ANALOG DEVICES

AN-790
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106 • Tel: 781/329-4700 • Fax: 781/461-3113 • www.analog.com

# How to Use the ADV202

by Christine Bako

## INTRODUCTION

This application note gives an overview of the architecture of the ADV202 and its functional blocks. It explains data flows in encode and decode modes, and what interfaces to use for uncompressed data input and compressed data output. An overview of how the HDATA bus can be configured for compressed data output or HIPI mode is also discussed. This application note also includes a synopsis of what is required to configure the ADV202 and an application-specific overview for standard definition, high definition, and still image applications.

## ADV202 ARCHITECTURE

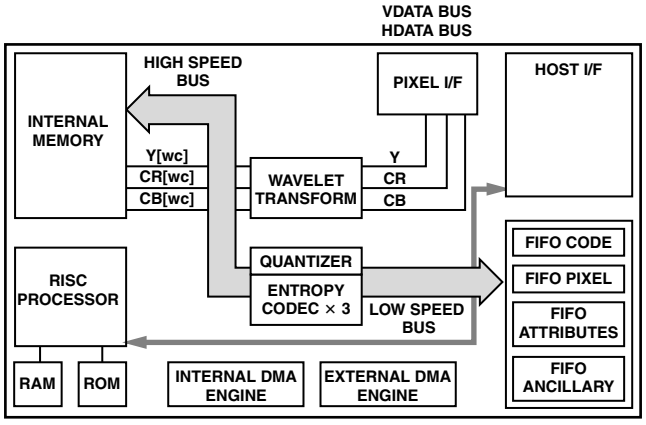


*Figure 1. Block Diagram of the ADV202*

### Block Diagram and Functional Description

The ADV202 contains the following main functional blocks:

- An embedded 32-bit RISC processor serves as a system controller. The RISC processor includes its own ROM and RAM for both program and data memory.
- Wavelet transform engine with 5/3 and 9/7 wavelet filters giving up to six transform levels.
- Three entropy codecs (ECs) generate JPEG2000 code stream using quantization, rate distortion optimization, and context modeling, and organize data into packets and layers. The three ECs can guarantee a throughput of up to 65 MSPS.
- Host interface can be configured for 32-bit or 16-bit control and 8-, 16-, or 32-bit data. Controls data transfers between FIFOs and controls two external DMA channels. Controls access to indirect and direct register.
- Pixel interface is used to process video or pixel data transfers from/to VDATA or HDATA bus.
- Internal DMA engine. Used to facilitate fast internal memory-to-memory data transfers.

### Data Flow in Encode and Decode Mode

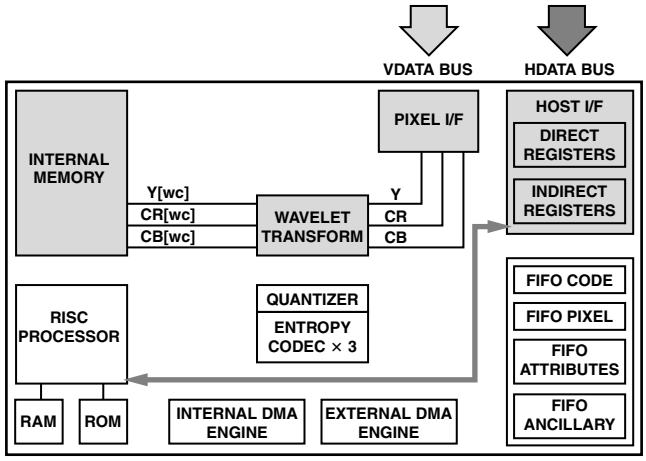


*Figure 2. ADV202 Encode Mode—1*

Video or pixel data can be input over the VDATA bus or, alternatively, pixel data can be input over the HDATA bus. In either case the video/pixel data is passed to the pixel interface.

The data is then deinterleaved and passed to the wavelet transform engine. The input data, organized in tiles or frames, is then decomposed into subbands using 5/3 or 9/7 wavelet filters.

