ADV212 JPEG2000 Video Processor User’s Guide

Revision 1.1

December 11, 2006

Analog Devices, Inc.

One Technology Way

Norwood, Mass. 02062-9106

Copyright Information

Disclaimer

ABOUT THIS DOCUMENT

Purpose

This document describes the features of the ADV212 single-chip JPEG 2000 codec and provides the information necessary to integrate the ADV212 into a system.

Audience

This document is intended for hardware engineers working on systems that include the ADV212. This document is also intended for software engineers who write applications for these systems.

Related Documents

The following supporting documentation can be downloaded from http://www.analog.com under ADV212 Technical Documentation:

- ADV212 datasheet — This document contains the package specifications and all electrical and timing specifications.
- *ADV212 Programming Guide* — This is a programming guide containing configuration examples for all modes mentioned in this *User’s Guide*. This also includes information on loading the firmware.
- Firmware — Up-to-date firmware can be downloaded from http://www.analog.com under ADV212 Technical Documentation.
- Product Change Notices and Errata sheets.

Under ADV202 Technical Documentation:

- *HIPI Mode and Still-Image Applications* — This document contains configuration examples for host interface pixel interface (HIPI) mode. While this document was written for the ADV202, it also applies to the ADV212.
- *AN-799 Test Modes* — This document lists recommended test procedures to verify correct hardware configuration. While this document was written for the ADV202, it also applies to the ADV212.

Conventions

This document uses certain conventions to assist you in identifying, locating, and understanding information.

Numbering Systems

Hexadecimal values are represented with the prefix “0x” followed by the value. For example: 0xFFFF.

Typographic Notation

The following typographic notations are used throughout this document:

Example	Description
[n]	A number enclosed in brackets represents a single bit in a register or in memory.
[n:m]	Numbers enclosed in brackets and separated by a colon represent the endpoints of a continuous range of bits in a register or in memory.
ⓘ	A note provides supplementary information on a related topic.
🛇	A caution identifies conditions or inappropriate usage of the product that could lead to undesirable results or product damage.

Special Terms

The following terms have special meanings:

Term	Meaning
assert	Refers to the state of a signal as follows: • An active-high signal is asserted when high (1). • An active-low signal is asserted when low (0).
byte	An 8-bit data object
deassert	Refers to the state of a signal as follows: • An active-high signal is deasserted when low (0). • An active-low signal is deasserted when high (1).
half-word	A 16-bit data object
space	A FIFO slot available for a word entry
word	A 32-bit data object

CONTENTS

1 ADV212 INTRODUCTION

The ADV212 is a single-chip JPEG2000 codec targeted for video and high-bandwidth image compression applications that can benefit from the enhanced quality and features provided by the JPEG2000 (J2K)–ISO/IEC15444-1 image compression standard. The ADV212 implements the computationally intensive operations of the JPEG2000 image compression standard and provides fully compliant code-stream generation for most applications.

Depending on the particular application requirements, the ADV212 can also provide varying levels of JPEG2000 compression support. It can provide raw code-block and attribute data output that allows the host software to have complete control over the generation of the JPEG2000 code-stream and other aspects of the compression process such as bit-rate control.

The ADV212 contains a dedicated wavelet transform engine, three entropy codecs, an on-board memory system, and an embedded RISC processor that can provide a complete JPEG2000 compression or decompression solution. The wavelet processor supports the 9/7 irreversible wavelet transform and the 5/3 wavelet transform in reversible and irreversible modes. The entropy codecs support all features in the JPEG2000 Part 1 specification, except for Maxshift ROI.

Capable of encoding and decoding both still images and video, the ADV212 operates on a rectangular array of pixel samples called a tile. A tile may contain a complete image up to the maximum supported size, or some portion of an image. The maximum horizontal tile size supported depends on the wavelet transform selected and the number of samples in the tile. Images larger than the ADV212's maximum tile size may be broken into individual tiles and then sent sequentially to the chip while still maintaining a single, fully compliant, JPEG2000 code-stream for the entire image. As an example of its throughput, the ADV212BBCZ-150 is capable of processing in real-time up to two D1 (720x480) video streams, while two ADV212BBCZ-150s are capable of processing high-definition video (720p/60 or 1080i/60) signals.

Features

The ADV212 has the following features:

- Complete single-chip JPEG2000 compression or decompression solution for video and still images. No external memory is required.
- Patented Spatial Ultra-efficient Recursive Filtering (SURF™) technology enables low-power and low-cost wavelet-based compression.
- Supports both 9/7 and 5/3 wavelet transforms with up to six levels of transform.
- The 5/3 wavelet has programmable tile or image size with widths up to 2048 pixels in the pixel interface’s three-component 4:2:2 mode, and up to 4096 pixels in single-component mode. The maximum tile or image height is 4096 pixels. For the 9/7 wavelet, those values are cut in half.
- Pixel interface directly supporting ITU.R-BT656, SMPTE125M PAL/ NTSC, SMPTE274M, SMPTE293M [525p], ITU.R-BT1358[625p] or any video format with a maximum input rate of 65 Msamples/sec for irreversible mode or 40 Msamples/sec for reversible mode. Two or more ADV212s can be combined to support full-frame SMPTE274M HDTV [1080i] or SMPTE296M [720p].
- Compression on a field-by-field basis or using de-interlaced fields for standard-definition television (SDTV) video sources for improved performance and compression efficiency.
- Flexible asynchronous SRAM-style host interface allows glueless connection to most 16- or 32-bit microcontrollers and ASICs.
- 2.5-3.3 V I/O and 1.5 V core supply.
- 12 mm x 12 mm 121-ball fpBGA, speed grade 115 MHZ or 13 mm x13 mm 144-ball fpBGA, speed grade 150 MHz.

Typical Applications

The type of applications the ADV212 is used for includes the following:

- Image archival and retrieval for standard-definition television (SDTV), high-definition television (HDTV) , and high-resolution still images
- Digital closed-circuit television (CCTV) for security / surveillance systems
- Digital cinema systems
- Professional and HDTV video editing and recording systems
- Professional digital camcorders and high-resolution digital still cameras
- Networked video and image distribution systems
- Wireless video and image distribution

ADV212 Core Architecture

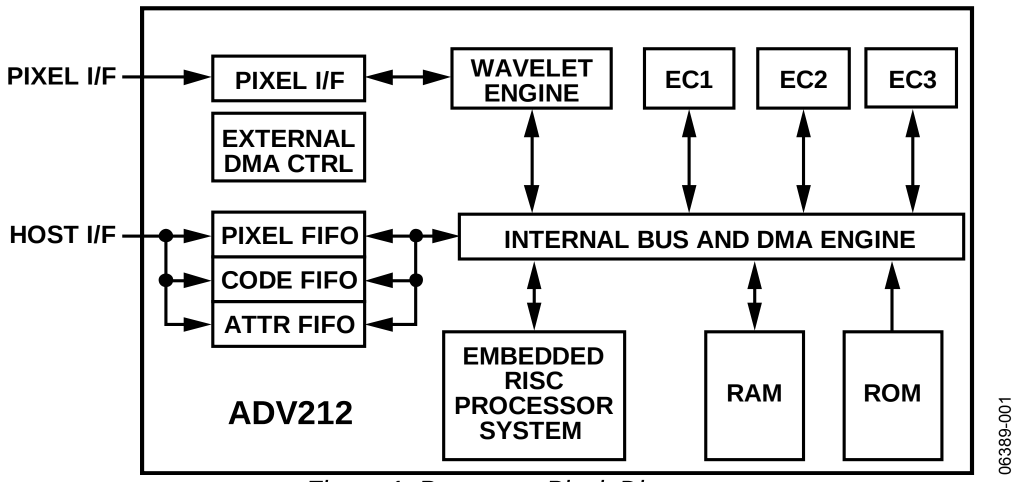

Figure 1. Processor Block Diagram

Functional Description

For a typical encode application, video or pixel data is input to the ADV212's pixel interface. Valid samples are then passed on to the wavelet engine, where each tile or frame is decomposed into subbands using the 5/3 or 9/7 filters. The resultant wavelet coefficients are then written to internal memory. The entropy codecs then code the wavelet data so that it conforms to the JPEG2000 standard. An internal direct memory access (DMA) engine provides high-bandwidth memory-to-memory transfers as well as high-performance transfers between functional blocks and memory.

Wavelet Engine

The ADV212 provides a dedicated wavelet transform processor based on Analog Devices' SURF technology. This processor can perform up to six wavelet decomposition levels on a tile.

In encode mode, the wavelet transform processor takes in uncompressed samples, performs the wavelet transform, and writes the wavelet coefficients in all frequency subbands to internal memory. Each of these subbands is then further broken down into code-blocks. The code-block dimensions can be user-defined, and are used by the wavelet transform processor to organize the wavelet coefficients into code-blocks when writing to internal memory. Each completed code-block is then entropy-coded by one of the entropy codecs.

In decode mode, wavelet coefficients are read from internal memory and are decoded into uncompressed samples.

Entropy Codecs

Each entropy codec block performs context modeling and arithmetic coding on a code block of the wavelet coefficients. Additionally, this block also performs the distortion metric calculations during compression that are required for optimal rate-distortion performance. Since the entropy coding process is the most computationally intensive operation in the JPEG2000 compression process, three dedicated hardware entropy codecs are provided on the ADV212.

Embedded Processor System

The ADV212 incorporates an embedded 32-bit RISC processor. This processor is used for configuration, control, and management of the dedicated hardware functions, as well as for code parsing and generation.

Memory System

The memory system’s main functions are to manage wavelet coefficient data and interim code-block and attribute data, and to provide temporary work space for creation, parsing, and storage of the JPEG2000 code-streams. The memory system can also be used as program and data memory for the embedded processor.

Internal DMA Engine

The internal DMA engine provides high-bandwidth memory-to-memory transfers as well as high-performance transfers between memory and functional blocks. This function is critical for high-speed code-stream generation and parsing.

Configurable FIFO Access

FIFOs are provided for pixel data, code-stream data, and attribute data. The FIFOs can be accessed directly from the host interface using normal addressed read and write cycles, or by external host DMA accesses using a DREQ/DACK protocol, or by one of the dedicated hardware handshake protocols. Each FIFO has a threshold that can be programmed to generate one interrupt when that threshold is reached.

2 ADV212 PHYSICAL INTERFACE

This chapter describes the ADV212 pixel and host interfaces.

ADV212 Pixel and Host Interfaces

There are several possible modes that can be used to interface to the ADV212. The designer can use the pixel interface VDATA bus and the host interface HDATA bus, or the HDATA bus alone.

Internal Embedded Processor

The ADV212 is a System-on-Chip (SoC) implementation, which means that it incorporates dedicated hardware functions including a processor and on-board firmware and software. It also means that configuration and management of the operations of the chip can be handled by the on-chip processor and its software. Other than basic bus and I/O configurations that the user has to set up, most of the configuration and control of the ADV212 is handled by the firmware. The firmware is responsible for setting registers in different functional blocks of the ADV212, scheduling events, and similar functions. These functional blocks are entirely controlled by the firmware; the user has no access to them.

ADV212 configuration and firmware load procedures are described in detail in the ADV212 programming guide.

Pixel Interface and VDATA Bus

The pixel interface can be used in applications where uncompressed pixel data is on a separate bus from compressed data. For example, when an ADV212 is in encode mode, a typical application has 8- or 10-bit digital input video on the VDATA bus. Often this comes from a video decoder (such as the ADV7189B), which converts the analog video into a digital format. When used with the VDATA interface, the HDATA bus is used to output compressed data from the ADV212.

A typical decode mode application uses a processor or DMA to write the compressed JPEG2000 stream using the HDATA bus and to drive a video encoder with the VDATA bus. This video encoder (such as the ADV7301A) converts the digital video back into an analog output.

The pixel interface can support input and output video data or still-image data in 8-, 10-, and 12-bit monochrome or YCbCr format. All pixel interface formats are left-justified (MSB) on the VDATA bus. If YCbCr data is used, it must be in 4:2:2 mode.

Video data can be input and output in several different modes on the VDATA bus. For these modes, the pixel clock must be input on the VCLK pin.

Optionally, the ADV212 can be used in de-interlaced mode when used in NTSC or PAL format, where two consecutive fields are interleaved and processed as one frame. This mode yields significantly better compression performance and efficiency for SDTV NTSC or PAL sources.

Additionally, high-definition digital video, such as SMPTE-274M (1080i) is supported using two or more ADV212 devices.

EAV/SAV Mode

The ADV212 can accept CCIR656 video formats. These modes use embedded time codes such as EAV and SAV in the video stream. This mode is set up by the user in the firmware parameters.

HVF Mode

The ADV212 can also accept video data accompanied by separate HSYNC, VSYNC, and FIELD (HVF) signals, where video data is interleaved onto a single bus. This is set up by the user in the firmware parameters.

Raw Video Mode

This mode is used for still-picture data and non-standard video. Video frame (VFRM), video strobe (VSTRB), and video ready (VRDY) are used to transfer pixel data between the host and the pixel interface. In this mode, the pixel interface can support up to 16-bit sample data. Since no timing codes or sync signals are associated with the data on the pixel interface, dimension registers (XTOT, YTOT) must be programmed to indicate the size of the image.

Host Interface and HDATA Bus

The ADV212 can connect directly to a wide variety of 16- and 32-bit host processors and ASICs using an asynchronous SRAM-style interface, DMA accesses, or a JDATA-mode interface. The ADV212 supports 16- and 32-bit buses for control and 8-, 16-, and 32-bit buses for data transfers. The control and data-channel bus widths can be specified independently, which allows the ADV212 to support applications that require control and data buses of different widths. The host interface is used for configuration, control, and status functions as well as for transferring compressed data streams. The host interface can be used for uncompressed data transfers in certain modes. The host interface may be shared for control and status communications, uncompressed data input and compressed data output, or compressed data input and uncompressed data output. The data streams may include:

- Uncompressed tile data (such as still-image data)
- Fully encoded JPEG2000 code-stream or unpackaged code-blocks
- Code-block attributes

The ADV212 uses big-endian word alignment for 16- and 32-bit transfers. All data is left-justified. This means that in a 32-bit system, the most significant byte is located at bits [31:24].

Pixel Input on the Host Interface

Pixel input on the host interface supports 8-, 10-, 12-, 14- or 16-bit raw pixel data formats. It can be used for pixel [still-image] input or compressed data output in encode mode, or compressed data input or pixel data output in decode mode. Since no timing codes or sync signals are associated with the input data on the host interface, dimension registers (XTOT, YTOT) must be programmed to indicate the size of the image.

See the application note *HIPI Mode and Still Image Applications* for details on how to use the ADV212 in HIPI mode.

HDATA Bus Configuration

The host interface provides several HDATA bus configurations to meet specific system requirements. The default bus mode uses the same HDATA pins to transfer both data and control information between the host and the ADV212.

The ADV212 can support 16- or 32-bit control transfers and 8-, 16-, or 32-bit data transfers. The size of the control and data buses can be selected independently by writing to the direct and indirect registers that configure bus size. This allows a 16-bit microcontroller to configure and control the ADV212 while still providing 32-bit data transfers to an ASIC or external memory system.

Pin Configuration and Bus Modes

The ADV212 provides a wide variety of control and data configurations that allow it to be used in many applications with little or no glue logic. The modes described below are configured using the BUSMODE register.

32-bit Host/32-bit Data Mode

In this mode the HDATA[31..0] pins provide full 32-bit-wide data access to the PIXEL, CODE, and ATTR FIFOs, as well as internal memory and registers.

16-bit Host/32-bit Data Mode

This mode allows a 16-bit host to configure and communicate with the ADV212 while still allowing 32-bit accesses to the PIXEL, CODE, and ATTR FIFOs using the external DMA capability. All addressed host accesses are 16-bits and therefore only use the HDATA[15..0] pins. The HDATA[31..16] pins are used to provide the additional 16-bits necessary to support the 32-bit external DMA transfers to and from the FIFOs for data.

16-bit Host/16-bit Data Mode

This mode uses 16-bit transfers if used for host interface transfers or external DMA data transfers. This mode allows for use of the extended pixel interface modes.

16-bit Host/8-bit Data in JDATA Bus Mode

This mode provides separate data input/output and host control interface pins. Host control accesses are 16 bits and use HDATA[15..0], while the compressed data is transferred over the dedicated 8-bit JDATA interface. JDATA uses a valid/hold synchronous transfer protocol based on MCLK. The direction of the JDATA bus is determined by the mode of the ADV212. If the ADV212 is encoding (compression) then JDATA[7..0] is an output. If the ADV212 is decoding (decompression) then JDATA[7..0] is an input. Host control accesses remain asynchronous. Host interface accesses are independent from JDATA data transfers.

Registers

Direct and indirect registers and internal memory are used to program the ADV212 as well as receive or send compressed video data.

Direct Registers

The most commonly accessed resources on the ADV212 are available as Direct Registers, which are addressed using the 4-bit ADDR bus. This provides access to 16 locations. Other resources on the ADV212 can be accessed with the use of the Indirect Address (IADDR) and Indirect Data (IDATA) registers.

The IDATA, IADDR, and FIFO direct registers are 32 bits wide. The PLL_HI and PLL_LO registers are 8 bits wide. The remaining registers are 16 bits wide.

When accessing 8- and 16-bit registers in 32-bit host mode, the unused most significant bits should be ignored on the HDATA bus.

Indirect Registers

Since the ADV212 contains both 16- and 32-bit registers and its internal memory is mapped as 32-bit data, a mechanism has been provided to allow a 16-bit host interface to access these registers and memory locations. This is accomplished with the staging register (STAGE). The STAGE register is not used in 32-bit host mode.

The STAGE register is a direct register used by the host interface to set the indirect address register (IADDR) as well as for writing and reading data to that indirect memory location. Prior to writing to the desired location, the STAGE register must be written with the most significant 16-bit word of the address for that memory location. This is followed by writing the least significant 16-bit word of the address to the IADDR register. These two16-bit words are combined to create the full 32-bit address. Similarly, to write a 32-bit data value, STAGE is first written with the most significant 16-bit word, followed by writing the least significant 16-bit word to the IDATA register.

