

Using the ADV202 in HIPI mode

OVERVIEW

This document will provide instructions on how to use the ADV202 in HIPI mode [Host Interface Pixel Interface mode] and applies to

ADV202 engineering samples ES2 and ES3 that are branded as follows:

ADV202xxx

SURF®

xxxxxx 0.V

Where V is larger or equal to 1. ADV202 chips branded with 0.0 will not function properly with the firmware associated with this technical note.

1. INTRODUCTION

The ADV202 allows input or output of video or still image data over the HDATA bus, therefore not making use of the dedicated video interface over the VDATA bus.

This mode is referred to as HIPI mode [Host Interface Pixel Interface] or host mode.

2. RECOMMENDED INTERFACE

In HIPI encode mode, pixel data is input on HDATA [31:0]. DMA channel 0 can be used to write pixel data into the Pixel FIFO, while DMA channel 1 can be used to read compressed data. The read/write transfers are controlled via DREQ0/DACK0 for channel 0 and DREQ1/DACK1/ for channel 1.

In decode mode the same protocol can be used with channel 0 for pixel data output and channel 1 for compressed data input.

Technical Note 002

Encode HIPI configuration

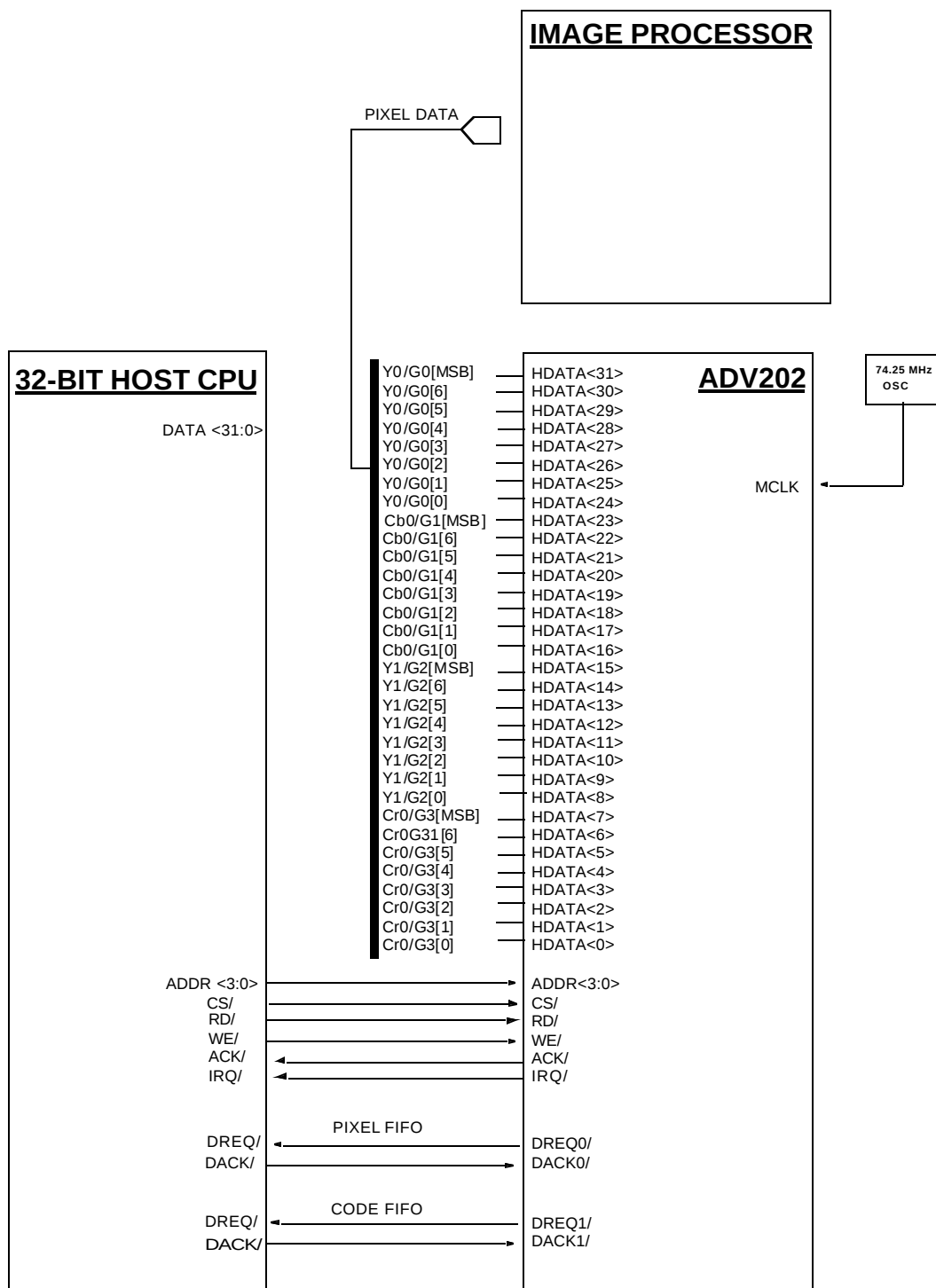


Figure 1. ADV202 in HIPI mode - Encode

Decode HIPI configuration

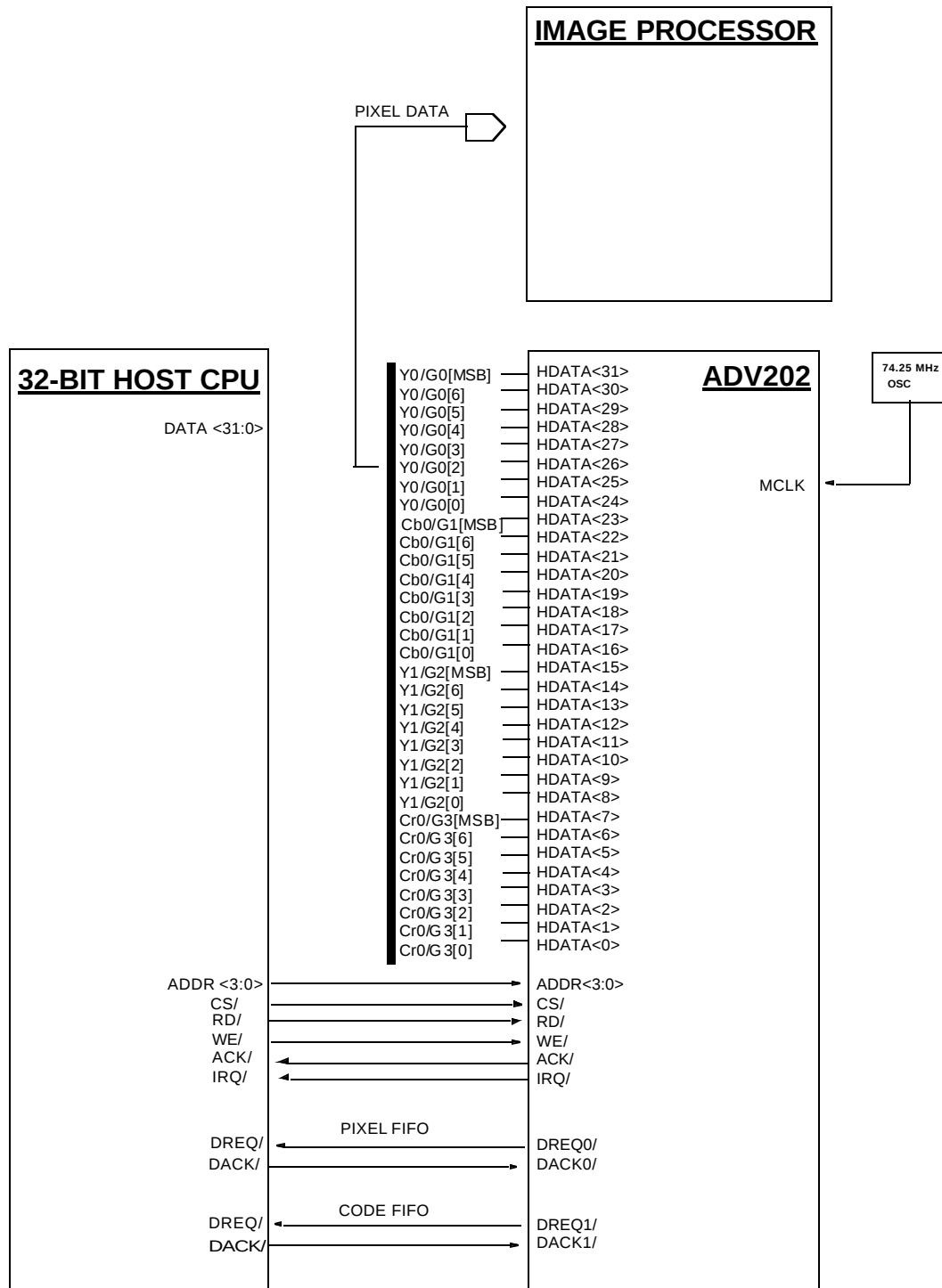


Figure 2. ADV202 in HIPI mode - Decode

Technical Note 002