The wavelet transform (WT) can perform up to six wavelet decomposition levels on a tile/frame. The resultant wavelet coefficients are then written to internal memory.

The ADV202 does not have any field buffers to perform compression. The wavelet coefficients are computed on a line by line basis and the entire wavelet coefficients are stored in internal memory.

Throughout operation high bandwidth memory to memory or memory to functional block transfers are handled by the internal DMA engine. The internal DMA engine and bus are connected to every functional block in the ADV202.

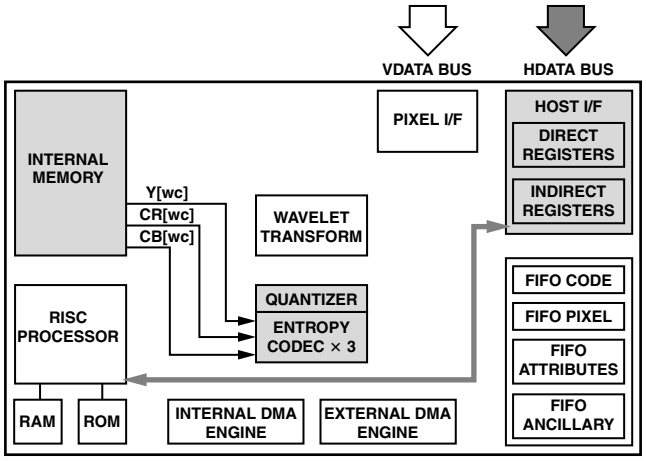


*Figure 3. ADV202 Encode Mode—2*

The wavelet coefficients are passed to three entropy codecs. Wavelet coefficients are arranged into units of code blocks, which are essentially bit streams of data. Processing is done on a code block by code block basis.

Compression is achieved using the following:

- Quantization—The code block bit stream is truncated (lossy compression) or not truncated (lossless) to reduce the amount of data.
- Rate Distortion Optimization—The amount of quantization depends on the output rate requirements or output quality requirements. The ECs will perform distortion metric calculations to find the optimal rate/distortion performance.
- Context Modeling—A process which assigns information about the significance of each individual coefficient. This allows data to be arranged according to its significance. After processing, the data is arranged into packets and layers defining the JPEG2000 code stream.

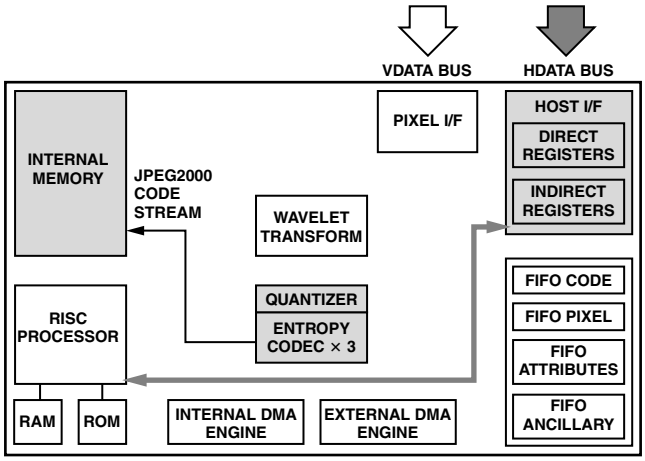


*Figure 4. ADV202 Encode Mode—3*

The output of the entropy codecs is passed back to internal memory.