When a register is read, the upper [most significant] 16-bit word is returned immediately on HDATA and the lower 16-bit word can be retrieved by reading the STAGE register on a subsequent access.

Programming examples can be found in the ADV212 programming guide.

This does not apply to the PIXEL, CODE, and ATTR FIFOs. These channels are always accessed at the specified data width and do not require the use of the STAGE register.

External DMA Interface

The External DMA Interface is provided to enable high-bandwidth data I/O between an external DMA controller and the ADV212 data FIFOs. There are two independent DMA channels that can each be assigned to any one of the data stream FIFOs (PIXEL, CODE, or ATTR).

The controller supports asynchronous DMA using a DREQ/DACK (Data-Request/Data-Acknowledge) protocol in either single or burst access modes. Additional functionality is provided by Fly-By mode and Dedicated Chip Select (DCS) modes.

Video Input Formats

The ADV212 supports a wide variety of formats for uncompressed video and still-image data. See “Video Input Formats” on page 69 for more information. The actual interface and bus modes selected for transferring uncompressed data determines the allowed size of the input data and the number of samples transferred with each access. The host interface can support 8-, 10-, 12-, 14-, and 16-bit data formats on the HDATA bus.

The pixel interface can support input and output video data or still-image data. All possible formats are listed in “System Registers” on page 26 of this user’s guide. All formats can support less precision than provided by specifying the actual data width and precision. This can be set in the PMODE register or through a firmware parameter.

The maximum allowable data-input rate is dependent on the wavelet transform and the data-width (or precision) of the input samples.

Table 26 and Table 27 of the ADV212 datasheet list the maximum allowable data input rates.

3 VIDEO APPLICATIONS

This section describes typical video applications using the ADV212 JPEG2000 Video Processor.

Video Applications

A single ADV212 can be in either Encode or Decode mode depending on what firmware the user has loaded into the device. An ADV212 can not change from Encode mode to Decode mode unless new firmware is loaded into the chip and the chip has gone through a reset and configuration process again.

Encode in a Multi-chip Application

Due to the pixel data rate limitation, an 1080i application requires at least 2 ADV212s to encode or decode full resolution 1080i or 720p video. In encode mode, the ADV212 accepts Y and CbCr data on separate buses. An encode example is shown in Figure 2 on page 18.

In decode mode, a master/slave configuration (as shown in Figure 3 on page 19) or a slave/slave configuration can be used in order to synchronize the outputs of the two ADV212s. See the application note "Using the ADV212 in a multi-chip application" for more details. This synchronization is necessary for glueless connection to standard video interface chips, such as the ADV7402.

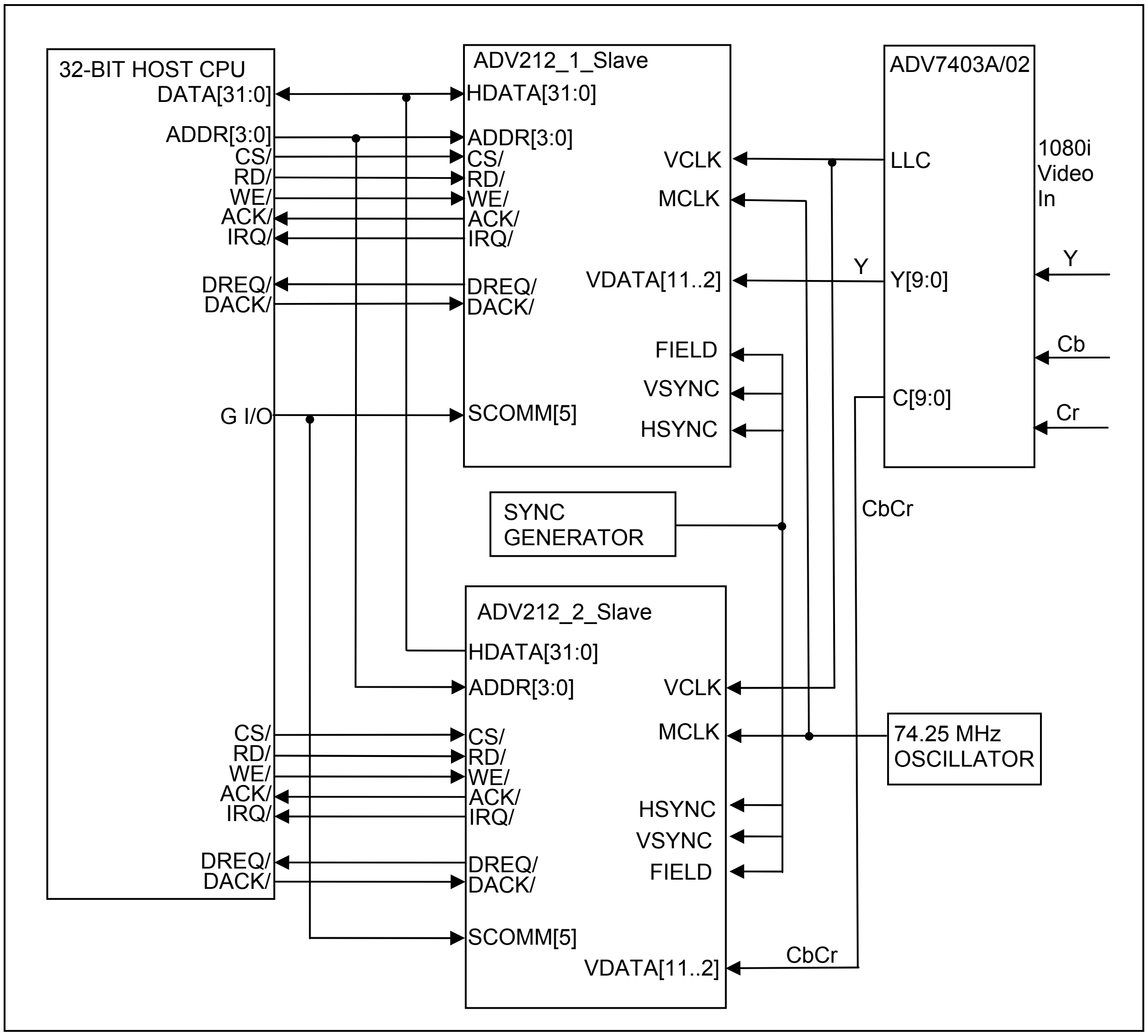

Figure 2. Encode in a Slave/Slave Multi-chip Application

Decode in Multi-chip Applications

In a master/slave configuration, it is expected that the master HVF outputs are connected to the slave HVF inputs and that each SCOMM[5] pin is connected to the same GPIO on the host.

In a slave/slave configuration the common HVF for both ADV212s is generated by an external sync generator and each SCOMM[5] is connected to the same GPIO output on the host.

SWIRQ1, Software Interrupt 1 in the EIRQIE register must be unmasked on both devices to enable multi-chip mode in order to synchronize video data at the VDATA outputs.

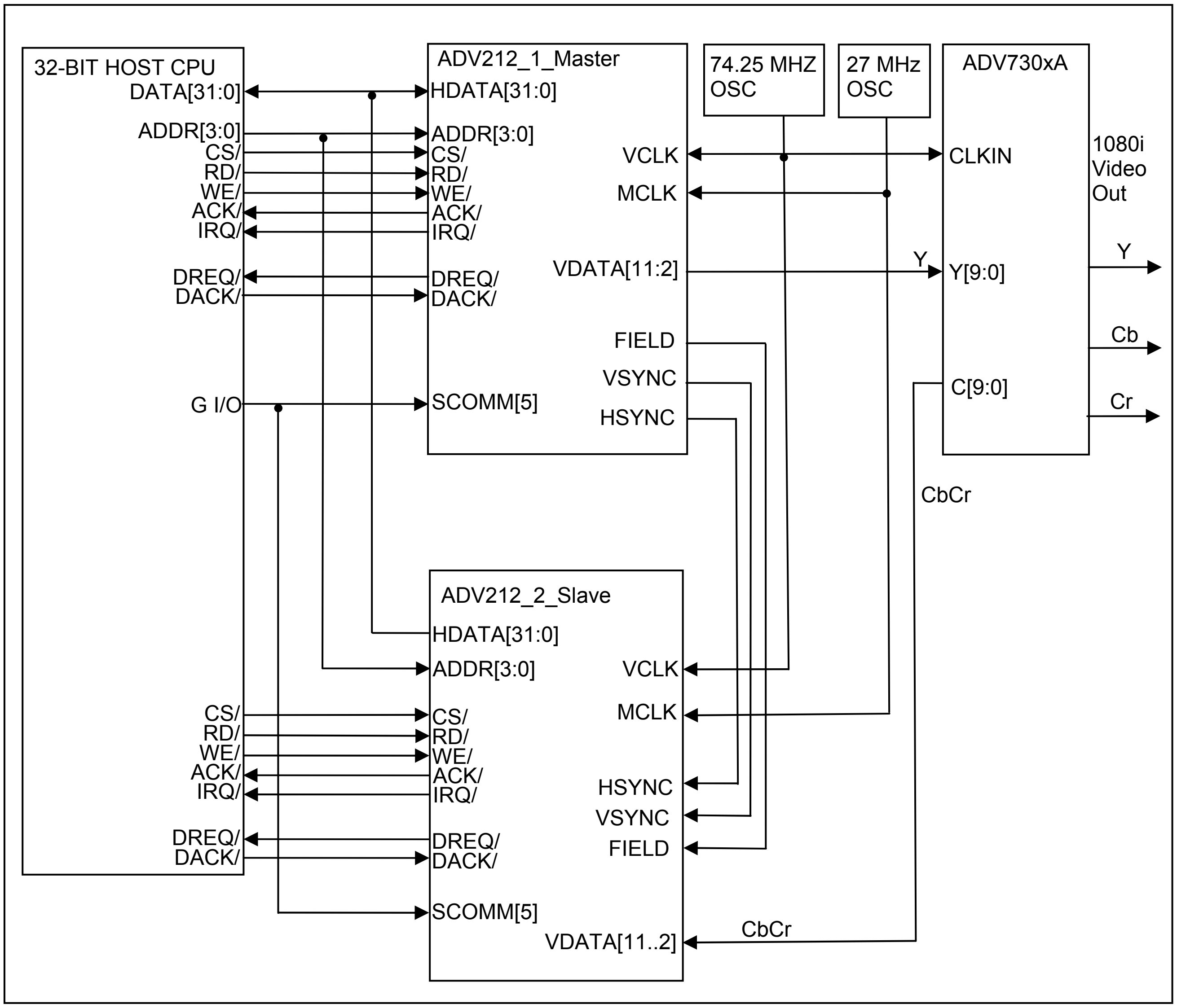

Figure 3. Decode in a Master/Slave Multi-chip Application

Digital Still Camera and Camcorder Applications

Raw Video pixel mode can be used for still-image applications since there is no blanking or timing information and the pixel data only contains raw image data.

The ADV212 programming guide should be consulted for raw video pixel configuration and programming sequence when using this mode.

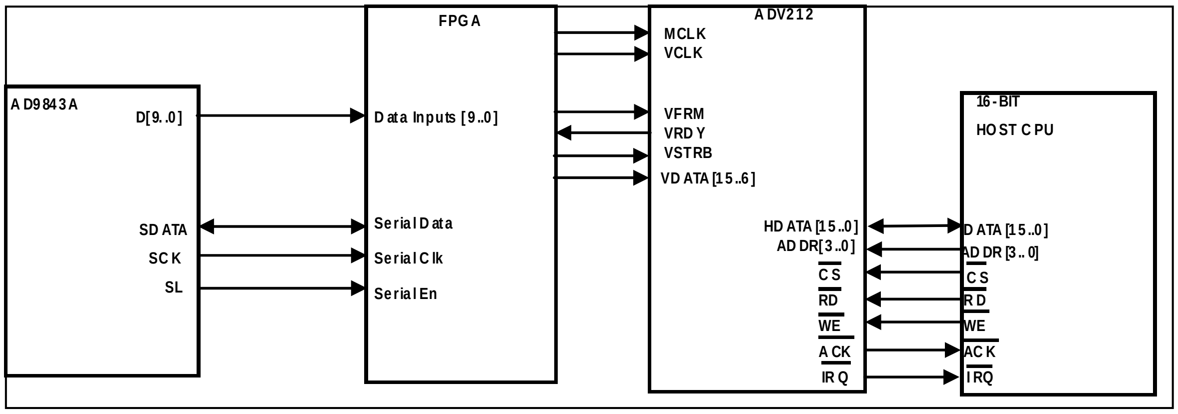

Figure 4. Digital Still Camera and Camcorder Applications

Encode and Decode SDTV Video Application

The figures below show a normal host mode SDTV application in 10-bit CCIR656 pixel interface mode.

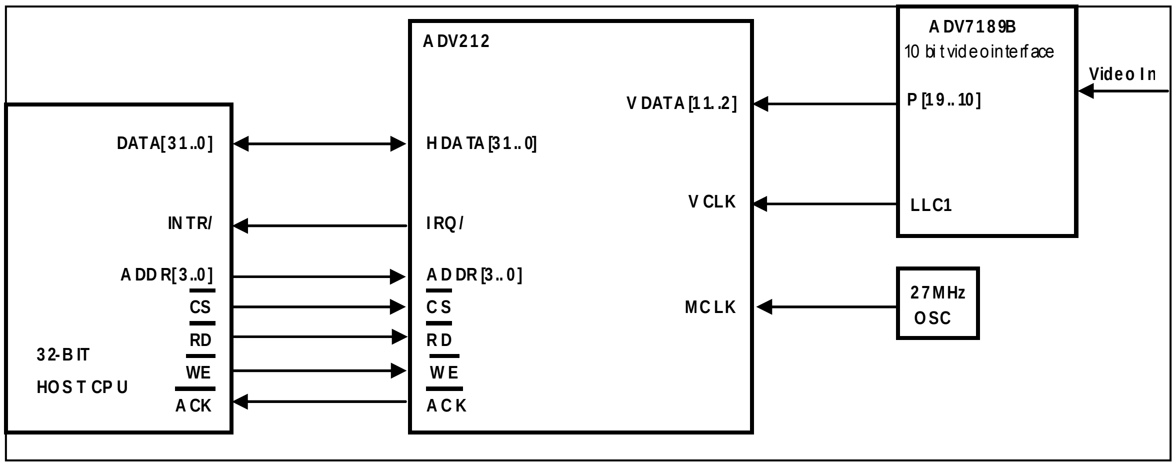

Figure 5. Encode SDTV Video Application

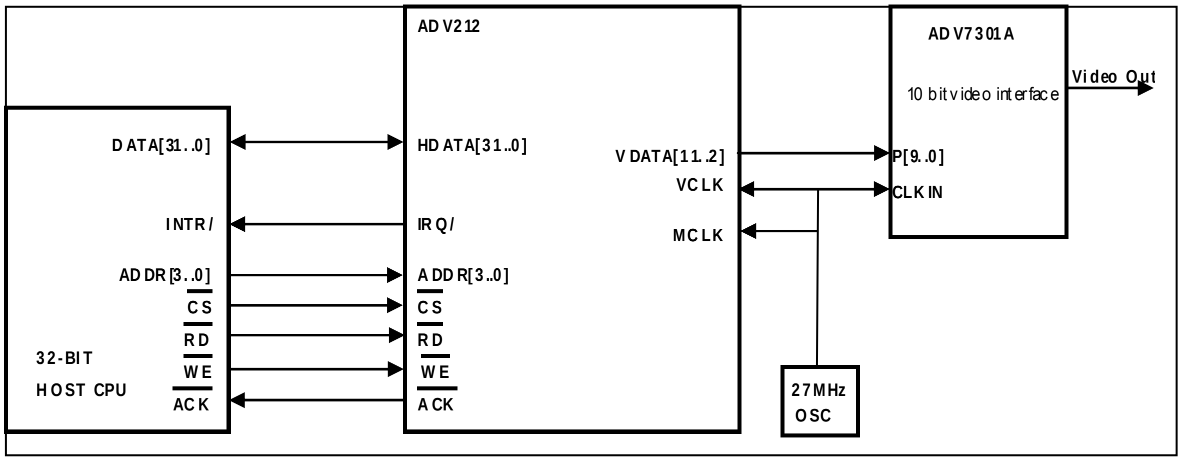

Figure 6. Decode SDTV Video Application

32-bit Host/32-bit FPGA Application

The figures below show encoding and decoding for a 10-bit CCIR656 FPGA application using DMA DREQ/DACK mode for compressed data transfer.

