ADV212 JPEG2000

Programming Guide

Revision 2.0
February 26, 2007

Analog Devices, Inc.
One Technology Way
Norwood, Mass. 02062-9106

COPYRIGHT INFORMATION

DISCLAIMER

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

ABOUT THIS DOCUMENT

Purpose

This document provides detailed instructions for implementing specific applications by initializing and programming the ADV212 single-chip JPEG2000 codec. This document also provides an explanation of the ADV212 firmware parameters implemented in the ADV212.

It is assumed that the ADV212 host interface is in either 32-bit or 16-bit mode, that you know which interface mode you will use, and that the interface is connected properly so that all direct registers are accessible.

This document applies to the ADV212 engineering samples that are branded as follows:

- ADV212xx
- SURF®

Interface Modes

This document contains encode and decode routines for these interface modes:

- Normal host mode using the threshold registers
- Dedicated chip select (DCS) mode
- DMA DREQ/DACK modes
- JDATA mode
- Custom-specific mode
- Raw video mode

Audience

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

Recommended Reading

Before you initialize the part, you should be familiar with the following documents available at http://www.analog.com

Under ADV202 Technical Documentation:

- AN-790, *How To Use The ADV202*, gives an overview and description of the interface modes mentioned in this guide. This document is written for the ADV202, but the information also applies to the ADV212.

  This is the first document you should look at when you are determining how to use the ADV212.
- AN-799, *ADV202 Test Modes*, contains instructions to verify correct hardware configuration and functional register access for the ADV202 and ADV212.

Under ADV212 Technical Documentation:

- ADV212 datasheet — This document contains the package specifications and all electrical and timing specifications.
- *ADV212 User's Guide* — A reference guide for the ADV212, the *User's Guide* contains detailed description of the ADV212 functionality and register descriptions and an overview of application examples and firmware parameters, as well as software and hardware configurations.
- Firmware — Up-to-date firmware can be downloaded from http://www.analog.com under ADV212 Technical Documentation.
- Product Change Notices and Errata sheets

Conventions

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

Timing Diagrams

The timing diagrams in this document show relative timing only. Setup and hold times cannot be inferred from or implied by these diagrams. The ADV212 datasheet provides precise timing information.

Numbering Systems

Hexadecimal values are represented with the prefix "0x" followed by the value. For example: 0xFFFF.
Binary values are represented with the prefix "0b" followed by the value. For example: 0b1111.

Typographic Notation

The following typographic notations are used throughout this document:

Example	Description
$\overline{\text{SIGNAL}}$ SIGNAL/	An overbar or following slash indicates an active low signal.
[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).
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).
byte	An 8-bit data object
half-word	A 16-bit data object
word	A 32-bit data object

REVISION HISTORY

2/07—Revision 2.0

TABLE OF CONTENTS

TABLE OF FIGURES

BASIC ENCODE AND DECODE INITIALIZATION ROUTINES

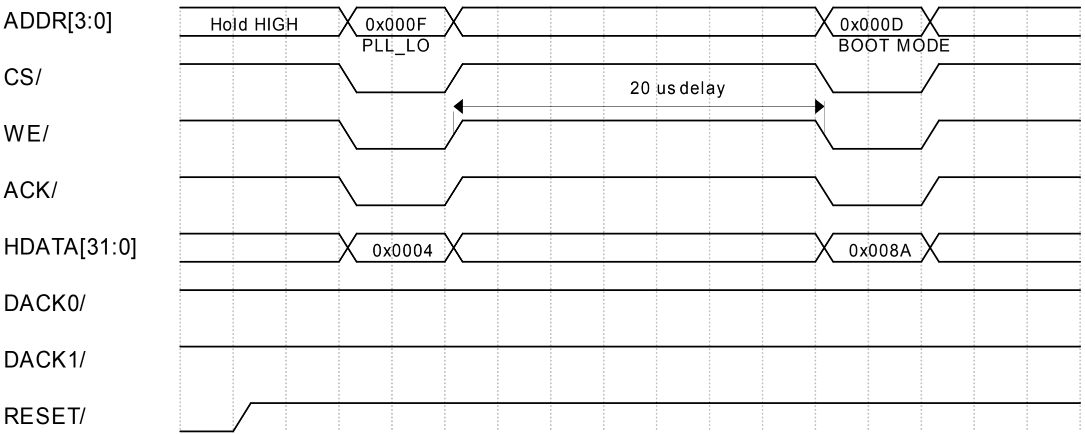
The ADV212 initialization routine follows a specific order of instructions. These instructions require direct register accesses.

To perform direct register accesses, the target system must be properly connected to the following pins on the ADV212:

- ADDR[3:0]—Up to 16 addresses
- $\overline{\text{CS}}$—Chip select
- $\overline{\text{WE}}$—Write enable
- $\overline{\text{RD}}$—Read enable
- $\overline{\text{ACK}}$—Acknowledge
- HDATA[15:0]—Read/write data half-word
- HDATA[31:16]—Optional extended data bus for 32-bit accesses

The target system must hold the state of the input pins ADDR, $\overline{\text{CS}}$, $\overline{\text{WE}}$, $\overline{\text{RD}}$, $\overline{\text{DACK}}$, and HDATA (for writes) until the ADV212 asserts the ACK signal (the ACK signal goes LOW). The access is not completed until the ACK is asserted. Thus for Read operations, the HDATA should not be sampled until after ACK is asserted. This is only necessary for asynchronous reads of the direct registers, and is not needed for DMA accesses. Also, the $\overline{\text{DACK0}}$ and $\overline{\text{DACK1}}$ pins must be held high at all times unless a true DMA or JDATA access occurs.

Figure 1 shows a normal host timing diagram with the pin settings required for operating the ADV212. The PLL_LO register and BOOT MODE register are set.



Setup and hold times cannot be inferred from and are not implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 1. Direct Register Access After Power-Up (Relative Timing)

Figure 2 shows a generic ADV212 encode flow. Normal host mode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register, followed by firmware load.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot, and then configure the BUSMODE and MMODE registers and application-specific registers. Next, you confirm that the correct firmware is loaded by reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. Encoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

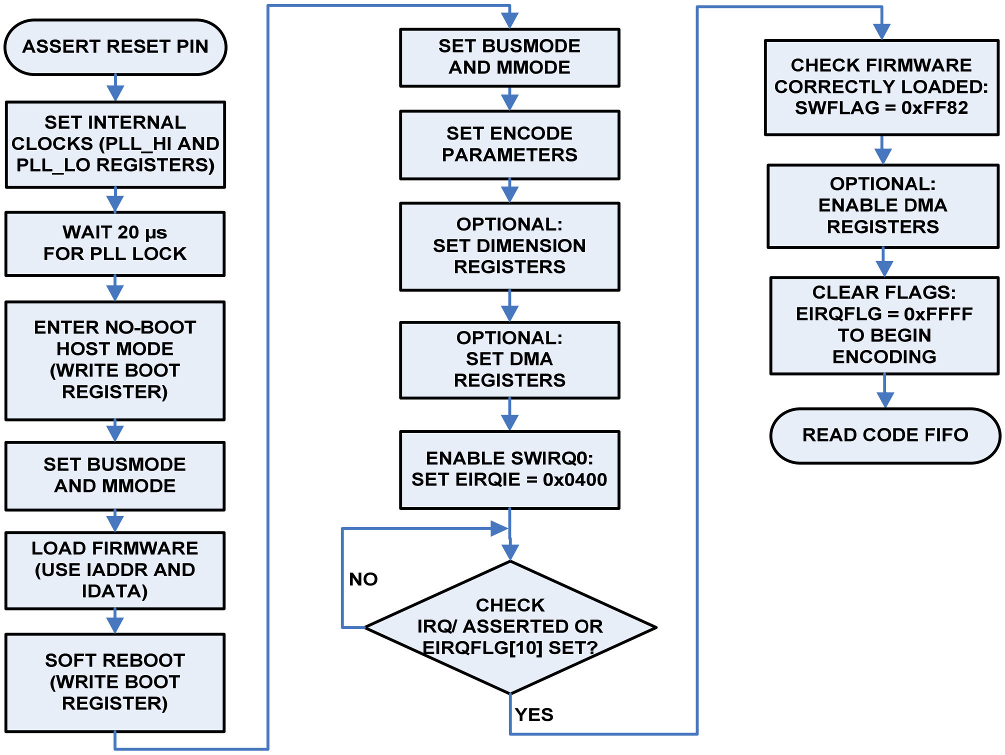

ASSERT RESET PIN
SET INTERNAL CLOCKS (PLL_HI AND PLL_LO REGISTERS)
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE (WRITE BOOT REGISTER)
SET BUSMODE AND MMODE
LOAD FIRMWARE (USE IADDR AND IDATA)
SOFT REBOOT (WRITE BOOT REGISTER)
SET BUSMODE AND MMODE
SET ENCODE PARAMETERS
OPTIONAL: SET DIMENSION REGISTERS
OPTIONAL: SET DMA REGISTERS
ENABLE SWIRQ0: SET EIRQIE = 0x0400
CHECK IRQ/ ASSERTED OR EIRQFLG[10] SET?
NO
YES
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFF82
OPTIONAL: ENABLE DMA REGISTERS
CLEAR FLAGS: EIRQFLG = 0xFFFF TO BEGIN ENCODING
READ CODE FIFO

Figure 2. General Encode Routine

Confirming Correct Firmware Load

There are two procedures for confirming whether the correct firmware is loaded just prior to executing the application, one that uses the ADV212 $\overline{\text{IRQ}}$ pin, and one that relies on polling. The version that relies on polling is used for systems that are not using the $\overline{\text{IRQ}}$ pin. Both procedures set up a handshake with the running program to make sure the program is executing correctly. If the procedure times out, then the program probably did not load correctly, or there was another problem in the boot procedure.

- Initialization Procedure Using the $\overline{\text{IRQ}}$ Pin:
    1. Write 0x0400 to EIRQIE (address 0x5) to unmask the SWIRQ0 interrupt (EIRQIE[10]).
    2. Wait for $\overline{\text{IRQ}}$ to be asserted (the pin goes LOW).
    3. Check that EIRQFLG[10] is set.
    4. Read the application ID from the SWFLAG register (address 0x7) to ensure the program has correctly initialized. For encode, SWFLAG is set to 0xFF82. For decode, SWFLAG is set to 0xFFA2.

- Initialization Procedure Using Polling:
    1. Write 0x0400 to EIRQIE (address 0x5) to unmask the SWIRQ0 interrupt.
    2. Poll EIRQFLG[10].
    3. Check that EIRQFLG[10] is set to ‘1’.
    4. Read the application ID from the SWFLAG register (address 0x7) to ensure the program has correctly initialized. For encode, SWFLAG is set to 0xFF82. For decode, SWFLAG is set to 0xFFA2.

Figure 3 shows a generic ADV212 decode flow. Normal host mode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register, followed by firmware load.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot, and then configure the BUSMODE and MMODE registers and application-specific registers. Next, you confirm that the correct firmware is loaded by reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions. Decoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

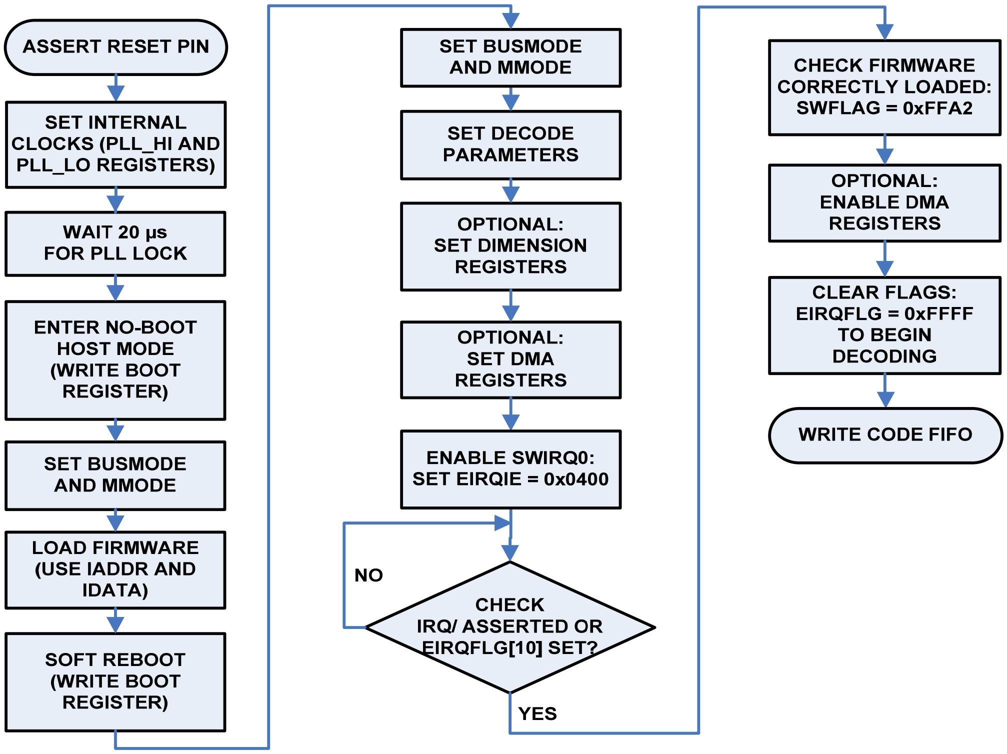

ASSERT RESET PIN
SET INTERNAL CLOCKS (PLL_HI AND PLL_LO REGISTERS)
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE (WRITE BOOT REGISTER)
SET BUSMODE AND MMODE
LOAD FIRMWARE (USE IADDR AND IDATA)
SOFT REBOOT (WRITE BOOT REGISTER)
SET BUSMODE AND MMODE
SET DECODE PARAMETERS
OPTIONAL: SET DIMENSION REGISTERS
OPTIONAL: SET DMA REGISTERS
ENABLE SWIRQ0: SET EIRQIE = 0x0400
NO
CHECK IRQ/ ASSERTED OR EIRQFLG[10] SET?
YES
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFFA2
OPTIONAL: ENABLE DMA REGISTERS
CLEAR FLAGS: EIRQFLG = 0xFFFF TO BEGIN DECODING
WRITE CODE FIFO

Figure 3. General Decode Routine

LCODE SIGNAL IN ENCODE MODE

When you use the ADV212 in encode mode, the firmware uses the SCOMM[4] pin (enabled by default) to enable the LCODE function. The LCODE signal is active for the last word of the block and indicates that the last 32-bit word of each field or frame is being read from the ADV212. For example, if you are using burst DMA DREQ/DACK mode with a burst length of 8, the ADV212 JPEG2000 output is generated in blocks of 8 x 32-bit words. The end-of-code (EOC) marker (0xFFD9) indicates the end of a field or frame in the JPEG2000 codestream. When the EOC marker falls in the middle of an 8-word block of data, zero-byte padding is added to complete the 8-word block.

The example below shows data being read from the ADV212 and the relative position of the LCODE indication:

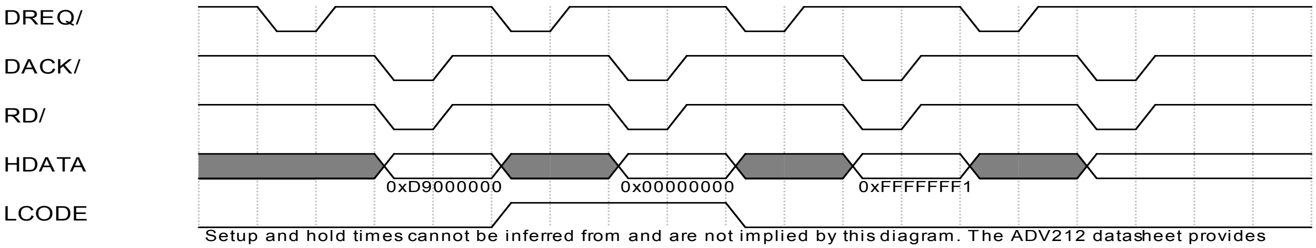
Example: Data from ADV212 CODE FIFO

```
0xXXXXXXXX
0xXXXXXXFF
0xD9000000 <-- EOC marker
0x00000000 <-- 8x32-bit boundary : LCODE is asserted and zero-byte padding added
0xFFFFFFF1 <-- Start of next field/frame
0xXXXXXXXX
```

LCODE (SCOMM[4]) Pin Timing

The LCODE (SCOMM[4]) pin timing is similar to the data pins (HDATA[31:0]), and it should be sampled on the rising edge of the $\overline{\text{RD}}$ signal.

Figure 4 shows the LCODE timing relative to the HDATA pins when the external DMA is set to single transfer mode. The DMA is programmed with the EDMODx registers.

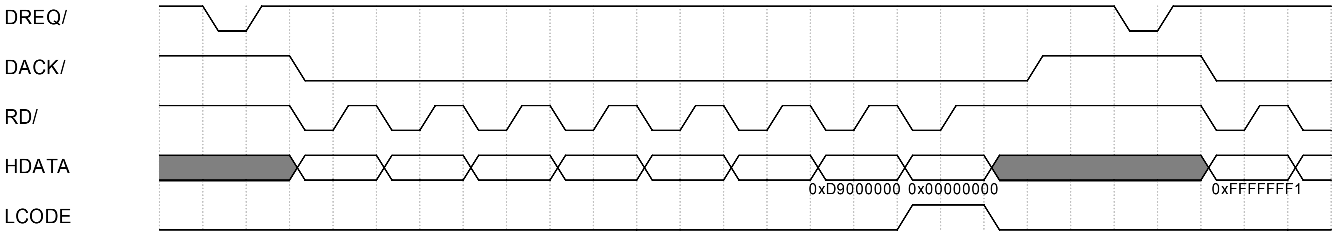


Setup and hold times cannot be inferred from and are not implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 4. LCODE Relative Timing Example (DMA Set to Single Transfer)

The LCODE indication is a mode-independent tag associated with the end of a field or frame. LCODE is also asserted when the CODE FIFO is read via indirect read access or when JDATA (streaming) mode is used. LCODE indicates the last 32-bit word of a field or frame. This means that when the host interface is set to 16-bit data width, LCODE is asserted for two read accesses. Similarly, when using 8-bit data width (including JDATA), LCODE is asserted for 4 accesses.

Figure 5 shows an example of LCODE timing relative to the HDATA pins when the external DMA is set to burst transfer mode (burst length = 8).



Setup and hold times cannot be inferred from and are not implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 5. LCODE Relative Timing Example (DMA Set to Burst Transfer)

32-BIT NORMAL HOST MODE ENCODE AND DECODE ROUTINES

Normal host mode accesses are used for direct register access, firmware loads, and parameter loads, and can also be used for transferring compressed data in normal host mode. Figure 6 shows sample pin settings for 32-bit normal host mode encode when you start the program and read compressed data out of the CODE FIFO.