In HIPI mode pixel data is transferred over the HDATA bus using the entire 32-bit wide HDATA bus width. The data can be in 8-bit packed format [4x 8-bit] or 16-bit packed format [2x16-bit]. Allowed input formats for HIPI mode are listed in the datasheet in the PMODE1 register under PFMT [address 0xFFFF0400]. The settings for PFMT in HIPI mode range from 14h to 1Bh with the current firmware. These formats are described in more detail in the datasheet that is available on the ftp site at:

ftp://ftp.analog.com/pub/Digital_Imaging

For RGB input PFMT in the PMODE1 register must be set to 14h and PMODE2/YUNI set to unipolar.

HIFI mode can be used with the present application software using an ADV202 evaluation board. At the time of writing the application software [Sirius version] as used with the ADV202 evaluation board supports only 640x480 4:2:2 YCbYCr or Luma data for HIPI mode, i.e. PFMT = 15h or 18h [refer to file format section for more detail].

3. FILE FORMAT

In HIPI mode the pixel data is expected to be input in raw pixel format, i.e. the data does not contain format or timing information or any header information, only raw data bytes. The video/pixel data is in either 8-bit packed or 16-bit packed format [refer to datasheet for more detail]. The output format can be programmed to be ADV202 Raw Format, j2c or j2p format.

When the ADV202 evaluation board is used with the ADV202 application software [Sirius version], the output file additionally contains a 64-bit header at the start of the file. The file starts with a header with a 32-bit tag [the word HIPI], followed by a format ID number:

0x48695069 = signature "HIPI"
0x00000000 = undefined format

For the currently released application software [Sirius version] the following two additional format IDs are defined:

0x00000001 = 640x480 8-bit packed 4:2:2 YCbYCr
0x00000002 = 640x480 16-bit packed Luma

The header is in little endian format [LSB leftmost], the rest of the file is big endian format [MSB leftmost].

This header is then followed by the ADV202 header information and the JPEG2000 compliant header, Attribute data or JPEG2000 packed packet header and the code block data [refer to the "Getting Started with the ADV202" application note for more detail on format].

The application software help file describes how HIPI mode can be used with the ADV202 evaluation board. The .pid and .hpi sample files can be viewed in a Hex editor after renaming them to .dat files.

Note: .pid and .hpi sample files can be downloaded from the ftp site at ftp://ftp.analog.com/pub/Digital_Imaging/ADV202_EvalKits/Video.

.pid files are source files of uncompressed data to be used in HIPI mode, .hpi files contain compressed data that were created in HIPI mode.

4. PROTOCOL

INPUT/OUTPUT RATE OVER HDATA BUS

This technical note describes how to use DREQ/DACK DMA mode in burst configuration to transfer data in HIPI mode.

In encode mode DMA channel 0 can be used for pixel data input and DMA channel 1 can be used for compressed data output. Pixel input data is assigned to the PIXEL FIFO and compressed data is assigned to the CODE FIFO.

In decode mode DMA channel 0 can be used for compressed data input and DMA channel 1 can be used for pixel data output. Pixel data is assigned to the PIXEL FIFO and compressed data is assigned to the CODE FIFO.