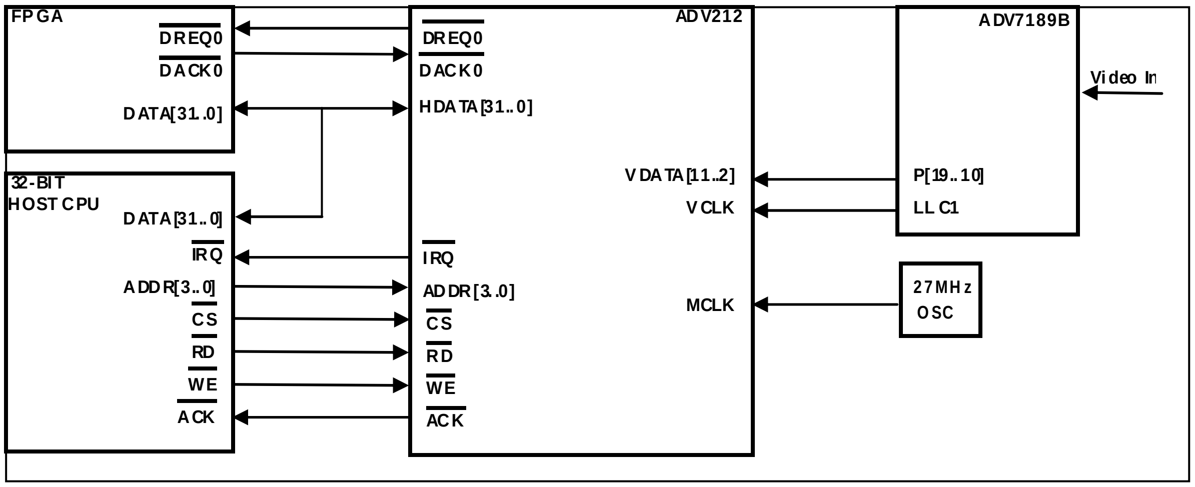

Figure 7. 32-bit Host/32-bit FPGA – Encode

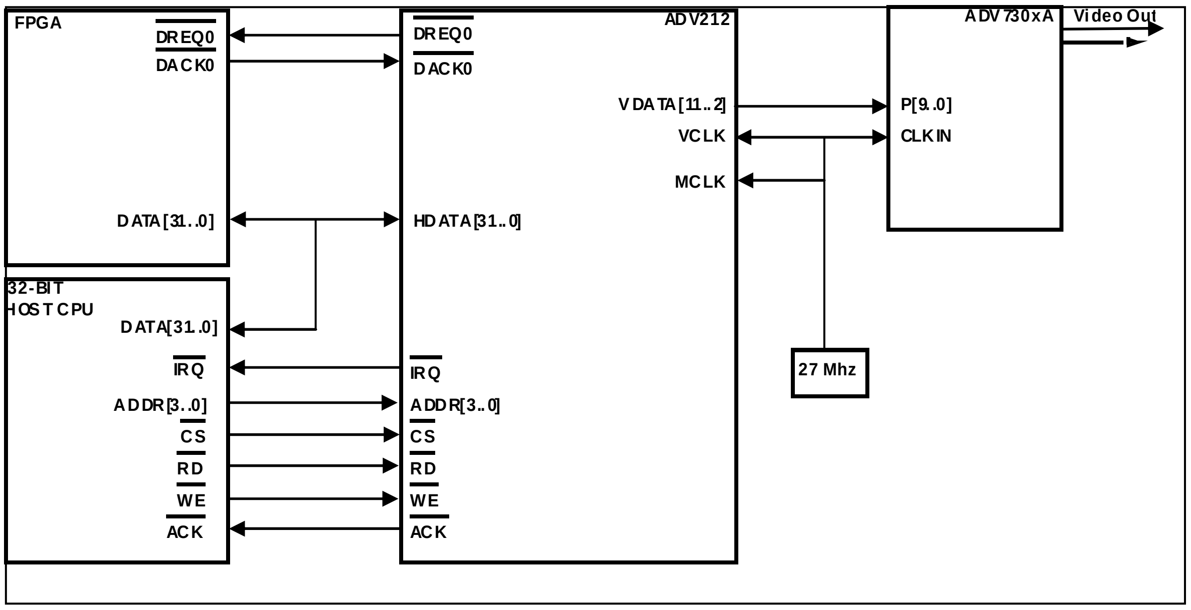

Figure 8. 32-bit Host/32-bit FPGA – Decode

The figure above shows an SDTV application that uses external DMA channels, configured by the EDMOD registers, to transfer compressed data to and from the ADV212. The ADV212 programming guide should be consulted for configuration and programming sequence when using this configuration.

HIPI (Host Interface Pixel Interface) Application

For more information about how to use the ADV212 in HIPI mode, the technical notes *ADV212 HIPI Mode* and *HIPI Mode Overview* at ftp://ftp.analog.com/pub/Digital_Imaging/ADV212_ApplicationNotes/ should be consulted.

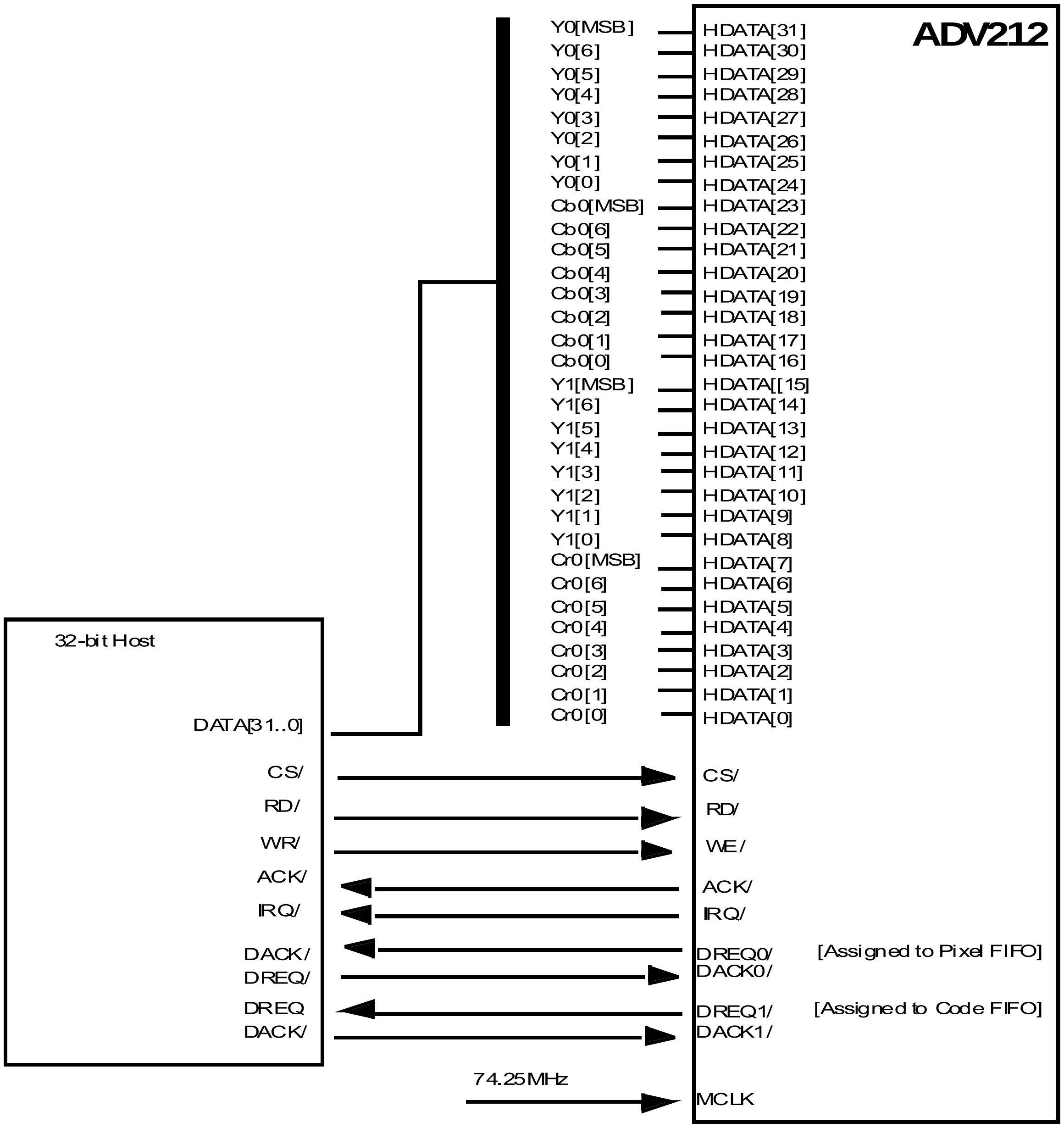

Figure 9. HIPI Application Encode

JDATA Application

JDATA Mode in a CCIR656 Application

The figure below shows an example of using a 16-bit host for configuration and JDATA mode for compressed video output.

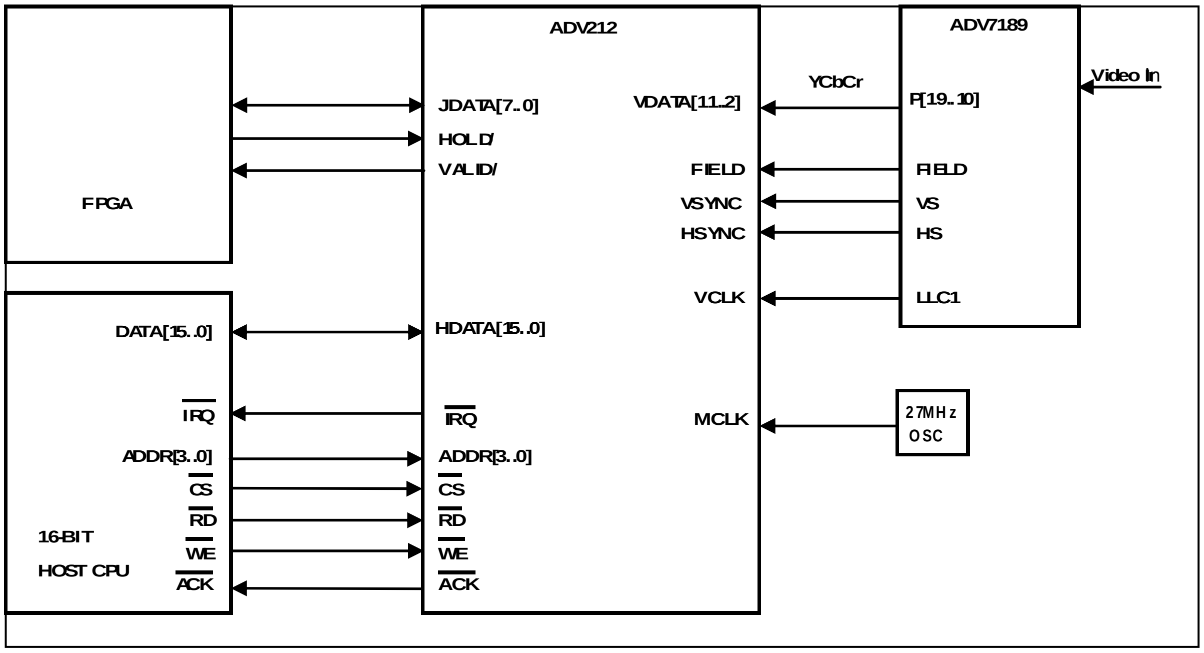

Figure 10. JDATA - 16-bit Host with Compressed Video Output

Raw Video Mode and JDATA Mode

On the ADV202, it was not possible to use JDATA Mode and Raw Video Mode simultaneously. This capability has been added in the ADV212.

To use Raw Video mode and JDATA mode on the ADV212 the following must be true:

- Uncompressed video data is interfaced over VDATA[15:0].
- Compressed video data is interfaced over JDATA[7:0].
- BUSMODE[6:4] must be set to 0x0003 for Raw Video Mode and JDATA Mode.
- EDMOD0[5:3] must be set to 0x0003.

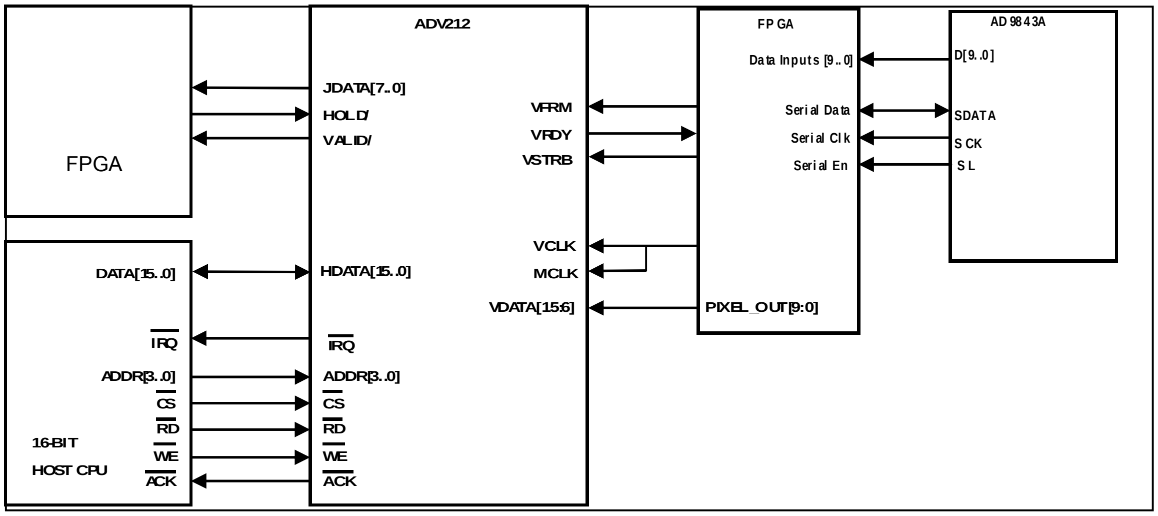

Figure 11. JDATA - 16-bit Host with Raw Video Input

4 SYSTEM REGISTERS

This section provides a functional description of the direct and indirect registers of the ADV212.

- There are 16 direct registers with a *direct* address range from 0x0 to 0xF. The registers are accessed using the ADDR [3..0] pins for address and the HDATA bus for data. These registers are 8, 16 or 32 bits wide. See Table 1 on page 28 for a list of these registers.
- The Indirect register address range is from *indirect* addresses 0xFFFF0400 to 0xFFFF14FC. These registers are 16 bits wide and aligned on word (4-byte) boundaries. See Table 25 on page 41 for a list of indirect registers. Indirect address and data registers can be accessed by properly setting the IADDR and IDATA direct registers.
- The indirect memory locations are used for the storage of the firmware and firmware parameters. The 32KB encode or decode firmware is stored at starting address 0x00050000.
  The firmware parameters are stored at starting address 0x00057F00. These memory locations are 32 bits wide and are aligned on 4-byte boundaries. Indirect memory address and data registers can be accessed by properly setting the IADDR and IDATA direct registers.

0x00
0x0F
Direct Registers

Figure 12. Direct Register Memory Map

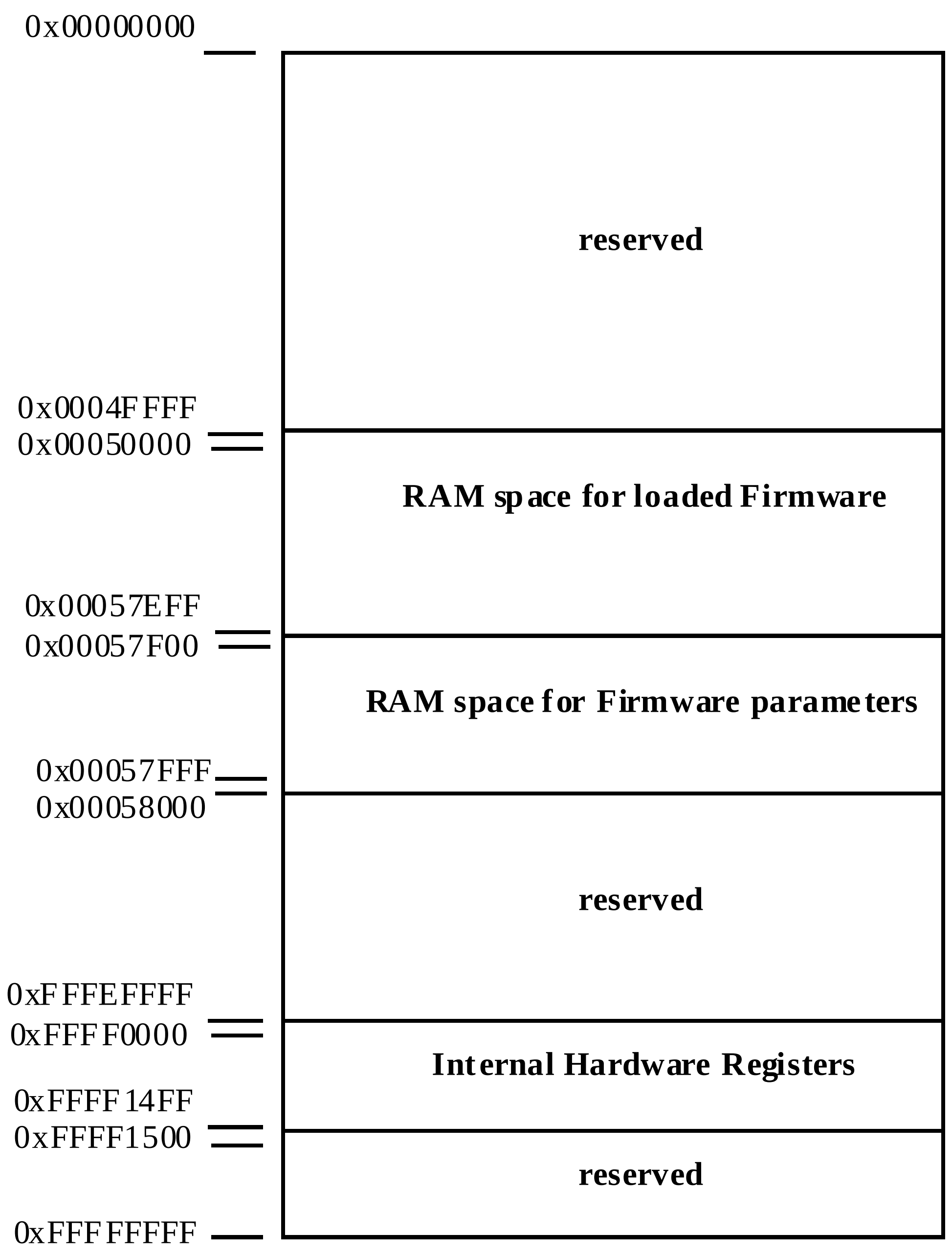

Figure 13. Indirect Registers and Firmware Memory Map

Direct Registers

The ADV212 has 16 Direct Registers as shown in Table 1. The Direct Registers are accessed by the host interface using ADDR [3..0], HDATA[31..0], $\overline{\text{CS}}$, $\overline{\text{RD}}$, $\overline{\text{WE}}$, and $\overline{\text{ACK}}$ pins.

The host must first initialize the Direct Registers before any application-specific operation can be implemented. The PLL_HI and PLL_LO registers are the only registers that can be read/write verified on bootup.

DIRECT REGISTER MEMORY MAP

Table 1.