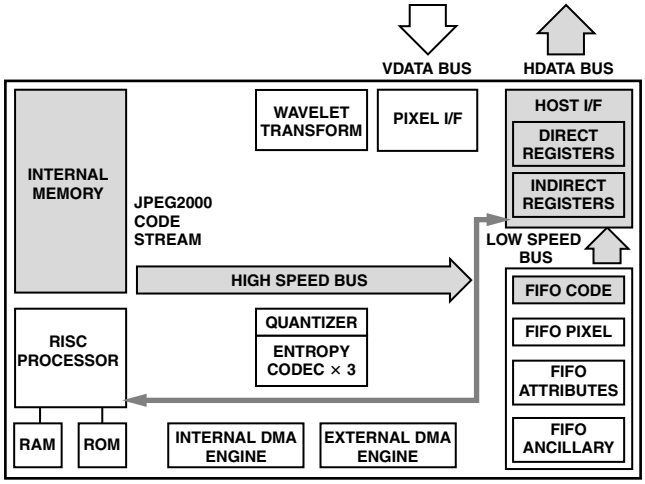


*Figure 5. ADV202 Encode Mode—4*

The JPEG2000 code stream is sent over a high speed bus to the CODE FIFO in order to buffer internal high speed bus and low speed host interface.

The data can then be output over the host interface using a common read/write access protocol ($\overline{CS}$, $\overline{RD}$, $\overline{WR}$, $\overline{ACK}$, ADDR) or over the external DMA engine in conjunction with an external DMA controller using a DREQ/DACK protocol.

The FIFOs have no other function other than to buffer internal high speed bus and the low speed host interface. The PIXEL FIFO is dedicated to pixel/component data. It is used when uncompressed pixel data is input over the HDATA bus and not the VDATA bus. In this case the pixel data is passed from the HDATA bus over the pixel interface to the PIXEL FIFO.

Generally the CODE FIFO is used for compressed data, the ATTR FIFO is used for attribute data such as distortion metrics, distortion slopes, or byte length, and ANCL FIFOs can be used for any other miscellaneous data.

**Decode Data Flow**

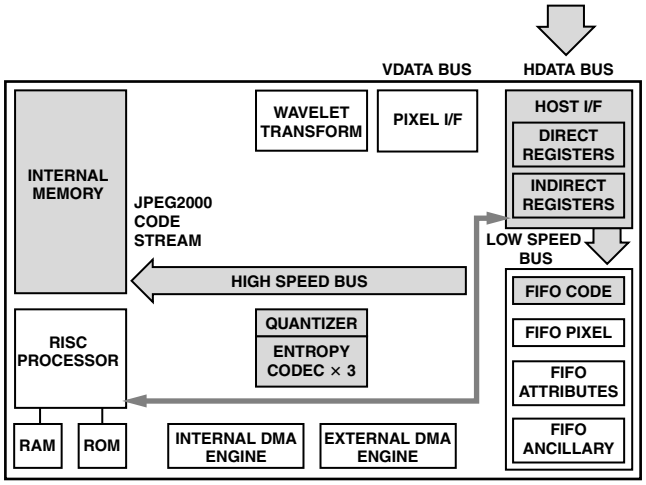


*Figure 6. ADV202 Decode Mode—1*

Compressed data is input over the host interface using the same protocols as in encode mode and passed on to the CODE FIFO from where it is sent to the internal memory. 500 kB are available for internal storage of wavelet data (uncompressed NTSC field = 345 kB).

Depending on the compression ratio, the ADV202 can store one or more fields of wavelet coefficient data.

In decode, the ADV202 works "ahead." It waits until information from 1+ field is present before starting to decode.

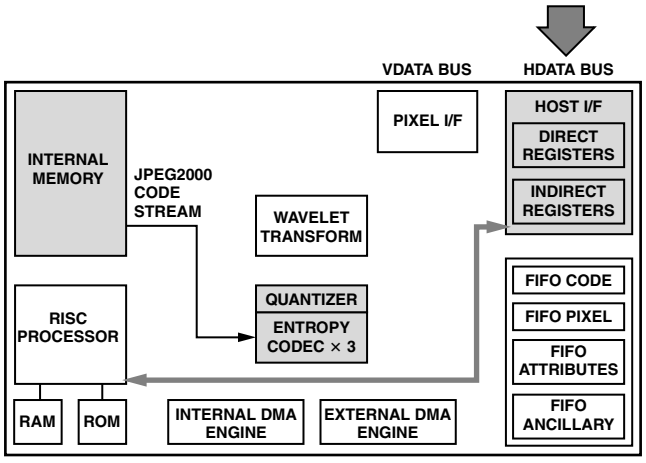


*Figure 7. ADV202 Decode Mode—2*

The JPEG2000 code stream is passed to the entropy codecs which decompose the code stream back into wavelet coefficients.