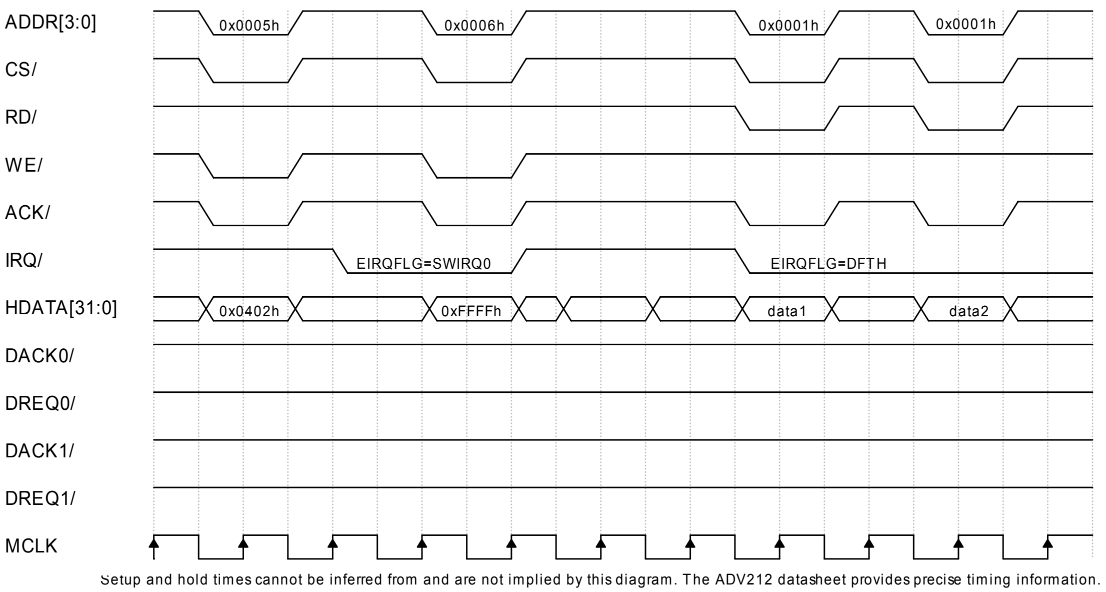


Figure 6. 32-bit Normal Host Mode Encode Relative Timing Example

Figure 7 shows sample pin settings for 32-bit normal host mode decode when you start the program and write compressed data into the CODE FIFO.

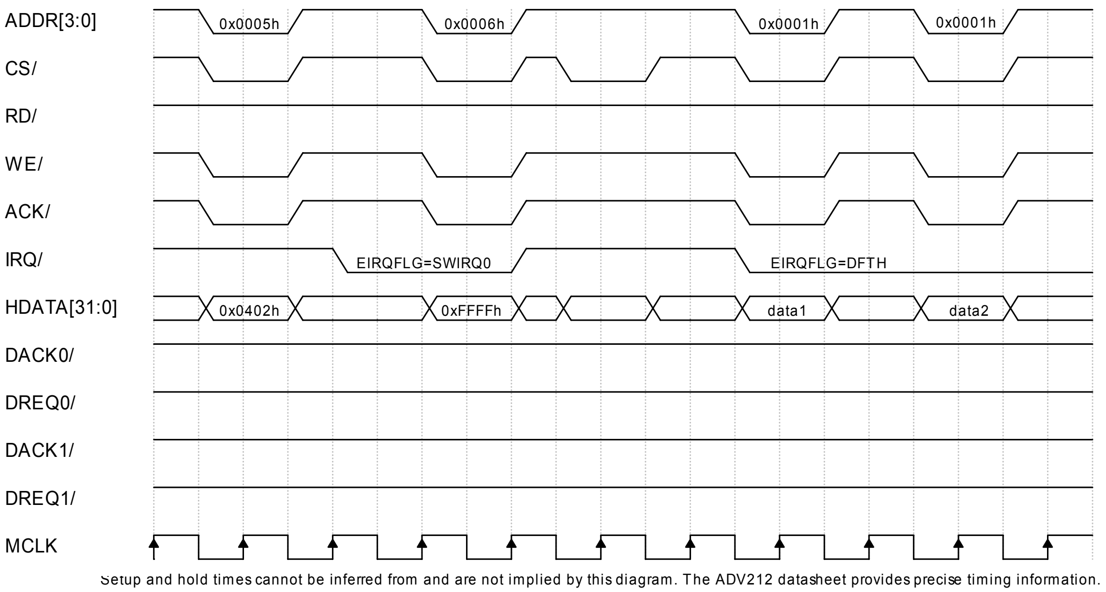
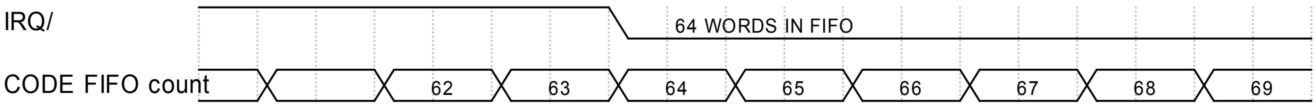


Figure 7. 32-bit Normal Host Mode Decode Relative Timing Example

CODE FIFO Threshold Register

The CODE FIFO threshold register, FFTHRC, is accessed at address location 0xFFFF141C. The FFTHRC register sets the threshold value for a threshold-specific interrupt that is asserted on the $\overline{\text{IRQ}}$ pin when the threshold conditions are met. This indicates that the ADV212 is ready for a compressed data transfer of the number of words programmed in the FFTHRC register.

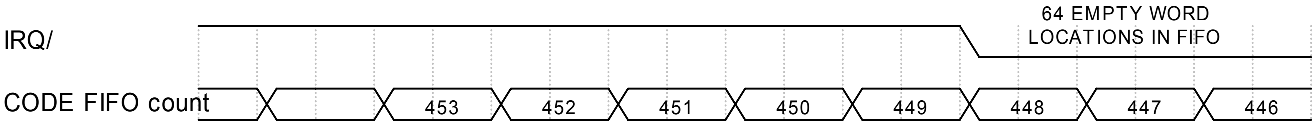
Figure 8 shows an encode mode example where the $\overline{\text{IRQ}}$ is asserted when the number of data words in the FIFO is greater than or equal to the threshold value.



Setup and hold times cannot be inferred from and are not implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 8. $\overline{\text{IRQ}}$ Asserted in Encode Mode (Threshold Set to 64 Words)

Figure 9 shows a decode mode example of a 512-entry FIFO. The $\overline{\text{IRQ}}$ is asserted when the number of *available* word locations is greater than or equal to the threshold value.



Setup and hold times cannot be inferred from and are not implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 9. $\overline{\text{IRQ}}$ Asserted in Decode Mode (Threshold Set to 64 Words)

Figure 10 shows an example of a 32-bit normal host mode encode flow that uses a 64-word threshold for compressed data out. Normal host mode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 32-bit host interface and a 32-bit wide data bus. The MMODE register is set to enable 32-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configuring the BUSMODE and MMODE registers and any application-specific registers. You set the encode parameters with the IADDR and IDATA registers. IADDR must be set to the memory location 0x00057F00 and IDATA must be written with the encode parameters. For more information on setting the encode parameters, see Table 8 on page 43. Next, you set the threshold register. Then, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions. Encoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

You can initiate data transfer after you check that the $\overline{\text{IRQ}}$ pin is asserted and that EIRQFLG[1] (DFTH) is set to 1. A setting of EIRQFLG[1] = 1 indicates that the CODE FIFO contains data and that the threshold condition for the CODE FIFO has been met.

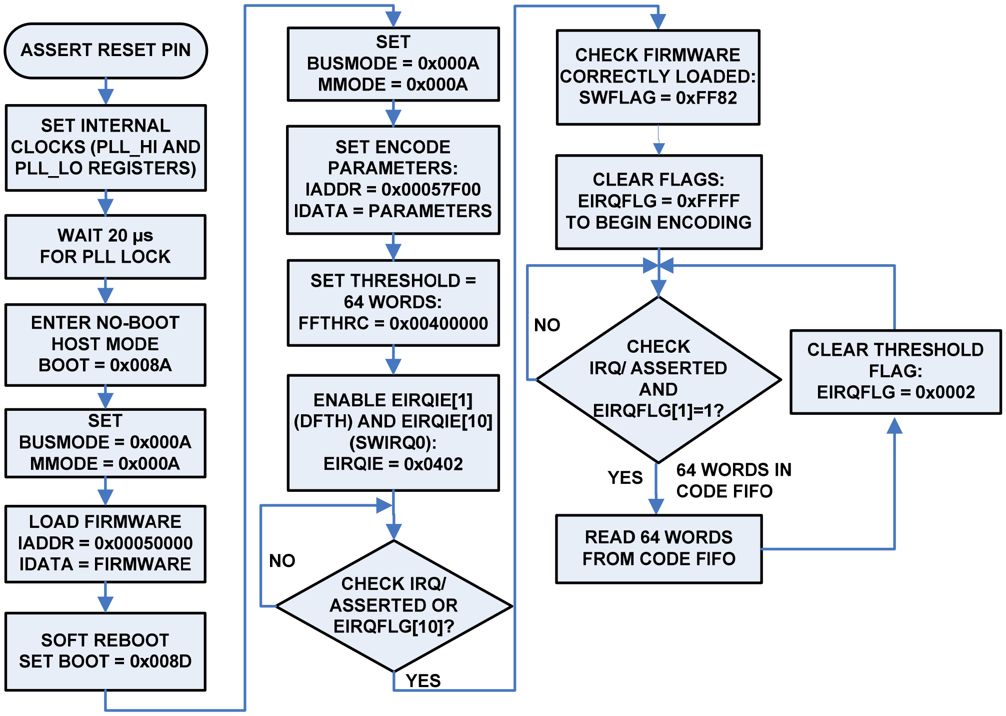

ASSERT RESET PIN
SET INTERNAL CLOCKS (PLL_HI AND PLL_LO REGISTERS)
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE BOOT = 0x008A
SET BUSMODE = 0x000A MMODE = 0x000A
LOAD FIRMWARE IADDR = 0x00050000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
SET BUSMODE = 0x000A MMODE = 0x000A
SET ENCODE PARAMETERS: IADDR = 0x00057F00 IDATA = PARAMETERS
SET THRESHOLD = 64 WORDS: FFTHRC = 0x00400000
ENABLE EIRQIE[1] (DFTH) AND EIRQIE[10] (SWIRQ0): EIRQIE = 0x0402
NO
CHECK IRQ/ ASSERTED OR EIRQFLG[10]?
YES
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFF82
CLEAR FLAGS: EIRQFLG = 0xFFFF TO BEGIN ENCODING
NO
CHECK IRQ/ ASSERTED AND EIRQFLG[1]=1?
YES
64 WORDS IN CODE FIFO
READ 64 WORDS FROM CODE FIFO
CLEAR THRESHOLD FLAG: EIRQFLG = 0x0002

Figure 10. Example of 32-Bit Normal Host Mode Encode Routine With Threshold

Figure 11 shows an example of a 32-bit normal host mode decode flow that uses a 64-word threshold for compressed data in. Normal host mode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 32-bit host interface and a 32-bit wide data bus. The MMODE register is set to enable 32-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configuring the BUSMODE and MMODE registers and any application-specific registers. You set the decode parameters with the IADDR and IDATA registers. IADDR must be set to the memory location 0x00057F00 and IDATA must be written with the decode parameters. For more information on setting the decode parameters, see Table 9 on page 47. Next, you set the threshold register. Then, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions. Decoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

You can initiate data transfer after you check that the $\overline{\text{IRQ}}$ pin is asserted and that EIRQFLG[1] (DFTH) is set to 1. A setting of EIRQFLG[1] = 1 indicates that the CODE FIFO contains data and that the threshold condition for the CODE FIFO has been met.

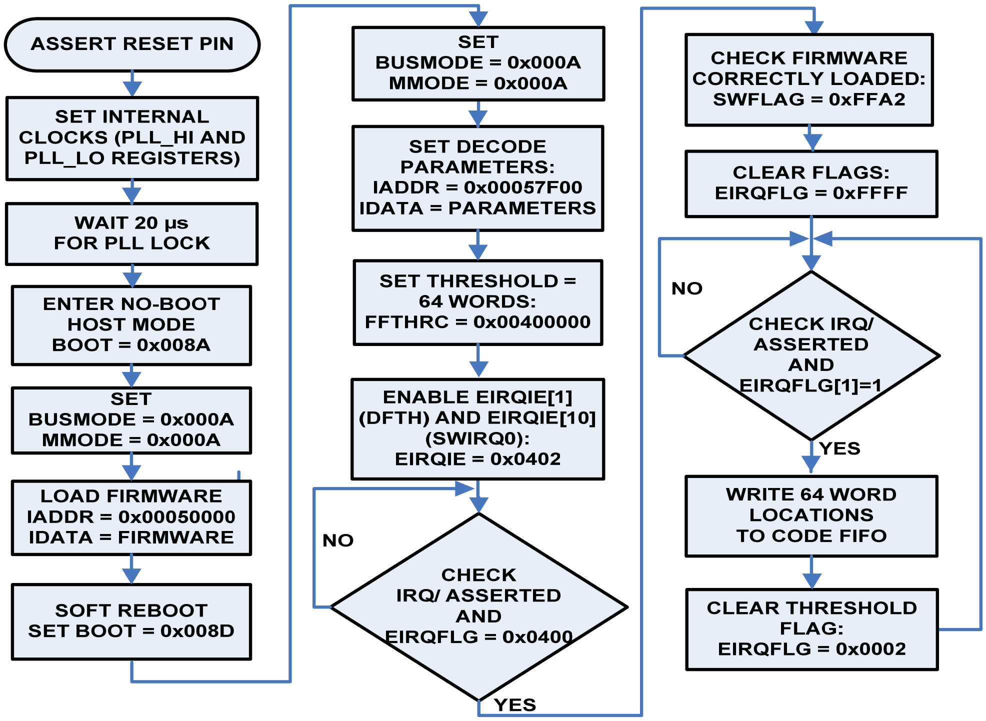

ASSERT RESET PIN
SET INTERNAL CLOCKS (PLL_HI AND PLL_LO REGISTERS)
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE BOOT = 0x008A
SET BUSMODE = 0x000A MMODE = 0x000A
LOAD FIRMWARE IADDR = 0x00050000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
SET BUSMODE = 0x000A MMODE = 0x000A
SET DECODE PARAMETERS: IADDR = 0x00057F00 IDATA = PARAMETERS
SET THRESHOLD = 64 WORDS: FFTHRC = 0x00400000
ENABLE EIRQIE[1] (DFTH) AND EIRQIE[10] (SWIRQ0): EIRQIE = 0x0402
NO
CHECK IRQ/ ASSERTED AND EIRQFLG = 0x0400
YES
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFFA2
CLEAR FLAGS: EIRQFLG = 0xFFFF
NO
CHECK IRQ/ ASSERTED AND EIRQFLG[1]=1
YES
WRITE 64 WORD LOCATIONS TO CODE FIFO
CLEAR THRESHOLD FLAG: EIRQFLG = 0x0002

Figure 11. Example of 32-Bit Normal Host Mode Decode Routine With Threshold

16-BIT NORMAL HOST MODE ENCODE AND DECODE ROUTINES

If you are using 16-bit normal host mode, you must take the following into account:

- When using a 16-bit host-control bus to access 32-bit resources, such as when reading or writing to the IADDR or IDATA registers, you must use the STAGE register. See the *ADV212 User's Guide* for more details.
- When performing 32-bit accesses in 16-bit mode when reading internal memory (0x50000 to 0x57FFF) and when using indirect address auto-increment, the data read-back value is halfword-swapped. That is, the write order is big-endian while the read order is little-endian.
- The FIFO count is in 32-bit words even when 16-bit mode is used. A threshold setting of 64 is met if 128 16-bit data words are present in the CODE FIFO (in encode mode) or 128 16-bit locations are available in the CODE FIFO (in decode mode).

See Table 7 on page 43 for a list of BUSMODE and MMODE register settings.

Figure 12 shows an example of a 16-bit normal host mode encode flow that uses a threshold for compressed data output. Normal host mode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 16-bit host interface and a 16-bit wide data bus. The MMODE register is set to enable 16-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configuring the BUSMODE and MMODE registers and any application-specific registers. You set the encode parameters with the IADDR and IDATA registers. IADDR must be set to the memory location 0x00057F00 and IDATA must be written with the encode parameters. For more information on setting the encode parameters, see Table 8 on page 43. Next, you set the threshold register. Then, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions. Encoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

You can initiate data transfer after you check that the $\overline{\text{IRQ}}$ pin is asserted and EIRQFLG[1] (DFTH) is set to 1. A setting of EIRQFLG[1] = 1 indicates that the CODE FIFO contains data and that the threshold condition for the CODE FIFO has been met. Reading 64 32-bit words from the CODE FIFO in 16-bit data mode requires 128 accesses.

