Only)*
Timer 1 value, unit is ms.
timer2_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x2E[7:0]; Address 0x2F[7:0]; Address 0x30[7:0]; Address 0x31[7:0] (Read Only)*
Timer 2 value, unit is ms.
timer3_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x32[7:0]; Address 0x33[7:0]; Address 0x34[7:0]; Address 0x35[7:0] (Read Only)*
Timer 3 value, unit is ms.
timer4_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x36[7:0]; Address 0x37[7:0]; Address 0x38[7:0]; Address 0x39[7:0] (Read Only)*
Timer 4 value, unit is ms.
timer5_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x3A[7:0]; Address 0x3B[7:0]; Address 0x3C[7:0]; Address 0x3D[7:0] (Read Only)*
Timer 5 value, unit is ms.
timer6_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x3E[7:0]; Address 0x3F[7:0]; Address 0x40[7:0]; Address 0x41[7:0] (Read Only)*
Timer 6 value, unit is ms.
timer7_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x42[7:0]; Address 0x43[7:0]; Address 0x44[7:0]; Address 0x45[7:0] (Read Only)*
Timer 7 value, unit is ms.
timer8_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x46[7:0]; Address 0x47[7:0]; Address 0x48[7:0]; Address 0x49[7:0] (Read Only)*
Timer 8 value, unit is ms.
timer1_irq_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x4A[7:0]; Address 0x4B[7:0]; Address 0x4C[7:0]; Address 0x4D[7:0] (Read Only)*
The number of times the timer 1 interrupt was generated.
timer2_irq_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x4E[7:0]; Address 0x4F[7:0]; Address 0x50[7:0]; Address 0x51[7:0] (Read Only)*
The number of times the timer 2 interrupt was generated.
timer3_irq_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x52[7:0]; Address 0x53[7:0]; Address 0x54[7:0]; Address 0x55[7:0] (Read Only)*
The number of times the timer 3 interrupt was generated.
timer4_irq_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x56[7:0]; Address 0x57[7:0]; Address 0x58[7:0]; Address 0x59[7:0] (Read Only)*
The number of times the timer 4 interrupt was generated.
timer5_irq_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x5A[7:0]; Address 0x5B[7:0]; Address 0x5C[7:0]; Address 0x5D[7:0] (Read Only)*
The number of times the timer 5 interrupt was generated.

timer6_irq_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x5E[7:0]; Address 0x5F[7:0]; Address 0x60[7:0]; Address 0x61[7:0] (Read Only)*
The number of times the timer 6 interrupt was generated.
timer7_irq_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x62[7:0]; Address 0x63[7:0]; Address 0x64[7:0]; Address 0x65[7:0] (Read Only)*
The number of times the timer 7 interrupt was generated.
timer8_irq_cnt[31:0], SPI Device Address 0x0B (TIMER), *Address 0x66[7:0]; Address 0x67[7:0]; Address 0x68[7:0]; Address 0x69[7:0] (Read Only)*
The number of times the timer 8 interrupt was generated.

4.2.7. *OSD Scaler*

The ADV8005 OSD core contains an arbitrary resolution conversion scaler. This scaler performs a scaling function if the OSD resolution inside the DDR2 memory is different from the output video. If the output video is interlaced, the OSD scaler can change the progressive OSD data to interlaced data for blending. As mentioned in Section 4.2.3, the OSD scaler also guarantees the correct synchronization of OSD data and input video data.

4.2.8. *OSD Master/Slave SPI Interface*

The ADV8005 OSD requires an external DDR2 memory and some configuration done to the OSD SPI registers in order to work. OSD data can be written to the DDR2 memory on startup by the ADV8005. In addition, to dynamically configure the OSD, configuration registers need to be controlled. Note that all this configuration is taken care of by *Blimp OSD* and the firmware, so a detailed explanation of the DDR2 SPI interface is not provided. For this reason, this section covers only top level information (enable/disable, muxing configuration of the OSD through the IO Map I^{2}C registers). The SPI slave hardware interface is also described in this section.

4.2.8.1. Overview

It is possible to access the DDR2 and OSD SPI registers in one of two ways:

- The ADV8005 SPI master interface (serial port 2) can pull in resource data to DDR2 memory from an external SPI flash memory, as shown in Figure 86.
- The system MCU (SPI master) can write OSD data into DDR2 memory using the ADV8005 SPI slave interface (serial port 1), as shown in Figure 87.

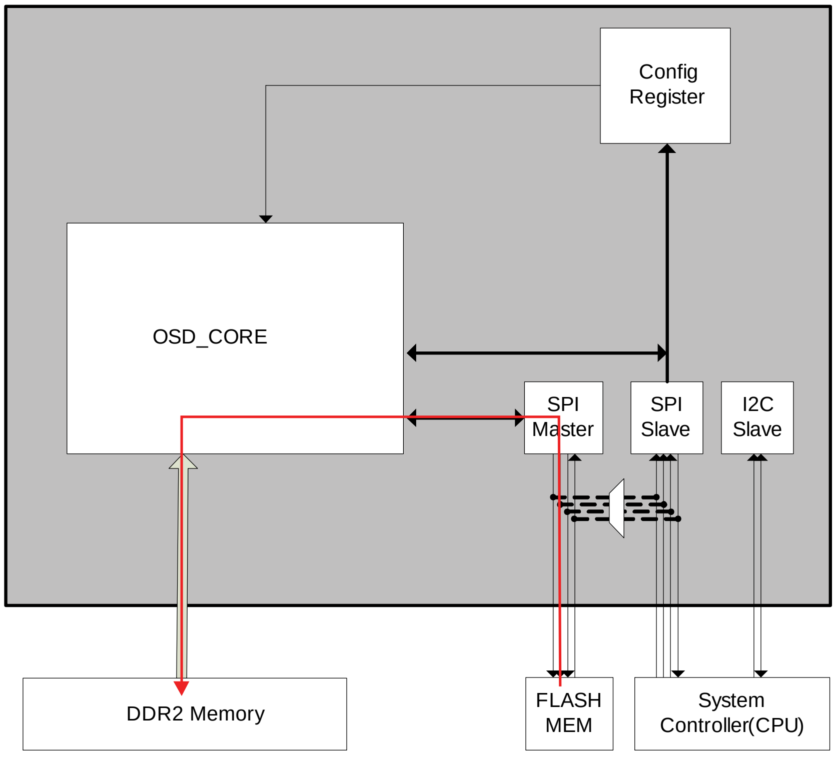


Figure 86: Data Loaded from SPI Flash Through ADV8005 SPI Master Interface

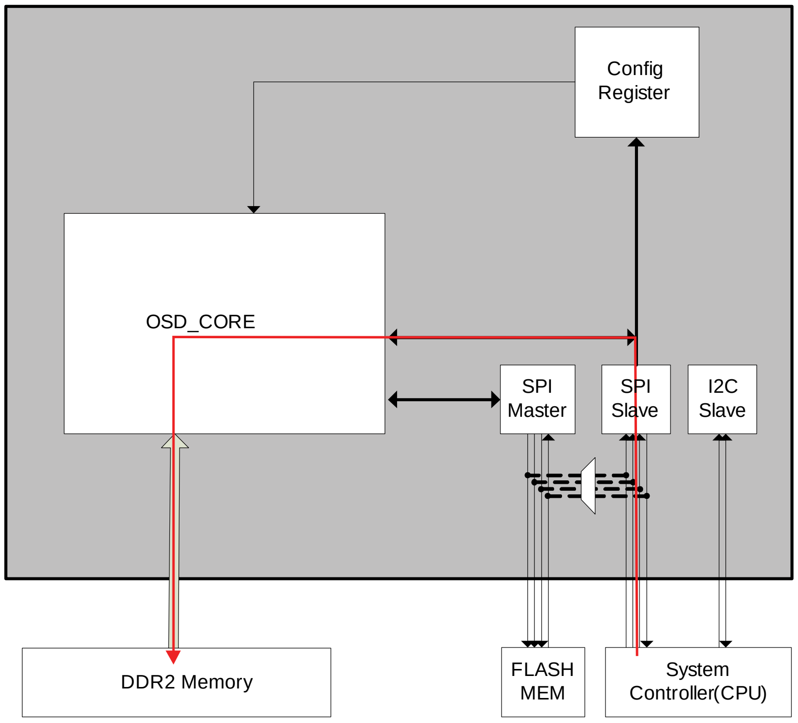


Figure 87: MCU as SPI Master Sending OSD Data Through ADV8005 SPI Slave Interface

Additionally, the system MCU (SPI master) can program the external flash by looping SPI commands through the SPI slave (serial port 1) and the SPI master (serial port 2) interfaces connected in a chain. In this mode, the OSD core just passes through MOSI, SS and SCLK signals from the MCU to the flash. Note that the system MCU is responsible for any error protection in this mode, as shown in Figure 88.

This option can be useful during the final debug stage of the OSD, in which the OSD design could be downloaded into the system SPI flash memory through, for example, the USB or RSR232 port of the MCU.

This mode can be enabled using the spi_loop_through mode which controls the mux shown in Figure 88.

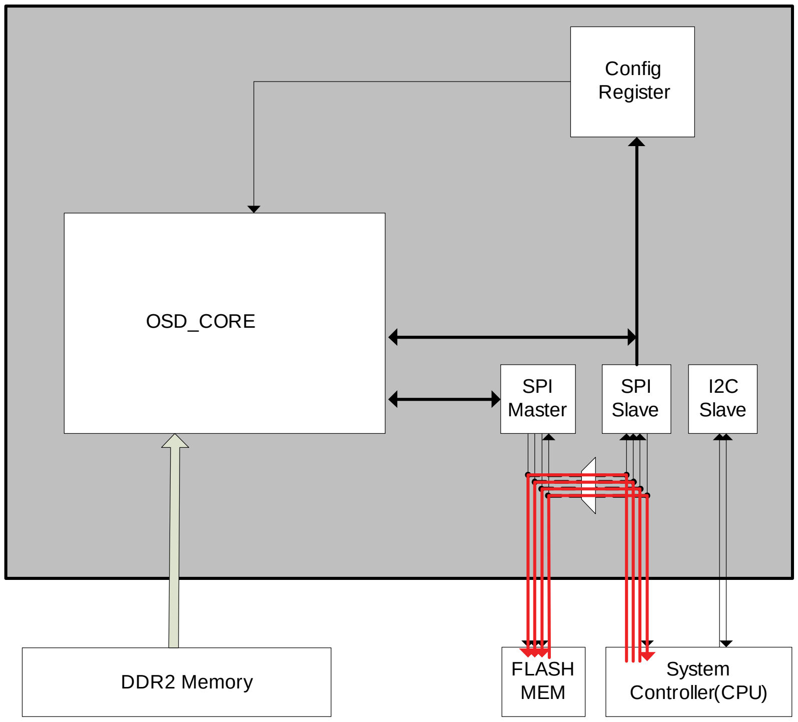


Figure 88: SPI Loopback Enabled so MCU Can Program SPI Flash

By default, the SPI ports are set in manual mode for the SPI which means the SPI pins are tristated (input). To make the SPI ports operational, the following register bits must be configured to automatic mode.

spi1_cs_oe_man_en, IO Map, *Address 0x1ACE[7]*
This bit is used to control the output enable manual override for spi1_cs.

Function

spi1_cs_oe_man_en	**Description**
0	Auto
1 (default)	manual override

spi1_miso_oe_man_en, IO Map, *Address 0x1ACE[6]*
This bit is used to control the output enable manual override for spi1_miso.

Function

spi1_miso_oe_man_en	**Description**
0	Auto
1 (default)	Manual override

spi1_mosi_oe_man_en, IO Map, *Address 0x1ACE[5]*
This bit is used to control the output enable manual override for spi1_mosi.

Function

spi1_mosi_oe_man_en	**Description**
0	Auto
1 (default)	Manual override

spi1_sclk_oe_man_en, IO Map, *Address 0x1ACE[4]*
This bit is used to control the output enable manual override for spi1_sclk.

Function

spi1_sclk_oe_man_en	**Description**
0	Auto
1 (default)	Manual override

spi2_cs_oe_man_en, IO Map, *Address 0x1ACE[3]*
This bit is used to control the output enable manual override for spi2_cs.

Function

spi2_cs_oe_man_en	Description
0	Auto
1 (default)	Manual override

spi2_miso_oe_man_en, IO Map, *Address 0x1ACE[2]*
This bit is used to control the output enable manual override for spi2_miso.

Function

spi2_miso_oe_man_en	Description
0	Auto
1 (default)	Manual override

spi2_mosi_oe_man_en, IO Map, *Address 0x1ACE[1]*
This bit is used to control the output enable manual override for spi2_mosi.

Function

spi2_mosi_oe_man_en	Description
0	Auto
1 (default)	Manual override

spi2_sclk_oe_man_en, IO Map, *Address 0x1ACE[0]*
This bit is used to control the output enable manual override for spi2_sclk.

Function

spi2_sclk_oe_man_en	Description
0	Auto
1 (default)	Manual override

For the majority of functions, the SPI ports can be left in automatic mode. If using the SPI ports in manual mode, the direction of the various pins can be configured using the following bits.

spi1_cs_oe_man, IO Map, *Address 0x1ACD[7]*
This bit is used to control the output enable for spi1 chip select.

Function

spi1_cs_oe_man	Description
0 (default)	Input
1	Output

spi1_miso_oe_man, IO Map, *Address 0x1ACD[6]*
This bit is used to control the output enable for spi1 'master in slave out'.

Function

spi1_miso_oe_man	Description
0 (default)	Input
1	Output

spi1_mosi_oe_man, IO Map, *Address 0x1ACD[5]*
This bit is used to control the output enable for spi1 'master out slave in'.

Function

spi1_mosi_oe_man	Description
0 (default)	Input
1	Output

spi1_sclk_oe_man, IO Map, *Address 0x1ACD[4]*
This bit is used to control the output enable for spi1 serial clock.

Function

spi1_sclk_oe_man	Description
0 (default)	Input
1	Output

spi2_cs_oe_man, IO Map, *Address 0x1ACD[3]*
This bit is used to control the output enable for spi2 chip select.

Function

spi2_cs_oe_man	**Description**
0 (default)	Input
1	Output

spi2_miso_oe_man, IO Map, *Address 0x1ACD[2]*
This bit is used to control the output enable for spi2 'master in slave out'.

Function

spi2_miso_oe_man	**Description**
0 (default)	Input
1	Output

spi2_mosi_oe_man, IO Map, *Address 0x1ACD[1]*
This bit is used to control the output enable for spi2 'master out slave in'.

Function

spi2_mosi_oe_man	**Description**
0 (default)	Input
1	Output

spi2_sclk_oe_man, IO Map, *Address 0x1ACD[0]*
This bit is used to control the output enable for spi2 serial clock.

Function

spi2_sclk_oe_man	**Description**
0 (default)	Input
1	Output

The SPI interface can be reset using spi_reset.

4.2.8.2. SPI Slave Interface

The ADV8005 SPI slave interface (serial port 1) is used by the MCU to send the OSD data to the DDR2 and to configure the OSD registers. Note that the SPI functions provided within the ADI libraries will automatically take care of any SPI transfer between the MCU and ADV8005.

Hence, the information in this section is provided just so the user can configure the MCU SPI master to match the ADV8005 SPI slave interface, and get both of them to communicate properly. Apart from this setup, the user should not try to access any other SPI register map (with the exception of the timer SPI registers), since all the OSD SPI communication is handled through the provided ADI firmware.

The SPI slave can support the following modes:

- CPOL = 0, CPHA=0
- CPOL = 0, CPHA=1
- CPOL = 1, CPHA=0
- CPOL = 1, CPHA=1

Figure 89 shows the effect that these settings may have on the data.

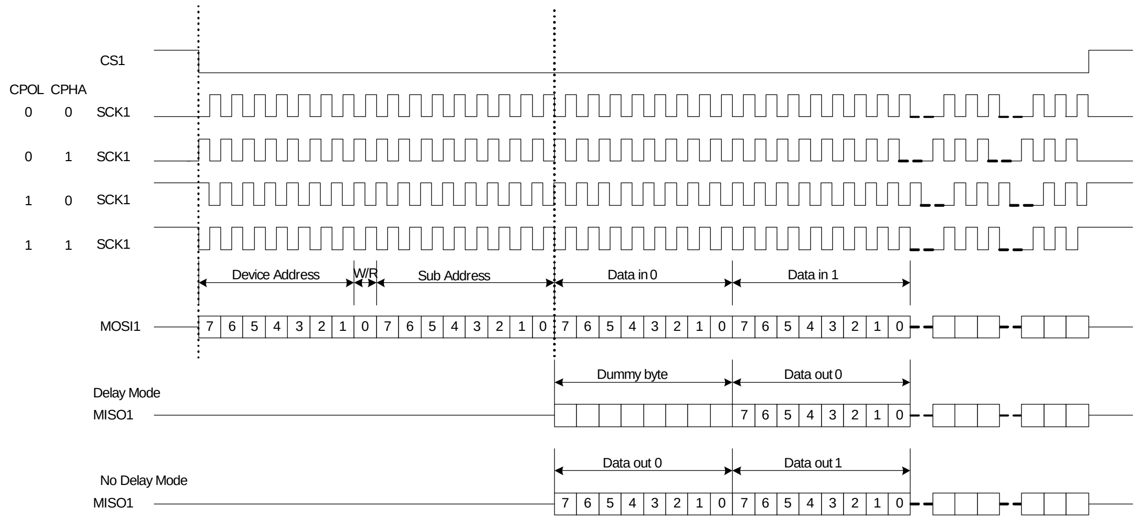


Figure 89: SPI Slave Interface Timing and Data Format

The CPOL/CPHA can be configured through the I^{2}C registers described below.

spi_slave_cpol, IO Map, *Address 0x1A14[3]*
This bit is used to select the SPI slave clock polarity.

Function

spi_slave_cpol	**Description**
0	Idle state, clock is low
1 (default)	Idle state, clock is high

spi_slave_cpha, IO Map, *Address 0x1A14[2]*
This bit is used to select the SPI slave clock phase.

Function

spi_slave_cpha	**Description**
0	Negedge used
1 (default)	Posedge used

As can be seen from Figure 89, the LSB bit of the device address sets whether the access is read or write.

The SPI subaddress is an 8-bit field and the data is also 8 bits wide with MSB sent first and LSB last.
The SPI slave readback has both delay mode and no delay mode, and it is controlled by the following SPI register.

slave_delay_mode, SPI Device Address 0x0A, *Address 0x85[0]*
SPI slave read data MISO1 output delay mode.

Function

slave_delay_mode	**Description**
0	No delay
1 «	Delay 8 clocks (8 bits dummy data)

In no delay mode, counting from the last rising edge of SCK1 (send subaddress) to the first falling edge of SCK1 (send out MISO1), there are about 10 system clock delays. Assuming the SCK1 is 50% duty cycle, only when SCK1 is slower than system clock/20 = 162 MHz/20 = 8.1 MHz, can no delay mode work normally.

If SCK1 is slower than 6 MHz, no delay mode can be set.

The ADV8005 features an analog antiglitch used to reject glitches on SCK1 (SPI slave). There are three modes of operation of this filter: bypass, 2 ns glitch rejection, and 5ns glitch rejection. The 2 ns glitch rejection mode should be used for clock frequencies between 10MHz and 40 MHz. The 5 ns glitch rejection mode should be used for clock frequencies of less than 10 MHz.

spi_filter_en, IO Map, *Address 0x1A2C[7]*
This bit is used to enable the SPI anti glitch filter.

Function

spi_filter_en	**Description**
0 (default)	Anti glitch filter disable
1	Anti glitch filter enable

spi_filter_sel, IO Map, *Address 0x1A2C[6]*
This bit is used to select the response of the SPI anti glitch filter.

Function

spi_filter_sel	**Description**
0 (default)	2ns glitch rejection
1	5ns glitch rejection

4.2.8.3. SPI Master Interface

The ADV8005 SPI master interface (serial port 2) is used by the ADV8005 to read the OSD binary file (output by *Blimp OSD*) from an external SPI flash memory, and to copy it to the DDR2 memory. Note that the library of functions provided by ADI will take care of this process; the information in this section is just provided so the user can find a suitable SPI flash memory which can be interfaced to the ADV8005 SPI master interface.

The SPI master is designed to be compatible with the M25P16 and supports the FAST_READ command. The SPI master clock can be configured to support up to 80 MHz. The SPI master, similar to the slave, can support the following modes:

- CPOL = 0, CPHA=0
- CPOL = 0, CPHA=1
- CPOL = 1, CPHA=0
- CPOL = 1, CPHA=1

Figure 90 shows the effect that these settings may have on the data.

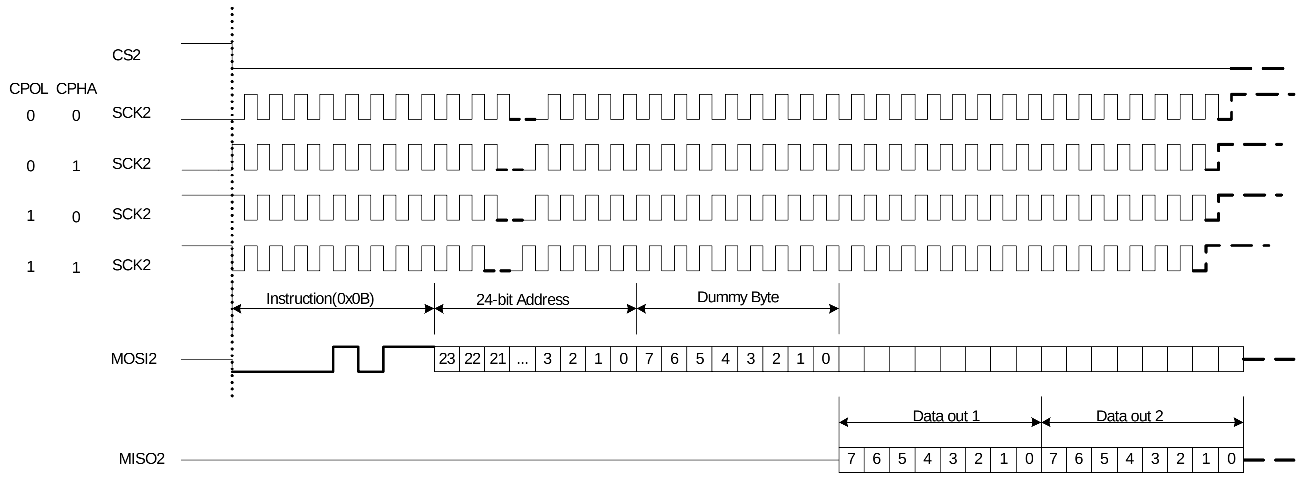


Figure 90: SPI Master Interface Timing and Data Format

The CPOL/CPHA can be configured through the following I²C registers.

spi_master_cpol, IO Map, *Address 0x1A14[1]*
This bit is used to select the SPI master clock polarity.

Function

spi_master_cpol	**Description**
0 (default)	Idle state, clock is low
1	Idle state, clock is high

spi_master_cpha, IO Map, *Address 0x1A14[0]*
This bit is used to select the SPI master clock phase.

Function

spi_master_cpha	**Description**
0 (default)	Negedge used
1	Posedge used

4.2.9. *OSD Initialization*

To configure ADV8005 to use the OSD, the following I²C writes are required:

0x1A14=0x0C: SPI mode select
0x1ACE=0x00: SPI bus enable
0x1ACC=0x10: Configure OSD HW int

Further SPI writes are required but these are controlled through the OSD.

5. SERIAL VIDEO RECEIVER

The Serial Video Rx on the ADV8005 can receive video data at rates of up to 3 GHz. This allows support for video formats ranging from SD to 4k x 2k @ 24Hz, 1080p120Hz and 1080p60 3D. The Serial Video Rx on the ADV8005 can receive video data at rates of up to 2.25 GHz. This allows support for video formats ranging from SD to 1080p @ 60Hz 12-bit. It is designed for chip-to-chip connection only and, as such, does not offer any DDC lines to facilitate HDCP or EDID operations.

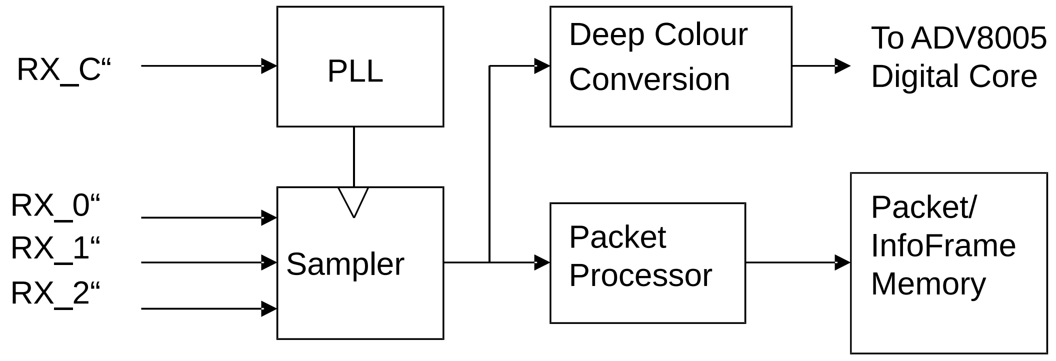


Figure 91: Functional Block Diagram of ADV8005 Serial Video Rx

This section outlines the various registers available to the user in the register map which is used to control the Serial Video Rx. These registers are used to configure the ADV8005 to accept input video from a device such as an HDMI transceiver (for example, ADV7623) or a front end device with HDMI output (for example, ADV7850).

5.1. + 5 V DETECT

The Serial Video Rx on the ADV8005 can monitor the level on the +5 V power signal pin. This +5 V signal can be used to reset the Rx section if requested. If +5 V detection is not being used, this pin should be connected to a +5 V supply. The controls for +5 V detection can be found in the following I^{2}C registers. These registers are valid even when the part is not processing TMDS information.

filt_5v_det_dis, HDMI RX Map, *Address 0xE256[7]*
This bit is used to disable the digital glitch filter on the HDMI 5V detect signals. The filtered signals are used as interrupt flags, and also used to reset the HDMI section. The filter works from an internal ring oscillator clock and is therefore available in power-down mode. The clock frequency of the ring oscillator is 42MHz +/-10%.

Note: If the 5 V pins are not used and left unconnected, the 5 V detect circuitry should be disconnected from the HDMI reset signal by setting dis_cable_det_rst to 1. This avoids holding the HDMI section in reset.

Function

filt_5v_det_dis	Description
0 (default)	Enabled
1	Disabled

Note: If the +5 V pins are not used and left unconnected, the +5 V detect circuitry should be disconnected from the Rx reset circuitry by setting dis_cable_det_rst to 1. This avoids holding the Rx section in reset.

filt_5v_det_timer[6:0], HDMI RX Map, *Address 0xE256[6:0]*
This bit is used to set the timer for the digital glitch filter on the HDMI +5 V detect inputs. The unit of this parameter is 2 clock cycles of the ring oscillator (~ 47ns). The input must be constantly high for the duration of the timer, otherwise the filter output remains low. The output of the filter returns low as soon as any change in the +5 V power signal is detected.

Function

filt_5v_det_timer[6:0]	Description
1011000 (default)	Approximately 4.2us
xxxxxxx	Time duration of +5 V deglitch filter. The unit of this parameter is 2 clock cycles of the ring oscillator (~ 47ns)

dis_cable_det_rst, HDMI RX Map, *Address 0xE248[6]*
This bit is used to disable the reset effects of cable detection. It should be set to 1 if the +5 V pins are unused and left unconnected.

Function

dis_cable_det_rst	Description
0 (default)	Resets the HDMI section if the 5 V input pin is inactive
1	Do not use the 5 V input pins as reset signal for the HDMI section

5.2. TMDS CLOCK ACTIVITY DETECTION

The ADV8005 Serial Video Rx provides circuitry to monitor TMDS clock activity and also the type of data on the Rx input lines. System software can poll these registers and configure the ADV8005 as required. rb_rx_tmds_clk_det and tmds_pll_locked can be used to determine if there is a valid clock on the TMDS clock input and if the Serial Video Rx has locked to this. If both of these are true, rx_hdmi_mode can be used to indicate the video data that is available on the Serial Video Rx, either DVI data or HDMI data.

rx_hdmi_mode, HDMI RX Map, *Address 0xE205[7] (Read Only)*
This bit is a readback to indicate whether the stream processed by the HDMI core is a DVI or an HDMI stream.

Function

rx_hdmi_mode	Description
0 (default)	DVI Mode Detected
1	HDMI Mode Detected

rb_rx_tmds_clk_det, IO Map, *Address 0x1ADF[3] (Read Only)*
This bit is used to indicate if there is a clock on the Serial Video RX input lines.

Function

rb_rx_tmds_clk_det	Description
0 (default)	No TMDS clock detected on the Serial Video RX input lines
1	TMDS clock detected on Serial Video RX input lines

tmds_pll_locked, HDMI RX Map, *Address 0xE204[1] (Read Only)*
This bit is a readback to indicate if the TMDS PLL is locked to the TMDS clock input of the selected HDMI port.

Function

tmds_pll_locked	Description
0 (default)	The TMDS PLL is not locked
1	The TMDS PLL is locked to the TMDS clock input of the selected HDMI port.

Note: The tmds_pll_locked flag should be considered valid if a TMDS clock is input on the Serial Video Rx.

freqtolerance[3:0], HDMI RX Map, *Address 0xE20D[3:0]*
Sets the tolerance in MHz for new TMDS frequency detection. This tolerance is used for the audio mute mask mt_msk_vclk_chng and the HDMI status bit new_tmds_frq_raw.

Function

freqtolerance[3:0]	Description
0100 «	Default tolerance in MHz for new TMDS frequency detection
xxxx	Tolerance in MHz for new TMDS frequency detection

5.3. CLOCK AND DATA TERMINATION CONTROL

The ADV8005 provides the clock_terma_disable control for TMDS clock and data termination on all Serial Video Rx input pins.

clock_terma_disable, HDMI RX Map, *Address 0xE283[0]*
This control is used to disable clock termination on port A. It can be used when term_auto is set to 0.

Function

clock_terma_disable	Description
0	Enable Termination port A
1 (default)	Disable Termination port A

5.4. AV MUTE STATUS

av_mute is used to indicate the status of the avmute bit in the general control packet. As with the TMDS clock detection bits, this register bit can be polled by the system software and the appropriate configuration done.

av_mute, HDMI RX Map, *Address 0xE204[6] (Read Only)*
This bit is a readback of AVMUTE status received in the last General Control packet received.

Function

av_mute	**Description**
0 (default)	AVMUTE not set
1	AVMUTE set

5.5. DEEP COLOR MODE SUPPORT

The ADV8005 supports HDMI streams with 24-bits per sample and deep color modes. The addition of a video FIFO (refer to Section 5.6 for more details) allows for the robust support of these modes.

The deep color mode information that the ADV8005 extracts from the general control packet can be read back from deep_color_mode[1:0]. It is possible to override the deep color mode that the ADV8005 unpacks from the video data encapsulated in the processed HDMI stream. This is achieved by configuring the override_deep_color_mode and deep_color_mode_user[1:0] controls.

deep_color_mode[1:0], HDMI RX Map, *Address 0xE20B[7:6] (Read Only)*
This control is a readback indicating the deep color mode information extracted from the general control packet.

Function

deep_color_mode[1:0]	**Description**
00 (default)	8-bits per channel
01	10-bits per channel
10	12-bits per channel
11	16-bits per channel (not supported)

override_deep_color_mode, HDMI RX Map, *Address 0xE240[6]*
This bit is used to override the Deep Color mode.

Function

override_deep_color_mode	**Description**
0 (default)	The HDMI section unpacks the video data according to the deep-color information extracted from the General Control packets. (Normal operation)
1	Override the deep color mode extracted from the General Control Packet. The HDMI section unpacks the video data according to the Deep Color mode set in DEEP_COLOR_MODE_USER[1:0].

deep_color_mode_user[1:0], HDMI RX Map, *Address 0xE240[5:4]*
This control is used to manually set the Deep Color mode. The value set in this register is effective when override_deep_color_mode is set to 1.

Function

deep_color_mode_user[1:0]	**Description**
00 (default)	8 bits per channel
01	10 bits per channel
10	12 bits per channel
11	16 bits per channel (not supported)

Notes:

- Deep color mode can be monitored via the deepcolor_mode_chng edge sensitive interrupt in the IO Map, which indicates if the color depth of the processed stream has changed.
- The ADV8005 can be configured to trigger an interrupt when the deepcolor_mode_chng edge sensitive interrupt in the IO Map changes from 0 to 1.

5.6. VIDEO FIFO

The ADV8005 contains a FIFO located after the TMDS decoding block (refer to Figure 92). Data arriving into the Serial Video Rx will be at 1X rate for non deep color modes (8-bits per channel), and 1.25X, 1.5X, or 2X for deep color modes (30, 36 and 48 bits respectively). Data unpacking and data rate reduction must be performed on the incoming data to provide the ADV8005 digital core with the correct data rate and data bit width. The video FIFO is used to pass data safely across the clock domains.

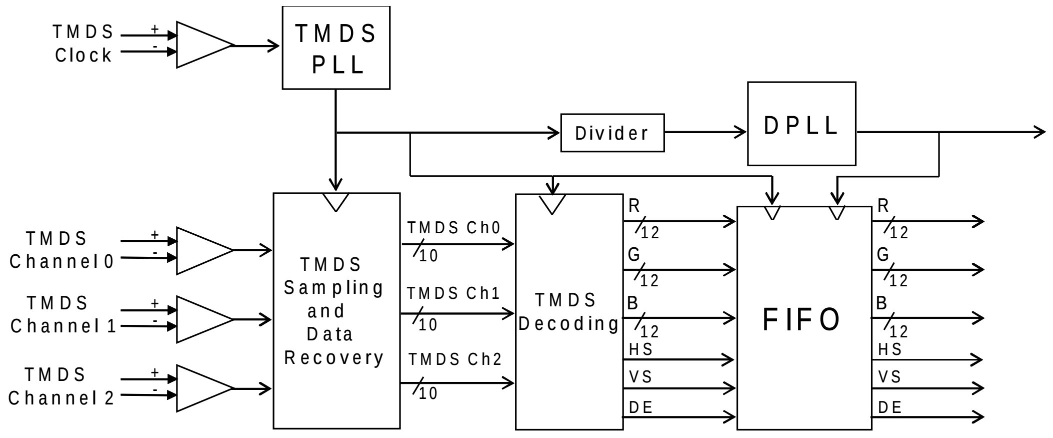


Figure 92: HDMI Video FIFO

The video FIFO is designed to operate completely autonomously. It automatically resynchronizes the read and write pointers if they are about to point to the same location. However, it is also possible for the user to observe and control the FIFO operation with a number of FIFO control and status registers described below.

dcfifo_level[2:0], HDMI RX Map, *Address 0xE21C[2:0] (Read Only)*
This signal is a readback to indicate the distance between the read and write pointers. Overflow and underflow will read as level 0. The ideal centered functionality will read as 0b100.

Function

dcfifo_level[2:0]	Description
000 (default)	FIFO has underflowed or overflowed
001	FIFO is about to overflow
010	FIFO has some margin
011	FIFO has some margin
100	FIFO perfectly balanced
101	FIFO has some margin.
110	FIFO has some margin.
111	FIFO is about to underflow

dcfifo_locked, HDMI RX Map, *Address 0xE21C[3] (Read Only)*
This bit is a readback to indicate if the Video FIFO is locked.

Function

dcfifo_locked	Description
0 (default)	Video FIFO is not locked. Video FIFO had to resynchronize between previous two Vsyncs
1	Video FIFO is locked. Video FIFO did not have to resynchronize between previous two Vsyncs

dcfifo_recenter, HDMI RX Map, *Address 0xE25A[2] (Self-Clearing)*
This bit is used as a reset to recenter the Video FIFO. This is a self clearing bit.

Function

dcfifo_recenter	Description
0 (default)	Video FIFO normal operation.
1	Video FIFO to re-centre.

dcfifo_kill_dis, HDMI RX Map, *Address 0xE21B[2]*
This bit is used to control whether or not the Video FIFO output is zeroed if there is more than one resynchronization of the pointers within 2 FIFO cycles. This behavior can be disabled with this bit.

Function

dcfifo_kill_dis	Description
0 (default)	FIFO output set to zero if more than one resynchronization is necessary during two FIFO cycles
1	FIFO output never set to zero regardless of how many resynchronizations occur

dcfifo_kill_not_locked, HDMI RX Map, *Address 0xE21B[3]*
This bit control is used to control whether or not the output of the Video FIFO is set to zero when the video PLL is unlocked.

Function

dcfifo_kill_not_locked	Description
0	FIFO data is output regardless of video PLL lock status
1 (default)	FIFO output is zeroed if video PLL is unlocked

The DCFIFO is programmed to reset itself automatically when the video PLL transitions from unlocked to locked. Note that the video PLL transition does not necessarily indicate that the overall system is stable.

dcfifo_reset_on_lock, HDMI RX Map, *Address 0xE21B[4]*
This bit is used to enable the reset/re-centering of video FIFO on video PLL unlock

Function

dcfifo_reset_on_lock	Description
0	Do not reset on video PLL lock
1 (default)	Reset FIFO on video PLL lock

5.7. PIXEL REPETITION

In HDMI mode, video formats with TMDS rates below 25 Mpixels/s require pixel repetition in order to be transmitted over the serial video link. When the ADV8005 receives this type of video format, it discards repeated pixel data automatically, based on the pixel repetition field available in the AVI InfoFrame.

When hdmi_pixel_repetition[3:0] is non zero, video pixel data is discarded and the pixel clock frequency is divided by hdmi_pixel_repetition + 1.

hdmi_pixel_repetition[3:0], HDMI RX Map, *Address 0xE205[3:0] (Read Only)*
This signal is a readback to provide the current HDMI pixel repetition value decoded from the AVI Infoframe received. The HDMI receiver automatically discards repeated pixel data and divides the pixel clock frequency appropriately as per the pixel repetition value.

Function

hdmi_pixel_repetition[3:0]	Description
0000 (default)	1x
0001	2x
0010	3x
0011	4x
0100	5x
0101	6x
0110	7x
0111	8x
1000	9x
1001	10x
1010 - 1111	Reserved

derep_n_override, HDMI RX Map, *Address 0xE241[4]*
This bit is used to allow the user to override the pixel repetition factor. derep_n is then used instead of hdmi_pixel_repetition[3:0] to discard video pixel data from the incoming HDMI stream.

Function

derep_n_override	Description
0 (default)	Automatic detection and processing of pixel repeated modes using the AVI infoframe information.
1	Enables manual setting of the pixel repetition factor as per DEREP_N[3:0].

derep_n[3:0], HDMI RX Map, *Address 0xE241[3:0]*
This signal is used to set the derepetition value if derep_n_override is set to 1.

Function

derep_n[3:0]	**Description**
0000 (default)	DEREP_N+1 indicates the pixel and clock discard factor
xxxx	DEREP_N+1 indicates the pixel and clock discard factor

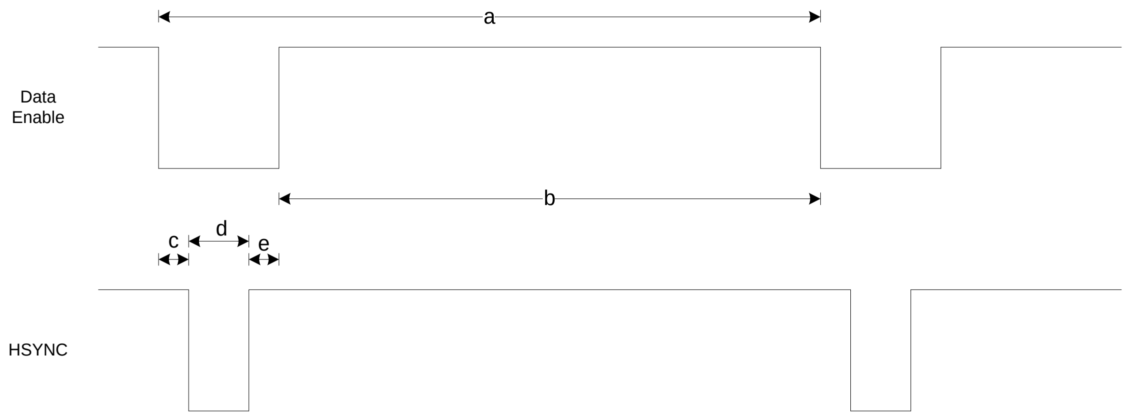
5.8. SYNC SIGNAL POLARITY READBACKS

These signals are used to indicate the polarity of the synchronization signals input to the Serial Video Rx input.

dvi_hsync_polarity, HDMI RX Map, *Address 0xE205[5] (Read Only)*
This bit is a readback to indicate the polarity of the HSync encoded in the input stream

Function

dvi_hsync_polarity	**Description**
0 (default)	The HSync is active low
1	The HSync is active high



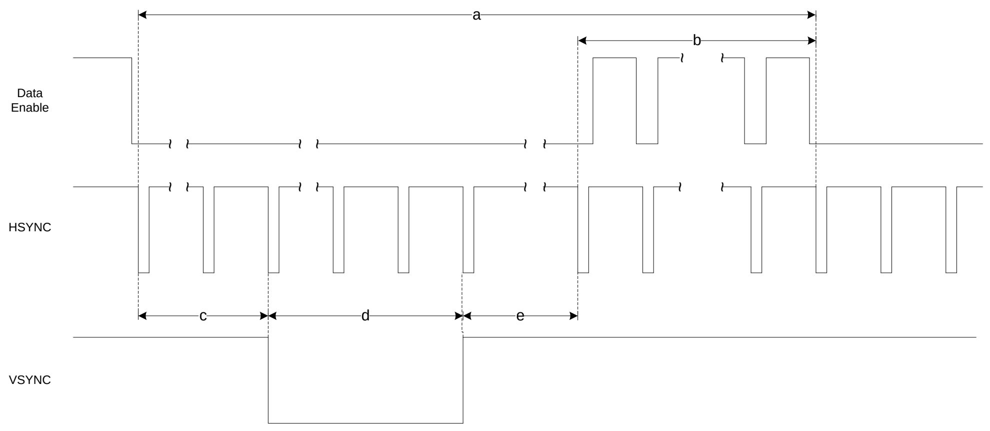
a	Total number of pixels per line	d	HSync width in pixel unit
b	Active number of pixels per line	e	HSync back porch width in pixel unit
c	HSync front porch width in pixel unit		

Figure 93: Horizontal Timing Parameters

dvi_vsync_polarity, HDMI RX Map, *Address 0xE205[4] (Read Only)*
This bit is a readback to indicate the polarity of the VSync encoded in the input stream

Function

dvi_vsync_polarity	**Description**
0 (default)	The VSync is active low
1	The VSync is active high



a	Total number of lines in field 0. Unit is in half lines.
b	Actives number of lines in field 0. Unit is in lines.
c	VSync front porch width in field 0. Unit is in half lines.
d	VSync pulse width in field 0. Unit is in half lines.
e	VSync back porch width in field 0. Unit is in half lines.

Figure 94: Vertical Parameters for Field 0

Note: Field 1 measurements should not be used for progressive video modes.

5.9. INFOFRAME REGISTERS

In HDMI, the auxiliary data is carried across the digital link using a series of packets. The ADV8005 Serial Video Rx can automatically detect and store the following HDMI packets:

- InfoFrames
- Audio content protection
- International Standard Recording Code (ISRC)
- Gamut metadata

Section 5.9.1 explains the method through which the ADV8005 can extract and store these InfoFrames.

5.9.1. *InfoFrame Collection Mode*

The ADV8005 has two modes for storing the InfoFrame packets sent from the source into the internal memory. By default, the ADV8005 only stores the InfoFrame packets received if the checksum is correct for each InfoFrame.

The ADV8005 also provides a mode to store every InfoFrame sent from the source, regardless of an InfoFrame packet checksum error. This can be configured by setting always_store_inf to 1.

always_store_inf, HDMI RX Map, *Address 0xE247[0]*
This bit is used to force InfoFrames with checksum errors to be stored.

Function

always_store_inf	**Description**
0 (default)	Stores data from received InfoFrames only if their checksum is correct
1	Always store the data from received InfoFrame regardless of their checksum

5.9.2. *InfoFrame Checksum Error Flags*

To determine if a checksum error has occurred with the InfoFrame packets, the user can poll the various status bits in the IO Map. There are several interrupt flags in the IO Map which indicate the status of the various InfoFrames. Refer to Section 8.2.2 for more details on the Serial Video Rx interrupts.

5.9.3. *AVI InfoFrame Registers*

Table 35 provides a list of readback registers for the AVI InfoFrame data. Refer to the EIA/CEA-861 specifications for a detailed explanation of the AVI InfoFrame fields.

Table 35: AVI InfoFrame Registers

InfoFrame Map Address	Access Type	Register Name	Byte Name[1]
0xE3E0	R/W	avi_packet_id[7:0]	Packet Type Value
0xE3E1	R	avi_inf_ver	InfoFrame version number
0xE3E2	R	avi_inf_len	InfoFrame length
0xE300	R	avi_inf_pb_0_1	Checksum
0xE301	R	avi_inf_pb_0_2	Data Byte 1
0xE302	R	avi_inf_pb_0_3	Data Byte 2
0xE303	R	avi_inf_pb_0_4	Data Byte 3
0xE304	R	avi_inf_pb_0_5	Data Byte 4
0xE305	R	avi_inf_pb_0_6	Data Byte 5
0xE306	R	avi_inf_pb_0_7	Data Byte 6
0xE307	R	avi_inf_pb_0_8	Data Byte 7
0xE308	R	avi_inf_pb_0_9	Data Byte 8
0xE309	R	avi_inf_pb_0_10	Data Byte 9
0xE30A	R	avi_inf_pb_0_11	Data Byte 10
0xE30B	R	avi_inf_pb_0_12	Data Byte 11
0xE30C	R	avi_inf_pb_0_13	Data Byte 12
0xE30D	R	avi_inf_pb_0_14	Data Byte 13
0xE30E	R	avi_inf_pb_0_15	Data Byte 14
0xE30F	R	avi_inf_pb_0_16	Data Byte 15
0xE310	R	avi_inf_pb_0_17	Data Byte 16
0xE311	R	avi_inf_pb_0_18	Data Byte 17
0xE312	R	avi_inf_pb_0_19	Data Byte 18
0xE313	R	avi_inf_pb_0_20	Data Byte 19
0xE314	R	avi_inf_pb_0_21	Data Byte 20
0xE315	R	avi_inf_pb_0_22	Data Byte 21
0xE316	R	avi_inf_pb_0_23	Data Byte 22
0xE317	R	avi_inf_pb_0_24	Data Byte 23
0xE318	R	avi_inf_pb_0_25	Data Byte 24
0xE319	R	avi_inf_pb_0_26	Data Byte 25
0xE31A	R	avi_inf_pb_0_27	Data Byte 26
0xE31B	R	avi_inf_pb_0_28	Data Byte 27

[1]As defined by the EIA/CEA-861 specifications

The AVI InfoFrame registers are considered valid if the following two conditions are met:

- avi_infoframe_det is 1.
- avi_inf_cksum_err is 0. This condition applies only if always_store_inf is set to 1.

5.9.4. *SPD InfoFrame Registers*

Table 36 provides a list of readback registers available for the SPD InfoFrame. Refer to the EIA/CEA-861 specifications for a detailed explanation of the SPD InfoFrame fields.

Table 36: SPD InfoFrame Registers

InfoFrame Map Address	Access Type	Register Name	Byte Name[1]
0xE3E6	R/W	spd_packet_id[7:0]	Packet Type Value
0xE3E7	R	spd_inf_ver	InfoFrame version number
0xE3E8	R	spd_inf_len	InfoFrame length
0xE32A	R	spd_inf_pb_0_1	Checksum
0xE32B	R	spd_inf_pb_0_2	Data Byte 1
0xE32C	R	spd_inf_pb_0_3	Data Byte 2
0xE32D	R	spd_inf_pb_0_4	Data Byte 3
0xE32E	R	spd_inf_pb_0_5	Data Byte 4
0xE32F	R	spd_inf_pb_0_6	Data Byte 5
0xE330	R	spd_inf_pb_0_7	Data Byte 6
0xE331	R	spd_inf_pb_0_8	Data Byte 7
0xE332	R	spd_inf_pb_0_9	Data Byte 8
0xE333	R	spd_inf_pb_0_10	Data Byte 9
0xE334	R	spd_inf_pb_0_11	Data Byte 10
0xE335	R	spd_inf_pb_0_12	Data Byte 11
0xE336	R	spd_inf_pb_0_13	Data Byte 12
0xE337	R	spd_inf_pb_0_14	Data Byte 13
0xE338	R	spd_inf_pb_0_15	Data Byte 14
0xE339	R	spd_inf_pb_0_16	Data Byte 15
0xE33A	R	spd_inf_pb_0_17	Data Byte 16
0xE33B	R	spd_inf_pb_0_18	Data Byte 17
0xE33C	R	spd_inf_pb_0_19	Data Byte 18
0xE33D	R	spd_inf_pb_0_20	Data Byte 19
0xE33E	R	spd_inf_pb_0_21	Data Byte 20
0xE33F	R	spd_inf_pb_0_22	Data Byte 21
0xE340	R	spd_inf_pb_0_23	Data Byte 22
0xE341	R	spd_inf_pb_0_24	Data Byte 23
0xE342	R	spd_inf_pb_0_25	Data Byte 24
0xE343	R	spd_inf_pb_0_26	Data Byte 25
0xE344	R	spd_inf_pb_0_27	Data Byte 26
0xE345	R	spd_inf_pb_0_28	Data Byte 27

[1]As defined by the EIA/CEA-861 specifications

The Source Product Descriptor (SPD) InfoFrame registers are considered valid if the following two conditions are met:

- spd_infoframe_det is 1.
- spd_inf_cksum_err is 0. This condition only applies if always_store_inf is set to 1.

5.9.5. *MPEG Source InfoFrame Registers*

Table 37 provides a list of readback registers available for the MPEG InfoFrame. Refer to the EIA/CEA-861 specifications for a detailed explanation of the MPEG InfoFrame fields.

Table 37: MPEG InfoFrame Registers

InfoFrame Map Address	Access Type	Register Name	Byte Name[1]
0xE3E9	R/W	ms_packet_id[7:0]	Packet Type Value
0xE3EA	R	ms_inf_vers	InfoFrame version number
0xE3EB	R	ms_inf_len	InfoFrame length
0xE346	R	ms_inf_pb_0_1	Checksum
0xE347	R	ms_inf_pb_0_2	Data Byte 1
0xE348	R	ms_inf_pb_0_3	Data Byte 2
0xE349	R	ms_inf_pb_0_4	Data Byte 3
0xE34A	R	ms_inf_pb_0_5	Data Byte 4
0xE34B	R	ms_inf_pb_0_6	Data Byte 5
0xE34C	R	ms_inf_pb_0_7	Data Byte 6
0xE34D	R	ms_inf_pb_0_8	Data Byte 7
0xE34E	R	ms_inf_pb_0_9	Data Byte 8
0xE34F	R	ms_inf_pb_0_10	Data Byte 9
0xE350	R	ms_inf_pb_0_11	Data Byte 10
0xE351	R	ms_inf_pb_0_12	Data Byte 11
0xE352	R	ms_inf_pb_0_13	Data Byte 12
0xE353	R	ms_inf_pb_0_14	Data Byte 13

[1]As defined by the EIA/CEA-861 specifications

- The MPEG InfoFrame registers are considered valid if the following two conditions are met: ms_infoframe_det is 1.
- ms_inf_cksum_err is 0. This condition applies only if always_store_inf is set to 1.

5.9.6. *Vendor Specific InfoFrame Registers*

Table 38 provides a list of readback registers available for the Vendor Specific InfoFrame.

Table 38: VS InfoFrame Registers

InfoFrame Map Address	R/W	Register Name	Byte Name
0xE3EC	R	vs_packet_id[7:0]	Packet Type Value
0xE3ED	R	vs_inf_vers	InfoFrame version number
0xE3EE	R	vs_inf_len	InfoFrame length
0xE354	R	vs_inf_pb_0_1	Checksum
0xE355	R	vs_inf_pb_0_2	Data Byte 1
0xE356	R	vs_inf_pb_0_3	Data Byte 2
0xE357	R	vs_inf_pb_0_4	Data Byte 3
0xE358	R	vs_inf_pb_0_5	Data Byte 4
0xE359	R	vs_inf_pb_0_6	Data Byte 5
0xE35A	R	vs_inf_pb_0_7	Data Byte 6
0xE35B	R	vs_inf_pb_0_8	Data Byte 7
0xE35C	R	vs_inf_pb_0_9	Data Byte 8
0xE35D	R	vs_inf_pb_0_10	Data Byte 9
0xE35E	R	vs_inf_pb_0_11	Data Byte 10
0xE35F	R	vs_inf_pb_0_12	Data Byte 11
0xE360	R	vs_inf_pb_0_13	Data Byte 12

InfoFrame Map Address	R/W	Register Name	Byte Name
0xE361	R	vs_inf_pb_0_14	Data Byte 13
0xE362	R	vs_inf_pb_0_15	Data Byte 14
0xE363	R	vs_inf_pb_0_16	Data Byte 15
0xE364	R	vs_inf_pb_0_17	Data Byte 16
0xE365	R	vs_inf_pb_0_18	Data Byte 17
0xE366	R	vs_inf_pb_0_19	Data Byte 18
0xE367	R	vs_inf_pb_0_20	Data Byte 19
0xE368	R	vs_inf_pb_0_21	Data Byte 20
0xE369	R	vs_inf_pb_0_22	Data Byte 21
0xE36A	R	vs_inf_pb_0_23	Data Byte 22
0xE36B	R	vs_inf_pb_0_24	Data Byte 23
0xE36C	R	vs_inf_pb_0_25	Data Byte 24
0xE36D	R	vs_inf_pb_0_26	Data Byte 25
0xE36E	R	vs_inf_pb_0_27	Data Byte 26
0xE36F	R	vs_inf_pb_0_28	Data Byte 27

The Vendor Specific InfoFrame registers are considered valid if the following two conditions are met:

- vs_infoframe_det is 1.
- vs_inf_cksum_err is 0. This condition applies only if always_store_inf is set to 1.