The highest in/output rate of compressed data is achieved over the HDATA bus using the external DMA DREQ/DACK mode in burst transfer configuration.

Maximum transfer rate for 32-bit data:

Currently, the maximum recommended DMA burst frequency for JCLK set to 150 MHz [ADV202-150 only] is 50 MHz. For other JCLK settings, the maximum DMA burst frequency can be calculated as:

Maximum DMA frequency = $0.35 \times \text{JCLK frequency}$

The JCLK frequency can be set in the PLL_HI and PLL_LO registers [refer to next section].

Maximum JCLK for 144-pin parts is 150 MHz, 121-pin parts it is 115 MHz.

The CODE FIFO in which the compressed data is to be stored is programmed to a size of no less than 256 32-bit words. For 32-bit wide data, the maximum number of accesses is limited to 256, for 16-bit data this is 512.

Therefore the max. data throughput rate for 32-bit data on the HDATA bus is:

$4 \text{ bytes} \times 50 \text{ MHz} = 200 \text{ MBytes/sec}$

Assuming a burst length of 128 accesses, if the CODE FIFO does not contain 128 words for the last portion of the compressed field, the FIFO is packed with zeroes to ensure that the field ends on a burst boundary.

It is of importance to note, that the data throughput rate of the interface used in the application [for example: a PCI interface to transfer/store compressed data on a PC system] supports these maximum data rates. Therefore a 33MHz/32-bit PCI interface has a maximum data throughput rate of :

$33 \text{ Mhz} \times 32\text{-bits} \times 0.96 \text{ [PCI overhead]} = \text{approx. } 125 \text{ MBytes/sec}$

PROGRAMMING THE INDIRECT REGISTERS OF THE ADV202 FOR HIPI MODE

Data can be input in three component [YCbCr], two component [CbCr] or single component format [R, G, B, Y, Cb or Cr].

In HIPI mode the PMODE 1 register must be programmed according to the input format used. Only values 14h - 1Bh for PFMT in the PMODE1 register are allowed in HIPI mode.

Technical Note 002

Tile/image size dimension of the input must be programmed. For this the only registers that require programming in HIPI mode are the XTOT and YTOT registers.

Indirect registers are generally accessed by the firmware. Summarizing the user has only to be concerned to program the following registers for HIPI mode:

0xFFFF0400	PMODE1, settings for PFMT
0xFFFF040C	XTOT
0xFFFF0410	YTOT
0xFFFF0448	VMODE, settings for MAS_SLV, ENC_DEC, HOST_MODE
0xFFFF1408	EDMOD0 for DMA channel 0
0xFFFF140C	EDMOD1 for DMA channel 1

The following is an outline on how to program the ADV202 for HIPI mode in encode mode, using DMA channel 0 as the input channel and DMA channel 1 to output compressed data.

Please also refer additionally to the application note "Getting Started with the ADV202" and the ADV202 datasheet.

5. PROGRAMMING THE ADV202 FOR HIPI MODE

The following applies to encode mode as well as decode mode, unless otherwise specified.

(1) Programming the ADV202 for HIPI mode :

1. Write 0x0008h to the PLL_HI register, 0x0084 to the PLL_LO register.
2. Wait for 20us to allow for the PLL to settle.
3. Write 0x008A to the BOOT register. This boot mode is used for applications where the firmware has to be loaded into the part as opposed to having the firmware stored in the ADV202's ROM.
4. Write 0x000A to BUSMODE. This sets the host control data width to 32-bits and the DMA data width to 32-bits.
5. Write 0x000A to MMODE. This sets the indirect data access width and indirect address step size to 32-bits.
6. Set the start of the program memory in writing 0x00050000 to IADDR.
7. Load the program into the memory.
This is achieved by writing every 32-bit value of the firmware to IDATA.
Firmware file for encode is <encode_x_x.sea>, for decode it is <decode_x_x.sea>.
8. Initiate a reboot by writing 0x008D to the BOOT register. This initiates program execution.