Address	Name	Description	R/W	Length
0x00	PIXEL	PIXEL FIFO Access Register	R/W	[31..0]
0x01	CODE	Compressed Code-Stream Access Register	R/W	[31..0]
0x02	ATTR	ATTR FIFO Access Register	R/W	[31..0]
0x03	Reserved	Reserved	R/W	[31..0]
0x04	Reserved	Reserved	R/W	[31..0]
0x05	EIRQIE	External Interrupt Enabled	R/W	[15..0]
0x06	EIRQFLG	External interrupt Flags	R/W	[15..0]
0x07	SWFLAG	Software Flag Register	R	[15..0]
0x08	BMODE	Bus Mode Configuration Register	R/W	[15..0]
0x09	MMODE	Miscellaneous Mode Register	R/W	[15..0]
0x0A	STAGE	Staging Register	R/W	[15..0]
0x0B	IADDR	Indirect Address Register	R/W	[31..0]
0x0C	IDATA	Indirect Data Register	R/W	[31..0]
0x0D	BOOT	Boot Mode Register	R/W	[15..0]
0x0E	PLL_HI	PLL Control Register - High Byte	R/W	[15..0]
0x0F	PLL_LO	PLL Control Register - Low Byte	R/W	[15..0]

PIXEL FIFO ACCESS REGISTER

Table 2.

Address	Name	Description	R/W	Reset Value
0x00	Pixel	PIXEL FIFO Access Register	R/W	Undefined

The 32-bit PIXEL register is used to access the PIXEL FIFO via normal addressed accesses and generally is used for pixel data input and output on the host interface (HIPI mode). The actual bus width when accessing this register depends on the host control data width selected [BUSMODE/HWIDTH].

- 16-bit mode will accept/return data on HDATA[15..0]
- 32-bit mode will accept/return data on HDATA[31..0]

In 16-bit mode, the upper unused bytes will be ignored on writes and return zeros on reads. The depth of the PIXEL FIFO is fixed to 256 x 32-bits.

A read operation when the PIXEL FIFO is empty or a write when it is full will cause the PFERR bit in the EIRQFLG direct register to be set.

COMPRESSED CODE-STREAM ACCESS REGISTER

Table 3.

Address	Name	Description	R/W	Reset Value
0x01	CODE	Compressed Code-Stream Access Register	R/W	Undefined

The 32-bit CODE register is used to access the JPEG2000 compressed code-stream FIFO via normal addressed accesses. It is also used for accessing raw code blocks when the ADV212 is in host interface pixel interface (HIPI) mode. The actual bus width when accessing this register depends on the host control data width selected [BUSMODE/HWIDTH].

In Encode mode, a host processor can read the compressed JPEG2000 stream using the CODE register in 16- or 32-bit mode. In Decode mode, it can write this compressed stream to the CODE register.

- 16-bit mode will accept/return data on HDATA[15..0]
- 32-bit mode will accept/return data on HDATA[31..0]

In 16-bit mode, the upper unused bytes will be ignored on writes and return zeros on reads.

A read operation when the CODE FIFO is empty or a write when it is full, will cause the DFERR bit in the EIRQFLG direct register to be set.

ATTRIBUTE REGISTER

Table 4.

Address	Name	Description	R/W	Reset Value
0x2	ATTR	ATTR FIFO Access	R/W	Undefined

The 32-bit ATTR register is used to access the attribute FIFO via normal addressed accesses. The actual bus width when accessing this register depends on the host control data width selected [BUSMODE/HWIDTH].

- 16-bit mode will accept/return data on HDATA[15..0]
- 32-bit mode will accept/return data on HDATA[31..0]
- In 16-bit mode, the upper unused bytes will be ignored.

A read operation when the CODE FIFO is empty or a write when it is full, will cause the AFERR bit in the EIRQFLG direct register to be set.

EXTERNAL INTERRUPT ENABLE REGISTER

Table 5.

Address	Name	Description	R/W	Reset Value
0x05	EIRQIE	External Interrupt Enables	R/W	See bit field descriptions.

This 16-bit EIRQIE register is used to enable the interrconditions that causes an external interrupt to occur on the IRQ/ pin. See the ADV212 programming guide for more information.

EIRQIE REGISTER BIT FIELDS

Table 6.

Bit	Name	Description	Reset Value
0	PFTH	PIXEL FIFO threshold condition exists (Level sensitive)	0
1	DFTH	CODE FIFO threshold condition exists (Level sensitive)	0
2	AFTH	ATTR FIFO threshold condition exists (Level sensitive)	0
3	Reserved	Reserved	0
4	PFERR	PIXEL FIFO has overflowed or underflowed	0
5	DFERR	Data FIFO has overflowed or underflowed.	0
6	AFERR	ATTR FIFO has overflowed or underflowed.	0
7	Reserved	Reserved	0
8	Reserved	Always write 0	0
9	Reserved	Always write 0	0
10	SWIRQ0	Software interrupt 0	0
11	SWIRQ1	Software interrupt 1	0
12	SWIRQ2	Software interrupt 2 *(reserved for future use)*	0
13	Reserved	Reserved	0
14	Reserved	Reserved	0
15	Reserved	Reserved	0

EXTERNAL INTERRUPT FLAG REGISTER

Table 7.

Address	Name	Description	R/W	Reset Value
0x6	EIRQFLG	External Interrupt Flags	R/W	See bit field descriptions.

The 16-bit EIRQFLG register indicates that external interrupt conditions are currently active. The bits in this register correspond directly to the bits in the external interrupt enable register (EIRQIE).

ⓘ Individual interrupts are cleared by writing a ‘1’ in the proper bit position of this register. A common mistake is trying to clear the interrupt flag before servicing the interrupt. For example, if you use the host processor to read JPEG2000 data out of the CODE FIFO after the DFTH interrupt is set (the CODE FIFO threshold condition is set). First the data must be read out of the FIFO, then the interrupt must be cleared by writing to the corresponding bit in the EIRQFLG register. Otherwise, if the interrupt flag is cleared before data is read out, the interrupt never gets cleared because the data count in the FIFO is still above the FIFO threshold.

EIRQFLG REGISTER BIT FIELD DESCRIPTIONS

Table 8.

Bit	Name	Description	Reset Value
0	PFTH	PIXEL FIFO threshold condition exists (Level sensitive)	1
1	DFTH	CODE FIFO threshold condition exists (Level sensitive)	1
2	AFTH	ATTR FIFO threshold condition exists (Level sensitive)	1
3	Reserved	Reserved	1
4	PFERR	PIXEL FIFO has overflowed or underflowed	0
5	DFERR	Data FIFO has overflowed or underflowed.	0
6	AFERR	ATTR FIFO has overflowed or underflowed.	0
7	Reserved	Reserved	0
8	Reserved	Reserved	0
9	Reserved	Reserved	0
10	SWIRQ0	Software interrupt 0	0
11	SWIRQ1	Software interrupt 1	0
12	SWIRQ2	Software interrupt 2	0
13	Reserved	Reserved	0
14	Reserved	Reserved	0
15	Reserved	Reserved	0

SWFLAG REGISTER

Table 9.

Address	Name	Description	R/W	Reset Value
0x7	SWFLAG	*Application ID*	R	See bit field descriptions.

The 16-bit SWFLAG register is used to read the Application ID after the ADV212 Encode or Decode firmware has been uploaded by the host processor. After the firmware has been uploaded and a soft reboot has been initiated, the embedded firmware takes control of this register and will write the Application ID into this register. Application IDs are different for Encode and Decode. This register is often used by the 16- or 32-bit host processor to ensure the firmware upload was successful.

BUSMODE REGISTER

Table 10.

Address	Name	Description	R/W	Reset Value
0x8	BUSMODE	Bus Mode Configuration	R/W	See bit field descriptions.

The 16-bit BUSMODE register configures host control and data bus.

BUSMODE REGISTER BIT FIELDS

Table 11.

Bit	Name	Description	Reset Value
1..0	HWIDTH	Host control data width 00 = Invalid 01 =16-bits 10 = 32-bits 11 = Invalid	'01'
3..2	DWIDTH	DMA data width 00 = 8 bits 01 =16 bits 10 = 32 bits 11 = Invalid	'01'
6...4	BCFG	Bus configuration 000 = Normal. HDATA [31..0] are available for host or data transfers according to the settings of HWIDTH and DWIDTH. For normal read/write access HWIDTH and DWIDTH are set to 1 or 2. 001 = JDATA Mode. Enables JDATA[7..0]. HWIDTH must be set to 1 and DWIDTH must be set to 0. 010 = Raw Video Mode. HWIDTH should be set to 16-bit and DWIDTH should be set to 16- or 8-bit mode. 011 = Simultaneous JDATA and Raw Video modes. HWIDTH should be set to 16-bit mode and DWIDTH set to 8-bit mode. 111= Reserved	'000'
7	Reserved	Reserved for future use; always write 0.	0
15...8	N/A	Reserved for future use; always write 0.	undefined

MMODE REGISTER

Table 12.

Address	Name	Description	R/W	Reset Value
0x9	MMODE	Misc. Mode Configuration	R/W	See bit field descriptions.

This 16-bit MMODE register configures miscellaneous functions for indirect access for 32-bit or 16-bit host interfaces.

Proper use and examples of this register can be found in "Host 32- and 16-bit Accesses to Direct/Indirect Registers and Indirect Memory" on page 60.

MMODE REGISTER BIT FIELDS

Table 13.

Bit	Name	Description	Reset Value
1..0	IWIDTH	Indirect access widths 00 = Invalid 01 = 16-bits 10 = 32-bits 11 = Invalid	'01'
3..2	IAUTOSTP	IADDR increment/decrement size. IADDR can automatically increment/decrement the IADDR location. This would require only one write to the IADDR register/memory. It can also be disabled., in which case each indirect register/memory access requires a write to IADDR and IDATA. 00 = 8-bit increment/decrement 01 = 16-bit increment/decrement 10 = 32-bit increment/decrement 11 = Disable auto increment/decrement	'01'
4	IAUTOMOD	Indirect address auto increment/decrement 0 = increment 1 = decrement	0
5	CTLREGAM	Control register address mode. Enable full indirect address map. *The control register address space is 0xFFFF0000 and above.* 0 = Full address mode. Provides full access to the ADV212's internal address space 1 = Control register mode. The upper 16-bits of the internal address are forced to the beginning of the control register address map. This eliminates setting the upper 16-bits of the indirect address register prior to each access.	0
15..6	Reserved	Reserved for future use; always write 0.	Undefined

STAGE REGISTER

Table 14.

Address	Name	Description	R/W	Reset Value
0xA	STAGE	Staging	R/W	Undefined

The 16-bit STAGE register is used when accessing 32-bit registers in 16-bit host mode. It is not required for a 32-bit host. It is used to access 32-bit words in indirect memory, indirect registers, and 32-bit direct registers. The STAGE register acts as a holding or "staging" register. For example, when using a 16-bit host the STAGE register is used in conjunction with the IADDR register to configure the 32-bit address or the IDATA register.

Proper use and examples of this register can be found in "Host 32- and 16-bit Accesses to Direct/Indirect Registers and Indirect Memory" on page 60.

IADDR REGISTER

Table 15.

Address	Name	Description	R/W	Reset Value
0xB	IADDR	Indirect Address	R/W	Undefined

The 32-bit IADDR register is used to set the address for indirect register and indirect memory read/write accesses.

The indirect address may optionally be auto-incremented/decremented by setting the IAUTOMOD and IAUTOSTP fields in the MMODE register.

Proper use and examples of this register can be found in "Host 32- and 16-bit Accesses to Direct/Indirect Registers and Indirect Memory" on page 60.

IADDR REGISTER

Table 16.

Address	Name	Description	R/W	Reset Value
0xC	IDATA	Indirect Data	R/W	Undefined

The 32-bit IDATA direct register is used to read and write data to an indirect register or indirect memory. The data value written to the IDATA register is written to the address in the IADDR register..

The indirect address may optionally be auto-incremented by setting the IAUTOMOD and IAUTOSTP fields in the MMODE register.

Proper use and examples of this register can be found in "Host 32- and 16-bit Accesses to Direct/Indirect Registers and Indirect Memory" on page 60.

BOOT REGISTER

Table 17.

Address	Name	Description	R/W	Reset Value
0xD	BOOT	Boot Mode	R/W	See bit field descriptions.

The 16-bit BOOT register is used to configure the boot mode and initiate a soft or hard reset of the ADV212.

A hard reset, one made by using the reset pin or setting the HARDRST bit, causes bits [2..1] to be loaded from the configuration pins as specified below, all other bits in this register will be set as specified in the bit field table.

The bits in the BOOT register are used to select various boot modes of the ADV212 after reset and to load specific instruction sets into memory. The boot mode can be configured via hardware [over the CFG pins] or in writing to the Boot Mode register. In a hardware configuration after a hard reset, the boot mode will be set to the values on the configuration pins CFG[2..1].These bits are not reset on a soft reset; see HARDRST and SOFTRST bit definitions below. The first boot mode after power-up is set by the CFG pins. Only boot mode 2 is available in hardware boot mode.

A soft reset, via setting the SOFTRST bit will only clear the SOFTRST bit, all other bits will remain unchanged. A soft reset affects all ADV212 internal registers except for the BOOT, PLL_LO, or PLL_HI registers. A hard reset, one made by using the reset pin or the BOOT register, resets all of the ADV212 internal registers.

BOOT REGISTER BIT FIELD DESCRIPTIONS

Table 18.

Bit	Name	Description	Reset Value
2..0	BOOTMODE	SOFTWARE BOOTMODE 000 Reserved. For internal use only. Do not use. 001 Reserved. For internal use only. Do not use. 010 No-boot host mode. TheADV212 does not boot but all internal registers and memory are accessible through normal host I/O operations. 011 Reserved. For internal use only. Do not use. 100 Reserved. 101 Co-processor boot Used in conjunction with the no-boot host mode and starts executing the loaded firmware. 110 Reserved. For internal use only. Do not use 111 Reserved. For internal use only. Do not use HARDWARE BOOTMODE 000 Reserved. For internal use only. Do not use. 001 Not Available 010 No-boot host mode, ADV212 does not boot but all internal registers and memory are accessible through normal host I/O operations. 101 Not Available 100 Reserved. 101 Not Available 110 Not Available 111 Not Available	000
3	Reserved	Reserved for future use; always write 1.	1
5..4	Reserved	Reserved for future use; always write 0.	0
6	HARDRST	Setting this bit will cause the ADV212 to execute a hard reset. This is identical to asserting a reset on the RESET pin. The value in BOOT MODE will be ignored since it will be immediately reloaded from the configuration pins. This bit will override the SOFTRST bit if both bits are set concurrently. The PLL control register is reset to its reset value as well as the BOOT MODE [2..1] bits representing the values on the CFG [2..1] pins.	0
7	SOFTRST	Setting this bit will cause the ADV212 to execute a soft reset. This is similar to asserting the RESET/ pin except that the value in BOOT MODE will be used to determine how the ADV212 will boot instead of using the configuration pins. BOOT MODE [2..0] and the PLL control register will not be reset. This bit will be cleared after the soft reset, but all other bits in this register will remain unchanged.	0
15..8	Reserved	Reserved for future use; always write 0.	Undefined

PLL_HI REGISTER

Table 19.

Address	Name	Description	R/W	Reset Value
0xE	PLL_HI	PLL Control high byte	R/W	See bit field descriptions.

The 16-bit PLL_HI register is used to configure the on-chip PLL. Internally, the ADV212 uses two clock domains that are generated by an on-chip PLL from the input clock MCLK.

JCLK is used to clock all of the JPEG2000 specific hardware blocks and HCLK is used to clock the embedded processor. A block diagram of the PLL and its control parameters are shown in Figure 14 on page 38.

Any time the PLL_HI register is changed, the host must wait at least 20us before reading or writing any other register. If this delay is not implemented, erratic behavior may result.

PLL_HI REGISTER BIT FIELDS

Table 20.

Bit	Name	Description	Reset Value
0	PLLPDN	Power down enable = 1	0
1	BYPASS	Bypass enabled = 1	0
2	Reserved	Always write 0	0
3	HCLKD	HCLK divider enabled = 1	1
4	ACK_FN	0 – $\overline{\text{ACK}}$ is actively driven 1 – $\overline{\text{ACK}}$ is configured as an open drain output	0
6..5	Reserved	ADI use only; always write 0	Undefined
7	Reserved	Reserved	1
15..8	N/A	Not Available	Undefined

PLL_LO REGISTER

Table 21.

Address	Name	Description	R/W	Reset Value
0xF	PLL_LO	PLL Control low byte	R/W	See bit field descriptions.

This 16-bit PLL_LO register is used to configure the on chip PLL.

Any time the PLL_LO register is modified, the host must wait at least 20us before reading or writing any other register. If this delay is not implemented, erratic behavior may result.

PLL_LO REGISTER BIT FIELD DESCRIPTIONS

Table 22.

Bit	Name	Description	Reset Value
4..0	PLLMULT	Multiplier [values from 0 to 31]	'00011'
5	LFB	Loop feedback divider enable = 1	0
6	Reserved	Always write 0	0
7	IPD	Input clock divider enable = 1	0
15..8	N/A	Not applicable	Undefined

In the figure below, the MCLK pin input is the external clock source.