5.10. PACKET REGISTERS

5.10.1. *ISRC Packet Registers*

Table 39 and Table 40 provide lists of the readback registers available for the ISRC packets. Refer to the HDMI 1.4 specifications for a detailed explanation of the ISRC packet fields.

Table 39: ISRC1 Packet Registers

InfoFrame Map Address	R/W	Register Name	Packet Byte No.[1]
0xF2	R/W	isrc1_packet_id[7:0]	Packet Type Value
0xF3	R	isrc1_header1	HB1
0xF4	R	isrc1_header2	HB2
0x8C	R	isrc1_pb_0_1	PB0
0x8D	R	isrc1_pb_0_2	PB1
0x8E	R	isrc1_pb_0_3	PB2
0x8F	R	isrc1_pb_0_4	PB3
0x90	R	isrc1_pb_0_5	PB4
0x91	R	isrc1_pb_0_6	PB5
0x92	R	isrc1_pb_0_7	PB6
0x93	R	isrc1_pb_0_8	PB7
0x94	R	isrc1_pb_0_9	PB8
0x95	R	isrc1_pb_0_10	PB9
0x96	R	isrc1_pb_0_11	PB10
0x97	R	isrc1_pb_0_12	PB11
0x98	R	isrc1_pb_0_13	PB12
0x99	R	isrc1_pb_0_14	PB13
0x9A	R	isrc1_pb_0_15	PB14
0x9B	R	isrc1_pb_0_16	PB15
0x9C	R	isrc1_pb_0_17	PB16
0x9D	R	isrc1_pb_0_18	PB17

InfoFrame Map Address	R/W	Register Name	Packet Byte No.[1]
0x9E	R	isrc1_pb_0_19	PB18
0x9F	R	isrc1_pb_0_20	PB19
0xA0	R	isrc1_pb_0_21	PB20
0xA1	R	isrc1_pb_0_22	PB21
0xA2	R	isrc1_pb_0_23	PB22
0xA3	R	isrc1_pb_0_24	PB23
0xA4	R	isrc1_pb_0_25	PB24
0xA5	R	isrc1_pb_0_26	PB25
0xA6	R	isrc1_pb_0_27	PB26
0xA7	R	isrc1_pb_0_28	PB27

[1]As defined by the HDMI 1.4 specifications

The ISRC1 packet registers are considered valid if the ISRC1 packet edge RAW interrupt is set to 1.

rx_isrc1_pckt_edge_raw, IO Map, *Address 0x1AFB[4] (Read Only)*
This readback indicates the raw status of the ISRC1 packet received signal. Once set this bit remains high until cleared via the corresponding clear bit.

Function

rx_isrc1_pckt_edge_raw	Description
0 (default)	No new ISRC1 packet received
1	ISRC1 packet with new content received

Table 40: ISRC2 Packet Registers

InfoFrame Map Address	R/W	Register Name	Packet Byte No.[1]
0xE3F5	R/W	isrc2_packet_id[7:0]	Packet Type Value
0x E3F6	R	isrc2_header1	HB1
0x E3F7	R	isrc2_header2	HB2
0x E3A8	R	isrc2_pb_0_1	PB0
0x E3A9	R	isrc2_pb_0_2	PB1
0x E3AA	R	isrc2_pb_0_3	PB2
0x E3AB	R	isrc2_pb_0_4	PB3
0x E3AC	R	isrc2_pb_0_5	PB4
0x E3AD	R	isrc2_pb_0_6	PB5
0x E3AE	R	isrc2_pb_0_7	PB6
0x E3AF	R	isrc2_pb_0_8	PB7
0x E3B0	R	isrc2_pb_0_9	PB8
0x E3B1	R	isrc2_pb_0_10	PB9
0x E3B2	R	isrc2_pb_0_11	PB10
0x E3B3	R	isrc2_pb_0_12	PB11
0x E3B4	R	isrc2_pb_0_13	PB12
0x E3B5	R	isrc2_pb_0_14	PB13
0x E3B6	R	isrc2_pb_0_15	PB14
0x E3B7	R	isrc2_pb_0_16	PB15
0x E3B8	R	isrc2_pb_0_17	PB16
0x E3B9	R	isrc2_pb_0_18	PB17
0x E3BA	R	isrc2_pb_0_19	PB18
0x E3BB	R	isrc2_pb_0_20	PB19
0x E3BC	R	isrc2_pb_0_21	PB20
0x E3BD	R	isrc2_pb_0_22	PB21
0x E3BE	R	isrc2_pb_0_23	PB22

InfoFrame Map Address	R/W	Register Name	Packet Byte No.[1]
0x E3BF	R	isrc2_pb_0_24	PB23
0x E3C0	R	isrc2_pb_0_25	PB24
0x E3C1	R	isrc2_pb_0_26	PB25
0x E3C2	R	isrc2_pb_0_27	PB26
0x E3C3	R	isrc2_pb_0_28	PB27

[1]As defined by the HDMI 1.4 specifications

The ISRC2 packet registers are considered valid if, and only if rx_isrc2_pckt_edge_raw is set to 1.

rx_isrc2_pckt_edge_raw, IO Map, *Address 0x1AFB[5] (Read Only)*
This readback indicates the raw status of the ISRC2 packet received signal. Once set this bit remains high until cleared via the corresponding clear bit.

Function

rx_isrc2_pckt_edge_raw	Description
0 (default)	No new ISRC2 packet received
1	ISRC2 packet with new content received

5.10.2. *Gamut Metadata Packets*

Refer to the HDMI 1.3/1.4 specifications for a detailed explanation of the Gamut Metadata packet fields.

Table 41: Gamut Metadata Packet Registers

HDMI Map Address	R/W	Register Name	Packet Byte No.[1]
0xE3F8	R/W	gamut_packet_id[7:0]	Packet Type Value
0xE3F9	R	gamut_header1	HB1
0xE3FA	R	gamut_header2	HB2
0xE3C4	R	gamut_mdata_pb_0_1	PB0
0xE3C5	R	gamut_mdata_pb_0_2	PB1
0xE3C6	R	gamut_mdata_pb_0_3	PB2
0xE3C7	R	gamut_mdata_pb_0_4	PB3
0xE3C8	R	gamut_mdata_pb_0_5	PB4
0xE3C9	R	gamut_mdata_pb_0_6	PB5
0xE3CA	R	gamut_mdata_pb_0_7	PB6
0xE3CB	R	gamut_mdata_pb_0_8	PB7
0xE3CC	R	gamut_mdata_pb_0_9	PB8
0xE3CD	R	gamut_mdata_pb_0_10	PB9
0xE3CE	R	gamut_mdata_pb_0_11	PB10
0xE3CF	R	gamut_mdata_pb_0_12	PB11
0xE3D0	R	gamut_mdata_pb_0_13	PB12
0xE3D1	R	gamut_mdata_pb_0_14	PB13
0xE3D2	R	gamut_mdata_pb_0_15	PB14
0xE3D3	R	gamut_mdata_pb_0_16	PB15
0xE3D4	R	gamut_mdata_pb_0_17	PB16
0xE3D5	R	gamut_mdata_pb_0_18	PB17
0xE3D6	R	gamut_mdata_pb_0_19	PB18
0xE3D7	R	gamut_mdata_pb_0_20	PB19
0xE3D8	R	gamut_mdata_pb_0_21	PB20
0xE3D9	R	gamut_mdata_pb_0_22	PB21
0xE3DA	R	gamut_mdata_pb_0_23	PB22
0xE3DB	R	gamut_mdata_pb_0_24	PB23
0xE3DC	R	gamut_mdata_pb_0_25	PB24

HDMI Map Address	R/W	Register Name	Packet Byte No.[1]
0xE3DD	R	gamut_mdata_pb_0_26	PB25
0xE3DE	R	gamut_mdata_pb_0_27	PB26
0xE3DF	R	gamut_mdata_pb_0_28	PB27

[1]As defined by the HDMI 1.3 specifications

The Gamut Metadata packet registers are considered valid if pkt_det_gamut is set to 1 (refer to Section 8.2.2 for more details).

gamut_irq_next_field, HDMI RX Map, *Address 0xE250[4]*
This bit is used to set the NEW_GAMUT_MDATA_RAW interrupt to detect when the new contents are applicable to next field or to indicate that the Gamut packet is new. This is done using header information of the gamut packet.

Function

gamut_irq_next_field	Description
0 (default)	Interrupt flag indicates that Gamut packet is new
1	Interrupt flag indicates that Gamut packet is to be applied next field

5.11. CUSTOMIZING PACKET/INFOFRAME STORAGE REGISTERS

The packet type value of each set of packet and InfoFrame registers in the Serial Video Rx InfoFrame Map is programmable. This allows the user to configure the ADV8005 to store the payload data of any packet and InfoFrames sent by the transmitter connected on the Serial Video Rx port.

Note: Writing to any of the following packet ID registers also clears the corresponding InfoFrame/packet detection bit.

avi_packet_id[7:0], HDMI RX Infoframe Map, *Address 0xE3E0[7:0]*
This control is used to set the AVI InfoFrame ID

Function

avi_packet_id[7:0]	Description
0xxxxxxx	Packet type value of packet stored in InfoFrame Map, Address 0x00 to 0x1B
1xxxxxxx	Packet type value of InfoFrame stored in InfoFrame Map, Address 0x00 to 0x1B

spd_packet_id[7:0], HDMI RX Infoframe Map, *Address 0xE3E6[7:0]*
This control is used to set the Source Product Descriptor InfoFrame ID

Function

spd_packet_id[7:0]	Description
0xxxxxxx	Packet type value of packet stored in InfoFrame Map, Address 0x2A to 0x45
1xxxxxxx	Packet type value of InfoFrame stored in InfoFrame Map, Address 0x2A to 0x45

aud_packet_id[7:0], HDMI RX Infoframe Map, *Address 0xE3E3[7:0]*
This control is used to set the Audio InfoFrame ID

Function

aud_packet_id[7:0]	Description
0xxxxxxx	Packet type value of packet stored in InfoFrame Map, Address 0x1C to 0x29
1xxxxxxx	Packet type value of InfoFrame stored in InfoFrame Map, Address 0x1C to 0x29

ms_packet_id[7:0], HDMI RX Infoframe Map, *Address 0xE3E9[7:0]*
This control is used to set the MPEG Source InfoFrame ID

Function

ms_packet_id[7:0]	Description
0xxxxxxx	Packet type value of packet stored in InfoFrame Map, Address 0x46 to 0x53
1xxxxxxx	Packet type value of InfoFrame stored in InfoFrame Map, Address 0x46 to 0x53

vs_packet_id[7:0], HDMI RX Infoframe Map, *Address 0xE3EC[7:0]*

This control is used to set the Vendor Specific InfoFrame ID

Function

vs_packet_id[7:0]	Description
0xxxxxxx	Packet type value of packet stored in InfoFrame Map, Address 0x54 to 0x6F
1xxxxxxx	Packet type value of packet stored in InfoFrame Map, Address 0x54 to 0x6F

acp_packet_id[7:0], HDMI RX Infoframe Map, *Address 0xE3EF[7:0]*

This control is used to set the ACP Packet ID

Function

acp_packet_id[7:0]	Description
0xxxxxxx	Packet type value of packet stored in InfoFrame Map, Address 0x70 to 0x8B
1xxxxxxx	Packet type value of InfoFrame stored in InfoFrame Map, Address 0x70 to 0x8B

isrc1_packet_id[7:0], HDMI RX Infoframe Map, *Address 0xE3F2[7:0]*

This control is used to set the ISRC1 Packet ID

Function

isrc1_packet_id[7:0]	Description
0xxxxxxx	Packet type value of packet stored in InfoFrame Map, Address 0x8C to 0xA7
1xxxxxxx	Packet type value of InfoFrame stored in InfoFrame Map, Address 0x8C to 0xA7

isrc2_packet_id[7:0], HDMI RX Infoframe Map, *Address 0xE3F5[7:0]*

This control is used to set the ISRC2 Packet ID

Function

isrc2_packet_id[7:0]	Description
0xxxxxxx	Packet type value of packet stored in InfoFrame Map, Address 0xA8 to 0xC3
1xxxxxxx	Packet type value of InfoFrame stored in InfoFrame Map, Address 0xA8 to 0xC3

gamut_packet_id[7:0], HDMI RX Infoframe Map, *Address 0xE3F8[7:0]*

This control is used to set the Gamut Metadata Packet ID

Function

gamut_packet_id[7:0]	Description
0xxxxxxx	Packet type value of packet stored in InfoFrame Map, Address 0xC4 to 0xDF
1xxxxxxx	Packet type value of InfoFrame stored in InfoFrame Map, Address 0xC4 to 0xDF

Note: The packet type values and corresponding packets should not be programmed in the packet type values registers. The general control packet (0x03) is always processed internally and cannot be stored in the packet/InfoFrame registers in the InfoFrame Map.

5.12. HDMI SECTION RESET STRATEGY

The following reset strategy is implemented for the HDMI section:

- Global chip reset – This means the ADV8005 Serial Video Rx core can be reset using the rx_reset or main_reset. A global chip reset is triggered by asserting the RESET pin to a low level. The HDMI section is reset when a global reset is triggered.
- Loss of TMDS clock or 5 V signal reset – A loss of TMDS clock or 5 V signal to the Serial Video Rx resets the entire Serial Video Rx section. The loss of a 5 V signal condition is discarded if dis_cable_det_rst is set high.
- DVI mode reset – The packet processing block, including InfoFrame memory, is held in reset when the Serial Video Rx processes a DVI stream.

6. HDMI TRANSMITTER

The HDMI transmitters on the ADV8005 are capable of outputting video data at up to 3 GHz and support 3D video output, ARC (common mode only), and audio output.

The dual transmitter variants of ADV8005 are the following:

- ADV8005KBCZ-8A
- ADV8005KBCZ-8N
- ADV8005KBCZ-8C

The single transmitter variant of the ADV8005 is the ADV8005KBCZ-8B.

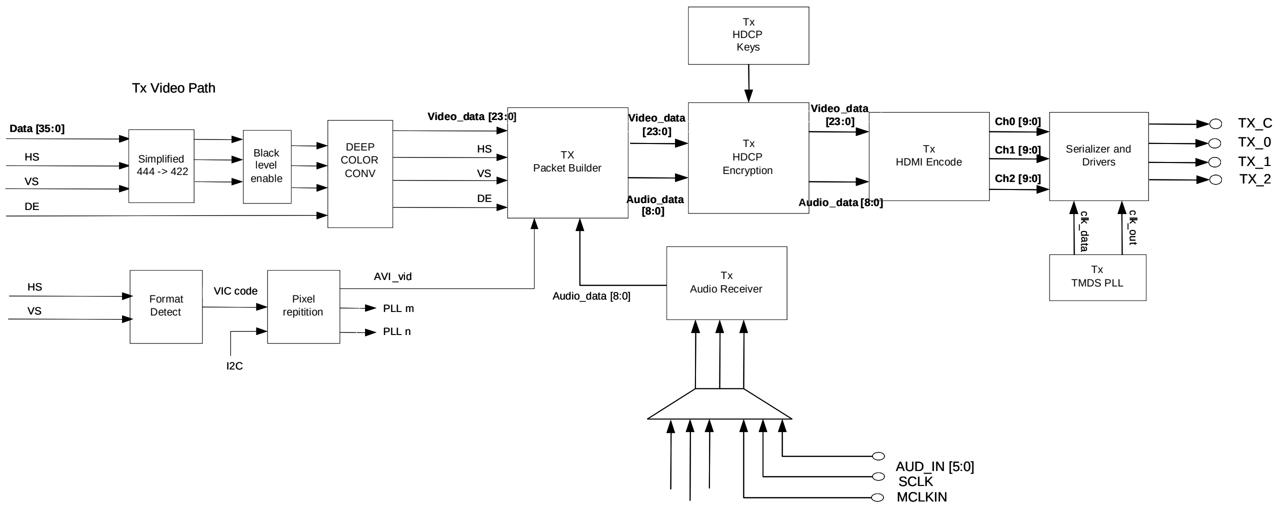


Figure 95: Functional Block Diagram of HDMI Tx Core

As the two ADV8005 HDMI transmitters can be configured independently, there are separate register maps for both the HDMI Tx1 and HDMI Tx2. The addresses for these register maps are listed in Table 42.

Table 42: HDMI Transmitter Memory Addresses

Register Map	Register Map Address
HDMI Tx1 Main Map	0xEC00 – 0xECFF
HDMI Tx1 EDID Map	0xEE00 – 0xEEFF
HDMI Tx1 UDP Map	0xF200 – 0xF2FF
HDMI Tx1 Test Map	0xF300 – 0xF3FF
HDMI Tx2 Main Map	0xF400 – 0xF4FF
HDMI Tx2 EDID Map	0xF600 – 0xF6FF
HDMI Tx2 UDP Map	0xFA00 – 0xFAFF
HDMI Tx2 Test Map	0xFB00 – 0xFBFF

While this chapter only references one instance of each HDMI Tx Map, the controls referenced are valid for both HDMI Tx1 and HDMI Tx2 register maps. The same register bits and controls as per Table 42 apply for both transmitters.

6.1. GENERAL CONTROLS

To operate the HDMI Tx core, it is necessary to monitor the Hot Plug Detect (HPD) signal from the downstream sink and power up the Tx core after the appropriate HPD becomes high. To power up the Tx core, system_pd must be programmed to 0 when the HPD_TX1 pin is high. The status of the HPD_TX1 pin is provided via hpd_state.

Some registers cannot be written to when the signal on the HPD_TXx input pin is low. When the level on the HPD_TX1 pin goes from high to low, some registers will be reset to their default value.

The best method to determine when the level of the signal on the HPD_TXx pin is high is to use the interrupt system. An interrupt can be enabled to notify level change on the HPD_TXx pin (refer to section 8 for more details regarding the ADV8005 interrupts).

The ADV8005 also features a rx_sense_state status bit which can be used to detect the presence of TMDS clock terminations from the sink. If the ADV8005 detects a voltage level higher than 1.8 V on the clock lines of its TMDS output port, rx_sense_int is triggered and rx_sense_state is set to 1.

The detection of TMDS clock terminations from downstream sink devices is useful to delay powering up the transmitter sections until the downstream sink devices are actually ready to receive signals. A typical implementation for a sink is to tie the transmitter 5 V power signal to HPD through a series resistor. In this case, the ADV8005 will detect a high level on HPD_TX1 (HPD_TX2 for HDMI Tx 2) regardless of whether or not the downstream sink is powered on and ready to receive a TMDS stream. For this reason, it is best to wait for both the rx_sense_state and hpd_state to be high before powering up the Tx core when trying to achieve minimum power consumption.

system_pd, TX2 Main Map, *Address 0xF441[6]*
This bit is used to power down the TX.

Function

system_pd	Description
0	Normal operation
1 (default)	Power down TX

hpd_state, TX2 Main Map, *Address 0xF442[6] (Read Only)*
This bit is used to readback the state of the hot plug detect.

Function

hpd_state	Description
0 (default)	Hot Plug Detect inactive (low)
1	Hot Plug active (high)

hpd_override[1:0], TX2 Main Map, *Address 0xF49F[5:4]*
This signal is used to select the source of the internal HPD signal.

Function

hpd_override[1:0]	Description
00 (default)	HPD from HPD pin and CDC HPD
01	HPD from CDC HPD
10	HPD from HPD pin
11	HPD set to 1

rx_sense_state, TX2 Main Map, *Address 0xF442[5] (Read Only)*
This bit is used to readback the state of the Rx sense.

Function

rx_sense_state	Description
0 (default)	HDMI clock termination not detected
1	HDMI clock termination detected

rx_sense_pd, TX2 Main Map, *Address 0xF4E6[5]*
This bit is used to enable the termination sense power down.

Function

rx_sense_pd	Description
0 (default)	Termination Sense Monitoring Enabled
1	Termination Sense Monitoring Disabled

Note: rx_sense_pd should not be applied during the configuration of the HDMI Tx as it disables an oscillator required to complete the configuration of the TMDS output clock channel. It is recommended to use rx_sense_pd when the HDMI Tx has been completely configured.

6.2. RESET STRATEGY

The HDMI Tx, and subsections of it, can be reset in a number of ways. Table 43 and Table 44 describe how each of the HDMI Tx maps are reset in response to a number of different events.

Table 43: HDMI Tx Main Map Reset Strategy

	IO Map	IO Map	IO Map	Tx Main Map	Tx Main Map	Event	Event
	tx1_reset	0x1AFC[7]	main_reset	system_pd	0xEC98[4] 0xF498[4]	Tx Hot Plug	Reset Pin
0x00 – 0x91	Reset	Reset	Reset			Reset	Reset
0x92 – 0x97	Reset		Reset				Reset
0x98 – 0xAE	Reset		Reset				Reset
0xAF – 0xBD	Reset		Reset	Reset			Reset
0xBE – 0xCF	Reset		Reset	Reset			Reset
0xD0 – 0xFE	Reset		Reset				Reset

Table 44: HDMI Tx Packet Map Reset Strategy

	IO Map	IO Map	IO Map	Tx Main Map	Tx Main Map	Event	Event
	tx1_reset	0x1AFC[7]	main_reset	system_pd	0xEC98[4] 0xF498[4]	Tx Hot Plug	Reset Pin
0x00 – 0xFF	Reset	Reset	Reset			Reset	Reset

6.3. HDMI DVI SELECTION

The HDMI Tx core supports the transmission of both HDMI and DVI streams. The type of stream the ADV8005 transmits is set via hdmi_dvi_sel_en. In DVI transmission mode, no packets will be sent and all registers relating to packets and InfoFrames will be disregarded. The current transmission mode can be confirmed by reading hdmi_dvi_sel.

hdmi_dvi_sel_en, TX2 Main Map, *Address 0xF4AF[2]*
This bit is used to enable the output mode control.

Function

hdmi_dvi_sel_en	Description
0	Automatic
1 (default)	Output mode set by hdmi_dvi_sel

hdmi_dvi_sel, TX2 Main Map, *Address 0xF4AF[1]*
This bit is used to control the output mode - DVI or HDMI.

Function

hdmi_dvi_sel	Description
0 (default)	DVI
1	HDMI

6.4. AV MUTE

The AV mute status is sent to the downstream sink through the general control packet. One purpose of the AV mute is to alert the sink of a change in the TMDS clock so the sink can mute audio and video while the TMDS clock it receives is unstable. Setting AV mute also pauses HDCP encryption, so the HDCP link between the HDMI Tx and the sink is maintained while the TMDS clock is not stable. Note that AV mute is not sufficient as a means to hide protected content because the content is still sent even when AV mute is enabled.

To use AV mute:

- Enable the GCP by setting gc_pkt_en to 1
- To set AV mute, clear clear_avmute (that is, clear_avmute = 0) and set set_avmute (that is, set_avmute = 1)
- To clear AV mute, clear set_avmute (that is, set_avmute = 0) and set clear_avmute (clear_avmute = 1)

Note that setting both set_avmute and clear_avmute is not a valid configuration.

set_avmute, TX2 Main Map, *Address 0xF44B[6]*
This bit is used to control the SET_AVMUTE signal.

Function

set_avmute	Description
0 (default)	Set SET_AVMUTE to 0
1	Set SET_AVMUTE to 1

clear_avmute, TX2 Main Map, *Address 0xF44B[7]*
This bit is used to control the CLEAR_AVMUTE signal.

Function

clear_avmute	Description
0 (default)	Set CLEAR_AVMUTE to 0
1	Set CLEAR_AVMUTE to 1

6.5. SOURCE PRODUCT DESCRIPTION INFOFRAME

The Source Product Description (SPD) InfoFrame contains the vendor name and product description. The transmission of SPD InfoFrames is enabled by setting spd_pkt_en to 1. When this bit is set, the HDMI Tx1 section transmits one SPD packet once every two video fields.

An application of this packet is to allow the sink to display the source information using an OSD. This information is in a 7-bit ASCII format. Refer to CEA 861 specification for more detail.

spd_pkt_en, TX2 Main Map, *Address 0xF440[6]*
This bit is used to enable the Source Product Descriptor InfoFrame.

Function

spd_pkt_en	Description
0 (default)	Disabled
1	Enabled

Table 45: SPD InfoFrame Configuration Register

Packet Map Address	Access Type	Register Name	Default Value	Byte Name
0xF200	R/W	spd_hb0[7:0]	0b00000000	Header Byte 0
0xF201	R/W	spd_hb1[7:0]	0b00000000	Header Byte 1
0xF202	R/W	spd_hb2[7:0]	0b00000000	Header Byte 2
0xF203	R/W	spd_pb0[7:0]	0b00000000	Data Byte 0
0xF204	R/W	spd_pb1[7:0]	0b00000000	Data Byte 1
0xF205	R/W	spd_pb2[7:0]	0b00000000	Data Byte 2
0xF206	R/W	spd_pb3[7:0]	0b00000000	Data Byte 3
0xF207	R/W	spd_pb4[7:0]	0b00000000	Data Byte 4
0xF208	R/W	spd_pb5[7:0]	0b00000000	Data Byte 5
0xF209	R/W	spd_pb6[7:0]	0b00000000	Data Byte 6
0xF20A	R/W	spd_pb7[7:0]	0b00000000	Data Byte 7
0xF20B	R/W	spd_pb8[7:0]	0b00000000	Data Byte 8
0xF20C	R/W	spd_pb9[7:0]	0b00000000	Data Byte 9
0xF20D	R/W	spd_pb10[7:0]	0b00000000	Data Byte 10
0xF20E	R/W	spd_pb11[7:0]	0b00000000	Data Byte 11
0xF20F	R/W	spd_pb12[7:0]	0b00000000	Data Byte 12
0xF210	R/W	spd_pb13[7:0]	0b00000000	Data Byte 13
0xF211	R/W	spd_pb14[7:0]	0b00000000	Data Byte 14
0xF212	R/W	spd_pb15[7:0]	0b00000000	Data Byte 15
0xF213	R/W	spd_pb16[7:0]	0b00000000	Data Byte 16
0xF214	R/W	spd_pb17[7:0]	0b00000000	Data Byte 17
0xF215	R/W	spd_pb18[7:0]	0b00000000	Data Byte 18

Packet Map Address	Access Type	Register Name	Default Value	Byte Name
0xF216	R/W	spd_pb19[7:0]	0b00000000	Data Byte 19
0xF217	R/W	spd_pb20[7:0]	0b00000000	Data Byte 20
0xF218	R/W	spd_pb21[7:0]	0b00000000	Data Byte 21
0xF219	R/W	spd_pb22[7:0]	0b00000000	Data Byte 22
0xF21A	R/W	spd_pb23[7:0]	0b00000000	Data Byte 23
0xF21B	R/W	spd_pb24[7:0]	0b00000000	Data Byte 24
0xF21C	R/W	spd_pb25[7:0]	0b00000000	Data Byte 25
0xF21D	R/W	spd_pb26[7:0]	0b00000000	Data Byte 26
0xF21E	R/W	spd_pb27[7:0]	0b00000000	Data Byte 27

6.6. SPARE PACKETS AND VSI SUPPORT

The user may configure the ADV8005 to send any type of packets or InfoFrames via the spare packets controls and associated configuration registers. The ADV8005 features four such spare packets that can be enabled via the spare_pkt0_en, spare_pkt1_en, spare_pkt3_en, and spare_pkt4_en controls bits. When a spare packet is enabled, the Tx transmits one of these enabled spare packets once every two video fields. No control exists over the specific timing when these are sent; however, it is always before the leading edge of VSYNC. These spare packets allow the ADV8005 to support the transmission of three Vendor Specific InfoFrames (VSI) as follows: VSI-Video, VSI-AUDIO and VSI-HDMI.

spare_pkt0_en, TX2 Main Map, *Address 0xF440[0]*
This bit is used to enable the Spare Packet 1.

Function

spare_pkt0_en	Description
0 (default)	Disabled
1	Enabled

spare_pkt1_en, TX2 Main Map, *Address 0xF440[1]*
This bit is used to enable the Spare Packet 2.

Function

spare_pkt1_en	Description
0 (default)	Disabled
1	Enabled

spare_pkt3_en, TX2 Test Map, *Address 0xFBBF[2]*
This bit is used to enable the Spare Packet 3.

Function

spare_pkt3_en	Description
0 (default)	Disabled
1	Enabled

spare_pkt4_en, TX2 Test Map, *Address 0xFBBF[1]*
This bit is used to enable the Spare Packet 4.

Function

spare_pkt4_en	Description
0 (default)	Disabled
1	Enabled

Table 46: Spare Packet 1 Configuration Register

Packet Map Address	Access Type	Register Name	Default Value	Byte Name
0xF2C0	R/W	spare1_hb0[7:0]	0b00000000	Header Byte 0
0xF2C1	R/W	spare1_hb1[7:0]	0b00000000	Header Byte 1
0xF202	R/W	spare1_hb2[7:0]	0b00000000	Header Byte 2
0xF2C3	R/W	spare1_pb0[7:0]	0b00000000	Data Byte 0
0xF2C4	R/W	spare1_pb1[7:0]	0b00000000	Data Byte 1

Packet Map Address	Access Type	Register Name	Default Value	Byte Name
0xF2C5	R/W	spare1_pb2[7:0]	0b00000000	Data Byte 2
0xF2C6	R/W	spare1_pb3[7:0]	0b00000000	Data Byte 3
0xF2C7	R/W	spare1_pb4[7:0]	0b00000000	Data Byte 4
0xF2C8	R/W	spare1_pb5[7:0]	0b00000000	Data Byte 5
0xF2C9	R/W	spare1_pb6[7:0]	0b00000000	Data Byte 6
0xF2CA	R/W	spare1_pb7[7:0]	0b00000000	Data Byte 7
0xF2CB	R/W	spare1_pb8[7:0]	0b00000000	Data Byte 8
0xF2CC	R/W	spare1_pb9[7:0]	0b00000000	Data Byte 9
0xF2CD	R/W	spare1_pb10[7:0]	0b00000000	Data Byte 10
0xF2CE	R/W	spare1_pb11[7:0]	0b00000000	Data Byte 11
0xF2CF	R/W	spare1_pb12[7:0]	0b00000000	Data Byte 12
0xF2D0	R/W	spare1_pb13[7:0]	0b00000000	Data Byte 13
0xF2D1	R/W	spare1_pb14[7:0]	0b00000000	Data Byte 14
0xF2D2	R/W	spare1_pb15[7:0]	0b00000000	Data Byte 15
0xF2D3	R/W	spare1_pb16[7:0]	0b00000000	Data Byte 16
0xF2D4	R/W	spare1_pb17[7:0]	0b00000000	Data Byte 17
0xF2D5	R/W	spare1_pb18[7:0]	0b00000000	Data Byte 18
0xF2D6	R/W	spare1_pb19[7:0]	0b00000000	Data Byte 19
0xF2D7	R/W	spare1_pb20[7:0]	0b00000000	Data Byte 20
0xF2D8	R/W	spare1_pb21[7:0]	0b00000000	Data Byte 21
0xF2D9	R/W	spare1_pb22[7:0]	0b00000000	Data Byte 22
0xF2DA	R/W	spare1_pb23[7:0]	0b00000000	Data Byte 23
0xF2DB	R/W	spare1_pb24[7:0]	0b00000000	Data Byte 24
0xF2DC	R/W	spare1_pb25[7:0]	0b00000000	Data Byte 25
0xF2DD	R/W	spare1_pb26[7:0]	0b00000000	Data Byte 26
0xF2DE	R/W	spare1_pb27[7:0]	0b00000000	Data Byte 27

Table 47: Spare Packet 2 Configuration Register

Packet Map Address	Access Type	Register Name	Default Value	Byte Name
0xF2E0	R/W	spare2_hb0[7:0]	0b00000000	Header Byte 0
0xF2E1	R/W	spare2_hb1[7:0]	0b00000000	Header Byte 1
0xF2E2	R/W	spare2_hb2[7:0]	0b00000000	Header Byte 2
0xF2E3	R/W	spare2_pb0[7:0]	0b00000000	Data Byte 0
0xF2E4	R/W	spare2_pb1[7:0]	0b00000000	Data Byte 1
0xF2E5	R/W	spare2_pb2[7:0]	0b00000000	Data Byte 2
0xF2E6	R/W	spare2_pb3[7:0]	0b00000000	Data Byte 3
0xF2E7	R/W	spare2_pb4[7:0]	0b00000000	Data Byte 4
0xF2E8	R/W	spare2_pb5[7:0]	0b00000000	Data Byte 5
0xF2E9	R/W	spare2_pb6[7:0]	0b00000000	Data Byte 6
0xF2EA	R/W	spare2_pb7[7:0]	0b00000000	Data Byte 7
0xF2EB	R/W	spare2_pb8[7:0]	0b00000000	Data Byte 8
0xF2EC	R/W	spare2_pb9[7:0]	0b00000000	Data Byte 9
0xF2ED	R/W	spare2_pb10[7:0]	0b00000000	Data Byte 10
0xF2EE	R/W	spare2_pb11[7:0]	0b00000000	Data Byte 11
0xF2FF	R/W	spare2_pb12[7:0]	0b00000000	Data Byte 12
0xF2F0	R/W	spare2_pb13[7:0]	0b00000000	Data Byte 13
0xF2F1	R/W	spare2_pb14[7:0]	0b00000000	Data Byte 14
0xF2F2	R/W	spare2_pb15[7:0]	0b00000000	Data Byte 15
0xF2F3	R/W	spare2_pb16[7:0]	0b00000000	Data Byte 16
0xF2F4	R/W	spare2_pb17[7:0]	0b00000000	Data Byte 17
0xF2F5	R/W	spare2_pb18[7:0]	0b00000000	Data Byte 18

Packet Map Address	Access Type	Register Name	Default Value	Byte Name
0xF2F6	R/W	spare2_pb19[7:0]	0b00000000	Data Byte 19
0xF2F7	R/W	spare2_pb20[7:0]	0b00000000	Data Byte 20
0xF2F8	R/W	spare2_pb21[7:0]	0b00000000	Data Byte 21
0xF2F9	R/W	spare2_pb22[7:0]	0b00000000	Data Byte 22
0xF2FA	R/W	spare2_pb23[7:0]	0b00000000	Data Byte 23
0xF2FB	R/W	spare2_pb24[7:0]	0b00000000	Data Byte 24
0xF2FC	R/W	spare2_pb25[7:0]	0b00000000	Data Byte 25
0xF2FD	R/W	spare2_pb26[7:0]	0b00000000	Data Byte 26
0xF2FE	R/W	spare2_pb27[7:0]	0b00000000	Data Byte 27

Table 48: Spare Packet 3 Configuration Register

Test Map Address	Access Type	Register Name	Default Value	Byte Name
0xF3C0	R/W	spare3_header0[7:0]	0b00000000	Header Byte 0
0xF3C1	R/W	spare3_header1[7:0]	0b00000000	Header Byte 1
0xF3C2	R/W	spare3_header2[7:0]	0b00000000	Header Byte 2
0xF3C3	R/W	spare3_byte0[7:0]	0b00000000	Data Byte 0
0xF3C4	R/W	spare3_byte1[7:0]	0b00000000	Data Byte 1
0xF3C5	R/W	spare3_byte2[7:0]	0b00000000	Data Byte 2
0xF3C6	R/W	spare3_byte3[7:0]	0b00000000	Data Byte 3
0xF3C7	R/W	spare3_byte4[7:0]	0b00000000	Data Byte 4
0xF3C8	R/W	spare3_byte5[7:0]	0b00000000	Data Byte 5
0xF3C9	R/W	spare3_byte6[7:0]	0b00000000	Data Byte 6
0xF3CA	R/W	spare3_byte7[7:0]	0b00000000	Data Byte 7
0xF3CB	R/W	spare3_byte8[7:0]	0b00000000	Data Byte 8
0xF3CC	R/W	spare3_byte9[7:0]	0b00000000	Data Byte 9
0xF3CD	R/W	spare3_byte10[7:0]	0b00000000	Data Byte 10
0xF3CE	R/W	spare3_byte11[7:0]	0b00000000	Data Byte 11
0xF3CF	R/W	spare3_byte12[7:0]	0b00000000	Data Byte 12
0xF3D0	R/W	spare3_byte13[7:0]	0b00000000	Data Byte 13
0xF3D1	R/W	spare3_byte14[7:0]	0b00000000	Data Byte 14
0xF3D2	R/W	spare3_byte15[7:0]	0b00000000	Data Byte 15
0xF3D3	R/W	spare3_byte16[7:0]	0b00000000	Data Byte 16
0xF3D4	R/W	spare3_byte17[7:0]	0b00000000	Data Byte 17
0xF3D5	R/W	spare3_byte18[7:0]	0b00000000	Data Byte 18
0xF3D6	R/W	spare3_byte19[7:0]	0b00000000	Data Byte 19
0xF3D7	R/W	spare3_byte20[7:0]	0b00000000	Data Byte 20
0xF3D8	R/W	spare3_byte21[7:0]	0b00000000	Data Byte 21
0xF3D9	R/W	spare3_byte22[7:0]	0b00000000	Data Byte 22
0xF3DA	R/W	spare3_byte23[7:0]	0b00000000	Data Byte 23
0xF3DB	R/W	spare3_byte24[7:0]	0b00000000	Data Byte 24
0xF3DC	R/W	spare3_byte25[7:0]	0b00000000	Data Byte 25
0xF3DD	R/W	spare3_byte26[7:0]	0b00000000	Data Byte 26
0xF3DE	R/W	spare3_byte27[7:0]	0b00000000	Data Byte 27

Table 49: Spare Packet 4 Configuration Register

Packet Map Address	Access Type	Register Name	Default Value	Byte Name
0xF3E0	R/W	spare4_header0[7:0]	0b00000000	Header Byte 0
0xF3E1	R/W	spare4_header1[7:0]	0b00000000	Header Byte 1
0xF3E2	R/W	spare4_header2[7:0]	0b00000000	Header Byte 2
0xF3E3	R/W	spare4_pb0[7:0]	0b00000000	Data Byte 0
0xF3E4	R/W	spare4_pb1[7:0]	0b00000000	Data Byte 1
0xF3E5	R/W	spare4_pb2[7:0]	0b00000000	Data Byte 2

Packet Map Address	Access Type	Register Name	Default Value	Byte Name
0xF3E6	R/W	spare4_pb3[7:0]	0b00000000	Data Byte 3
0xF3E7	R/W	spare4_pb4[7:0]	0b00000000	Data Byte 4
0xF3E8	R/W	spare4_pb5[7:0]	0b00000000	Data Byte 5
0xF3E9	R/W	spare4_pb6[7:0]	0b00000000	Data Byte 6
0xF3EA	R/W	spare4_pb7[7:0]	0b00000000	Data Byte 7
0xF3EB	R/W	spare4_pb8[7:0]	0b00000000	Data Byte 8
0xF3EC	R/W	spare4_pb9[7:0]	0b00000000	Data Byte 9
0xF3ED	R/W	spare4_pb10[7:0]	0b00000000	Data Byte 10
0xF3EE	R/W	spare4_pb11[7:0]	0b00000000	Data Byte 11
0xF3EF	R/W	spare4_pb12[7:0]	0b00000000	Data Byte 12
0xF3F0	R/W	spare4_pb13[7:0]	0b00000000	Data Byte 13
0xF3F1	R/W	spare4_pb14[7:0]	0b00000000	Data Byte 14
0xF3F2	R/W	spare4_pb15[7:0]	0b00000000	Data Byte 15
0xF3F3	R/W	spare4_pb16[7:0]	0b00000000	Data Byte 16
0xF3F4	R/W	spare4_pb17[7:0]	0b00000000	Data Byte 17
0xF3F5	R/W	spare4_pb18[7:0]	0b00000000	Data Byte 18
0xF3F6	R/W	spare4_pb19[7:0]	0b00000000	Data Byte 19
0xF3F7	R/W	spare4_pb20[7:0]	0b00000000	Data Byte 20
0xF3F8	R/W	spare4_pb21[7:0]	0b00000000	Data Byte 21
0xF3F9	R/W	spare4_pb22[7:0]	0b00000000	Data Byte 22
0xF3FA	R/W	spare4_pb23[7:0]	0b00000000	Data Byte 23
0xF3FB	R/W	spare4_pb24[7:0]	0b00000000	Data Byte 24
0xF3FC	R/W	spare4_pb25[7:0]	0b00000000	Data Byte 25
0xF3FD	R/W	spare4_pb26[7:0]	0b00000000	Data Byte 26
0xF3FE	R/W	spare4_pb27[7:0]	0b00000000	Data Byte 27

6.7. SYSTEM MONITORING

6.7.1. *General Status and Interrupts*

The ADV8005 utilizes both interrupts and status bits to indicate the status of internal operations and errors in the Tx core. These interrupt and status are listed in Table 50, Table 51, and Table 52. Refer to Section 8.4 for details on the use of Tx interrupts.

Table 50: HDMI Tx Interrupt Bits in HDMI Tx Main Map Register 0xEC96

Bit Name	Bit Position	Description
hdcp_authenticated_int	1 (Second LSB)	When set to 1 it indicates that HDCP/EDID state machine transitioned from state 3 to state 4. Once set, it remains high until it is cleared by setting it to 1.
edid_ready_int	2	When set to 1 it indicates that EDID has been read from Rx and is available in Packet Map. Once set, it remains high until it is cleared by setting it to 1.
vsync_int	5	When set to 1 it indicates that leading edge detected on VSync input to Tx core. Once set, it remains high until it is cleared by setting it to 1.
rx_sense_int	6	When set to 1 it indicates that TMDS clock lines voltage has crossed 1.8 V from high to low or low to high. Once set, it remains high until it is cleared by setting it to 1.
hpd_int	7	When set to 1 it indicates that transition for high to low or low to high was detected on input HPD signal. Once set, it remains high until it is cleared by setting it to 1.

Table 51: HDMI Tx Interrupt Bits in Main Map Register 0xEC97

Bit Name	Bit Position	Description
bksv_flag_int	6	When set to 1 it indicates that the KSVs from the downstream sink have been read and available in the Memory Map. Once set, it remains high until it is cleared by setting it to 1.
hdcp_error_int	7	When set to 1 it indicates that the HDCP/EDID controller has reported an error. This error is available in HDCP_CONTROLLER_ERROR. Once set, it remains high until it is cleared by setting it to 1.

Table 52: Status Bits in Main Map Register 0xEC42

Bit Name	Bit Position	Description
hpd_state	6	See description for hpd_state on page 198
rx_sense_state	5	See description for rx_sense_state on page 198

6.7.2. *VSYNC Interrupt*

The time taken for the VSYNC interrupt to trigger depends on the processing done internally, such as 422 to 444 conversion or CSC adjustments. The table below details the typical time taken for VSYNC interrupts to trigger after the VSYNC has been received at the TTL input. For these measurements, the video was routed routed from the TTL input directly to the TX and not routed through the VSP. If the data is routed through the 422 to 444 block, an extra delay of 2 pixel clock cycles can be expected. If the CSC block is also included, an additional delay of 9 clock periods can be expected.

Table 53 Typical times for VSYNC interrupt to Trigger using the TTL digital input Port

Video Format	Pixel Clock Frequency	Typical VSYNC Interrupt Delay
4K2K30	148 MHz	2 µs
1080P60	148 MHz	2 µs
480P60	27 MHz	4 µs
480i60	13.5 MHz	5 µs

6.8. EDID/HDCP CONTROLLER STATUS

The Tx core features an EDID/HDCP controller which handles EDID extraction from the downstream sink. This EDID/HDCP controller also handles HDCP authentication with downstream sink. The tasks that the Tx EDID/HDCP controller performs are described in Section 6.12 and Section 6.13.

The current state of the Tx EDID/HDCP controller can be read from the hdcp_controller_state[3:0] status field.

hdcp_controller_state[3:0], TX2 Main Map, *Address 0xF4C8[3:0] (Read Only)*
This signal is used to readback the state of the EDID/HDCP controller.

Function

hdcp_controller_state[3:0]	Description
0000 (default)	In Reset (No Hot Plug Detected)
0001	Reading EDID
0010	In Idle state (Waiting for HDCP Request)
0011	Initializing HDCP
0100	HDCP enabled
0101	Initializing HDCP Repeater
0110 - 1111	Reserved

6.9. EDID/HDCP CONTROLLER ERROR CODES

If an HDCP authentication occurs between the ADV8005 and the downstream sink, the ADV8005 can trigger an interrupt to notify this error to the user or the controlling CPU. The EDID/HDCP controller will then report the HDCP error code via the status field hdcp_controller_error[3:0]. The error code is only valid when the hdcp_error_int interrupt bit is set to 1. The last error code will remain in the HDCP/EDID controller error field even when the interrupt is cleared.

hdcp_controller_error[3:0], TX2 Main Map, *Address 0xF4C8[7:4] (Read Only)*
This signal is used to readback the error code when the HDCP controller error interrupt HDCP_ERROR_INT is 1.

Function

hdcp_controller_error[3:0]	Description
0000 (default)	No error
0001	Bad receiver BKSV
0010	Ri Mismatch
0011	Pj Mismatch
0100	I2C error (usually no acknowledge)
0101	Timed out waiting for downstream repeater
0110	Maximum cascade of repeaters exceeded
0111	SHA-1 Hash check of KSV list fail

6.10. VIDEO SETUP

6.10.1. *Input Format*

The HDMI Tx core of the ADV8005 receives video data from the ADV8005 digital core via a 36-bit wide bus and four synchronization signals: the pixel clock, the data enable, and the horizontal and vertical synchronization signals. The HDMI Tx core always receives the video data in a 4:4:4 and SDR format from the VSP core.
It is possible to send YCrCb 4:2:2 data from the TMDS RX directly to the HDMI Tx. In which case register 0xEC15 must be set appropriately in the Tx main map.

vfe_input_id[3:0], TX1 Main Map, *Address 0xEC15[3:0]*
This signal is used to specify the video input format.

Function

vfe_input_id[3:0]	Description
0000 «	RGB 444 or YCbCr 444
0001	YCbCr 422
0101	Pseudo 422 YCbCr

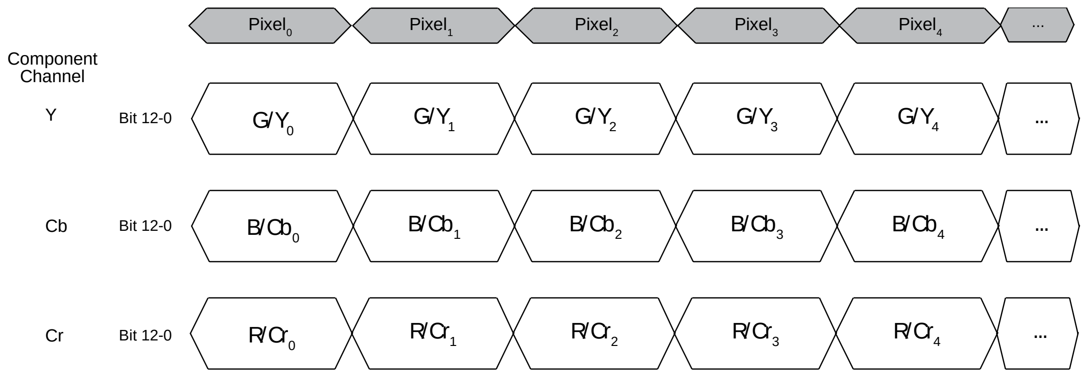


Figure 96: Format of Video Data Input into HDMI Tx Core

6.10.2. *Video Mode Detection*

The video mode detection feature can inform the user of the CEA-861 defined Video Identification Code (VIC) of the video being input to the Tx core, as well as some additional formats. If a CEA 861 format is detected, the VIC is contained in vic_detected[5:0]. Some additional non CEA 861 formats are contained in aux_vic_detected[2:0].

For some standards for which the VIC cannot be detected, the user needs to configure the following registers:

- The aspect ratio (set via the aspect_ratio bit) is used to distinguish between CEA-861 video timing codes where the aspect ratio is the only difference
- For 240p and 288p modes, the number of total lines can be selected in the progressive_mode_info[1:0] field
- The VIC detected is also affected by the pixel repetition (see Section 5.7 for more details)

The detected VIC is sent in the AVI InfoFrames unless pixel repetition is applied to the video stream transmitted by the ADV8005. When pixel repetition is applied to the video data, the VIC sent in the AVI InfoFrame may be different as the VIC is automatically determined by the ADV8005. To override the VIC detection, the pixel repetition mode must be set to manual by setting pr_value_manual[1:0] to 0b10 or 0b11. The desired VIC is then set. The Tx core can support non CEA 861 formats, but the VIC will not be automatically detected for these formats. In this case, the VIC should manually be set to the value 0.

vic_detected[5:0], TX2 Main Map, *Address 0xF43E[7:2] (Read Only)*
This signal is used to readback the input video code (VIC) detected (refer to the CEA-861 specification).

aux_vic_detected[2:0], TX2 Main Map, *Address 0xF43F[7:5] (Read Only)*
This register returns the format of video inputs that have a resolution not defined in the CEA 861 specification.

Function

aux_vic_detected[2:0]	Description
000 (default)	Set by Register 3E
001	240p Not Active
010	576i not active
011	288p not active
100	480i active
101	240p active
110	576i active
111	288p active

aspect_ratio, TX2 Main Map, *Address 0xF417[1]*
This bit is used to set the aspect ratio of input video. This bit is used to distinguish between CEA-861D video timing codes where aspect ratio is the only difference.

Function

aspect_ratio	Description
0 (default)	4:3
1	16:9

progressive_mode_info[1:0], TX2 Main Map, *Address 0xF43F[4:3] (Read Only)*
This bit is used to specify additional information for 240p or 288p input formats.

Function

progressive_mode_info[1:0]	Description
00 (default)	Reserved
01	262 total lines per frame for 240p and 312 total lines per frame for 288p
10	263 total lines per frame for 240p and 313 total lines per frame for 288p
11	Reserved for 240p and 314 total lines per frame for 288p

6.10.3. *Pixel Repetition*

Pixel repetition is used in HDMI to increase the amount of blanking period available to send packets or to increase the pixel clock to meet the minimum TMDS clock rate of 25 MHz. The ADV8005 offers three choices for the user to implement pixel repetition in the Tx core. These choices or modes are described below and can be set via pr_mode[1:0]:

Automatic mode: In automatic mode, the ADV8005 uses the audio sampling rate and the detected VIC information as parameters to decide if pixel repetition is needed to obtain sufficient blanking periods to send the audio. For an I2S input stream, the sampling rate is always set by the user via the i2s_sf[3:0] field**.** In the case of an SPDIF stream, the source of the audio sampling rate information is set via the audio_sampling_freq_sel bit. If the pixel repetition factor is adjusted to meet bandwidth requirements, the detected input VIC may be different from the VIC sent to the downstream sink. The VIC of the actual video sent across the HDMI link to the downstream sink, and which is included in the AVI InfoFrame, can be read from the vic_to_rx[5:0] field.

Manual mode: In the manual pixel repetition mode, the VIC sent in the AVI InfoFrame needs to be set. The factor between the pixel clock input to the Tx core and the output TMDS clock frequency must be programmed in the pr_pll_manual[1:0] field. The pixel repetition value sent to the HDMI sink must be programmed in pr_value_manual[1:0]. Refer to the latest HDMI specification for more details on valid pixel repetition formats.

Max mode: The max mode works in the same way as the automatic mode, except that it always selects the highest pixel repetition factor the Tx core is capable of. This makes the video timing independent of the audio sampling rate. This mode is not typically used.

pr_mode[1:0], TX2 Main Map, *Address 0xF43B[6:5]*
This signal is used to specify the pixel repetition mode selection. This should be set to 00 unless a non CEA-861 standard video resolution must be supported.

Function

pr_mode[1:0]	Description
00 (default)	auto mode
01	max mode
10	manual mode
11	manual mode

pr_pll_manual[1:0], TX2 Main Map, *Address 0xF43B[4:3]*
This signal is used to specify the ratio between the input pixel clock and the TMDS output clock when manual pixel repetition is enabled.

Function

pr_pll_manual[1:0]	Description
00 (default)	x1
01	x2
10	x4
11	x4

pr_value_manual[1:0], TX2 Main Map, *Address 0xF43B[2:1]*
This signal is used to specify the user programmed pixel repetition sent to the downstream sink. This field is used in manual pixel repetition.

Function

pr_value_manual[1:0]	Description
00 (default)	x1
01	x2
10	x4
11	x4

vic_to_rx[5:0], TX2 Main Map, *Address 0xF43D[5:0] (Read Only)*
This signal is used to set the AVI InfoFrame video code (VIC) to send to the downstream sink.

Function

vic_to_rx[5:0]	Description
xxxxxx	VIC sent to the downstream sink

6.10.4. *Video Related Packets and InfoFrames*

Video related packets and InfoFrames which include the AVI InfoFrame, MPEG InfoFrame and Gamut Metadata packet (GMP) are described in Section 6.10.5, Section 6.10.6, and Section 6.10.7.

6.10.5. *AVI InfoFrame*

The AVI InfoFrame is defined in the latest CEA 861 specification. The user can enable the transmission of AVI InfoFrames to the downstream sink by setting the aviif_pkt_en bit. When the transmission of AVI InfoFrames is enabled, the Tx transmits an AVI InfoFrame once every two video fields. Table 54 provides the list of registers that can be used to configure AVI InfoFrames.

aviif_pkt_en, TX2 Main Map, *Address 0xF444[4]*
This bit is used to enable the AVI InfoFrame Packet.