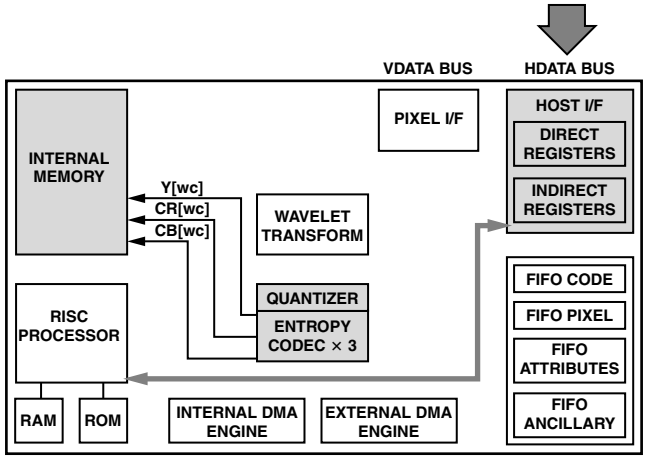


*Figure 8. ADV202 Decode Mode—3*

The wavelet coefficients are passed back into internal memory.

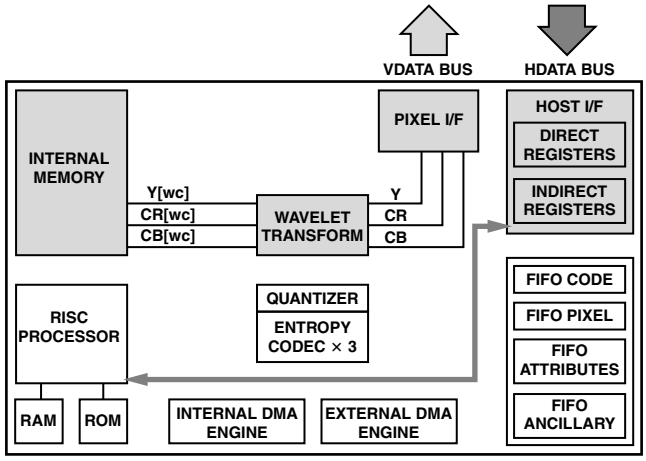


*Figure 9. ADV202 Decode Mode—4*

The wavelet coefficients are read from internal memory. They are recomposed into uncompressed component samples and passed to the pixel interface where the components are interleaved into a video stream and output over the VDATA or HDATA bus.

## ADV202 INTERFACES AND FEATURES

### Data Input Rate Limits

#### *Number of ADV202s Required for an Application*

There are four limiting factors that determine how many ADV202s are required for a particular application:

1. Data input rate.

   Data input rate = active vertical resolution × active horizontal resolution × field rate × components per pixel [MSPS]

   Refer to the ADV202 data sheet, Table 23.

2. Tile width of the input image.

   Refer to the ADV202 data sheet, Table 24.

3. Samples per image.

   Samples per image is limited to 1.024 Msamples. This refers to the number of active samples of the input image/frame or field. Note that YCbCr in 4:2:2 format contains 2 samples per pixel.

   For example, standard definition NTSC contains 720 × 240 × 2 samples/field.

4. Number of code blocks per image.

   Is limited to 610 code blocks per image/frame or field.

   Since code block size is programmable, active resolutions of 1920 × 1080, 720 × 483, 1024 × 1024, and others are supported.

### Ways to Interface to the ADV202

#### *HDATA and VDATA*

VDATA

- Input/output of YCbCr 4:2:2 data either with accompanied HVF sync signals or with embedded EAV/SAV.
- YCbCr 4:2:2 can be single 8-, 10-, or 12-bit wide input/output or using Y and CbCr on separate buses 16-, 20-, or 24-bit wide.
- Nonstandard video data or pixel data in 8-, 10-, 12-, or 16-bit format.