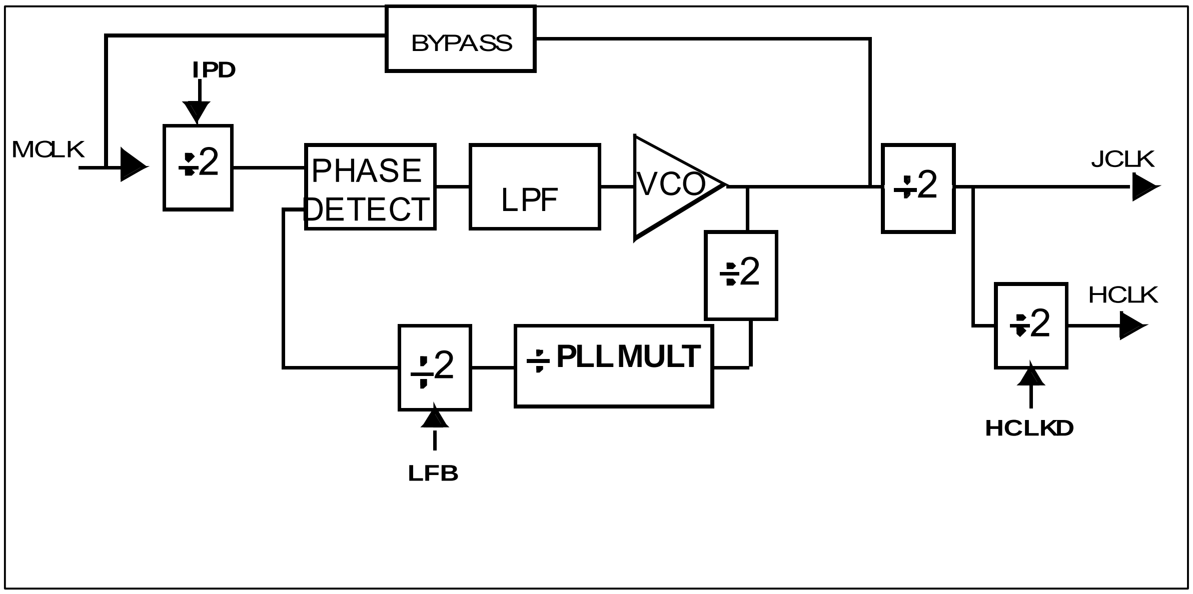

Figure 14. PLL Architecture

RECOMMENDED REGISTER SETTINGS FOR THE PLL REGISTERS

Table 23.

IPD	LFB	PLLMULT	HCLKD	HCLK	JCLK
0	0	N	0	N * MCLK	N * MCLK
0	0	N	1	N * MCLK / 2	N * MCLK
0	1	N	0	2 * N * MCLK	2 * N * MCLK
0	1	N	1	N * MCLK	2 * N * MCLK
1	0	N	0	N * MCLK / 2	N * MCLK / 2
1	0	N	1	N * MCLK / 4	N * MCLK / 2
1	1	N	0	N * MCLK	N * MCLK
1	1	N	1	N * MCLK / 2	N * MCLK

PLL Rules For Various Speed Grades

The ADV212 uses the PLL_HI and PLL_LO direct registers to configure the PLL. Any time the PLL_LO register is modified, the host must wait at least 20 µs before reading or writing any other register. If this delay is not implemented, erratic behavior can result.

The PLL can be programmed to have any possible final multiplier value as long as the following are true:

- JCLK > 50 MHz and < 150 MHz (144-pin version)
- JCLK > 50 MHz and < 115 MHz (121-pin version)
- HCLK < 108 MHz (144-pin, 150MHz version)
- HCLK < 81 MHz (121-pin version)
- JCLK ≥ 2 × VCLK for single-component input
- JCLK ≥ 2 × VCLK for YCbCr [4:2:2] input
- In JDATA mode (JDATA), JCLK must be 4 × MCLK or higher.
- The maximum burst frequency for external DMA modes is ≤ 0.36 JCLK.
- For MCLK frequencies greater than 50 MHz, the input clock divider must be enabled, that is, IPD set to 1
- IPD cannot be enabled for MCLK frequencies below 20 MHz.
- De-Interlace modes require JCLK >= 4x MCLK
- It is not recommended to use an LLC output from a video decoder as a clock source for MCLK

For example, to achieve the lowest power consumption, an MCLK frequency of 27 MHz is recommended for a standard definition CCIR656 input. The PLL circuit is recommended to have a multiplier of 3. This sets JCLK and HCLK to 81 MHz.

RECOMMENDED VALUES FOR PLL_HI AND PLL_LO REGISTERS

Table 24.

VIDEO STANDARD	CLKIN FREQUENCY ON MCLK	PLL_HI	PLL_LO	JCLK
SMPTE125M or ITU-R.BT656 [NTSC or PAL]	27 MHz	0x0008	0x0004	108 MHz
SMPTE293M [525p]	27 MHz	0x0008	0x0004	108 MHz
ITU-R.BT1358 [625p]	27 MHz	0x0008	0x0004	108 MHz
SMPTE274M [1080i]	74.25 MHz	0x0008	0x0084	149 MHz

Internal Hardware Registers

The indirect registers are accessed by the host processor through the IADDR and IDATA direct registers. If a 16-bit host is used, the STAGE direct register is also used. See Figure 13 on page 27. A common indirect access by the external host processor is to write indirect registers, such as the EDMOD and FIFO threshold registers.

In certain modes, such as Custom Specific or HIPI mode, indirect registers are accessed by the host. In standard modes, many of the indirect registers are controlled by the internal ADV212 processor. In these standard modes, the host processor simply needs to configure the indirect DMA or FIFO threshold registers.

Both 32-bit and 16-bit hosts can access the indirect registers. 32-bit access use the IADDR and IDATA registers while 16-bit hosts use IADDR, IDATA, and the STAGE register. Additional information on accessing and configuring these registers can be found in the *Getting Started with the ADV212* programming guide.

If the CTRLREGAM bit in the MMODE register is set (1), the user only needs to load the IADDR register with the lower 16 bits address [15..0]. The upper address bits default to 0xFFFF. The can be advantageous for 16-bit hosts. If the CTRLREGAM bit is cleared (0), a full 32-bit address must be written to the IADDR register.

INTERNAL HARDWARE REGISTERS

Table 25.

INDIRECT REGISTER ADDRESS	NAME	DESCRIPTION
0xFFFF0400	PMODE1	Pixel/Video Format
0xFFFF0404	COMP_CNT_STATUS	Component count
0xFFFF0408	LINE_CNT_STATUS	Line count
0xFFFF040C	XTOT	Total Samples per line
0xFFFF0410	YTOT	Total Lines per frame
0xFFFF0414	F0_START	Start Line of Field 0 [F0]
0xFFFF0418	F1_START	Start Line of Field 1 [F1]
0xFFFF041C	V0_START	Start of active video Field 0 [F0]
0xFFFF0420	V1_START	Start of active video Field 1 [F1]
0xFFFF0424	V0_END	End of active video Field 0 [F0]
0xFFFF0428	V1_END	End of active video Field 1 [F1]
0xFFFF042C	PIXEL_START	Horizontal start of active video
0xFFFF0430	PIXEL_END	Horizontal end of active video
0xFFFF0440	MS_CNT_DEL	Master/Slave Delay
0xFFFF0444	Reserved	Reserved
0xFFFF0448	PMODE2	Pixel Mode 2
0xFFFF044C	VMODE	Video Mode
0xFFFF0450	V0_REGION_ST	Start of vertical cropped region Field 0
0xFFFF0454	V1_REGION_ST	Start of vertical cropped region Field 1
0xFFFF0458	V0_REGION_END	End of vertical cropped region Field 0
0xFFFF045C	V1_REGION_END	End of vertical cropped region Field 1
0xFFFF0460	PIXEL_START_REF	Pixel start reference
0xFFFF0464	PIXEL_END_REF	Pixel end reference
0xFFFF1408	EDMOD0	External DMA mode register 0
0xFFFF140C	EDMOD1	External DMA mode register 1
0xFFFF1410	FFTHRP	FIFO Threshold for PIXEL FIFO
0xFFFF1414	Reserved	Reserved
0xFFFF1418	Reserved	Reserved
0xFFFF141C	FFTHRC	FIFO Threshold for CODE FIFO
0xFFFF1420	FFTHRA	FIFO Threshold for ATTR FIFO

0xFFFF1424	Reserved	Reserved
0xFFFF1428	Reserved	Reserved
0xFFFF142C	Reserved	Reserved
0xFFFF1430	Reserved	Reserved
0xFFFF1434 - 0xFFFF14F3	Reserved	Reserved
0xFFFF14F4	HWREV	Hardware Revision Register

Indirect Registers: Configuration and Status

These indirect configuration and status registers are typically controlled by the ADV212, except when in a custom specific video input mode or HIPI mode. In standard video input modes, they do not need to be programmed by the external host processor except for EDMOD and the FIFO threshold registers to configure compressed data transfer to/from the ADV212.

PMODE1 REGISTER

Table 26.

Address	Name	Description	R/W	Reset Value
0xFFFF0400	PMODE1	Pixel Mode 1	R/W	See bit field descriptions.

The PMODE1 register configures the VDATA or HDATA bus for a specific pixel format. See “Video Input Formats” on page 69 for a thorough explanation of the types of video formats used on the VDATA bus and HDATA bus.

PMODE1 REGISTER BIT FIELD DESCRIPTIONS

Table 27.

Bit	Name	Description	Reset Value
4..0	PFMT	VDATA 0x00 - 0x03 Reserved 0x04 Single Component (8, 10, or 12 bit) 0x05 Cb/Y/Cr/Y interleaved (8, 10, or 12 bit) 0x06 Reserved 0x07 Cb/Cr interleaved (8, 10, or 12 bit)	0x05
		HDATA 0x14 32-bit 4x8-bit Single Component 0x15 32-bit 4x8-bit Packed YCbYCr 0x16 32-bit 4x8-bit Packed YYCbCr 0x17 32-bit 4x8-bit Packed CbCrCbCr 0x18 32-bit 2x16-bit Packed Single Component 0x19 32-bit 2x16-bit Packed YCb/YCr 0x1A 32-bit 2x16-bit Packed YY/CbCr 0x1B 32-bit 2x16-bit Packed CbCr 0x1F -0x1C Reserved	
7..5	Reserved	Always write 0	0
10..8	PREC	0x0 8-bit precision 0x1 10-bit precision 0x2 12-bit precision 0x3 14-bit precision 0x4 16-bit precision 0x05 - 0x7 Reserved	0x01
15..11	Reserved	Always write 0	0

SAMPLE COUNT STATUS REGISTER

Table 28.

Address	Name	Description	R/W	Reset Value
0xFFFF0404	COMP_CNT_STATUS	Sample Count	R	0x0000

The 16-bit COMP_CNT_STATUS register indicates the current value of the pixel interface horizontal sample counter. This register should not be polled during applications.

LINE COUNT STATUS REGISTER

Table 29.

Address	Name	Description	R/W	Reset Value
0xFFFF0408	LINE_CNT_STAT	Line Count.	R	0x0000

The 16-bit LINE_CNT_STAT register indicates the current value of the pixel interface line counter. This register should not be polled during applications.

Indirect Registers: Video Timing and Dimension

This section describes the indirect registers that affect video timing and video dimension. For more detailed information, see "Video Modes" on page 65. These indirect configuration and status registers are typically controlled by the ADV212, except when in a custom specific video input mode or HIPI mode. In standard video input modes, they do not need to be programmed by the external host processor.

The ADV212 supports an enhanced cropping mode set in the VMODE register. With CROP_MODE enabled, a video window can be defined anywhere in the active field. For video encode modes only, data outside the crop window is discarded. For video decode modes, data outside the crop window is discarded and replaced with black field data. This mode only affects the actual video data. The original dimension registers are still used to define the frame size and associated synchronization.

XTOT DIMENSION REGISTER

Table 30.

Address	Name	Description	R/W	Reset Value
0xFFFF040C	XTOT	Total samples per line	R/W	0x064B

The 16-bit XTOT register is used to set total number of samples per line.
For monochrome input, the number of samples is equivalent to the number of pixels. For YCbCr video, the number of samples is twice the number of pixels per line.

YTOT DIMENSION REGISTER

Table 31.

Address	Name	Description	R/W	Reset Value
0xFFFF0410	YTOT	Total lines per frame	R/W	0x020D

The 16-bit YTOT register is used to set the total lines per frame.

F0_START REGISTER

Table 32.

Address	Name	Description	R/W	Reset Value
0xFFFF0414	F0_START	Start of Field 0. Only used in Decode mode to identify the line number at which the Field bit transitions from Field 1 to Field 0.	R/W	0x0004

The 16-bit F0_START register is used to indicate the start of Field 0 in units of number of lines. Line count starts with line 1. See Figure 15, "Video Dimension Attributes Used In Register Settings" on page 65.

F1_START REGISTER

Table 33.

Address	Name	Description	R/W	Reset Value
0xFFFF0418	F1_START	Start of field 1. Only used in Decode mode to identify the line number at which the Field bit transitions from Field 0 to Field 1.	R/W	0x010A

The 16-bit F1_START register is used to indicate the start of Field 1 in units of number of lines. Line count starts with line 1. See Figure 15, "Video Dimension Attributes Used In Register Settings" on page 65.

V0_START REGISTER

Table 34.

Address	Name	Description	R/W	Reset Value
0xFFFF041C	V0_START	Start of active video in field 0	R/W	0x0014

The 16-bit V0_START register is used to set the first active video line in Field 0 that will be captured by the ADV212. Line count starts with line 1. See Figure 15, "Video Dimension Attributes Used In Register Settings" on page 65.

V1_START REGISTER

Table 35.

Address	Name	Description	R/W	Reset Value
0xFFFF0420	V1_START	Start of active video in field 1	R/W	0x011B

The 16-bit V1_START register is used to set the first active video line in Field 1 that will be captured by the ADV212. Line count starts with line 1. See Figure 15, "Video Dimension Attributes Used In Register Settings" on page 65.

V0_END REGISTER

Table 36.

Address	Name	Description	R/W	Reset Value
0xFFFF0424	V0_END	End of active video in field 0	R/W	0x0107

The 16-bit V0_END register is used to set the vertical end of the active video in Field 0. This register should hold the value of the last active line number. Line count starts with line 1. See Figure 15, "Video Dimension Attributes Used In Register Settings" on page 65.

V1_END REGISTER

Table 37.

Address	Name	Description	R/W	Reset Value
0xFFFF0428	V1_END	End of active video in field 1	R/W	0x020D

The 16-bit V1_END register is used to set the vertical end of the active video in Field 1. This register should hold the value of the last active line number. Line count starts with line 1. See Figure 15, "Video Dimension Attributes Used In Register Settings" on page 65.

PIXEL_START REGISTER

Table 38.

Address	Name	Description	R/W	Reset Value
0xFFFF042C	PIXEL_START	Start of horizontal active video	R/W	0x0001

The 16-bit PIXEL_START register is used to set the horizontal start (that is, the first active sample in the line) of the active video in units of samples (horizontal counter starts at sample 1). When VMODE[12] is set to 1 and cropping mode is enabled, this register sets the horizontal start of the cropped area. See Figure 15, "Video Dimension Attributes Used In Register Settings" on page 65.

PIXEL_END REGISTER

Table 39.

Address	Name	Description	R/W	Reset Value
0xFFFF0430	PIXEL_END	End of horizontal active video	R/W	0x05A0

The 16-bit PIXEL_END register is used to set the horizontal end (that is, the last active sample in the line) of the active video in units of samples (horizontal counter starts at sample 1). When VMODE[12] is set to 1 and cropping mode is enabled, this register sets the horizontal end of the cropped area. See Figure 15, "Video Dimension Attributes Used In Register Settings" on page 65.

MS_CNT_DEL REGISTER

Table 40.

Address	Name	Description	R/W	Reset Value
0xFFFF0440	MS_CNT_DEL	Multi-chip control	R	0

The 16-bit MS_CNT_DEL register is used only in multi-chip sync mode. See the Application Note "ADV212 multi-chip application" for additional information.

PIXEL MODE 2 REGISTER

Table 41.

Address	Name	Description	R/W	Reset Value
0xFFFF0448	PMODE2	Pixel Mode 2	R/W	0

The 16-bit PMODE2 register is mainly used to set up the programmable edge control for the input clock and sync signals.

PIXEL MODE 2 REGISTER BIT FIELD DESCRIPTIONS

Table 42.

Bit	Name	Description	Reset Value
0	VCLK_POL	VCLK active edge 1=rising, 0=falling	1
1	VSYNC_POL	VSYNC active edge 1=rising, 0=falling	0
2	HSYNC_POL	HSYNC active edge 1=rising, 0=falling	0
3	FIELD_POL	FIELD active edge 1=rising, 0=falling	0
4	YUNI	Y Unipolar 1=on, 0=off This bit should be set to '1' for most applications.	0
5	CUNI	C Unipolar 1=on, 0=off This bit should be set to '1' for most applications.	0
6	Reserved	Always write 0	0
7	Reserved	Always write 0	0
8	Reserved	Always write 0	0
9	Reserved	Always write 0	0
10	Reserved	Always write 0	0
11	Reserved	Always write 0	0
12	Reserved	Always write 0	0
13	Reserved	Always write 0	0
14	Reserved	Always write 0	0
15	Reserved	Always write 0	0

VIDEO MODE REGISTER

Table 43.

Address	Name	Description	R/W	Reset Value
0xFFFF044C	VMODE	Video Mode	R/W	See bit field descriptions.

The 16-bit VMODE register is controlled internally (except in custom-specific mode or HIPI mode). It sets the pixel interface basic operating mode.

VIDEO MODE REGISTER BIT FIELD DESCRIPTIONS

Table 44.

Bit	Name	Description	Reset Value
0	MAS_SLV	Master or Slave operation is only selectable in Decode mode. 1 = MASTER 0 = SLAVE	0
1	ENC_DEC	Sets ENCODE or DECODE mode 1 = ENCODE 0 = DECODE	1
2	MP_656	Video input timing control. 0 = EAV/SAV mode = 0; for video input with embedded timing codes 1= HVF mode = 1; for video input with separate H,V,F sync signals.	0
3	Reserved	Reserved	0
4	HOST_MODE	Pixel data over HDATA bus. 1 = on 0 = off	0
5	RAW_MODE	Raw video input on the VDATA bus. HVF mode is not required 1 = on 0 = off	0
6	Reserved	Always write 0.	0
7	PRGRSV_SCN	Progressive scan mode. Must be set if video input on the VDATA bus is non-interlaced. 1 = on 0 = off	0
8	Reserved	Always write 0.	0
9	Reserved	Always write 0.	0
10	Reserved	Always write 0.	0
11	CNT_RD_EN	Read PI control counters (CO_CNT_STATUS and LINE_CNT_STATUS) 1 = on 0 = off	0
12	CROP_MODE	Crop mode enable: 1 = on 0 = off	0
15:13	Reserved	Always write 0	0

When VMODE[12], CROP_MODE, is disabled, the registers used for crop mode default to transparent values. The PIXEL_START and PIXEL_END registers define the horizontal active region, and the V0_START, V0_END, V1_START, and V1_END registers define the vertical active region.