Function

aviif_pkt_en	Description
0	Disable AVI InfoFrame
1 (default)	Enable AVI InfoFrame

Table 54: AVI InfoFrame Configuration Registers

HDMI Tx Main Map Address	Bit Location	Access Type	Default Value	Field or Byte Name[1]
0xEC52	[2:0]	R/W	0b0100	InfoFrame version number
0xEC53	[4:0]	R/W	0b01101	InfoFrame length
0xEC54	[7:0]	R/W	0b00000000	Checksum[2]
0xEC55	[7:0]	R/W	0b00000000	Data Byte 1
0xEC56	[7:0]	R/W	0b00000000	Data Byte 2
0xEC57	[7:0]	R/W	0b00000000	Data Byte 3
0xEC58	[7]	R/W	0b0	Bit 7 of Data Byte 4
0xEC59	[7:4]	R/W	0b0000	Bits [7:4] of Data Byte 5
0xEC5A	[7:0]	R/W	0b00000000	Data Byte 6
0xEC5B	[7:0]	R/W	0b00000000	Data Byte 7
0xEC5C	[7:0]	R/W	00000000	Data Byte 8
0xEC5D	[7:0]	R/W	00000000	Data Byte 9
0xEC5E	[7:0]	R/W	00000000	Data Byte 10
0xEC5F	[7:0]	R/W	00000000	Data Byte 11
0xEC60	[7:0]	R/W	00000000	Data Byte 12
0xEC61	[7:0]	R/W	00000000	Data Byte 13
0xEC62	[7:0]	R/W	00000000	Data Byte 14
0xEC63	[7:0]	R/W	00000000	Data Byte 15
0xEC64	[7:0]	R/W	00000000	Data Byte 16
0xEC65	[7:0]	R/W	00000000	Data Byte 17
0xEC66	[7:0]	R/W	00000000	Data Byte 18
0xEC67	[7:0]	R/W	00000000	Data Byte 19
0xEC68	[7:0]	R/W	00000000	Data Byte 20
0xEC69	[7:0]	R/W	00000000	Data Byte 21
0xEC6A	[7:0]	R/W	00000000	Data Byte 22
0xEC6B	[7:0]	R/W	00000000	Data Byte 23
0xEC6C	[7:0]	R/W	00000000	Data Byte 24
0xEC6D	[7:0]	R/W	00000000	Data Byte 25
0xEC6E	[7:0]	R/W	00000000	Data Byte 26
0xEC6F	[7:0]	R/W	00000000	Data Byte 27

1 As defined in the latest CEA 861 specification

2. Only used when auto_checksum_en = 0

6.10.6. *MPEG InfoFrame*

The MPEG InfoFrame is defined in the latest CEA 861 specification. Currently, the specification does not recommend using this InfoFrame. The transmission of MPEG InfoFrames can be enabled by setting the mpeg_pkt_en bit. When the transmission of MPEG InfoFrames is enabled, the ADV8005 transmits an MPEG InfoFrame once every two video fields. Table 55 provides a list of registers that can be used to configure MPEG InfoFrames.

mpeg_pkt_en, TX2 Main Map, *Address 0xF440[5]*
This bit is used to enable the MPEG Packet.

Function

mpeg_pkt_en	Description
0 (default)	Disabled
1	Enabled

Table 55: MPEG InfoFrame Configuration Registers

Packet Map Address	Access Type	Field Name	Default Value	Byte Name[1]
0xF220	R/W	mpeg_hb0[7:0]	0b00000000	Header Byte 0
0xF221	R/W	mpeg_hb1[7:0]	0b00000000	Header Byte 1
0xF222	R/W	mpeg_hb2[7:0]	0b00000000	Header Byte 2
0xF223	R/W	mpeg_pb0[7:0]	0b00000000	Data Byte 0
0xF224	R/W	mpeg_pb1[7:0]	0b00000000	Data Byte 1
0xF225	R/W	mpeg_pb2[7:0]	0b00000000	Data Byte 2
0xF226	R/W	mpeg_pb3[7:0]	0b00000000	Data Byte 3
0xF227	R/W	mpeg_pb4[7:0]	0b00000000	Data Byte 4
0xF228	R/W	mpeg_pb5[7:0]	0b00000000	Data Byte 5
0xF229	R/W	mpeg_pb6[7:0]	0b00000000	Data Byte 6
0xF22A	R/W	mpeg_pb7[7:0]	0b00000000	Data Byte 7
0xF22B	R/W	mpeg_pb8[7:0]	0b00000000	Data Byte 8
0xF22C	R/W	mpeg_pb9[7:0]	0b00000000	Data Byte 9
0xF22D	R/W	mpeg_pb10[7:0]	0b00000000	Data Byte 10
0xF22E	R/W	mpeg_pb11[7:0]	0b00000000	Data Byte 11
0xF22F	R/W	mpeg_pb12[7:0]	0b00000000	Data Byte 12
0xF230	R/W	mpeg_pb13[7:0]	0b00000000	Data Byte 13
0xF231	R/W	mpeg_pb14[7:0]	0b00000000	Data Byte 14
0xF232	R/W	mpeg_pb15[7:0]	0b00000000	Data Byte 15
0xF233	R/W	mpeg_pb16[7:0]	0b00000000	Data Byte 16
0xF234	R/W	mpeg_pb17[7:0]	0b00000000	Data Byte 17
0xF235	R/W	mpeg_pb18[7:0]	0b00000000	Data Byte 18
0xF236	R/W	mpeg_pb19[7:0]	0b00000000	Data Byte 19
0xF237	R/W	mpeg_pb20[7:0]	0b00000000	Data Byte 20
0xF238	R/W	mpeg_pb21[7:0]	0b00000000	Data Byte 21
0xF239	R/W	mpeg_pb22[7:0]	0b00000000	Data Byte 22
0xF23A	R/W	mpeg_pb23[7:0]	0b00000000	Data Byte 23
0xF23B	R/W	mpeg_pb24[7:0]	0b00000000	Data Byte 24
0xF23C	R/W	mpeg_pb25[7:0]	0b00000000	Data Byte 25
0xF23D	R/W	mpeg_pb26[7:0]	0b00000000	Data Byte 26
0xF23E	R/W	mpeg_pb27[7:0]	0b00000000	Data Byte 27

1 As defined in the latest CEA 861 specification

6.10.7. *Gamut Metadata*

The Gamut metadata packet (GMP) contains the sources Gamut boundary description. It is defined in the latest HDMI specification.

The contents of the GMP can be set via the Packet Map registers listed in Table 56. The user can enable the transmission of GMP to the downstream sink by setting the gm_pkt_en bit. When the transmission of GMP is enabled, the ADV8005 transmits a GMP once every two video fields.

The ADV8005 transmits the GMP data starting 400 pixel clock cycles after the leading edge of VSync. In order to avoid corrupting the GMP data during transmission, it is recommended that the user synchronizes all I^{2}C writes to the GMP registers so that the write begins 512 pixel clock cycles after the VSync leading edge. The VSync interrupt of the ADV8005 should be used to synchronize this timing. Figure 97 illustrates this timing requirement.

gm_pkt_en, TX2 Main Map, *Address 0xF440[2]*
This bit is used to enable the Gamut Metadata Packet.

Function

gm_pkt_en	Description
0 (default)	Disabled
1	Enabled

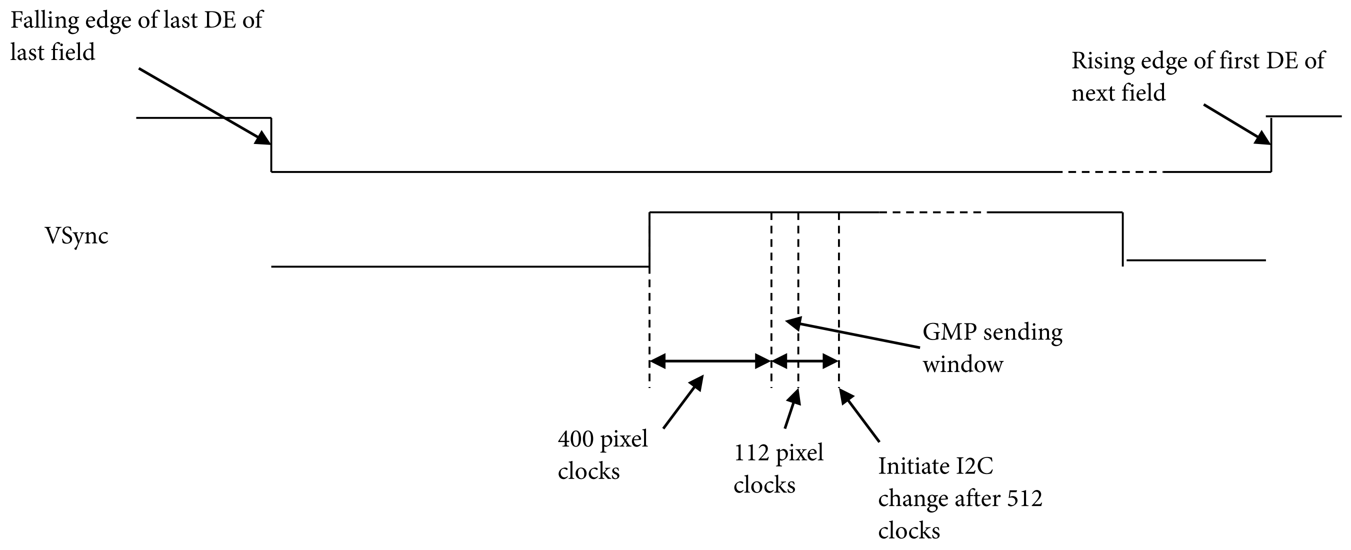

Falling edge of last DE of last field
Rising edge of first DE of next field
VSync
GMP sending window
400 pixel clocks
112 pixel clocks
Initiate I2C change after 512 clocks

Figure 97: I²C Write Timing if GMP Data

Table 56: Gamut Metadata Packet Configuration Registers

Packet Map Address	Access Type	Field Name	Default Value	Byte Name1
0xF2A0	R/W	gmp_hb0[7:0]	0b00000000	Header Byte 0
0xF2A1	R/W	gmp_hb1[7:0]	0b00000000	Header Byte 1
0xF2A2	R/W	gmp_hb2[7:0]	0b00000000	Header Byte 2
0xF2A3	R/W	gmp_pb0[7:0]	0b00000000	Data Byte 0
0xF2A4	R/W	gmp_pb1[7:0]	0b00000000	Data Byte 1
0xF2A5	R/W	gmp_pb2[7:0]	0b00000000	Data Byte 2
0xF2A6	R/W	gmp_pb3[7:0]	0b00000000	Data Byte 3
0xF2A7	R/W	gmp_pb4[7:0]	0b00000000	Data Byte 4
0xF2A8	R/W	gmp_pb5[7:0]	0b00000000	Data Byte 5
0xF2A9	R/W	gmp_pb6[7:0]	0b00000000	Data Byte 6
0xF2AA	R/W	gmp_pb7[7:0]	0b00000000	Data Byte 7
0xF2AB	R/W	gmp_pb8[7:0]	0b00000000	Data Byte 8
0xF2AC	R/W	gmp_pb9[7:0]	0b00000000	Data Byte 9
0xF2AD	R/W	gmp_pb10[7:0]	0b00000000	Data Byte 10
0xF2AE	R/W	gmp_pb11[7:0]	0b00000000	Data Byte 11
0xF2AF	R/W	gmp_pb12[7:0]	0b00000000	Data Byte 12
0xF2A0	R/W	gmp_pb13[7:0]	0b00000000	Data Byte 13
0xF2A1	R/W	gmp_pb14[7:0]	0b00000000	Data Byte 14
0xF2A2	R/W	gmp_pb15[7:0]	0b00000000	Data Byte 15
0xF2A3	R/W	gmp_pb16[7:0]	0b00000000	Data Byte 16
0xF2A4	R/W	gmp_pb17[7:0]	0b00000000	Data Byte 17
0xF2A5	R/W	gmp_pb18[7:0]	0b00000000	Data Byte 18
0xF2A6	R/W	gmp_pb19[7:0]	0b00000000	Data Byte 19
0xF2A7	R/W	gmp_pb20[7:0]	0b00000000	Data Byte 20
0xF2A8	R/W	gmp_pb21[7:0]	0b00000000	Data Byte 21
0xF2A9	R/W	gmp_pb22[7:0]	0b00000000	Data Byte 22
0xF2AA	R/W	gmp_pb23[7:0]	0b00000000	Data Byte 23
0xF2AB	R/W	gmp_pb24[7:0]	0b00000000	Data Byte 24
0xF2AC	R/W	gmp_pb25[7:0]	0b00000000	Data Byte 25
0xF2AD	R/W	gmp_pb26[7:0]	0b00000000	Data Byte 26
0xF2AE	R/W	gmp_pb27[7:0]	0b00000000	Data Byte 27

[1] As defined in the latest HDMI specification

6.11. AUDIO SETUP

6.11.1. *Audio Architecture*

The ADV8005 is capable of receiving audio data in I2S, SPDIF, DSD or High Bit Rate (HBR) formats. When the input audio is captured from the audio input pins, it is then converted into audio packets for transmission over the HDMI output interface.

The ADV8005 HDMI TX1 and TX2 process audio input streams independently, the following bits select which audio format is expected on the audio pins.

aud_input_mode[1:0], IO Map, *Address 0x1A08[7:6]*
This signal is used to select the audio input mode.

Function

aud_input_mode[1:0]	**Description**
00 (default)	Single mode.
01	Dual mode; TX1 with I2S stream, TX2 with SPDIF stream.
10	Dual mode; TX1 with SPDIF stream, TX2 with I2S stream.
11	Dual mode; TX1 with SPDIF stream 1, TX2 with SPDIF stream 2.

Table 57: HDMI Tx Supported Audio Input Modes from Audio Input Pins

Pin\aud_input_mode[1:0]	**0**		**1**		**2**		**3**	
	Single Mode		**Dual Mode 1**		**Dual Mode 2**		**Dual Mode 3**	
	Tx1	**Tx2**	**Tx1**	**Tx2**	**Tx1**	**Tx2**	**Tx1**	**Tx2**
DSD_CLK	DSD_CLK	DSD_CLK	MCLK			MCLK		MCLK
MCLK	MCLK	MCLK		MCLK	MCLK		MCLK	
SCLK	SCLK	SCLK	SCLK			SCLK		
AUD_IN[0]	DSD.0/SPDIF	DSD.0/SPDIF		SPDIF	SPDIF		SPDIF	
AUD_IN[1]	DSD.1/I2S.0	DSD.1/I2S.0	I2S0			I2S0		
AUD_IN[2]	DSD.2/I2S.1	DSD.2/I2S.1	I2S1			I2S1		
AUD_IN[3]	DSD.3/I2S.2	DSD.3/I2S.2	I2S2			I2S2		
AUD_IN[4]	DSD.4/I2S.3	DSD.4/I2S.3	I2S3			I2S3		
AUD_IN[5]	DSD.5/LRCLK	DSD.5/LRCLK	LRCLK			LRCLK		SPDIF

6.11.2. *Audio from Serial Video Rx*

On the ADV8003, it is possible to route audio packets directly from the Serial Video Rx to the HDMI Tx. To achieve passthrough of audio from the Serial Video Rx to the HDMI Tx, the following bit in the HDMI Tx must be configured.

rx_aud_packet_sel, TX2 Main Map, *Address 0xF40B[0]*
This bit is used to select the source of audio packet data routed into the HDMI transmitter.

Function

rx_aud_packet_sel	**Description**
0 (default)	Get audio packet from external audio pins
1	Get audio packet from internal audio receiver

The datapath can be enabled per Tx so that one Tx can receive audio from I2S lines and one from the Serial Video Rx. Alternatively, both Txs can receive audio from the Serial Video Rx or from the audio input pins. Along with setting rx_aud_packet_sel, the mclk_ratio[1:0] must be set to 00 and mclk_en must be set to 1.

The audio InfoFrame is not transferred internally from the serial video Rx to the HDMI Tx. This needs to be done by software.

6.11.3. *Audio Configuration*

The audio_input_sel[2:0], audio_mode[1:0], and i2s_format[1:0] fields must be used to configure the Tx core according to the incoming audio input. Refer to Figure 98 to Figure 104 for more information on the audio timing formats. There is a manual control to set the audio sample packet layout to be 0 or 1 using the ext_layoutand layout_sel controls.

ext_layout, TX2 Main Map, *Address 0xF44A[3]*
This bit is used to set the external audio layout value.

Function

ext_layout	Description
0 (default)	Dual channel
1	Multi channel

layout_sel, TX2 Main Map, *Address 0xF44A[2]*
This bit is used to select the audio layout value.

Function

layout_sel	Description
0 (default)	Internal layout
1	External layout

audio_input_sel[2:0], TX2 Main Map, *Address 0xF40A[6:4]*
This signal is used to specify the audio mode when the input format of the audio is specified.

Function

audio_input_sel[2:0]	Description
000 (default)	I2S
001	SPDIF
010	One Bit Audio (DSD)
011	High Bit Rate (HBR) Audio
100	Reserved

i2s_format[1:0], TX2 Main Map, *Address 0xF40C[1:0]*
This signal is used to set the format of the I2S audio stream input to the part.

Function

i2s_format[1:0]	Description
00 (default)	I2S
01	Right justified
10	Left justified
11	AES3 direct mode

audio_mode[1:0], TX2 Main Map, *Address 0xF40A[3:2]*
This signal is used to specify the exact audio mode when the input format of the audio is specified.
Case 1: DSD (audio_input_select = 0b010): 0x = DSD raw mode; 1x = SDIF-3 mode
Case 2: HBR (audio_input_select = 0b011): 00 = 4 stream, with Bi-Phase Mark (BPM) encoding; 01 = 4 stream, without BPM encoding; 10 = 1 stream, with BPM encoding; 11 = 1 stream, without BPM encoding
Case 3: DST (audio_input_select = 0b100): x0 = normal mode; 01 = DST 2x clock; 10 = DST 1x clock (DDR)

Function

audio_mode[1:0]	Description
Case 1	DSD (AUDIO_INPUT_SELECT = 0b010)
0x	DSD raw mode
1x	SDIF-3 mode
Case 2	HBR (AUDIO_INPUT_SELECT = 0b011)
00 (default)	4 stream, with Bi-Phase Mark (BPM) encoding
01	4 stream, without BPM encoding
10	1 stream, with BPM encoding
11	1 stream, without BPM encoding

mclk_ratio[1:0], TX2 Main Map, *Address 0xF40A[1:0]*
This signal is used to specify the ratio between the audio sampling frequency and the clock described using the N and CTS values.

Function

mclk_ratio[1:0]	Description
00	128*fs
01 (default)	256*fs
10	384*fs
11	512*fs

mclk_en, TX2 Main Map, *Address 0xF40B[5]*
This bit is used to select the audio master clock that is used by the audio block.

Function

mclk_en	Description
0 (default)	Use internally generated MCLK
1	Use external MCLK

Table 58: Valid Configuration for audio_mode[1:0]

audio_input_sel Value	audio_mode Value Options	Corresponding Configuration
0b010	0b0x	DSD in raw mode
	0b1x	DSD in SDIF-3 mode
0b011	0b00	HBR input as 4 streams, with Bi-Phase Mark (BPM) encoding
	0b01	HBR input as 4 stream, without BPM encoding
	0b10	HBR input as 4 stream, without BPM encoding
	0b11	HBR input as 1 stream, without BPM encoding

Table 59: Audio Input Format Summary

Input							Output	
audio_input_sel Value	audio_mode Value	I2s_format Value	Audio Input Signal	Clock Pins	Encoding	ADV8005 Input Pin Mapping	Format	Packet Type
0b000	0bXX	0b00	I2S[3:0]	SCLK, LRCLK, MCLK[1]	Normal	AUD_IN[4:0] AUD_IN[5] SCLK MCLK	Standard I2S	Audio Sample Packet
0b000	0bXX	0b01	I2S[3:0]	SCLK, LRCLK, MCLK[1]	Normal	AUD_IN[4:0] AUD_IN[5] SCLK MCLK	Right justified	Audio Sample Packet
0b000	0bXX	0b10	I2S[3:0]	SCLK, LRCLK, MCLK[1]	Normal	AUD_IN[4:0] AUD_IN[5] SCLK MCLK	Left justified	Audio Sample Packet
0b000	0bXX	0b11	I2S[3:0]	SCLK, LRCLK, MCLK[1]	Normal	AUD_IN[4:0] AUD_IN[5] SCLK MCLK	AES3 direct	Audio Sample Packet
0b001	0b00	0bXX	SPDIF	MCLK[1]	Biphase Mark	AUD_IN[0] MCLK	IEC60958 or IEC61937	Audio Sample Packet
0b010	0b1X	0bXX	DSD[5:0]	SCLK	Normal	AUD_IN[5:0] SCLK	DSD	DSD Packet
0b010	0b1X	0bXX	DSD[5:0]]	SCLK	SDIF-3	AUD_IN[5:0] SCLK	DSD	DSD Packet
0b011	0b00	0bXX	I2S[3:0]	MCLK	Biphase Mark	AUD_IN[4:0] MCLK	IEC61937	HBR Packet

Input							Output	
audio_input _sel Value	**audio_mode Value**	**I2s_format Value**	**Audio Input Signal**	**Clock Pins**	**Encoding**	**ADV8005 Input Pin Mapping**	**Format**	**Packet Type**
0b011	0b01	0b00	I2S[3:0]	SCLK, MCLK[1]	Normal	AUD_IN[4:0] SCLK MCLK	Standard I2S	HBR Packet
0b011	0b01	0b01	I2S[3:0]	SCLK, MCLK	Normal	AUD_IN[4:0] SCLK MCLK	Right justified	HBR Packet
0b011	0b01	0b10	I2S[3:0]	SCLK, MCLK[1]	Normal	AUD_IN[4:0] SCLK MCLK	Left justified	HBR Packet
0b011	0b01	0b11	I2S[3:0]	SCLK, MCLK[1]	Normal	AUD_IN[4:0] SCLK MCLK	AES3 Direct	HBR Packet
0b011	0b10	0bXX	SPDIF	MCLK	Biphase Mark	AUD_IN[0] MCLK	IEC61937	HBR Packet
0b011	0b11	0b00	SPDIF	SCLK, MCLK[1]	Normal	AUD_IN[0] SCLK MCLK	Standard I2S	HBR Packet
0b011	0b11	0b01	I2S[3:0]	SCLK, MCLK[1]	Normal	AUD_IN[4:0] SCLK MCLK	Right Justified	HBR Packet
0b011	0b11	0b10	I2S[3:0]	SCLK, MCLK[1]	Normal	AUD_IN[4:0] SCLK MCLK	Left Justified	HBR Packet
0b011	0b11	0b11	I2S[3:0]	MCLK	Normal	AUD_IN[4:0] MCLK	IEC61937	HBR Packet

[1] Optional signal

6.11.3.1. *I2S Audio*

The ADV8005 can receive up to four stereo channels of I2S audio at up to a 192 kHz sampling rate. The number of I2S channels the Tx processes can be selected with audioif_cc[2:0]. The selection of the active I2S channels is done via the i2s_en[3:0] field. The audio sampling frequency of the input stream must be set appropriately via the i2s_sf[3:0] field. This value is used along with the VIC to determine the pixel repetition factor that the Tx core applies to the video data (refer to Section 6.10.3). The value programmed in i2s_sf[3:0] is also used to be sent across the TMDS output link in the channel status data information contained in the Audio Sample packets.

The placement of I2S channels into the Audio Sample subpackets defined in the HDMI specification can be specified in the following fields:

- subpkt0_l_src
- subpkt0_r_src
- subpkt1_l_src
- subpkt1_r_src
- subpkt2_l_src
- subpkt2_r_src
- subpkt3_l_src
- subpkt3_r_src

When these fields are set to their default values, all I2S channels are placed in their respective position (for example, I2S0 left channel in channel 0 left position, I2S3 right channel in channel 3 right position, and so on) but this mapping is completely programmable if desired.

The ADV8005 can receive standard I2S, left-justified, right-justified, and direct AES3 stream formats with a sample word width between 16 bits and 24 bits. The format of the input I2S stream is set via i2s_format[1:0] while the audio sample word width is set via the word_length[3:0] field. The ADV8005 can also receive an I2S stream in both 64-bit and 32-bit modes, so either 32- or 16-bit clock (that is, the signal input through

SCLK pin) edges or cycles per channel are valid. The ADV8005 will adapt to 32- or 64-bit modes automatically, and the current mode can be read in the i2s_32bit_mode field. Refer to Figure 100 to Figure 104 for timing diagrams on I2S streams input to the ADV8005.

When the ADV8005 is configured to receive a direct AES3 stream, the stream it receives should have IEC60958-like subframes (refer to Figure 98) with the stream formatted as follows:

- Data should be aligned as shown in Figure 98.
- Preamble left out as shown in Figure 99.
- Parity bit is replaced by the block start flag. The ADV8005 automatically computes the parity bit.

The channel status data collected from the audio stream input to the AUD_IN[0] pin is used in the Audio Sample packets sent by the ADV8005 to the downstream sink. The channel status data can alternately be programmed by setting the cs_bit_override bit. When cs_bit_override is set to 1, setting audio_sampling_freq_sel allows the programming of the audio sampling frequency used for the channel status bits while all other channel status data is extracted from the audio stream input to I2S0. The sampling frequency is set via the i2s_sf[3:0] field.

Note: All four stereo channels (AUD_IN[3:0]) are enabled by setting i2s_en[3:0] to 0xF and audioif_cc[2:0] to 0x7. If one stereo channel only is needed, the I2S audio stream data must be input to AUD_IN[0]. The i2s_en[3:0] and audioif_cc[2:0] control fields must be set to 1.

When audio_sampling_freq_sel is set to 1, the audio sampling frequency programmed via I2S_SF is used for the determination of the pixel repetition factor (refer to Section 5.7 for more details).

audioif_sf[2:0], TX2 Main Map, *Address 0xF474[4:2]*
This signal is used to specify the Audio Sampling Frequency in the Audio InfoFrame.

Function

audioif_sf[2:0]	**Description**
000 (default)	Case 1: Not DSD audio or Case 2: DSD audio (AUDIO_INPUT_SEL = 0b010)
001	64 x 32 kHz
010	64 x 44.1 kHz
011	64 x 48 kHz
100	64 x 88.2 kHz
101	64 x 96 kHz
110	64 x 176.4 kHz
111	64 x 192 kHz

audioif_cc[2:0], TX2 Main Map, *Address 0xF473[2:0]*
This signal is used to set the Audio Channel Count (Audio InfoFrame).

Function

audioif_cc[2:0]	**Description**
000 (default)	Refer to Stream Header
001	2 channels
010	3 channels
011	4 channels
100	5 channels
101	6 channels
110	7 channels
111	8 channels

i2s_en[3:0], TX2 Main Map, *Address 0xF40C[5:2]*
This signal is used to enable the I2S pins.

Function

i2s_en[3:0]	**Description**
0000	All I2S disabled
1111 (default)	All I2S enabled

i2s_sf[3:0], TX2 Main Map, *Address 0xF415[7:4]*
This signal is used to set the Sampling frequency for I2S audio. This information is used both by the audio Rx and the pixel rep. Other values reserved.

Function

i2s_sf[3:0]	Description
0000 (default)	44.1kHz
0001	Do not use
0010	48kHz
0011	32kHz
0100	Do not use
0101	Do not use
0110	Do not use
0111	Do not use
1000	88.2kHz
1001	Do not use
1010	96kHz
1011	Do not use
1100	176.4kHz
1101	Do not use
1110	192kHz
1111	Do not use

subpkt0_l_src[2:0], TX2 Main Map, *Address 0xF40E[5:3]*
This signal is used to specify the source of sub packet 0, left channel.

Function

subpkt0_l_src[2:0]	Description
000 (default)	I2S[0], left channel
001	I2S[0], right channel
010	I2S[1], left channel
011	I2S[1], right channel
100	I2S[2], left channel
101	I2S[2], right channel
110	I2S[3], left channel
111	I2S[3], right channel

subpkt0_r_src[2:0], TX2 Main Map, *Address 0xF40E[2:0]*
This signal is used to specify the source of sub packet 0, right channel.

Function

subpkt0_r_src[2:0]	Description
000	I2S[0], left channel
001 (default)	I2S[0], right channel
010	I2S[1], left channel
011	I2S[1], right channel
100	I2S[2], left channel
101	I2S[2], right channel
110	I2S[3], left channel
111	I2S[3], right channel

subpkt1_l_src[2:0], TX2 Main Map, *Address 0xF40F[5:3]*
This signal is used to specify the source of sub packet 1, left channel.

Function

subpkt1_l_src[2:0]	Description
000	I2S[0], left channel
001	I2S[0], right channel
010 (default)	I2S[1], left channel
011	I2S[1], right channel
100	I2S[2], left channel
101	I2S[2], right channel
110	I2S[3], left channel
111	I2S[3], right channel

subpkt1_r_src[2:0], TX2 Main Map, *Address 0xF40F[2:0]*

This signal is used to specify the source of sub packet 1, right channel.

Function

subpkt1_r_src[2:0]	Description
000	I2S[0], left channel
001	I2S[0], right channel
010	I2S[1], left channel
011 (default)	I2S[1], right channel
100	I2S[2], left channel
101	I2S[2], right channel
110	I2S[3], left channel
111	I2S[3], right channel

subpkt2_l_src[2:0], TX2 Main Map, *Address 0xF410[5:3]*

This signal is used to specify the source of sub packet 2, left channel.

Function

subpkt2_l_src[2:0]	Description
000	I2S[0], left channel
001	I2S[0], right channel
010	I2S[1], left channel
011	I2S[1], right channel
100 (default)	I2S[2], left channel
101	I2S[2], right channel
110	I2S[3], left channel
111	I2S[3], right channel

subpkt2_r_src[2:0], TX2 Main Map, *Address 0xF410[2:0]*

This signal is used to specify the source of sub packet 2, right channel.

Function

subpkt2_r_src[2:0]	Description
000	I2S[0], left channel
001	I2S[0], right channel
010	I2S[1], left channel
011	I2S[1], right channel
100	I2S[2], left channel
101 (default)	I2S[2], right channel
110	I2S[3], left channel
111	I2S[3], right channel

subpkt3_l_src[2:0], TX2 Main Map, *Address 0xF411[5:3]*

This signal is used to specify the source of sub packet 3, left channel.

Function

subpkt3_l_src[2:0]	Description
000	I2S[0], left channel
001	I2S[0], right channel
010	I2S[1], left channel
011	I2S[1], right channel
100	I2S[2], left channel
101	I2S[2], right channel
110 (default)	I2S[3], left channel
111	I2S[3], right channel

subpkt3_r_src[2:0], TX2 Main Map, *Address 0xF411[2:0]*
This signal is used to specify the source of sub packet 3, right channel.

Function

subpkt3_r_src[2:0]	**Description**
000	I2S[0], left channel
001	I2S[0], right channel
010	I2S[1], left channel
011	I2S[1], right channel
100	I2S[2], left channel
101	I2S[2], right channel
110	I2S[3], left channel
111 (default)	I2S[3], right channel

i2s_32bit_mode, TX2 Main Map, *Address 0xF442[3] (Read Only)*
This bit is used to readback the I2S mode detection. It shows the number of SCLK periods per LRCLK period.

Function

i2s_32bit_mode	**Description**
0 (default)	I2S 32 bit mode detected
1	I2S 64 bit mode detected

cs_bit_override, TX2 Main Map, *Address 0xF40C[6]*
This bit is used to select the source of channel status bits when using I2S Mode 4.

Function

cs_bit_override	**Description**
0 (default)	Use channel status bits from I2S stream
1	Use channel status bits programmed in I2C registers

audio_sampling_freq_sel, TX2 Main Map, *Address 0xF40C[7]*
This bit is used to select whether the audio sampling frequency is set automatically or manually (via I2C).

Function

audio_sampling_freq_sel	**Description**
0	Use sampling frequency from I2S stream, for SPDIF stream
1 (default)	Use sampling frequency from I2C registers

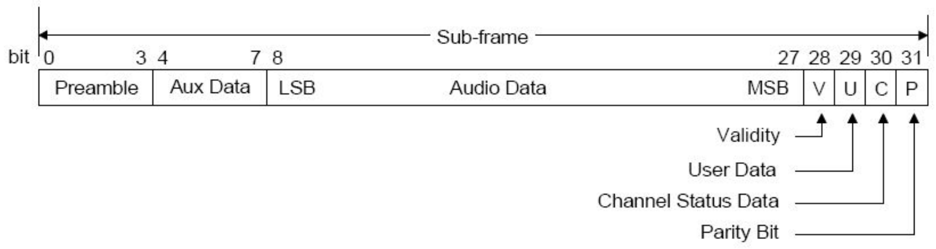


Figure 98: IEC60958 Sub Stream

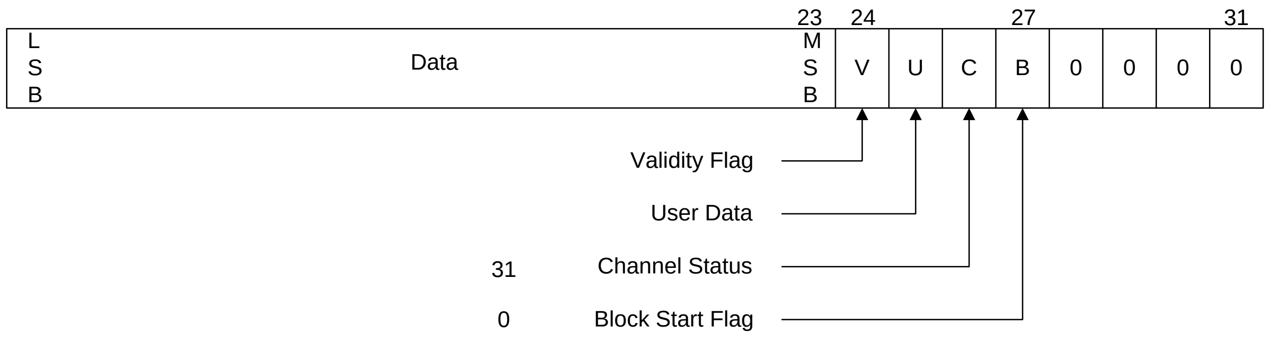


Figure 99: AES3 Stream Format Input to ADV8005

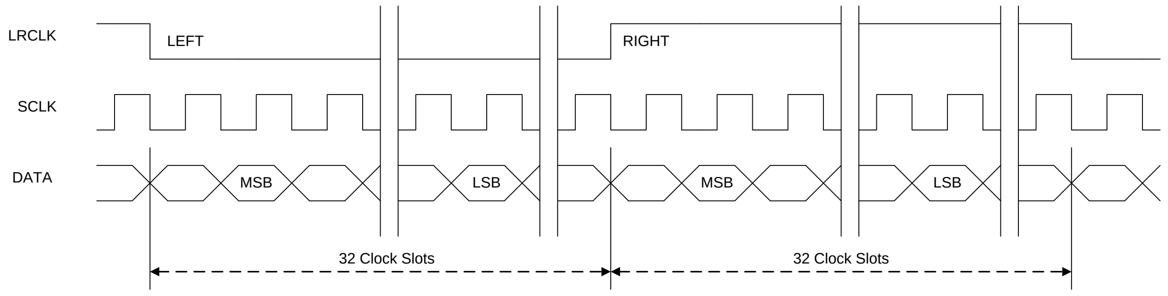


Figure 100: Timing of Standard I2S Stream Input to ADV8005

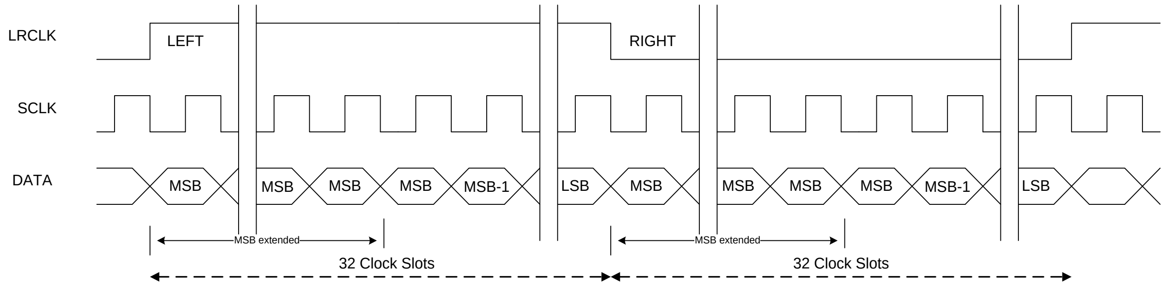


Figure 101: Timing for Right-Justified I2S Stream Input to ADV8005

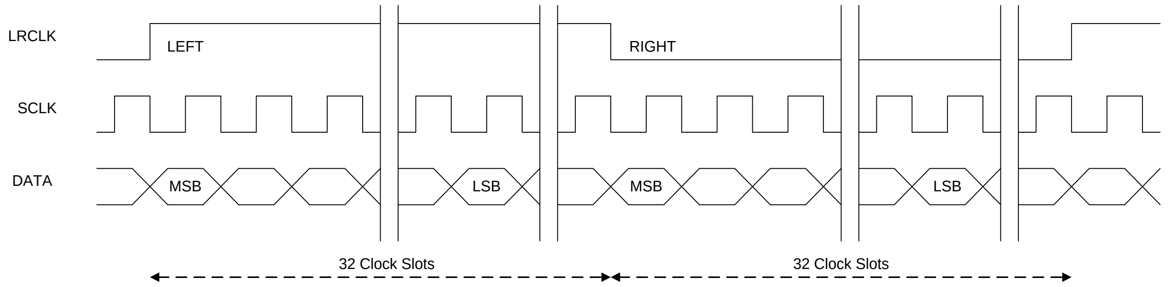


Figure 102: Timing for Left-Justified I2S Stream Input to ADV8005

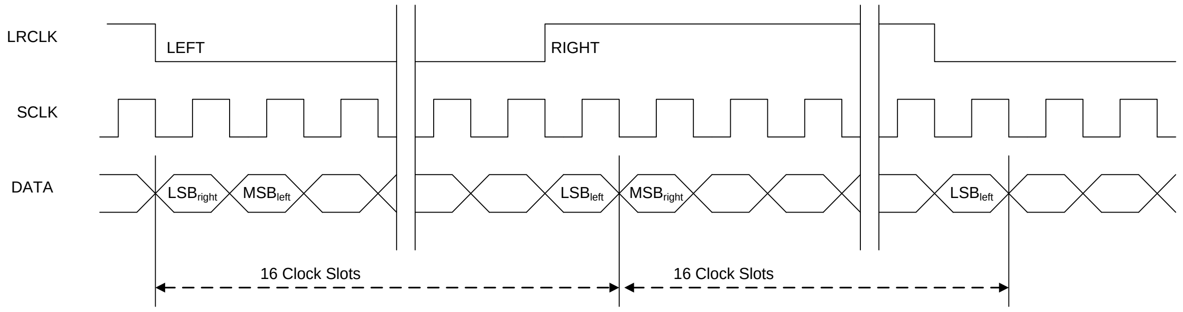


Figure 103: Timing for I2S Stream in 32-bit Mode

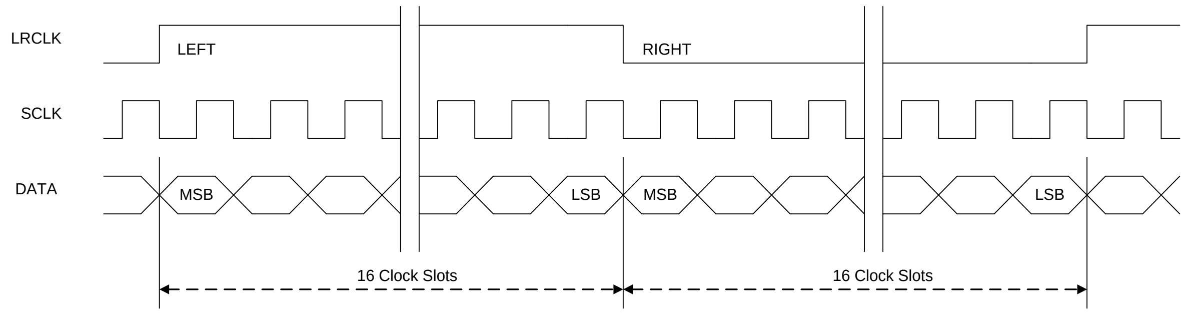


Figure 104: Timing for I2S Stream in Left or Right-Justified and 32-bit Modes

6.11.3.2. SPDIF Audio

The ADV8005 can receive two channel LPCM or encoded multichannel audio up to a 192 kHz sampling rate via the SPDIF input interface. The detected sampling frequency for the SPDIF input stream can be read via the spdif_sf[3:0] field.

It is possible to set the sampling audio sampling frequency of the input SPDIF stream. This is done by setting audio_sampling_freq_sel to 1. When audio_sampling_freq_sel is set to 1, the sampling frequency used to determine the pixel repetition factor (refer to Section 6.11.1) is not extracted from the input SPDIF stream and must be programmed in the i2s_sf[3:0] field. Note that the sampling frequency that is used in the Audio Sample packets sent to the downstream sink can be read from the spdif_sf[3:0] field.

The ADV8005 is capable of accepting SPDIF with or without an audio master clock input to through the input pin MCLK. When the ADV8005 does not receive an audio master clock, the ADV8005 uses the bit clock input via the SCLK pin to internally generate an audio master clock and determine the CTS value.

spdif_sf[3:0], TX2 Main Map, *Address 0xF404[7:4] (Read Only)*
This signal is used to readback the audio sampling frequency from the SPDIF channel.

Function

spdif_sf[3:0]	**Description**
0000 (default)	44.1kHz
0001	NA
0010	48 kHz
0011	32kHz
0100	NA
0101	NA
0110	NA
0111	NA
1000	88.2kHz
1001	NA
1010	96kHz
1011	NA
1100	176.4kHz
1101	NA
1110	192kHz
1111	NA

6.11.3.3. DSD Audio

The ADV8005 uses 1-bit Audio Sample packets to transmit DSD audio data across the HDMI link to the downstream sink. The ADV8005 supports up to six channels of DSD data which can be input onto six data lines clocked by the signal input to DSD_CLK.

The ADV8005 can be configured to receive a DSD stream by setting audio_input_sel[2:0] to 0b010. The mode of the DSD stream input to the ADV8005 can be set via the audio_mode[1:0] field. The audio sampling frequency must be set via the audioif_sf[2:0] field. Note the DSD clock input to SCLK has a frequency that is 64 times the audio sampling frequency programmed in the audioif_sf[2:0] field.

Refer to Table 59 for additional details on the DSD modes supported by the ADV8005.

Table 60: Valid Configuration for audioif_sf[2:0] Address B8 (Main), Address 0x74[4:2]

audio_input_sel Value	audioif_sf Value Options	Corresponding Configuration
≠0b010	0b000	Not DSD Audio
0b010	0b001	DSD Audio, 64x32 kHz
	0b010	DSD Audio, 64x44.1 kHz
	0b011	DSD Audio, 64x48 kHz
	0b100	DSD Audio, 64x88.2 kHz
	0b101	DSD Audio, 64x96 kHz
	0b110	DSD Audio, 64x176.4 kHz
	0b111	DSD Audio, 64x192 kHz

6.11.3.4. HBR Audio

The ADV8005 uses an HBR audio packet to transmit across the TMDS link compressed audio streams conforming to IEC 61937 and with high bit rate (that is, bit rate higher than 6.144 Mbps).

The ADV8005 can be configured to receive an HBR stream by setting audio_input_sel[2:0] to 0b011. The use of one or four input stream(s) with or without biphase mark (BPM) encoding can be selected via the audio_mode[1:0] field. Note that an audio master clock input through the pin MCLK_IN is always required for the BPM encoding modes. For HBR mode, the audio sampling frequency must be set via the audioif_sf[2:0] field.

papb_sync can be toggled from 0 to 1 to synchronize the Pa and Pb syncword, which marks the beginning of a stream repetition with the subpacket 0. For data bursts with a repetition period, which is a multiple of four frames, the synchronization will persist. If the data burst does not have a repetition period of four frames, setting papb_sync is not needed but will not have any negative effects. The transition of the bit from 0 to 1 causes the one time synchronization, so setting the bit from 1 to 0 will have no effect.

The mapping between the I2S input signals to the Tx core and the HBR subpackets can be via the following controls:

- subpkt0_l_src
- subpkt0_r_src
- subpkt1_l_src
- subpkt1_r_src
- subpkt2_l_src
- subpkt2_r_src
- subpkt3_l_src
- subpkt3_r_src

Note: When the HBR input stream is coming from an ADI HDMI Rx device or from the Rx section of the ADV8005, the fields listed above are set to the respective default values. Since there is no standard for chip to chip HBR transfer, different settings may be required to map the HBR stream input to the Tx core and a non ADI HDMI Rx device.

Refer to Table 59 for additional details on the HBR modes supported by the ADV8005.

papb_sync, TX2 Main Map, *Address 0xF447[6]*
This bit is used to synchronize the Pa and Pb syncwords with subpacket 0 for HBR audio.

Function

papb_sync	Description
0 (default)	No function
1	Synchronize Pa and Pb syncwords with subpacket 0

6.11.4. *N and CTS Parameters*

The audio data carried across the HDMI link to the downstream sink, which is driven by a TMDS clock only, does not retain the original audio sample clock. The task of recreating this clock at the sink is called Audio Clock Regeneration (ACR). There are varieties of ACR methods that can be implemented in an HDMI sink, each with a different set of performance characteristics. The HDMI specification does not attempt to define exactly how these mechanisms operate. It does, however, present a possible configuration and defines the data items that the HDMI source shall supply to the HDMI sink in order to allow the HDMI sink to adequately regenerate the audio clock.

The HDMI specification also defines how that data shall be generated. In many video source devices, the audio and video clocks are generated from a common clock (coherent clocks). In that situation, there exists a rational (integer divided by integer) relationship between these two clocks. The ACR architecture can take advantage of this rational relationship and can also work in an environment where there is no such relationship between these two clocks, that is, where the two clocks are truly asynchronous or where their relationship is unknown.

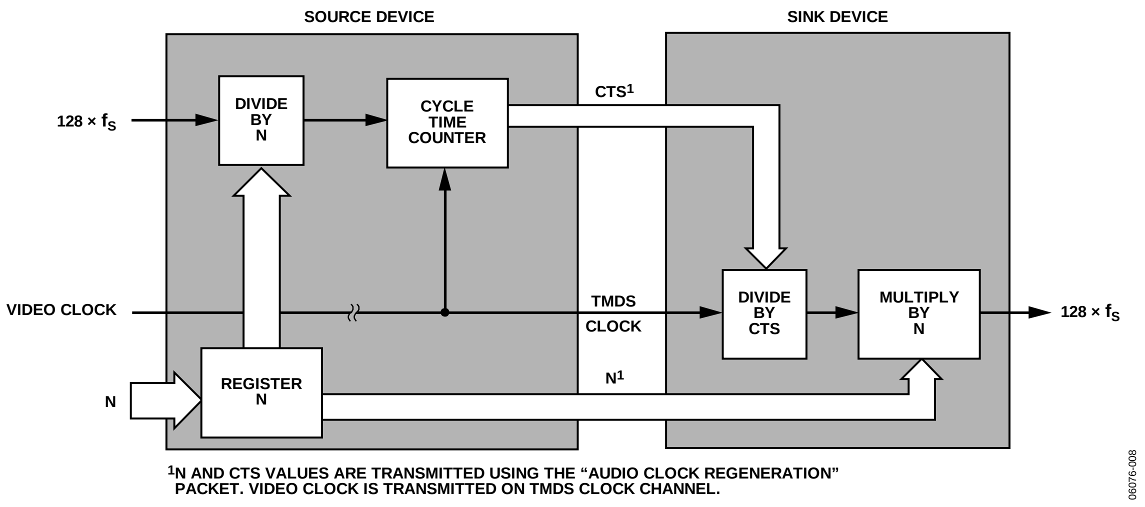


Figure 105: Audio Clock Regeneration

Figure 105 illustrates the overall system architecture model used by HDMI Rxs for audio clock regeneration. The HDMI source determines the fractional relationship between the video clock and an audio reference clock (128*fs) and passes the numerator and denominator for that fraction to the sink across the HDMI link. The sink may then recreate the audio clock from the TMDS clock by using a clock divider and a clock multiplier. The relationship between the two clocks is shown in Equation 23.

$$128 f_s = f_{TMDS\_CLK} \frac{N}{CTS}$$

Equation 23: Relationship Between Audio Reference and TMDS Clocks

The source determines the value of the numerator N as specified in the HDMI specification. Typically, this value N is used in a clock divider to generate an intermediate clock that is slower than the 128*fs clock by the factor N. The source typically determines the value of the denominator Cycle Time Stamp (CTS) by counting the number of TMDS clocks in each of the 128*fs/N clocks.

6.11.4.1. *N Parameter*

N is an integer number and is calculated using Equation 24 with the recommended optimal value shown in Equation 25 which approximately equals N for coherent audio and video clock sources. Table 61 to Table 63 can be used to determine the value of N. For non coherent sources or sources where coherency is not known, Equation 24, Equation 25, and Equation 26 should be used.

$$128^*fS/1500Hz \le N \le 128^*fS/300Hz$$

Equation 24: Restriction for N Value

$$128^*fs/1000Hz$$

Equation 25: Optimal N Value

6.11.4.2. CTS Parameter

The CTS value is an integer number that satisfies Equation 26.

$$CTS_{Average} = \frac{f_{TMDS\_CLK} N}{128 f_s}$$

Equation 26: Relationship Between N and CTS

6.11.4.3. Recommended N and Expected CTS Values

The recommended values of N for several standard pixel clocks are given in Table 61 to Table 63.

The ADV8005 has two modes for CTS generation.

Manual mode: Manual mode is selected by setting cts_sel to 1. In manual mode, the user can program the CTS number directly into the chip via the cts_manual[19:0] field. Manual mode is good for coherent audio and video, where the audio and video clocks are generated from the same crystal; thus CTS should be a fixed number.

Automatic mode: Automatic mode is selected by setting cts_sel to 0. In automatic mode, the chip computes the CTS based on the actual audio and video rates. The result can be read from the cts_internal[19:0] field. Automatic mode is good for incoherent audio or video, where there is no simple integer ratio between the audio and video clock.

The 20-bit n value used by the Tx core of the ADV8005 can be programmed in the n[19:0] field.

cts_sel, TX2 Main Map, *Address 0xF40A[7]*
This bit is used to specify whether CTS is automatically or manually set.

Function

cts_sel	Description
0 (default)	Automatic CTS. Use the internally generated CTS value
1	Manual CTS. Use the CTS programmed via CTS_MANUAL[19:0]

cts_manual[19:0], TX2 Main Map, *Address 0xF407[3:0]; Address 0xF408[7:0]; Address 0xF409[7:0]*
This signal is used to manually set the Cycle Time Stamp (CTS). This parameter is used with the N parameter to regenerate the audio clock in the receiver.

cts_internal[19:0], TX2 Main Map, *Address 0xF404[3:0]; Address 0xF405[7:0]; Address 0xF406[7:0] (Read Only)*
This signal is used to readback the automatically generated Cycle Time Stamp (CTS) parameter. This parameter is used with the N parameter to regenerate the audio clock in the receiver.

n[19:0], TX2 Main Map, *Address 0xF401[3:0]; Address 0xF402[7:0]; Address 0xF403[7:0]*
This signal is used to specifies the audio clock regeneration parameter N. This parameter is used with CTS to regenerate the audio clock in the receiver.

Table 61: Recommended N and Expected CTS Values for 32 kHz Audio

	32 kHz	
Pixel Clock (MHz)	N	CTS
25.2/1.001	4576	28125
25.2	4096	25200
27	4096	27000
27 * 1.001	4096	27027
54	4096	54000
54 * 1.001	4096	54054
74.25/1.001	11648	210937 – 210938
74.25	4096	74250

	32 kHz	
Pixel Clock (MHz)	**N**	**CTS**
148.5/1.001	11648	421875
148.5	4096	148500
Other	4096	Measured

Table 62: Recommended N and Expected CTS Values for 44.1 kHz and Multiples

	44.1kHz		88.2 kHz		176.4 kHz	
Pixel Clock (MHz)	**N**	**CTS**	**N**	**CTS**	**N**	**CTS**
25.2 / 1.001	7007	31250	14014	31250	28028	31250
25.2	6272	28000	12544	28000	25088	28000
27	6272	30000	12544	30000	25088	30000
27 * 1.001	6272	30030	12544	30030	25088	30030
54	6272	60000	12544	60000	25088	60000
54 * 1.001	6272	60060	12544	60060	25088	60060
74.25 / 1.001	17836	234375	35672	234375	71344	234375
74.25	6272	82500	12544	82500	25088	82500
148.5 / 1.001	8918	234375	17836	234375	35672	234375
148.5	6272	16500	12544	16500	25088	16500
Other	6272	Measured	12544	Measured	25088	Measured

Table 63: Recommended N and Expected CTS Values for 48 kHz and Multiples

	48 kHz		96 kHz		192 kHz	
Pixel Clock (MHz)	**N**	**CTS**	**N**	**CTS**	**N**	**CTS**
25.2 / 1.001	6864	28125	13728	28125	27456	28125
25.2	6144	25200	12288	25200	24576	25200
27	6144	27000	12288	27000	24576	27000
27 * 1.001	6144	27027	12288	27027	24576	27027
54	6144	54000	12288	54000	24576	54000
54 * 1.001	6144	54054	12288	54054	24576	54054
74.25 / 1.001	11648	140625	35672	140625	46592	140625
74.25	6144	74250	12288	74250	24576	74250
148.5 / 1.001	5824	140625	17836	140625	23296	140625
148.5	6144	148500	12288	148500	24576	148500
Other	6144	Measured	12288	Measured	24576	Measured

6.11.5. *Audio Sample Packets*

By setting audioif_cc[2:0] to a value greater then 2 (that is, 3 channel or more), the eight channel audio packet format will be used. The I2S can be routed to different subpackets using the following fields:

- subpkt0_l_src
- subpkt0_r_src
- subpkt1_l_src
- subpkt1_r_src
- subpkt2_l_src
- subpkt2_r_src
- subpkt3_l_src
- subpkt3_r_src

The audioif_ca[7:0] must be set to a speaker mapping that corresponds to the I2S input stream to subpacket routing. Using SPDIF has a default setting of two channels.