9. Write 0x000A to BUSMODE.

10. Write 0x000A to MMODE.

(2) Pre-initialization Routine for ADV202

1. Write 0x00057F00 to IADDR. This sets the start address for the encode or decode parameters which will be loaded into the ADV202.

2. Write 0x04000503 to IDATA.

[refer to Table 3 in the “ Getting Started with the ADV202” application note for more detail]

In this example:

04 = Custom Specific video format

00 = Ignored in HIPI mode, default to 8-bit precision

05 = 5 Levels of Wavelet transforms

03 = Ignored in HIPI mode, default to uni-polar component polarity

3. Write 0x01000000 to IDATA for both encode or decode mode.

01 = Codeblock size 64x32

00 = Irreversible 9x7 using fixed table

00 = Skip no fields

00 = Do not output attribute data.

4. For HIPI encode:

Write 0x02000500 to IDATA

02 = Target Quality per video field

500 = Quality factor for Rate Control

Write 0x00000000 to IDATA

00 = JPEG2000 Progressive Style : LRCP

00 = ignored in HIPI mode. Sync and clk polarities and selection between HVF and EAV/SAV are not selectable in HIPI mode.

00 = Quantization factor is 1x

00 = Code to 'ADV202 Raw format'

For HIPI decode:

Write 0x00000000 to IDATA.

Write 0x00000000 to IDATA.

5. The following register values will apply to encode and decode mode. These values are used to program the ADV202 to accept the format of the input file and to activate HIPI mode.

In this example a total resolution of 720x525, YCbCr, 8-bit data is used.

It is recommended to use a format with an even number of pixels per line.

Write 0xFFFF0400 to IADDR (0xb); select PMODE register

Write 0x00150000 to IDATA (0xc); 8-bit YCbYCr

Write 0xFFFF040C to IADDR (0xb); XTOT

Write 0x05A00000 to IDATA (0xc); 1440

Technical Note 002

Write 0xFFFF0410 to IADDR (0xb); YTOT
Write 0x020D0000 to IDATA (0xc); 525
Write 0xFFFF0414 to IADDR (0xb); F0_START
Write 0x00010000 to IDATA (0xc); Ignored in HIPI mode
Write 0xFFFF0418 to IADDR (0xb); F1_START
Write 0x00000000 to IDATA (0xc); Ignored in HIPI mode
Write 0xFFFF041C to IADDR (0xb); V0_START
Write 0x00000000 to IDATA (0xc); Ignored in HIPI mode
Write 0xFFFF0420 to IADDR (0xb); V1_START
Write 0x00000000 to IDATA (0xc); Ignored in HIPI mode
Write 0xFFFF0424 to IADDR (0xb); V0_END
Write 0x00000000 to IDATA (0xc); Ignored in HIPI mode
Write 0xFFFF0428 to IADDR (0xb); V1_END
Write 0x00000000 to IDATA (0xc); Ignored in HIPI mode
Write 0xFFFF042C to IADDR (0xb); PIXEL_START
Write 0x00000000 to IDATA (0xc); Ignored in HIPI mode
Write 0xFFFF0430 to IADDR (0xb); PIXEL_END
Write 0x00000000 to IDATA (0xc); Ignored in HIPI mode
Write 0xFFFF0448 to IADDR (0xb); PMODE2
Write 0x00000000 to IDATA (0xc); Ignored in HIPI mode

Write 0xFFFF044C to IADDR (0xb); VMODE
Write 0x00120000 to IDATA (0xc), Host mode enabled for HIPI encode mode. Or

Write 0xFFFF044C to IADDR (0xb); VMODE
Write 0x00100000 to IDATA (0xc), Host mode enabled for HIPI decode mode.

(3) Initialization Routine for ADV202

1. Write to 0x0400 to EIRQIE to unmask SWIRQ0.
2. Wait for IRQ to be asserted [going low].
3. Read Application ID to ensure the program has correctly initialized. Here : 0xFF82 for encode or 0xFFA2 for decode.

(4) Post-Initialization Routine to configure DMA channel 0 for pixel input

1. Write 0xFFFF1408 to IADDR.
2. Write 0x21100000 to IDATA.
This configures DMA channel 0 to 128 bursts of 32-bit words, assigned to the Pixel data FIFO and DREQ pulse width to 4 Jclkcycles [refer to datasheet, figure 1 for definition of Jclkcycles].