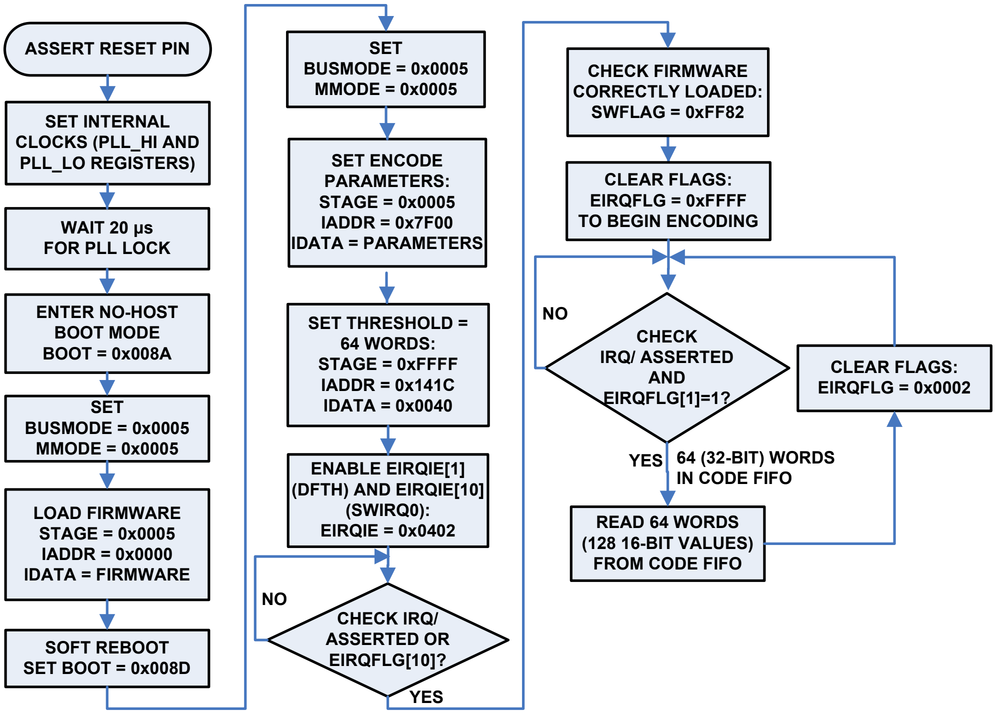

ASSERT RESET PIN
SET INTERNAL CLOCKS (PLL_HI AND PLL_LO REGISTERS)
WAIT 20 µs FOR PLL LOCK
ENTER NO-HOST BOOT MODE BOOT = 0x008A
SET BUSMODE = 0x0005 MMODE = 0x0005
LOAD FIRMWARE STAGE = 0x0005 IADDR = 0x0000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
SET BUSMODE = 0x0005 MMODE = 0x0005
SET ENCODE PARAMETERS: STAGE = 0x0005 IADDR = 0x7F00 IDATA = PARAMETERS
SET THRESHOLD = 64 WORDS: STAGE = 0xFFFF IADDR = 0x141C IDATA = 0x0040
ENABLE EIRQIE[1] (DFTH) AND EIRQIE[10] (SWIRQ0): EIRQIE = 0x0402
NO
CHECK IRQ/ ASSERTED OR EIRQFLG[10]?
YES
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFF82
CLEAR FLAGS: EIRQFLG = 0xFFFF TO BEGIN ENCODING
NO
CHECK IRQ/ ASSERTED AND EIRQFLG[1]=1?
CLEAR FLAGS: EIRQFLG = 0x0002
YES
64 (32-BIT) WORDS IN CODE FIFO
READ 64 WORDS (128 16-BIT VALUES) FROM CODE FIFO

Figure 12. Example of 16-bit Normal Host Mode Encode Routine With Threshold

Figure 13 shows an example of a 16-bit normal host mode decode flow that uses a threshold for compressed data input. Normal host mode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 16-bit host interface and a 16-bit wide data bus. The MMODE register is set to enable 16-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configuring the BUSMODE and MMODE registers and any application-specific registers. You set the decode parameters with the IADDR and IDATA registers. IADDR must be set to the memory location 0x00057F00 and IDATA must be written with the decode parameters. For more information on setting the decode parameters, see Table 9 on page 47. Next, you set the threshold register. Then, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions. Decoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

You can initiate data transfer after you check that the $\overline{\text{IRQ}}$ pin is asserted and EIRQFLG[1] (DFTH) is set to 1. A setting of EIRQFLG[1] = 1 indicates that the CODE FIFO contains data and that the threshold condition for the CODE FIFO has been met. Writing 64 32-bit words to the CODE FIFO in 16-bit data mode requires 128 accesses.

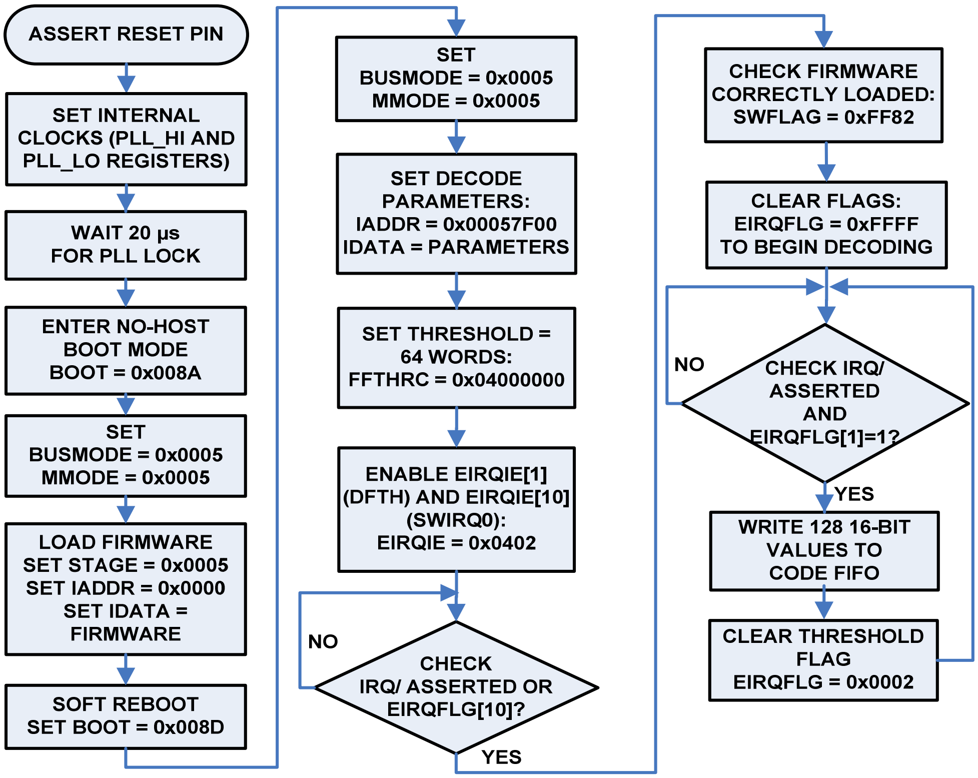

ASSERT RESET PIN
SET INTERNAL CLOCKS (PLL_HI AND PLL_LO REGISTERS)
WAIT 20 µs FOR PLL LOCK
ENTER NO-HOST BOOT MODE BOOT = 0x008A
SET BUSMODE = 0x0005 MMODE = 0x0005
LOAD FIRMWARE SET STAGE = 0x0005 SET IADDR = 0x0000 SET IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
SET BUSMODE = 0x0005 MMODE = 0x0005
SET DECODE PARAMETERS: IADDR = 0x00057F00 IDATA = PARAMETERS
SET THRESHOLD = 64 WORDS: FFTHRC = 0x04000000
ENABLE EIRQIE[1] (DFTH) AND EIRQIE[10] (SWIRQ0): EIRQIE = 0x0402
NO
CHECK IRQ/ ASSERTED OR EIRQFLG[10]?
YES
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFF82
CLEAR FLAGS: EIRQFLG = 0xFFFF TO BEGIN DECODING
NO
CHECK IRQ/ ASSERTED AND EIRQFLG[1]=1?
YES
WRITE 128 16-BIT VALUES TO CODE FIFO
CLEAR THRESHOLD FLAG EIRQFLG = 0x0002

Figure 13. Example of 16-bit Normal Host Mode Decode Routine With Threshold

DMA MODES

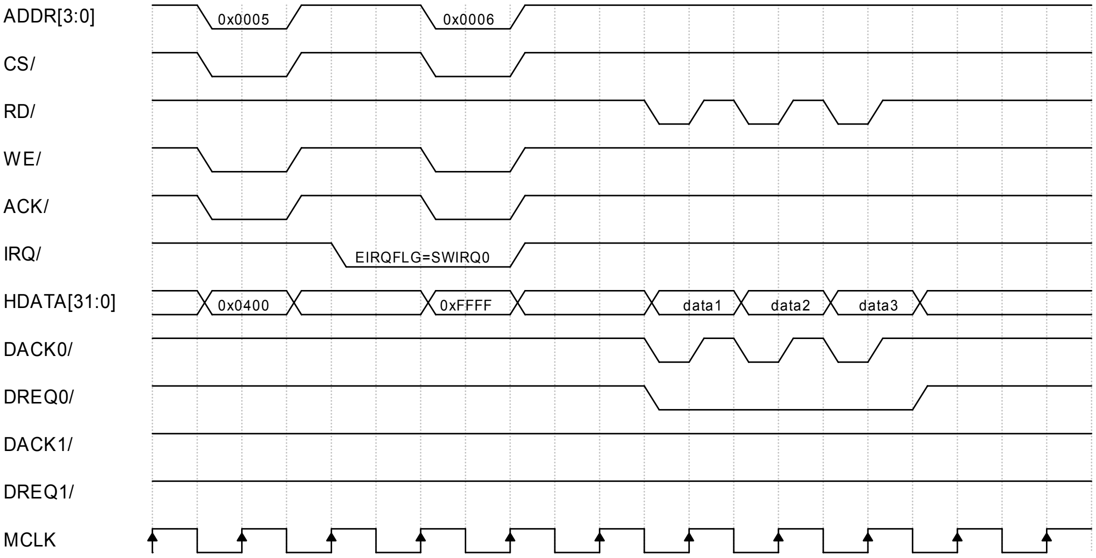
This document provides examples of two types of DMA modes: dedicated chip select (DCS) mode and DREQ/DACK mode. You use an EDMODx register to select a DMA channel. DMA provides fast, low-overhead data transfer between the host processor and the ADV212 data FIFOs.

In general, only one DMA channel is used for compressed data transfer, so only DMA Channel 0 is used and only the EDMOD0 register is configured.

32-bit DCS Encode and Decode Routines

In DCS mode, direct register configuration, firmware loads, and parameter loads function as in normal host mode. DCS mode provides information on the data FIFOs based on the FIFO threshold registers (FFTHRP, FFTHRC, FFTHRA). The FSRQx (DREQx pins) signals are asserted when the number of data words or spaces in the FIFOs is greater than the value in the corresponding threshold register.

Figure 14 shows example pin settings for DCS encode mode when you start the program and read data from the CODE FIFO. In encode mode, DREQ0 is asserted as long as there is data in the CODE FIFO.



Setup and hold times cannot be inferred from and are not implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 14. 32-bit DCS Encode Mode Relative Timing Example

Figure 15 shows an example of a 32-bit DCS mode encode flow. DCS mode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 32-bit host interface and a 32-bit wide data bus. The MMODE register is set to enable 16-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configure the BUSMODE and MMODE registers and any application-specific registers. You set the encode parameters with the IADDR and IDATA registers. IADDR must be set to the memory location 0x00057F00 and IDATA must be written with the encode parameters. For more information on setting the encode parameters, see Table 8 on page 43. Next, you set the EDMODx register. The EDMODx register configures the ADV212 to use external DMA channels to transfer compressed data. Then, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions.

The CODE FIFO threshold register, FFTHRC, is also set in DCS mode. Encoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

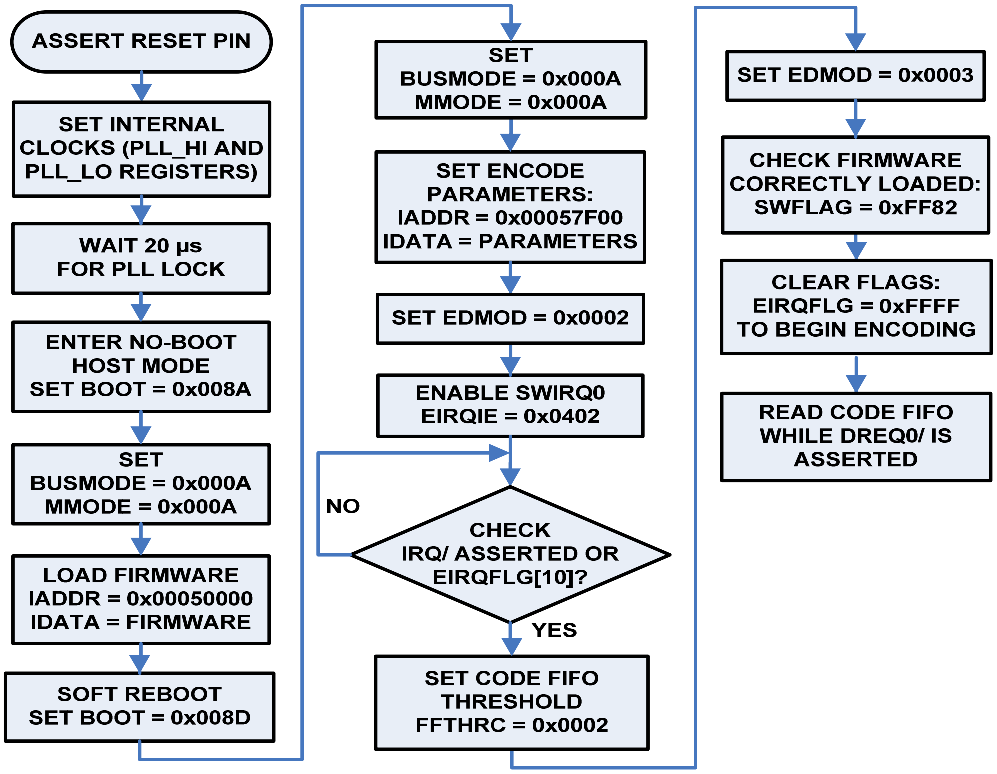
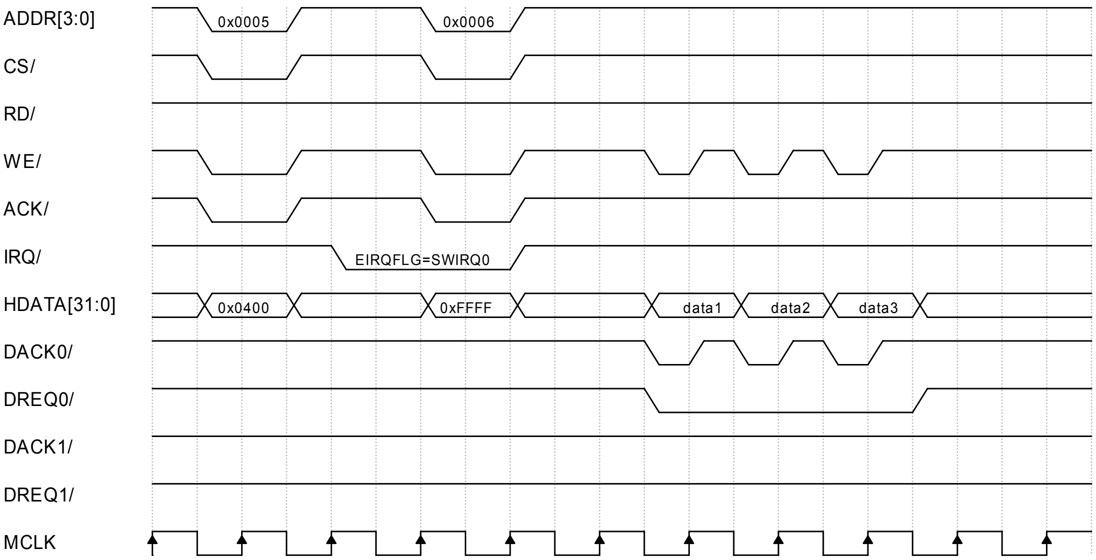

ASSERT RESET PIN
SET INTERNAL CLOCKS (PLL_HI AND PLL_LO REGISTERS)
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE SET BOOT = 0x008A
SET BUSMODE = 0x000A MMODE = 0x000A
LOAD FIRMWARE IADDR = 0x00050000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
SET BUSMODE = 0x000A MMODE = 0x000A
SET ENCODE PARAMETERS: IADDR = 0x00057F00 IDATA = PARAMETERS
SET EDMOD = 0x0002
ENABLE SWIRQ0 EIRQIE = 0x0402
NO
CHECK IRQ/ ASSERTED OR EIRQFLG[10]?
YES
SET CODE FIFO THRESHOLD FFTHRC = 0x0002
SET EDMOD = 0x0003
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFF82
CLEAR FLAGS: EIRQFLG = 0xFFFF TO BEGIN ENCODING
READ CODE FIFO WHILE DREQ0/ IS ASSERTED

Figure 15. Example of 32-bit DCS Mode Encode Routine

Figure 16 shows example pin settings for DCS decode mode when you start the program and write data to the CODE FIFO. In decode mode, $\overline{\text{DREQ0}}$ is asserted as long as there is space available in the CODE FIFO to take in data.



Setup and hold times cannot be inferred from and are not implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 16. 32-bit DCS Mode Decode Relative Timing Example

Figure 17 shows an example of a 32-bit DCS mode decode flow. DCS mode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 32-bit host interface and a 32-bit wide data bus. The MMODE register is set to enable 32-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configure the BUSMODE and MMODE registers and any application-specific registers. You set the decode parameters with the IADDR and IDATA registers. IADDR must be set to the memory location 0x00057F00 and IDATA must be written with the decode parameters. For more information on setting the decode parameters, see Table 9 on page 47. Next, you set the EDMODx register. The EDMODx register configures the ADV212 to use external DMA channels to transfer compressed data. Then, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions.

The CODE FIFO threshold register, FFTHRC, is also set in DCS mode. Decoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

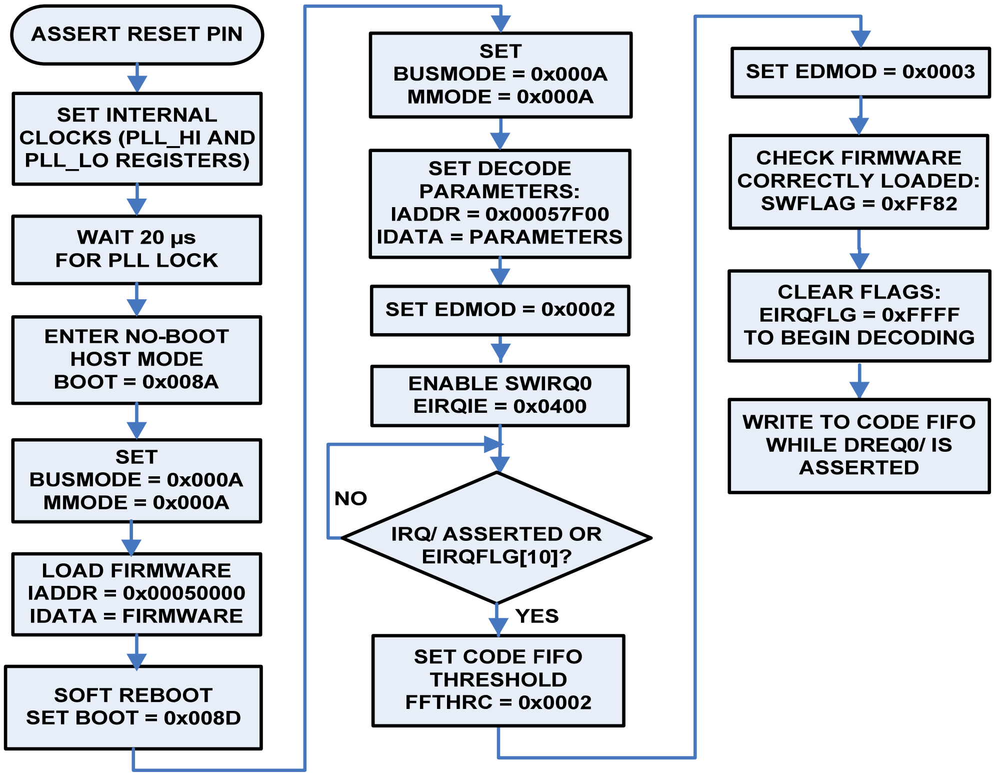

ASSERT RESET PIN
SET INTERNAL CLOCKS (PLL_HI AND PLL_LO REGISTERS)
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE BOOT = 0x008A
SET BUSMODE = 0x000A MMODE = 0x000A
LOAD FIRMWARE IADDR = 0x00050000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
SET BUSMODE = 0x000A MMODE = 0x000A
SET DECODE PARAMETERS: IADDR = 0x00057F00 IDATA = PARAMETERS
SET EDMOD = 0x0002
ENABLE SWIRQ0 EIRQIE = 0x0400
NO
IRQ/ ASSERTED OR EIRQFLG[10]?
YES
SET CODE FIFO THRESHOLD FFTHRC = 0x0002
SET EDMOD = 0x0003
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFF82
CLEAR FLAGS: EIRQFLG = 0xFFFF TO BEGIN DECODING
WRITE TO CODE FIFO WHILE DREQ0/ IS ASSERTED

Figure 17. Example of 32-bit DCS Mode Decode Routine

32-bit DMA DREQ/DACK Modes Encode and Decode Routines

In 32-bit DMA DREQ/DACK modes, direct register configuration, firmware loads, and parameter loads function as in normal host mode. There are two types of DMA DREQ/DACK mode, single and burst. You can see the ADV212 data sheet for detailed timing diagrams of the following modes:

- Single DMA DREQ/DACK mode—DREQ is asserted for every data access to the data FIFO.
- Burst DMA DREQ/DACK mode—DREQ is asserted once at the beginning of the burst.