The audio packets use the channel status format conforming to the IEC 60958 specification. When the part is configured to receive an I2S stream, the information sent in the channel status fields is provided by the following fields:

- cr_bit
- a_info
- clk_acc
- category_code
- source_number
- word_length
- channel_status
- i2s_sf

Table 64 provides a mapping between the channel status bit encapsulated in the Audio Sample packets sent across the HDMI link to the downstream sink and corresponding ADV8005 fields located in the Tx Main register map. Note that the mapping shown in Table 64 is the only application for I2S modes 0, 1, 2 and 3 set via the i2s_format[1:0] field.

When the part is configured to receive an SPDIF stream, the channel status information is taken from the input SPDIF stream.

audioif_ca[**7:0**], TX2 Main Map, *Address 0xF476[7:0]*
This register is used to set the Speaker Mapping or placement (Audio InfoFrame).

Function

audioif_ca[7:0]	**Description**
00000000 (default)	Default value
xxxxxxxx	Speaker mapping

cr_bit, TX2 Main Map, *Address 0xF412[5]*
This bit is used to set the Channel Status Copyright Information. Refer to the IEC 60958-3 specification.

Function

cr_bit	**Description**
0 (default)	Copyright asserted
1	Copyright not asserted

a_info[**2:0**], TX2 Main Map, *Address 0xF412[4:2]*
This signal is used to set the Channel Status Emphasis information. Refer to the IEC 60958-3 specification.

Function

a_info[2:0]	**Description**
000 (default)	2 audio channels without pre-emphasis
001	2 audio channels with 50/15uS pre-emphasis
010	Reserved (for 2 audio channels with pre-emphasis)
011	Reserved (for 2 audio channels with pre-emphasis)
100-111	Reserved

clk_acc[**1:0**], TX2 Main Map, *Address 0xF412[1:0]*
This signal is used to set the Channel Status Clock Accuracy information. Refer to the IEC 60958-3 specification.

Function

clk_acc[1:0]	**Description**
00 (default)	level II - normal accuracy +/-1000 x 10^-6
01	level I - high accuracy +/- 50 x 10^-6
10	level III - variable pitch shifted clock
11	Reserved

category_code[**7:0**], TX2 Main Map, *Address 0xF413[7:0]*
This register is used to set the Channel Status Category Code. Refer to the IEC 60958-3 specification.

Function

category_code[7:0]	**Description**
00000000 (default)	Default value
xxxxxxxx	Channel Status category code

source_number[3:0], TX2 Main Map, *Address 0xF414[7:4]*
This signal is used to set the Channel Status source number.

Function

source_number[3:0]	**Description**
0000 (default)	Default value
xxxx	Channel Status source number

word_length[3:0], TX2 Main Map, *Address 0xF414[3:0]*
This signal is used to set the Channel Status Audio Word Length. Refer to the IEC 60958-3 specification.

Function

word_length[3:0]	**Description**
0000 (default)	Not specified
0001	Not specified
0010	16 bits
0011	20 bits
0100	18 bits
0101	22 bits
0110	Reserved
0111	Reserved
1000	19 bits
1001	23 bits
1010	20 bits
1011	24 bits
1100	17 bits
1101	21 bits
1110	Reserved
1111	Reserved

channel_status[1:0], TX2 Main Map, *Address 0xF412[7:6]*
This signal is used to set the Channel Status bits [1:0]. Set to 0b00 as specified in IEC60958-3. Refer to IEC60958-3 specification.

Function

channel_status[1:0]	**Description**
xx	Channel status bits 0 and 1

Table 64: I^2S Channel Status ADV8005 Register Map Location of Fixed Value

Channel Status Bit	**Channel Status Bit Name**	**Main Map Bit Location or Fixed Value**	**Main Map Bit Name or Fixed Value**
0	Consumer use	0xEC12[6]	channel_status[0]
1	Audio sample word	0xEC12[7]	channel_status[1]
2	Copyright	0xEC12[5]	cr_bit
3	Emphasis	0xEC12[2]	a_info[0]
4	Emphasis	0xEC12[3]	a_info[1]
5	Emphasis	0xEC12[4]	a_info[2]
6	Mode	0	0
7	Mode	0	0
8	Category code	0xEC13[0]	category_code[0]
9	Category code	0xEC13[1]	category_code[1]
10	Category code	0xEC13[2]	category_code[2]
11	Category code	0xEC13[3]	category_code[3]
12	Category code	0xEC13[4]	category_code[4]
13	Category code	0xEC13[5]	category_code[5]
14	Category code	0xEC13[6]	category_code[6]
15	Category code	0xEC13[7]	category_code[7]
16	Source number	0xEC14[4]	source_number[0]
17	Source number	0xEC14[5]	source_number[1]
18	Source number	0xEC14[6]	source_number[2]
19	Source number	0xEC14[7]	source_number[3]

Channel Status Bit	Channel Status Bit Name	Main Map Bit Location or Fixed Value	Main Map Bit Name or Fixed Value
20	Channel number	See Figure 106	See Figure 106
21	Channel number	See Figure 106	See Figure 106
22	Channel number	See Figure 106	See Figure 106
23	Channel number	See Figure 106	See Figure 106
24	Sampling frequency	0xEC15[4]	i2s_sf[0]
25	Sampling frequency	0xEC15[5]	i2s_sf[1]
26	Sampling frequency	0xEC15[6]	i2s_sf[2]
27	Sampling frequency	0xEC15[7]	i2s_sf[3]
28	Clock accuracy	0xEC12[0]	clk_acc[0]
29	Clock accuracy	0xEC12[1]	clk_acc[1]
30	Not defined	0	0
31	Not defined	0	0
32	Word length	0xEC14[0]	word_length[0]
33	Word length	0xEC14[1]	word_length[1]
34	Word length	0xEC14[2]	word_length[2]
35	Word length	0xEC14[3]	word_length[3]
36	Original sampling frequency	0	0
37	Original sampling frequency	0	0
38	Original sampling frequency	0	0
39	Original sampling frequency	0	0
40	CGMS-A	0	0
41	CGMS-A	0	0
42-191	Not defined	0	0

Figure 106 shows how the channel number bits 20 to 23 are set, based on the layout bit and bit sample_present.spX which indicates if subpacket X contains audio samples(s). The layout bit in the Audio Sample packet header and the sample_present.spX bit are determined based on the values programmed in the audioif_cc[2:0] field.

For example, if audioif_cc[2:0] is set to 0b001 which indicates stereo audio, the layout bit will be zero and all Audio Sample subpackets will contain information for channels 1 and 2. If audioif_cc[2:0] is set to 0b011, indicating four channels, the layout bit will be 1; sample_present.sp0 will be 1, sample_present.sp1 will be 1, sample_present.sp2 will be 0, and sample_present.sp2 will be 0.

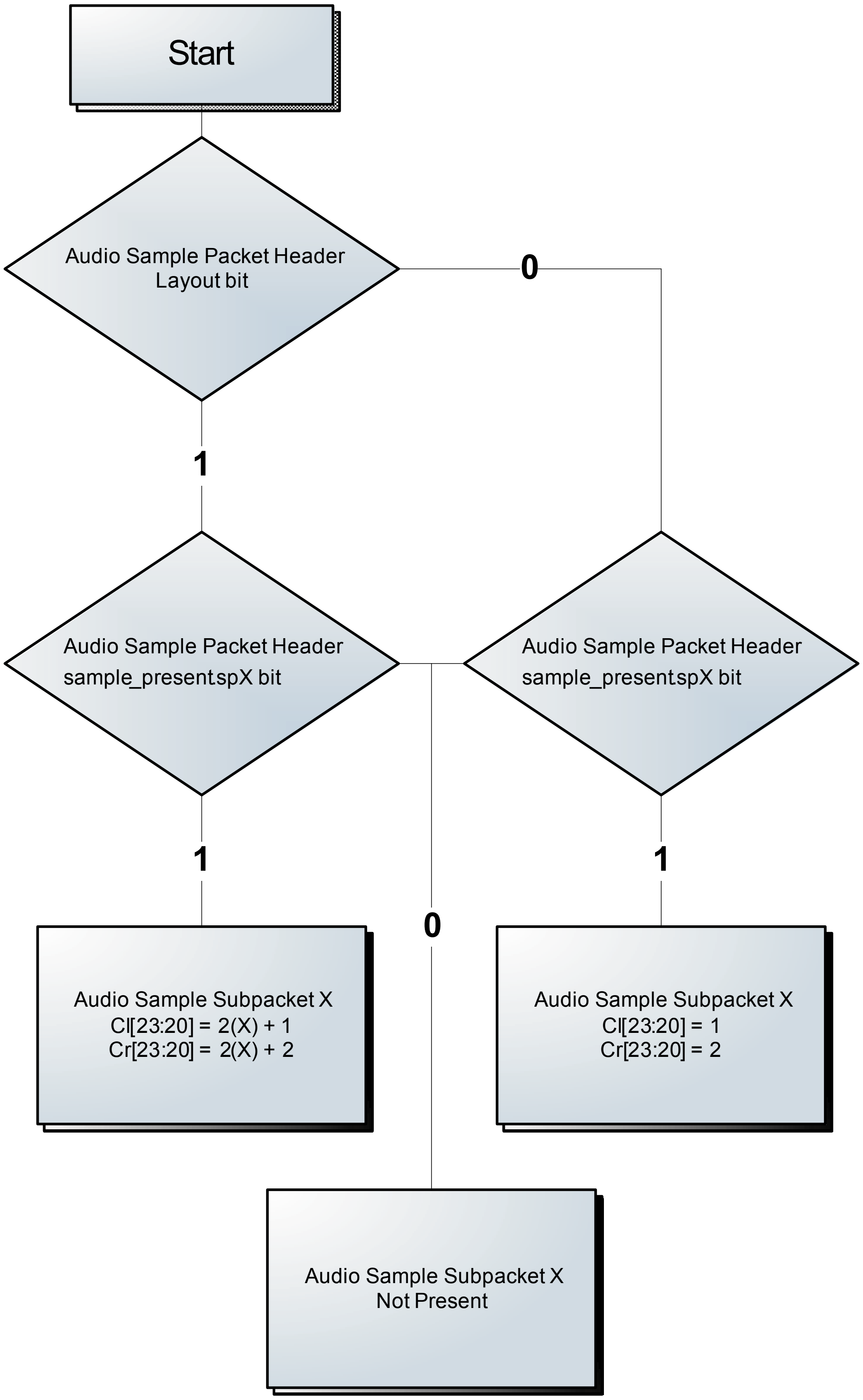


Figure 106: Definition of Channel Status Bits 20 to 23

6.11.6. *Audio InfoFrame*

The audio InfoFrame allows the sink to identify the characteristics of an audio stream before the channel status information is available.

The ADV8005 can be configured to transmit audio InfoFrame by setting audioif_pkt_en to 1. When the transmission of audio InfoFrame is enabled, the ADV8005 transmits an audio InfoFrame once every two video fields. Table 65 provides the list of registers that can be used to configure audio InfoFrames.

audioif_pkt_en, TX2 Main Map, *Address 0xF444[3]*
This bit is used to enable the Audio InfoFrame.

Function

audioif_pkt_en	Description
0	Disable audio InfoFrame
1 (default)	Enable audio InfoFrame

Table 65: Audio InfoFrame Configuration Registers

HDMI Tx Main Map Address	Bit Location	Access Type	Default Value	Field or Byte Name[1]
0xEC70	[2:0]	R/W	0b001	InfoFrame version number
0xEC71	[4:0]	R/W	0b01010	InfoFrame length
0xEC72	[7:0]	R/W	0b00000000	Checksum[2]
0xEC73	[7:0]	R/W	0b00000000	Data Byte 1
0xEC74	[7:0]	R/W	0b00000000	Data Byte 2
0xEC75	[7:0]	R/W	0b00000000	Data Byte 3
0xEC76	[7:0]	R/W	0b00000000	Data Byte 4
0xEC77	[7:0]	R/W	0b00000000	Data Byte 5
0xEC78	[7:0]	R/W	0b00000000	Data Byte 6
0xEC79	[7:0]	R/W	0b00000000	Data Byte 7
0xEC7A	[7:0]	R/W	00000000	Data Byte 8
0xEC7B	[7:0]	R/W	00000000	Data Byte 9
0xEC7C	[7:0]	R/W	00000000	Data Byte 10

[1] As defined in the latest CEA 861 specification
[2] Only used when auto_checksum_en = 0

6.11.7. *ACP Packet*

The Audio Content Protection (ACP) packet is used for transmitting content related information about the active audio stream. Using the ACP packet will be defined in the license agreement of the protected audio stream.

The contents of the ACP packet can be set via the set of Packet Map registers listed in Table 66. The user can enable the transmission of an ACP packet to the downstream sink by setting the acp_pkt_en bit. When the transmission of ACP packets is enabled, the ADV8005 transmits an APC packets once every two video fields.

acp_pkt_en, TX2 Main Map, *Address 0xF440[4]*
This bit is used to enable the ACP Packet.

Function

acp_pkt_en	Description
0 (default)	Disabled
1	Enabled

Table 66: ACP Packet Configuration Registers

Packet Map Address	Access Type	Field Name	Default Value	Byte Name[1]
0x40	R/W	acp_hb0[7:0]	0b00000000	Header Byte 0
0x41	R/W	acp_hb1[7:0]	0b00000000	Header Byte 1
0x42	R/W	acp_hb2[7:0]	0b00000000	Header Byte 2
0x43	R/W	acp_pb0[7:0]	0b00000000	Data Byte 0
0x44	R/W	acp_pb1[7:0]	0b00000000	Data Byte 1
0x45	R/W	acp_pb2[7:0]	0b00000000	Data Byte 2
0x46	R/W	acp_pb3[7:0]	0b00000000	Data Byte 3
0x47	R/W	acp_pb4[7:0]	0b00000000	Data Byte 4
0x48	R/W	acp_pb5[7:0]	0b00000000	Data Byte 5
0x49	R/W	acp_pb6[7:0]	0b00000000	Data Byte 6
0x4A	R/W	acp_pb7[7:0]	0b00000000	Data Byte 7
0x4B	R/W	acp_pb8[7:0]	0b00000000	Data Byte 8
0x4C	R/W	acp_pb9[7:0]	0b00000000	Data Byte 9
0x4D	R/W	acp_pb10[7:0]	0b00000000	Data Byte 10
0x4E	R/W	acp_pb11[7:0]	0b00000000	Data Byte 11
0x4F	R/W	acp_pb12[7:0]	0b00000000	Data Byte 12
0x50	R/W	acp_pb13[7:0]	0b00000000	Data Byte 13
0x51	R/W	acp_pb14[7:0]	0b00000000	Data Byte 14
0x52	R/W	acp_pb15[7:0]	0b00000000	Data Byte 15
0x53	R/W	acp_pb16[7:0]	0b00000000	Data Byte 16
0x54	R/W	acp_pb17[7:0]	0b00000000	Data Byte 17
0x55	R/W	acp_pb18[7:0]	0b00000000	Data Byte 18
0x56	R/W	acp_pb19[7:0]	0b00000000	Data Byte 19
0x57	R/W	acp_pb20[7:0]	0b00000000	Data Byte 20
0x58	R/W	acp_pb21[7:0]	0b00000000	Data Byte 21
0x59	R/W	acp_pb22[7:0]	0b00000000	Data Byte 22
0x5A	R/W	acp_pb23[7:0]	0b00000000	Data Byte 23
0x5B	R/W	acp_pb24[7:0]	0b00000000	Data Byte 24
0x5C	R/W	acp_pb25[7:0]	0b00000000	Data Byte 25
0x5D	R/W	acp_pb26[7:0]	0b00000000	Data Byte 26
0x5E	R/W	acp_pb27[7:0]	0b00000000	Data Byte 27

[1] As defined in the latest CEA 861 specification

6.11.8. *ISRC Packet*

If the Supports_AI bit in the Vendor Specific Data Block (VSDB) of the sink EDID is set at 1, the International Standard Recording Code (ISRC) packets 1 and 2 can be transmitted.

The ADV8005 can be configured to transmit ISRC packet by setting isrc_pkt_en to 1. When the transmission of an ISRC packet is enabled, the ADV8005 transmits an ISRC packet once every two video fields. Table 67 and Table 68 provide the list of registers that can be used to configure ISRC packets.

isrc_pkt_en, TX2 Main Map, *Address 0xF440[3]*
This bit is used to enable the ISRC Packet.

Function

isrc_pkt_en	Description
0 (default)	Disabled
1	Enabled

Table 67: ISRC1 Packet Configuration Registers

Packet Map Address	Access Type	Field Name	Default Value	Byte Name[1]
0xF260	R/W	isrc1_hb0[7:0]	0b00000000	Header Byte 0
0xF261	R/W	isrc1_hb1[7:0]	0b00000000	Header Byte 1
0xF262	R/W	isrc1_hb2[7:0]	0b00000000	Header Byte 2
0xF263	R/W	isrc1_pb0[7:0]	0b00000000	Data Byte 0
0xF264	R/W	isrc1_pb1[7:0]	0b00000000	Data Byte 1
0xF265	R/W	isrc1_pb2[7:0]	0b00000000	Data Byte 2
0xF266	R/W	isrc1_pb3[7:0]	0b00000000	Data Byte 3
0xF267	R/W	isrc1_pb4[7:0]	0b00000000	Data Byte 4
0xF268	R/W	isrc1_pb5[7:0]	0b00000000	Data Byte 5
0xF269	R/W	isrc1_pb6[7:0]	0b00000000	Data Byte 6
0xF26A	R/W	isrc1_pb7[7:0]	0b00000000	Data Byte 7
0xF26B	R/W	isrc1_pb8[7:0]	0b00000000	Data Byte 8
0xF26C	R/W	isrc1_pb9[7:0]	0b00000000	Data Byte 9
0xF26D	R/W	isrc1_pb10[7:0]	0b00000000	Data Byte 10
0xF26E	R/W	isrc1_pb11[7:0]	0b00000000	Data Byte 11
0xF26F	R/W	isrc1_pb12[7:0]	0b00000000	Data Byte 12
0xF270	R/W	isrc1_pb13[7:0]	0b00000000	Data Byte 13
0xF271	R/W	isrc1_pb14[7:0]	0b00000000	Data Byte 14
0xF272	R/W	isrc1_pb15[7:0]	0b00000000	Data Byte 15
0xF273	R/W	isrc1_pb16[7:0]	0b00000000	Data Byte 16
0xF274	R/W	isrc1_pb17[7:0]	0b00000000	Data Byte 17
0xF275	R/W	isrc1_pb18[7:0]	0b00000000	Data Byte 18
0xF276	R/W	isrc1_pb19[7:0]	0b00000000	Data Byte 19
0xF277	R/W	isrc1_pb20[7:0]	0b00000000	Data Byte 20
0xF278	R/W	isrc1_pb21[7:0]	0b00000000	Data Byte 21
0xF279	R/W	isrc1_pb22[7:0]	0b00000000	Data Byte 22
0xF27A	R/W	isrc1_pb23[7:0]	0b00000000	Data Byte 23
0xF27B	R/W	isrc1_pb24[7:0]	0b00000000	Data Byte 24
0xF27C	R/W	isrc1_pb25[7:0]	0b00000000	Data Byte 25
0xF27D	R/W	isrc1_pb26[7:0]	0b00000000	Data Byte 26
0xF27E	R/W	isrc1_pb27[7:0]	0b00000000	Data Byte 27

[1] As defined in the latest CEA 861 specification

Table 68: ISRC2 Packet Configuration Registers

Packet Map Address	Access Type	Field Name	Default Value	Byte Name[1]
0xF280	R/W	isrc2_hb0[7:0]	0b00000000	Header Byte 0
0xF281	R/W	isrc2_hb1[7:0]	0b00000000	Header Byte 1
0xF282	R/W	isrc2_hb2[7:0]	0b00000000	Header Byte 2
0xF283	R/W	isrc2_pb0[7:0]	0b00000000	Data Byte 0
0xF284	R/W	isrc2_pb1[7:0]	0b00000000	Data Byte 1
0xF285	R/W	isrc2_pb2[7:0]	0b00000000	Data Byte 2
0xF286	R/W	isrc2_pb3[7:0]	0b00000000	Data Byte 3
0xF287	R/W	isrc2_pb4[7:0]	0b00000000	Data Byte 4
0xF288	R/W	isrc2_pb5[7:0]	0b00000000	Data Byte 5
0xF289	R/W	isrc2_pb6[7:0]	0b00000000	Data Byte 6
0xF28A	R/W	isrc2_pb7[7:0]	0b00000000	Data Byte 7
0xF28B	R/W	isrc2_pb8[7:0]	0b00000000	Data Byte 8
0xF28C	R/W	isrc2_pb9[7:0]	0b00000000	Data Byte 9
0xF28D	R/W	isrc2_pb10[7:0]	0b00000000	Data Byte 10
0xF28E	R/W	isrc2_pb11[7:0]	0b00000000	Data Byte 11
0xF28F	R/W	isrc2_pb12[7:0]	0b00000000	Data Byte 12

Packet Map Address	Access Type	Field Name	Default Value	Byte Name[1]
0xF290	R/W	isrc2_pb13[7:0]	0b00000000	Data Byte 13
0xF291	R/W	isrc2_pb14[7:0]	0b00000000	Data Byte 14
0xF292	R/W	isrc2_pb15[7:0]	0b00000000	Data Byte 15
0xF293	R/W	isrc2_pb16[7:0]	0b00000000	Data Byte 16
0xF294	R/W	isrc2_pb17[7:0]	0b00000000	Data Byte 17
0xF295	R/W	isrc2_pb18[7:0]	0b00000000	Data Byte 18
0xF296	R/W	isrc2_pb19[7:0]	0b00000000	Data Byte 19
0xF297	R/W	isrc2_pb20[7:0]	0b00000000	Data Byte 20
0xF298	R/W	isrc2_pb21[7:0]	0b00000000	Data Byte 21
0xF299	R/W	isrc2_pb22[7:0]	0b00000000	Data Byte 22
0xF29A	R/W	isrc2_pb23[7:0]	0b00000000	Data Byte 23
0xF29B	R/W	isrc2_pb24[7:0]	0b00000000	Data Byte 24
0xF29C	R/W	isrc2_pb25[7:0]	0b00000000	Data Byte 25
0xF29D	R/W	isrc2_pb26[7:0]	0b00000000	Data Byte 26
0xF29E	R/W	isrc2_pb27[7:0]	0b00000000	Data Byte 27

[1] As defined in the latest CEA 861 specification

6.12. EDID HANDLING

6.12.1. *Reading the EDID*

The Tx core of the ADV8005 features an EDID/HDCP controller which can read the EDID content of the downstream sink through the DDC lines, TXDDC_SCL and TXDDC_SDA. This EDID/HDCP controller begins buffering segment 0 of the downstream sink EDID once the sink HPD is detected and the Tx core of the ADV8005 is powered up. The system can request additional segments by programming the EDID segment pointer edid_segment[7:0]. edid_ready_int (refer to Section 6.8) indicates that a 256-byte EDID read has been completed, and the EDID content can be read from the EDID Map.

edid_segment[7:0], TX2 Main Map, *Address 0xF4C4[7:0]*
This register is used to set the segment of the EDID read from the downstream receiver.

Function

edid_segment[7:0]	Description
xxxxxxxx	User programmed EDID segment value

6.12.2. *EDID Definitions*

Extended EDID (E-EDID) supports up to 256 segments. A segment is a 256-byte segment of EDID data containing one or two 128-byte EDID blocks. A typical HDMI sink will have only two EDID blocks and so will only use segment 0. The first EDID block is always a base EDID structure defined in the VESA EDID specifications; the second EDID block is usually the CEA extension defined in the CEA-861 specification.

The ADV8005 has a single memory location used to store EDID and HDCP information read from the downstream sink. During HDCP repeater initialization, the EDID data read from the sink is overwritten with HDCP information which is also read from the sink. The sink EDID is not reread after HDCP initialization. The user can request the ADV8005 to rebuffer an EDID segment by using the edid_reread control.

6.12.3. *Additional Segments*

The EDID block 0 byte number 0x7E tells how many additional EDID blocks are available. If byte 0x7E is greater than 1, additional EDID segments will need to be read. If there is more than one segment, the second block (that is, block 1) is required to be an EDID extension map. This map should be parsed according to the VESA EDID specification to determine where additional EDID blocks are stored in the sink EDID storage device such as EEPROM, RAM, and so on.

The ADV8005 is capable of accessing up to 256 segments from EDID of the sink as allowed by the EDID specification. By writing the desired segment number to the edid_segment[7:0] field, the ADV8005 will automatically access the correct portion of the sink EDID over the Tx DDC lines and load the 256 bytes into the EDID/HDCP memory. When the action is complete, the ADV8005 triggers the edid_ready_int interrupt (refer to Section 6.8). The EDID data read from the sink can then be accessed from the Tx EDID Map. If the host controller needs access to previously requested EDID information, then it can be stored in its own memory.

Figure 107 shows how to implement software to read EDID from the downstream sink using the ADV8005.

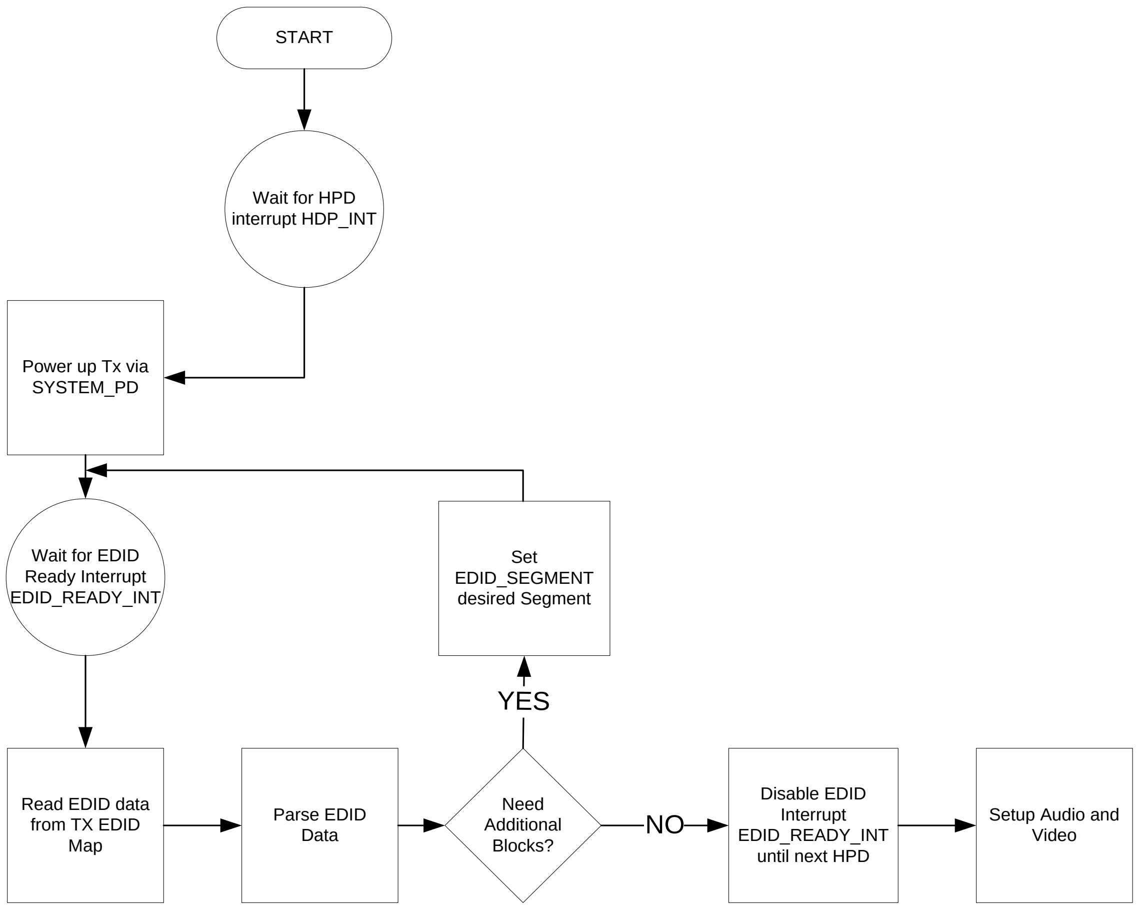

START
Wait for HPD interrupt HDP_INT
Power up Tx via SYSTEM_PD
Wait for EDID Ready Interrupt EDID_READY_INT
Set EDID_SEGMENT desired Segment
YES
Read EDID data from TX EDID Map
Parse EDID Data
Need Additional Blocks?
NO
Disable EDID Interrupt EDID_READY_INT until next HPD
Setup Audio and Video

Figure 107: Reading Sink EDID Through ADV8005

6.12.4. *edid_tries Control*

edid_tries[3:0] can be used to set the number of times the Tx EDID/HDCP controller will try to read the sink EDID after a failure. Each time an EDID read fails with an I^{2}C Not Acknowledged (NACK), this value of edid_tries[3:0] is decremented. Once the edid_tries[3:0] reaches the value 0, the Tx EDID/HDCP controller will not attempt to read the EDID until edid_tries[3:0] is set to a value other than 0. This could be used if a sink asserts high its HPD signal before the DDC bus is ready, resulting in several NACKs as the ADV8005 attempts to read the EDID.

edid_tries[3:0], TX2 Main Map, *Address 0xF4C9[3:0]*
This signal is used to control the number of times that the EDID read will be attempted if unsuccessful.

Function

edid_tries[3:0]	**Description**
xxxx	Number of time the EDID/HDCP controller attempts to read the EDID

6.12.5. *EDID Reread Control*

If the EDID data from the sink is read in and the host determines that the data needs to be reread, edid_reread can be set from 0 to 1, and the current segment set via edid_segment[7:0] will be reread. Rereading the sink EDID may be useful, for example, if the host finds that one EDID checksum read from the sink is invalid.

Note: It is also possible to reread the EDID from the sink by toggling the Tx core power down system_pd from 0 to 1.

edid_reread, TX2 Main Map, *Address 0xF4C9[4]*
This bit is used to request a the EDID controller to reread the current segment if toggled from 0 to 1 for 10 times consecutively.

Function

edid_reread	Description
0 (default)	No action
1	Request the EDID/HDCP controller to read the EDID

6.13. HDCP HANDLING

6.13.1. *One Sink and No Upstream Devices*

The ADV8005 has a built-in controller, the Tx EDID/HDCP controller which handles HDCP transmitter states, including handling downstream HDCP repeaters. To activate HDCP from a system level, the host controller needs to set hdcp_desired to 1 and frame_encryption_en to 1. This informs the ADV8005 that the video stream it outputs should be encrypted. The ADV8005 takes control from there and implements all the remaining tasks defined by the HDCP 1.4 specification.

Before sending audio and video, the BKSV of the downstream sink should be compared with the revocation list which is compiled by managing System Renewability Messages (SRMs) provided on the source content (for example. DVD, Blue-ray Disc), and the bksv_flag_int interrupt bit should be cleared. After the HDCP link is established between the ADV8005 and the downstream sink, the system controller should monitor the status of HDCP by reading enc_on every two seconds. The Tx EDID/HDCP controller error interrupt will activate and hdcp_error_int will be set to 1 if there is an error relating to the controller. The meaning of the error can be determined by checking hdcp_controller_error[3:0].

bksv_flag_int, TX2 Main Map, *Address 0xF497[6]*
This bit is used to readback and control the BKSV Flag interrupt.

Function

bksv_flag_int	Description
0 (default)	Interrupt not active
1	Interrupt active. The KSVs from the downstream sink have been read and available in the Memory Map

hdcp_desired, TX2 Main Map, *Address 0xF4AF[7]*
This bit is used to request HDCP encryption.

Function

hdcp_desired	Description
0 (default)	Input audio and video content not to be encrypted
1	The input audio and video content should be encrypted

frame_encryption_en, TX2 Main Map, *Address 0xF4AF[4]*
This bit is used to request HDCP frame encryption.

Function

frame_encryption_en	Description
0	Current video frame should not be encrypted
1 «	Current video frame should be encrypted

bksv[39:32], TX2 Main Map, *Address 0xF4C3[7:0] (Read Only)*
This register is used to readback the BKSV Byte 4 read from the downstream receiver by the HDCP controller.

enc_on, TX2 Main Map, *Address 0xF4B8[6] (Read Only)*
This bit is used to readback the HDCP encryption status.

Function

enc_on	Description
0 (default)	The audio and video content is not being encrypted
1	The audio and video content is being encrypted

6.13.2. *Multiple Sinks and No Upstream Devices*

When connecting the ADV8005 as a source to an HDMI input of a repeater, it is necessary to read all BKSVs from downstream devices. These BKSVs must be checked against a revocation list, which will be provided on the source content.

bksv_count[6:0] will read 0 when the first BKSV interrupt occurs with bksv_flag_int set to 1. After the first BKSV interrupt is cleared, if the sink connected to the ADV8005 is a repeater, a second BKSV interrupt will occur. The ADV8005 will automatically read up to 13 5-byte BKSVs at a time and store these in the EDID memory. These BKSVs can be accessed from the EDID Map, as shown in Table 69. The number of additional BKSVs available stored in the EDID Map can be obtained from bksv_count[6:0]. If there are more than 13 additional BKSVs to be processed, the ADV8005 will collect the next up to 13 BKSVs from the sink, then generate another BKSV interrupt with bksv_flag_int set to 1 when the next set is ready.

Table 69: KSV Fields Accessed From EDID Map

KSV Number	Field Name	Register Addresses
0	bksv0_byte_0[7:0] bksv0_byte_1[7:0] bksv0_byte_2[7:0] bksv0_byte_3[7:0] bksv0_byte_4[7:0]	0xEE00[7:0] byte 0 0xEE01[7:0] byte 1 0xEE02[7:0] byte 2 0xEE03[7:0] byte 3 0xEE04[7:0] byte 4
1	bksv1_byte_0[7:0] bksv1_byte_1[7:0] bksv1_byte_2[7:0] bksv1_byte_3[7:0] bksv1_byte_4[7:0]	0xEE05[7:0] byte 0 0xEE06[7:0] byte 1 0xEE07[7:0] byte 2 0xEE08[7:0] byte 3 0xEE09[7:0] byte 4
2	bksv2_byte_0[7:0] bksv2_byte_1[7:0] bksv2_byte_2[7:0] bksv2_byte_3[7:0] bksv2_byte_4[7:0]	0xEE0A[7:0] byte 0 0xEE0B[7:0] byte 1 0xEE0C[7:0] byte 2 0xEE0D[7:0] byte 3 0xEE0E [7:0] byte 4
3	bksv3_byte_0[7:0] bksv3_byte_1[7:0] bksv3_byte_2[7:0] bksv3_byte_3[7:0] bksv3_byte_4[7:0]	0xEE0F[7:0] byte 0 0xEE10[7:0] byte 1 0xEE11[7:0] byte 2 0xEE12[7:0] byte 3 0xEE13[7:0] byte 4
4	bksv4_byte_0[7:0] bksv4_byte_1[7:0] bksv4_byte_2[7:0] bksv4_byte_3[7:0] bksv4_byte_4[7:0]	0xEE14[7:0] byte 0 0xEE15[7:0] byte 1 0xEE16[7:0] byte 2 0xEE17[7:0] byte 3 0xEE18[7:0] byte 4
5	bksv5_byte_0[7:0] bksv5_byte_1[7:0] bksv5_byte_2[7:0] bksv5_byte_3[7:0] bksv5_byte_4[7:0]	0xEE19[7:0] byte 0 0xEE1A[7:0] byte 1 0xEE1B[7:0] byte 2 0xEE1C[7:0] byte 3 0xEE1D[7:0] byte 4
6	bksv6_byte_0[7:0] bksv6_byte_1[7:0] bksv6_byte_2[7:0] bksv6_byte_3[7:0] bksv6_byte_4[7:0]	0xEE1E[7:0] byte 0 0xEE1F[7:0] byte 1 0xEE20[7:0] byte 2 0xEE21[7:0] byte 3 0xEE22[7:0] byte 4
7	bksv7_byte_0[7:0] bksv7_byte_1[7:0] bksv7_byte_2[7:0] bksv7_byte_3[7:0] bksv7_byte_4[7:0]	0xEE23[7:0] byte 0 0xEE24[7:0] byte 1 0xEE25[7:0] byte 2 0xEE26[7:0] byte 3 0xEE27[7:0] byte 4

KSV Number	Field Name	Register Addresses
8	bksv8_byte_0[7:0] bksv8_byte_1[7:0] bksv8_byte_2[7:0] bksv8_byte_3[7:0] bksv8_byte_4[7:0]	0xEE28[7:0] byte 0 0xEE29[7:0] byte 1 0xEE2A[7:0] byte 2 0xEE2B[7:0] byte 3 0xEE2C[7:0] byte 4
9	bksv9_byte_0[7:0] bksv9_byte_1[7:0] bksv9_byte_2[7:0] bksv9_byte_3[7:0] bksv9_byte_4[7:0]	0xEE2D[7:0] byte 0 0xEE2E[7:0] byte 1 0xEE2F[7:0] byte 2 0xEE30[7:0] byte 3 0xEE31[7:0] byte 4
10	bksv10_byte_0[7:0] bksv10_byte_1[7:0] bksv10_byte_2[7:0] bksv10_byte_3[7:0] bksv10_byte_4[7:0]	0xEE32[7:0] byte 0 0xEE33[7:0] byte 1 0xEE34[7:0] byte 2 0xEE35[7:0] byte 3 0xEE36[7:0] byte 4
11	bksv11_byte_0[7:0] bksv11_byte_1[7:0] bksv11_byte_2[7:0] bksv11_byte_3[7:0] bksv11_byte_4[7:0]	0xEE37[7:0] byte 0 0xEE38[7:0] byte 1 0xEE39[7:0] byte 2 0xEE3A[7:0] byte 3 0xEE3B[7:0] byte 4
12	bksv12_byte_0[7:0] bksv12_byte_1[7:0] bksv12_byte_2[7:0] bksv12_byte_3[7:0] bksv12_byte_4[7:0]	0xEE3C[7:0] byte 0 0xEE3D[7:0] byte 1 0xEE3E[7:0] byte 2 0xEE3F[7:0] byte 3 0xEE40[7:0] byte 4

The BKVS interrupt bit bksv_flag_int set to 1 should be cleared by setting bksv_flag_int to 1 after each set of BKSVs is read. To check when authentication is complete, the system should monitor hdcp_controller_state[3:0] and wait until this field reaches the value or state 4. At this time, the last host controller should be used to compare the BKSV list read from the sink with the revocation list. Once the host controller has verified none of the BKSVs read from the sink are revoked, the ADV8005 can be configured to send content down to the sink.

bksv_count[6:0], TX2 Main Map, *Address 0xF4C7[6:0] (Read Only)*
This signal is used to specify the total number of downstream HDCP devices.

Function

bksv_count[6:0]	Description
xxxxxxx	Total number of downstream HDCP devices

6.13.3. *Software Implementation*

Figure 108 shows a block diagram of HDCP software implementation for all cases using the ADV8005 Tx HDCP/EDID controller state machine. The necessary interactions with the ADV8005 registers and EDID memory, as well as when these interactions should take place, are illustrated in the diagram. Note that there is no need to interact with the DDC bus directly because all of the DDC functionality is controlled by the Tx HDCP/EDID controller and follows the HDCP specification 1.4.

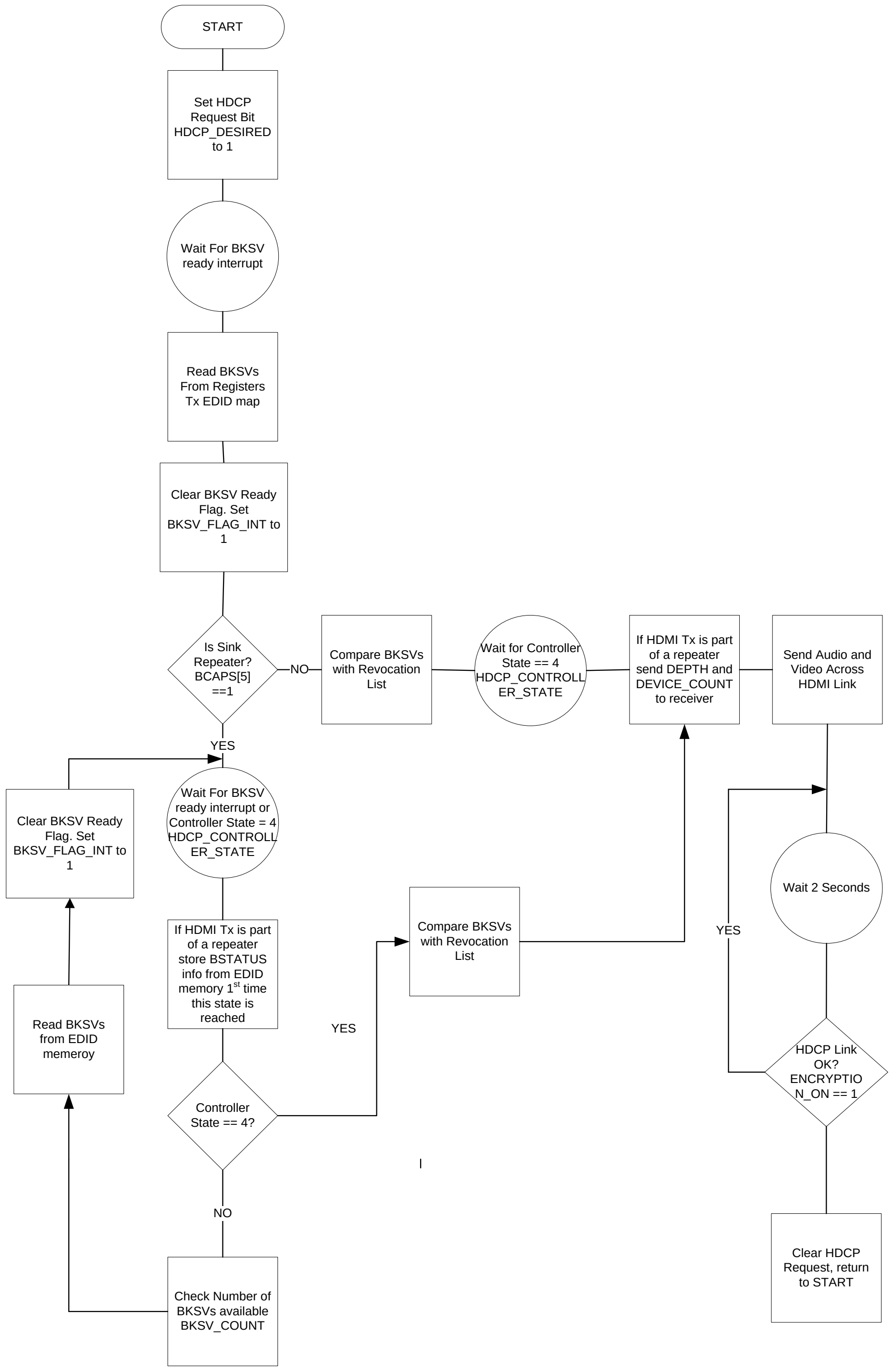

START
Set HDCP Request Bit HDCP_DESIRED to 1
Wait For BKSV ready interrupt
Read BKSVs From Registers Tx EDID map
Clear BKSV Ready Flag. Set BKSV_FLAG_INT to 1
Is Sink Repeater? BCAPS[5] ==1
NO
Compare BKSVs with Revocation List
Wait for Controller State == 4 HDCP_CONTROLLER_STATE
If HDMI Tx is part of a repeater send DEPTH and DEVICE_COUNT to receiver
Send Audio and Video Across HDMI Link
YES
Clear BKSV Ready Flag. Set BKSV_FLAG_INT to 1
Wait For BKSV ready interrupt or Controller State = 4 HDCP_CONTROLLER_STATE
If HDMI Tx is part of a repeater store BSTATUS info from EDID memory 1st time this state is reached
Compare BKSVs with Revocation List
YES
Wait 2 Seconds
YES
Read BKSVs from EDID memeroy
Controller State == 4?
HDCP Link OK? ENCRYPTION_ON == 1
NO
Check Number of BKSVs available BKSV_COUNT
Clear HDCP Request, return to START

Figure 108: HDCP Software Implementation

6.13.4. *AV Mute*

AV mute can be enabled once HDCP authentication is completed between the ADV8005 and the downstream sink. This can be used to maintain HDCP synchronization while changing video resolutions. While the KSVs for the downstream devices are being collected, an active HDCP link capable of sending encrypted video is established, but video should not be sent across the link until the KSVs have been compared with the revocation list.

It is not recommended to rely on AV mute to avoid sending audio and video during HDCP authentication. This is because AV mute does not actually mute audio or video in the Tx. It requests the function from the sink device. The best way to avoid sending unauthorized audio and video is to not send data to the Tx core of the ADV8005 until authentication between the ADV8005 and the downstream sink is complete. Another option is to black out the video data input to the Tx core and disable the audio inputs to mute the audio. Refer to Section 6.4 for an explanation of how to enable AV mute. Refer to Section 6.11 for an explanation of how to disable the various audio inputs.

6.14. AUDIO RETURN CHANNEL

The ADV8005 features an Audio Return Channel (ARC) Rx in each HDMI Tx that supports the extraction of an SPDIF stream from the ARC component of an HDMI Ethernet and Audio Channel (HEAC) signal output by a downstream sink. The ADV8005 can process the HEAC signal output by the downstream sink in only common mode.

The ARC Rxs are powered up by default but can be powered down using the tx1_arc_powerdown and tx2_arc_powerdown bits. The ARC pins are disabled by default and must be manually enabled to configure the ADV8005 to output ARC audio. The pins can be manually enabled by setting both arc_pins_oe_man and arc_pins_oe_man_en to 1. The SPDIF signal extracted by the ARC Rx can be output on the ARC1_OUT and ARC2_OUT pins.

tx1_arc_s_end_hpd and tx2_arc_s_end_hpd must both be left at the default value (1’b0) at all times – regardless of whether single-ended or common-mode ARC is being received.

tx1_arc_powerdown, IO Map, *Address 0x1A87[7]*
This bit is used to powerdown the TX1 ARC block.

Function

tx1_arc_powerdown	**Description**
0 (default)	Power up ARC
1	Power down ARC

tx2_arc_powerdown, IO Map, *Address 0x1A89[7]*
This bit is used to powerdown the TX2 ARC block.

Function

tx2_arc_powerdown	**Description**
0 (default)	Power up ARC
1	Power down ARC

arc_pins_oe_man, IO Map, *Address 0x1ACA[7]*
This bit is used to control the output enable for ARC outputs.

Function

arc_pins_oe_man	**Description**
0 (default)	Input
1	Output

arc_pins_oe_man_en, IO Map, *Address 0x1ACB[7]*
This bit is used to control the manual override for ARC outputs.

Function

arc_pins_oe_man_en	**Description**
0 (default)	Auto
1	Manual override

To increase the noise immunity of the ADV8005 ARC Rxs, it is recommended to enable the input hysteresis block on both blocks via tx1_arc_bias_hyst_adj and tx2_arc_bias_hyst_adj.

tx1_arc_bias_hyst_adj, IO Map, *Address 0x1A88[1]*
This bit is used to control the addition of hysteresis to the TX1 ARC.

Function

tx1_arc_bias_hyst_adj	**Description**
0 (default)	Normal
1	ADD hysteresis

tx2_arc_bias_hyst_adj, IO Map, *Address 0x1A8A[1]*
This bit is used to control the addition of hysteresis to the TX2 ARC.

Function

tx2_arc_bias_hyst_adj	**Description**
0 (default)	Normal
1	ADD hysteresis

6.15. CHARGE INJECTION SETTINGS

The clock and data charge injection controls are used to tune the strength of an AC-coupled driver on the outputs from the ADV8005 Tx. This driver is used to boost the ramp rate on the output waveform, helping to open the eye particularly at higher transmission speeds. The charge injection settings used in the ADV8005 software driver are optimized for the evaluation board and may require
adjustment for end-user systems. The three data channels should be configured to the same value charge injection setting. The clock charge injection value may require adjustment to a separate value to meet rise / fall time requirements.
chg_inj_ch0[3:0], TX2 Main Map, *Address 0xF481[7:4]*
Binary control of charge injection for data channel 0 with LSB cap value of 77fF
chg_inj_ch1[3:0], TX2 Main Map, *Address 0xF481[3:0]*
Binary control of charge injection for data channel 1 with LSB cap value of 77fF
chg_inj_ch2[3:0], TX2 Main Map, *Address 0xF482[7:4]*
Binary control of charge injection for data channel 2 with LSB cap value of 77fF
chg_inj_clk[3:0], TX2 Main Map, *Address 0xF482[3:0]*
Binary control of charge injection for clock channel with LSB cap value of 77fF

6.16. ENABLING AND DISABLING THE HDMI TMDS OTUPUTS

The clock and data predriver controls are used to enable or disable the current switching outputs from the ADV8005 HDMI Tx.

pre_en_ch0, TX2 Main Map, *Address 0xF480[3]*
Enable data channel 0

Function

pre_en_ch0	**Description**	
1 (default)	Enabled	0 = Disabled

pre_en_ch1, TX2 Main Map, *Address 0xF480[2]*
Enable data channel 1

Function

pre_en_ch1	**Description**	
1 (default)	Enabled	0 = Disabled

pre_en_ch2, TX2 Main Map, *Address 0xF480[1]*
Enable data channel 2

Function

pre_en_ch2	**Description**	
1 (default)	Enabled	0 = Disabled

pre_en_clk, TX2 Main Map, *Address 0xF480[0]*
Enable clock channel

Function

pre_en_clk	**Description**	
1 (default)	Enabled	0 = Disabled

To disable a TMDS output, it is recommended to follow this sequence:

1. Disable the charge injection for the channel
2. Disable the predriver for the channel

To enable a TMDS output, the opposite sequence should be followed:

1. Enable the predriver for the channel
2. Enable the required charge injection for the channel

In summary, the charge injection should only be enabled while the predriver is also enabled.

6.17. HDMI TX SOURCE TERMINATION

When an ADV8005 HDMI Tx output is connected to a sink device, the capabilities of the sink device's receiver must first be considered. If the sink device is limited to receiving a maximum TMDS clock frequency less than or equal to 165 MHz, the source termination must be disabled in the ADV8005 Tx connected to that sink device.

If the sink device can receive a TMDS clock frequency above 165 MHz, then the TMDS clock frequency of the ADV8005 HDMI Tx connected to that sink will dictate what source termination settings are used. In this case, the Tx source termination settings must be configured as follows:

- ADV8005 Tx source termination disabled if the ADV8005 Tx TMDS clock frequency is less than or equal to 165 MHz
- ADV8005 Tx source termination enabled if the ADV8005 Tx TMDS clock frequency is greater than 165 MHz

Therefore, for 4k x 2k, Tx source termination should be enabled on the ADV8005 HDMI Tx output. To disable the source termination, a manual over-ride must be enabled. The manual over-ride value is an open-circuit if the control is set to 0.

Figure 109 provides an overview of the ADV8005 HDMI Tx source termination requirements.

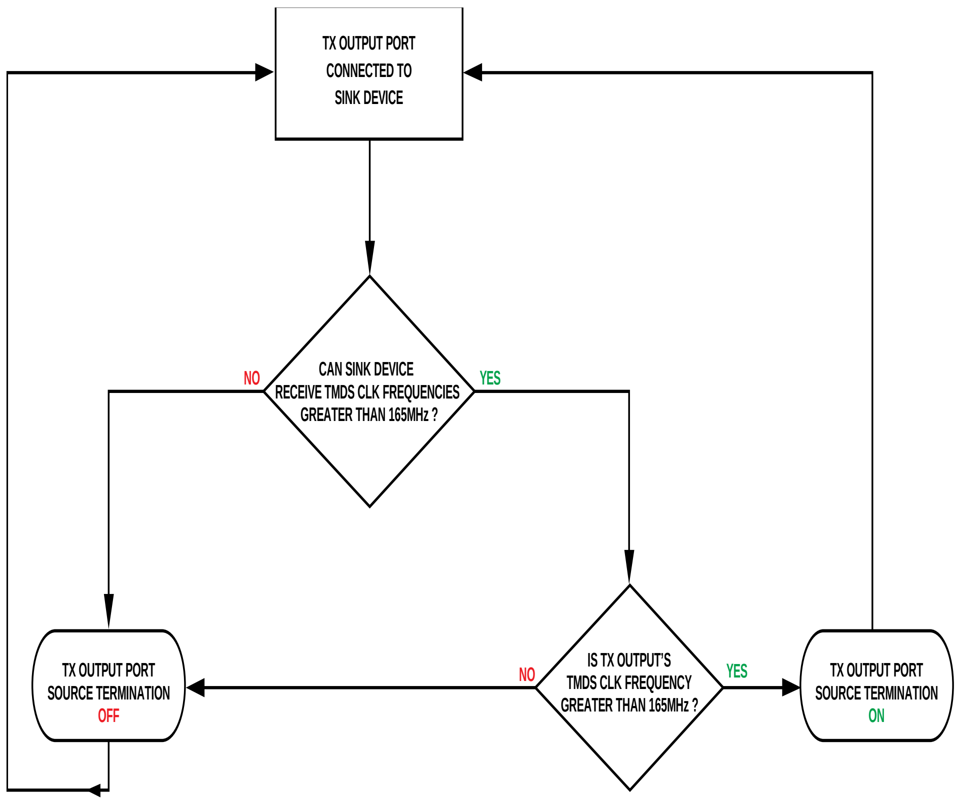

TX OUTPUT PORT CONNECTED TO SINK DEVICE
CAN SINK DEVICE RECEIVE TMDS CLK FREQUENCIES GREATER THAN 165MHz ?
NO
YES
IS TX OUTPUT'S TMDS CLK FREQUENCY GREATER THAN 165MHz ?
NO
YES
TX OUTPUT PORT SOURCE TERMINATION OFF
TX OUTPUT PORT SOURCE TERMINATION ON

Figure 109: ADV8005 Tx Source Termination Requirements

6.18. HDMI ACR PACKET TRANSMISSION

A mode has been added to the HDMI Tx to ensure more efficient transmission of audio samples in 176.4 kHz and 192 kHz modes. This ensures ACR packets can get sent more frequently on the ADV8005 for these modes than is the case for the equivalent modes on the ADV8003.

arc_eff_tran_en, TX2 Main Map, *Address 0xF447[0]*
When enable it ensures more efficient transmission of ARC packets and audio samples in 176.4kHz and 192kHz modes. This ensures ACR packets can get sent at the right rate (~1ms).