3. Write 0xFFFF1408 to IADDR.

4. Write 0x21110000 to IDATA.

(5) Post-Initialization Routine to configure DMA channel 1 for compressed data output

1. Write 0xFFFF140C to IADDR.

2. Write 0x21120000 to IDATA.

This configures DMA channel 0 to 128 bursts of 32-bit words, assigned to the Compressed data/Code-block data FIFO and DREQ pulse width to 4 Jclkcycles [refer to datasheet, figure 1 for definition of Jclkcycles].

3. Write 0xFFFF140C to IADDR.

4. Write 0x21130000 to IDATA.

(6) Start program for ADV202

Write 0x0400 to EIRQFLG on the ADV202 to clear the software interrupt [SWIRQ0] and start the program.

(7) Data transfer in encode mode.

Active pulses on DREQ0 indicate that the ADV202 is ready to receive a burst of data. The host should then initiate a burst data transfer using DMA channel 0 [PIXEL FIFO]. Active pulses on DREQ1 indicate that the ADV202 is ready to transmit a burst of data. The host should then initiate a burst data transfer using DMA channel 1 [CODE FIFO].

In this application the DMA DREQ/DACK is programmed to:

DREQx polarity = active low [EDMODx register - DR0POL]

DACKx polarity = active low [EDMODx register - DA0POL]

DREQx pulse = 4 Jclkcycles [EDMODx register - DRxPULS]

Technical Note 002

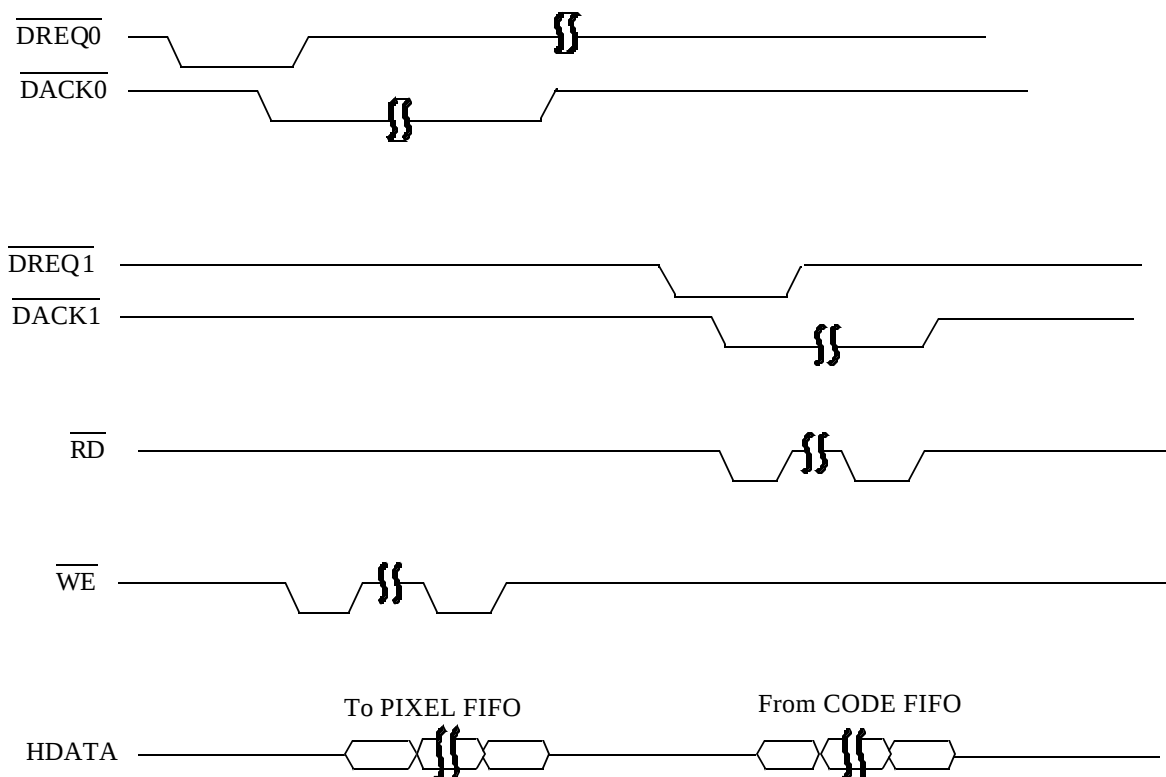


FIGURE 3. Data transfer in encode mode using DREQ/DACK DMA mode in burst transfer configuration