For each of these modes, the DREQ pulse width can be configured using EDMOD0[14:11], the DR0PULS bit field:

- When DR0PULS is set to 0b0000, DREQ is asserted until the host responds by asserting DACK and RD (for encode) or WE (for decode).
- When DR0PULS is set to a non-zero value (0b0001 to 0b1111), this represents the DREQ signal pulse in terms of JCLK cycles. JCLK is the ADV212 internal processing clock. JCLK is configured by the PLL register settings. For more information on JCLK and the PLL registers, see the ADV212 datasheet.

In single DMA DREQ/DACK encode mode, the ADV212 asserts DREQ when there is at least one data transfer available in the assigned FIFO. In decode mode, the ADV212 asserts DREQ when the ADV212 is ready to receive at least one data transfer to the assigned FIFO.

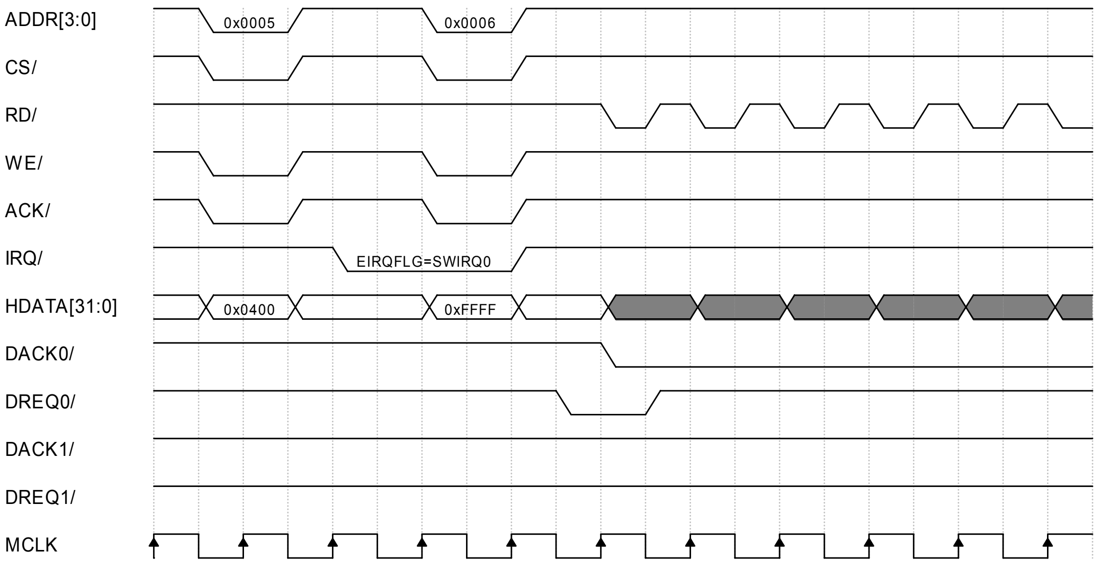
In burst DMA DREQ/DACK encode mode, the ADV212 asserts DREQ as soon as enough data, as set by the burst size, is present in the assigned FIFO. In decode mode, the ADV212 asserts DREQ when the ADV212 is ready to receive data, as set by the burst size, to the assigned FIFO. (Burst size is set in EDMOD0[8:6].)

Table 1 shows examples of EDMOD0 register settings for various DMA DREQ/DACK modes and configurations.

Table 1. DMA CONFIGURATION PARAMETERS AND EDMOD0 REGISTER SETTINGS

Routine	DREQ Pulse Configuration	DMA Mode	Burst Cycles	EDMOD0
Encode	Controlled by DACK and RD	Single	n/a	0x000A
Encode	Fixed pulse width to 4 JCLK cycles	Single	n/a	0x200A
Decode	Controlled by DACK and WE	Single	n/a	0x000A
Decode	Fixed pulse width to 4 JCLK cycles	Single	n/a	0x200A
Encode	Controlled by DACK and RD	Burst	16	0x0052
Encode	Fixed pulse width to 4 JCLK cycles	Burst	16	0x2052
Decode	Controlled by DACK and WE	Burst	16	0x0052
Decode	Fixed pulse width to 4 JCLK cycles	Burst	16	0x2052

Figure 18 shows example pin settings for DMA DREQ/DACK burst mode encode relative timing when you start the program and read the first burst out of the CODE FIFO (EDMOD0 = 0x0012, set for a burst length of 8):



Setup and hold times cannot be inferred from and are not implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 18. DMA DREQ/DACK Burst Mode Encode Relative Timing Example

Figure 19 shows an example of a 32-bit DMA DREQ/DACK burst mode encode flow. DMA DREQ/DACK burst mode encode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 32-bit host interface and a 32-bit wide data bus. The MMODE register is set to enable 16-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configure the BUSMODE and MMODE registers and any application-specific registers. You set the encode parameters with the IADDR and IDATA registers. IADDR must be set to the memory location 0x00057F00 and IDATA must be written with the encode parameters. For more information on setting the encode parameters, see Table 8 on page 43. Next, you set the EDMODx register. The EDMODx register configures the ADV212 to use external DMA channels to transfer compressed data. Then, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions. Encoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

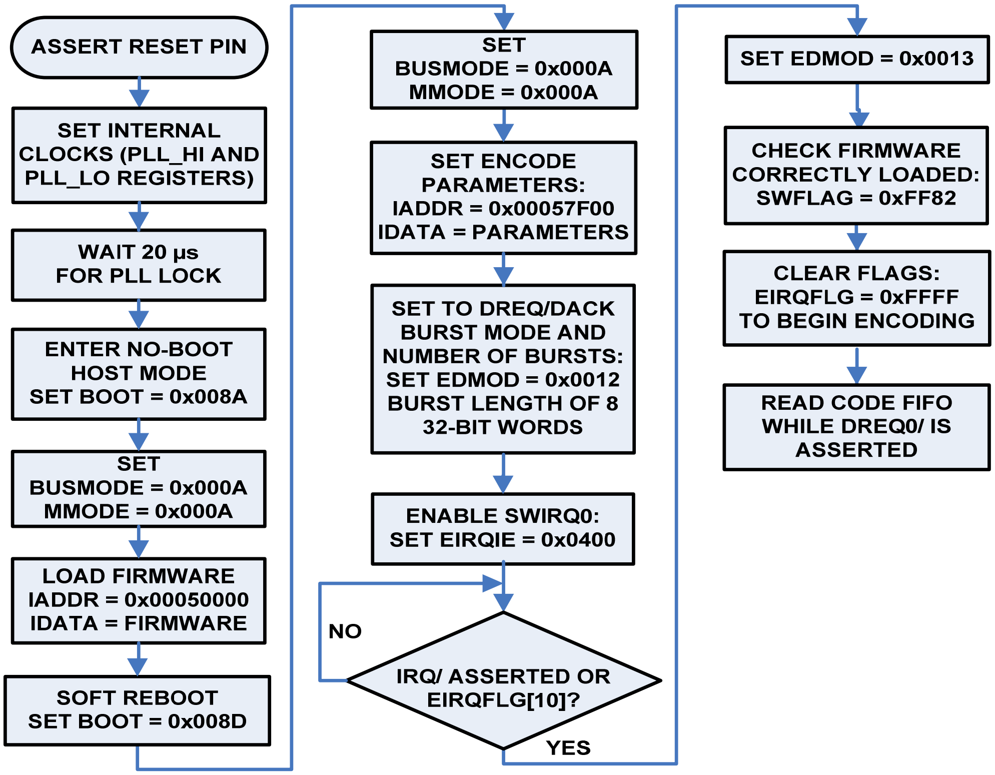
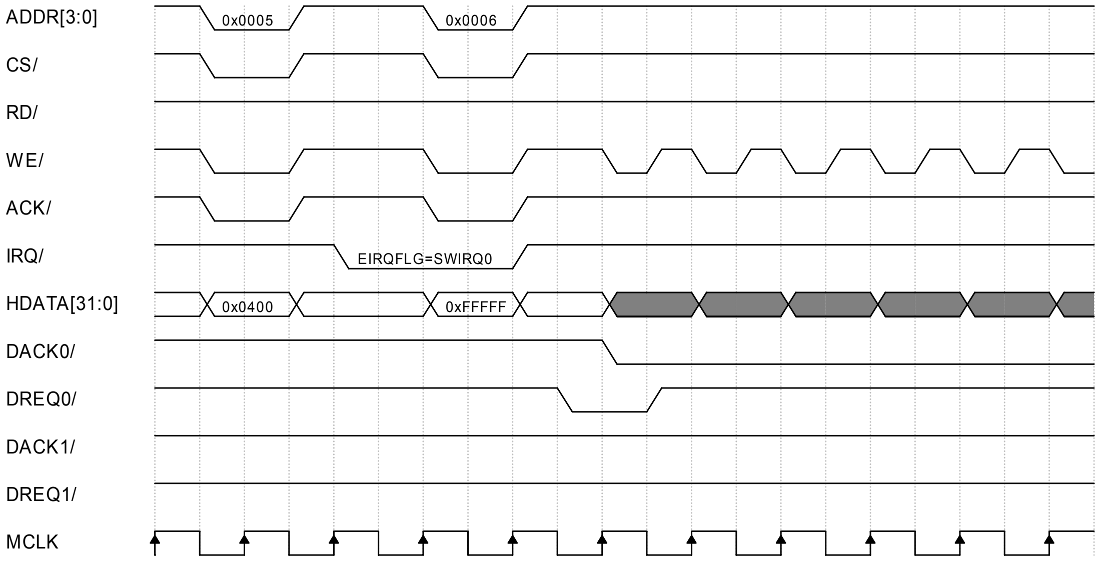

ASSERT RESET PIN
SET INTERNAL CLOCKS (PLL_HI AND PLL_LO REGISTERS)
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE SET BOOT = 0x008A
SET BUSMODE = 0x000A MMODE = 0x000A
LOAD FIRMWARE IADDR = 0x00050000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
SET BUSMODE = 0x000A MMODE = 0x000A
SET ENCODE PARAMETERS: IADDR = 0x00057F00 IDATA = PARAMETERS
SET TO DREQ/DACK BURST MODE AND NUMBER OF BURSTS: SET EDMOD = 0x0012 BURST LENGTH OF 8 32-BIT WORDS
ENABLE SWIRQ0: SET EIRQIE = 0x0400
NO
IRQ/ ASSERTED OR EIRQFLG[10]?
YES
SET EDMOD = 0x0013
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFF82
CLEAR FLAGS: EIRQFLG = 0xFFFF TO BEGIN ENCODING
READ CODE FIFO WHILE DREQ0/ IS ASSERTED

Figure 19. Example of DMA DREQ/DACK Burst Mode Encode Routine

Figure 20 shows example pin settings for DMA DREQ/DACK burst mode decode relative timing when you start the program and write the first burst into the CODE FIFO (EDMOD0 = 0x0012, set for a burst length of 8):



Setup and hold times cannot be inferred from and are not implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 20. DMA DREQ/DACK Burst Decode Relative Timing Example

Figure 21 shows an example of a 32-bit DMA DREQ/DACK burst mode decode flow. DMA DREQ/DACK burst mode decode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 32-bit host interface and a 32-bit data bus. The MMODE register is set to enable 32-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configure the BUSMODE and MMODE registers and any application-specific registers. You set the decode parameters with the IADDR and IDATA registers. IADDR must be set to the memory location 0x00057F00 and IDATA must be written with the decode parameters. For more information on setting the decode parameters, see Table 9 on page 47. Next, you set the EDMODx register. The EDMODx register configures the ADV212 to use external DMA channels to transfer compressed data. Then, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions. Decoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

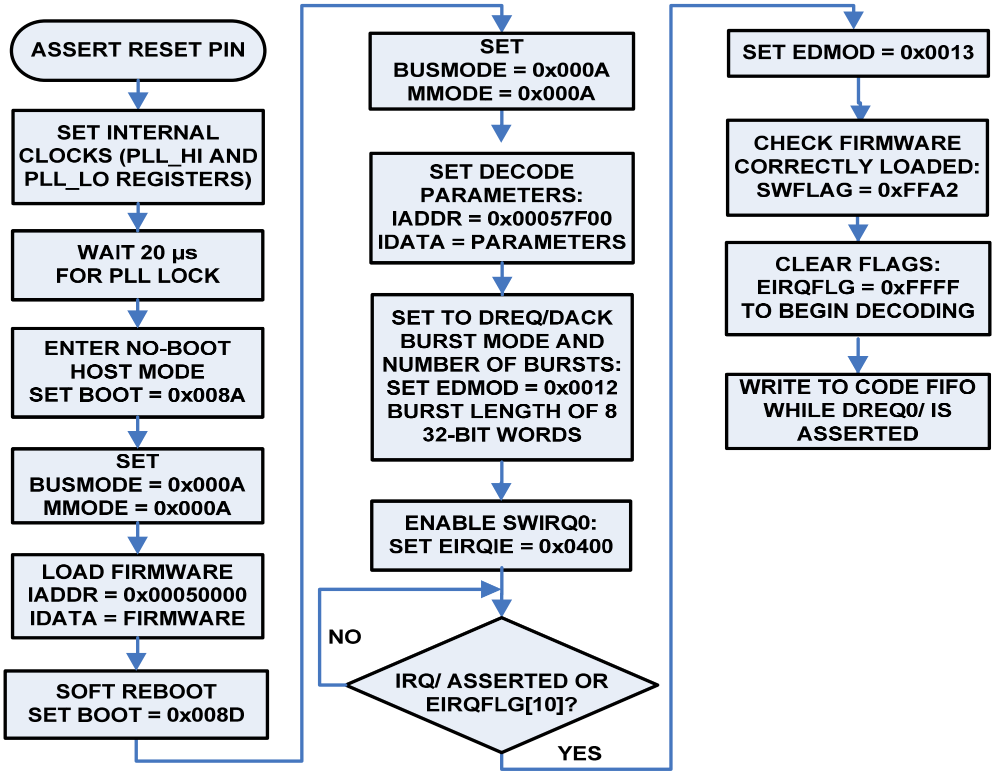

ASSERT RESET PIN
SET INTERNAL CLOCKS (PLL_HI AND PLL_LO REGISTERS)
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE SET BOOT = 0x008A
SET BUSMODE = 0x000A MMODE = 0x000A
LOAD FIRMWARE IADDR = 0x00050000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
SET BUSMODE = 0x000A MMODE = 0x000A
SET DECODE PARAMETERS: IADDR = 0x00057F00 IDATA = PARAMETERS
SET TO DREQ/DACK BURST MODE AND NUMBER OF BURSTS: SET EDMOD = 0x0012 BURST LENGTH OF 8 32-BIT WORDS
ENABLE SWIRQ0: SET EIRQIE = 0x0400
NO
IRQ/ ASSERTED OR EIRQFLG[10]?
YES
SET EDMOD = 0x0013
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFFA2
CLEAR FLAGS: EIRQFLG = 0xFFFF TO BEGIN DECODING
WRITE TO CODE FIFO WHILE DREQ0/ IS ASSERTED

Figure 21. Example of DREQ/DACK DMA Burst Mode Decode Routine

16-BIT HOST JDATA MODE ENCODE AND DECODE ROUTINES

JDATA mode allows streaming transfer 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. Only 16-bit host mode is available with JDATA, because the 8-bit JDATA code stream uses the HDATA[31:24] pins. JDATA compressed data transfers are synchronized to the MCLK. (See the ADV212 data sheet for JDATA mode timing specifications.) JDATA requires that the frequency of JCLK be at least four times the frequency of MCLK. This means that for 150 MHz parts, the maximum MCLK can be 37.25 MHz.

Set BUSMODE[1:0], the DWIDTH bit field, to 8-bit; set BUSMODE[3:2], the HWIDTH bit field, to 16-bit. Set BUSMODE[6:4], the BCFG (bus configuration) bit field to JDATA, which assigns the HDATA [31:24] pins to JDATA[7:0].

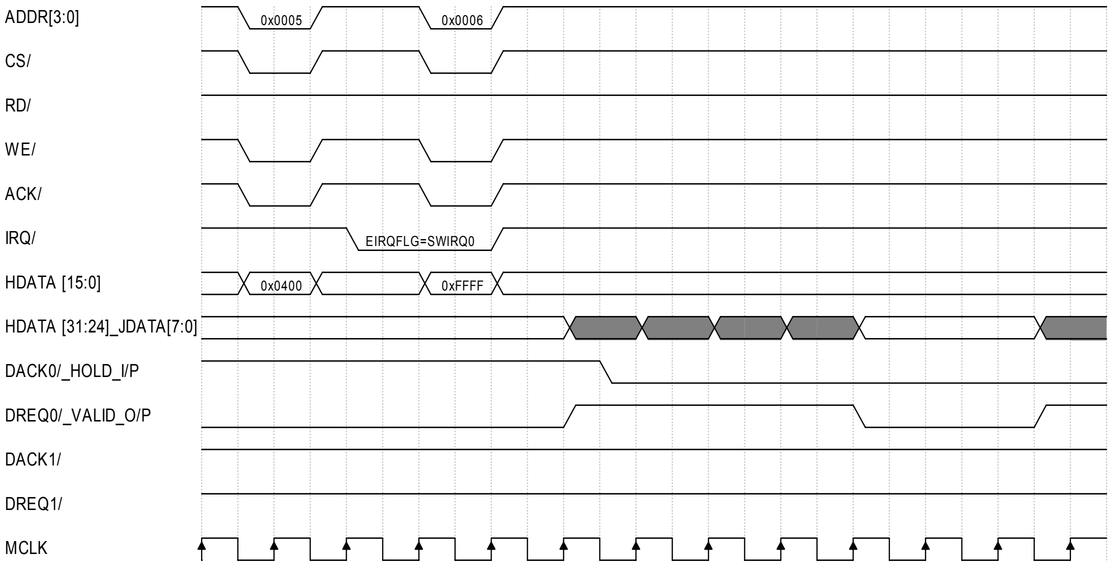
When JDATA mode is used, the CODE FIFO is dedicated to DMA Channel 0. The other data FIFOs (PIXEL or ATTR) are not available when JDATA mode is used. Attempting to access these FIFOs interferes with JDATA operation. However, direct and indirect registers can be accessed during JDATA transfers.