When VMODE[12], CROP_MODE, is enabled, you use the V0_START, V0_END, V1_START, and V1_END registers to define the vertical reference active field and the V0_REGION_ST, V0_REGION_END, V1_ REGION_ST, and V1_ REGION_END registers to define the cropped area within the active field. The PIXEL_START_REF and PIXEL_END_REF registers define the horizontal reference active field while the PIXEL_START and PIXEL_END registers define the cropped area within the active field.

The EAV/SAV time codes and HVF syncs are based on the reference active field registers, not the cropped window settings.

For video decode modes, the cropped area can be anywhere within the reference active field. All pixel data outside the cropped area is set to black sample data.

For progressive modes, this mode ignores Field 1 registers; they can be left to their default values.

VERTICAL ACTIVE REGION 0 START REGISTER

Table 45.

Address	Name	Description	R/W	Reset Value
0xFFFF0450	V0_REGION_ST	Start of new vertical active region (Field 0)	R/W	0x0000

The 16-bit V0_REGION_ST register is used to define the vertical cropped area. Line count starts with line 1. See Figure 15, "Video Dimension Attributes Used In Register Settings" on page 65.

VERTICAL ACTIVE REGION 1 START REGISTER

Table 46.

Address	Name	Description	R/W	Reset Value
0xFFFF0454	V1_REGION_ST	Start of new vertical active region (Field 1)	R/W	0x0000

The 16-bit V1_REGION_ST register is used to define the vertical cropped area. Line count starts with line 1. See Figure 15, "Video Dimension Attributes Used In Register Settings" on page 65.

VERTICAL ACTIVE REGION 0 END REGISTER

Table 47.

Address	Name	Description	R/W	Reset Value
0xFFFF0458	V0_REGION_END	End of new vertical active region (Field 0)	R/W	0xFFFF

The 16-bit V0_REGION_END register is used to define the vertical cropped area. See Figure 15, "Video Dimension Attributes Used In Register Settings" on page 65.

VERTICAL ACTIVE REGION 1 END REGISTER

Table 48.

Address	Name	Description	R/W	Reset Value
0xFFFF045C	V1_REGION_END	End of new vertical active region (Field 1)	R/W	0xFFFF

The 16-bit V1_REGION_END register is used to define the vertical cropped area. See Figure 15, "Video Dimension Attributes Used In Register Settings" on page 65.

PIXEL START REFERENCE REGISTER

Table 49.

Address	Name	Description	R/W	Reset Value
0xFFFF0460	PIXEL_START_REF	Pixel start reference	R/W	PIXEL_START

The 16-bit PIXEL_START_REF register is used to set the horizontal start of the reference area. See Figure 15, "Video Dimension Attributes Used In Register Settings" on page 65.

PIXEL END REFERENCE REGISTER

Table 50.

Address	Name	Description	R/W	Reset Value
0xFFFF0464	PIXEL_END_REF	Pixel end reference	R/W	PIXEL_END

The 16-bit PIXEL_END_REF register is used to set the horizontal end of the reference area. See Figure 15, "Video Dimension Attributes Used In Register Settings" on page 65.

Indirect Registers: External DMA

To use the external DMA channels for compressed data accesses, the EDMOD0 and EDMOD1 registers must be programmed by the host processor.

EXTERNAL DMA MODE 0 REGISTER

Table 51.

Address	Name	Description	R/W	Reset Value
0xFFFF1408	EDMOD0	External DMA Mode	R/W	See bit field descriptions.

The 16-bit EDMOD0 register is used to assign a data FIFO to Channel 0 and to select transfer modes.

EXTERNAL DMA MODE 0 REGISTER BIT FIELD DESCRIPTIONS

Table 52.

Bit	Name	Description	Reset Value
0	DMEN0	Enable external DMA channel 0 0=disabled 1=enabled	0
2..1	DMSEL0	DMA Channel 0 Assignment 00 = Pixel Data 01 = Compressed Data 10 = Attribute Data 11 = Reserved	‘00’
5..3	DMMOD0	DMA Channel 0 mode 000 - Dedicated Chip Select DMA mode 001 - Single transfer DREQ/DACK DMA mode 010 - Burst transfer DREQ/DACK DMA mode 011 - JDATA mode 100 - Reserved 101 - Single transfer Fly-by DMA mode 110 - Burst transfer Fly-by DMA mode 111 - Reserved	‘000’
8..6	DMBL0	DMA Channel 0 burst length [in number of accesses] 000 = 8 - not available for DWIDTH of 8-bit 001 = 16 - for DWIDTH of 8/16 or 32-bits 010 = 32 - for DWIDTH of 8/16 or 32-bits 011 = 64 - for DWIDTH of 8/16 or 32-bits 100 = 128 - for DWIDTH of 8/16 or 32-bits 110 = 512 - for DWIDTH of 8/16-bit, not recommended for 32-bits, 111 - 1024 - for DWIDTH of 8-bit; not recommended for 16-bit host, not available for 32-bit hosts.	‘000’
9	DR0POL	$\overline{\text{DREQ0}}$ $\overline{\text{FSRQ0}}$ VALID polarity 0 = Active Low 1 = Active High	0
10	DA0POL	$\overline{\text{DACK0}}$ $\overline{\text{FCS0}}$ HOLD polarity 0 = Active Low 1 = Active High	0
14..11	DR0PULS	$\overline{\text{DREQ0}}$ pulse width [in JCLK cycles] If DR0PULS is set to ‘0000’, then $\overline{\text{DREQ0}}$ remains asserted until $\overline{\text{DACK0}}$ and the proper strobe ($\overline{\text{RD}}$or $\overline{\text{WE}}$) are asserted.	‘0001’
15	Reserved	Reserved for future use. Always write 0.	0

EXTERNAL DMA MODE 1 REGISTER

Table 53.

Address	Name	Description	R/W	Reset Value
0xFFFF140C	EDMOD1	External DMA Mode	R/W	See bit field descriptions.

The 16-bit EDMOD1 register is used to assign a data FIFO to Channel 1 and to select transfer modes.

ⓘ JDATA mode is not available in EDMOD1.

EXTERNAL DMA MODE 1 REGISTER BIT FIELD DESCRIPTIONS

Table 54.

Bit	Name	Description	Reset Value
0	DMEN1	Enable external DMA channel 1 0=disabled 1=enabled	0
2..1	DMSEL1	DMA Channel 1 assignment 00 = Pixel Data 01 = Compressed Data 10 = Attribute Data 11 = Reserved	‘01’
5..3	DMMOD1	DMA Channel 1 mode 000 - Dedicated Chip Select DMA mode 001 - Single transfer DREQ/DACK DMA mode 010 - Burst transfer DREQ/DACK DMA mode 011 - Reserved 100 - Reserved 101 - Single transfer Fly-by DMA mode 110 - Burst transfer Fly-by DMA mode 111 - Reserved	‘000’
7..6	DMBL1	DMA Channel 1 burst length [in number of accesses] 000 = 8 - not available for DWIDTH of 8-bit 001 = 16 - for DWIDTH of 8/16 or 32-bits 010 = 32 - for DWIDTH of 8/16 or 32-bits 011 = 64 - for DWIDTH of 8/16 or 32-bits 100 = 128 - for DWIDTH of 8/16 or 32-bits 110 = 512 - for DWIDTH of 8/16-bit, for 32-bits, not recommended 111 - 1024 - for DWIDTH of 8-bit, not recommended for 16-bit hosts, not available for 32-bit hosts	‘00’
9	DR1POL	$\overline{\text{DREQ1}}$ $\overline{\text{FSRQ1}}$ polarity 0 = Active Low 1 = Active High	0
10	DA1POL	$\overline{\text{DACK1}}$ $\overline{\text{FCS1}}$ polarity 0 = Active Low 1 = Active High	0
14..11	DR1PULS	$\overline{\text{DREQ1}}$ pulse width [in JCLK cycles]. If DR1PULS is set to ‘0000’, then $\overline{\text{DREQ1}}$ remains asserted until $\overline{\text{DACK1}}$ and the proper strobe ($\overline{\text{RD}}$or $\overline{\text{WE}}$) are asserted.	‘0001’
15	Reserved	Reserved for future use. Always write 0.	0

EDMOD Register Descriptions

EDMOD0 and EDMOD1 registers configure the ADV212 for external DMA operation. External DMA provides two channels for fast, low-overhead data transfers between the host processor and the and the ADV212 data FIFOs. The use of external DMA allows for much more efficient transfers than the use of normal host mode accesses.

The external DMA registers include settings for Single/Burst DMA, Fly-By Mode DMA, JDATA mode (EDMOD0 only), and Dedicated Chip Select (DCS) DMA. These registers are used in conjunction with the Bus Mode (BUSMODE) and MMODE registers.

External DMA Width

The desired data width when accessing data FIFOs is set in the BUSMODE register. The DWIDTH field in this register (direct address location 0x08) can be programmed to byte, 16-bit , or word (32-bit) data widths. The default data-width is 16-bit.

When accessing the FIFOs through the Direct Registers using normal host modes, the HWIDTH parameter in the BUSMODE register sets the data width. DWIDTH is used to set the data width when DMA modes are used

DMA Mode - DREQ/DACK DMA Mode and FLY-BY DMA Mode

The EDMOD registers are used to enable the External DMA interface, assign the desired FIFO to each channel, and select modes of operation. All DMA modes make use of the Data Request (DREQ0/DREQ1) and Data Acknowledge (DACK0/DACK1) pins.

DREQ/DACK DMA mode and Fly-By DMA mode can be used in single transfer or burst transfer modes.

A programming example for DREQ/DACK DMA mode can be found in the *Getting Started with the ADV212* programming guide.

DMA Mode - Dedicated Chip Select (DCS) DMA Mode

This DMA mode is used with the FIFO threshold registers (FFTHRP, FFTHRC, FFTHRA). The FSRQx (DREQx pins) signals are asserted when the number of words or spaces programmed in the threshold registers is available in the selected FIFO. The data is transferred over the HDATA bus using the FCSx (DACKx pins) and RD and WE signals. The FSRQx signals are deasserted when the number of words or spaces in the selected FIFO is less than the threshold for the corresponding FIFO.

A programming example for DCS mode can be found in the *Getting Started with the ADV212* programming guide.

JDATA - DATA Mode

This mode allows streaming input/output of compressed data using the VALID/HOLD protocol and the JDATA[7..0] pins. When JDATA mode is enabled, 32-bit data and host modes are not available. To assign HDATA [31..24] pins to JDATA[7..0], the BUSMODE register must first be programmed to JDATA. The polarity of the VALID (DREQ0 pin) and HOLD (DACK0 pin) is set in the EDMOD0 register through bits DR0POL/DA0POL. The DMSEL0 field in EDMOD0 must be set to "1" (CODE FIFO). The DMMOD0 field must be set to "3" (JDATA Mode). It is recommended that the user set up all mode bits in EDMOD0 without enabling the interface (DMEN0). Then, the HOLD pin should be asserted [HOLD should be in its active state], and then de-asserted [de-activated] after the JDATA interface has been enabled by the host. Data transfers occur when the ADV212 asserts VALID and the external device de-asserts HOLD.

The VALID signal always functions as an output from the ADV212 and the HOLD signal always functions as an input to the ADV212. Data transfers occur when VALID is asserted and HOLD is de-asserted at the rising edge of MCLK.

When JDATA mode is used, the Compressed Data (CODE) FIFO is dedicated to DMA Channel 0. The other data FIFOs (PIXEL or ATTR) are not available when JDATA mode is used. Attempting to access these FIFOs will interfere with JDATA operation.

JDATA is referenced with MCLK and requires a PLL_MULT setting of
JCLK ≥ 4 * MCLK.

BUSMODE/DWIDTH should be set to 8-bit. BUSMODE/HWIDTH should be set to 16-bit.

A programming example for JDATA mode can be found in the *Getting Started with the ADV212* programming guide.

Single Transfer - DREQ/DACK DMA

In this mode, the ADV212 will assert DREQx when there is at least one data transfer available for the assigned FIFO. In encode mode, the DREQx assertion indicates that output data is present in the FIFO. In decode mode, this indicates that space is available in the FIFO for at least one data transfer. The external device must respond with DACKx, in conjunction with RD/WE/HDATA activity. In Single Transfer Fly-By Mode, the functionality of the RD and WE pins is reversed.

Burst Transfer - DREQ/DACK DMA

In this mode, the ADV212 will assert DREQx when there is at least one burst data transfer available for the assigned FIFO. In encode mode, the DREQx assertion indicates that output data is present in the FIFO. In decode mode, this indicates that space is available in the FIFO for at least one burst data transfer. The external device must respond with DACKx, in conjunction with RD/WE/HDATA activity.

In Burst Transfer Fly-By Mode, the functionality of the RD and WE pins is reversed.

Burst Length

FIFO width is always 32-bit words, but the programmed Burst Lengths correspond to the number of accesses (8-, 16-, or 32- bits wide). A given FIFO depth can support the same number of word transfers, twice the number of 16-bit ‘word’ transfers, and four times the number of byte transfers.

For example, Burst Length (EDMODx/DMBLx) can be programmed to up to 64 word transfers or 128 16-bit word transfers.

If the assigned FIFO does not contain the programmed number of accesses (that is, words) for the last portion of the compressed field, the FIFO is padded with zeroes to ensure that the field ends on a burst boundary.

Indirect Registers: FIFO Threshold

This section describes the ADV212 FIFO threshold registers. For additional details on how to use FIFO accesses, see “Indirect Registers: External DMA” on page 51.

The FIFO threshold registers hold the values that are compared to the corresponding pixel (PIXEL), code (CODE), and attribute (ATTR) FIFO count registers in certain transfer modes. The values in the threshold registers are programmed by the host and represent the number of words or spaces that trigger a FIFO service request. (A space is a FIFO slot available for a word entry.)

When the ADV212 is in encode mode, a FIFO service request is triggered when the number of words in the CODE FIFO or ATTR FIFO is greater than or equal to the corresponding threshold register value. If the PIXEL FIFO is used in encode mode (HIPI mode), a FIFO service request is triggered when the number of spaces indicated by the PIXEL FIFO count register is greater than or equal to the value in the PIXEL FIFO threshold register.

When the ADV212 is in decode mode, a FIFO service request is triggered when the number of spaces in the CODE FIFO or ATTR FIFO is greater than or equal to the corresponding threshold register value. If the PIXEL FIFO is used in decode mode (HIPI mode), a FIFO service request is triggered when the number of words indicated by the PIXEL FIFO count register is greater than or equal to the value in the PIXEL FIFO threshold register.

FIFO service requests can be used when code, attribute, or pixel (in HIPI mode) transfers are performed in one of the following methods:

- Dedicated chip select (DCS) DMA mode

  External DMA mode registers (EDMOD0/EDMOD1) are programmed to DCS DMA mode (see page 55 for details). The FIFO service request ($\overline{\text{FSRQ0}}$ and $\overline{\text{FSRQ1}}$) signals on the DREQ0 and DREQ1 pins are asserted when the FIFO threshold condition for the programmed FIFO is met. The host responds by using the FIFO select ($\overline{\text{FCS0}}$ and $\overline{\text{FCS1}}$) signals on the DACK0 and DACK1 pins and a pulse on $\overline{\text{RD}}$ or $\overline{\text{WE}}$ for the corresponding DMA channel to service the FIFO.

- External interrupt request signal

  The $\overline{\text{IRQ}}$ pin is asserted when a FIFO service request is triggered and the external interrupt request enable register (EIRQIE) is programmed so that the corresponding FIFO threshold interrupt is enabled. To initialize the FIFO service request, the host programs the FIFO threshold registers, enables FIFO threshold conditions by setting the PFTH, DFTH, or AFTH (PIXEL, CODE, or ATTR FIFO threshold-enable) bits in the EIRQIE register, and then clears the corresponding bits in the external interrupt flag (EIRQFLG) register.

  When the host detects $\overline{\text{IRQ}}$ pin assertion, the FIFO is serviced and the EIRQFLG bits are cleared. If more than one FIFO has been enabled by the host, the host first reads the EIRQFLG register to determine which FIFO requires servicing. The EIRQFLG FIFO threshold conditions are cleared by the host after FIFO servicing is completed.

- Polling External Interrupt FIFO Threshold Flags

  This method is used when the EIRQIE register is not programmed to allow EIRQFLG interrupt conditions to be indicated on the $\overline{\text{IRQ}}$ pin. To initialize the FIFO service request, the host programs the FIFO threshold registers and then clears the corresponding bits in the External Interrupt Flag (EIRQFLG) register. The host can now poll the EIRQFLG register and service the corresponding FIFO(s) when a threshold condition is detected. The EIRQFLG FIFO threshold conditions are cleared by the host after FIFO servicing is completed.

PIXEL FIFO THRESHOLD REGISTER

Table 55.

Address	Name	Description	R/W	Reset Value
0xFFFF1410	FFTHRP	FIFO Threshold for PIXEL FIFO Mode	R/W	0x0000

The 16-bit FFTHRP register sets the threshold for the PIXEL data FIFO. The FFTHRP register can be set to a maximum of 256 32-bit words.

CODE FIFO THRESHOLD REGISTER

Table 56.

Address	Name	Description	R/W	Reset Value
0xFFFF141C	FFTHRC	FIFO Threshold for CODE FIFO Mode	R/W	0x0000

The 16-bit FFTHRC register sets the threshold for the CODE data FIFO. The FFTHRC register can be set to a maximum of 512 32-bit words.

ATTR FIFO THRESHOLD REGISTER

Table 57.

Address	Name	Description	R/W	Reset Value
0xFFFF1420	FFTHRA	FIFO Threshold for ATTR FIFO	R/W	0x0000

The 16-bit FFTHRA register sets the threshold for the ATTR data FIFO. The maximum threshold value for the ATTR FIFO is 256 words. The ATTR FIFO is only used for specialized applications.

Chip Identification

The chip ID is a 32-bit value where bits [31:16] contain the part number and bits [15:0] contain the hardware revision register. You can read the entire chip identification by accessing JTAG scan chain #5. a 32-bit little-endian scan chain.