Function

arc_eff_tran_en	**Description**
0	ARC packet efficient transmision disable.
1 (default)	ARC packet efficient transmition enable.

7. VIDEO ENCODER INTRODUCTION TO THE ADV8005

7.1. INTRODUCTION

The ADV8005 encoder core consists of six high speed, Noise Shaped Video (NSV), 12-bit video DACs which provide support for composite (CVBS), S-Video (Y-C), and component (YPrPb/RGB) analog outputs in standard definition (SD), enhanced definition (ED), or high definition (HD) video formats.

Simultaneous SD and ED/HD input and output modes are supported. The ADV8005 encoder processor provides two independent signal paths for SD and ED/HD, so different video processing (filtering, color conversion, and so on) can be individually and simultaneously applied to each of the streams.

The input to the SD encoder block is always a 16/20/24-bit 4:2:2 YCbCr stream, and a 24/30/36-bit 4:4:4 YCbCr stream for ED/HD modes. Although the encoder cannot take an RGB input stream in, it features a CSC matrix which enables the generation of RGB video signals at the component output.

The oversampling at 216 MHz (SD and ED) and 297 MHz (HD) ensures that external output filtering is not required. The block diagram for the ADV8005 encoder core is shown in Figure 110.

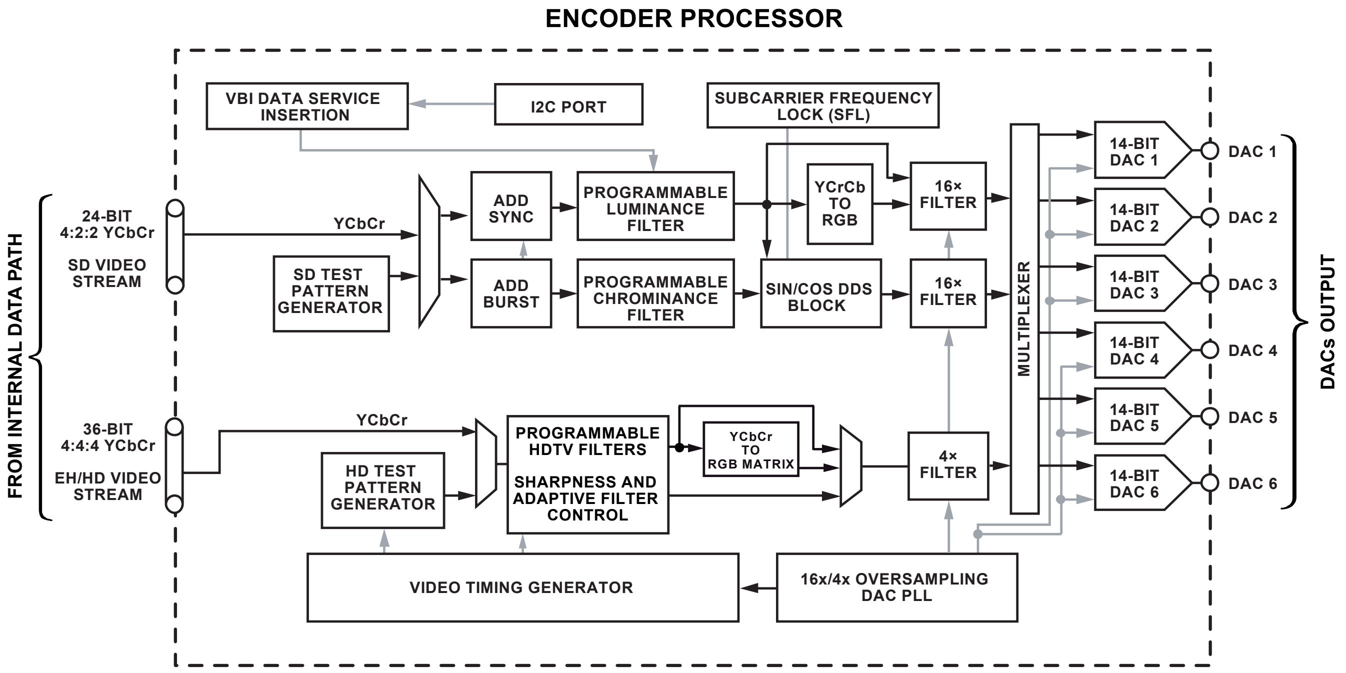


Figure 110: ADV8005 Encoder Block Diagram

Note: The video encoder variants of the ADV8005 are ADV8005KBCZ-8A/8N. The variants of ADV8005 with no encoder are ADV8005KBCZ-8B/8C.

7.2. INPUT CONFIGURATION

The ADV8005 encoder core is capable of supporting independent SD and ED/HD video outputs, and also both SD and ED/HD video in simultaneous mode.

The data coming either from the VSP section or directly from the ADV8005 front-end input, is input to the SD encoder through two 8/10/12-bit SDR buses; the ED/HD encoder is accessed through three 8/10/12-bit SDR buses.

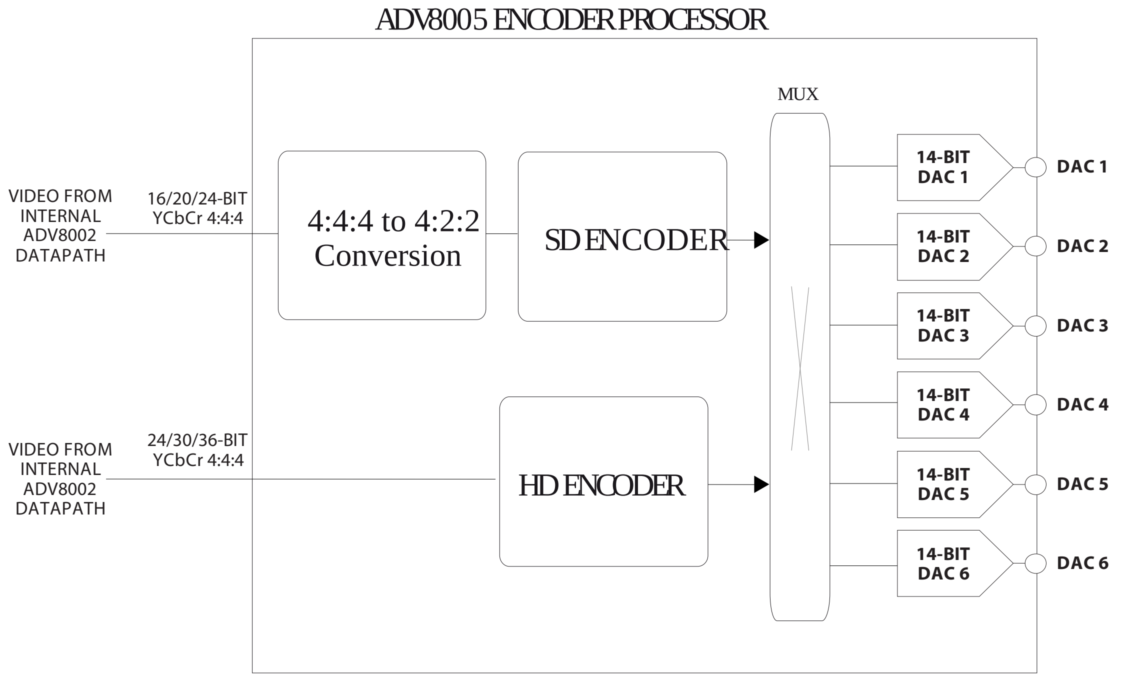


Figure 111: Simplified View of ADV8005 Encoder Block

The video being routed to the SD and ED/HD encoders can be selected through the 0x0004[7:4] register (ED/HD encoder) and 0x0004[3:0] (SD encoder). Refer Section 2.2.1 for more information.

Once the desired video has been routed to the encoder, the mode of the incoming video data needs to be set using func_mode[2:0].

func_mode[**2:0**], Encoder Map, *Address 0xE401[6:4]*
This signal is used to select the input mode to the encoder.

Function

func_mode[2:0]	**Description**
000 (default)	SD Input Only
001	ED/HD-SDR input only
010	Reserved
011	Simultaneous SD and ED/HD-SDR
100	Reserved
101	Reserved
110	Reserved
111	Reserved

Once the input configuration to the encoder section is configured, the input standard to the SD and/or HD encoder must be selected. Table 70 lists the possible input standards supported by the ADV8005 encoder core. Note that if using the ADV8005 de-interlacer and/or scaler, the input standard of the encoder must be set to that of the output of the VSP section. If bypassing the VSP section, the user should set this to the standard of the external input video.

If configuring the HD encoder, the input standard must be set using hd_enc_ip_mode[4:0].

hd_enc_ip_mode[4:0], Encoder Map, *Address 0xE430[7:3]*
This signal is used to select the ED/HD output standard.

Function

hd_enc_ip_mode[4:0]	**Description**
00000 (default)	SMPTE293M-1996 483P 60/1.001 OR ITU-R BT.1358 483P 60/1.001
00010	BTA T-1004 EDTV2 483P 60/1.001 OR ITU-R BT.1362 483P 60/1.001
00011	ITU-R BT.1358 576P 50
00100	ITU-R BT.1362 576P 50
00101	SMPTE296M-2001(1) 720P 60 OR SMPTE296M-2001(2) 720P 60/1.001
00110	SMPTE296M-2001(3) 720P 50
00111	SMPTE296M-2001(4) 720P 30 OR SMPTE296M-2001(5) 720P 30/1.001
01000	SMPTE296M-2001(6) 720P 25 OR
01001	SMPTE296M-2001(7) 720P 24 OR SMPTE296M-2001(8) 720P 24/1.001
01010	SMPTE240M-1999 1035I 30 OR SMPTE240M-1999 1035I 30/1.001
01011	SMPTE274-1998(1) 1080P 60 OR SMPTE274-1998(2) 1080P 60/1.001
01100	SMPTE274-1998(3) 1080P 50
01101	SMPTE274-1998(4) 1080I 30 OR SMPTE274-1998(5) 1080I 30/1.001
01110	SMPTE274-1998(6) 1080I 25
01111	SMPTE274-1998(7) 1080P 30 OR SMPTE274-1998(8) 1080P 30/1.001
10000	SMPTE274-1998(9) 1080P 25
10001	SMPTE274-1998(10) 1080P 24 OR SMPTE274-1998(11) 1080P 24/1.001
10011	SMPTE295-1997 1080I 25
10100	SMPTE295-1997 1080P 50
10110	ITU-R BT.709-5 1152I 50

For the SD encoder, the input standard can be configured using sd_enc_ip_mode[1:0]. If using the SD encoder, the SD standard can also be set using the automatic mode which is configured using sd_autodetect_en. If manually setting this SD standard, the automatic mode must be disabled.

When using the encoder in an SD-only mode, it is required that sd_enc_inp_sel[3:0] and hd_enc_inp_sel[3:0] are set to the same format.

sd_enc_ip_mode[1:0], Encoder Map, *Address 0xE480[1:0]*
This signal is used to select the SD standard.

Function

sd_enc_ip_mode[1:0]	**Description**
00 (default)	NTSC
01	PAL B/D/G/H/I
10	PAL M
11	PAL N

sd_autodetect_en, Encoder Map, *Address 0xE487[5]*
This bit is used to enable the encoder section to auto-detect the input standard.

Function

sd_autodetect_en	**Description**
1	Enabled
0 (default)	Disabled

When enabled (sd_autodetect_en set to 1), the ADV8005 encoder core can automatically identify an NTSC or a PAL B/D/G/H/I input stream. The ADV8005 encoder core is also configured to correctly encode the identified standard. The SD standard bits (sd_enc_ip_mode[1:0]) and the subcarrier frequency registers are not updated to reflect the identified standard; all registers retain their default or user defined values. These registers should, therefore, not be used as a way of determining the decoded standard.

Table 70: Standards Directly Supported by ADV8005 Encoder Processor

Active Resolution	I/P	Frame Rate (Hz)	Standard
720 × 240	P	59.94	
720 × 288	P	50	
720 × 480	I	29.97	ITU-R BT.601/656
720 × 576	I	25	ITU-R BT.601/656
720 × 483	P	59.94	SMPTE 293M
720 × 483	P	59.94	BTA T-1004
720 × 483	P	59.94	ITU-R BT.1358
720 × 576	P	50	ITU-R BT.1358
720 × 483	P	59.94	ITU-R BT.1362
720 × 576	P	50	ITU-R BT.1362
1920 × 1035	I	30	SMPTE 240M
1920 × 1035	I	29.97	SMPTE 240M
1280 × 720	P	60, 50, 30, 25, 24	SMPTE 296M
1280 × 720	P	23.97, 59.94, 29.97	SMPTE 296M
1920 × 1080	I	30, 25	SMPTE 274M
1920 × 1080	I	29.97	SMPTE 274M
1920 × 1080	I	25	SMPTE 295
1920 × 1080	I	50	ITU-R BT.709-5
1920 × 1080	P	30, 25, 24	SMPTE 274M
1920 × 1080	P	23.98, 29.97	SMPTE 274M
1920 × 1080	P	24	ITU-R BT.709-5
1920 × 1080	P	50	SMPTE 295
1920 × 1080	P	50, 59.94, 60	SMPTE 274M

I = interlaced, P = progressive.

7.3. OUTPUT CONFIGURATION

Once the input to the encoder section has been configured, the user can configure the output of the encoder DACs. Depending on the input mode specified by the func_mode[2:0] register**,** the DAC outputs can be configured accordingly using dac1_sel[2:0] to dac6_sel[2:0].

It is important to note that if the func_mode[2:0] signal is set to simultaneous mode; then DACs 1-3 can only output the ED/HD signals of YPbPr or RGB, and DACs 4-6 can only output the SD signals of CVBS or black burst or luma or chroma. It is possible to multiplex any of the ED/HD signals out on any of the DACs 1 to 3 in simultaneous mode. Similarly, it is possible to multiplex any of the SD signals out on any of the DACs 4 to 6.

It should also be noted that to enable the DAC outputs from the NON-ROVI ADV8005 part (ADV8005KBCZ-8N) 00h must be written to Encoder map, register 0xE4E0.

dac1_sel[2:0], Encoder Map, *Address 0xE429[6:4]*
This signal selects the data that is supplied to DAC 1.

Function

dac1_sel[2:0]	Description
0 (default)	CVBS or Black Burst
1	Luma
2	Chroma
3	Y/G
4	Pb/B
5	Pr/R

dac2_sel[2:0], Encoder Map, *Address 0xE429[2:0]*
This signal selects the data that is supplied to DAC 2.

Function

dac2_sel[2:0]	Description
0	CVBS or Black Burst
1 (default)	Luma
2	Chroma
3	Y/G
4	Pb/B
5	Pr/R

dac3_sel[2:0], Encoder Map, *Address 0xE42A[6:4]*
This signal selects the data that is supplied to DAC 3.

Function

dac3_sel[2:0]	Description
0	CVBS or Black Burst
1	Luma
2 (default)	Chroma
3	Y/G
4	Pb/B
5	Pr/R

dac4_sel[2:0], Encoder Map, *Address 0xE42A[2:0]*
This signal selects the data that is supplied to DAC 4.

Function

dac4_sel[2:0]	Description
0	CVBS or Black Burst
1	Luma
2	Chroma
3 (default)	Y/G
4	Pb/B
5	Pr/R

dac5_sel[2:0], Encoder Map, *Address 0xE42B[6:4]*
This signal selects the data that is supplied to DAC 5.

Function

dac5_sel[2:0]	Description
0	CVBS or Black Burst
1	Luma
2	Chroma
3	Y/G
4 (default)	Pb/B
5	Pr/R

dac6_sel[2:0], Encoder Map, *Address 0xE42B[2:0]*
This signal selects the data that is supplied to DAC 6.

Function

dac6_sel[2:0]	Description
0	CVBS or Black Burst
1	Luma
2	Chroma
3	Y/G
4	Pb/B
5 (default)	Pr/R

7.4. ADDITIONAL DESIGN FEATURES

This section outlines the various design features of the encoder which can be used to improve the overall video quality and the ADV8005 performance in a system. Many of these functions are optional and should be set depending on a user's application.

7.4.1. *Output Oversampling*

The ADV8005 encoder core includes two on-chip phase-locked loops (PLLs) that allow for oversampling of SD, ED, and HD video data. Oversampling effectively increases the bandwidth of the output video data, which means that expensive analog filters are not needed at the DAC outputs, thus resulting in reduced BOM costs. Table 71 shows the various oversampling rates supported in the ADV8005 encoder core.

Two PLLs are used for oversampling the analog output video, depending on the mode. When SD modes only are being output, PLL1 is used for output oversampling. When HD modes only are being output, PLL2 is used for output oversampling. In dual modes where both SD and HD formats are being output, PLL1 and PLL2 are both used for SD and HD video respectively.

pll_pdn, Encoder Map, *Address 0xE400[1]*
This bit is used to control the PLL and oversampling. This control allows the internal PLL 1 circuit to be powered down and the oversampling feature to be switched off. By default this is disabled, setting this bit to 0 enables this feature.

Function

pll_pdn	Description
0	PLL On
1 (default)	PLL Off

Table 71: Output Oversampling Modes and Rates

Input Mode Register 0xE401, Bits[6:4]	PLL and Oversampling Control Register 0xE400, Bit 1	Oversampling Mode and Rate
SD only	1	SD (2×)
SD only	0	SD (16×)
ED only	1	ED (1×)
ED only	0	ED (8×)
HD only	1	HD (1×)
HD only	0	HD (4×)
SD and ED	1	SD (2×) and ED (8×)
SD and ED	0	SD (16×) and ED (8×)
SD and HD	1	SD (2×) and HD (4×)
SD and HD	0	SD (16×) and HD (4×)
ED only (at 54 MHz)	1	ED only (at 54 MHz) (1×)
ED only (at 54 MHz)	0	ED only (at 54 MHz) (8×)

7.4.2. *Subcarrier Frequency Lock (SFL) Mode*

The ADV8005 encoder core can be used in Subcarrier Frequency Lock (SFL) mode (rtcen = 11). When SFL mode is enabled, the SFL pin can receive a serial digital stream from an ADI decoder (for example, ADV784x) which is used to lock the subcarrier frequency. This enables the ADV8005 encoder to stay locked to a video pixel clock which drifts over the time (this happens with poor video sources like VCRs). Since the color subcarrier in SD modes is generated from the input pixel clock to the ADV8005, these variations on its frequency may alter the final color on the CBVS or Y/C output.

Hence, the SFL mode allows the ADV8005 encoder core to automatically alter the subcarrier frequency to compensate for these line length variations. When the part is connected to a device such as an ADV784x video decoder that outputs a digital data stream in the SFL format, the part automatically changes to the compensated subcarrier frequency on a line-by-line basis. This digital data stream is 67-bits wide, and the subcarrier is contained in Bit 0 to Bit 21. Each bit is two clock cycles long.

rtcen[1:0], Encoder Map, *Address 0xE484[2:1]*
This signal is used to select the Sub-carrier Frequency Lock mode. The value of these register bits along with the status of the SFL pin determine the operation.

Function

rtcen[1:0]	Description
00 (default)	Disabled.
11	SFL mode enabled.

7.4.3. *SD VCR FF/RW Synchronization*

In DVD record applications where the encoder is used with a decoder, the VCR FF/RW synchronization control bit can be used for nonstandard input video. This is in fast forward or rewind modes.

In fast forward mode, the sync information at the start of a new field in the incoming video usually occurs before the correct number of lines/fields is reached. In rewind mode, this sync signal usually occurs after the total number of lines/fields is reached. Conventionally, this means that the output video has corrupted field signals because one signal is generated by the incoming video and another is generated when the internal line/field counters reach the end of a field.

When the VCR FF/RW sync control is enabled (dvd_r = 1), the line/field counters are updated according to the incoming VSync signal and when the analog output matches the incoming VSync signal. This control is available in all slave timing modes except slave mode 0.

dvd_r, Encoder Map, *Address 0xE482[5]*
This bit is used to enable the SD VCR FF/RW sync feature.

Function

dvd_r	**Description**
1	Enabled
0 (default)	Disabled

7.4.4. *Vertical Blanking Interval*

The ADV8005 encoder core is able to accept input data that contains VBI data (such as CGMS, WSS, VITS) in SD, ED, and HD modes.

If VBI is disabled, VBI data is not present at the output and the entire VBI is blanked. These control bits are valid in all master and slave timing modes. In order to enable this feature, vbi_data_en is set to 1.

For the SMPTE 293M (525p) standard, VBI data can be inserted on Line 13 to Line 42 of each frame or on Line 6 to Line 43 for the ITU-R BT.1358 (625p) standard. VBI data can be present on Line 10 to Line 20 for NTSC and on Line 7 to Line 22 for PAL. If CGMS is enabled and VBI is disabled, the CGMS data is, nevertheless, available at the output.

7.4.5. *SD Subcarrier Frequency Control*

The ADV8005 encoder core is able to generate the color subcarrier used in CVBS and S-Video (Y-C) outputs from the input pixel clock. Four 8-bit registers are used to set up the subcarrier frequency. The value of these registers is calculated using Equation 27 and Equation 28.

$$Subcarrier\ Frequency\ Register = \frac{Number\ of\ subcarrier\ periods\ in\ one\ video\ line}{Number\ of\ 27\ \text{MHz}\ clk\ cycles\ in\ one\ video\ line} \times 2^{32}$$

Equation 27: SD Subcarrier Frequency Calculation

where the sum is rounded to the nearest integer. For example, in NTSC mode:

$$Subcarrier\ Register\ Value = \left(\frac{227.5}{1716}\right) \times 2^{32} = 569408543$$

Equation 28: SD Subcarrier Frequency Calculation

where:
Subcarrier Register Value = 569408543d = 0×21F07C1F
SD F_{SC} Register 0: 0x1F
SD F_{SC} Register 1: 0x7C
SD F_{SC} Register 2: 0xF0
SD F_{SC} Register 3: 0x21

7.4.5.1. *Programming the FSC*

The subcarrier frequency register value is divided into four FSC registers, as shown in Equation 28. The subcarrier frequency registers (fsc[31:0]) must be updated sequentially, starting with Subcarrier Frequency Register 0 and ending with Subcarrier Frequency Register 3. The reason for this is because the subcarrier frequency only updates when Subcarrier Frequency Register 3 has been updated. The SD input standard autodetection feature (sd_autodetect_en) must be disabled. The registers to be programmed are described below.

fsc[31:0], Encoder Map, *Address 0xE48F[7:0]; Address 0xE48E[7:0]; Address 0xE48D[7:0]; Address 0xE48C[7:0]*
This register is used to set the subcarrier frequency value.
Table 72 outlines the values that should be written to the subcarrier frequency registers for NTSC and PAL B/D/G/H/I.

Table 72: Typical F_{SC} Values

Register	Description	NTSC	PAL B/D/G/H/I
0xE48C	$F_{SC}0$	0x1F	0xCB
0xE48D	$F_{SC}1$	0x7C	0x8A
0xE48E	$F_{SC}2$	0xF0	0x09
0xE48F	$F_{SC}3$	0x21	0x2A

7.4.6. *SD Non Interlaced Mode (240p/288p)*

The ADV8005 encoder core supports an SD non interlaced mode. Using this mode, progressive inputs at twice the frame rate of NTSC and PAL (240p/59.94 Hz and 288p/50 Hz respectively) can be input into the ADV8005 encoder. If the user selects the input to be 240p or 288p, sd_non_interlaced must be set correspondingly. Refer to Section 7.2 for more details on setting the input format.

sd_non_interlaced, Encoder Map, *Address 0xE488[1]*
This bit is used to enable the support of SD non-interlaced modes.

Function

sd_non_interlaced	Description
1	Enabled
0 (default)	Disabled

Note: All input configurations, output configurations, and features available in NTSC and PAL modes are available in SD non interlaced mode. For 240p/59.94 Hz input, the ADV8005 encoder core should be configured for NTSC operation. For 288p/50 Hz input, the ADV8005 encoder core should be configured for PAL operation.

7.4.7. *Filters*

The ADV8005 encoder core offers numerous filtering options for both SD and ED/HD as well as for both luma and chroma data.

7.4.7.1. *SD Filters*

Table 73 provides details on the numerous available SD filters.

Table 73: Internal Filter Specifications

Filter	Pass-Band Ripple (dB)[1]	3 dB Bandwidth (MHz)[2]
Luma LPF NTSC	0.16	4.24
Luma LPF PAL	0.1	4.81
Luma Notch NTSC	0.09	2.3/4.9/6.6
Luma Notch PAL	0.1	3.1/5.6/6.4
Luma SSAF	0.04	6.45
Luma CIF	0.127	3.02
Luma QCIF	Monotonic	1.5
Chroma 0.65 MHz	Monotonic	0.65
Chroma 1.0 MHz	Monotonic	1

Filter	Pass-Band Ripple (dB)[1]	3 dB Bandwidth (MHz)[2]
Chroma 1.3 MHz	0.09	1.395
Chroma 2.0 MHz	0.048	2.2
Chroma 3.0 MHz	Monotonic	3.2
Chroma CIF	Monotonic	0.65
Chroma QCIF	Monotonic	0.5

[1] Pass-band ripple is the maximum fluctuation from the 0 dB response in the pass band, measured in decibels. The pass band is defined to have 0 Hz to fc (Hz) frequency limits for a low-pass filter and 0 Hz to f1 (Hz), and f2 (Hz) to infinity for a notch filter, where fc, f1, and f2 are the −3 dB points.
[2] 3 dB bandwidth refers to the −3 dB cutoff frequency.

The luma filter supports several different frequency responses, including two low-pass responses, two notch responses, an extended (SSAF) response with or without gain boost attenuation, a CIF response, and a QCIF response. These can be configured using luma_filter_sel[2:0].

luma_filter_sel[**2:0**], Encoder Map, *Address 0xE480[4:2]*
This signal is used to configure the luma filters for SD data.

Function

luma_filter_sel[2:0]	Description
000	LPF NTSC
001	LPF PAL
010	Notch NTSC
011	Notch PAL
100 (default)	SSAF Luma
101	Luma CIF
110	Luma QCIF
111	Reserved

If SD SSAF gain is enabled, there are 13 response options in the −4 dB to +4 dB range. The desired response can be programmed using register 0xA2. The variation in frequency responses is shown in Figure 112, Figure 113, and Figure 114. The registers required for enabling and controlling the SSAF filter gain are described below.

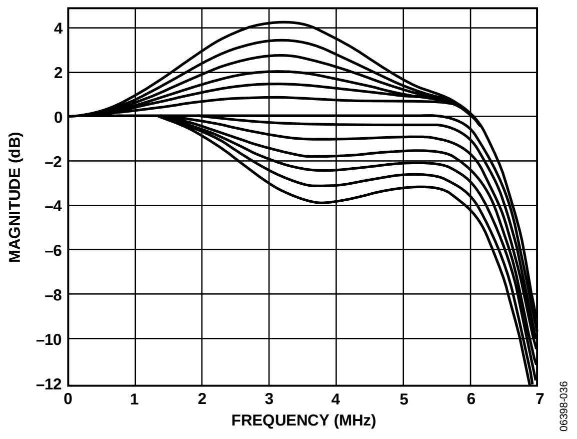


Figure 112: SD Luma SSAF Filter, Programmable Responses

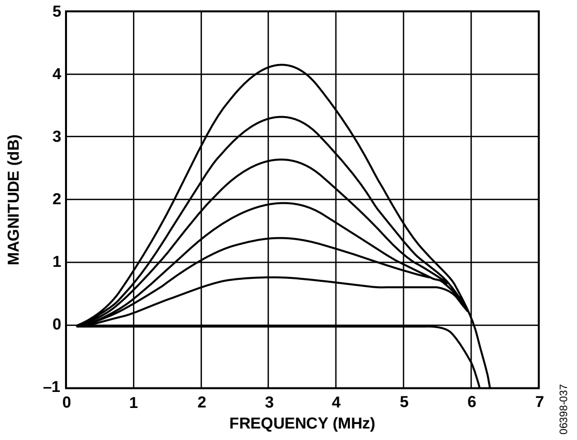


Figure 113: SD Luma SSAF Filter, Programmable Gains

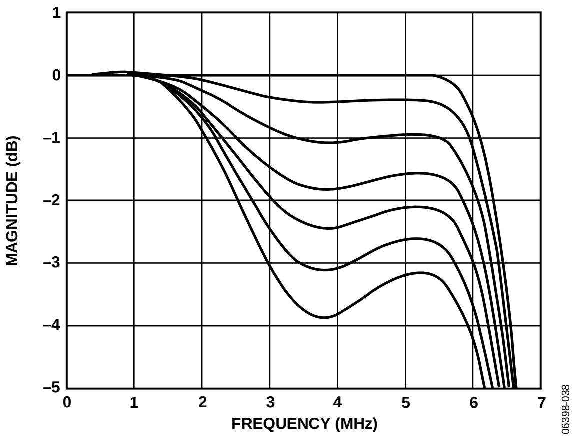


Figure 114: SD Luma SSAF Filter, Programmable Attenuation

peak_en, Encoder Map, *Address 0xE487[4]*
This bit is used to enable the SD SSAF filter gain.

Function

peak_en	**Description**
1	Enabled
0 (default)	Disabled

peak[3:0], Encoder Map, *Address 0xE4A2[3:0]*
This signal is used to configure the SD luma SSAF gain/attenuation (only applicable if subaddress 0x87, Bit 4 = 1).

Function

peak[3:0]	**Description**
0000 (default)	-4dB
	...
0100	0dB
	...
1000	+4dB

The chroma filters support several different frequency responses, including six low-pass responses, a CIF response, and a QCIF response. These can be configured using chroma_filter_sel[2:0].

chroma_filter_sel[2:0], Encoder Map, *Address 0xE480[7:5]*
This signal is used to configure the chroma filters for SD data.

Function

chroma_filter_sel[2:0]	**Description**
000 (default)	1.3MHz
001	0.65MHz
010	1MHz
011	2MHz
100	Reserved
101	Chroma CIF
110	Chroma QCIF
111	3MHz

In addition to the chroma filters listed with chroma_filter_sel[2:0], there is an SSAF filter that is specifically designed for the color difference component outputs, Pr and Pb. This filter has a cutoff frequency of ~2.7 MHz and a gain of –40 dB at 3.8 MHz. Refer to Figure 115 for more details. To enable this filter, wide_uv_filt should be set to 1.

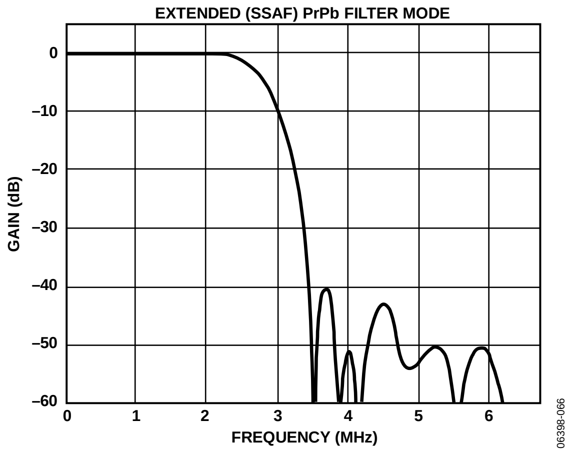


Figure 115: PrPb SSAF Filter

wide_uv_filt, Encoder Map, *Address 0xE482[0]*
This bit is used to enable the SSAF filter for PrPb SD data.

Function

wide_uv_filt	**Description**
1 (default)	Enabled
0	Disabled

If this filter is disabled, one of the chroma filters shown in Table 73 can be selected and used for the CVBS or luma/chroma signal.

7.4.7.2. *ED/HD Filters*

The ADV8005 encoder core also includes a sinc compensation filter designed to counter the effect of sinc roll-off in DAC 1, DAC 2, and DAC 3 while operating in ED/HD mode. The benefit of the filter is illustrated in Figure 116 and Figure 117 which show the effect of the filter when enabled and disabled. This filter is enabled by default but can be disabled using sinc_filt_df_en.

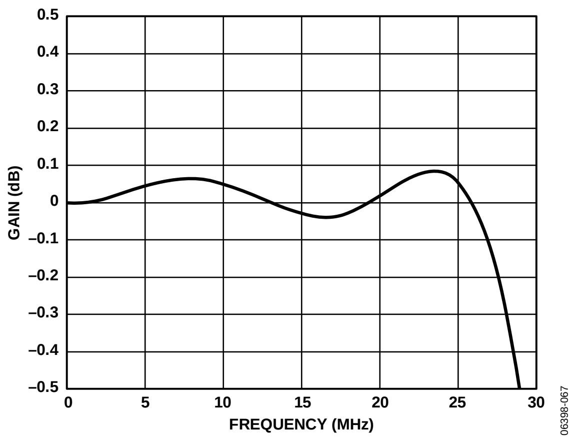


Figure 116: ED/HD Sinc Compensation Filter Enabled

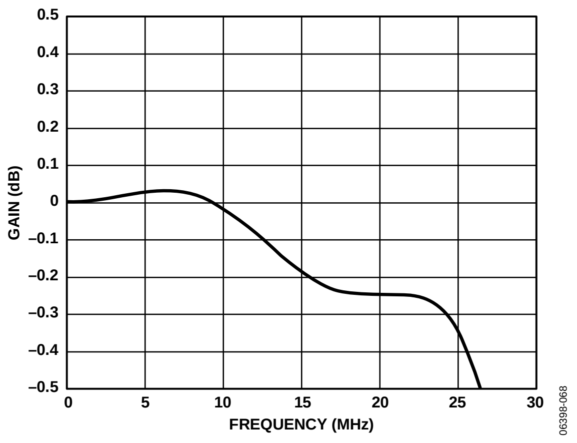


Figure 117: ED/HD Sinc Compensation Filter Disabled

sinc_filt_df_en, Encoder Map, *Address 0xE433[3]*

This bit is used to disable the sinc compensation filter on DAC1, DAC2 and DAC3.

Function

sinc_filt_df_en	Description
0	Disabled
1 «	Enabled

7.4.8. *ED/HD Test Pattern Generator*

ADV8005 is able to internally generate ED/HD black bar, uniform background color or hatch test patterns. It is not possible to output a color bar test pattern while EH/HD video is being routed through the encoder. This test pattern can be enabled using hdtv_tp_en and the test pattern used can be determined using hdtv_flat_tp.

y_colour[7:0], cr_colour[7:0]**,** and cb_colour[7:0] are used to program the output color of the internal ED/HD test pattern generator, whether it is the lines of the crosshatch pattern or the uniform field test pattern. They are not functional as color controls for external pixel data input.

hdtv_tp_en, Encoder Map, *Address 0xE431[2]*
This bit is used to enable the ED/HD test pattern generator.

Function

hdtv_tp_en	Description
0 (default)	ED/HD test pattern off
1	ED/HD test pattern on.

The values for the luma (Y) and the color difference (Cr and Cb) signals used to obtain white, black, and saturated primary and complementary colors conform to the ITU-R BT.601-4 standard.

Table 74 shows sample color values that can be programmed into the color registers when the output standard selection is set to EIA 770.2/EIA770.3 (Reg 0x30, Bits[1:0] = 00).

hdtv_flat_tp, Encoder Map, *Address 0xE431[3]*
This bit is used to select the pattern used by the internal test pattern generator.

Function

hdtv_flat_tp	Description
0 (default)	Hatch
1	Field/frame

y_colour[7:0], Encoder Map, *Address 0xE436[7:0]*
This register is used to control the ED/HD test pattern, Y level.

cr_colour[7:0], Encoder Map, *Address 0xE437[7:0]*
This register is used to control the ED/HD test pattern, Cr level.

cb_colour[7:0], Encoder Map, *Address 0xE438[7:0]*
This register is used to control the ED/HD test pattern, Cb level.

Table 74: Sample Color Values for EIA 770.2/EIA770.3 ED/HD Output Standard Selection

Sample Color	Y Value		Cr Value		Cb Value	
White	235	(0xEB)	128	(0x80)	128	(0x80)
Black	16	(0x10)	128	(0x80)	128	(0x80)
Red	81	(0x51)	240	(0xF0)	90	(0x5A)
Green	145	(0x91)	34	(0x22)	54	(0x36)
Blue	41	(0x29)	110	(0x6E)	240	(0xF0)
Yellow	210	(0xD2)	146	(0x92)	16	(0x10)
Cyan	170	(0xAA)	16	(0x10)	166	(0xA6)
Magenta	106	(0x6A)	222	(0xDE)	202	(0xCA)

7.4.9. *Color Space Conversion Matrix*

The input to the encoder block on the ADV8005 should always be in a YCbCr color space. If an RGB color space is present at the input pins, the CSC on the I/O block of the ADV8005 can be used to convert it to YCbCr. It is possible, however, to convert from YCbCr to an RGB stream in the encoder. The encoder output color space can be programmed using yuv_out.

yuv_out, Encoder Map, *Address 0xE402[5]*
This bit is used to select the output colour space for the encoder.

Function

yuv_out	Description
0	RGB component outputs.
1 (default)	YPrPb component outputs.

7.4.10. *ED/HD Manual CSC Matrix Adjust Feature*

The ED/HD manual CSC matrix adjust feature provides custom coefficient manipulation for the YPbPr to RGB CSC and is used in ED and HD modes only. matrix_prog_en can be used to enable this feature.

matrix_prog_en, Encoder Map, *Address 0xE402[3]*
This bit is used to enable the manual mode for the ED/HD colour space converter.

Function

matrix_prog_en	**Description**
0 (default)	Automatic Mode
1	Manual Mode

Normally, there is no need to enable this feature because the CSC matrix automatically performs the CSC based on the input mode chosen (ED or HD) and the output color space selected using yuv_out. If the user needs to automatically update the CSC coefficients, the following procedure is followed.

If the user selects the RGB output color space, the ED/HD CSC matrix scaler uses the following equations:

$$R = GY \times Y + RV \times Pr$$
$$G = GY \times Y - (GU \times Pb) - (GV \times Pr)$$
$$B = GY \times Y + BU \times Pb$$

Note: Subtractions in these equations are implemented in the hardware.

The following registers need to be programmed with these values:

- gy [9:0] – Reg 0xE405 [7:0], Reg 0xE403 [1:0]
- gu [9:0] – Reg 0xE406 [7:0], Reg 0xE404 [7:6]
- gv [9:0] – Reg 0xE407 [7:0], Reg 0xE404 [5:4]
- bu [9:0] – Reg 0xE408 [7:0], Reg 0xE404 [3:2]
- rv [9:0] – Reg 0xE409 [7:0], Reg 0xE404 [1:0]

On powerup, the CSC matrix is programmed with the default values shown in Table 75.

Table 75: ED/HD Manual CSC Matrix Default Values

Register	**Default**
0x03	0x03
0x04	0xF0
0x05	0x4E
0x06	0x0E
0x07	0x24
0x08	0x92
0x09	0x7C

When the ED/HD manual CSC matrix adjust feature is enabled, the default coefficient values in Reg 0xE403 to Reg 0xE409 are correct for the HD color space only. The color components are converted according to the following 1080i and 720p standards (SMPTE 274M, SMPTE 296M):

$$R = Y + 1.575Pr$$
$$G = Y - 0.468Pr - 0.187Pb$$
$$B = Y + 1.855Pb$$

The conversion coefficients should be multiplied by 315 before being written to the ED/HD CSC matrix registers. This is reflected in the default values for gy = 0x13B, gu = 0x03B, gv = 0x093, bu = 0x248, and rv = 0x1F0.

If the ED/HD manual CSC matrix adjust feature is enabled and another input standard (such as ED) is used, the scale values for gy, gu, gv, bu, and rv must be adjusted according to this input standard color space. The user should consider that the color component conversion may use different scale values.

For example, SMPTE 293M uses the following conversion:

$R = Y + 1.402Pr$
$G = Y - 0.714Pr - 0.344Pb$
$B = Y + 1.773Pb$

The programmable CSC matrix is used for external ED/HD pixel data and is not functional when internal test patterns are enabled.

7.4.10.1. *Programming the CSC Matrix*

If the user needs to manually provide the coefficients for the CSC matrix for ED/HD, this procedure is followed:

1. Enable the ED/HD manual CSC matrix adjust feature (matrix_prog_en).
2. Set the output to RGB (yuv_out).
3. Disable sync on YPrPb (Reg 0xE435, Bit 2).
4. Enable sync on RGB (optional) (Reg 0xE402, Bit 4).

The gy value controls the green signal output level, the bu value controls the blue signal output level, and the rv value controls the red signal output level.

7.4.11. *SD Luma and Color Scale Control*

When enabled, the SD luma and color scale control feature can be used to scale the SD Y, Cb, and Cr output levels. This feature can be enabled using scale_ycbcr_en. This feature affects all SD output signals, regardless of the encoder output, that is, CVBS, Y-C, YPrPb, and RGB.

scale_ycbcr_en, Encoder Map, *Address 0xE487[0]*
This bit is used to enable the SD luma and colour scale control feature.

Function

scale_ycbcr_en	Description
1	Enabled
0 (default)	Disabled

When enabled, three 10-bit registers (SD Y scale, SD Cb scale, and SD Cr scale) control the scaling of the SD Y, Cb, and Cr output levels. The SD Y scale register contains the scaling factor used to scale the Y level from 0.0 to 1.5 times its initial level. The SD Cb scale and SD Cr scale registers contain the scaling factors used to scale the Cb and Cr levels from 0.0 to 2.0 times their initial levels, respectively.
The registers needed to scale the outputs are contrast[9:0], cb_scale[9:0] and cr_scale[9:0].

contrast[7:0], IO Map, *Address 0x1A2B[7:0]*
This register is used to adjust the contrast value for Y channel. This register uses 1.7 notation.

Function

contrast[7:0]	Description
0x00	Gain of 0
0x80 «	Unity gain
0xFF	Gain of 2

cb_scale[9:0], Encoder Map, *Address 0xE49E[7:0]; Address 0xE49C[3:2]*
This signal is used to set the SD Cb scale value.

cr_scale[9:0], Encoder Map, *Address 0xE49F[7:0]; Address 0xE49C[5:4]*
This signal is used to set the SD Cr scale value.

To use this function, the values to be written to these 10-bit registers are calculated using the following equation:

Y, Cb, or Cr Scale Value = Scale Factor × 512

For example, if Scale Factor = 1.3
Y, Cb, or Cr Scale Value = 1.3 × 512 = 665.6
Y, Cb, or Cr Scale Value = 666 (rounded to the nearest integer)
Y, Cb, or Cr Scale Value = 1010 0110 10b
Reg 0xE49C, SD scale LSB register = 0x2A
Reg 0xE49D, SD Y scale register = 0xA6
Reg 0xE49E, SD Cb scale register = 0xA6
Reg 0xE49F, SD Cr scale register = 0xA6

Note: It is recommended that the SD luma scale saturation feature, saturate_luma, be enabled when scaling the Y output level to avoid excessive Y output levels.

saturate_luma, Encoder Map, *Address 0xE487[1]*
This bit is used to enable the SD luma scale saturation.

Function

saturate_luma	Description
1	Enabled
0 (default)	Disabled

7.4.12. *SD Hue Adjust Control*

When enabled, the SD hue adjust control register is used to adjust the hue on the SD composite and chroma outputs. To enable this feature, hue_en must be programmed to 1.

hue_en, Encoder Map, *Address 0xE487[2]*
This bit is used to enable the hue adjust function.

Function

hue_en	Description
1	Enabled
0 (default)	Disabled

Register 0xE4A0 contains the bits required to vary the hue of the video data, that is, the variance in phase of the subcarrier during active video with respect to the phase of the subcarrier during the color burst. The ADV8005 encoder provides a range of ±22.5° in increments of 0.17578125°. For normal operation (zero adjustment), this register is set to 0x80. Value 0xFF and value 0x00 represent the upper and lower limits, respectively, of the attainable adjustment in NTSC mode. Value 0xFF and value 0x01 represent the upper and lower limits, respectively, of the attainable adjustment in PAL mode.

The hue adjust value is calculated using the following equation:

$$\text{Hue Adjust (°)} = 0.17578125° (HCR_d - 128)$$

where HCR_d is the hue adjust control register (decimal).

For example, to adjust the hue by +4°, write 0x97 to hue[7:0].

$$\left(\frac{4}{0.17578125}\right) + 128 \approx 151d = 0x97$$

where the sum is rounded to the nearest integer.

To adjust the hue by −4°, write 0x69 to hue[7:0].

$$\left(\frac{-4}{0.17578125}\right) + 128 \approx 105d = 0x69$$

where the sum is rounded to the nearest integer.

hue[7:0], Encoder Map, *Address 0xE4A0[7:0]*
This register is used to set the SD hue adjust value.

Function

hue[7:0]	**Description**
0x00 (default)	SD Hue Value

7.4.13. *SD Brightness Detect*

The ADV8005 encoder core allows the user to monitor the brightness level of the incoming video data. This feature is used to monitor the average brightness of the incoming Y signal on a field-by-field basis. The information is read from the I^2C and, based on this information, the color saturation, contrast, and brightness controls can be adjusted, for example, to compensate for very dark pictures.

The luma data is monitored in the active video area only. The average brightness I^2C register is updated on the falling edge of every $\overline{\text{VSYNC}}$ signal. This can be monitored using bright_detect_val[7:0].

bright_detect_val[7:0], Encoder Map, *Address 0xE4BA[7:0] (Read Only)*
This register is used to adjust the SD brightness value.

Function

bright_detect_val[7:0]	**Description**
0xXX	(Larger settings results in a brighter output)

7.4.14. *SD Brightness Control*

When this feature is enabled, the SD brightness/WSS control register, setup[6:0], is used to control brightness by adding a programmable setup level onto the scaled Y data. To enable this feature, setup_en must be configured.

setup_en, Encoder Map, *Address 0xE487[3]*
This bit is used to enable the SD brightness control feature.

Function

setup_en	**Description**
1	Enabled
0 (default)	Disabled

For NTSC with pedestal, the setup can vary from 0 IRE to 22.5 IRE. For NTSC without pedestal and for PAL, the setup can vary from −7.5 IRE to +15 IRE. Refer to Figure 118 for more details.
The SD brightness control register is an 8-bit register. The seven LSBs of this 8-bit register are used to control the brightness level, which can be a positive or negative value.

For example, to add a +20 IRE brightness level to an NTSC signal with pedestal, the procedure is as follows:

$$0 \times (\text{SD Brightness Value}) =$$
$$0 \times (\text{IRE Value} \times 2.015631) =$$
$$0 \times (20 \times 2.015631) = 0 \times (40.31262) \approx 0\text{x}28$$

To add a –7 IRE brightness level to a PAL signal, write 0x72 to setup[6:0].

$$0 \times (\text{SD Brightness Value}) =$$
$$0 \times (\text{IRE Value} \times 2.075631) =$$
$$0 \times (7 \times 2.015631) = 0\text{x}(14.109417) \approx 0001110\text{b}$$
$$0001110\text{b into twos complement} = 1110010\text{b} = 0\text{x}72$$

setup[6:0], Encoder Map, *Address 0xE4A1[6:0]*
This signal is used to specify the SD brightness value.

Table 76: Sample Brightness Control Values[1]

Setup Level (NTSC) with Pedestal	Setup Level (NTSC) Without Pedestal	Setup Level (PAL)	Brightness Control Value (setup[6:0])
22.5 IRE	15 IRE	15 IRE	0x1E
15 IRE	7.5 IRE	7.5 IRE	0x0F
7.5 IRE	0 IRE	0 IRE	0x00
0 IRE	−7.5 IRE	−7.5 IRE	0x71

[1] Values in the range of 0x3F to 0x44 may result in an invalid output signal.

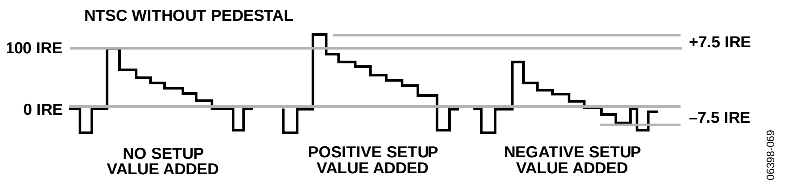


Figure 118: Examples of Brightness Control Values

7.4.15. *Double Buffering*

Double buffered registers are updated once per field. Double buffering improves overall performance because modifications to register settings are not made during active video but take effect prior to the start of the active video on the next field. This can be enabled for both SD and ED/HD using db_en and db_en_hdtv respectively.

7.4.15.1. *ED/HD Doubling Buffering*

db_en_hdtv, Encoder Map, *Address 0xE433[7]*
This bit is used to enable the double buffering on the appropriate ED/HD registers.

Function

db_en_hdtv	Description
0 (default)	Cb after falling edge of HSYNC
1	Cr after falling edge of HSYNC

Double buffering can be activated on the following ED/HD functions: the ED/HD gamma A and gamma B curves and the ED/HD CGMS registers.

7.4.15.2. *SD Doubling Buffering*

db_en, Encoder Map, *Address 0xE488[2]*
This bit is used to enable double buffering on the appropriate SD registers.

Function

db_en	Description
1	Enabled
0 (default)	Disabled

Double buffering can be activated on the following SD functions: the SD gamma A and gamma B curves, SD Y scale, SD Cr scale, SD Cb scale, SD brightness, SD closed captioning, and SD Macrovision bits (Reg 0xE4E0, Bits [5:0]).

7.4.16. *Programmable DAC Gain Control*

It is possible to adjust the DAC output signal gain up or down from its absolute level. This is illustrated in Figure 119.

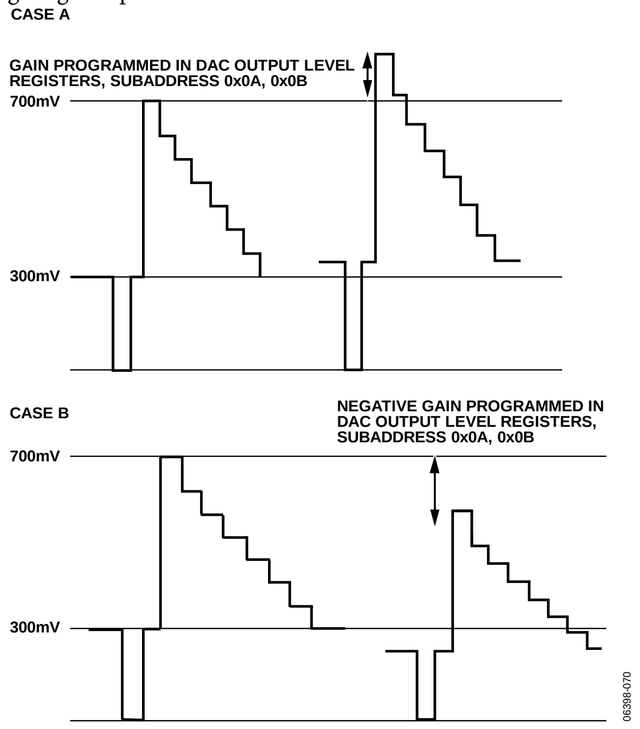

CASE A
GAIN PROGRAMMED IN DAC OUTPUT LEVEL REGISTERS, SUBADDRESS 0x0A, 0x0B
700mV
300mV
CASE B
NEGATIVE GAIN PROGRAMMED IN DAC OUTPUT LEVEL REGISTERS, SUBADDRESS 0x0A, 0x0B
700mV
300mV
06398-070

Figure 119: Programmable DAC Gain – Positive and Negative Gain

In Case A of Figure 119, the video output signal is gained. The absolute level of the sync tip and the blanking level increase with respect to the reference video output signal. The overall gain of the signal is increased from the reference signal.

In Case B of Figure 119, the video output signal is reduced. The absolute level of the sync tip and the blanking level decrease with respect to the reference video output signal. The overall gain of the signal is reduced from the reference signal.

The range of this feature is specified for ±7.5% of the nominal output from the DACs. For example, if the output current of the DAC is 4.33 mA, the DAC gain control feature can change this output current from 4.008 mA (–7.5%) to 4.658 mA (+7.5%). To enable the gain for the relevant set of DACs, dac4to6_tuning[7:0] and dac1to3_tuning[7:0] must be configured.

dac4to6_tuning[7:0], Encoder Map, *Address 0xE40A[7:0]*
This register is used to set the gain for DACs 4-6 output voltage.