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 at the rising edge of MCLK when the VALID signal is asserted and the HOLD signal is deasserted.

Set all of the EDMOD0 mode bits (DR0POL, DA0POL, DMSEL0, and so on) before you enable the DMA channel with EDMOD0[0], the DMEN0 bit field.

Use the DR0POL bit field (EDMOD0[9]) to set the polarity of the VALID ($\overline{\text{DREQ0}}$ pin) and the DA0POL bit field (EDMOD0[10]) to set the polarity of the HOLD ($\overline{\text{DACK0}}$ pin). Set the DMA channel 0 to use the CODE FIFO by setting the DMSEL0 bit field (EDMOD0[2:1] to ‘01’ and set DMA channel 0 to use JDATA mode by setting the DMMOD0 bit field (EDMOD0[5:3]) to ‘011’.

Figure 22 shows example pin settings for JDATA mode encode and decode timing when VALID and HOLD are set to active high. The difference on the JDATA bus is that in encode, the ADV212 drives the data; in decode, the external device drives the data.



Setup and hold times cannot be inferred from and are not implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 22. JDATA Mode Encode/Decode Relative Timing Example

Figure 23 shows an example of a JDATA mode encode flow. JDATA mode encode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 16-bit host interface and an 8-bit data bus. The MMODE register is set to enable 16-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configure the BUSMODE and MMODE registers and any application-specific registers. You set the encode parameters with the IADDR and IDATA registers. IADDR must be set to the memory location 0x00057F00 and IDATA must be written with the encode parameters. For more information on setting the encode parameters, see Table 8 on page 43. Next, you set the EDMODx register. The EDMODx register configures the ADV212 to use external DMA channels to transfer compressed data. Then, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions. Encoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

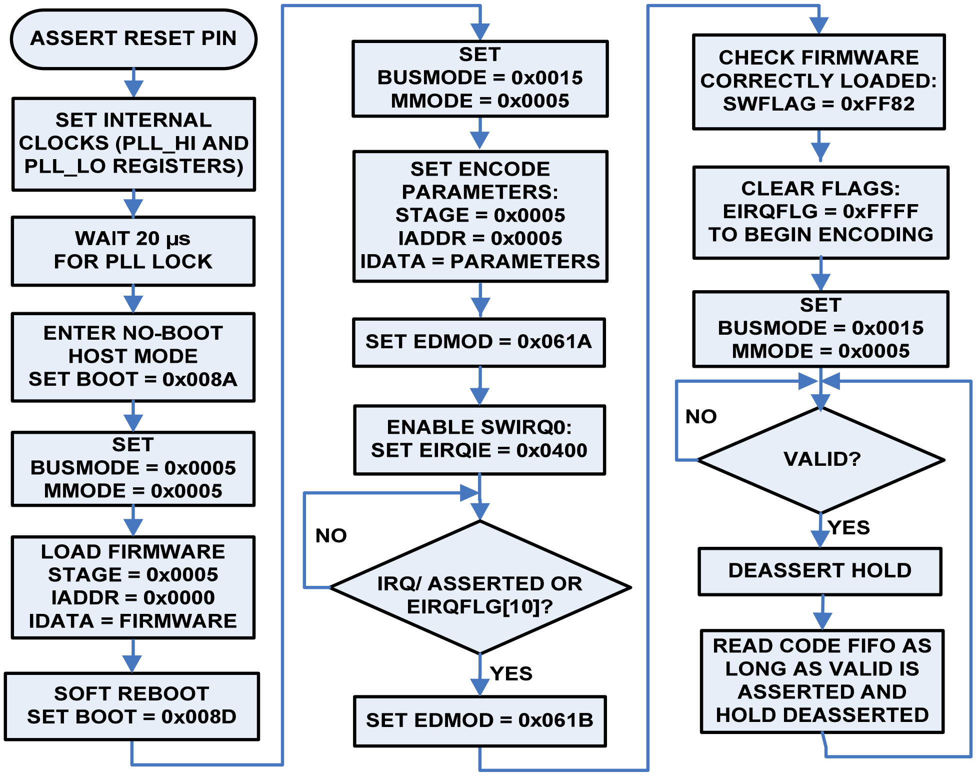

ASSERT RESET PIN
SET INTERNAL CLOCKS (PLL_HI AND PLL_LO REGISTERS)
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE SET BOOT = 0x008A
SET BUSMODE = 0x0005 MMODE = 0x0005
LOAD FIRMWARE STAGE = 0x0005 IADDR = 0x0000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
SET BUSMODE = 0x0015 MMODE = 0x0005
SET ENCODE PARAMETERS: STAGE = 0x0005 IADDR = 0x0005 IDATA = PARAMETERS
SET EDMOD = 0x061A
ENABLE SWIRQ0: SET EIRQIE = 0x0400
NO
IRQ/ ASSERTED OR EIRQFLG[10]?
YES
SET EDMOD = 0x061B
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFF82
CLEAR FLAGS: EIRQFLG = 0xFFFF TO BEGIN ENCODING
SET BUSMODE = 0x0015 MMODE = 0x0005
NO
VALID?
YES
DEASSERT HOLD
READ CODE FIFO AS LONG AS VALID IS ASSERTED AND HOLD DEASSERTED

Figure 23. Example of JDATA Mode Encode Routine

Figure 24 shows an example of a JDATA mode decode flow. JDATA mode decode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 16-bit host interface and an 8-bit data bus. The MMODE register is set to enable 16-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configure the BUSMODE and MMODE registers and any application-specific registers. You set the decode parameters with the IADDR and IDATA registers. IADDR must be set to the memory location 0x00057F00 and IDATA must be written with the decode parameters. For more information on setting the decode parameters, see Table 9 on page 47. Next, you set the EDMODx register. The EDMODx register configures the ADV212 to use external DMA channels to transfer compressed data. Then, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions. Decoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

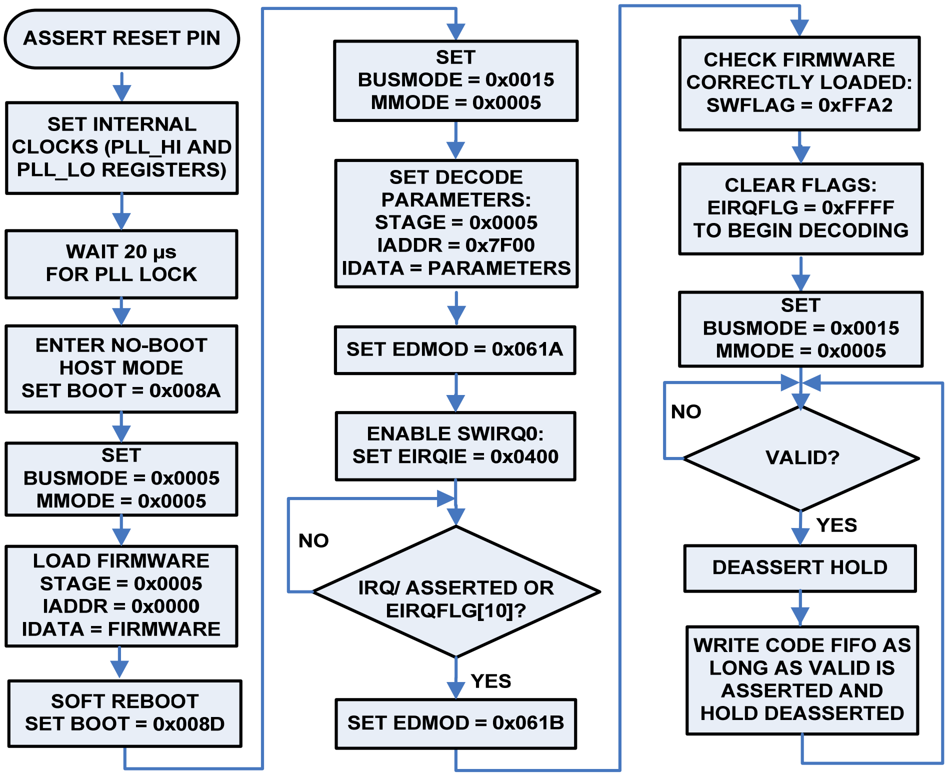

ASSERT RESET PIN
SET INTERNAL CLOCKS (PLL_HI AND PLL_LO REGISTERS)
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE SET BOOT = 0x008A
SET BUSMODE = 0x0005 MMODE = 0x0005
LOAD FIRMWARE STAGE = 0x0005 IADDR = 0x0000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
SET BUSMODE = 0x0015 MMODE = 0x0005
SET DECODE PARAMETERS: STAGE = 0x0005 IADDR = 0x7F00 IDATA = PARAMETERS
SET EDMOD = 0x061A
ENABLE SWIRQ0: SET EIRQIE = 0x0400
NO
IRQ/ ASSERTED OR EIRQFLG[10]?
YES
SET EDMOD = 0x061B
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFFA2
CLEAR FLAGS: EIRQFLG = 0xFFFF TO BEGIN DECODING
SET BUSMODE = 0x0015 MMODE = 0x0005
NO
VALID?
YES
DEASSERT HOLD
WRITE CODE FIFO AS LONG AS VALID IS ASSERTED AND HOLD DEASSERTED

Figure 24. Example of JDATA Mode Decode Routine

CUSTOM-SPECIFIC MODE ENCODE AND DECODE ROUTINES

Custom-specific mode is used for any input format that does not comply to the standards listed for VFORMAT in Table 8 on page 43 and Table 9 on page 47.

Custom-specific mode has the following limitations:

- Maximum image input size: 1.048 Msamples/image. For YCbCr 4:2:2 data, there are 2 samples per pixel.
- Maximum number of code blocks per image: 610. The number of code blocks per image can be controlled with the CBSIZE ADV212 firmware parameter (see Table 8). Also see the technical note "*How to estimate the number of codeblocks/image*".
- Data input rate: see the ADV212 data sheet.
- Minimum of six lines of vertical blanking
- Minimum of 16 pixels of horizontal blanking
- H and V should not be asserted simultaneously.

In custom-specific mode, the pixel interface register settings override the firmware parameter settings for PREC, UNI, and PICFG. See the *ADV212 User's Guide* for more information about the PMODE1 register, the PMODE2 register, and the VMODE register.

Table 2. CUSTOM-SPECIFIC REGISTER SETTINGS

Firmware Parameter Setting	Register Setting
PREC	PMODE1[10:8] PREC. Precision
UNI	PMODE2[4] YUNI. Y unipolar PMODE2[5] CUNI. C unipolar
PICFG	PMODE2[0] VCLK_POL. VCLK active edge PMODE2[1] VSYNC_POL. VSYNC active edge PMODE2[2] HSYNC_POL. HSYNC active edge PMODE2[3] FIELD_POL. FIELD active edge
—	PMODE1[4:0] PFMT. Pixel format

The VMODE register must be fully specified.

Table 3 lists the dimension registers for nonstandard video input that are configured in custom-specific mode. For a detailed bit description of each register, see the *ADV212 User's Guide.*

Table 3. ADV212 DIMENSION REGISTERS

Indirect Register Address	Name	Description
0xFFFF0400	PMODE1	Pixel/video format
0xFFFF040C	XTOT	Total samples per line
0xFFFF0410	YTOT	Total lines per frame/field
0xFFFF0414	F0_START	Start line of Field 0
0xFFFF0418	F1_START	Start line of Field 1
0xFFFF041C	V0_START	Start of active video Field 0
0xFFFF0420	V1_START	Start of active video Field 1
0xFFFF0424	V0_END	End of active video Field 0
0xFFFF0428	V1_END	End of active video Field 1
0xFFFF042C	PIXEL_START	Horizontal start of active video
0xFFFF0430	PIXEL_END	Horizontal end of active video
0xFFFF0448	PMODE2	Pixel Mode 2 Register
0xFFFF044C	VMODE	Video mode

In custom-specific mode, the video is expected to be input on VDATA [11:0] with HVF syncs or with EAV/SAV codes. Polarities for H, V, and F are programmed in the PMODE2 register. If HVF syncs are used, only the first active HSYNC transition at the start of active video is recognized, and that transition starts the horizontal counters. There is no HSYNC pulse-width requirement; HSYNC must be asserted at the start of each active line.

For vertical synchronization, the ADV212 uses the inactive edge of VSYNC. In progressive modes, the VSYNC inactive edge occurs in Field 0. In interlaced modes, a VSYNC inactive edge occurs in both Field 0 and Field 1, but the ADV212 ignores the Field 0 VSYNC inactive edge and re-synchronizes to the Field 1 VSYNC inactive edge on every frame.

When you use raw video mode, the pixel data is expected to be input left-justified on VDATA [15:0] with VFRM, VRDY, and VSTRB syncs.

For noninterlaced formats or frame-based input formats such as still images, you do not need to set the F1_START, V1_START or V1_END registers.

F0_START and F1_START are used in decode mode to identify the line number at which the field bit transitions from 0 to 1. In decode master mode, the field register settings determine the field output transition. In decode slave mode, the field register settings and the input field bit determine the location of the pixel interface field transition.

Encode Mode

In encode mode, the part is always in a slave configuration. Input data must be accompanied by separate H, V, and F signals or embedded EAV/SAV timing codes. In both cases, the dimension registers must reflect the video standard of the input.

The active video region to be processed is calculated using the values of the Vx_START, Vx_END, PIXEL_START, PIXEL_END, XTOT, and YTOT registers. However, this only applies to input video modes using the VDATA bus. For HIPI or raw video modes, only the values for XTOT and YTOT need to be programmed; all other dimension register values are ignored. (For more information on HIPI mode, see “*ADV202 HIPI mode*” and “*Using the ADV202 in HIPI mode*”.)

Decode Mode

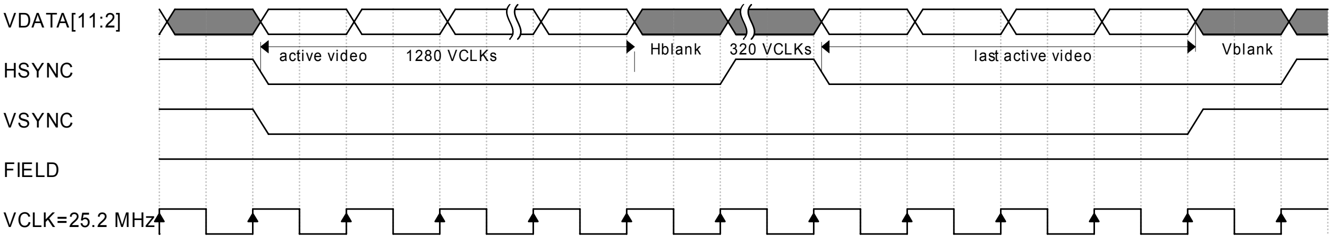
In decode mode, the part can be in either slave mode or master mode. In decode slave mode, video output is synchronized to the incoming HVF signals. The dimension register settings must match the input format as determined by the HVF inputs. In decode master mode, HSYNC, VSYNC, and FIELD or EAV/SAV codes are generated according to the settings of these registers:

- XTOT
- YTOT
- F0_START
- F1_START
- F0_END
- F1_END
- V0_START
- V1_START
- V0_END
- V1_END
- PI XEL_START
- PIXEL_END

The VMODE register must be programmed to decode master mode to enable the generation of these timing signals.

Custom-Specific Mode Configuration Example

In this example, custom-specific mode is configured for VGA, progressive video, 10-bit YCbCr data with 800 × 525 @ 60 Hz total and 640 × 480 @ 60 Hz active resolution. The input VCLK frequency is 25.2 MHz. The PMODE2 register is set to VSYNC polarity '0' , HSYNC polarity '0' and VCLK polarity '1'.



Setup and hold times cannot be inferred from and are not implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 25. Custom-specific Mode Encode and Decode Relative Timing Example

Table 4. CUSTOM-SPECIFIC EXAMPLE—DIMENSION REGISTER SETTINGS FOR HVF PROGRESSIVE VIDEO MODE

Indirect Register Address	Name	Register Setting for Encode	Register Settings for Decode
0xFFFF0400	PMODE1	0x0005	0x0005
0xFFFF040C	XTOT	0x0640 [1600]	0x0640
0xFFFF0410	YTOT	0x020D [525]	0x020D
0xFFFF0414	F0_START	0x0001 [1]	0x0001
0xFFFF0418	F1_START	0x0000 [0]	0x0000
0xFFFF041C	V0_START	0x002E [46]	0x002E
0xFFFF0420	V1_START	0x0000 [0]	0x0000
0xFFFF0424	V0_END	0x020D [525]	0x020D
0xFFFF0428	V1_END	0x0000 [0]	0x0000
0xFFFF042C	PIXEL_START	0x0001 [1]	0x0001
0xFFFF0430	PIXEL_END	0x0500 [1280]	0x0500
0xFFFF0448	PMODE2	0x0031 for polarity as in Figure 25.	0x0031
0xFFFF044C	VMODE	0x0086 HVF mode	0x0085 (Decode Master)

Figure 26 shows an example of a custom-specific mode encode flow using DCS mode for compressed data output. (DCS mode is an example; you can use any mode.) Custom-specific mode encode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 32-bit host interface and a 32-bit data bus. The MMODE register is set to enable 32-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configure the BUSMODE and MMODE registers and any application-specific registers. You set the encode parameters and the EDMODx register. The EDMODx register configures the ADV212 to use external DMA channels to transfer compressed data. Next, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions.