You can read the part number from address 0xFFFF0254, and you can read the hardware revision register from 0xFFFF0250 or 0xFFFF14F4.

CHIP ID

Table 58.

Bit	Address	Description
31..16	0xFFFF0254	Part number: 0x27DB
15..0	0xFFFF0250 or 0xFFFF14F4	Hardware revision register; see the bit descriptions in Table 59.

HARDWARE REVISION REGISTER

Table 59.

Bit	Description
15..8	Hardware revision
7..1	Software revision
0	Marker (always 0x1)

This 32-bit indirect register describes the ADV212 hardware revision of the chip. The hardware revision value is independent from the revision of the firmware implemented in the ROM.

Host 32- and 16-bit Accesses to Direct/Indirect Registers and Indirect Memory

The ADV212 is versatile in that it can support both 32 and 16-bit host interface sizes. The following tables provide some examples of common bus interface configurations for 32- and 16-bit host processor.

The Direct Registers are accessed by the physical host interface using pins ADDR [3..0], HDATA[31..0], $\overline{\text{CS}}$, $\overline{\text{RD}}$, $\overline{\text{WE}}$, and $\overline{\text{ACK}}$ pins.

The Indirect Registers are accessed by writing and reading the IADDR register and the IDATA register. The IADDR register acts as a pointer to the internal indirect registers and memory. If a 16-bit host is used, the STAGE direct register is also used.

Accesses Using A 32-bit Host

The table below shows the common register settings for accessing direct, indirect register and indirect memory for a 32-bit host. All accesses to indirect registers and indirect memory are done using the IADDR and IDATA direct registers.

32-BIT HOST INTERFACE SETTINGS

Table 60.

MEMORY	LOCATION	BUSMODE REGISTER	MMODE REGISTER	EXAMPLES

32-bit Direct Register	0x00 – 0x0F	DWIDTH = 32 bits
		HWIDTH = 32 bits
			IWIDTH = 32 bits
			IAUTOSTP = 32 bits
				Example: Write 0xFFFF1408 to the IADDR register, then read the IADDR register.
				
				Sequence:
				1) Write 0xFFFF1408 to the IADDR register
				2) Read IADDR register (value should be 0xFFFF1408)

16-bit Direct Register	0x00 – 0x0F	DWIDTH = 32 bits
		HWIDTH = 32 bits
			IWIDTH = 32 bits
			IAUTOSTP = 32 bits
				Example: Write 0x000A to the MMODE register
				
				Sequence:
				1) Write 0x0000000A to the MMODE register
				
				Note: (HDATA[31..16]) bits are ignored

16-bit Indirect Register	0xFFFF0000 – 0xFFFF14C0	DWIDTH = 32 bits
		HWIDTH = 32 bits
			IWIDTH = 32 bits
			IAUTOSTP = 32 bits
				Example: Write 0x0802 to the EDMOD0 register (0xFFFF1408)
				
				Sequence:
				1) Write 0xFFFF1408 to the IADDR register
				2) Write 0x08020000 to the IDATA register
				
				Example: Read the EDMOD0 register (0xFFFF1408)
				
				Sequence:
				1) Write 0xFFFF1408 to the IADDR register
				2) Read the IDATA register (value should be 08020000 from previous write)
				
				Note: (HDATA[15..0]) bits are ignored on Indirect writes.
				
				Indirect Registers are 16-bits aligned on 32-bit boundaries utilizing the [31.15] byte lanes.

32-bit Indirect Memory	0x00057000 – 0x00057FF	DWIDTH = 32 bits
		HWIDTH = 32 bits
			IWIDTH = 32 bits
			IAUTOSTP = 32 bits
				Example: Write 0x12345678 to memory location 0x00057F00
				
				Sequence:
				1) Write 0x00057F00 to the IADDR register
				2) Write 0x12345678 to the IDATA register
				
				Example: Read memory location 0x00057F00.
				Sequence:
				1) Write 0x00057F00 to the IADDR register
				2) Read the IDATA register (value should be 0x12345678 from previous write)

Accesses Using a 16-bit Host

The table below shows the common register settings for accessing direct, indirect register and indirect memory for a 16-bit host.

ⓘ MMODE register settings: Accesses to Indirect Registers and Indirect memory will change depending on the setting of the IWIDTH and IAUTOSTP bits. The IAUTOSTP bits in MMODE can be set for autoincrement or autodecrement of 32-, 16-, or 8-bit. This decreases the host overhead by automatically incrementing the address pointer. The host can simply continue sequential address writes and reads until finished.

16-BIT HOST INTERFACE SETTINGS

Table 61.

MEMORY	LOCATION	BUSMODE REGISTER	MMODE REGISTER	EXAMPLES

32-bit Direct Register	0x00 – 0x0F	DWIDTH = 16 bits
		HWIDTH = 16 bits
			IWIDTH = 16 bits
			IAUTOSTP = 16 bits
				Example: Write 0xFFFF1408 to the IADDR register
				
				Sequence:
				1) Write 0xFFFF to the STAGE register.
				2) Write 0x1408 to the IADDR register.
				
				Example: Read the value written to the IADDR register
				
				Sequence:
				1) Read the IADDR register (value should be 0xFFFF)
				2) Read the STAGE register (value should be 0x1408)

16-bit Direct Register	0x00 – 0x0F	DWIDTH = 16 bits
		HWIDTH = 16 bits
			IWIDTH = 16 bits
			IAUTOSTP = 16 bits
				Example: Write 0x000A to the MMODE register
				
				Sequence:
				1) Write 0x000A to the MMODE register

16-bit Indirect Register	0xFFFF0000 – 0xFFFF14C0	DWIDTH = 16 bits
		HWIDTH = 16 bits
			IWIDTH = 16 bits
			IAUTOSTP = 32 bits
				Example: Write 0x0802 to the EDMOD0 register (0xFFFF1408)
				
				Sequence:
				1) Write 0xFFFF to the STAGE register.
				2) Write 0x1408 to the IADDR register.
				3) Write 0x0802 to the IDATA register.
				
				Example: Read the value written to the EDMOD0 register (0xFFFF1408)
				
				Sequence:
				1) Write 0xFFFF to the STAGE register.
				2) Write 0x1408 to the IADDR register.
				3) Read the IDATA register (value should be 0x0802)
				
				Note: Since Indirect Registers are on 32-bit boundaries, the IAUTOSTP bits should be set for 32-bit increments.

32-bit Indirect Memory	0x00057000 – 0x00057FFF	DWIDTH = 16 bits
		HWIDTH = 16 bits
			IWIDTH = 16 bits
			IAUTOSTP = 16 bits
				Example: Write 0x12345678 to memory location 0x00057F00
				
				Sequence:
				1) Write 0x0005 to the STAGE register.
				2) Write 0x7F00 to the IADDR register.
				3) Write 0x1234 to the IDATA register.
				4) Write 0x5678 to the IDATA register.
				
				Example: Read the value written to memory location 0x00057F00
				
				Sequence:
				1) Write 0x0005 to the STAGE register.

				2) Write 0x7F00 to the IADDR register. 3) Read the IDATA register.(value should be 0x5678) 4) Read the IDATA register.(value should be 0x1234) Note: In this mode, the read back value is swapped. The written order is Big-Endian, the read order is Little Endian.

32-bit Indirect Memory	0x00057000 – 0x00057FFF	DWIDTH = 16 bits
		HWIDTH = 16 bits
			IWIDTH = 32 bits
			IAUTOSTP = 32 bits
				Example: Write 0x12345678 to memory location 0x00057F00
				
				Sequence:
				1) Write 0x0005 to the STAGE register.
				2) Write 0x7F00 to the IADDR register.
				3) Write 0x1234 to the STAGE register.
				4) Write 0x5678 to the IDATA register.
				
				Continuous writes would be achieved with a write to the STAGE register, followed by a write to the IDATA register.
				
				Example: Read the value written to memory location 0x00057F00
				
				Sequence:
				1) Write 0x0005 to the STAGE register.
				2) Write 0x7F00 to the IADDR register.
				3) Read the IDATA register.(value should be 0x1234)
				4) Read the STAGE register.(value should be 0x5678)
				
				Note: By setting the MMODE register to 0x000A, the STAGE register is utilized for reads and writes instead of just the IDATA register only

Video Modes

This section describes the different modes used by the ADV212.

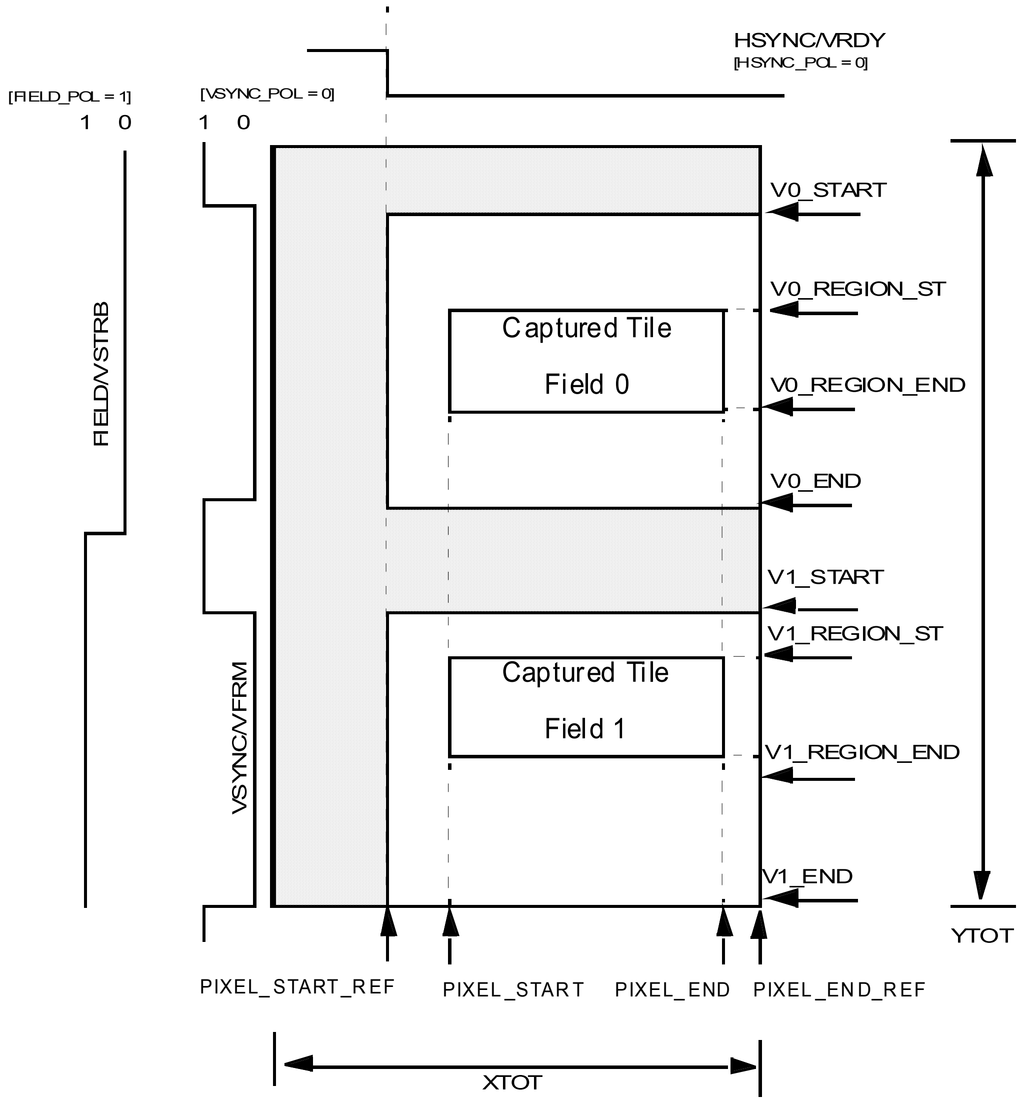

Figure 15. Video Dimension Attributes Used In Register Settings

Encode Mode

In encode mode, the ADV212 is always in slave configuration. The ADV212 synchronizes itself to the incoming sync signals or by the embedded sync codes in the video data stream. Input data can be accompanied by separate H,V, F signals or embedded timing codes. In both cases, the register settings reflect the video standard of the input.

When CROP_MODE is off, you set the following registers:

- XTOT
- YTOT
- V0_START
- V1_START
- V0_END
- V1_END
- PIXEL_START
- PIXEL_END

When CROP_MODE is on, you set the following registers:

- XTOT
- YTOT
- V0_START
- V0_REGION_ST
- V1_START
- V1_REGION_ST
- V0_END
- V0_REGION_END
- V1_END
- V1_REGION_END
- PIXEL_START
- PIXEL_START_REF
- PIXEL_END
- PIXEL_END_REF

You calculate the active video region to be processed using these registers. This does apply to all input modes using the VDATA bus; HIPI mode only requires programmed values for XTOT and YTOT while all other dimension register values are ignored. See the *"ADV212_HIPI_mode"* application note for more information.

Decode Slave Mode

In decode slave mode video output is synchronized to the incoming HVF signals. In both cases, the register settings reflect the video standard of the input. The part synchronizes itself to the incoming sync signals. Fields are identified by the incoming FIELD signal.

When CROP_MODE is off, you set the following registers:

- XTOT
- YTOT
- V0_START
- V1_START
- V0_END
- V1_END
- PIXEL_START
- PIXEL_END

When CROP_MODE is on, you set the following registers:

- XTOT
- YTOT
- V0_START
- V0_REGION_ST
- V1_START
- V1_REGION_ST
- V0_END
- V0_REGION_END
- V1_END
- V1_REGION_END
- PIXEL_START_REF
- PIXEL_END_REF

EAV/SAV Mode with Interlaced Modes

This section describes what to do when replacing the ADV202 with the ADV212 in Decode Slave mode for interlaced modes [NTSC, PAL, 1080i] while EAV/SAV mode is used.

On the ADV212 the Field bit in the timing codes is updated differently than on the ADV202. The ADV212 updates on the EAV first, then the SAV for a particular line, whereas the ADV202 only updates the SAV for that particular line.

The data sequence for EAV/SAV video data is :
1. FF 00 00 EAV
2. Digital line blanking
3. FF 00 00 SAV
4. Digital active line data

When using the ADV212, change the F0_START and F1_START register settings. Subtract 1 from the value of the ADV202 register settings for F0_START and F1_START as follows:

- NTSC Example:
  ADV202 F0_START = 4
  ADV202 F1_START = 266

  change to :

  ADV212 F0_START = 3
  ADV212 F1_START = 265

- PAL Example:
  ADV202 F0_START = 4
  ADV202 F1_START = 266

  change to :

  ADV212 F0_START = 3
  ADV212 F1_START = 265

- 1080i Example:
  ADV202 F0_START = 4
  ADV202 F1_START = 266

  change to :

  ADV212 F0_START = 3
  ADV212 F1_START = 265

Decode Slave Frame Drop

In previous releases, vertical synchronization was obtained only once, on the first inactive edge of VSYNC. Subsequent VSYNCs were ignored.

The ADV212 is sensitive to every end-of-frame VSYNC transition. Therefore, to avoid any frame drops, the VSYNC inactive edge should not be asserted during the active portion of a line. Rather, it should be asserted during the horizontal blanking of the last active line in a field.

Example:
HSYNC polarity = 0 = negative edge.
VSYNC polarity = 0 = negative edge.

The polarity is set with the PICFG decode parameter for non-custom specific modes or in the PMODE2 register for Custom Specific mode. For these polarity settings, VSYNC rising edge should be applied when HSYNC = 1 during horizontal blanking.

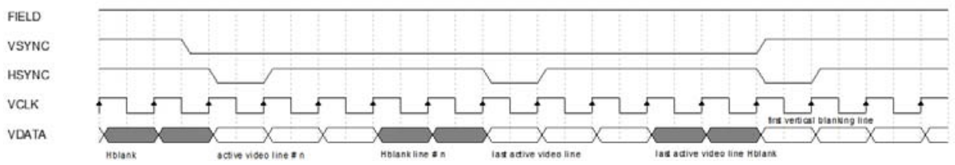

Figure 16. DV212 Incorrect H, V, F Sync Input in Decode Slave Mode

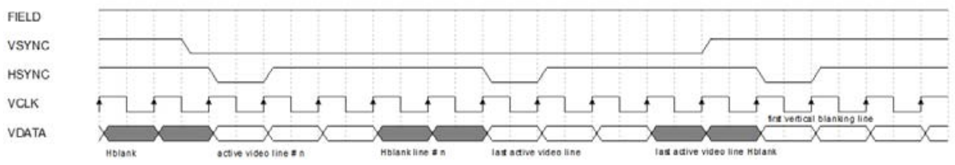

Figure 17. DV212 Correct H, V, F Sync Input in Decode Slave Mode

VDATA Output Delays

The ADV212 has the following VDATA output delays for decode slave modes. These delays are true for custom and default video modes. (This does not include RAW or HIPI modes.)

NOTE: The VDATA output delay is defined as the number of VCLK cycles between the HSYNC active edge and ADV212 valid output video.

Single-Component and Two-Component Mode:
EAV/SAV = 8 VCLK cycles. (HSYNC active edge and first 0xFF time code)
HVF = 10 VCLK cycles. (HSYNC active edge and first VDATA sample)

Three-Component 4:2:2 Mode.
EAV/SAV = 10 VCLK cycles. (HSYNC active edge and first 0xFF time code)
HVF = 11 VCLK cycles. (HSYNC active edge and first VDATA sample)

Decode Master Mode

VSYNC, HSYNC, FIELD, or alternatively EAV/SAV codes, are generated according to the register settings of: XTOT, YTOT, F0_START, F1_START, V0_START, V1_START, V0_END, V1_END, PIXEL_START, PIXEL_END. To enable the generation of these timing signals in decode mode, the VMODE register must be programmed to decode master mode and MP_656 must be set to 1 for HVF generation or 0 for EAV/SAV insertion into the video output stream .

Video Input Formats

Pin Input Formats on HDATA and VDATA bus

The tables below show the input pin configurations using HDATA and VDATA bus.