Function

dac4to6_tuning[7:0]	**Description**
11000000	-7.5%
11000001	-7.382%
11000010	-7.364%
	...
11111111	-0.018%
00000000 (default)	0%
00000001	0.018%
00000010	0.036%
	...
00111111	+7.382%
01000000	+7.5%

dac1to3_tuning[7:0], Encoder Map, *Address 0xE40B[7:0]*
This register is used to set the gain for DACs 1-3 output voltage.

Function

dac1to3_tuning[7:0]	**Description**
11000000	-7.5%
11000001	-7.382%
11000010	-7.364%
	...
11111111	-0.018%
00000000 (default)	0%
00000001	0.018%
00000010	0.036%
	...
00111111	+7.382%
01000000	+7.5%

The reset value of the control registers is 0x00; that is, nominal DAC current is output. Table 77 shows how the output current of the DACs varies for a nominal 4.33 mA output current.

Table 77: DAC Gain Control

DAC Gain Register Value	DAC Current (mA)	% Gain	Note
0100 0000 (0x40)	4.658	7.5000%	
0011 1111 (0x3F)	4.653	7.3820%	
0011 1110 (0x3E)	4.648	7.3640%	
…	…	…	
…	…	…	
0000 0010 (0x02)	4.43	0.0360%	
0000 0001 (0x01)	4.38	0.0180%	
0000 0000 (0x00)	4.33	0.0000%	Reset value, nominal
1111 1111 (0xFF)	4.25	−0.0180%	
1111 1110 (0xFE)	4.23	−0.0360%	
…	…	…	
…	…	…	
1100 0010 (0xC2)	4.018	−7.3640%	
1100 0001 (0xC1)	4.013	−7.3820%	
1100 0000 (0xC0)	4.008	−7.5000%	

7.4.17. *Gamma Correction*

Generally, gamma correction is applied to compensate for the nonlinear relationship between the signal input and the output brightness level (as perceived on a CRT). It can also be applied wherever nonlinear processing is used.

Gamma correction uses the function:

$Signal_{OUT} = (Signal_{IN})^{\gamma}$

where γ is the gamma correction factor.

Gamma correction is available for SD and ED/HD video. For both variations, there are twenty 8-bit registers, used to program the Gamma Correction Curve A and Gamma Correction Curve B.

Gamma correction is performed on the luma data only. The user can choose one of two correction curves, Curve A or Curve B. Only one of these curves can be used at a time.

The shape of the gamma correction curve is controlled by defining the curve response at 10 different locations along the curve. By altering the response at these locations, the shape of the gamma correction curve can be modified. Between these points, linear interpolation is used to generate intermediate values. Considering that the curve has a total length of 256 points, the 10 programmable locations are at the following points: 24, 32, 48, 64, 80, 96, 128, 160, 192, and 224. The following locations are fixed and cannot be changed: 0, 16, 240, and 255.

From the curve locations, 16 to 240, the values at the programmable locations and, therefore, the response of the gamma correction curve, should be calculated to produce the following result:

$$x_{DESIRED} = (x_{INPUT})^{\gamma}$$

where:
$x_{DESIRED}$ is the desired gamma corrected output.
x_{INPUT} is the linear input signal.
γ is the gamma correction factor.

To program the gamma correction registers, the 10 programmable curve values are calculated using Equation 29.

$$\gamma_n = \left(\left(\frac{n-16}{240-16} \right)^{\gamma} \times (240-16) \right) + 16$$

Equation 29: Gamma Correction Calculation

where:
y_n is the value to be written into the gamma correction register for point *n* on the gamma correction curve.
n = 24, 32, 48, 64, 80, 96, 128, 160, 192, or 224.
γ is the gamma correction factor.

For example, setting $\gamma = 0.5$ for all programmable curve data points results in the following y_n values:

$y_{24} = [(8/224)^{0.5} \times 224] + 16 = 58$
$y_{32} = [(16/224)^{0.5} \times 224] + 16 = 76$
$y_{48} = [(32/224)^{0.5} \times 224] + 16 = 101$
$y_{64} = [(48/224)^{0.5} \times 224] + 16 = 120$
$y_{80} = [(64/224)^{0.5} \times 224] + 16 = 136$
$y_{96} = [(80/224)^{0.5} \times 224] + 16 = 150$
$y_{128} = [(112/224)^{0.5} \times 224] + 16 = 174$
$y_{160} = [(144/224)^{0.5} \times 224] + 16 = 195$
$y_{192} = [(176/224)^{0.5} \times 224] + 16 = 214$
$y_{224} = [(208/224)^{0.5} \times 224] + 16 = 232$

Where the sum of each equation is rounded to the nearest integer, these must then all be converted to hex.

The gamma curves in Figure 120 and Figure 121 are examples only; any user defined curve in the range from 16 to 240 is acceptable.

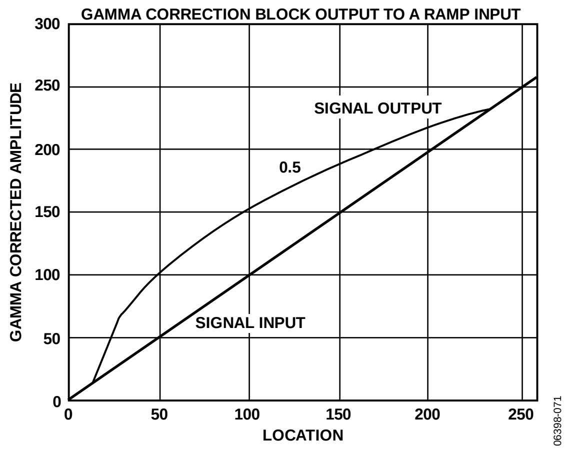


Figure 120: Signal Input (Ramp) and Signal Output for Gamma 0.5

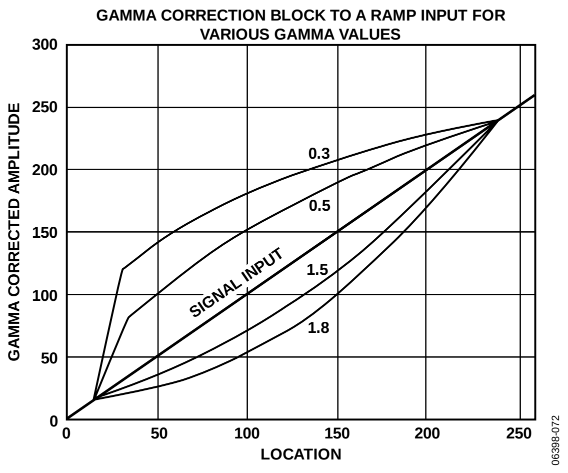


Figure 121: Signal Input (Ramp) and Selectable Output Curves

7.4.17.1. ED/HD Gamma Correction

To enable the gamma correction curves for ED/HD standards, gamma_en_hdtv must be programmed.

gamma_en_hdtv, Encoder Map, *Address 0xE435[5]*
This bit is used to enable the gamma correction curves for ED/HD video data.

Function

gamma_en_hdtv	**Description**
0 (default)	Disabled
1	Enabled

The ED/HD gamma correction curves are provided in Table 78 and Table 79.

Table 78: ED/HD Gamma Curve A

Curve Type	Point	Register Address
ED/HD Gamma Curve A	(A0 – Point 24)	0xE444
ED/HD Gamma Curve A	(A1 – Point 32)	0xE445
ED/HD Gamma Curve A	(A2 – Point 48)	0xE446
ED/HD Gamma Curve A	(A3 – Point 64)	0xE447
ED/HD Gamma Curve A	(A4 – Point 80)	0xE448
ED/HD Gamma Curve A	(A5 – Point 96)	0xE449
ED/HD Gamma Curve A	(A6 – Point 128)	0xE44A
ED/HD Gamma Curve A	(A7 – Point 160)	0xE44B
ED/HD Gamma Curve A	(A8 – Point 192)	0xE44C
ED/HD Gamma Curve A	(A9 – Point 224)	0xE44D

Table 79: ED/HD Gamma Curve B

Curve Type	Point	Register Address
ED/HD Gamma Curve B	(B0 – Point 24)	0xE44E
ED/HD Gamma Curve B	(B1 – Point 32)	0xE44F
ED/HD Gamma Curve B	(B2 – Point 48)	0xE450
ED/HD Gamma Curve B	(B3 – Point 64)	0xE451
ED/HD Gamma Curve B	(B4 – Point 80)	0xE452
ED/HD Gamma Curve B	(B5 – Point 96)	0xE453
ED/HD Gamma Curve B	(B6 – Point 128)	0xE454
ED/HD Gamma Curve B	(B7 – Point 160)	0xE455
ED/HD Gamma Curve B	(B8 – Point 192)	0xE456
ED/HD Gamma Curve B	(B9 – Point 224)	0xE457

To select between both the A and B curves for the ED/HD gamma correction, the gamma_curve_b_hdtv must be programmed.

gamma_curve_b_hdtv, Encoder Map, *Address 0xE435[4]*
This bit is used to select the gamma correction curves for ED/HD video data.

Function

gamma_curve_b_hdtv	Description
0 (default)	Gamma Correction Curve A
1	Gamma Correction Curve B

7.4.17.2. SD Gamma Correction

To enable the gamma correction curves for SD standards, gamma_en must be programmed.

gamma_en, Encoder Map, *Address 0xE488[6]*
This bit is used to enable the gamma correction curves for SD video data.

Function

gamma_en	Description
1	Enabled
0 (default)	Disabled

The SD gamma correction curves are provided in Table 80.

Table 80: SD Gamma Curve A

Curve Type	Point	Register Address
SD Gamma Curve A	(A0 – Point 24)	0xA6
SD Gamma Curve A	(A1 – Point 32)	0xA7
SD Gamma Curve A	(A2 – Point 48)	0xA8
SD Gamma Curve A	(A3 – Point 64)	0xA9
SD Gamma Curve A	(A4 – Point 80)	0xAA
SD Gamma Curve A	(A5 – Point 96)	0xAB
SD Gamma Curve A	(A6 – Point 128)	0xAC
SD Gamma Curve A	(A7 – Point 160)	0xAD
SD Gamma Curve A	(A8 – Point 192)	0xAE
SD Gamma Curve A	(A9 – Point 224)	0xAF
SD Gamma Curve B	(B0 – Point 24)	0xB0
SD Gamma Curve B	(B1 – Point 32)	0xB1
SD Gamma Curve B	(B2 – Point 48)	0xB2
SD Gamma Curve B	(B3 – Point 64)	0xB3
SD Gamma Curve B	(B4 – Point 80)	0xB4
SD Gamma Curve B	(B5 – Point 96)	0xB5
SD Gamma Curve B	(B6 – Point 128)	0xB6
SD Gamma Curve B	(B7 – Point 160)	0xB7
SD Gamma Curve B	(B8 – Point 192)	0xB8
SD Gamma Curve B	(B9 – Point 224)	0xB9

To select between both the A and B curves, gamma_curve_b must be programmed.

gamma_curve_b, Encoder Map, *Address 0xE488[7]*
This bit is used to select the gamma correction curves for SD video data.

Function

gamma_curve_b	Description
0 (default)	Gamma Correction Curve A
1	Gamma Correction Curve B

7.4.18. *ED/HD Sharpness Filter and Adaptive Filter Controls*

There are three filter modes available on the ADV8005 encoder block: a sharpness filter mode and two adaptive filter modes.

7.4.18.1. ED/HD Sharpness Filter Mode

To enhance or attenuate the Y signal in the frequency ranges shown in Figure 122, the ED/HD sharpness filter must be enabled (sharp_en set to 1) and the ED/HD adaptive filter must be disabled (adapt_en set to 0).

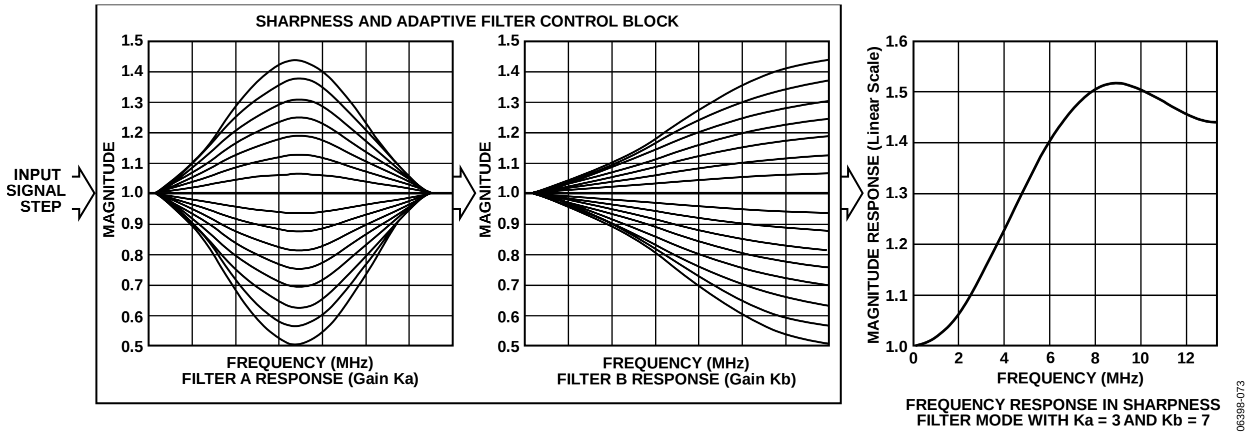


Figure 122: ED/HD Sharpness and Adaptive Filter Control Block

To enable the ED/HD sharpness filter, the following bit must be written to.

sharp_en, Encoder Map, *Address 0xE431[7]*
This bit is used to enable the ED/HD sharpness filter on the luma data. By default this is set to 0 which means the filter is disabled.

Function

sharp_en	Description
0 (default)	Disabled
1	Enabled

Likewise, the adaptive filter must be disabled by writing to the following bit.

adapt_en, Encoder Map, *Address 0xE435[7]*
This bit is used to enable the ED/HD adaptive filter.

Function

adapt_en	Description
0 (default)	Disabled
1	Enabled

To select one of the 256 individual responses, the corresponding gain values, ranging from −8 to +7 for each filter, must be programmed into the ED/HD sharpness filter gain register. These are programmed using kb[3:0] and ka[3:0].

kb[3:0], Encoder Map, *Address 0xE440[7:4]*
This signal is used to configure the ED/HD sharpness filter gain, value B.

Function

kb[3:0]	Description
0000 (default)	Gain B 0
0001	Gain B +1
	...
0111	Gain B +7
1000	Gain B -8
	...
1110	Gain B -2
1111	Gain B -1

ka[**3:0**], Encoder Map, *Address 0xE440[3:0]*
This signal is used to configure the ED/HD sharpness filter gain, value A.

Function

ka[3:0]	Description
0000 (default)	Gain A 0
0001	Gain A +1
	...
0111	Gain A +7
1000	Gain A -8
	...
1110	Gain A -2
1111	Gain A -1

7.4.18.2. ED/HD Adaptive Filters

The ED/HD adaptive filter (Threshold A, Threshold B, and Threshold C) registers, the ED/HD adaptive filter (Gain 1, Gain 2, and Gain 3) registers, and the ED/HD sharpness filter gain register are used in adaptive filter mode. To activate the adaptive filter control, the ED/HD sharpness filter and the ED/HD adaptive filter must be enabled. Refer to the register tables above for enabling and disabling the sharpness and adaptive filter.

The derivative of the incoming signal is compared to the three programmable threshold values; that is the ED/HD adaptive filter (Threshold A, Threshold B, and Threshold C) registers. These registers (thold_a[7:0], thold_b[7:0] and thold_c[7:0]) are described below. The recommended threshold range is 16 to 235, although any value in the range of 0 to 255 can be used.

thold_a[**7:0**], Encoder Map, *Address 0xE45B[7:0]*
This register is used to set the ED/HD adaptive filter threshold A.

thold_b[**7:0**], Encoder Map, *Address 0xE45C[7:0]*
This register is used to set the ED/HD adaptive filter threshold B.

thold_c[**7:0**], Encoder Map, *Address 0xE45D[7:0]*
This register is used to set the ED/HD adaptive filter threshold C.

The edges can then be attenuated with the settings in the ED/HD adaptive filter (Gain 1, Gain 2, and Gain 3) registers. Refer to the registers fil_resp_aa[3:0], fil_resp_ab[3:0], fil_resp_bb[3:0], fil_resp_ca[3:0] and fil_resp_cb[3:0] for details on setting the adaptive filter gains.

fil_resp_ab[**3:0**], Encoder Map, *Address 0xE458[7:4]*
This signal is used to set the adaptive filter gain 1 for the ED/HD standard. This is value B.

Function

fil_resp_ab[3:0]	Description
0000 (default)	Gain B 0
0001	Gain B +1
	...
0111	Gain B +7
1000	Gain B -8
	...
1110	Gain B -2
1111	Gain B -1

fil_resp_aa[3:0], Encoder Map, *Address 0xE458[3:0]*
This signal is used to set the adaptive filter gain 1 for the ED/HD standard. This is value A.

Function

fil_resp_aa[3:0]	Description
0000 (default)	Gain A 0
0001	Gain A +1
	...
0111	Gain A +7
1000	Gain A -8
	...
1110	Gain A -2
1111	Gain A -1

fil_resp_bb[3:0], Encoder Map, *Address 0xE459[7:4]*
This signal is used to set the adaptive filter gain 2 for the ED/HD standard. This is value B.

Function

fil_resp_bb[3:0]	Description
0000 (default)	Gain B 0
0001	Gain B +1
	...
0111	Gain B +7
1000	Gain B -8
	...
1110	Gain B -2
1111	Gain B -1

fil_resp_cb[3:0], Encoder Map, *Address 0xE45A[7:4]*
This signal is used to set the adaptive filter gain 3 for the ED/HD standard. This is value B.

Function

fil_resp_cb[3:0]	Description
0000 (default)	Gain B 0
0001	Gain B +1
	...
0111	Gain B +7
1000	Gain B -8
	...
1110	Gain B -2
1111	Gain B -1

fil_resp_ca[3:0], Encoder Map, *Address 0xE45A[3:0]*
This signal is used to set the adaptive filter gain 3 for the ED/HD standard. This is value A.

Function

fil_resp_ca[3:0]	Description
0000 (default)	Gain A 0
0001	Gain A +1
	...
0111	Gain A +7
1000	Gain A -8
	...
1110	Gain A -2
1111	Gain A -1

7.4.18.3. ED/HD Adaptive Filter Modes

Two adaptive filter modes are available: mode A and mode B.

Mode A is used when the ED/HD adaptive filter mode control is set to 0. In this case, filter B (LPF) is used in the adaptive filter block. In addition, only the programmed values for Gain B in the ED/HD sharpness filter gain register and ED/HD adaptive filter (Gain 1, Gain 2, and Gain 3) registers are applied when needed. The Gain A values are fixed and cannot be changed.

Mode B is used when ED/HD adaptive filter mode control is set to 1. In this mode, a cascade of filter A and filter B is used. Both settings for Gain A and Gain B in the ED/HD sharpness filter gain register and ED/HD adaptive filter (Gain 1, Gain 2, and Gain 3) registers become active when needed. The mode is selected using adapt_bc.

adapt_bc, Encoder Map, *Address 0xE435[6]*
This bit is used to select the adaptive filter mode.

Function

adapt_bc	Description
0 (default)	Mode A
1	Mode B

7.4.18.4. ED/HD Sharpness Filter and Adaptive Filter Application Examples

Sharpness Filter Application

The ED/HD sharpness filter can be used to enhance or attenuate the Y video output signal. The register settings in Table 81 are used to achieve the results shown in Figure 123. Input data is generated by an external signal source. The reference in the table can be matched with the appropriate scope plot.

Table 81: ED/HD Sharpness Control Settings for Figure 123

Register	Register Setting	Reference[1]
0xE400	0xFC	
0xE401	0x10	
0xE402	0x20	
0xE430	0x00	
0xE431	0x81	
0xE440	0x00	a
0xE440	0x08	b
0xE440	0x04	c
0xE440	0x40	d
0xE440	0x80	e
0xE440	0x22	f

[1] See Figure 123.

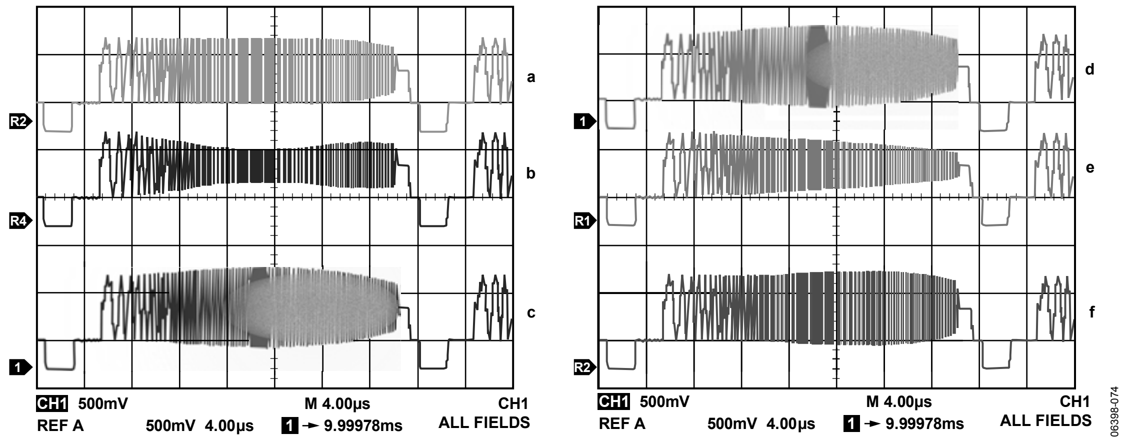


Figure 123: ED/ HD Sharpness Filter Control with Different Gain Settings for ED/HD Sharpness Filter Gain Values

Adaptive Filter Control Application

The register settings in Table 82 are used to obtain the results shown in Figure 125, that is, to remove the ringing on the input Y signal, as shown in Figure 124. Input data is generated by an external signal source.

Table 82: Register Settings for Figure 125

Register	Register Setting
0xE400	0xFC

Register	Register Setting
0xE401	0x38
0xE402	0x20
0xE430	0x00
0xE431	0x81
0xE435	0x80
0xE440	0x00
0xE458	0xAC
0xE459	0x9A
0xE45A	0x88
0xE45B	0x28
0xE45C	0x3F
0xE45D	0x64

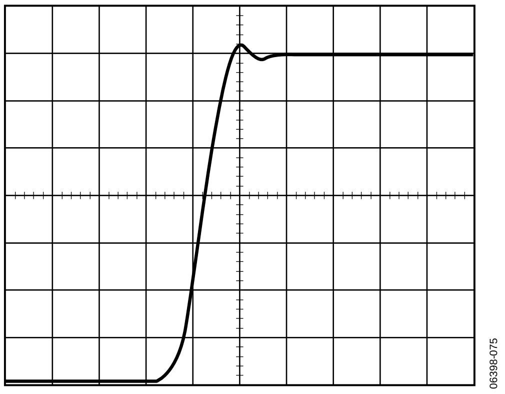


Figure 124: Input Signal to ED/HD Adaptive Filter

The effects of selecting between the two adaptive filter modes (using adapt_bc) can be seen in Figure 125 and Figure 126.

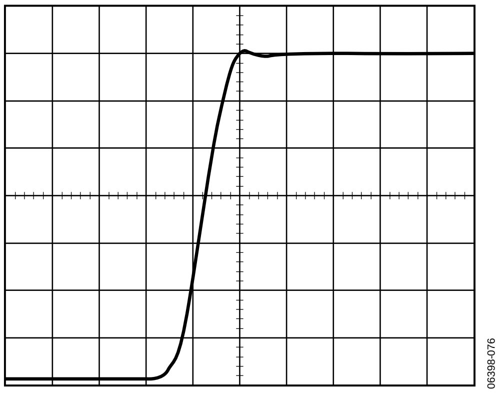


Figure 125: Output Signal from ED/HD Adaptive Filter (Mode A)

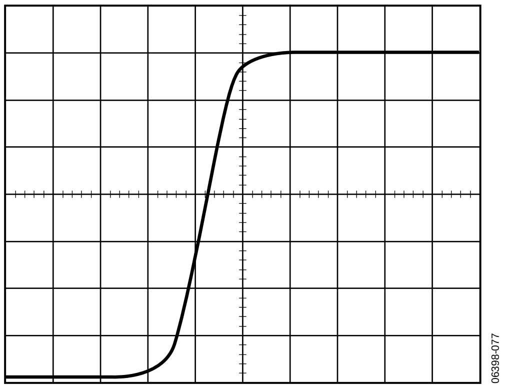


Figure 126: Output Signal from ED/HD Adaptive Filter (Mode B)

7.4.19. *SD Digital Noise Reduction*

The ADV8005 encoder block offers a feature for digital noise reduction (DNR). DNR is applied to the Y data only. A filter block selects the high frequency, low amplitude components of the incoming signal (DNR input select). The absolute value of the filter output is compared to a programmable threshold value (DNR threshold control). Two DNR modes are available: DNR mode and DNR sharpness mode. Refer to Figure 127.

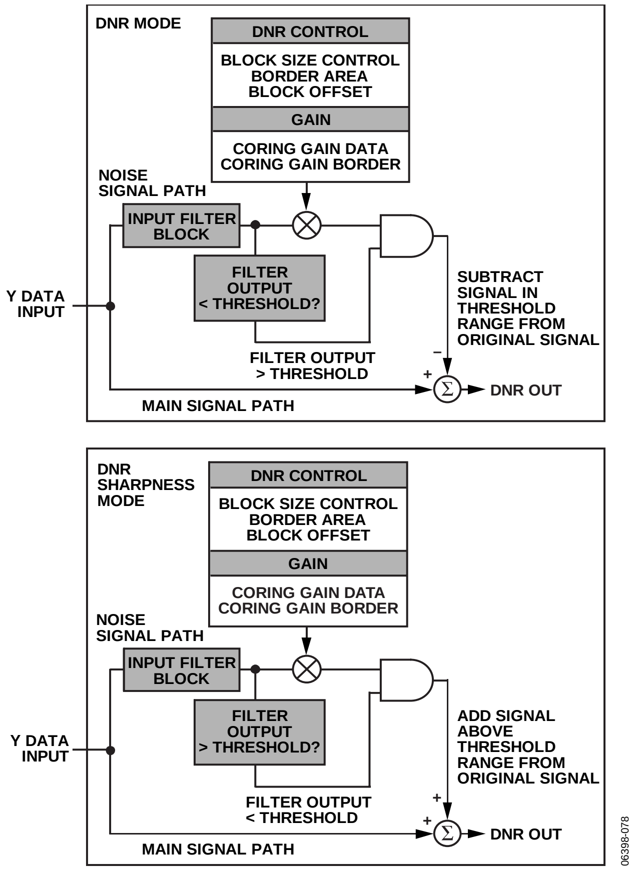


Figure 127: SD DNR Block Diagram

In DNR mode, if the absolute value of the filter output is smaller than the threshold, it is assumed to be noise. A programmable amount (coring gain border, coring gain data) of this noise signal is subtracted from the original signal. In DNR sharpness mode, if the absolute value of the filter output is less than the programmed threshold, it is assumed to be noise. Otherwise, if the level exceeds the threshold, now identified as a valid signal, a fraction of the signal (coring gain border, coring gain data) is added to the original signal to boost high frequency components and sharpen the video image.

In MPEG systems, it is common to process the video information in blocks of 8 pixels × 8 pixels for MPEG2 systems or 16 pixels × 16 pixels for

MPEG1 systems (block size control). DNR can be applied to the resulting block transition areas that are known to contain noise. Generally, the block transition area contains two pixels. It is possible to define this area to contain four pixels (border area).

It is also possible to compensate for variable block positioning or differences in YCrCb pixel timing with the use of the DNR block offset. The digital noise reduction registers are three 8-bit registers. They are used to control the DNR processing.

To enable the SD DNR feature, dnr_en must be programmed.

dnr_en, Encoder Map, *Address 0xE481[7]*
This bit is used to enable the SD Digital Noise Reduction (DNR) function.

Function

dnr_en	**Description**
1	Enabled
0 (default)	Disabled

7.4.19.1. Coring Gain Border

dnr_coring_gain_a[3:0] is the gain factor applied to border areas (refer to Figure 133 for more information on border areas). In DNR mode, the range of gain values is 0 to -1 in decrements of 1/8. This factor is applied to the DNR filter output that lies below the set threshold range. The result is then subtracted from the original signal.

In DNR sharpness mode, the range of gain values is 0 to 0.5 in increments of 1/16. This factor is applied to the DNR filter output that lies above the threshold range. The result is added to the original signal.

dnr_coring_gain_a[3:0], Encoder Map, *Address 0xE4A3[7:4]*
This signal is used to configure the coring gain border (in Digital Noise Reduction (DNR) mode, the values in brackets apply).

Function

dnr_coring_gain_a[3:0]	**Description**
0000 (default)	No gain
0001	+1/16 [−1/8]
0010	+2/16 [−2/8]
0011	+3/16 [−3/8]
0100	+4/16 [−4/8]
0101	+5/16 [−5/8]
0110	+6/16 [−6/8]
0111	+7/16 [−7/8]
1000	+8/16 [−1]

7.4.19.2. Coring Gain Data

dnr_coring_gain_b[3:0] is the gain factor applied to the luma data inside the MPEG pixel block. In DNR mode, the range of gain values is 0 to -1 in decrements of 1/8. This factor is applied to the DNR filter output that lies below the set threshold range. The result is then subtracted from the original signal.

In DNR sharpness mode, the range of gain values is 0 to 0.5 in increments of 1/16. This factor is applied to the DNR filter output that lies above the threshold range. The result is added to the original signal. Figure 128 explains the difference between SD DNR border gain and data gain.

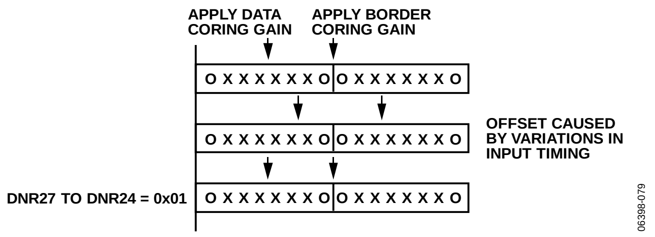

APPLY DATA CORING GAIN
APPLY BORDER CORING GAIN
O X X X X X X O O X X X X X X O
O X X X X X X O O X X X X X X O
OFFSET CAUSED BY VARIATIONS IN INPUT TIMING
DNR27 TO DNR24 = 0x01
O X X X X X X O O X X X X X X O
06398-079

Figure 128: SD DNR Offset Control

dnr_coring_gain_b[3:0], Encoder Map, *Address 0xE4A3[3:0]*
This signal is used to configure the coring gain data (in Digital Noise Reduction (DNR) mode, the values in brackets apply).

Function

dnr_coring_gain_b[3:0]	Description
0000 (default)	No gain
0001	+1/16 [−1/8]
0010	+2/16 [−2/8]
0011	+3/16 [−3/8]
0100	+4/16 [−4/8]
0101	+5/16 [−5/8]
0110	+6/16 [−6/8]
0111	+7/16 [−7/8]
1000	+8/16 [−1]

7.4.19.3. DNR Threshold

dnr_threshold[5:0] is used to define the threshold value in the range of 0 to 63. The range is an absolute value.

dnr_threshold[5:0], Encoder Map, *Address 0xE4A4[5:0]*
This signal is used to configure the Digital Noise Reduction (DNR) threshold.

Function

dnr_threshold[5:0]	Description
000000 (default)	0
000001	1
	...
111110	62
111111	63

7.4.19.4. Border Area

When blk_border_2 is set to 1, the block transition area can be defined to consist of four pixels. If this bit is set to logic 0, the border transition area consists of two pixels, where one pixel refers to two clock cycles at 27 MHz.

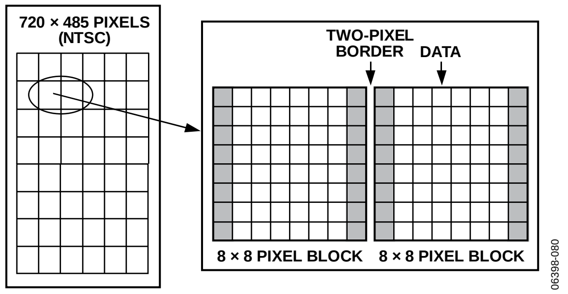


Figure 129: SD DNR Border Area

blk_border_2, Encoder Map, *Address 0xE4A4[6]*
This bit is used to select the Digital Noise Reduction (DNR) border area.

Function

blk_border_2	Description
0 (default)	2 pixels
1	4 pixels

7.4.19.5. Block Size Control

dnr_mpeg_1 is used to select the size of the data blocks to be processed. Setting the block size control function to 1 defines a 16 pixel × 16 pixel data block, and 0 defines an 8 pixel × 8 pixel data block, where one pixel refers to two clock cycles at 27 MHz.

dnr_mpeg_1, Encoder Map, *Address 0xE4A4[7]*
This bit is used to select the Digital Noise Reduction (DNR) block size.

Function

dnr_mpeg_1	Description
1	16 pixels
0 (default)	8 pixels

7.4.19.6. DNR Input Select Control

dnr_fmode_control[2:0] is used to select the filter which is applied to the incoming Y data. The signal that lies in the pass band of the selected filter is the signal that is DNR processed. Figure 130 shows the filter responses selectable with this control.

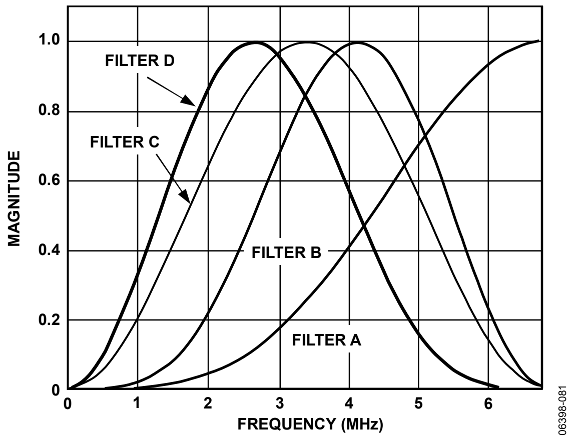


Figure 130: SD DNR Input Filter Select

dnr_fmode_control[2:0], Encoder Map, *Address 0xE4A5[2:0]*
This signal is used to configure the Digital Noise Reduction (DNR) input filter.

Function

dnr_fmode_control[2:0]	Description
001	Filter A
010	Filter B
011	Filter C
100	Filter D

7.4.19.7. DNR Mode Control

DNR works on the principle of defining low amplitude, high frequency signals as probable noise and subtracting this noise from the original signal.

In DNR mode, it is possible to subtract a fraction of the signal that lies below the set threshold, assumed to be noise, from the original signal. The threshold is set using dnr_threshold[5:0].

When dnr_enable_sharpness is enabled, it is possible to add a fraction of the signal that lies above the set threshold to the original signal because this data is assumed to be valid data and not noise. The overall effect is that the signal is boosted (similar to using the extended SSAF filter).

dnr_enable_sharpness, Encoder Map, *Address 0xE4A5[3]*
This bit is used to select the Digital Noise Reduction (DNR) mode.

Function

dnr_enable_sharpness	Description
0 (default)	DNR mode
1	DNR sharpness mode

7.4.19.8. DNR Block Offset Control

blk_offset[3:0] allows a shift of the data block of 15 pixels maximum. The coring gain positions are fixed. The block offset shifts the data in steps of one pixel so that the border coring gain factors can be applied at the same position regardless of variations in input timing of the data.

blk_offset[3:0], Encoder Map, *Address 0xE4A5[7:4]*
This signal is used to configure the Digital Noise Reduction (DNR) block offset.

Function

blk_offset[3:0]	Description
0000 (default)	0 pixel offset
0001	One pixel offset
	...
1110	14 pixel offset
1111	15 pixel offset

7.4.19.9. SD Active Video Edge Control

The ADV8005 encoder core is able to control fast rising and falling signals at the start and end of active video in order to minimize ringing artifacts.

When the active video edge control feature is enabled, the first three pixels and the last three pixels of the active video on the luma channel are scaled so that maximum transitions on these pixels are not possible. This feature is highlighted in Figure 131.

At the start of active video, the first three pixels are multiplied by 1/8, 1/2, and 7/8, respectively. Approaching the end of active video, the last three pixels are multiplied by 7/8, 1/2, and 1/8, respectively. All other active video pixels pass through unprocessed. Figure 132 and Figure 133 show the difference between having this feature enabled and disabled. This feature can be enabled using slope_en.

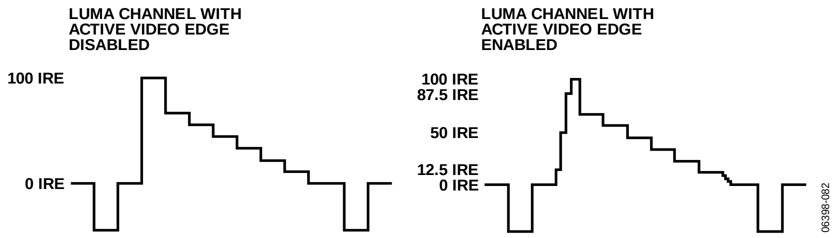

LUMA CHANNEL WITH ACTIVE VIDEO EDGE DISABLED
100 IRE
0 IRE
LUMA CHANNEL WITH ACTIVE VIDEO EDGE ENABLED
100 IRE
87.5 IRE
50 IRE
12.5 IRE
0 IRE
06398-082

Figure 131: Example of Active Video Edge Functionality

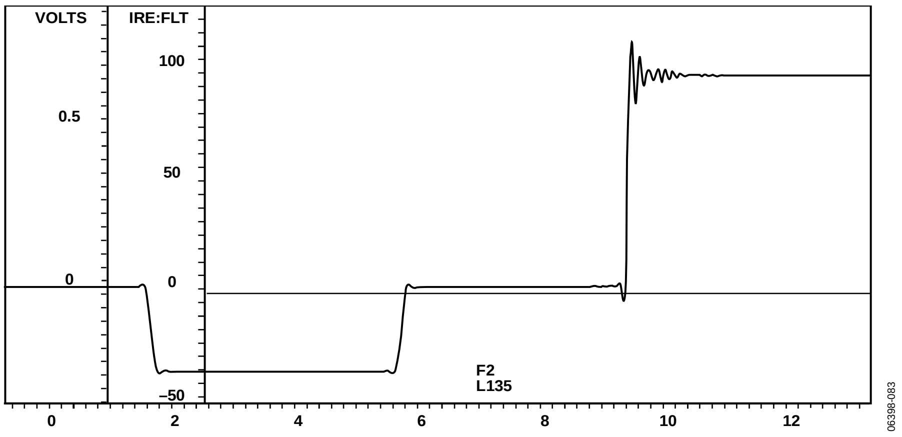


Figure 132: Example of Video Output with SD Active Video Edge Control Disabled

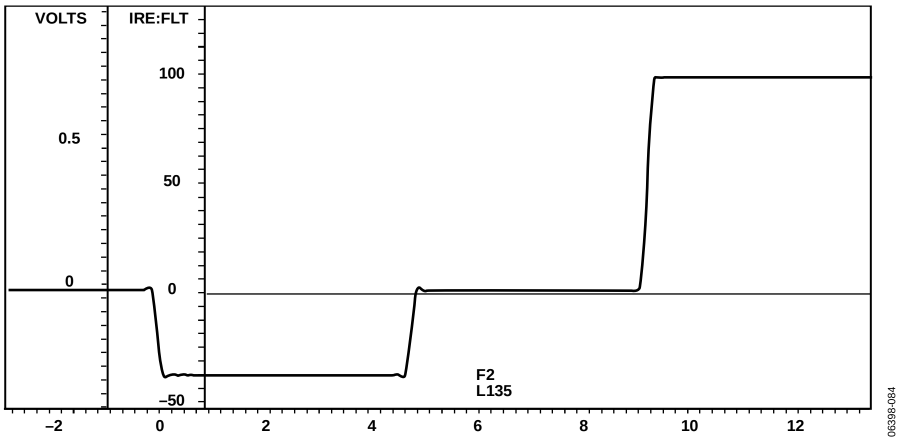


Figure 133: Example of Video Output with SD Active Video Edge Control Enabled

slope_en, Encoder Map, *Address 0xE482[7]*
This bit is used to enable the SD active video edge control.

Function

slope_en	Description
1	Enabled
0 (default)	Disabled

If a pattern with sharp transitions is being output through the encoder and the user does not want slope_en to have an effect because it softens the edges, it is possible to use sd_under_limiter[1:0] and sd_y_min_value to control possible ringing artifacts on the output of the encoder.

sd_under_limiter[1:0], Encoder Map, *Address 0xE489[1:0]*
This signal is used to configure the SD undershoot limiter.

Function

sd_under_limiter[1:0]	Description
00 (default)	Disabled
01	-11IRE
10	-6IRE
11	-1.5IRE

sd_y_min_value, Encoder Map, *Address 0xE48A[6]*
This bit is used to configure the SD minimum luma value.

Function

sd_y_min_value	Description
0 (default)	-40IRE
1	-7.5IRE

7.5. VERTICAL BLANKING INTERVAL

The ADV8005 is capable of accepting input VBI data (for example, CGMS, WSS, and CCAP) in SD, ED and HD modes. If VBI is disabled, for SD mode, see vbi_open, for HD mode, see vbi_data_en. VBI data is not present at the encoder output and the entire VBI is blanked. These control bits are valid in all modes.

For SMPTE 293M (525p), VBI data can be inserted on Lines 13 to 42 of each frame. For ITU-R BT.1358 (625p), VBI data can be inserted on Lines 6 to 43 For NTSC, VBI data can be inserted on Lines 10 to 20. For PAL, VBI data can be inserted on Lines 7 to 22.

If CGMS is enabled and VBI is disabled, the CGMS data is available at the output.

vbi_open, Encoder Map, *Address 0xE483[4]*
This bit is used to enable data on the Vertical Blanking Interval (VBI) to be accepted as valid data. This is valid for SD video data only.

Function

vbi_open	Description
1	Enabled
0 «	Disabled

vbi_data_en, Encoder Map, *Address 0xE483[4]*
This bit is used to enable data on the Vertical Blanking Interval (VBI) to be accepted as valid data. This is valid for SD video data only.

Function

vbi_data_en	Description
1	Enabled
0 (default)	Disabled

7.6. DAC CONFIGURATIONS

The ADV8005 encoder features six DACs which all operate in low-drive mode. Low-drive mode is defined as 4.33 mA full-scale current into a 300 Ω load, R_L.

The ADV8005 encoder has two R_{SET} pins which are used to control the full-scale DAC output current and, therefore, the DAC output voltage levels; this is achieved through a resistor connected between the R_{SET} pin and GND. For low-drive operation, both R_{SET1} and R_{SET2} must have a value of 4.12 kΩ, and R_L must have a value of 300 Ω. The resistors connected to the R_{SET1} and R_{SET2} pins should have a 1% tolerance.

The ADV8005 encoder uses two pins for compensating the DAC reference buffer, COMP1 and COMP2. A 2.2 nF capacitor should be connected from each of these pins to AVDD2.

7.6.1. *Voltage Reference*

The ADV8005 contains an on-chip voltage reference that can be used as a board level voltage reference via the V_{REF} pin. Alternatively, the ADV8005 can be used with an external voltage reference by connecting the reference source to the V_{REF} pin. For optimal performance, an external voltage reference such as the AD1580 is used with the ADV8005 encoder reference voltage. If an external voltage reference is not used, a 0.1 µF capacitor should be connected from the V_{REF} pin to AVDD2.

7.6.2. *Video Output Buffer and Optional Output Filter*

A video buffer is necessary on the DAC outputs to match the 300Ω output impedance of the ADV8005 encoder output to the 75Ω input impedance of the sink device. ADI produces a range of op amps suitable for this application, for example, the AD8061. For more information about line driver buffering circuits, refer to the relevant op amp datasheet.

An optional reconstruction (anti-imaging) low-pass filter (LPF) may be required on the ADV8005 encoder processor DAC outputs if the part is connected to a device that requires this filtering.

The filter specifications vary with the application. The use of 16× (SD), 8× (ED), or 4× (HD) oversampling can remove the requirement for a reconstruction filter altogether. Refer to Section 7.4.1 for more details on output oversampling.

For applications requiring an output buffer and reconstruction filter, the ADA4430-1, ADA4411-3, and ADA4410-6 integrated video filter buffers should be considered.

Table 83: Output Filter Requirements

Application	Oversampling	Cutoff Frequency (MHz)	Attenuation −50 dB at (MHz)
SD	2×	>6.5	20.5
SD	16×	>6.5	209.5
ED	1×	>12.5	14.5
ED	8×	>12.5	203.5
HD	1×	>30	44.25
HD	4×	>30	267

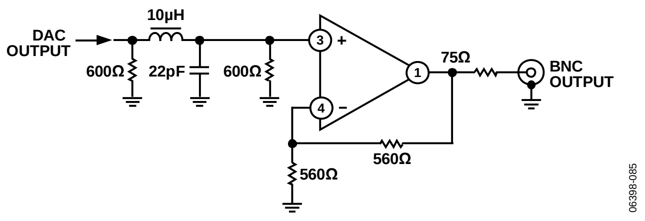


Figure 134: Example of Output Filter for SD, 16× Oversampling

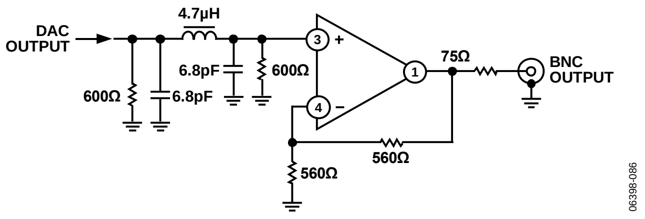


Figure 135: Example of Output Filter for ED, 8× Oversampling

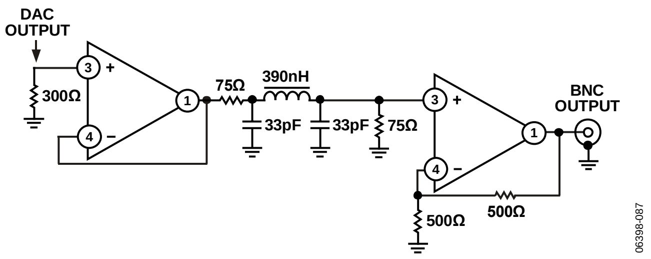


Figure 136: Example of Output Filter for HD, 4× Oversampling

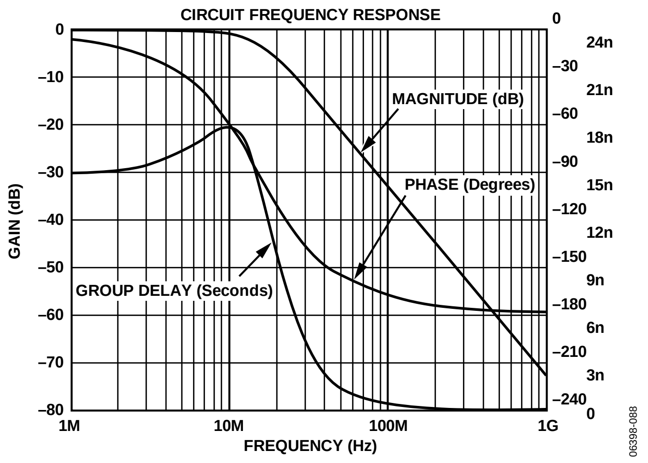


Figure 137: Output Filter Plot for SD, 16× Oversampling

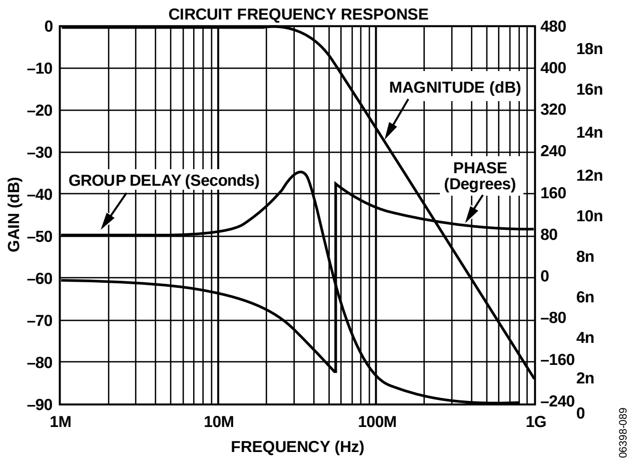


Figure 138: Output Filter Plot for ED, 8× Oversampling

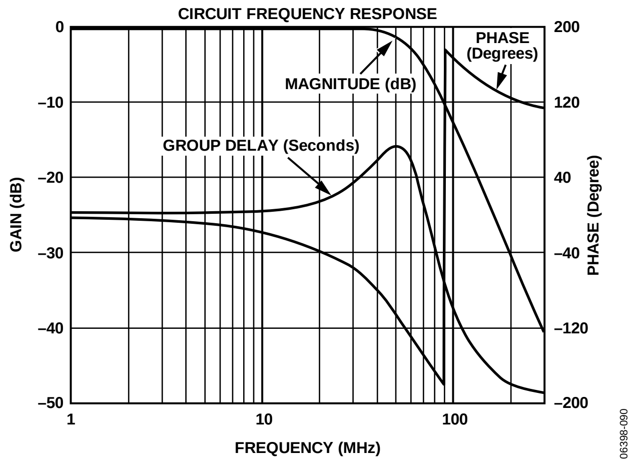


Figure 139: Output Filter Plot for HD, 4× Oversampling

8. INTERRUPTS

The ADV8005 has a comprehensive set of interrupt registers located in the IO Map and HDMI Main Maps of both the Serial Video Rx and HDMI transmitters. These interrupts can be used to indicate certain events in the Serial Video Rx section, OSD, and VSP, and also the HDMI Tx.

The ADV8005 features several interrupt controllers which handle three separate interrupt signals. These three interrupt signals are available on the interrupt pins INT0, INT1, and INT2. There is one interrupt available for the Serial Video Rx inputs, which is available for use on the interrupt INT2 pin. There is a shared interrupt available for both HDMI transmitters on INT1. There is also an interrupt pin made available to be used for a number of interrupts in the OSD core. These are available on INT0.

Note: The dual transmitter variants of ADV8005 are ADV8005KBCZ-8A, ADV8005KBCZ-8N and ADV8005KBCZ-8C. The single transmitter variant of ADV8005 is ADV8005KBCZ-8B. Any references to interrupts relating to HDMI Tx2 are not applicable to these parts.

8.1. INTERRUPT PINS

The ADV8005 features three dedicated interrupt pins, INT0, INT1, and INT2. These pins can be configured as open drain or standard IO pads and can be configured as outputs or inputs. By default, they are set to standard TTL inputs. The following registers are used for setting these pins.

int_pin_od_en[2:0], IO Map, *Address 0x1ACC[2:0]*
This signal is used to select whether the interrupt pins are configured as TTL or as open drain. INT0 is linked to the OSD interrupts, INT1 is linked to the HDMI TX interrupts and INT2 is linked to the Serial Video RX interrupts.

Function

int_pin_od_en[2:0]	Description
000 (default)	All interrupts TTL
001	INT0 open drain
010	INT1 open drain
100	INT2 open drain
111	All interrupts open drain

int_pin_oe[2:0], IO Map, *Address 0x1ACC[6:4]*
This signal is used to enable the INT0, INT1 and INT2 interrupt pins. INT0 is linked to the OSD interrupts, INT1 is linked to the HDMI TX interrupts and INT2 is linked to the Serial Video RX interrupts.

Function

int_pin_oe[2:0]	Description
000 (default)	All interrupts tristated
001	INT0 interrupt enabled
010	INT1 interrupt enabled
100	INT2 interrupt enabled
111	All interrupts enabled

8.1.1. *Interrupt Duration*

The interrupt duration can be programmed independently for interrupt pin INT2. When an interrupt event occurs, the interrupt pin INT2 becomes active with a programmable duration as described in this section.

intrq_dur_sel[1:0], IO Map, *Address 0x1A69[3:2]*
This signal is used to set the interrupt signal duration for the Serial Video RX interrupts output on pin INT2.

Function

intrq_dur_sel[1:0]	Description
00 (default)	4 Xtal periods
01	16 Xtal periods
10	64 Xtal periods
11	Active until cleared

Note: When the active until cleared interrupt duration is selected and the event that caused an interrupt ends, the interrupt persists until it is cleared or masked.

8.1.2. *Storing Masked Interrupts*

store_unmasked_irqs, IO Map, *Address 0x1A69[7]*
This bit is used to specify whether the HDMI status flags for any HDMI interrupt should be triggered regardless of whether the mask bits are set. This bit allows an HDMI interrupt to trigger and allows this interrupt to be read back through the corresponding status bit without triggering an interrupt on the interrupt pin. The status is stored until the clear bit is used to clear the status register and allows another interrupt to occur.

Function

store_unmasked_irqs	**Description**
0 (default)	Do not store triggered interrupts
1	Store triggered interrupts

8.2. SERIAL VIDEO RX INTERRUPTS

8.2.1. *Introduction*

This section describes the interrupt support provided for the Serial Video Rx on the ADV8005. The Serial Video Rx interrupts are OR'd together and connected to the ADV8005 INT2 pin.

The ADV8005 Serial Video Rx interrupt architecture provides the following types of bits:

- Raw bits
- Status bits
- Interrupt mask bits
- Clear bits

Raw bits are defined as being either edge-sensitive or level-sensitive. The following compares an edge-sensitive interrupt and a level-sensitive interrupt to demonstrate the difference.

level_sensitive_int_raw, IO, *Address 0xXX (Read Only)*
This readback indicates the raw status of the level sensitive interrupt. This bit is set to one when a condition occurs and is reset to 0 when the condition is no longer apparent.