The CODE FIFO threshold register, FFTHRC, is also set in DCS mode. Encoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

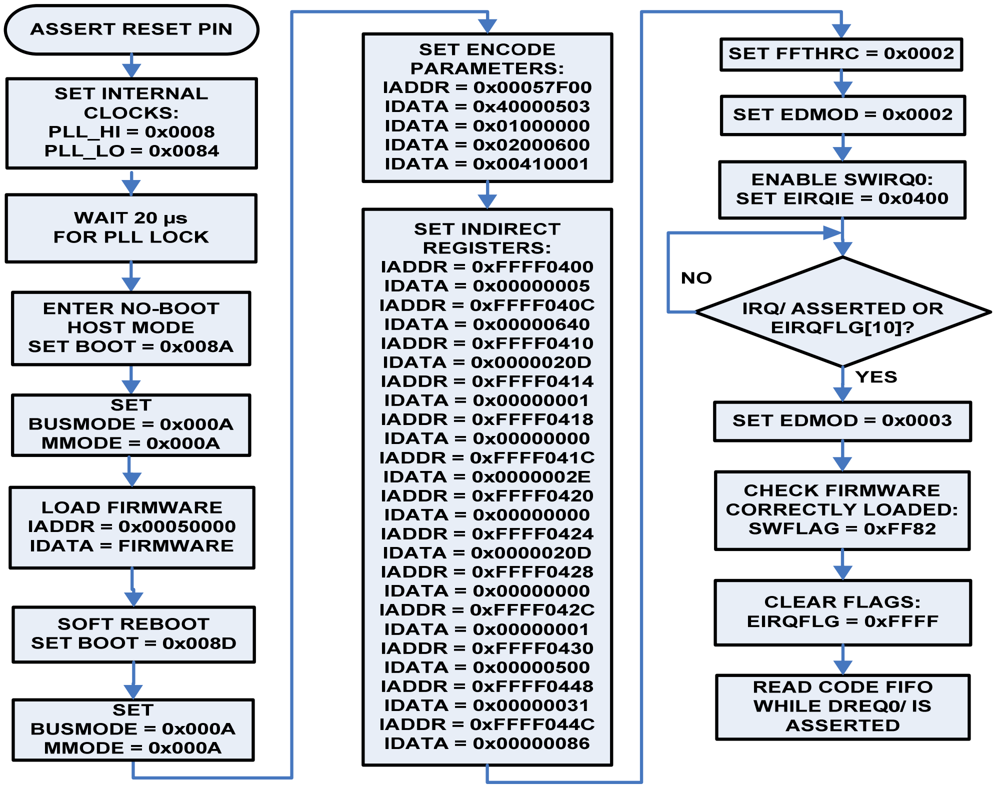

ASSERT RESET PIN
SET INTERNAL CLOCKS: PLL_HI = 0x0008 PLL_LO = 0x0084
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE SET BOOT = 0x008A
SET BUSMODE = 0x000A MMODE = 0x000A
LOAD FIRMWARE IADDR = 0x00050000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
SET BUSMODE = 0x000A MMODE = 0x000A
SET ENCODE PARAMETERS: IADDR = 0x00057F00 IDATA = 0x40000503 IDATA = 0x01000000 IDATA = 0x02000600 IDATA = 0x00410001
SET INDIRECT REGISTERS: IADDR = 0xFFFF0400 IDATA = 0x00000005 IADDR = 0xFFFF040C IDATA = 0x00000640 IADDR = 0xFFFF0410 IDATA = 0x0000020D IADDR = 0xFFFF0414 IDATA = 0x00000001 IADDR = 0xFFFF0418 IDATA = 0x00000000 IADDR = 0xFFFF041C IDATA = 0x0000002E IADDR = 0xFFFF0420 IDATA = 0x00000000 IADDR = 0xFFFF0424 IDATA = 0x0000020D IADDR = 0xFFFF0428 IDATA = 0x00000000 IADDR = 0xFFFF042C IDATA = 0x00000001 IADDR = 0xFFFF0430 IDATA = 0x00000500 IADDR = 0xFFFF0448 IDATA = 0x00000031 IADDR = 0xFFFF044C IDATA = 0x00000086
SET FFTHRC = 0x0002
SET EDMOD = 0x0002
ENABLE SWIRQ0: SET EIRQIE = 0x0400
NO
IRQ/ ASSERTED OR EIRQFLG[10]?
YES
SET EDMOD = 0x0003
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFF82
CLEAR FLAGS: EIRQFLG = 0xFFFF
READ CODE FIFO WHILE DREQ0/ IS ASSERTED

Figure 26. Example of Custom-specific Mode Encode Routine

Figure 27 shows an example of a custom-specific mode decode flow using DCS mode for compressed data input. (DCS mode is an example; you can use any mode.) Custom-specific mode decode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 32-bit host interface and a 32-bit data bus. The MMODE register is set to enable 32-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configure the BUSMODE and MMODE registers and any application-specific registers. You set the decode parameters and the EDMODx register. The EDMODx register configures the ADV212 to use external DMA channels to transfer compressed data. Next, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions.

The CODE FIFO threshold register, FFTHRC, is also set in DCS mode. Decoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

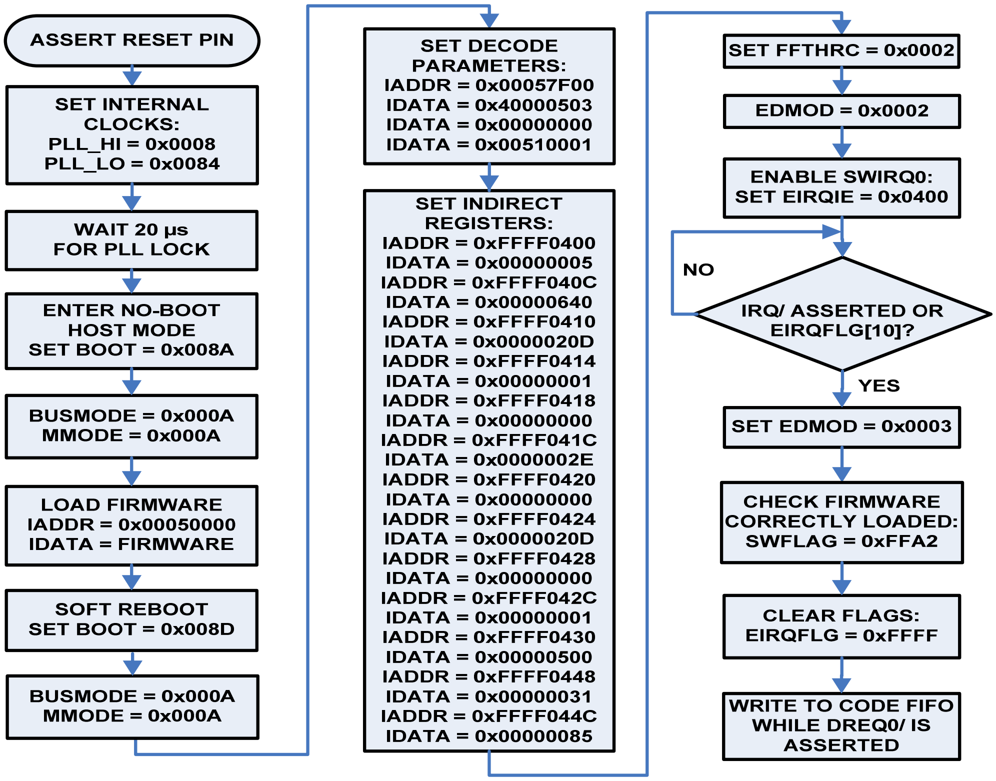

ASSERT RESET PIN
SET INTERNAL CLOCKS: PLL_HI = 0x0008 PLL_LO = 0x0084
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE SET BOOT = 0x008A
BUSMODE = 0x000A MMODE = 0x000A
LOAD FIRMWARE IADDR = 0x00050000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
BUSMODE = 0x000A MMODE = 0x000A
SET DECODE PARAMETERS: IADDR = 0x00057F00 IDATA = 0x40000503 IDATA = 0x00000000 IDATA = 0x00510001
SET INDIRECT REGISTERS: IADDR = 0xFFFF0400 IDATA = 0x00000005 IADDR = 0xFFFF040C IDATA = 0x00000640 IADDR = 0xFFFF0410 IDATA = 0x0000020D IADDR = 0xFFFF0414 IDATA = 0x00000001 IADDR = 0xFFFF0418 IDATA = 0x00000000 IADDR = 0xFFFF041C IDATA = 0x0000002E IADDR = 0xFFFF0420 IDATA = 0x00000000 IADDR = 0xFFFF0424 IDATA = 0x0000020D IADDR = 0xFFFF0428 IDATA = 0x00000000 IADDR = 0xFFFF042C IDATA = 0x00000001 IADDR = 0xFFFF0430 IDATA = 0x00000500 IADDR = 0xFFFF0448 IDATA = 0x00000031 IADDR = 0xFFFF044C IDATA = 0x00000085
SET FFTHRC = 0x0002
EDMOD = 0x0002
ENABLE SWIRQ0: SET EIRQIE = 0x0400
NO
IRQ/ ASSERTED OR EIRQFLG[10]?
YES
SET EDMOD = 0x0003
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFFA2
CLEAR FLAGS: EIRQFLG = 0xFFFF
WRITE TO CODE FIFO WHILE DREQ0/ IS ASSERTED

Figure 27. Example of Custom-specific Mode Decode Routine

RAW PIXEL MODE ENCODE AND DECODE ROUTINES

In raw pixel mode, pixel data is transferred on the VDATA bus. The transfer data is expected to have no timing information or blanking region associated with it. The region to be captured and the frame transitions are determined by VRDY (HSYNC), VFRM (VSYNC), VSTRB (FIELD), and the dimension registers XTOT and YTOT. Pixel data should be transferred on VDATA[15:0] with one pixel data sample per VCLK cycle.

Raw pixel mode must be enabled in the VMODE register and the ADV212 parameter VFORMAT must be set to custom-specific. Raw pixel mode only supports single-component transfers or 3-component 4:2:2 in YCbYCr format. 32-bit host mode is not available with raw pixel mode.

Raw Pixel Mode Configuration Example

In these examples, raw pixel mode is configured for VGA, single-component, 8-bit Y data with 640 × 480 @ 60 Hz active resolution.

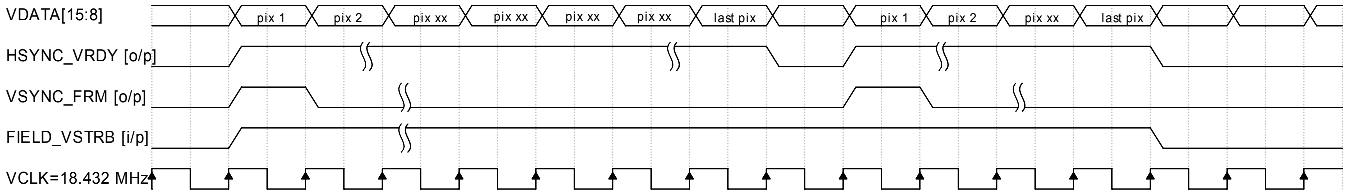
Raw Pixel Mode Encode

Figure 28 shows HSYNC_VRDY, VSYNC_VFRM, and FIELD_VSTRB programmed for active high polarity.

HSYNC_VRDY (output): When this signal is asserted high, the ADV212 is ready to accept pixel data.

VSYNC_VFRM (input): This signal must be asserted high for one VCLK cycle, at the start of a frame, on the first pixel sample.

FIELD_VSTRB (input): When this signal is asserted high, there is valid pixel data on the VDATA bus for the ADV212 to capture.



Setup and hold times cannot be inferred from or implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 28. Raw Pixel Mode Encode Relative Timing Example

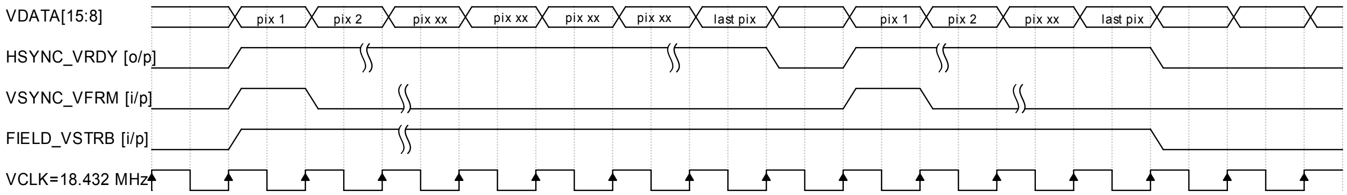
Raw Pixel Mode Decode

Figure 29 shows HSYNC_VRDY, VSYNC_VFRM, and FIELD_VSTRB programmed for active high polarity.

HSYNC_VRDY (input): When this signal is asserted high, the ADV212 has valid pixel data to output.

VSYNC_VFRM (output): The ADV212 asserts this signal for one VCLK cycle, at the start of a frame, on the first pixel sample.

FIELD_VSTRB (input): When this signal is asserted high, the host is ready to accept pixel data from the ADV212.



Setup and hold times cannot be inferred from and are not implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 29. Raw Pixel Mode Decode Relative Timing Example

Table 5. DIMENSION REGISTER SETTINGS FOR RAW PIXEL MODE EXAMPLE

Indirect Register Address	Name	Register Setting for Encode	Register Settings for Decode
0xFFFF0400	PMODE1	0x0004	0x0004
0xFFFF040C	XTOT	0x0280 [640]	0x0280 [640]
0xFFFF0410	YTOT	0x01E0 [480]	0x01E0 [480]
0xFFFF0414	F0_START	0x0000	0x0000
0xFFFF0418	F1_START	0x0000	0x0000
0xFFFF041C	V0_START	0x0001	0x0001
0xFFFF0420	V1_START	0x0000	0x0000
0xFFFF0424	V0_END	0x01E0	0x01E0
0xFFFF0428	V1_END	0x0000	0x0000
0xFFFF042C	PIXEL_START	0x0001	0x0001
0xFFFF0430	PIXEL_END	0x0280	0x0280
0xFFFF0448	PMODE2	0x003F for polarity as in Figure 28.	0x003F for polarity as in Figure 29.
0xFFFF044C	VMODE	0x0022	0x0021

Figure 30 shows an example of a raw pixel mode encode flow using DCS mode for compressed data output. (DCS mode is an example; you can use any mode.) Raw pixel mode encode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 16-bit host interface and a 16-bit data bus. The MMODE register is set to enable 16-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configure the BUSMODE and MMODE registers and any application-specific registers. You set the encode parameters and the EDMODx register. The EDMODx register configures the ADV212 to use external DMA channels to transfer compressed data. Next, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions.

The CODE FIFO threshold register, FFTHRC, is also set in DCS mode. Encoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

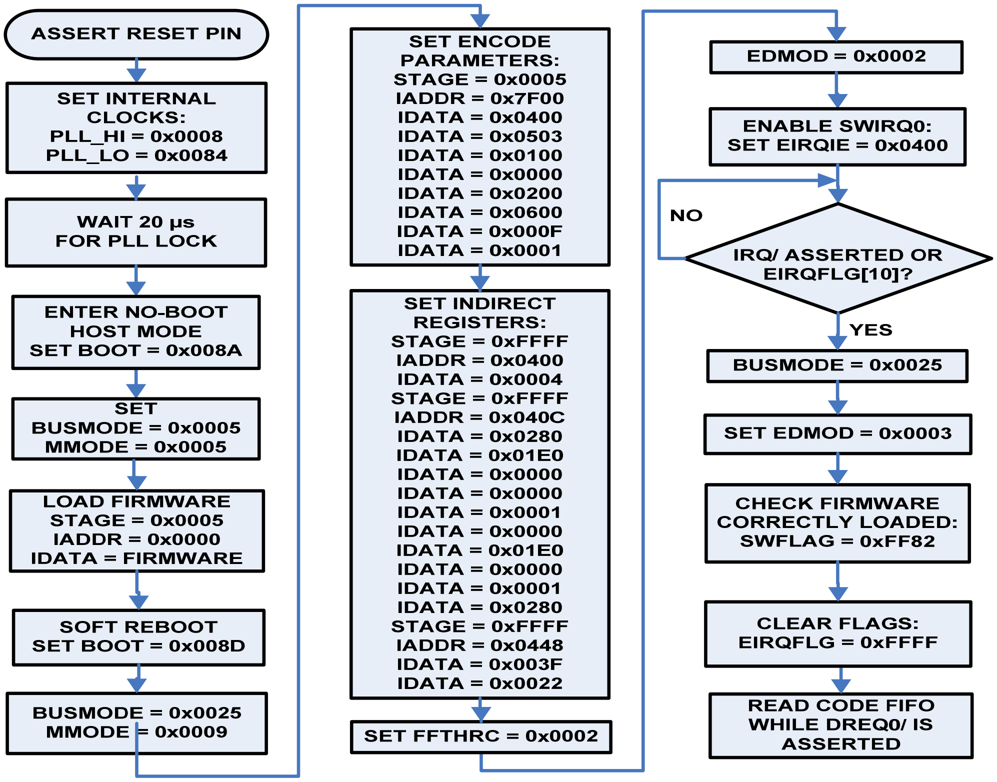

ASSERT RESET PIN
SET INTERNAL CLOCKS: PLL_HI = 0x0008 PLL_LO = 0x0084
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE SET BOOT = 0x008A
SET BUSMODE = 0x0005 MMODE = 0x0005
LOAD FIRMWARE STAGE = 0x0005 IADDR = 0x0000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
BUSMODE = 0x0025 MMODE = 0x0009
SET ENCODE PARAMETERS: STAGE = 0x0005 IADDR = 0x7F00 IDATA = 0x0400 IDATA = 0x0503 IDATA = 0x0100 IDATA = 0x0000 IDATA = 0x0200 IDATA = 0x0600 IDATA = 0x000F IDATA = 0x0001
SET INDIRECT REGISTERS: STAGE = 0xFFFF IADDR = 0x0400 IDATA = 0x0004 STAGE = 0xFFFF IADDR = 0x040C IDATA = 0x0280 IDATA = 0x01E0 IDATA = 0x0000 IDATA = 0x0000 IDATA = 0x0001 IDATA = 0x0000 IDATA = 0x01E0 IDATA = 0x0000 IDATA = 0x0001 IDATA = 0x0280 STAGE = 0xFFFF IADDR = 0x0448 IDATA = 0x003F IDATA = 0x0022
SET FFTHRC = 0x0002
EDMOD = 0x0002
ENABLE SWIRQ0: SET EIRQIE = 0x0400
NO
IRQ/ ASSERTED OR EIRQFLG[10]?
YES
BUSMODE = 0x0025
SET EDMOD = 0x0003
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFF82
CLEAR FLAGS: EIRQFLG = 0xFFFF
READ CODE FIFO WHILE DREQ0/ IS ASSERTED

Figure 30. Example of Raw Pixel Mode Encode Routine (DCS)

Figure 31 shows an example of a raw pixel mode decode flow using DCS mode for compressed data input. (DCS mode is an example; you can use any mode.) Raw pixel mode decode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 16-bit host interface and a 16-bit data bus. The MMODE register is set to enable 16-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configure the BUSMODE and MMODE registers and any application-specific registers. You set the decode parameters and the EDMODx register. The EDMODx register configures the ADV212 to use external DMA channels to transfer compressed data. Next, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions.