HDATA

- Input/output of pixel data in 8-, 10-, 12-, 14-, or 16-bit format.
- $\overline{RD}$/$\overline{WR}$ data to direct registers to configure ADV202.
- $\overline{RD}$/$\overline{WR}$ data to indirect registers to configure ADV202.
- $\overline{RD}$/$\overline{WR}$ data to FIFOs using DMA modes to access compressed data.

### HDATA—Normal Host Mode

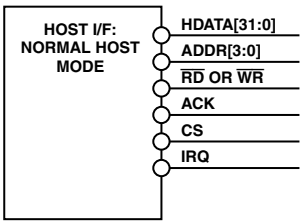


*Figure 10. ADV202—Normal Host Configuration*

- Must be used to write/read to direct and indirect registers to configure ADV202.
- Must be used regardless of what interface mode is used to initialize the part.
- Can be used to input compressed data in decode mode or output compressed data in encode mode.

### HDATA—JDATA Mode

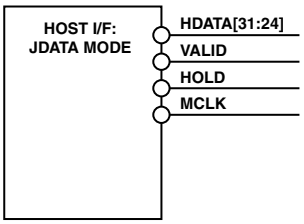


*Figure 11. ADV202—JDATA Mode*

- JDATA mode can be used to input compressed data in decode mode, or output compressed data in encode mode.
- JDATA is a synchronous interface where HDATA is synced to MCLK.
- In encode mode JDATA is always an output; in decode mode JDATA is always an input.
- This mode is for real-time applications where video data is input over VDATA while compressed data is output over JDATA.
- JDATA is configured for an 8-bit wide data bus.
- VALID is asserted by the ADV202 to indicate that the part is ready to input/output data over the JDATA bus.
- HOLD must be asserted by the host after VALID has been asserted.

**HDATA—DMA Mode**

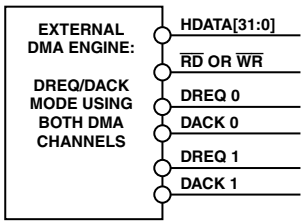


*Figure 12. ADV202 DREQ/DACK Mode*

- Any DMA mode can be used to input compressed data in decode mode, or output compressed data in encode mode.
- The three DMA modes available on the ADV202 are:

  Dedicated Chip Select

  DREQ/DACK DMA

  Fly-By DMA
- The ADV202 has two DMA channels:

  Single DMA channel applications use a configuration similar to JDATA mode, where uncompressed data is input over VDATA while compressed data is output over one of the DMA channels (encode mode).

  Dual DMA channel applications input uncompressed pixel data over DMA Channel 0 while DMA Channel 1 is used to output compressed data (encode mode). This mode is called HIPI mode.

**Which Interface to Use for What Application**

Normal Host Mode

- Least efficient but lowest pin count, i.e., does not require DREQ/DACK pins
- Maximum data throughput = 120 Mbps

JDATA Mode

- Synchronous interface
- Maximum data throughput = 37 Mbps

DREQ/DACK DMA Mode

- DREQ/ is asserted by the ADV202 whenever data is ready
- No FIFO threshold programming required
- Maximum data throughput = 200 Mbps (burst)

DCS DMA Mode

- Hardwired directly from the FIFOs
- FIFO thresholds must be programmed by the user
- Maximum data throughput = 200 Mbps

Fly-By DMA Mode

- Functionally the same as DREQ/DACK except that $\overline{RD}$ and $\overline{WR}$ pin functionalities are swapped
- Designed for compatibility with previous JPEG2000 chip

Refer to the ADV202 data sheet for timing diagrams.