Function

level_sensitive_int_raw	**Description**
0 «	Event/condition not currently occurring
1	Event/condition currently occurring

edge_sensitive_int_raw, IO, *Address 0xXX (Read Only)*
This readback indicates the status of the edge sensitive interrupt. When set to 1, it indicates that an event has occurred. Once set, this bit remains high until the interrupt is cleared via edge_sensitive_int_clr.

Function

edge_sensitive_int_raw	**Description**
0 «	No event/condition occurred
1	Event/condition occurred

Level-sensitive bit, level_sensitive_int_raw, always represents the current status of whether or not a particular event or condition is occurring, e.g. if the part is receiving AVI InfoFrames. It is not a latched bit and never requires to be cleared.

Edge-sensitive bit, edge_sensitive_int_raw, indicates that a transient event or condition has occurred; it is latched and it needs to be cleared. This approach is adopted for important events which have a transient nature e.g. if the part has received a new AVI InfoFrame. If edge_sensitive_int_raw did not latch and returned to 0 sometime after the event occurred, the user could miss the fact that the event or condition occurred. Therefore, edge-sensitive raw bits do not truly represent the current status; instead, they represent the status of an edge event that happened in the past. To clear a latched bit, the user must set the corresponding clear bit to 1.

Figure 140, Figure 141 and Figure 142 provide a graphical example of what how edge and level sensitive interrupts operate.

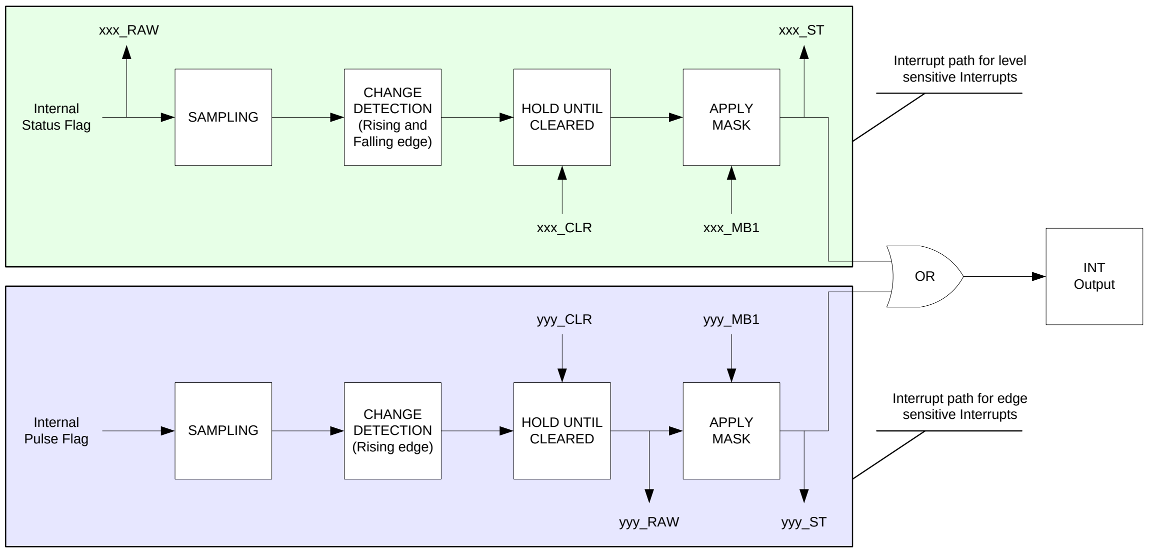


Figure 140: Level and Edge-Sensitive Raw, Status and Interrupt Generation

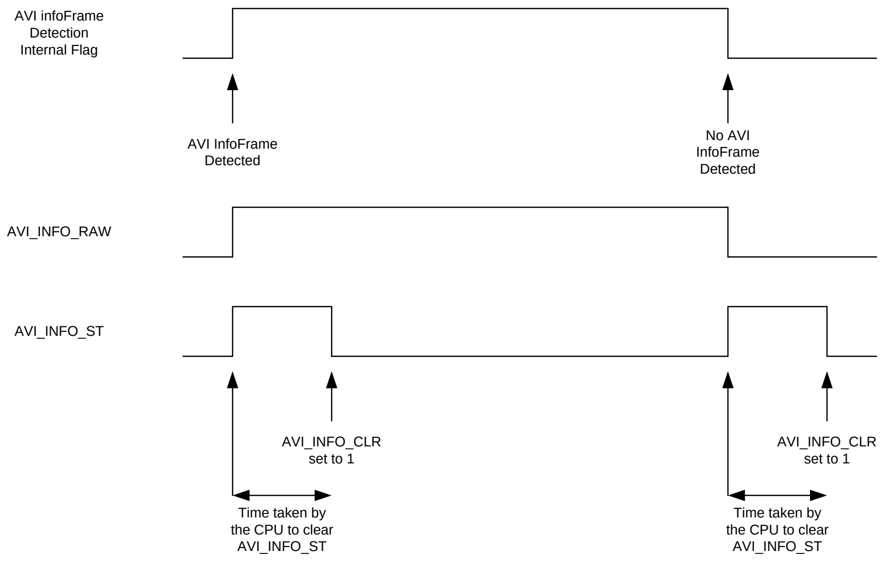

AVI infoFrame Detection Internal Flag
AVI InfoFrame Detected
No AVI InfoFrame Detected
AVI_INFO_RAW
AVI_INFO_ST
AVI_INFO_CLR set to 1
AVI_INFO_CLR set to 1
Time taken by the CPU to clear AVI_INFO_ST
Time taken by the CPU to clear AVI_INFO_ST

Figure 141: AVI_INFO_RAW and AVI_INFO_ST Timing

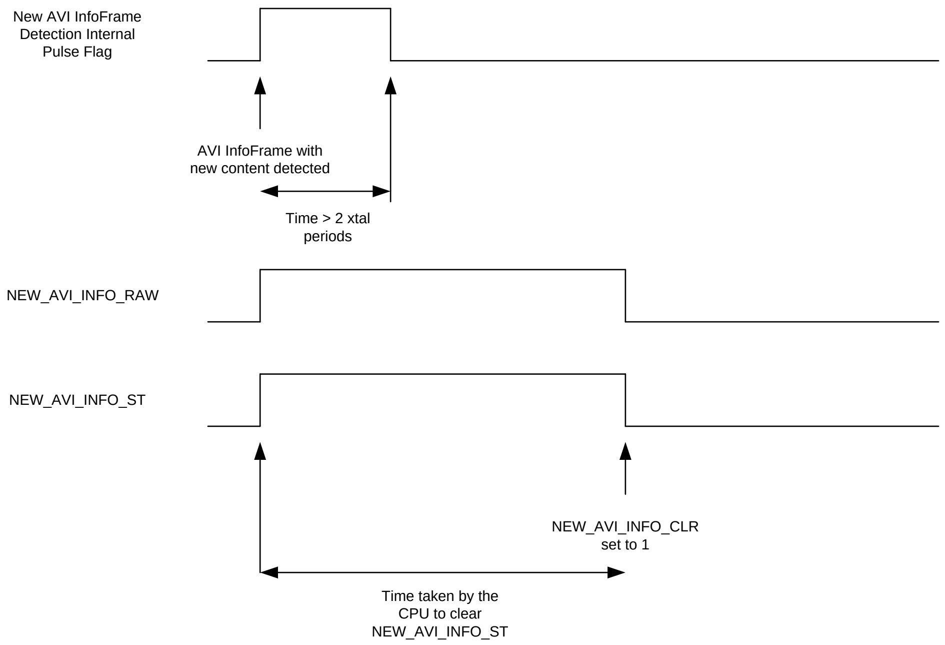

New AVI InfoFrame Detection Internal Pulse Flag
AVI InfoFrame with new content detected
Time > 2 xtal periods
NEW_AVI_INFO_RAW
NEW_AVI_INFO_ST
NEW_AVI_INFO_CLR set to 1
Time taken by the CPU to clear NEW_AVI_INFO_ST

Figure 142: NEW_AVI_INFO_RAW and NEW_AVI_INFO_ST Timing

All raw bits have corresponding status bits. The status bits always work in the same manner whether the raw bit is edge or level-sensitive. Status bits have the following characteristics:

- Enabled by setting the corresponding interrupt mask bit
- Always latched and must be cleared by the corresponding clear bit

For a given interrupt; when the interrupt mask bit is set, the interrupt status bit goes high and an interrupt is generated on the INT2 pin if the interrupt raw bit changes state. To return the interrupt status bit to low, the interrupt clear bit must be set. The status bits, interrupt mask bit and clear bits for level_sensitive_int and edge_sensitive_int are described here for completeness.

level_sensitive_int_st, IO, *Address 0xXX (Read Only)*
This readback indicates the latched status of the level_sensitive_int_raw signal. This bit is only valid if enabled via the corresponding INT1 interrupt mask bit. Once set, this bit remains high until the interrupt is cleared level_sensitive_int_clr.

Function

level_sensitive_int_st	Description
0 «	level_sensitive_int_raw did not change state
1	level_sensitive_int_raw changed state

edge_sensitive_int _st, IO, *Address 0xXX (Read Only)*
This readback indicates the latched status for edge_sensitive_int_raw. This bit is only valid if enabled via the corresponding INT1 interrupt mask bit. Once set, this bit remains high until the interrupt is cleared via edge_sensitive_int_clr.

Function

edge_sensitive_int_st	Description
0 «	edge_sensitive_int_raw not changed state
1	edge_sensitive_int_raw changed state

level_sensitive_int_clr, IO, *Address 0xXX (Self-Clearing)*
This control is used to clear the level_sensitive_int_st bits. This is a self clearing bit.

Function

level_sensitive_int_clr	Description
0 «	No function
1	Clear level_sensitive_int_st

edge_sensitive_int _clr, IO, *Address 0xXX (Self-Clearing)*
This control is used to clear the edge_sensitive_int_raw and edge_sensitive_int_st bits. This is a self clearing bit.

Function edge_sensitive_int_clr	Description
0 «	No function
1	Clear edge_sensitive_int_raw and edge_sensitive_int_st

level_sensitive_int_mb2, IO, *Address 0xXX[0]*
This control is used to set the INT2 interrupt mask for the level_sensitive_int interrupt. When set, when the level sensitive interrupt event triggers and an interrupt is generated on INT2.

Function level_sensitive_int_mb2	Description
0 «	Disable level_sensitive_int detection interrupt for INT2
1	Enable level_sensitive_int detection interrupt for INT2

edge_sensitive_int _mb2, IO, *Address 0xXX*
This control is used to set the INT2 interrupt mask for the edge_sensitive_int interrupt. When set, a new edge sensitive interrupt event will cause edge_sensitive_int_st to be set and an interrupt will be generated on INT2.

Function edge_sensitive_int_mb2	Description
0 «	Disable edge_sensitive_int detection interrupt for INT2
1	Enable edge_sensitive_int detection interrupt for INT2

In this section, all raw bits are classified as being triggered by either level-sensitive or edge-sensitive events, with the following understanding of the terminology.

Level-sensitive events are events that are generally either high or low and which are not expected to change rapidly. The raw bit for level-sensitive events is not latched and, therefore, always represents the true real-time status of the event in question.
Edge-sensitive events are events that only exist for an instant. The raw bits for edge-sensitive events are latched and, therefore, represent the occurrence of an edge-sensitive event that happened in the past. Raw bits for edge-sensitive events must be cleared by the corresponding clear bit.

8.2.2. *Interrupt Architecture Overview*

The following is a complete list of Serial Video Rx interrupts, their mode of operation (edge or level sensitive) and a description of each interrupt.

Table 84: Serial Video Rx Level Sensitive Interrupts

Interrupt	Mode of Operation	Description
rx_cable_det_raw/st/mb1/clr	Level sensitive	Used to detect if the Serial Video inputs are connected to an upstream IC
rx_tmdspll_lck_raw/st/mbx/clr	Level sensitive	Used to indicate if the TMDS PLL has locked to the incoming TMDS clock
rx_tmds_clk_det_raw/st/mbx/clr	Level sensitive	Used to indicate activity on the TMDS clock line
rx_video_3d_raw/st/mbx/clr	Level sensitive	Used to indicate if the incoming video is 3D format
rx_av_mute_raw/st/mbx/clr	Level sensitive	Used to indicate the AVMUTE value from the general control packet
rx_hdmi_mode_raw/st/mbx/clr	Level sensitive	Used to indicate if the incoming video is HDMI mode or DVI mode
rx_gen_ctl_pckt_raw/st/mbx/clr	Level sensitive	Used to indicate if a general control packet has been detected
rx_gamut_mdata_ pckt_raw/st/mbx/clr	Level sensitive	Used to indicate if a gamut metadata packet has been detected
rx_isrc2_pckt_raw/st/mbx/clr	Level sensitive	Used to indicate if an ISRC2 packet has been detected
rx_isrc1_pckt_raw/st/mbx/clr	Level sensitive	Used to indicate if an ISRC1 packet has been detected
rx_vs_info_frm_raw/st/mbx/clr	Level sensitive	Used to indicate if a vendor specific InfoFrame has been detected
rx_ms_info_frm_raw/st/mbx/clr	Level sensitive	Used to indicate if an MPEG source InfoFrame has been detected
rx_spd_info_frm_ raw/st/mbx/clr	Level sensitive	Used to indicate if an SPD InfoFrame has been detected
rx_avi_info_frm_raw/st/mbx/clr	Level sensitive	Used to indicate if an AVI InfoFrame has been detected

Table 85: Serial Video Rx Edge Sensitive Interrupts

Interrupt	**Mode of Operation**	**Description**
rx_vs_inf_cks_err_ edge_raw/st/mb2/clr	Edge sensitive	Used to indicate if there was an error with the vendor specific InfoFrame
rx_ms_inf_cks_err _edge_ raw/st/mb2/clr	Edge sensitive	Used to indicate if there was an error with the MPEG source InfoFrame
rx_spd_inf_cks_er r_edge_raw/st/mb2/clr	Edge sensitive	Used to indicate if there was an error with a SPD InfoFrame
rx_avi_inf_cks_err _edge_raw/st/mb2/clr	Edge sensitive	Used to indicate if there was an error with the AVI InfoFrame
rx_deepcolor_chn g_edge_raw/st/mb2/clr	Edge sensitive	Used to indicate if the incoming video is deep color. The exact mode can be determined by reading the DEEP_COLOR_MODE register
rx_tmds_clk_chng _edge_raw/st/mb2/clr	Edge sensitive	Used to indicate if the incoming TMDS clock has changed frequency
rx_pkt_err_edge_ raw/st/mb2/clr	Edge sensitive	Used to indicate if there was an error with any HDMI packet
rx_gamut_mdata_ pckt_edge_raw/st/mb2/clr	Edge sensitive	Used to indicate if a gamut metadata packet was detected
rx_isrc2_pckt_edg eraw/st/mb2/clr	Edge sensitive	Used to indicate if an ISRC2 packet was detected
rx_isrc1_pckt_edg eraw/st/mb2/clr	Edge sensitive	Used to indicate if an ISRC1 packet was detected
rx_vs_info_frm_e dge_raw/st/mb2/clr	Edge sensitive	Used to indicate if a vendor specific InfoFrame was detected
rx_ms_info_frm_e dge_raw/st/mb2/clr	Edge sensitive	Used to indicate if an MPEG source InfoFrame was detected
rx_spd_info_frm_ edge_raw/st/mb2/clr	Edge sensitive	Used to indicate if a source product descriptor InfoFrame was detected
rx_avi_info_frm_e dge_raw/st/mb2/clr	Edge sensitive	Used to indicate if an AVI InfoFrame was detected

8.2.2.1. *Multiple Interrupt Events*

If an interrupt event occurs, and then a second interrupt event occurs before the system controller has cleared or masked the first interrupt event, the ADV8005 does not generate a second interrupt signal. The system controller should check all unmasked interrupt status bits as more than one may be active.

8.2.3. *Serial Video Interrupts Validity Checking Process*

All Serial Video interrupts have a set of conditions that must be taken into account for validation in the system firmware. When the ADV8005 alerts the system controller with a Serial Video interrupt, the host must check that the following validity conditions for that interrupt are met before processing that interrupt. This is valid for all the interrupts described above.

- ADV8005 is configured in HMDI mode
- rx_tmds_clk_det_raw is set to 1 if the Serial Video Rx input is being used
- rx_tmdspll_lck_raw bit is set to 1

8.3. VSP AND OSD SECTION

This section describes the interrupts provided by the ADV8005 OSD and VSP section. These interrupts are not accessed through the I2C interface as the interrupts for the Serial Video Rx and HDMI Tx are; these interrupts are accessed through the SPI interface. These interrupts are not documented in detail as they are handled transparently to the user by the *Blimp OSD* software tool. Interrupts from this section are output on the INT0 pin for use by the system microcontroller.

8.3.1. *Interrupt Architecture Overview*

The following three interrupts are required by the VSP and OSD section:

Table 86: VSP and OSD Interrupts

Interrupt	Description
OSD_CFG_DONE	Used to indicate to the system controller that the configuration within the ADV8005 RAM memories has completed
DMA_IRQ	Used to indicate to the system controller that the current DMA operation has taken place
DMA_RAM_IRQ	Used to indicate to the system controller that the DMA hardware block can be read from/written to by SPI
TIMER_IRQ	Used to indicate to the system controller that a timer has expired
ANIM_DONE_IRQ	Used to indicate to the system controller that an animation has completed

The following controls are available to the user for indicating interrupts on the VSP and OSD interrupts.

vsp_int_pol[1:0], IO Map, *Address 0x1A76[3:2]*
This signal is used to control the VSP interrupt polarity.

Function

vsp_int_pol[1:0]	Description
00 (default)	VSP interrupt is logical AND of VSP/OSD interrupts
01	VSP interrupt is inverted logical AND of VSP/OSD interrupts
10	VSP interrupt is logical OR of VSP/OSD interrupts
11	VSP interrupt is inverted logical OR of VSP/OSD interrupts

8.4. HDMI TX CORE

8.4.1. *Introduction*

This section describes the interrupt support provided for the HDMI Tx cores of the ADV8005. The HDMI Tx interrupts are OR'd together and connected to the ADV8005 INT1 pin.

The ADV8005 HDMI Tx interrupt architecture provides the following types of bits:

- Interrupt status/clear bits
- Interrupt mask bits

The interrupt status/clear bits are dual purpose; when an interrupt event or condition occurs, if the interrupt mask bit is set, the status bit gets latched to 1. The interrupt can only be cleared by writing a value of 1 to the status/clear bit.

The interrupts mask bits are used to selectively activate an interrupt bit on the interrupt out pin INT1. The interrupt output pin is active when one or more interrupts bits are set and their corresponding interrupt mask bit is also set. Note that any given mask bit does not affect its corresponding interrupt bit but only affects the level on the interrupt output pin INT1. The enables for all the HDMI transmitter interrupts are described below.

8.4.2. *Interrupt Architecture Overview*

The following is a complete list of HDMI Tx interrupts and their descriptions:

Table 87: HDMI Tx Interrupts

Interrupt	Description
hpd_int/ hpd_int_en	Used to indicate the HDMI transmitter is connected to an HDMI Rx
rx_sense_int/ rx_sense_int_en	Used to detect if an HDMI Rx is connected to the HDMI transmitter
vsync_int/ vsync_int_en	Used to flag the falling edge on a VSync signal
edid_ready_int/ edid_ready_int_en	Used to indicate if the HDMI Rx EDID is ready for reading
hdcp_authenticated_int/ hdcp_authenticated_int_en	Used to indicate if the HDCP protocol has been authenticated
ri_ready_int/ ri_ready_int_en	Used to indicate if the HDCP Ri is ready
hdcp_error_int/hdcp_error_int_en	Used to indicate if a HDCP error has occurred
bksv_flag_int/ bksv_flag_int_en	Used to indicate if the BKSV flag is set

8.4.3. *HDMI Tx Interrupt Polarity*

This register is used to configure various logical operations which are available to the user when using the HDMI Tx interrupts.

tx_int_pol[1:0], IO Map, *Address 0x1A76[1:0]*
This signal is used to control the TX interrupt polarity.

Function

tx_int_pol[1:0]	Description
00 (default)	Tx interrupt is logical AND of Tx1/Tx2 interrupts
01	Tx interrupt is inverted logical AND of Tx1/Tx2 interrupts
10	Tx interrupt is logical OR of Tx1/Tx2 interrupts
11	Tx interrupt is inverted logical OR of Tx1/Tx2 interrupts

APPENDIX A

PCB LAYOUT RECOMMENDATIONS

The ADV8005 is a high precision, high speed, mixed signal device. It is important to have a well laid out PCB board in order to achieve the maximum performance from the part. The following sections are a guide for designing a board using the ADV8005.

Analogue/Digital Video Interface Outputs

The HDMI TMDS trace pairs must have a 100Ω differential impedance and should be routed in the shortest trace length possible to minimize the possibility of cross talk with other signals. The HDMI TMDS trace pairs must be routed on the same side of the PCB as the ADV8005 and should not be routed through vias to any other layers. A solid plane must be maintained underneath the HDMI TMDS trace pairs for their full trace length. Any external ESD suppressors should be placed as close as possible to the HDMI connector to reduce the impact on impedance TDR measurements.

If the ADV8005 is to support 3 GHz signals from the HDMI Txs, it is recommended the TMDS trace widths are set to 0.2 mm. The spacing of the traces, the height of the copper and the trace's height above the ground plan should all be controlled to maintain the trace impedance with this trace width.

The encoder analog outputs must have a 75Ω characteristic impedance and should be routed in the shortest trace length possible to minimize the possibility of cross talk with other signals. To assist in reducing cross talk, ground traces can be added between adjacent encoder analog outputs. The encoder analog outputs must be routed on the same side of the PCB as the ADV8005 and should not be routed through vias to any other layers. A solid plane must be maintained underneath the encoder analog outputs for their full trace length. The termination resistors on the encoder analog outputs should be kept as close as possible to the ADV8005. Any external filtering on the encoder outputs should be placed as close as possible to the analog connectors.

External DDR2 Memory Requirements

The ADV8005 must be placed as close to and on the same side of the PCB as the external DDR2 memories. Balanced T-routing should be used for all shared connections between the ADV8005 and the external DDR2 memories. All traces should be 75Ω and impedance controlled to ensure robust timing. Traces should be routed on the same side of the PCB as the devices where possible. If this is not possible, all traces should be kept on the outer layers.

All differential signals (for example, DDR_CK and DDR_CKB) should be treated as described above. These signals should be routed in parallel and on the same side of the PCB. Match the DDR_CK trace length to DDR_CKB trace length to 20 mils (0.5 mm). Any stubs on the clock lines should be kept as short as possible to avoid signal reflections.

The following 4-byte wide data lanes should be matched to within 50 mils on the PCB layout. The precise matching of these signals is critical.

- DDR3_DM3, DDR_DQS3, DDR_DQSB3, DDR_DQ31 – DDR_DQ24
- DDR2_DM2, DDR_DQS2, DDR_DQSB2, DDR_DQ23 – DDR_DQ16
- DDR1_DM1, DDR_DQS1, DDR_DQSB1, DDR_DQ15 – DDR_DQ8
- DDR0_DM0, DDR_DQS0, DDR_DQSB0, DDR_DQ7 – DDR_DQ0

Different byte lanes are to be matched to 200 mils (5.08 mm) of each other. 47Ω series termination resistors should be placed as close to the source (ADV8005) as possible on the following signals:

- Address signals – DDR_A12-DDR_A0 and DDR_BA0-DDR_BA2
- Clock differential signals – DDR_CK and DDR_CKB (use discrete resistors for these two signals)
- Control signal – DDR_CKE and command signals – DDR_CSB, DDR_RASB, DDR_CASB, and DDR_WEB
- Data mask signals – DDR_DM3-DDR_DM0

47 Ω series termination resistors should be placed in the middle of the trace on the following signals:

- Data bus signals – DDR_DQ31-DDR_DQ0
- Data strobe signals – DDR_DQS3 DDR_DQS3B-DDR_DQS0 DDR_DQS0B

The DDR2 reference voltage (DDR_VREF) should be routed as far away as possible from other signals to avoid any variations on the voltage. This trace should be wide. There should be a 100 nF decoupling cap close to the DDR2 reference voltage pins as well as the ADV8005 reference pin.

Power Supply Bypassing

It is recommended to bypass each power supply pin with a 0.1 uF and a 10 nF capacitor where possible. The fundamental idea is to have a bypass capacitor within 0.5 cm of each power pin.

Current should flow from the power plane to the capacitor to the power pin. The power connection should not be made between the capacitor and the power pin. Generally, the best approach is to place a via underneath the 10 nF capacitor pads down to the power plane (refer to Figure 143).

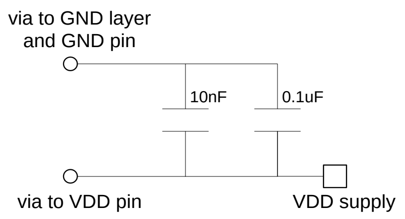


Figure 143: Recommended Power Supply Decoupling

It is recommended to individually filter all supplies to prevent switching noise on some supplies coupling onto other more sensitive supplies. For example, DVDD consumes a significant amount of current and will also suffer significant switching noise. DVDD must be isolated from more sensitive supplies such as PVDD3, PVDD5 and PVDD6.

The DVDD and DVDD_DDR supplies should be connected to the same supply – PVDD_DDR should be filtered from DVDD to provide a noise free power supply.

It is recommended to use a single ground plane for the ADV8005. Careful attention must be paid to the layout of any internal power supply planes when traces run on adjacent layers – traces on a layer directly above or below a power supply layer must not cross between two power supply planes as this will impact the return current paths.

General Digital Inputs and Outputs

The trace length that the digital inputs/outputs have to sink/source should be minimized. Longer traces have higher capacitance, which requires more current that can cause more internal digital noise. Shorter traces reduce the possibility of reflections. It is recommended to route traces in the shortest trace length possible and keep the number of layer transitions to a minimum.

If possible, the digital output driver capacitance loading should be limited to less than 15 pF. This can be accomplished easily by keeping traces short and by connecting the outputs to only one device. Loading the outputs with excessive capacitance increases the current transients inside the ADV8005, creating more digital noise on its power supplies.

Particular attention must be paid to the routing of clock and sync signals, for example, PCLK, OSD_CLK, HS, OSD_HS, VS, OSD_VS, DE, OSD_DE, XTALN, and XTALP. Any noise that gets onto these signals can add jitter to the system. Therefore, the trace length should be minimized, and digital or other high frequency traces should not be run near it.

XTAL and Load Cap Value Selection

The ADV8005 requires a 27 MHz crystal. Figure 144 shows an example of a reference clock circuit for the ADV8005. Special care must be taken when using a crystal circuit to generate the reference clock for the ADV8005. Small variations in reference clock frequency can impair the performance of the ADV8005.

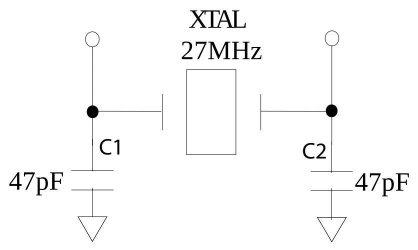


Figure 144: Crystal Circuit

These guidelines are followed to ensure correct operation:

- Use the correct frequency crystal (27 MHz recommended). Tolerance should be 50 ppm or better.
- Know the C_{load} for the crystal part number selected. The value of capacitors C1 and C2 must be matched to the C_{load} for the specific crystal part number in the user's system.

  To find C1 and C2, use the following formula:

  $C1 = C2 = 2(C_{load} - C_{stray}) - C_{pg}$

  where C_{stray} is usually 2 to 3 pF, depending on board traces and C_{pg} (pin-to-ground-capacitance) is 4 pF for the ADV8005.

  Example:

  C_{load} = 30 pF, $C1$ = 50 pF, $C2$ = 50 pF (in this case, 47 pF is the nearest real-life cap value to 50 pF)

Encoder Component Placement

External component placement must be carefully considered – they should be kept as far away from noisy circuits as possible, as close to the ADV8005 as possible and preferably on the same layer as the ADV8005. The external loop filter (connected to PVDD3), COMP, termination resistors, V_{REF}, and R_{SETx} circuits must all be laid out carefully otherwise noise may couple onto the SD or HD encoder outputs.

Any external filter and buffer components connected to the encoder analog outputs should be placed close to the ADV8005 to minimize the possibility of noise cross talk between neighboring circuitry. The encoder analog output traces should be kept as short as possible to reduce the possibility of any signal integrity issues and to minimize the effect of trace capacitance on output bandwidth.

HDMI Transmitter Component Placement

External component placement must be carefully considered – they should be kept as far away as possible from noisy circuits, as close to the ADV8005 as possible and preferably on the same layer as the ADV8005. The R_TX1 and R_TX2 resistors and PVDD5 and PVDD6 power supplies must all be carefully laid out otherwise the HDMI transmitter performance, for example, HDMI compliance testing, may be reduced.

Power Supply Design and Sequencing

The ADV8005 requires only two regulators, one 3.3 V and one 1.8 V. The recommended power supply design is illustrated in Figure 145.

If using more than one 1.8 V regulator to supply ADV8005, it must be ensured that DVDD_DDR, PVDD_DDR and DVDD are supplied by the same regulator.

The power-up sequence of the ADV8005 is as follows:

1. Hold RESET and PDN pins low.
2. Bring up the 3.3 V supplies (DVDD_IO, AVDD1, and AVDD2).
3. A delay of a minimum of 20 ms is required from the point in which the 3.3 V reaches its minimum recommended value (that is, 3.14 V) before powering up the 1.8 V supplies.
4. Bring up the 1.8 V supplies (DVDD, CVDD1, PVDD1, PVDD2, PVDD3, AVDD3, DVDD_DDR, and PVDD_DDR). These should be powered up together, that is, there should be a difference of less than 0.3 V between them.
5. $\overline{\text{RESET}}$ may be pulled high after supplies have been powered up.
6. A complete $\overline{\text{RESET}}$ is recommended after power up. This can be performed by the system microcontroller.

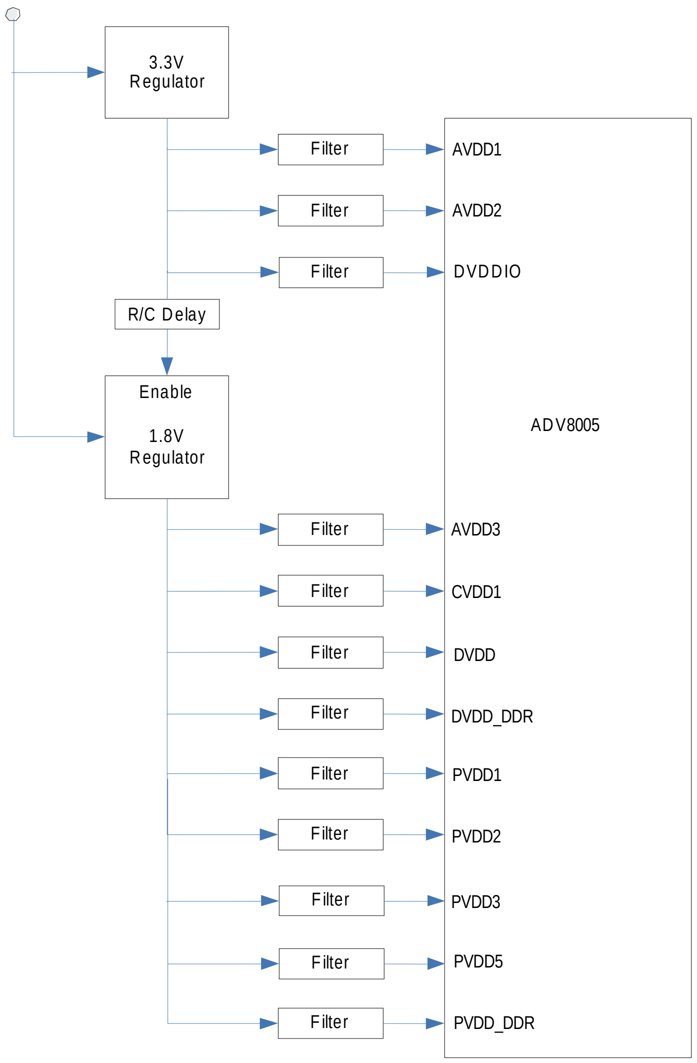


Figure 145: Power Supply Design

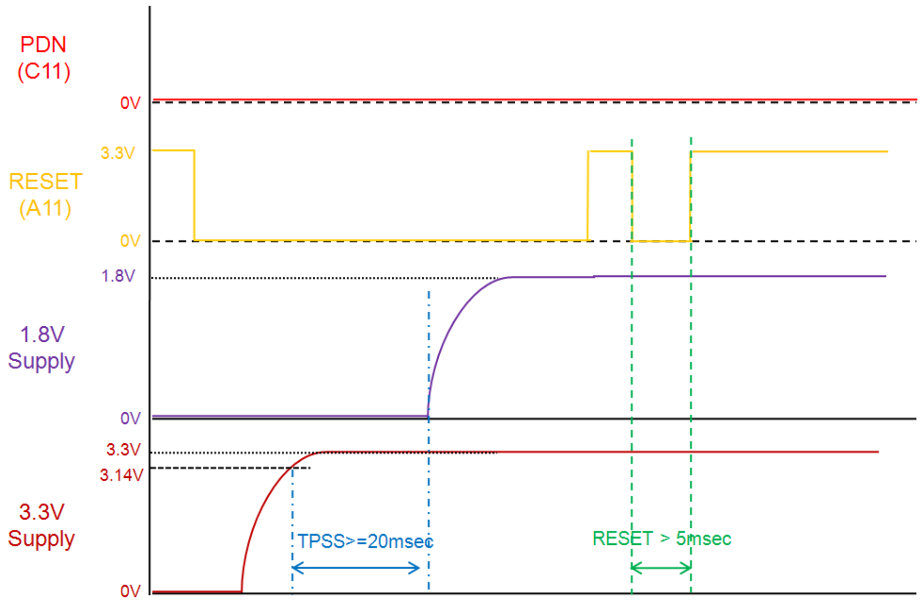


Figure 146: Power Supply Sequence

APPENDIX B

UNUSED PIN LIST

Location	Mnemonic	Type	Description if Unused	Pin Type
A1	OSD_IN[23]/EXT_DIN[7]	OSD video input/ miscellaneous digital	Float this pin as it is disabled by default.	Bi-directional digital IO
A2	OSD_DE	OSD video sync	Float this pin as it is disabled by default.	Bi-directional digital IO
A3	OSD_CLK/EXT_CLK	OSD video sync	Float this pin as it is disabled by default.	Bi-directional digital IO
A4	AUD_IN[1]	Audio input	Float this pin as it is disabled by default.	Digital input
A5	AUD_IN[2]	Audio input	Float this pin as it is disabled by default.	Digital input
A6	AUD_IN[5]	Audio input	Float this pin as it is disabled by default.	Digital input
A7	ARC2_OUT	Audio output	Connect this pin to ground through a 4.7kΩ resistor.	Digital output
A8	MOSI1	Serial port control	Float this pin as it is disabled by default.	Digital output
A9	SCK2	Serial port control	Float this pin as it is disabled by default.	Digital output
A10	$\overline{\text{CS2}}$	Serial port control	Float this pin as it is disabled by default.	Digital output
A11	$\overline{\text{RESET}}$	Miscellaneous digital	This pin must be connected.	N/A
A12	XTALN	Miscellaneous digital	This pin must be connected.	N/A
A13	PVDD2	Power	PLL Digital Supply Voltage (1.8 V).	N/A
A14	NC	No connect	Float this pin.	Digital output
A15	NC	No connect	Float this pin.	Digital output
A16	CVDD1	Power	Comparator Supply Voltage (1.8 V).	N/A
A17	RX_CN	Rx input	Float this pin.	Digital input
A18	RX_0N	Rx input	Float this pin.	Digital input
A19	RX_1N	Rx input	Float this pin.	Digital input
A20	RX_2N	Rx input	Float this pin.	Digital input
A21	CVDD1	Power	Comparator Supply Voltage (1.8 V).	N/A
A22	RSET1	Miscellaneous analog[1]	Float this pin.	Analog input
A23	VREF	Miscellaneous analog	Float this pin.	Analog input
B1	OSD_IN[21]/EXT_DIN[5]	OSD video input/ miscellaneous digital	Float this pin as it is disabled by default.	Bi-directional digital IO
B2	OSD_IN[22]/EXT_DIN[6]	OSD video input/ miscellaneous digital	Float this pin as it is disabled by default.	Bi-directional digital IO
B3	OSD_VS	OSD video sync	Float this pin as it is disabled by default.	Bi-directional digital IO
B4	AUD_IN[0]	Audio input	Float this pin as it is disabled by default.	Digital input
B5	AUD_IN[3]	Audio input	Float this pin as it is disabled by default.	Digital input
B6	SFL	SFL	Float this pin as it is disabled by default.	Digital input

Location	Mnemonic	Type	Description if Unused	Pin Type
B7	ARC1_OUT	Audio output	Connect this pin to ground through a 4.7kΩ resistor.	Digital output
B8	MISO1	Serial port control	Float this pin as it is disabled by default.	Digital output
B9	MOSI2	Serial port control	Float this pin as it is disabled by default.	Digital output
B10	MISO2	Serial port control	Float this pin as it is disabled by default.	Digital Input
B11	ALSB	I^{2}C control	Connect this pin to ground through a 4.7kΩ resistor.	Digital Input
B12	XTALP	Miscellaneous digital	This pin must be connected.	N/A
B13	PVDD1	Power	PLL Analog Supply Voltage (1.8 V).	N/A
B14	NC	No connect	Float this pin.	Digital output
B15	NC	No connect	Float this pin.	Digital output
B16	GND	GND	Ground.	N/A
B17	RX_CP	Rx input	Float this pin.	Digital input
B18	RX_0P	Rx input	Float this pin.	Digital input
B19	RX_1P	Rx input	Float this pin.	Digital input
B20	RX_2P	Rx input	Float this pin.	Digital input
B21	GND	GND	Ground.	N/A
B22	COMP1	Miscellaneous analog1	Connect this pin to ground through a 4.7kΩ resistor.	Analog input
B23	DAC4	Analog video output	Float this pin.	Analog output
C1	OSD_IN[19]/EXT_DIN[3]	OSD video input/ miscellaneous digital	Float this pin as it is disabled by default.	Bi-directional digital IO
C2	OSD_IN[20]/EXT_DIN[4]	OSD video input/ miscellaneous digital	Float this pin as it is disabled by default.	Bi-directional digital IO
C3	GND	GND	Ground.	N/A
C4	AUD_IN[4]	Audio input	Float this pin as it is disabled by default.	Digital input
C5	DSD_CLK	Audio input	Float this pin as it is disabled by default.	Digital input
C6	SCLK	Audio input	Float this pin as it is disabled by default.	Digital input
C7	SCL	I^{2}C control	This pin must be connected.	N/A
C8	SCK1	Serial port control	Connect this pin to ground through a 4.7kΩ resistor.	Digital input
C9	GND	GND	Ground.	N/A
C10	INT0	Miscellaneous digital	Connect this pin to ground through a 4.7kΩ resistor.	Digital output
C11	PDN	Miscellaneous digital	This pin must be connected.	N/A
C12	GND	GND	Ground.	N/A
C13	GND	GND	Ground.	N/A
C14	NC	No connect	Float this pin.	Digital output
C15	NC	No connect	Float this pin.	Digital output
C16	RX_HPD	Rx input	Float this pin.	Digital output
C17	AVDD1	Power	Serial Video Rx Inputs Analog Supply (3.3 V).	N/A
C18	GND	GND	Ground.	N/A
C19	GND	GND	Ground.	N/A

Location	Mnemonic	Type	Description if Unused	Pin Type
C20	AVDD1	Power	Serial Video Rx Inputs Analog Supply (3.3 V).	N/A
C21	AVDD1	Power	Serial Video Rx Inputs Analog Supply (3.3 V).	N/A
C22	DAC5	Analog video output	Float this pin.	Analog output
C23	DAC6	Analog video output	Float this pin.	Analog output
D1	OSD_IN[16]/EXT_DIN[0]	OSD video input/ miscellaneous digital	Float this pin as it is disabled by default.	Bi-directional digital IO
D2	OSD_IN[17]/EXT_DIN[1]	OSD video input/ miscellaneous digital	Float this pin as it is disabled by default.	Bi-directional digital IO
D3	OSD_IN[18]/EXT_DIN[2]	OSD video input/ miscellaneous digital	Float this pin as it is disabled by default.	Bi-directional digital IO
D4	GND	GND	Ground.	N/A
D5	DVDD_IO	Power	Digital Interface Supply (3.3 V).	N/A
D6	MCLK	Audio input	Float this pin as it is disabled by default.	Digital input
D7	SDA	I^{2}C control	This pin must be connected.	N/A
D8	$\overline{\text{CS1}}$	Serial port control	Float this pin as it is disabled by default.	Digital input
D9	GND	GND	Ground.	N/A
D10	INT1	Miscellaneous digital	Connect this pin to ground through a 4.7kΩ resistor.	Digital output
D11	INT2	Miscellaneous digital	Connect this pin to ground through a 4.7kΩ resistor.	Digital output
D12	DVDD_IO	Power	Digital Interface Supply (3.3 V).	N/A
D13	TEST1	Miscellaneous digital	Float this pin.	Digital output
D14	NC	No connect	Float this pin.	Digital output
D15	NC	No connect	Float this pin.	Digital output
D16	RX_5V	Rx input	Connect this pin to +5V.	Digital input
D17	NC	No connect	Float this pin.	Digital input
D18	NC	No connect	Float this pin.	Digital input
D19	RTERM	Serial Video Rx input	Float this pin.	Analog input
D20	AVDD2	Power	Encoder Analog Power Supply (3.3 V).	N/A
D21	AVDD2	Power	Encoder Analog Power Supply (3.3 V).	N/A
D22	DAC1	Analog video output	Float this pin.	Analog output
D23	DAC2	Analog video output	Float this pin.	Analog output
E1	OSD_IN[13]/VBI_SCK	OSD video input/ miscellaneous digital	Float this pin as it is disabled by default.	Bi-directional digital IO

Location	Mnemonic	Type	Description if Unused	Pin Type
E2	OSD_IN[14]/VBI_MOSI	OSD video input/ miscellaneous digital	Float this pin as it is disabled by default.	Bi-directional digital IO
E3	OSD_IN[15]/VBI_$\overline{\text{CS}}$	OSD video input/ miscellaneous digital	Float this pin as it is disabled by default.	Bi-directional digital IO
E4	DVDD_IO	Power	Digital Interface Supply (3.3 V).	N/A
E20	TEST2	Miscellaneous analog	Float this pin.	Digital output
E21	GND	GND	Ground.	N/A
E22	COMP2	Miscellaneous analog1	Float this pin.	Analog input
E23	DAC3	Analog video output	Float this pin.	Analog output
F1	OSD_IN[9]	OSD video input	Float this pin as it is disabled by default.	Bi-directional digital IO
F2	OSD_IN[10]	OSD video input	Float this pin as it is disabled by default.	Bi-directional digital IO
F3	OSD_IN[11]	OSD video input	Float this pin as it is disabled by default.	Bi-directional digital IO
F4	OSD_IN[12]	OSD video input/ miscellaneous digital	Float this pin as it is disabled by default.	Bi-directional digital IO
F20	RSET2	Miscellaneous analog1	Float this pin.	Analog input
F21	PVDD3	Power	Encoder PLL Supply (1.8 V).	N/A
F22	GND	GND	Ground.	N/A
F23	DNC	Do Not Connect	Float this pin.	Digital output
G1	OSD_IN[5]	OSD video input	Float this pin as it is disabled by default.	Bi-directional digital IO
G2	OSD_IN[6]	OSD video input	Float this pin as it is disabled by default.	Bi-directional digital IO
G3	OSD_IN[7]	OSD video input	Float this pin as it is disabled by default.	Bi-directional digital IO
G4	OSD_IN[8]	OSD video input	Float this pin as it is disabled by default.	Bi-directional digital IO
G7	GND	GND	Ground.	N/A
G8	GND	GND	Ground.	N/A
G9	GND	GND	Ground.	N/A
G10	DVDD	Power	Digital Power Supply (1.8 V).	N/A
G11	GND	GND	Ground.	N/A
G12	GND	GND	Ground.	N/A
G13	DVDD	Power	Digital Power Supply (1.8 V).	N/A
G14	GND	GND	Ground.	N/A
G15	GND	GND	Ground.	N/A
G16	GND	GND	Ground.	N/A
G17	GND	GND	Ground.	N/A
G20	ELPF1	Miscellaneous analog1	This pin must be connected.	N/A
G21	ELPF2	Miscellaneous analog1	This pin must be connected.	N/A

Location	Mnemonic	Type	Description if Unused	Pin Type
G22	GND	GND	Ground.	N/A
G23	AVDD3	Power	HDMI Analog Power Supply (1.8 V).	N/A
H1	OSD_IN[1]	OSD video input	Float this pin as it is disabled by default.	Bi-directional digital IO
H2	OSD_IN[2]	OSD video input	Float this pin as it is disabled by default.	Bi-directional digital IO
H3	OSD_IN[3]	OSD video input	Float this pin as it is disabled by default.	Bi-directional digital IO
H4	OSD_IN[4]	OSD video input	Float this pin as it is disabled by default.	Bi-directional digital IO
H7	GND	GND	Ground.	N/A
H8	GND	GND	Ground.	N/A
H9	GND	GND	Ground.	N/A
H10	GND	GND	Ground.	N/A
H11	GND	GND	Ground.	N/A
H12	GND	GND	Ground.	N/A
H13	GND	GND	Ground.	N/A
H14	GND	GND	Ground.	N/A
H15	GND	GND	Ground.	N/A
H16	GND	GND	Ground.	N/A
H17	GND	GND	Ground.	N/A
H20	GND	GND	Ground.	N/A
H21	GND	GND	Ground.	N/A
H22	TX1_2+	HDMI Tx1	Float this pin.	Digital output
H23	TX1_2–	HDMI Tx1	Float this pin.	Digital output
J1	DE	Digital video sync	Float this pin as it is disabled by default.	Digital input
J2	HS	Digital video sync	Float this pin as it is disabled by default.	Digital input
J3	OSD_HS	Digital video sync	Float this pin as it is disabled by default.	Bi-directional digital IO
J4	OSD_IN[0]	OSD video input	Float this pin as it is disabled by default.	Bi-directional digital IO
J7	DVDD	Power	Digital Power Supply (1.8 V).	N/A
J8	GND	GND	Ground.	N/A
J9	GND	GND	Ground.	N/A
J10	GND	GND	Ground.	N/A
J11	GND	GND	Ground.	N/A
J12	GND	GND	Ground.	N/A
J13	GND	GND	Ground.	N/A
J14	GND	GND	Ground.	N/A
J15	GND	GND	Ground.	N/A
J16	GND	GND	Ground.	N/A
J17	DVDD	Power	Digital Power Supply (1.8 V).	N/A
J20	DDC1_SDA	HDMI Tx1	Float this pin.	Digital output
J21	GND	GND	Ground.	N/A
J22	TX1_1+	HDMI Tx1	Float this pin.	Digital output
J23	TX1_1–	HDMI Tx1	Float this pin.	Digital output
K1	VS	Digital video sync	Float this pin as it is disabled by default.	Digital input

Location	Mnemonic	Type	Description if Unused	Pin Type
K2	PCLK	Digital Video Sync	Float this pin as it is disabled by default.	Digital input
K3	DVDD_IO	Power	Digital Interface Supply (3.3 V).	N/A
K4	DVDD_IO	Power	Digital Interface Supply (3.3 V).	N/A
K7	GND	GND	Ground.	N/A
K8	GND	GND	Ground.	N/A
K9	GND	GND	Ground.	N/A
K10	GND	GND	Ground.	N/A
K11	GND	GND	Ground.	N/A
K12	GND	GND	Ground.	N/A
K13	GND	GND	Ground.	N/A
K14	GND	GND	Ground.	N/A
K15	GND	GND	Ground.	N/A
K16	GND	GND	Ground.	N/A
K17	GND	GND	Ground.	N/A
K20	DDC1_SCL	HDMI Tx1	Float this pin.	Digital output
K21	GND	GND	Ground.	N/A
K22	TX1_0+	HDMI Tx1	Float this pin.	Digital output
K23	TX1_0–	HDMI Tx1	Float this pin.	Digital output
L1	P[32]	Digital video input	Float this pin as it is disabled by default.	Bi-directional digital IO
L2	P[33]	Digital video input	Float this pin as it is disabled by default.	Bi-directional digital IO
L3	P[34]	Digital video input	Float this pin as it is disabled by default.	Bi-directional digital IO
L4	P[35]	Digital video input	Float this pin as it is disabled by default.	Bi-directional digital IO
L7	DVDD	Power	Digital Power Supply (1.8 V).	N/A
L8	GND	GND	Ground.	N/A
L9	GND	GND	Ground.	N/A
L10	GND	GND	Ground.	N/A
L11	GND	GND	Ground.	N/A
L12	GND	GND	Ground.	N/A
L13	GND	GND	Ground.	N/A
L14	GND	GND	Ground.	N/A
L15	GND	GND	Ground.	N/A
L16	GND	GND	Ground.	N/A
L17	GND	GND	Ground.	N/A
L20	HPD_TX1	HDMI Tx1	Float this pin.	Analog input (5V Tol)
L21	GND	GND	Ground.	N/A
L22	TX1_C+	HDMI Tx1	Float this pin.	Digital output
L23	TX1_C–	HDMI Tx1	Float this pin.	Digital output
M1	P[28]	Digital video input	Float this pin as it is disabled by default.	Bi-directional digital IO
M2	P[29]	Digital video input	Float this pin as it is disabled by default.	Bi-directional digital IO
M3	P[30]	Digital video input	Float this pin as it is disabled by default.	Bi-directional digital IO
M4	P[31]	Digital video input	Float this pin as it is disabled by default.	Bi-directional digital IO
M7	GND	GND	Ground.	N/A

Location	Mnemonic	Type	Description if Unused	Pin Type
M8	GND	GND	Ground.	N/A
M9	GND	GND	Ground.	N/A
M10	GND	GND	Ground.	N/A
M11	GND	GND	Ground.	N/A
M12	GND	GND	Ground.	N/A
M13	GND	GND	Ground.	N/A
M14	GND	GND	Ground.	N/A
M15	GND	GND	Ground.	N/A
M16	GND	GND	Ground.	N/A
M17	GND	GND	Ground.	N/A
M20	R_TX1	HDMI Tx11	Float this pin.	Digital output
M21	PVDD5	Power	HDMI Tx PLL Power Supply (1.8 V). This pin is a voltage regulator output. Connect a decoupling capacitor between this pin and ground.	N/A
M22	HEAC_1+	HDMI Tx1	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
M23	HEAC_1−	HDMI Tx1	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
N1	P[24]	Digital video input	Float this pin as it is disabled by default.	Bi-directional digital IO
N2	P[25]	Digital video input	Float this pin as it is disabled by default.	Bi-directional digital IO
N3	P[26]	Digital video input	Float this pin as it is disabled by default.	Bi-directional digital IO
N4	P[27]	Digital video input	Float this pin as it is disabled by default.	Bi-directional digital IO
N7	GND	GND	Ground.	N/A
N8	GND	GND	Ground.	N/A
N9	GND	GND	Ground.	N/A
N10	GND	GND	Ground.	N/A
N11	GND	GND	Ground.	N/A
N12	GND	GND	Ground.	N/A
N13	GND	GND	Ground.	N/A
N14	GND	GND	Ground.	N/A
N15	GND	GND	Ground.	N/A
N16	GND	GND	Ground.	N/A
N17	GND	GND	Ground.	N/A
N20	DNC	Do Not Connect	Float this pin.	Digital output
N21	PVDD5	Power	HDMI Tx PLL Power Supply (1.8 V). This pin is a voltage regulator output. Connect a decoupling capacitor between this pin and ground.	N/A
N22	AVDD3	Power	HDMI Analog Power Supply (1.8 V).	N/A
N23	NC	No connect	Connect this pin to ground.	Digital input
P1	P[20]	Digital video input	Float this pin as it is disabled by default.	Digital input
P2	P[21]	Digital video input	Float this pin as it is disabled by default.	Digital input
P3	P[22]	Digital video input	Float this pin as it is disabled by default.	Digital input