The CODE FIFO threshold register, FFTHRC, is also set in DCS mode. Decoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

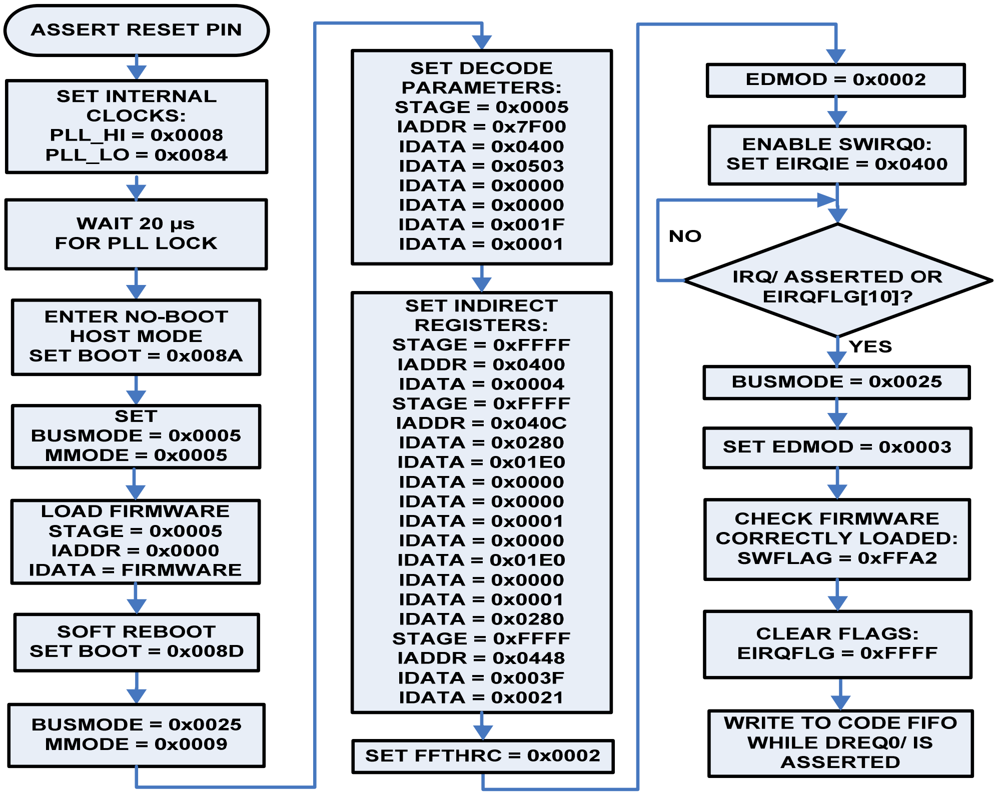
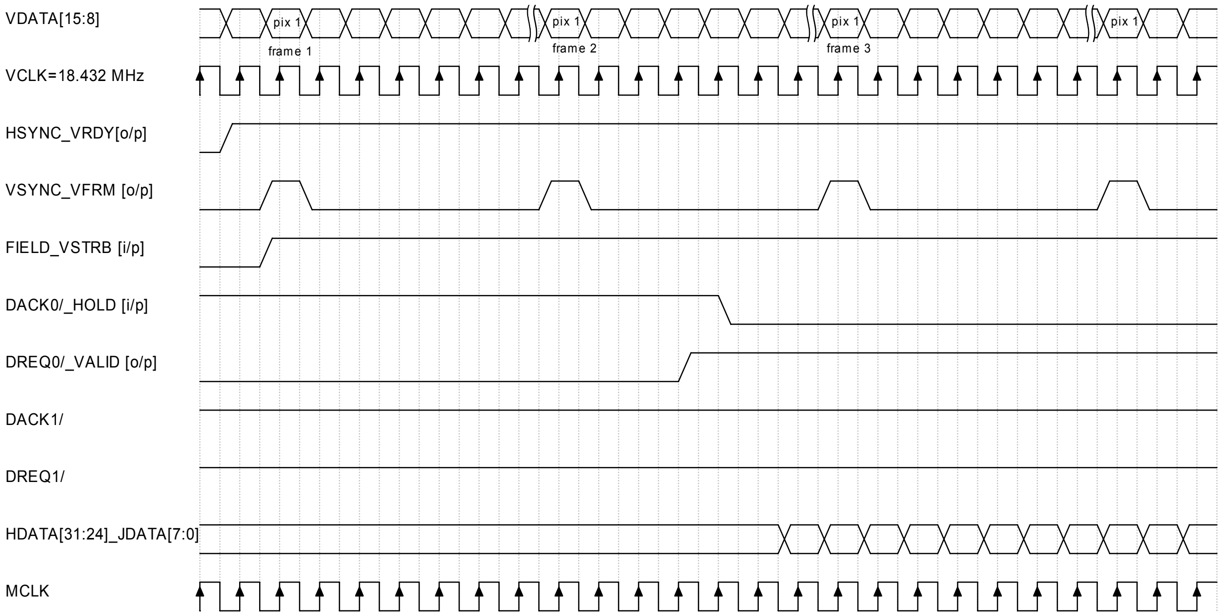

ASSERT RESET PIN
SET INTERNAL CLOCKS: PLL_HI = 0x0008 PLL_LO = 0x0084
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE SET BOOT = 0x008A
SET BUSMODE = 0x0005 MMODE = 0x0005
LOAD FIRMWARE STAGE = 0x0005 IADDR = 0x0000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
BUSMODE = 0x0025 MMODE = 0x0009
SET DECODE PARAMETERS: STAGE = 0x0005 IADDR = 0x7F00 IDATA = 0x0400 IDATA = 0x0503 IDATA = 0x0000 IDATA = 0x0000 IDATA = 0x001F IDATA = 0x0001
SET INDIRECT REGISTERS: STAGE = 0xFFFF IADDR = 0x0400 IDATA = 0x0004 STAGE = 0xFFFF IADDR = 0x040C IDATA = 0x0280 IDATA = 0x01E0 IDATA = 0x0000 IDATA = 0x0000 IDATA = 0x0001 IDATA = 0x0000 IDATA = 0x01E0 IDATA = 0x0000 IDATA = 0x0001 IDATA = 0x0280 STAGE = 0xFFFF IADDR = 0x0448 IDATA = 0x003F IDATA = 0x0021
SET FFTHRC = 0x0002
EDMOD = 0x0002
ENABLE SWIRQ0: SET EIRQIE = 0x0400
NO
IRQ/ ASSERTED OR EIRQFLG[10]?
YES
BUSMODE = 0x0025
SET EDMOD = 0x0003
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFFA2
CLEAR FLAGS: EIRQFLG = 0xFFFF
WRITE TO CODE FIFO WHILE DREQ0/ IS ASSERTED

Figure 31. Example of Raw Pixel Mode Decode Routine (DCS)

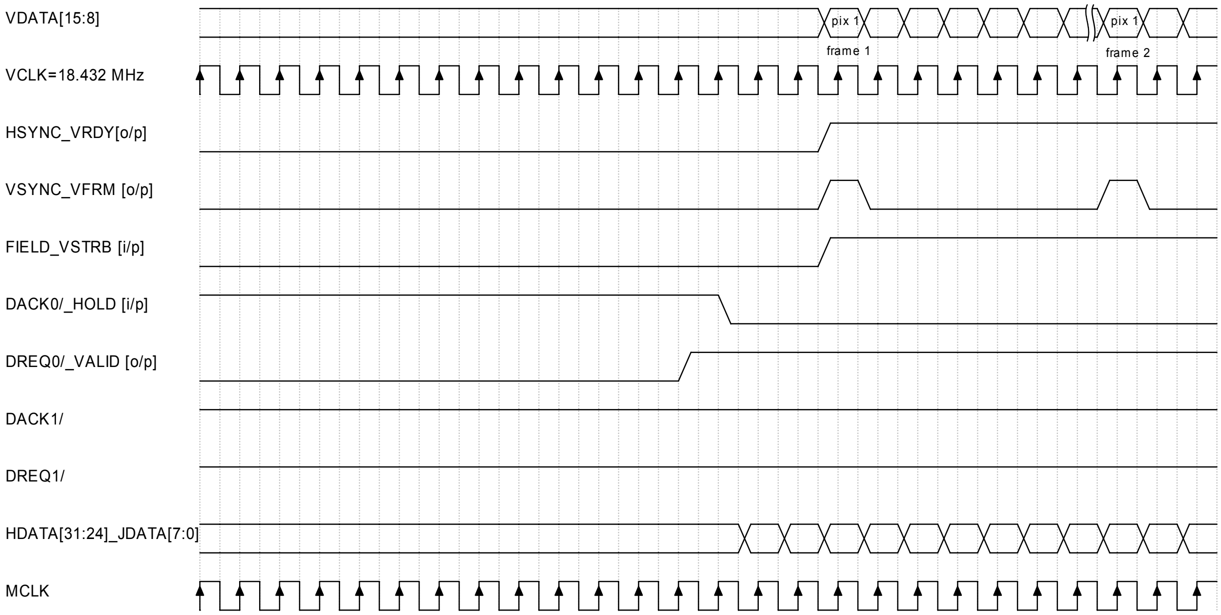
Raw Pixel Mode – JDATA Mode

The synchronous 8-bit JDATA interface can be used to transfer compressed data. In Figure 32, encode, the VSYNC_VFRM signal is input. In Figure 33, decode, the VSYNC_VFRM signal is output.



Setup and hold times cannot be inferred from and are not implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 32. Raw Pixel Mode Encode Relative Timing Example (JDATA)



Setup and hold times cannot be inferred from and are not implied by this diagram. The ADV212 datasheet provides precise timing information.

Figure 33. Raw Pixel Mode Decode Relative Timing Example (JDATA)

Figure 34 shows an example of a raw pixel mode encode flow using JDATA mode for compressed data output. (JDATA mode is an example; you can use any mode.) Raw pixel mode encode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 16-bit host interface and an 8-bit data bus. The MMODE register is set to enable 16-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configure the BUSMODE and MMODE registers and any application-specific registers. You set the encode parameters and the EDMODx register. The EDMODx register configures the ADV212 to use external DMA channels to transfer compressed data. Next, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See “Confirming Correct Firmware Load” on page 9 for detailed instructions. Encoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

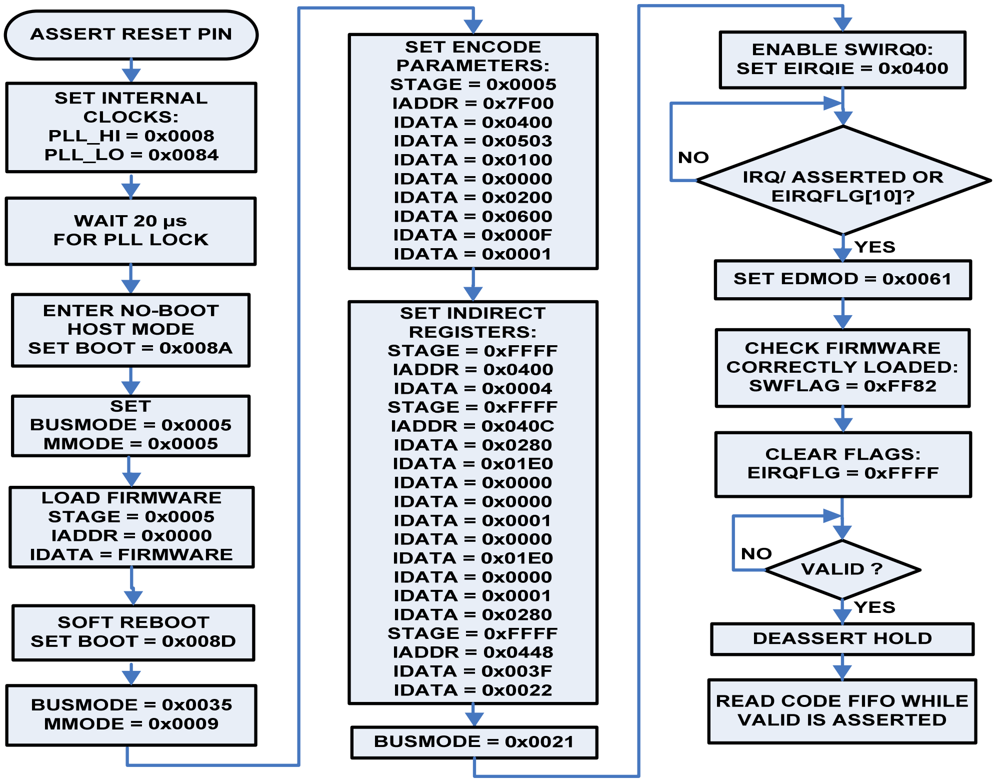

ASSERT RESET PIN
SET INTERNAL CLOCKS: PLL_HI = 0x0008 PLL_LO = 0x0084
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE SET BOOT = 0x008A
SET BUSMODE = 0x0005 MMODE = 0x0005
LOAD FIRMWARE STAGE = 0x0005 IADDR = 0x0000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
BUSMODE = 0x0035 MMODE = 0x0009
SET ENCODE PARAMETERS: STAGE = 0x0005 IADDR = 0x7F00 IDATA = 0x0400 IDATA = 0x0503 IDATA = 0x0100 IDATA = 0x0000 IDATA = 0x0200 IDATA = 0x0600 IDATA = 0x000F IDATA = 0x0001
SET INDIRECT REGISTERS: STAGE = 0xFFFF IADDR = 0x0400 IDATA = 0x0004 STAGE = 0xFFFF IADDR = 0x040C IDATA = 0x0280 IDATA = 0x01E0 IDATA = 0x0000 IDATA = 0x0000 IDATA = 0x0001 IDATA = 0x0000 IDATA = 0x01E0 IDATA = 0x0000 IDATA = 0x0001 IDATA = 0x0280 STAGE = 0xFFFF IADDR = 0x0448 IDATA = 0x003F IDATA = 0x0022
BUSMODE = 0x0021
ENABLE SWIRQ0: SET EIRQIE = 0x0400
NO
IRQ/ ASSERTED OR EIRQFLG[10]?
YES
SET EDMOD = 0x0061
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFF82
CLEAR FLAGS: EIRQFLG = 0xFFFF
NO
VALID ?
YES
DEASSERT HOLD
READ CODE FIFO WHILE VALID IS ASSERTED

Figure 34. Example of Raw Pixel Mode Encode Routine (JDATA)

Figure 35 shows an example of a raw pixel mode decode flow using JDATA mode for compressed data input. (JDATA mode is an example; you can use any mode.) Raw pixel mode decode initialization starts with direct register accesses of the PLL registers, the BOOT register, the MMODE register, and the BUSMODE register. In this example, the BUSMODE register is set to enable a 16-bit host interface and an 8-bit data bus. The MMODE register is set to enable 16-bit indirect register access capability. After loading these registers, the firmware can be loaded.

After you load the firmware and the ADV212 firmware parameters into ADV212 memory, you perform a soft reboot by setting the BOOT register, and then configure the BUSMODE and MMODE registers and any application-specific registers. You set the decode parameters and the EDMODx register. The EDMODx register configures the ADV212 to use external DMA channels to transfer compressed data. Next, you confirm that the correct firmware is loaded by setting the EIRQIE register and reading the application ID. Correct firmware load can be confirmed by an interrupt or poll routine of the EIRQFLG register. See "Confirming Correct Firmware Load" on page 9 for detailed instructions. Decoding starts after you confirm the correct firmware load and clear the EIRQFLG register.

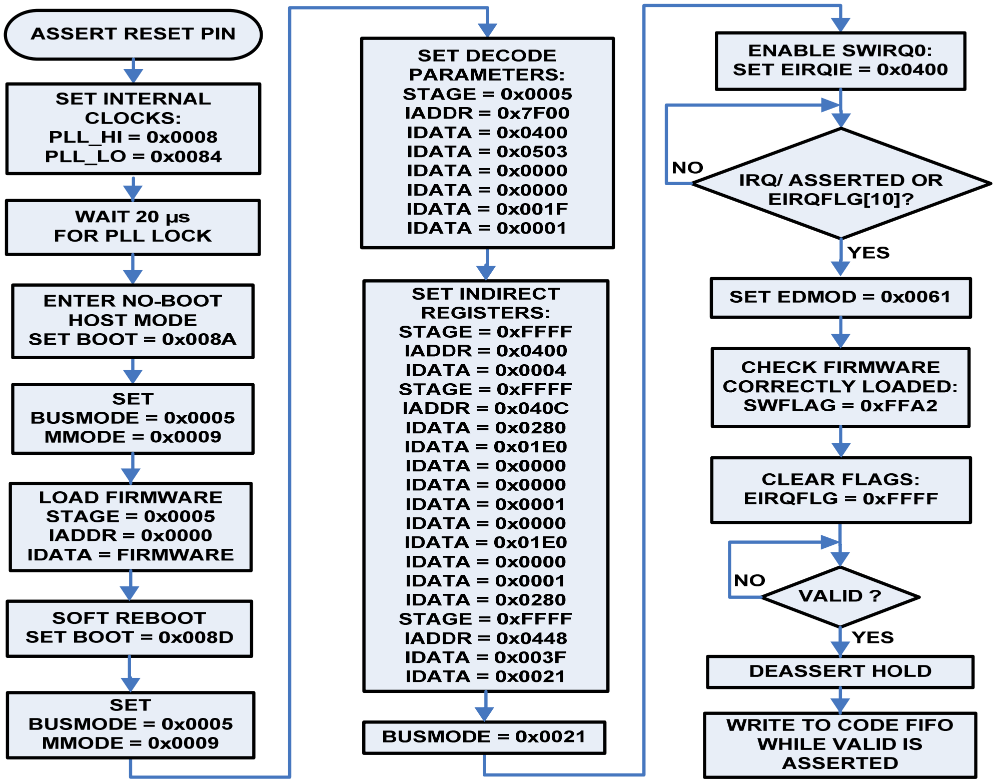

ASSERT RESET PIN
SET INTERNAL CLOCKS: PLL_HI = 0x0008 PLL_LO = 0x0084
WAIT 20 µs FOR PLL LOCK
ENTER NO-BOOT HOST MODE SET BOOT = 0x008A
SET BUSMODE = 0x0005 MMODE = 0x0009
LOAD FIRMWARE STAGE = 0x0005 IADDR = 0x0000 IDATA = FIRMWARE
SOFT REBOOT SET BOOT = 0x008D
SET BUSMODE = 0x0005 MMODE = 0x0009
SET DECODE PARAMETERS: STAGE = 0x0005 IADDR = 0x7F00 IDATA = 0x0400 IDATA = 0x0503 IDATA = 0x0000 IDATA = 0x0000 IDATA = 0x001F IDATA = 0x0001
SET INDIRECT REGISTERS: STAGE = 0xFFFF IADDR = 0x0400 IDATA = 0x0004 STAGE = 0xFFFF IADDR = 0x040C IDATA = 0x0280 IDATA = 0x01E0 IDATA = 0x0000 IDATA = 0x0000 IDATA = 0x0001 IDATA = 0x0000 IDATA = 0x01E0 IDATA = 0x0000 IDATA = 0x0001 IDATA = 0x0280 STAGE = 0xFFFF IADDR = 0x0448 IDATA = 0x003F IDATA = 0x0021
BUSMODE = 0x0021
ENABLE SWIRQ0: SET EIRQIE = 0x0400
NO
IRQ/ ASSERTED OR EIRQFLG[10]?
YES
SET EDMOD = 0x0061
CHECK FIRMWARE CORRECTLY LOADED: SWFLAG = 0xFFA2
CLEAR FLAGS: EIRQFLG = 0xFFFF
NO
VALID ?
YES
DEASSERT HOLD
WRITE TO CODE FIFO WHILE VALID IS ASSERTED

Figure 35. Example of Raw Pixel Mode Decode Routine (JDATA)

APPENDIX

PLL Settings and Internal Clocks

See the ADV212 datasheet for a detailed description of the ADV212 internal clock and PLL register settings.