**PROGRAMMING THE ADV202**

**Programming Sequence Overview to Configure the ADV202 for a Specific Application**

1. Initialize ADV202—The user has to write to direct registers to configure the ADV202 using normal host mode to:

   Set operating frequency

   Set data bus width

   Set control bus width

   Set DMA mode (optional)

   Load firmware into ADV202 memory
2. JPEG2000 Parameters—The user then has to write application-specific encode or decode parameters to internal memory using normal host mode.
3. After initialization and loading the parameters to internal memory, the program is started and the firmware will control all internal operations between functional blocks, internal data transfers, timing, setting of all necessary registers in conjunction with the parameter settings, and the direct register settings.

**JPEG2000 Parameters**

Present firmware supports 236 byte locations for parameter settings in encode mode.

These bytes must be written to a specific memory location within the ADV202 using normal host mode.

Parameters include:

Non-JPEG2000 specific

- Video format (PAL, NTSC, 1080i, etc.)
- Bit precision
- Frame drop (select input frame rate)

JPEG2000 specific

- Code block size
- Number of transform levels
- 9/7 or 5/3 wavelet transform
- Reversible or irreversible compression
- Compression ratio = variable bit rate output
- Fixed bit rate output = variable compression rate
- Progression order
- Quantization factor
- File format output (j2c, jp2, Raw-ADV202)
- Visual weighting

**ADV202 File Formats**

ADV202 raw format is an ADV202-specific format. This file format is compatible only with the ADV202, and is a very basic structured JPEG2000 data stream. It contains an ADV202-specific header.

j2c is an "upgrade" to the ADV202 raw format. It contains a ADV202 header, a JPEG2000-compliant header, and JPEG2000 packet headers containing all JPEG2000 markers. This file format is compatible with ADV202 and the JPEG2000 software from kakadusoftware.com (free download). The j2c is a file format supported by the ISO/IEC15444-1 JPEG2000 standard.

jp2 is one of the official JPEG2000 formats supported by the ISO/IEC15444-1 standard. It includes all the information of a j2c file with the added jp2 file syntax that provides additional information about the code stream.

The ADV202 header contains information about:

- Field identification
- Number of fields
- File format
- Video format
- Header version
- Number of compressed data words
- Number of attribute words

The JPEG2000-compliant header and packed headers contain JPEG2000 main header and tile header markers (see Annex A of the JPEG2000 standard). For example:

- SOC – start of code stream – denoted as 0xFF4F
- EOC – end of code stream – denoted as 0xFFD9
- COD – coding style – 0xFF52
- PLM – packet length, main header – 0xFF57
- PLT – packet length, tile header – 0xFF58
- PPM – packed packet header, main header – 0xFF60
- PPT – packed packet header, tile header – 0xFF61

The jp2 file syntax includes information about:

- JPEG2000 file type
- Bit depth of components
- Color space of image (RGB, YCbCr, or monochrome)
- Number of component
- Image size
- Compression type

Refer to Annex I of the JPEG2000 standard.

**APPLICATIONS**

**Standard Definition Video**

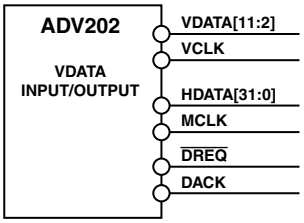


*Figure 13. Block Diagram of a Standard Definition Video Application*

Standard Definition video input/output is supported directly by the ADV202 (YCbCr 4:2:2, ITU.R.BT656 in PAL or NTSC).

Video can be input with EAV/SAV or with separate HVF signals.

For non-656 formats a custom-specific mode can be used to input:

- Single component data [Y or R or G or B]
- CbCr data
- YCbCr data in any non-CCIR656 format

Video input can be in 8-, 10-, or 12-bit format. These formats can be input/output on a single bus or using 2 × 8-, 2 × 10-, or 2 × 12-bit bus.