INPUT PIN CONFIGURATION AND ASSIGNMENT ON VDATA BUS

Table 62.

DATA FORMAT	**FORMAT**	**VDATA PINS [MSB]…[LSB]**
Video	8-bit	VDATA[11]…[4]
	10-bit	VDATA[11]…[2]
	12-bit	VDATA[11]…[0]
Pixel	8-bit	VDATA[15]…[8]
	10-bit	VDATA[15]…[6]
	12-bit	VDATA[15]…[4]
	16-bit	VDATA[15]…[0]

CONFIGURATION AND ASSIGNMENT ON HDATA BUS

Table 63.

DATA FORMAT	**FORMAT**	**HDATA (31..24)**	**HDATA (23..16)**	**HDATA (15..8)**	**HDATA (7..0)**	
Pixel	8-bit	HDATA [31]...[24]	HDATA [23]...[16]	HDATA [15]...[8]	HDATA [7]...[0]	
		Sn	Sn+1.	Sn+2	Sn+3	1 component
		Cbn	Crn	Cbn+1	Crn+1	2 component
		Yn, Yn+2...	Yn+1, Yn+3.	Cbn, Cbn+1.	Crn, Crn+1...	3 component
		Yn, Yn+2...	Cbn, Cbn+1.	Yn+1, Yn+3.	Crn, Crn+1...	3 component

10-, 12-, 14- AND 16-BIT INPUT CONFIGURATION ON HDATA BUS

Table 64.

DATA FORMAT	FORMAT	HDATA UPPER (msb..lsb)	HDATA LOWER (msb..lsb)	
Pixel	10-bit	HDATA [31]...[22]	HDATA [15]...[6]	
		Sn	Sn+1	1 component
		Cbn, Cbn+1	Cr, Crn+1	2 component
		Yn, Yn+1...	Cbn, Crn	3 component
		Yn, Cbn	Yn+1, Crn	3 component
Pixel	12-bit	HDATA [31]...[20]	HDATA [15]...[4]	
		Sn	Sn+1	1 component
		Cbn, Cbn+1	Cr, Crn+1	2 component
		Yn, Yn+1...	Cbn, Crn	3 component
		Yn, Cbn	Yn+1, Crn	3 component
Pixel	14-bit	HDATA [31]...[18]	HDATA [15]...[2]	
		Sn	Sn+1	1 component
		Cbn, Cbn+1	Cr, Crn+1	2 component
		Yn, Yn+1...	Cbn, Crn	3 component
		Yn, Cbn	Yn+1, Crn	3 component
Pixel	16-bit	HDATA [31]...[16]	HDATA [15]...[0]	
		Sn	Sn+1	1 component
		Cbn, Cbn+1	Cr, Crn+1	2 component
		Yn, Yn+1...	Cbn, Crn	3 component
		Yn, Cbn	Yn+1, Crn	3 component

Video Input Formats on HDATA bus

The tables below show the 8-,10-, 12-, 14-, 2x8-, and 16- bit video input formats on the HDATA bus. These input formats are controlled by the PFMT and PREC register settings in the PMODE1 register.

DATA ALIGNMENT FOR 8-BIT COMPONENT INPUT FORMATS ON THE HDATA BUS

Table 65.

8-bit Three-Component YCbYCr4:2:2			**8-bit Three-Component YYCbCr4:2:2**		
msb(7)	Y_n	lsb(0)	msb(7)	Y_n	lsb(0)
msb(7)	Cb_n	lsb(0)	msb(7)	Y_{n+1}	lsb(0)
msb(7)	Y_{n+1}	lsb(0)	msb(7)	Cb_n	lsb(0)
msb(7)	Cr_n	lsb(0)	msb(7)	Cr_n	lsb(0)
Repeat			Repeat		
8-bit Single Component			**8-bit Two-Component CbCr**		
msb(7)	S_n	lsb(0)	msb(7)	Cb_n	lsb(0)
Repeat			msb(7)	Cr_n	lsb(0)
			Repeat		

10-BIT COMPONENT INPUT FORMATS ON HDATA BUS

Table 66.

10-bit Three-Component YCbYCr4:2:2			**10-bit Three-Component YYCbCr4:2:2**		
msb(9)	Y_n	lsb(0)	msb(9)	Y_n	lsb(0)
msb(9)	Cb_n	lsb(0)	msb(9)	Y_{n+1}	lsb(0)
msb(9)	Y_{n+1}	lsb(0)	msb(9)	Cb_n	lsb(0)
msb(9)	Cr_n	lsb(0)	msb(7)	Cr_n	lsb(0)
Repeat			Repeat		
10-bit Single Component			**10-bit Two-Component CbCr**		
msb(9)	S_n	lsb(0)	msb(9)	Cb_n	lsb(0)
Repeat			msb(9)	Cr_n	lsb(0)
			Repeat		

12-BIT COMPONENT INPUT FORMATS ON HDATA BUS

Table 67.

12-bit Three-Component YCbYCr4:2:2			**12-bit Three-Component YYCbCr4:2:2**		
msb(11)	Y_n	lsb(0)	msb(11)	Y_n	lsb(0)
msb(11)	Cb_n	lsb(0)	msb(11)	Y_{n+1}	lsb(0)
msb(11)	Y_{n+1}	lsb(0)	msb(11)	Cb_n	lsb(0)
msb(11)	Cr_n	lsb(0)	msb(11)	Cr_n	lsb(0)
Repeat			Repeat		
12-bit Single Component			**12-bit Two-Component CbCr**		
msb(11)	S_n	lsb(0)	msb(11)	Cb_n	lsb(0)
Repeat			msb(11)	Cr_n	lsb(0)
			Repeat		

14-BIT COMPONENT INPUT FORMATS ON HDATA BUS

Table 68.

14-bit Three-Component YCbYCr4:2:2			**14-bit Three-Component YYCbCr4:2:2**		
msb(13)	Y_n	lsb(0)	msb(13)	Y_n	lsb(0)
msb(13)	Cb_n	lsb(0)	msb(13)	Y_{n+1}	lsb(0)
msb(13)	Y_{n+1}	lsb(0)	msb(13)	Cb_n	lsb(0)
msb(13)	Cr_n	lsb(0)	msb(13)	Cr_n	lsb(0)
Repeat			*Repeat*		
14-bit Single Component			**14-bit Two-Component CbCr**		
msb(13)	S_n	lsb(0)	msb(13)	Cb_n	lsb(0)
	Repeat		msb(13)	Cr_n	lsb(0)
			Repeat		

2X8-BIT THREE-COMPONENT INPUT FORMATS ON HDATA BUS

Table 69.

2x8-bit Three-Component YCbYCr4:2:2					
msb(7)	Y_n	lsb(0)	msb(7)	Cb_n	lsb(0)
msb(7)	Y_{n+1}	lsb(0)	msb(7)	Cr_n	lsb(0)
Repeat					
2x8-bit Three-Component YYCbYCr4:2:2					
msb(7)	Y_n	lsb(0)	msb(6)	Y_{n+1}	lsb(0)
msb(7)	Cb_n	lsb(0)	msb(6)	Cr_n	lsb(0)
Repeat					

2X8-BIT SINGLE- AND TWO-COMPONENT INPUT FORMATS ON HDATA BUS

Table 70.

2x8-bit Two-Component CbCr					
msb(7)	Cb_n	lsb(0)	msb(7)	Cr_n	lsb(0)
Repeat					
2x8-bit Single Component					
msb(7)	S_n	lsb(0)	msb(7)	S_{n+1}	lsb(0)
Repeat					

16-BIT INPUT FORMATS ON HDATA BUS

Table 71.

16-bit Three-Component YCbYCr4:2:2			**16-bit Three-Component YYCbCr4:2:2**		
msb(15)	Y_n	lsb(0)	msb(15)	Y_n	lsb(0)
msb(15)	Cb_n	lsb(0)	msb(15)	Y_{n+1}	lsb(0)
msb(15)	Y_{n+1}	lsb(0)	msb(15)	Cb_n	lsb(0)
msb(15)	Cr_n	lsb(0)	msb(15)	Cr_n	lsb(0)
Repeat			Repeat		
16-bit Single Component			**16-bit Two-Component CbCr**		
msb(15)	S_n	lsb(0)	msb(15)	Cb_n	lsb(0)
	Repeat		msb(15)	Cr_n	lsb(0)
			Repeat		

Video Input Formats on VDATA bus

The tables below show the 8-, 10-, 12-bit video input formats. These input formats are controlled by the PFMT and PREC register settings in the PMODE1 register.

All data transmitted on the VDATA bus is always MSB aligned (left justified); unused pins on the bus are read as 0 while writes on the unused pins are ignored by the ADV212. In HVF and EAV/SAV modes, the VDATA[11..0] pins are used.

8-BIT THREE-COMPONENT CbYCrY EAV/SAV MODE

Table 72.

VDATA(11)		VDATA(4)	VDATA(3..0)
msb(7)	0xFF	lsb(0)	‘1111’
msb(7)	0x00	lsb(0)	‘0000’
msb(7)	0x00	lsb(0)	‘0000’
msb(7)	SAV	lsb(0)	‘0000’
msb(7)	Cb_n	lsb(0)	‘0000’
msb(7)	Y_n	lsb(0)	‘0000’
msb(7)	Cr_n	lsb(0)	‘0000’
msb(7)	Y_{n+1}	lsb(0)	‘0000’
Repeat			
msb(7)	0xFF	lsb(0)	‘1111’
msb(7)	0x00	lsb(0)	‘0000’
msb(7)	0x00	lsb(0)	‘0000’
msb(7)	EAV	lsb(0)	‘0000’

8-BIT TWO-COMPONENT CbCr EAV/SAV MODE

Table 73.

VDATA(11)		VDATA(4)	VDATA(3..0)
msb(7)	0xFF	lsb(0)	‘1111’
msb(7)	0x00	lsb(0)	‘0000’
msb(7)	0x00	lsb(0)	‘0000’
msb(7)	SAV	lsb(0)	‘0000’
msb(7)	Cb_n	lsb(0)	‘0000’
msb(7)	Cr_n	lsb(0)	‘0000’
msb(7)	Cb_{n+1}	lsb(0)	‘0000’
msb(7)	Cr_{n+1}	lsb(0)	‘0000’
Repeat			
msb(7)	0xFF	lsb(0)	‘1111’
msb(7)	0x00	lsb(0)	‘0000’
msb(7)	0x00	lsb(0)	‘0000’
msb(7)	EAV	lsb(0)	‘0000’

8-BIT SINGLE-COMPONENT EAV/SAV MODE

Table 74.

VDATA(11)		VDATA(4)	VDATA(3..0)
msb(7)	0xFF	lsb(0)	'1111'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	SAV	lsb(0)	'0000'
msb(7)	S_n	lsb(0)	'0000'
msb(7)	S_{n+1}	lsb(0)	'0000'
msb(7)	S_{n+2}	lsb(0)	'0000'
Repeat			
msb(7)	0xFF	lsb(0)	'1111'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	EAV	lsb(0)	'0000'

8-BIT THREE-COMPONENT CbYCrY HVF MODE

Table 75.

VDATA(11)		VDATA(4)	VDATA(3..0)
msb(7)	Cb_n	lsb(0)	'0000'
msb(7)	Y_n	lsb(0)	'0000'
msb(7)	Cr_n	lsb(0)	'0000'
msb(7)	Y_{n+1}	lsb(0)	'0000'
Repeat			

8-BIT TWO-COMPONENT CbCr HVF MODE

Table 76.

VDATA(11)		VDATA(4)	VDATA(3..0)
msb(7)	Cb_n	lsb(0)	'0000'
msb(7)	Cr_n	lsb(0)	'0000'
msb(7)	Cb_{n+1}	lsb(0)	'0000'
msb(7)	Cr_{n+1}	lsb(0)	'0000'
Repeat			

8-BIT SINGLE-COMPONENT HVF MODE

Table 77.

VDATA(11)		VDATA(4)	VDATA(3..0)
msb(7)	S_n	lsb(0)	'0000'
Repeat			

8-BIT SINGLE-COMPONENT RAW MODE

Table 78.

VDATA(15)		VDATA(8)	VDATA(7..0)
msb(7)	S_n	lsb(0)	'00'
		Repeat	

10-BIT THREE-COMPONENT YCbCr EAV/SAV MODE

Table 79.

VDATA(11)		VDATA(4)	VDATA(3..2)	VDATA(1..0)
msb(7)	0xFF	lsb(0)	'11'	'11'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	SAV	lsb(0	'00'	'00'
msb(9)	Cb_n		lsb(0)	'00'
msb(9)	Y_n		lsb(0)	'00'
msb(9)	Cr_n		lsb(0)	'00'
msb(9)	Y_{n+1}		lsb(0)	'00'
		Repeat		
msb(7)	0xFF	lsb(0	'11'	'11'
msb(7)	0x00	lsb(0	'00'	'00'
msb(7)	0x00	lsb(0	'00'	'00'
msb(7)	EAV	lsb(0	'00'	'00'

10-BIT TWO-COMPONENT CbCr HVF MODE

Table 80.

VDATA(11)		VDATA(4)	VDATA(3..2)	VDATA(1..0)
msb(7)	0xFF	lsb(0)	'11'	'11'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	SAV	lsb(0)	'00'	'00'
msb(9)	Cb_n		lsb(0)	'00'
msb(9)	Cr_n		lsb(0)	'00'
msb(9)	Cb_{n+1}		lsb(0)	'00'
msb(9)	Cr_{n+1}		lsb(0)	'00'
		Repeat		
msb(7)	0xFF	lsb(0)	'11'	'11'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	EAV	lsb(0)	'00'	'00'

10-BIT SINGLE-COMPONENT EAV/SAV MODE

Table 81.

VDATA(11)		VDATA(4)	VDATA(3..2)	VDATA(1..0)
msb(7)	0xFF	lsb(0)	'11'	'11'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	SAV	lsb(0)	'00'	'00'
msb(9)	S_n		lsb(0	'00'
msb(9)	S_{n+1}		lsb(0	'00'
msb(9)	S_{n+2}		lsb(0	'00'
msb(9)	S_{n+3}		lsb(0	'00'
Repeat				
msb(7)	0xFF	lsb(0)	'11'	'11'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	EAV	lsb(0)	'00'	'00'

10-BIT THREE-COMPONENT CbYCrY EAV/SAV MODE

Table 82.

VDATA(11)		VDATA(2)	VDATA(1..0)
msb(9)	Cb_n	lsb(0)	'00'
msb(9)	Y_n	lsb(0)	'00'
msb(9)	Cr_n	lsb(0)	'00'
msb(9)	Y_{n+1}	lsb(0)	'00'
Repeat			

10-BIT TWO-COMPONENT CbCr HVF MODE

Table 83.

VDATA(11)		VDATA(2)	VDATA(1..0)
msb(9)	Cb_n	lsb(0)	'00'
msb(9)	Cr_n	lsb(0)	'00'
msb(9)	Cb_{n+1}	lsb(0)	'00'
msb(9)	Cr_{n+1}	lsb(0)	'00'
Repeat			

10-BIT SINGLE COMPONENT - HVF MODE

Table 84.

VDATA(11)		VDATA(2)	VDATA(1..0)
msb(9)	S_n	lsb(0)	'00'
Repeat			

10-BIT SINGLE COMPONENT RAW MODE

Table 85.

VDATA(15)		VDATA(6)	VDATA(5..0)
msb(9)	S_n	lsb(0)	'00'
Repeat			

12-bit Pixel Interface

12-BIT THREE-COMPONENT YCbCr EAV/SAV MODE

Table 86.

VDATA(11)			VDATA(3..0)
msb(7)	0xFF	lsb(0)	'1111'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	SAV	lsb(0)	'0000'
msb(11)	Cb_n		lsb(0)
msb(11)	Y_n		lsb(0)
msb(11)	Cr_n		lsb(0)
msb(11)	Y_{n+1}		lsb(0)
Repeat			
msb(7)	0xFF	lsb(0)	'1111'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	EAV	lsb(0)	'0000'

12-BIT TWO-COMPONENT CbCr EAV/SAV MODE

Table 87.

VDATA(11)			**VDATA(3..0)**
msb(7)	0xFF	lsb(0)	'1111'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	SAV	lsb(0)	'0000'
msb(11)	Cb_n		lsb(0)
msb(11)	Cr_n		lsb(0)
msb(11)	Cb_{n+1}		lsb(0)
msb(11)	Cr_{n+1}		lsb(0)
Repeat			
msb(7)	0xFF	lsb(0)	'1111'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	EAV	lsb(0)	'0000'

12-BIT SINGLE-COMPONENT EAV/SAV MODE

Table 88.

VDATA(11)			**VDATA(3..0)**
msb(7)	0xFF	lsb(0)	'1111'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	SAV	lsb(0)	'0000'
msb(11)	S_n		lsb(0)
msb(11)	S_{n+1}		lsb(0)
msb(11)	S_{n+2}		lsb(0)
msb(11)	S_{n+3}		lsb(0)
Repeat			
msb(7)	0xFF	lsb(0)	'1111'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	EAV	lsb(0)	'0000'

12-BIT THREE-COMPONENT CbYCrY HVF MODE

Table 89.

VDATA(11)			**VDATA(0)**
msb(11)	Cb_n		lsb(0)
msb(11)	Y_n		lsb(0)
msb(11)	Cr_n		lsb(0)
msb(11)	Y_{n+1}		lsb(0)
Repeat			

12-BIT TWO-COMPONENT CbCr HVF MODE

Table 90.

VDATA(11)			**VDATA(0)**
msb(11)	Cb_n		lsb(0)
msb(11)	Cr_n		lsb(0)
msb(11)	Cb_{n+1}		lsb(0)
msb(11)	Cr_{n+1}		lsb(0)
Repeat			

12-BIT SINGLE -COMPONENT HVF MODE

Table 91.

VDATA(11)			**VDATA(1..0)**
msb(11)	S_n		lsb(0)
Repeat			

12-BIT SINGLE-COMPONENT RAW MODE

Table 92.

VDATA(15)			**VDATA(4)**
msb(11)	S_n		lsb(0)
Repeat			