Location	Mnemonic	Type	Description if Unused	Pin Type
P4	P[23]	Digital video input	Float this pin as it is disabled by default.	Digital input
P7	DVDD	Power	Digital Power Supply (1.8 V).	N/A
P8	GND	GND	Ground.	N/A
P9	GND	GND	Ground.	N/A
P10	GND	GND	Ground.	N/A
P11	GND	GND	Ground.	N/A
P12	GND	GND	Ground.	N/A
P13	GND	GND	Ground.	N/A
P14	GND	GND	Ground.	N/A
P15	GND	GND	Ground.	N/A
P16	GND	GND	Ground.	N/A
P17	DVDD	Power	Digital Power Supply (1.8 V).	N/A
P20	DDC2_SCL	HDMI Tx2	Float this pin.	Digital output
P21	GND	GND	Ground.	N/A
P22	TX2_2+	HDMI Tx2	Float this pin.	Digital output
P23	TX2_2−	HDMI Tx2	Float this pin.	Digital output
R1	P[16]	Digital video input	Float this pin as it is disabled by default.	Digital input
R2	P[17]	Digital video input	Float this pin as it is disabled by default.	Digital input
R3	P[18]	Digital video input	Float this pin as it is disabled by default.	Digital input
R4	P[19]	Digital video input	Float this pin as it is disabled by default.	Digital input
R7	GND	GND	Ground.	N/A
R8	GND	GND	Ground.	N/A
R9	GND	GND	Ground.	N/A
R10	GND	GND	Ground.	N/A
R11	GND	GND	Ground.	N/A
R12	GND	GND	Ground.	N/A
R13	GND	GND	Ground.	N/A
R14	GND	GND	Ground.	N/A
R15	GND	GND	Ground.	N/A
R16	GND	GND	Ground.	N/A
R17	GND	GND	Ground.	N/A
R20	DDC2_SDA	HDMI Tx2	Float this pin.	Digital output
R21	GND	GND	Ground.	N/A
R22	TX2_1+	HDMI Tx2	Float this pin.	Digital output
R23	TX2_1−	HDMI Tx2	Float this pin.	Digital output
T1	P[14]	Digital video input	Float this pin as it is disabled by default.	Digital input
T2	P[15]	Digital video input	Float this pin as it is disabled by default.	Digital input
T3	GND	GND	Ground.	N/A
T4	GND	GND	Ground.	N/A
T7	GND	GND	Ground.	N/A
T8	GND	GND	Ground.	N/A
T9	GND	GND	Ground.	N/A
T10	GND	GND	Ground.	N/A
T11	GND	GND	Ground.	N/A

Location	Mnemonic	Type	Description if Unused	Pin Type
T12	GND	GND	Ground.	N/A
T13	GND	GND	Ground.	N/A
T14	GND	GND	Ground.	N/A
T15	GND	GND	Ground.	N/A
T16	GND	GND	Ground.	N/A
T17	GND	GND	Ground.	N/A
T20	HPD_TX2	HDMI Tx2	Float this pin.	Analog input
T21	GND	GND	Ground.	N/A
T22	TX2_0+	HDMI Tx2	Float this pin.	Digital output
T23	TX2_0–	HDMI Tx2	Float this pin.	Digital output
U1	P[10]	Digital video input	Float this pin as it is disabled by default.	Digital input
U2	P[11]	Digital video input	Float this pin as it is disabled by default.	Digital input
U3	P[12]	Digital video input	Float this pin as it is disabled by default.	Digital input
U4	P[13]	Digital video input	Float this pin as it is disabled by default.	Digital input
U7	GND	GND	Ground.	N/A
U8	GND	GND	Ground.	N/A
U9	DVDD	Power	Digital Power Supply (1.8 V).	N/A
U10	GND	GND	Ground.	N/A
U11	GND	GND	Ground.	N/A
U12	DVDD	Power	Digital Power Supply (1.8 V).	N/A
U13	GND	GND	Ground.	N/A
U14	GND	GND	Ground.	N/A
U15	DVDD	Power	Digital Power Supply (1.8 V).	N/A
U16	GND	GND	Ground.	N/A
U17	GND	GND	Ground.	N/A
U20	R_TX2	HDMI Tx2	Float this pin.	Digital output
U21	GND	GND	Ground.	N/A
U22	TX2_C+	HDMI Tx2	Float this pin.	Digital output
U23	TX2_C–	HDMI Tx2	Float this pin.	Digital output
V1	P[6]	Digital video input	Float this pin as it is disabled by default.	Digital input
V2	P[7]	Digital video input	Float this pin as it is disabled by default.	Digital input
V3	P[8]	Digital video input	Float this pin as it is disabled by default.	Digital input
V4	P[9]	Digital video input	Float this pin as it is disabled by default.	Digital input
V20	GND	GND	Ground.	N/A
V21	PVDD6	Power	HDMI Tx PLL Power Supply (1.8 V). This pin is a voltage regulator output. Connect a decoupling capacitor between this pin and ground.	N/A
V22	HEAC_2+	HDMI Tx2	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
V23	HEAC_2–	HDMI Tx2	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
W1	P[2]	Digital video input	Float this pin as it is disabled by default.	Digital input

Location	Mnemonic	Type	Description if Unused	Pin Type
W2	P[3]	Digital video input	Float this pin as it is disabled by default.	Digital input
W3	P[4]	Digital video input	Float this pin as it is disabled by default.	Digital input
W4	P[5]	Digital video input	Float this pin as it is disabled by default.	Digital input
W20	TEST3	Miscellaneous digital	Float this pin.	Digital output
W21	PVDD6	Power	HDMI Tx PLL Power Supply (1.8 V). This pin is a voltage regulator output. Connect a decoupling capacitor between this pin and ground.	N/A
W22	AVDD3	Power	HDMI Analog Power Supply (1.8 V).	N/A
W23	NC	No connect	Float this pin.	Digital output
Y1	P[0]	Digital video input	Float this pin as it is disabled by default.	Digital input
Y2	P[1]	Digital video input	Float this pin as it is disabled by default.	Digital input
Y3	DDR_DQS[2]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
Y4	GND	GND	Ground.	N/A
Y5	DDR_DQ[23]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
Y6	DVDD_DDR	Power	DDR Interface Supply (1.8 V).	N/A
Y7	DDR_DQS[3]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
Y8	GND	GND	Ground.	N/A
Y9	DDR_A[11]	DDR interface	Float this pin.	Digital output
Y10	DVDD_DDR	Power	DDR Interface Supply (1.8 V).	N/A
Y11	DDR_A[4]	DDR interface	Float this pin.	Digital output
Y12	GND	GND	Ground.	N/A
Y13	DDR_$\overline{\text{CAS}}$	DDR interface	Float this pin.	Digital output
Y14	DVDD_DDR	Power	DDR Interface Supply (1.8 V).	N/A
Y15	DDR_$\overline{\text{CK}}$	DDR interface	Float this pin.	Digital output
Y16	GND	GND	Ground.	N/A
Y17	DDR_DQ[9]	DDR Interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
Y18	DVDD_DDR	Power	DDR Interface Supply (1.8 V).	N/A
Y19	DDR_DQ[14]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
Y20	GND	GND	Ground.	N/A
Y21	DDR_DQ[6]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
Y22	PVDD_DDR	Power	DDR Interface PLL Supply (1.8 V).	N/A
Y23	GND	GND	Ground.	N/A
AA1	DDR_DQ[18]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AA2	GND	GND	Ground.	N/A
AA3	GND	GND	Ground.	N/A
AA4	DDR_$\overline{\text{DQS[2]}}$	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AA5	DDR_DQ[26]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AA6	DVDD_DDR	Power	DDR Interface Supply (1.8 V).	N/A

Location	Mnemonic	Type	Description if Unused	Pin Type
AA7	DDR_$\overline{\text{DQS}}$[3]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AA8	NC/GND	No connect/GND	For New ADV8005 Designs, Float this pin. For Designs That Must Maintain Consistency with ADV8005, this Pin can be Grounded.	N/A
AA9	DDR_A[8]	DDR interface	Float this pin.	Digital output
AA10	DVDD_DDR	Power	DDR Interface Supply (1.8 V).	N/A
AA11	DDR_A[2]	DDR interface	Float this pin.	Digital output
AA12	GND	GND	Ground.	N/A
AA13	DDR_$\overline{\text{CS}}$	DDR interface	Float this pin.	Digital output
AA14	DVDD_DDR	Power	DDR Interface Supply (1.8 V).	
AA15	DDR_CK	DDR interface	Float this pin.	Digital output
AA16	GND	GND	Ground.	N/A
AA17	DDR_DQ[11]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AA18	DVDD_DDR	Power	DDR Interface Supply (1.8 V).	N/A
AA19	DDR_DM[1]	DDR interface	Float this pin.	Digital output
AA20	DDR_DM[0]	DDR interface	Float this pin.	Digital output
AA21	GND	GND	Ground.	N/A
AA22	GND	GND	Ground.	N/A
AA23	DDR_DQ[3]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AB1	DDR_DQ[21]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AB2	DDR_DQ[19]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AB3	DDR_DQ[17]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AB4	DDR_DM[2]	DDR interface	Float this pin.	Digital output
AB5	DDR_DQ[30]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AB6	DDR_DM[3]	DDR interface	Float this pin.	Digital output
AB7	DDR_DQ[31]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AB8	DDR_DQ[29]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AB9	DDR_A[12]	DDR interface	Float this pin.	Digital output
AB10	DDR_A[6]	DDR interface	Float this pin.	Digital output
AB11	DDR_A[3]	DDR interface	Float this pin.	Digital output
AB12	DDR_A[0]	DDR interface	Float this pin.	Digital output
AB13	DDR_BA[0]	DDR interface	Float this pin.	Digital output
AB14	DDR_$\overline{\text{RAS}}$	DDR interface	Float this pin.	Digital output
AB15	DDR_CKE	DDR interface	Float this pin.	Digital output
AB16	DDR_DQ[12]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AB17	DDR_DQS[1]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AB18	DDR_DQ[8]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AB19	DDR_DQ[13]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO

Location	Mnemonic	Type	Description if Unused	Pin Type
AB20	DDR_DQ[0]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AB21	DDR_DQ[5]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AB22	DDR_DQS[0]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AB23	DDR_DQ[4]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AC1	DDR_DQ[16]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AC2	DDR_DQ[20]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AC3	DDR_DQ[22]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AC4	DDR_DQ[25]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AC5	DDR_DQ[28]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AC6	DDR_DQ[27]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AC7	DDR_DQ[24]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AC8	DDR_A[9]	DDR interface	Float this pin.	Digital output
AC9	DDR_A[5]	DDR interface	Float this pin.	Digital output
AC10	DDR_A[7]	DDR interface	Float this pin.	Digital output
AC11	DDR_A[1]	DDR interface	Float this pin.	Digital output
AC12	DDR_A[10]	DDR interface	Float this pin.	Digital output
AC13	DDR_BA[1]	DDR interface	Float this pin.	Digital output
AC14	DDR_BA[2]	DDR interface	Float this pin.	Digital output
AC15	DDR_$\overline{\text{WE}}$	DDR interface	Float this pin.	Digital output
AC16	DDR_VREF	DDR interface	Connect to DVDD_DDR.	Digital input
AC17	DDR_DQ[10]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AC18	DDR_$\overline{\text{DQS[1]}}$	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AC19	DDR_DQ[15]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AC20	DDR_DQ[7]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AC21	DDR_DQ[2]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AC22	DDR_$\overline{\text{DQS[0]}}$	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO
AC23	DDR_DQ[1]	DDR interface	Connect this pin to ground through a 4.7kΩ resistor.	Bi-directional digital IO

[1] Sensitive node. Careful layout is important. The associated circuitry should be kept as close as possible to the ADV8005.

APPENDIX C

PIXEL INPUT AND OUTPUT FORMATS

Table 88: RGB Input Formats

		ADV8005 PIN NAME	8-BIT SDR 4:2:2	10-BIT SDR 4:2:2	12-BIT SDR 4:2:2	16-BIT SDR 4:2:2	20-BIT SDR 4:2:2	24-BIT SDR 4:2:2	24-BIT SDR 4:4:4	30-BIT SDR 4:4:4	36-BIT SDR 4:4:4	8-BIT DDR 4:2:2		10-BIT DDR 4:2:2		12-BIT DDR 4:2:2		24-BIT SDR 4:4:4 (a)	8-BIT x2 SDR 4:4:4	8-BIT x2 SDR 4:2:2	10-BIT x2 SDR 4:2:2	12-BIT x2 SDR 4:2:2	30-BIT SDR 4:4:4	21-BIT SDR 4:4:4	30-BIT SDR 4:4:4
												Clock Rise	Clock Fall	Clock Rise	Clock Fall	Clock Rise	Clock Fall								
			RGB Colourspace																						
59	Sub TTL Input	OSD_IN.23	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	R7	Z	Z	Z	Z	Z	G6	R9
58		OSD_IN.22	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	R6	Z	Z	Z	Z	Z	G5	R8
57		OSD_IN.21	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	R5	Z	Z	Z	Z	Z	G4	R7
56		OSD_IN.20	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	R4	Z	Z	Z	Z	Z	G3	R6
55		OSD_IN.19	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	R3	Z	Z	Z	Z	Z	G2	R5
54		OSD_IN.18	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	R2	Z	Z	Z	Z	Z	G1	R4
53		OSD_IN.17	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	R1	Z	Z	Z	Z	Z	G0	R3
52		OSD_IN.16	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	R0	Z	Z	Z	Z	Z	Z	R2
51		OSD_IN.15	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	G7	Z	Z	Z	Z	Z	R6	R1
50		OSD_IN.14	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	G6	Z	Z	Z	Z	Z	R5	R0
49		OSD_IN.13	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	G5	Z	Z	Z	Z	Z	R4	G9
48		OSD_IN.12	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	G4	Z	Z	Z	Z	Z	R3	G8
47		OSD_IN.11	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	G3	G1.7	Z	Z	Z	Z	R2	G7
46		OSD_IN.10	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	G2	G1.6	Z	Z	Z	Z	R1	G6
45		OSD_IN.9	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	G1	G1.5	Z	Z	Z	Z	R0	G5
44		OSD_IN.8	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	G0	G1.4	Z	Z	Z	Z	Z	G4
43		OSD_IN.7	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	B7	G1.3	Z	Z	Z	Z	B6	G3
42		OSD_IN.6	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	B6	G1.2	Z	Z	Z	Z	B5	G2
41		OSD_IN.5	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	B5	G1.1	Z	Z	Z	Z	B4	G1

40		OSD_IN.4	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	B4	G1.0	Z	Z	Z	Z	B3	G0
39		OSD_IN.3	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	B3	B1.7	Z	Z	Z	Z	B2	B9
38		OSD_IN.2	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	B2	B1.6	Z	Z	Z	Z	B1	B8
37		OSD_IN.1	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	B1	B1.5	Z	Z	Z	Z	B0	B7
36		OSD_IN.0	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	B0	B1.4	Z	Z	Z	Z	Z	B6
35		P.35	Z	Z	Z	Z	Z	Z	R7	R9	R11	Z	Z	Z	Z	Z	Z	Z	B1.3	Z	Z	Z	Z	Z	B5
34		P.34	Z	Z	Z	Z	Z	Z	R6	R8	R10	Z	Z	Z	Z	Z	Z	Z	B1.2	Z	Z	Z	Z	Z	B4
33		P.33	Z	Z	Z	Z	Z	Z	R5	R7	R9	Z	Z	Z	Z	Z	Z	Z	B1.1	Z	Z	Z	Z	Z	B3
32		P.32	Z	Z	Z	Z	Z	Z	R4	R6	R8	Z	Z	Z	Z	Z	Z	Z	B1.0	Z	Z	Z	Z	Z	B2
31		P.31	Z	Z	Z	Z	Z	Z	R3	R5	R7	Z	Z	Z	Z	Z	Z	Z	R1.7	Z	Z	Z	Z	Z	B1
30		P.30	Z	Z	Z	Z	Z	Z	R2	R4	R6	Z	Z	Z	Z	Z	Z	Z	R1.6	Z	Z	Z	Z	Z	B0
29		P.29	Z	Z	Z	Z	Z	Z	R1	R3	R5	Z	Z	Z	Z	Z	Z	Z	R1.5	Z	Z	Z	R9	Z	Z
28		P.28	Z	Z	Z	Z	Z	Z	R0	R2	R4	Z	Z	Z	Z	Z	Z	Z	R1.4	Z	Z	Z	R8	Z	Z
27		P.27	Z	Z	Z	Z	Z	Z	Z	R1	R3	Z	Z	Z	Z	Z	Z	Z	R1.3	Z	Z	Z	R7	Z	Z
26	Main TTL Input	P.26	Z	Z	Z	Z	Z	Z	Z	R0	R2	Z	Z	Z	Z	Z	Z	Z	R1.2	Z	Z	Z	R6	Z	Z
25		P.25	Z	Z	Z	Z	Z	Z	Z	Z	R1	Z	Z	Z	Z	Z	Z	Z	R1.1	Z	Z	Z	R5	Z	Z
24		P.24	Z	Z	Z	Z	Z	Z	Z	Z	R0	Z	Z	Z	Z	Z	Z	Z	R1.0	Z	Z	Z	R4	Z	Z
23		P.23	Z	Z	Z	Z	Z	Z	G7	G9	G11	Z	Z	Z	Z	Z	Z	R7	G2.7	Z	Z	Z	R3	R6	Z
22		P.22	Z	Z	Z	Z	Z	Z	G6	G8	G10	Z	Z	Z	Z	Z	Z	R6	G2.6	Z	Z	Z	R2	R5	Z
21		P.21	Z	Z	Z	Z	Z	Z	G5	G7	G9	Z	Z	Z	Z	Z	Z	R5	G2.5	Z	Z	Z	R1	R4	Z
20		P.20	Z	Z	Z	Z	Z	Z	G4	G6	G8	Z	Z	Z	Z	Z	Z	R4	G2.4	Z	Z	Z	R0	R3	Z
19		P.19	Z	Z	Z	Z	Z	Z	G3	G5	G7	Z	Z	Z	Z	Z	Z	R3	G2.3	Z	Z	Z	G9	R2	Z
18		P.18	Z	Z	Z	Z	Z	Z	G2	G4	G6	Z	Z	Z	Z	Z	Z	R2	G2.2	Z	Z	Z	G8	R1	Z
17		P.17	Z	Z	Z	Z	Z	Z	G1	G3	G5	Z	Z	Z	Z	Z	Z	R1	G2.1	Z	Z	Z	G7	R0	Z
16		P.16	Z	Z	Z	Z	Z	Z	G0	G2	G4	Z	Z	Z	Z	Z	Z	R0	G2.0	Z	Z	Z	G6	Z	Z

15		P.15	Z	Z	Z	Z	Z	Z	Z	G1	G3	Z	Z	Z	Z	Z	Z	G7	B2.7	Z	Z	Z	G5	G6	Z
14		P.14	Z	Z	Z	Z	Z	Z	Z	G0	G2	Z	Z	Z	Z	Z	Z	G6	B2.6	Z	Z	Z	G4	G5	Z
13		P.13	Z	Z	Z	Z	Z	Z	Z	Z	G1	Z	Z	Z	Z	Z	Z	G5	B2.5	Z	Z	Z	G3	G4	Z
12		P.12	Z	Z	Z	Z	Z	Z	Z	Z	G0	Z	Z	Z	Z	Z	Z	G4	B2.4	Z	Z	Z	G2	G3	Z
11		P.11	Z	Z	Z	Z	Z	Z	B7	B9	B11	Z	Z	Z	Z	Z	Z	G3	B2.3	Z	Z	Z	G1	G2	Z
10		P.10	Z	Z	Z	Z	Z	Z	B6	B8	B10	Z	Z	Z	Z	Z	Z	G2	B2.2	Z	Z	Z	G0	G1	Z
9		P.9	Z	Z	Z	Z	Z	Z	B5	B7	B9	Z	Z	Z	Z	Z	Z	G1	B2.1	Z	Z	Z	B9	G0	Z
8		P.8	Z	Z	Z	Z	Z	Z	B4	B6	B8	Z	Z	Z	Z	Z	Z	G0	B2.0	Z	Z	Z	B8	Z	Z
7		P.7	Z	Z	Z	Z	Z	Z	B3	B5	B7	Z	Z	Z	Z	Z	Z	B7	R2.7	Z	Z	Z	B7	B6	Z
6		P.6	Z	Z	Z	Z	Z	Z	B2	B4	B6	Z	Z	Z	Z	Z	Z	B6	R2.6	Z	Z	Z	B6	B5	Z
5		P.5	Z	Z	Z	Z	Z	Z	B1	B3	B5	Z	Z	Z	Z	Z	Z	B5	R2.5	Z	Z	Z	B5	B4	Z
4		P.4	Z	Z	Z	Z	Z	Z	B0	B2	B4	Z	Z	Z	Z	Z	Z	B4	R2.4	Z	Z	Z	B4	B3	Z
3		P.3	Z	Z	Z	Z	Z	Z	Z	B1	B3	Z	Z	Z	Z	Z	Z	B3	R2.3	Z	Z	Z	B3	B2	Z
2		P.2	Z	Z	Z	Z	Z	Z	Z	B0	B2	Z	Z	Z	Z	Z	Z	B2	R2.2	Z	Z	Z	B2	B1	Z
1		P.1	Z	Z	Z	Z	Z	Z	Z	Z	B1	Z	Z	Z	Z	Z	Z	B1	R2.1	Z	Z	Z	B1	B0	Z
0		P.0	Z	Z	Z	Z	Z	Z	Z	Z	B0	Z	Z	Z	Z	Z	Z	B0	R2.0	Z	Z	Z	B0	Z	Z

Table 89: YCbCr Input Formats

		ADV 8005 PIN NAME	8-BIT SDR 4:2:2	10-BIT SDR 4:2:2	12-BIT SDR 4:2:2	16-BIT SDR 4:2:2	20-BIT SDR 4:2:2	24-BIT SDR 4:2:2	24-BIT SDR 4:4:4	30-BIT SDR 4:4:4	36-BIT SDR 4:4:4	8-BIT DDR 4:2:2		10-BIT DDR 4:2:2		12-BIT DDR 4:2:2		24-BIT SDR 4:4:4 (a)	8-BIT x2 SDR 4:4:4	8-BIT x2 SDR 4:2:2	10-BIT x2 SDR 4:2:2	12-BIT x2 SDR 4:2:2	30-BIT SDR 4:4:4	21-BIT SDR 4:4:4	30-BIT SDR 4:4:4
												Clock Rise	Clock Fall	Clock Rise	Clock Fall	Clock Rise	Clock Fall								
			YCbCr Colourspace																						
	SubTTL Input	OSD_DE	OSD_DE	OSD_DE	OSD_DE	OSD_DE	OSD_DE	OSD_DE	Z	Z	Z	OSD_DE	OSD_DE	OSD_DE	OSD_DE	OSD_DE	OSD_DE	OSD_DE	Z	Z	Z	Z	Z	OSD_DE	OSD_DE
		OSD_VS	OSD_VS	OSD_VS	OSD_VS	OSD_VS	OSD_VS	OSD_VS	Z	Z	Z	OSD_VS	OSD_VS	OSD_VS	OSD_VS	OSD_VS	OSD_VS	OSD_VS	Z	Z	Z	Z	Z	OSD_VS	OSD_VS
		OSD_CLK	OSD_CLK	OSD_CLK	OSD_CLK	OSD_CLK	OSD_CLK	OSD_CLK	Z	Z	Z	OSD_CLK	OSD_CLK	OSD_CLK	OSD_CLK	OSD_CLK	OSD_CLK	OSD_CLK	Z	Z	Z	Z	Z	OSD_CLK	OSD_CLK
59		OSD_IN.23	Cb7/Cr7, Y7	Cb9/Cr9, Y9	Cb11/Cr11,Y11	Y7	Y9	Y11	Z	Z	Z	Y7	Cb7, Cr7	Y9	Cb9, Cr9	Y11	Cb11,Cr11	Cr7	Z	Z	Z	Z	Z	Cr6	Cr9
58		OSD_IN.22	Cb6/Cr6, Y6	Cb8/Cr8, Y8	Cb10/Cr10,Y10	Y6	Y8	Y10	Z	Z	Z	Y6	Cb6, Cr6	Y8	Cb8, Cr8	Y10	Cb10,Cr10	Cr6	Z	Z	Z	Z	Z	Cr5	Cr8
57		OSD_IN.21	Cb5/Cr5, Y5	Cb7/Cr7, Y7	Cb9/Cr9,Y9	Y5	Y7	Y9	Z	Z	Z	Y5	Cb5, Cr5	Y7	Cb7, Cr7	Y9	Cb9, Cr9	Cr5	Z	Z	Z	Z	Z	Cr4	Cr7
56		OSD_IN.20	Cb4/Cr4, Y4	Cb6/Cr6, Y6	Cb8/Cr8,Y8	Y4	Y6	Y8	Z	Z	Z	Y4	Cb4, Cr4	Y6	Cb6, Cr6	Y8	Cb8, Cr8	Cr4	Z	Z	Z	Z	Z	Cr3	Cr6
55		OSD_IN.19	Cb3/Cr3, Y3	Cb5/Cr5, Y5	Cb7/Cr7,Y7	Y3	Y5	Y7	Z	Z	Z	Y3	Cb3, Cr3	Y5	Cb5, Cr5	Y7	Cb7, Cr7	Cr3	Z	Z	Z	Z	Z	Cr2	Cr5
54		OSD_IN.18	Cb2/Cr2, Y2	Cb4/Cr4, Y4	Cb6/Cr6,Y6	Y2	Y4	Y6	Z	Z	Z	Y2	Cb2, Cr2	Y4	Cb4, Cr4	Y6	Cb6, Cr6	Cr2	Z	Z	Z	Z	Z	Cr1	Cr4
53		OSD_IN.17	Cb1/Cr1, Y1	Cb3/Cr3, Y3	Cb5/Cr5,Y5	Y1	Y3	Y5	Z	Z	Z	Y1	Cb1, Cr1	Y3	Cb3, Cr3	Y5	Cb5, Cr5	Cr1	Z	Z	Z	Z	Z	Cr0	Cr3
52		OSD_IN.16	Cb0/Cr0, Y0	Cb2/Cr2, Y2	Cb4/Cr4,Y4	Y0	Y2	Y4	Z	Z	Z	Y0	Cb0, Cr0	Y2	Cb2, Cr2	Y4	Cb4, Cr4	Cr0	Z	Z	Z	Z	Z	Z	Cr2
51		OSD_IN.15	Z	Cb1/Cr1, Y1	Cb3/Cr3,Y3	Z	Y1	Y3	Z	Z	Z	Z	Z	Y1	Cb1, Cr1	Y3	Cb3, Cr3	Y7	Z	Z	Z	Z	Z	Y6	Cr1
50		OSD_IN.14	Z	Cb0/Cr0, Y0	Cb2/Cr2,Y2	Z	Y0	Y2	Z	Z	Z	Z	Z	Y0	Cb0, Cr0	Y2	Cb2, Cr2	Y6	Z	Z	Z	Z	Z	Y5	Cr0

49		OSD_IN.13	Z	Z	Cb1/Cr1,Y1	Z	Z	Y1	Z	Z	Z	Z	Z	Z	Z	Y1	Cb1, Cr1	Y5	Z	Z	Z	Z	Z	Y4	Y9
48		OSD_IN.12	Z	Z	Cb0/Cr0,Y0	Z	Z	Y0	Z	Z	Z	Z	Z	Z	Z	Y0	Cb0, Cr0	Y4	Z	Z	Z	Z	Z	Y3	Y8
47		OSD_IN.11	Z	Z	Z	Cb7, Cr7	Cb9, Cr9	Cb11,Cr11	Z	Z	Z	Z	Z	Z	Z	Z	Z	Y3	Y1.7	Y1.7	Y1.9	Y1.11	Z	Y2	Y7
46		OSD_IN.10	Z	Z	Z	Cb6, Cr6	Cb8, Cr8	Cb10,Cr10	Z	Z	Z	Z	Z	Z	Z	Z	Z	Y2	Y1.6	Y1.6	Y1.8	Y1.10	Z	Y1	Y6
45		OSD_IN.9	Z	Z	Z	Cb5, Cr5	Cb7, Cr7	Cb9, Cr9	Z	Z	Z	Z	Z	Z	Z	Z	Z	Y1	Y1.5	Y1.5	Y1.7	Y1.9	Z	Y0	Y5
44		OSD_IN.8	Z	Z	Z	Cb4, Cr4	Cb6, Cr6	Cb8, Cr8	Z	Z	Z	Z	Z	Z	Z	Z	Z	Y0	Y1.4	Y1.4	Y1.6	Y1.8	Z	Z	Y4
43		OSD_IN.7	Z	Z	Z	Cb3, Cr3	Cb5, Cr5	Cb7, Cr7	Z	Z	Z	Z	Z	Z	Z	Z	Z	Cb7	Y1.3	Y1.3	Y1.5	Y1.7	Z	Cb6	Y3
42		OSD_IN.6	Z	Z	Z	Cb2, Cr2	Cb4, Cr4	Cb6, Cr6	Z	Z	Z	Z	Z	Z	Z	Z	Z	Cb6	Y1.2	Y1.2	Y1.4	Y1.6	Z	Cb5	Y2
41		OSD_IN.5	Z	Z	Z	Cb1, Cr1	Cb3, Cr3	Cb5, Cr5	Z	Z	Z	Z	Z	Z	Z	Z	Z	Cb5	Y1.1	Y1.1	Y1.3	Y1.5	Z	Cb4	Y1
40		OSD_IN.4	Z	Z	Z	Cb0, Cr0	Cb2, Cr2	Cb4, Cr4	Z	Z	Z	Z	Z	Z	Z	Z	Z	Cb4	Y1.0	Y1.0	Y1.2	Y1.4	Z	Cb3	Y0
39		OSD_IN.3	Z	Z	Z	Z	Cb1, Cr1	Cb3, Cr3	Z	Z	Z	Z	Z	Z	Z	Z	Z	Cb3	Cb1.7	Z	Y1.1	Y1.3	Z	Cb2	Cb9
38		OSD_IN.2	Z	Z	Z	Z	Cb0, Cr0	Cb2, Cr2	Z	Z	Z	Z	Z	Z	Z	Z	Z	Cb2	Cb1.6	Z	Y1.0	Y1.2	Z	Cb1	Cb8
37		OSD_IN.1	Z	Z	Z	Z	Z	Cb1, Cr1	Z	Z	Z	Z	Z	Z	Z	Z	Z	Cb1	Cb1.5	Z	Z	Y1.1	Z	Cb0	Cb7
36		OSD_IN.0	Z	Z	Z	Z	Z	Cb0, Cr0	Z	Z	Z	Z	Z	Z	Z	Z	Z	Cb0	Cb1.4	Z	Z	Y1.0	Z	Z	Cb6
35	Main TTL Input	P.35	Z	Z	Z	Z	Z	Z	Cr7	Cr9	Cr11	Z	Z	Z	Z	Z	Z	Z	Cb1.3	Cb.7	Cb.9	Cb.11	Z	Z	Cb5
34		P.34	Z	Z	Z	Z	Z	Z	Cr6	Cr8	Cr10	Z	Z	Z	Z	Z	Z	Z	Cb1.2	Cb.6	Cb.8	Cb.10	Z	Z	Cb4
33		P.33	Z	Z	Z	Z	Z	Z	Cr5	Cr7	Cr9	Z	Z	Z	Z	Z	Z	Z	Cb1.1	Cb.5	Cb.7	Cb.9	Z	Z	Cb3
32		P.32	Z	Z	Z	Z	Z	Z	Cr4	Cr6	Cr8	Z	Z	Z	Z	Z	Z	Z	Cb1.0	Cb.4	Cb.6	Cb.8	Z	Z	Cb2
31		P.31	Z	Z	Z	Z	Z	Z	Cr3	Cr5	Cr7	Z	Z	Z	Z	Z	Z	Z	Cr1.7	Cb.3	Cb.5	Cb.7	Z	Z	Cb1
30		P.30	Z	Z	Z	Z	Z	Z	Cr2	Cr4	Cr6	Z	Z	Z	Z	Z	Z	Z	Cr1.6	Cb.2	Cb.4	Cb.6	Z	Z	Cb0
29		P.29	Z	Z	Z	Z	Z	Z	Cr1	Cr3	Cr5	Z	Z	Z	Z	Z	Z	Z	Cr1.5	Cb.1	Cb.3	Cb.5	Cr9	Z	Z
28		P.28	Z	Z	Z	Z	Z	Z	Cr0	Cr2	Cr4	Z	Z	Z	Z	Z	Z	Z	Cr1.4	Cb.0	Cb.2	Cb.4	Cr8	Z	Z

27		P.27	Z	Z	Z	Z	Z	Z	Z	Cr1	Cr3	Z	Z	Z	Z	Z	Z	Z	Cr1.3	Z	Cb.1	Cb.3	Cr7	Z	Z
26		P.26	Z	Z	Z	Z	Z	Z	Z	Cr0	Cr2	Z	Z	Z	Z	Z	Z	Z	Cr1.2	Z	Cb.0	Cb.2	Cr6	Z	Z
25		P.25	Z	Z	Z	Z	Z	Z	Z	Z	Cr1	Z	Z	Z	Z	Z	Z	Z	Cr1.1	Z	Z	Cb.1	Cr5	Z	Z
24		P.24	Z	Z	Z	Z	Z	Z	Z	Z	Cr0	Z	Z	Z	Z	Z	Z	Z	Cr1.0	Z	Z	Cb.0	Cr4	Z	Z
23		P.23	Cb7/Cr7,Y7	Cb9/Cr9,Y9	Cb11/Cr11,Y11	Y7	Y9	Y11	Y7	Y9	Y11	Y7	Cb7,Cr7	Y9	Cb9,Cr9	Y11	Cb11,Cr11	Cr7	Y2.7	Y2.7	Y2.9	Y2.11	Cr3	Cr6	Z
22		P.22	Cb6/Cr6,Y6	Cb8/Cr8,Y8	Cb10/Cr10,Y10	Y6	Y8	Y10	Y6	Y8	Y10	Y6	Cb6,Cr6	Y8	Cb8,Cr8	Y10	Cb10,Cr10	Cr6	Y2.6	Y2.6	Y2.8	Y2.10	Cr2	Cr5	Z
21		P.21	Cb5/Cr5,Y5	Cb7/Cr7,Y7	Cb9/Cr9,Y9	Y5	Y7	Y9	Y5	Y7	Y9	Y5	Cb5,Cr5	Y7	Cb7,Cr7	Y9	Cb9,Cr9	Cr5	Y2.5	Y2.5	Y2.7	Y2.9	Cr1	Cr4	Z
20		P.20	Cb4/Cr4,Y4	Cb6/Cr6,Y6	Cb8/Cr8,Y8	Y4	Y6	Y8	Y4	Y6	Y8	Y4	Cb4,Cr4	Y6	Cb6,Cr6	Y8	Cb8,Cr8	Cr4	Y2.4	Y2.4	Y2.6	Y2.8	Cr0	Cr3	Z
19		P.19	Cb3/Cr3,Y3	Cb5/Cr5,Y5	Cb7/Cr7,Y7	Y3	Y5	Y7	Y3	Y5	Y7	Y3	Cb3,Cr3	Y5	Cb5,Cr5	Y7	Cb7,Cr7	Cr3	Y2.3	Y2.3	Y2.5	Y2.7	Y9	Cr2	Z
18		P.18	Cb2/Cr2,Y2	Cb4/Cr4,Y4	Cb6/Cr6,Y6	Y2	Y4	Y6	Y2	Y4	Y6	Y2	Cb2,Cr2	Y4	Cb4,Cr4	Y6	Cb6,Cr6	Cr2	Y2.2	Y2.2	Y2.4	Y2.6	Y8	Cr1	Z
17		P.17	Cb1/Cr1,Y1	Cb3/Cr3,Y3	Cb5/Cr5,Y5	Y1	Y3	Y5	Y1	Y3	Y5	Y1	Cb1,Cr1	Y3	Cb3,Cr3	Y5	Cb5,Cr5	Cr1	Y2.1	Y2.1	Y2.3	Y2.5	Y7	Cr0	Z
16		P.16	Cb0/Cr0,Y0	Cb2/Cr2,Y2	Cb4/Cr4,Y4	Y0	Y2	Y4	Y0	Y2	Y4	Y0	Cb0,Cr0	Y2	Cb2,Cr2	Y4	Cb4,Cr4	Cr0	Y2.0	Y2.0	Y2.2	Y2.4	Y6	Z	Z
15		P.15	Z	Cb1/Cr1,Y1	Cb3/Cr3,Y3	Z	Y1	Y3	Z	Y1	Y3	Z	Z	Y1	Cb1,Cr1	Y3	Cb3,Cr3	Y7	Cb2.7	Z	Y2.1	Y2.3	Y5	Y6	Z
14		P.14	Z	Cb0/Cr0,Y0	Cb2/Cr2,Y2	Z	Y0	Y2	Z	Y0	Y2	Z	Z	Y0	Cb0,Cr0	Y2	Cb2,Cr2	Y6	Cb2.6	Z	Y2.0	Y2.2	Y4	Y5	Z
13		P.13	Z	Z	Cb1/Cr1,Y1	Z	Z	Y1	Z	Z	Y1	Z	Z	Z	Z	Y1	Cb1,Cr1	Y5	Cb2.5	Z	Z	Y2.1	Y3	Y4	Z
12		P.12	Z	Z	Cb0/Cr0,Y0	Z	Z	Y0	Z	Z	Y0	Z	Z	Z	Z	Y0	Cb0,Cr0	Y4	Cb2.4	Z	Z	Y2.0	Y2	Y3	Z
11		P.11	Z	Z	Z	Cb7,Cr7	Cb9,Cr9	Cb11,Cr11	Cb7	Cb9	Cb11	Z	Z	Z	Z	Z	Z	Y3	Cb2.3	Cr.7	Cr.9	Cr.11	Y1	Y2	Z
10		P.10	Z	Z	Z	Cb6,Cr6	Cb8,Cr8	Cb10,Cr10	Cb6	Cb8	Cb10	Z	Z	Z	Z	Z	Z	Y2	Cb2.2	Cr.6	Cr.8	Cr.10	Y0	Y1	Z
9		P.9	Z	Z	Z	Cb5,Cr5	Cb7,Cr7	Cb9,Cr9	Cb5	Cb7	Cb9	Z	Z	Z	Z	Z	Z	Y1	Cb2.1	Cr.5	Cr.7	Cr.9	Cb9	Y0	Z

8		P.8	Z	Z	Z	Cb4, Cr4	Cb6, Cr6	Cb8, Cr8	Cb4	Cb6	Cb8	Z	Z	Z	Z	Z	Z	Y0	Cb2.0	Cr.4	Cr.6	Cr.8	Cb8	Z	Z
7		P.7	Z	Z	Z	Cb3, Cr3	Cb5, Cr5	Cb7, Cr7	Cb3	Cb5	Cb7	Z	Z	Z	Z	Z	Z	Cb7	Cr2.7	Cr.3	Cr.5	Cr.7	Cb7	Cb6	Z
6		P.6	Z	Z	Z	Cb2, Cr2	Cb4, Cr4	Cb6, Cr6	Cb2	Cb4	Cb6	Z	Z	Z	Z	Z	Z	Cb6	Cr2.6	Cr.2	Cr.4	Cr.6	Cb6	Cb5	Z
5		P.5	Z	Z	Z	Cb1, Cr1	Cb3, Cr3	Cb5, Cr5	Cb1	Cb3	Cb5	Z	Z	Z	Z	Z	Z	Cb5	Cr2.5	Cr.1	Cr.3	Cr.5	Cb5	Cb4	Z
4		P.4	Z	Z	Z	Cb0, Cr0	Cb2, Cr2	Cb4, Cr4	Cb0	Cb2	Cb4	Z	Z	Z	Z	Z	Z	Cb4	Cr2.4	Cr.0	Cr.2	Cr.4	Cb4	Cb3	Z
3		P.3	Z	Z	Z	Z	Cb1, Cr1	Cb3, Cr3	Z	Cb1	Cb3	Z	Z	Z	Z	Z	Z	Cb3	Cr2.3	Z	Cr.1	Cr.3	Cb3	Cb2	Z
2		P.2	Z	Z	Z	Z	Cb0, Cr0	Cb2, Cr2	Z	Cb0	Cb2	Z	Z	Z	Z	Z	Z	Cb2	Cr2.2	Z	Cr.0	Cr.2	Cb2	Cb1	Z
1		P.1	Z	Z	Z	Z	Z	Cb1, Cr1	Z	Z	Cb1	Z	Z	Z	Z	Z	Z	Cb1	Cr2.1	Z	Z	Cr.1	Cb1	Cb0	Z
0		P.0	Z	Z	Z	Z	Z	Cb0, Cr0	Z	Z	Cb0	Z	Z	Z	Z	Z	Z	Cb0	Cr2.0	Z	Z	Cr.0	Cb0	Z	Z
		VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	VID_DE	Z
		VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	VID_HS	Z
		VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	VID_VS	Z
		VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	VID_CLK	Z

Table 90: Alpha Blending Input Formats

Alpha Format	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE
OSD_DE	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
OSD_VS	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
OSD_CLK	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
OSD_IN.23	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A7	Z	Z
OSD_IN.22	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A6	Z	Z
OSD_IN.21	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A5	Z	Z
OSD_IN.20	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A4	Z	Z
OSD_IN.19	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A3	Z	Z
OSD_IN.18	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A2	Z	Z
OSD_IN.17	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A1	Z	Z
OSD_IN.16	Z	Z	Z	Z	Z	Z	Z	Z	Z	A7	Z	Z	A0	Z	Z
OSD_IN.15	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A7	Z	A7	Z
OSD_IN.14	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A6	Z	A6	Z
OSD_IN.13	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A7	A5	Z	A5	Z
OSD_IN.12	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A6	A4	Z	A4	Z
OSD_IN.11	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A3	Z
OSD_IN.10	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A2	Z
OSD_IN.9	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A1	Z
OSD_IN.8	Z	Z	Z	Z	Z	Z	Z	Z	Z	A6	Z	Z	Z	A0	Z
OSD_IN.7	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A7
OSD_IN.6	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A6
OSD_IN.5	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A5
OSD_IN.4	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A4
OSD_IN.3	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A3	Z	Z	A3
OSD_IN.2	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A2	Z	Z	A2
OSD_IN.1	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	A5	A1	Z	Z	A1
OSD_IN.0	Z	Z	Z	Z	Z	Z	Z	Z	Z	A5	A4	A0	Z	Z	A0
P.35	Z	Z	Z	Z	Z	Z	A7	Z	Z	Z	Z	Z	Z	Z	Z
P.34	Z	Z	Z	Z	Z	Z	A6	Z	Z	Z	Z	Z	Z	Z	Z
P.33	Z	Z	Z	Z	Z	Z	A5	Z	Z	Z	Z	Z	Z	Z	Z
P.32	Z	Z	Z	Z	Z	Z	A4	Z	Z	Z	Z	Z	Z	Z	Z
P.31	Z	Z	Z	Z	Z	Z	A3	Z	Z	Z	Z	Z	Z	Z	Z
P.30	Z	Z	Z	Z	Z	Z	A2	Z	Z	Z	Z	Z	Z	Z	Z

P.29	Z	Z	Z	Z	Z	Z	A1	Z	Z	Z	Z	Z	Z	Z	Z
P.28	Z	Z	Z	Z	Z	Z	A0	Z	Z	Z	Z	Z	Z	Z	Z
P.27	Z	A3	Z	A3	A7	A7	Z	Z	Z	Z	Z	Z	Z	Z	Z
P.26	Z	A2	Z	A2	A6	A6	Z	Z	Z	Z	Z	Z	Z	Z	Z
P.25	Z	A1	Z	A1	A5	A5	Z	Z	Z	Z	Z	Z	Z	Z	Z
P.24	Z	A0	Z	A0	A4	A4	Z	Z	Z	Z	Z	Z	Z	Z	Z
P.23	Z	Z	Z	Z	Z	Z	Z	A7	Z	Z	Z	Z	Z	Z	Z
P.22	Z	Z	Z	Z	Z	Z	Z	A6	Z	Z	Z	Z	Z	Z	Z
P.21	Z	Z	Z	Z	Z	Z	Z	A5	Z	Z	Z	Z	Z	Z	Z
P.20	Z	Z	Z	Z	Z	Z	Z	A4	Z	Z	Z	Z	Z	Z	Z
P.19	Z	Z	Z	Z	Z	Z	Z	A3	Z	Z	Z	Z	Z	Z	Z
P.18	Z	Z	Z	Z	Z	Z	Z	A2	Z	Z	Z	Z	Z	Z	Z
P.17	Z	Z	Z	Z	Z	Z	Z	A1	Z	Z	Z	Z	Z	Z	Z
P.16	Z	Z	Z	Z	Z	Z	Z	A0	Z	Z	Z	Z	Z	Z	Z
P.15	A3	Z	A7	A7	Z	A3	Z	Z	Z	Z	Z	Z	Z	Z	Z
P.14	A2	Z	A6	A6	Z	A2	Z	Z	Z	Z	Z	Z	Z	Z	Z
P.13	A1	Z	A5	A5	Z	A1	Z	Z	Z	Z	Z	Z	Z	Z	Z
P.12	A0	Z	A4	A4	Z	A0	Z	Z	Z	Z	Z	Z	Z	Z	Z
P.11	Z	Z	Z	Z	Z	Z	Z	Z	A7	Z	Z	Z	Z	Z	Z
P.10	Z	Z	Z	Z	Z	Z	Z	Z	A6	Z	Z	Z	Z	Z	Z
P.9	Z	Z	Z	Z	Z	Z	Z	Z	A5	Z	Z	Z	Z	Z	Z
P.8	Z	Z	Z	Z	Z	Z	Z	Z	A4	Z	Z	Z	Z	Z	Z
P.7	Z	Z	Z	Z	Z	Z	Z	Z	A3	Z	Z	Z	Z	Z	Z
P.6	Z	Z	Z	Z	Z	Z	Z	Z	A2	Z	Z	Z	Z	Z	Z
P.5	Z	Z	Z	Z	Z	Z	Z	Z	A1	Z	Z	Z	Z	Z	Z
P.4	Z	Z	Z	Z	Z	Z	Z	Z	A0	Z	Z	Z	Z	Z	Z
P.3	A7	A7	A3	Z	A3	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
P.2	A6	A6	A2	Z	A2	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
P.1	A5	A5	A1	Z	A1	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
P.0	A4	A4	A0	Z	A0	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
VID_DE	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
VID_HS	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
VID_VS	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
VID_CLK	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z

Table 91: RGB TTL Output Formats

ADV8005 PIN NAME		24-BIT SDR 4:4:4	30-BIT SDR 4:4:4	36-BIT SDR 4:4:4
OSD_DE		DE_OUT	DE_OUT	DE_OUT
OSD_VS		VS_OUT	VS_OUT	VS_OUT
OSD_HS		HS_OUT	HS_OUT	HS_OUT
OSD_CLK		CLK_OUT	CLK_OUT	CLK_OUT
OSD_IN.23		R7	R9	R11
OSD_IN.23		R6	R8	R10
OSD_IN.21		R5	R7	R9
OSD_IN.20		R4	R6	R8
OSD_IN.19		R3	R5	R7
OSD_IN.18		R2	R4	R6
OSD_IN.17		R1	R3	R5
OSD_IN.16		R0	R2	R4
OSD_IN.15		G7	R1	R3
OSD_IN.14		G6	R0	R2
OSD_IN.13		G5	G9	R1
OSD_IN.12		G4	G8	R0
OSD_IN.11		G3	G7	G11
OSD_IN.10		G2	G6	G10
OSD_IN.9		G1	G5	G9
OSD_IN.8		G0	G4	G8
OSD_IN.7		B7	G3	G7
OSD_IN.6		B6	G2	G6
OSD_IN.5		B5	G1	G5
OSD_IN.4		B4	G0	G4
OSD_IN.3		B3	B9	G3
OSD_IN.2		B2	B8	G2
OSD_IN.1		B1	B7	G1
OSD_IN.0		B0	B6	G0
P.35		Z	B5	B11
P.34		Z	B4	B10
P.33		Z	B3	B9
P.32		Z	B2	B8
P.31		Z	B1	B7
P.30		Z	B0	B6
P.29		Z	Z	B5
P.28		Z	Z	B4
P.27		Z	Z	B3
P.26		Z	Z	B2
P.25		Z	Z	B1
P.24		Z	Z	B0
P.23		Z	Z	Z
P.22		Z	Z	Z
P.21		Z	Z	Z
P.20		Z	Z	Z
P.19		Z	Z	Z
P.18		Z	Z	Z

P.17		Z	Z	Z
P.16		Z	Z	Z
P.15		Z	Z	Z
P.14		Z	Z	Z
P.13		Z	Z	Z
P.12		Z	Z	Z
P.11		Z	Z	Z
P.10		Z	Z	Z
P.9		Z	Z	Z
P.8		Z	Z	Z
P.7		Z	Z	Z
P.6		Z	Z	Z
P.5		Z	Z	Z
P.4		Z	Z	Z
P.3		Z	Z	Z
P.2		Z	Z	Z
P.1		Z	Z	Z
P.0		Z	Z	Z

Table 92: YCrCb TTL Output Formats

ADV8005 PIN NAME		16-BIT SDR 4:2:2	20-BIT SDR 4:2:2	24-BIT SDR 4:2:2	24-BIT SDR 4:4:4	30-BIT SDR 4:4:4	36-BIT SDR 4:4:4
		YCbCr Colorspace					
OSD_DE		DE_OUT	DE_OUT	DE_OUT	DE_OUT	DE_OUT	DE_OUT
OSD_VS		VS_OUT	VS_OUT	VS_OUT	VS_OUT	VS_OUT	VS_OUT
OSD_HS		HS_OUT	HS_OUT	HS_OUT	HS_OUT	HS_OUT	HS_OUT
OSD_CLK		CLK_OUT	CLK_OUT	CLK_OUT	CLK_OUT	CLK_OUT	CLK_OUT
OSD_IN.23		Y7	Y9	Y11	Cr7	Cr9	Cr11
OSD_IN.22		Y6	Y8	Y10	Cr6	Cr8	Cr10
OSD_IN.21		Y5	Y7	Y9	Cr5	Cr7	Cr9
OSD_IN.20		Y4	Y6	Y8	Cr4	Cr6	Cr8
OSD_IN.19		Y3	Y5	Y7	Cr3	Cr5	Cr7
OSD_IN.18		Y2	Y4	Y6	Cr2	Cr4	Cr6
OSD_IN.17		Y1	Y3	Y5	Cr1	Cr3	Cr5
OSD_IN.16		Y0	Y2	Y4	Cr0	Cr2	Cr4
OSD_IN.15		Z	Y1	Y3	Y7	Cr1	Cr3
OSD_IN.14		Z	Y0	Y2	Y6	Cr0	Cr2
OSD_IN.13		Z	Z	Y1	Y5	Y9	Cr1
OSD_IN.12		Z	Z	Y0	Y4	Y8	Cr0
OSD_IN.11		Cb7,Cr7	Cb9,Cr9	Cb11,Cr11	Y3	Y7	Y11
OSD_IN.10		Cb6,Cr6	Cb8,Cr8	Cb10,Cr10	Y2	Y6	Y10
OSD_IN.9		Cb5,Cr5	Cb7,Cr7	Cb9,Cr9	Y1	Y5	Y9
OSD_IN.8		Cb4,Cr4	Cb6,Cr6	Cb8,Cr8	Y0	Y4	Y8
OSD_IN.7		Cb3,Cr3	Cb5,Cr5	Cb7,Cr7	Cb7	Y3	Y7
OSD_IN.6		Cb2,Cr2	Cb4,Cr4	Cb6,Cr6	Cb6	Y2	Y6
OSD_IN.5		Cb1,Cr1	Cb3,Cr3	Cb5,Cr5	Cb5	Y1	Y5
OSD_IN.4		Cb0,Cr0	Cb2,Cr2	Cb4,Cr4	Cb4	Y0	Y4
OSD_IN.3		Z	Cb1,Cr1	Cb3,Cr3	Cb3	Cb9	Y3
OSD_IN.2		Z	Cb0,Cr0	Cb2,Cr2	Cb2	Cb8	Y2

OSD_IN.1		Z	Z	Cb1,Cr1	Cb1	Cb7	Y1
OSD_IN.0		Z	Z	Cb0,Cr0	Cb0	Cb6	Y0
P.35		Z	Z	Z	Z	Cb5	Cb11
P.34		Z	Z	Z	Z	Cb4	Cb10
P.33		Z	Z	Z	Z	Cb3	Cb9
P.32		Z	Z	Z	Z	Cb2	Cb8
P.31		Z	Z	Z	Z	Cb1	Cb7
P.30		Z	Z	Z	Z	Cb0	Cb6
P.29		Z	Z	Z	Z	Z	Cb5
P.28		Z	Z	Z	Z	Z	Cb4
P.27		Z	Z	Z	Z	Z	Cb3
P.26		Z	Z	Z	Z	Z	Cb2
P.25		Z	Z	Z	Z	Z	Cb1
P.24		Z	Z	Z	Z	Z	Cb0
P.23		Z	Z	Z	Z	Z	Z
P.22		Z	Z	Z	Z	Z	Z
P.21		Z	Z	Z	Z	Z	Z
P.20		Z	Z	Z	Z	Z	Z
P.19		Z	Z	Z	Z	Z	Z
P.18		Z	Z	Z	Z	Z	Z
P.17		Z	Z	Z	Z	Z	Z
P.16		Z	Z	Z	Z	Z	Z
P.15		Z	Z	Z	Z	Z	Z
P.14		Z	Z	Z	Z	Z	Z
P.13		Z	Z	Z	Z	Z	Z
P.12		Z	Z	Z	Z	Z	Z
P.11		Z	Z	Z	Z	Z	Z
P.10		Z	Z	Z	Z	Z	Z
P.9		Z	Z	Z	Z	Z	Z
P.8		Z	Z	Z	Z	Z	Z
P.7		Z	Z	Z	Z	Z	Z
P.6		Z	Z	Z	Z	Z	Z
P.5		Z	Z	Z	Z	Z	Z
P.4		Z	Z	Z	Z	Z	Z
P.3		Z	Z	Z	Z	Z	Z
P.2		Z	Z	Z	Z	Z	Z
P.1		Z	Z	Z	Z	Z	Z
P.0		Z	Z	Z	Z	Z	Z
VID_DE		Z	Z	Z	Z	Z	Z
VID_HS		Z	Z	Z	Z	Z	Z
VID_VS		Z	Z	Z	Z	Z	Z
VID_CLK		Z	Z	Z	Z	Z	Z

NOTES

I^{2}C refers to a communications protocol originally developed by Philips Semiconductors (now NXP Semiconductors).

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC in the United States and other countries.