**High Definition Video**

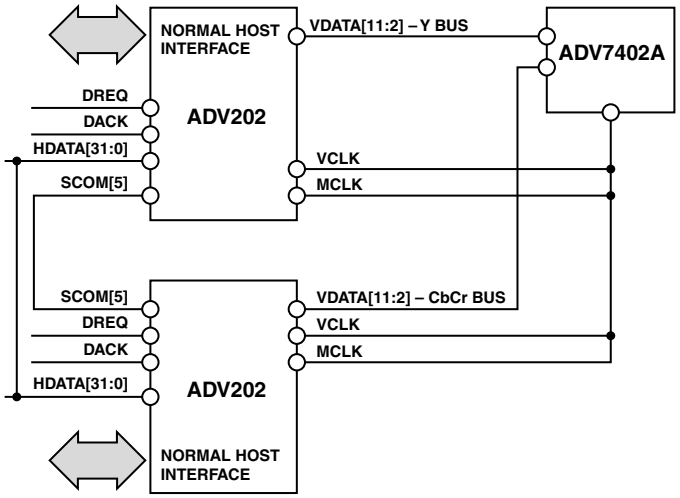


*Figure 14. Block Diagram of a 1080i Application—Encode*

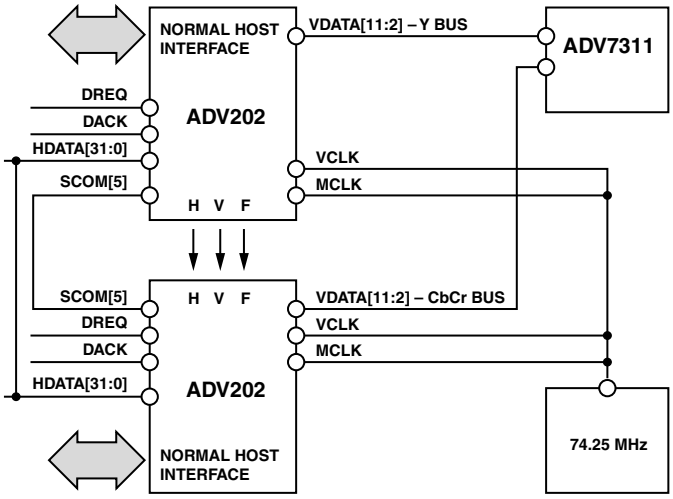


*Figure 15. Block Diagram of 1080i Application—Decode*

HD standards will require a minimum of two ADV202s for full resolution processing.

In encode mode all ADV202s are configured in slave mode. In this mode the input data must contain EAV/SAV codes.

SCOMM[5] is used to sync the encode process to make sure that each ADV202 starts to encode at the start of the same field. SCOMM[5] is asserted by the host.

HD decode can be configured in two modes: Master/Slave or Slave/Slave.

The Master/Slave configuration is shown here. SCOMM[5] is used to sync the outputs of the ADV202s. SCOMM[5] should be asserted by the host after each ADV202 has set its "Ready to output:" flag, [SWIRQ1].

**Still Image Application**

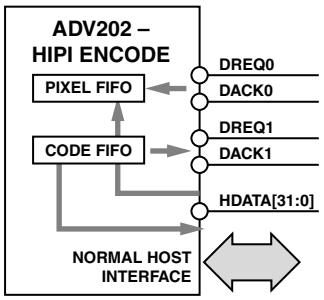


*Figure 16. Still Image Application*

For still image data in YCbCr 4:2:2 or single component format, the VDATA bus can be used in <Raw Video> mode as the interface to input/output pixel data. In this mode, the regions to be captured by the ADV202 are defined by the Hsync and Vsync inputs.

Usually the HDATA bus is used for still image applications to input (encode) or output (decode) pixel data. This mode is called <HIPI mode>. When using this mode, still image data has to be input in raw format, that is raw pixel data without timing information. The region of the image to be captured is defined by internal registers and not by incoming $H_{SYNC}$ or $V_{SYNC}$ signals. The HDATA bus is shared between uncompressed data input and compressed data output. Access is controlled by DREQ0/DACK0 and DREQ1/DACK1 signals.

For questions about ADI's ADV2xx, please visit our product page at analog.com.

AN05532–0–6/05(0)