(8) Data transfer in decode mode

Active pulses on DREQ1 indicate that the ADV202 is ready to send a burst of data. The host should then initiate a burst data transfer to channel 0 (Pixel Data).

In this application the DMA DREQ/DACK is programmed to:

DREQx polarity = active low [EDMODx register - DR0POL]
DACKx polarity = active low [EDMODx register - DA0POL]
DREQx pulse = 4 Jclkcycles [EDMODx register - DRxPULS]
[refer to datasheet for more detail]

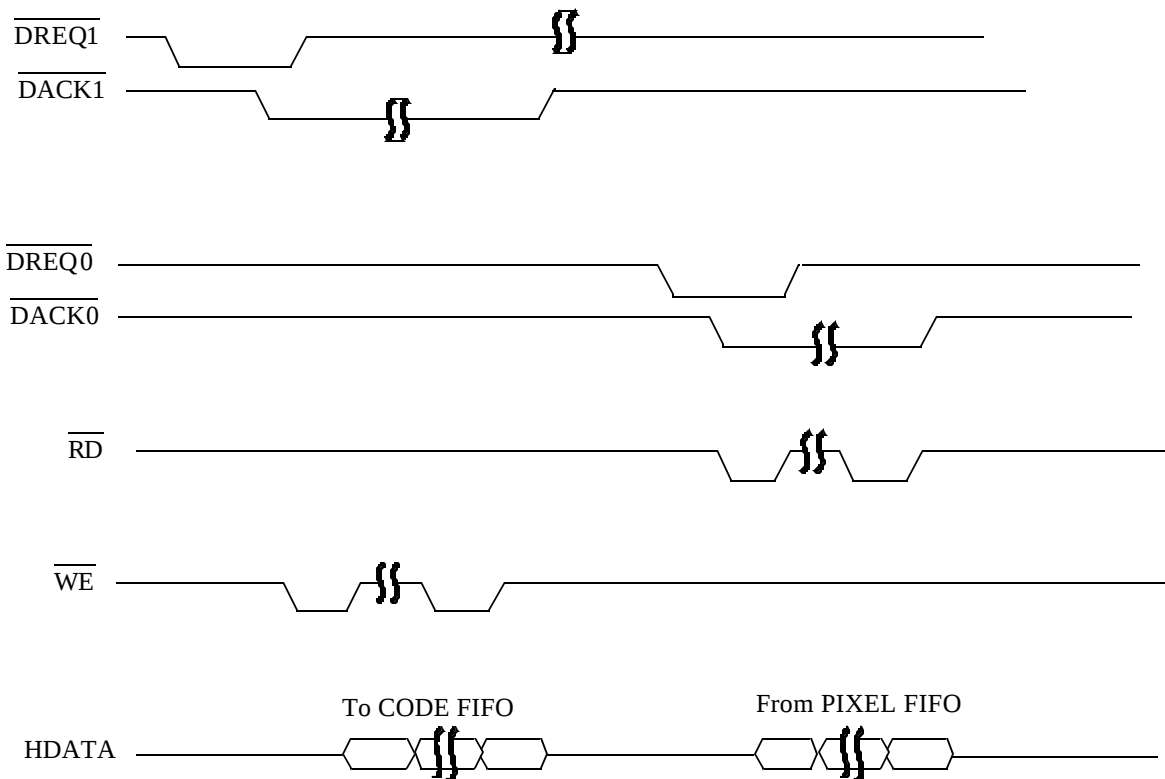


FIGURE 4. Data transfer in decode mode using DREQ/DACK DMA mode in burst transfer configuration