The tables in this section show recommended register settings for most common MCLK frequencies used.

Table 6. RECOMMENDED PLL REGISTER SETTINGS

Video Standard	MCLK Input [MHz]	PLL_HI Address = 0xE	PLL_LO Address = 0xF
ITU.R.BT656	27	0x0008	0x0004
1080i, SMPTE274M	74.25	0x0008	0x0084
1080i, SMPTE274M	37.125	0x0008	0x0004
1080i, SMPTE274M	25	0x0008	0x0006
525p, SMPTE293M	27	0x0008	0x0004
625p, ITU.R.BT1358	27	0x0008	0x0004

BUSMODE and MMODE Register Settings

Table 7. BUSMODE AND MMODE REGISTER SETTINGS

Interface	BUSMODE Address = 0x8	MMODE Address = 0x9
32-bit	0x000A	0x000A
16-bit	0x0005	0x0005
JDATA mode	0x0015	0x0005 or 0x0009
Raw pixel mode	0x0025	0x0005 or 0x0009

ADV212 Firmware Parameter Encode Settings

Table 8. ENCODE CONFIGURATION PARAMETER SETTINGS

Big-endian Byte Offset	Number of Bytes	Name	Parameter Description
0x0	1	VFORMAT	**Video Standard** 0 = NTSC 4:2:2 1 = PAL 4:2:2 2 = 1080i/60 luminance 3 = 1080i/60 chrominance 4 = Custom-specific 5 = 720/60p luminance 6 = 720/60p chrominance 7 = Reserved 8 = NTSC de-interlaced 9 = PAL de-interlaced
0x1	1	PREC	**Precision** 0 = 8-bit 1 = 10-bit 2 = 12-bit
0x2	1	XFORMLEV	**Number of Wavelet Transform Levels** 1 = 1 level of transform 2 = 2 levels of transform 3 = 3 levels of transform 4 = 4 levels of transform 5 = 5 levels of transform 6 = 6 levels of transform (not recommended)

Big-endian Byte Offset	Number of Bytes	Name	Parameter Description
0x3	1	UNI	**Component Polarity** 0 = Bipolar C, Bipolar Y 1 = Unipolar C, Bipolar Y 2 = Bipolar C, Unipolar Y 3 = Unipolar C, Unipolar Y Should be set to '3' for most applications.
0x4	1	CBSIZE	**Code Block Dimensions** 0 = 32 × 32 (does not work for 1080i or 720p) 1 = 64 × 32 2 = 64 × 64 3 = 128 × 32 (recommended setting)
0x5	1	WKERNEL	**Wavelet Kernel** 0 = Irreversible 9 × 7 using fixed table (lossy) 1 = Reversible 5 × 3 (lossless) 2 = Irreversible 5 × 3 (lossy)
0x6	1	STALLPAR	**Skip Fields in Encode Mode** 0 = Capture all fields 1 to 63 = After capturing a field, skip this number of fields 255 = Stop capture
0x7	1	ATTRTYPE	**Attribute Data Output Format** 0 = Do not output attribute data 1 = Use ADV-JP2000 attribute data format 2 = Log2 convex hull without adjustments 3 = Log2 convex hull with quantization adjust 4 = ADV212 field header output to ATTR FIFO 5 = ADV212 field header and packet byte length output to ATTR FIFO
0x8	1	RCTYPE	**Rate Control Algorithm** 0 = None (no truncation) 1 = Target size per video field or image within 5% of RCVAL 2 = Target quality per video field or image. 3 = Multilayered target size per video field or image. Requires RCVAL and LTARGET parameters also. 4 = Multilayered target quality per video field or image. Requires RCVAL and LTARGET parameters also.
0x9	3	RCVAL	**Target Size or Quality Factor** If RCTYPE = 0, then RCVAL = n/a If RCTYPE = 1, then RCVAL = Size of field or image in bytes If RCTYPE = 2, then RCVAL = Quality factor If RCTYPE ≥ 3, then RCVAL = Number of layers
0xC	1	J2KPROG	**JPEG2000 Progression Style** 0 = LRCP 1 = RLCP 2 = RPCL 3 = PCRL 4 = CPRL

Big-endian Byte Offset	Number of Bytes	Name	Parameter Description
0xD	1	PICFG	**Pixel Interface Bit Configuration Settings** Bit 7 = Reserved (write 0) Bit 6 = HVF or EAV/SAV 1 = Use external HVF pins 0 = Use embedded SAV/EAV Bit 5 = Reserved (write 0) Bit 4 = Reserved (write 0) Bit 3 = FIELD polarity (1 = positive, 0 = negative) Bit 2 = HSYNC polarity (1 = positive, 0 = negative) Bit 1 = VSYNC polarity (1 = positive, 0 = negative) Bit 0 = VCLK polarity (1 = positive, 0 = negative)
0xE	1	QFACT	**Quantization Factor** 255 = 255/256 factor 254 = 254/256 factor … 2 = 2/256 factor 1 = 1/256 factor 0 = 256/256 factor (1×)
0xF	1	COD_STYLE	**Output Code Stream Format** COD_STYLE [7:4] applies to J2C or JP2 formats only COD_STYLE [7]: 0 = Packet headers with packet body 1 = Use PPT COD_STYLE [6]: 0 = No PLT 1 = Include PLT COD_STYLE [5]: 0 = No SOP 1 = Include SOP COD_STYLE [4] : 0 = No EPH 1 = Include EPH COD_STYLE [3]: Reserved COD_STYLE [2:0]: 0 = ADV212 raw format 1 = J2C format 2 = JP2 format YCbCr (for 4:2:2 mode only) 3 = JP2 format Greyscale

Big-endian Byte Offset	Number of Bytes	Name	Parameter Description
0x10	2 x 19	STEPSIZES [18:0]	**Quantization Step Sizes Using 5 Exponent Bits Plus 11 Mantissa Bits** For more detail see the ISO/IEC15444-1 specification, section Annex E. There is one step size assigned per subband (note there is only one set of step sizes defined which all components use). The order of subband step sizes for this parameter list is as follows for 6 transform levels: LL 1HL (for > 0 transform levels) 1LH (for > 0 transform levels) 1HH (for > 0 transform levels) 2HL (for > 1 transform levels) 2LH (for > 1 transform levels) 2HH (for > 1 transform levels) 3HL (for > 2 transform levels) 3LH (for > 2 transform levels) 3HH (for > 2 transform levels) 4HL (for > 3 transform levels) 4LH (for > 3 transform levels) 4HH (for > 3 transform levels) 5HL (for > 4 transform levels) 5LH (for > 4 transform levels) 5HH (for > 4 transform levels) 6HL (for > 5 transform levels) 6LH (for > 5 transform levels) 6HH (for > 5 transform levels)
0x36	1	LOAD_SS	**Step Sizes Load Options** Bit 0 of this byte must be 0 prior to loading the step sizes and/or QFACT. After writing the step sizes or QFACT, set Bit 0 to 1 to tell the firmware that the step sizes or QFACT are valid and ready for use. After the firmware loads the new step sizes or QFACT, this bit is again reset to '0'.
0x37	1	LOAD_VW	**Visual Weighting Options** 0 = No visual weighting or use previously loaded visual weights. 1 = Load custom visual weights using VW_Y, VW_CB and VW_CR parameters. 2 = Load visual weights as recommended in ISO/IEC 15444-3 Standard for Motion JPEG2000
0x38	2 x 19	VW_Y	Visual weighting factors for luminance.
0x5E	2 x 19	VW_CB	Visual weighting factors for Cb
0x84	2 x 19	VW_CR	Visual weighting factors for Cr
0xAA	6	RESERVED	Reserved
0xB0	4 x 16	LTARGET [1:16]	Target size or target quality for each layer in multilayer mode

ADV212 Firmware Parameter Decode Settings

Table 9. DECODE CONFIGURATION PARAMETER SETTINGS

Big-Endian Byte Offset	Number of Bytes in Parameter	Parameter Name	Description
0x0	1	VFORMAT	**Video Standard** 0 = NTSC 4:2:2 1 = PAL 4:2:2 2 = 1080i Luminance 3 = 1080i Chrominance 4 = Custom-specific 5 = 720/60p Luminance 6 = 720/60p Chrominance 7 = Reserved 8 = NTSC de-interlaced 9 = PAL de-interlaced
0x1	1	PREC	**Precision** 0 = 8-bit 1 = 10-bit 2 = 12-bit
0x2	1	Reserved	
0x3	1	UNI	**Component polarity** 0 = Bipolar C, Bipolar Y 1 = Unipolar C, Bipolar Y 2 = Bipolar C, Unipolar Y 3 = Unipolar C, Unipolar Y Should be set to '3' for most applications.
0x4	1	Reserved	
0x5	1	Reserved	
0x6	1	Reserved	
0x7	1	Reserved	
0x8	1	Reserved	
0x9	1	PICFG	**Pixel Interface Configuration Settings** Bit 7 = Reserved (always write 0) Bit 6 = Use HVF pins 1 = Use external HVF pins 0 = Use embedded SAV/EAV Bit 5 = Reserved (write 0) Bit 4 = Decode (1 = master, 0 = slave) Bit 3 = FIELD polarity (1 = positive, 0 = negative) Bit 2 = HSYNC polarity (1 = positive, 0 = negative) Bit 1 = VSYNC polarity (1 = positive, 0 = negative) Bit 0 = VCLK polarity (1 = positive, 0 = negative)
0xA	1	DRES	**Decode Resolution Settings** Bit 7 = Scale image size enable 0 = Keep image size the same 1 = Scale image down to desired resolution size. See bits [2:0] for settings. Bits[6:3] = Reserved (write 0) Bits[2:0] = Highest resolution level to decode (up to maximum of XFORMLEV parameter) 0 = Decode all resolutions 1 = Decode to 1/4 resolution 2 = Decode to 1/16 resolution 3 = Decode to 1/64 resolution 4 = Decode to 1/256 resolution 5 = Decode to 1/1024 resolution 6 = Decode to 1/4096 resolution

Big-Endian Byte Offset	Number of Bytes in Parameter	Parameter Name	Description
0xB	1	COD_STYLE	**Output Code Stream Format:** COD_STYLE [2:0] 0 = ADV212 raw format 1 = J2C Format 2 = JP2 YCbCr 3 = JP2 Greyscale

ADV212 Firmware Parameter Descriptions

VFORMAT

NTSC 4:2:2	This setting should be used only for a resolution of 720 × 243 @ 60 fields/sec.
PAL 4:2:2	This setting should be used only for a resolution of 720 × 288 @ 50 fields/sec.
1080i Luminance	This setting should be used only for a resolution of 1920 × 540 @ 60 fields/sec.
1080i Chrominance	This setting should be used only for a resolution of 1920 × 540 @ 60 fields/sec.
Custom-specific	This setting must be used for all settings that do not conform to VFORMAT 0, 1, 2, 3, 5, 6, 7, 8 or 9.
720/60p Luminance	This setting should be used only for a resolution of 1280 × 720 @ 60 frames/sec.
720/60p Chrominance	This setting should be used only for a resolution of 1280 × 720 @ 60 frames/sec.
NTSC De-interlaced	This setting should be used only for NTSC standard definition at a resolution of 720 × 243@60 fields/sec. The input is in interlaced format, but the compression is on a frame-by-frame basis. The ADV212 compresses an odd and even field together. This can improve compression efficiency by up to 40-50%. De-interlaced mode can only handle 8-bit pixel component samples. VDATA[3:0] is ignored in encode and is set to 0 in decode.
PAL De-interlaced	This setting should be used only for PAL standard definition at a resolution of 720 × 288 @ 50 fields/sec. The input is in interlaced format, but the compression is on a frame-by-frame basis. The ADV212 compresses an odd and even field together. This can improve compression efficiency by up to 40-50%. De-interlaced mode can only handle 8-bit pixel component samples. VDATA[3:0] is ignored in encode and is set to 0 in decode.

CBSIZE

Code stream generation is facilitated with large code blocks, but code block height should be small to minimize ADV212 memory usage. For this reason, a code block size of 32 × 32 does not work for 1080i or 720p modes. The most robust code block size for 1080i and 720p modes is 128 × 32. This CBSIZE also provides the lowest latency.

The smaller the code blocks, the larger the number of code blocks that are generated. The CPU bandwidth required to perform rate control and J2K generation are directly related to the number of code blocks present.

For recommendations on how to estimate the number of code blocks per image, see the technical note "*How to estimate the number of codeblocks/image*".

ATTRTYPE

The ADV212 is configured to output code block attribute data via the ATTR FIFO if the ATTRTYPE parameter is set to 1, 2, or 3. Also, the host must enable both DMA channels when using this feature. One DMA channel is assigned to the CODE FIFO, and the other DMA channel is assigned to the ATTR FIFO.

If ATTRTYPE is set to 4, the 16-byte ADV212 compressed image header is output to the ATTR FIFO instead of being placed at the beginning of each compressed image. This allows systems to obtain this header information without the need to parse the code stream data.

If ATTRTYPE is set to 5, then a more detailed byte-length summary is output to the ATTR FIFO to support external merging of HD Luminance and HD Chrominance streams from two different ADV212s.

RCTYPE and RCVAL

This version of the firmware has five options for performing the rate-control algorithm. The first option does not truncate the compressed code block. The four other options truncate the compressed code block.

The options for the rate-control algorithm are as follows:

- No truncation
- Target size per video field or image
- Target quality per video field or image
- Multilayer target size
- Multilayer target quality

NO TRUNCATION

Each compressed code block is assembled into the code stream without being truncated, which provides the maximum possible quality. This also results in large variations in the number of bytes generated per compressed image.

TARGET SIZE PER VIDEO FIELD OR IMAGE

The number of bytes per compressed field or image is provided in the RCVAL parameter. If target size is chosen as a parameter, the data output rate must be within +5% to -100% of the programmed value.

For example, to achieve 12 Mbps (assuming NTSC 60 fields/sec), RCVAL should be set to (12,000,000/(8 ×60)) = 25,000 bytes per field = 0x0061A8. The rate-control algorithm tries to achieve this size with the best possible quality, but there are variations from field to field.

TARGET QUALITY PER VIDEO FIELD OR IMAGE

The target quality value is provided in the RCVAL parameter. As this value increases, the quality actually decreases, while the number of bytes per compressed field or image also decreases. If target quality is chosen as a parameter, the picture quality is held at a constant while the data output rate changes.

Assuming the default quantization step sizes are being used, highest quality is achieved at RCVAL = 0x000100 or below, and lowest quality is achieved at RCVAL = 0x000A00 and above.

MULTILAYER TARGET SIZE

This option allows the user to program bytes per compressed field or image per layer when using multilayered codestream output. The number of layers should be programmed into the RCVAL parameter. The target byte length for each layer must also be programmed. This is done by setting LTARGET.

To use Multilayer Target Size, perform the following procedure:

1. Set RCTYPE to 3.
2. Program RCVAL for the number of layers. RCVAL can be programmed from a minimum value of 1 up to a maximum of 16.
3. Set LTARGET values to the bytes per image for each layer.

MULTILAYER TARGET QUALITY

To use Multilayer Target Quality, perform the following procedure:

1. Set RCTYPE to 4.
2. Program RCVAL for the number of layers. RCVAL can be programmed from a minimum value of 1 up to a maximum of 16.
3. Set LTARGET values to the target quality for each layer.

Table 10. RCVAL SETTINGS FOR VARIOUS DATA OUTPUT RATES FOR TARGET SIZE

Video	Horizontal Pixels per Field	Lines per Field	Field Rate	Data Output Rate Mbps	Bytes per Field	RCVAL
NTSC	720	243	59.94	20	41708	0x00A2EC
NTSC	720	243	59.94	15	31281	0x007A31
NTSC	720	243	59.94	10	20854	0x005176
NTSC	720	243	59.94	9	18750	0x00493E
NTSC	720	243	59.94	8	16683	0x00412B
NTSC	720	243	59.94	7	14598	0x003906
NTSC	720	243	59.94	6	12513	0x0030E1
NTSC	720	243	59.94	5	10417	0x0028B1
NTSC	720	243	59.94	4	8342	0x002096
NTSC	720	243	59.94	3	6256	0x001870
NTSC	720	243	59.94	2	4167	0x001047
NTSC	720	243	59.94	1	2085	0x000825
PAL	720	288	50	20	50000	0x00C350
PAL	720	288	50	15	37500	0x00927C
PAL	720	288	50	10	25000	0x0061A8
PAL	720	288	50	9	22500	0x0057E4
PAL	720	288	50	8	20000	0x004E20
PAL	720	288	50	7	17500	0x00445C
PAL	720	288	50	6	15000	0x003A98
PAL	720	288	50	5	12500	0x0030D4
PAL	720	288	50	4	10000	0x002710
PAL	720	288	50	3	7500	0x001D4C
PAL	720	288	50	2	5000	0x001388
PAL	720	288	50	1	2500	0x0009C4

Table 11. RCVAL SETTINGS FOR TARGET QUALITY

RCVAL	Description
0x0100	Near mathematically lossless
0x0300	Visually lossless
0x0500	Noticeable artifacts for busy images, visually lossless for soft images
0x0700	Noticeable artifacts but details of image are still present
0x0900	Very noticeable artifacts; used for low bit rate applications

J2KPROG

This value selects one of the five progression styles defined in the ISO/IEC15444-1 standard. If J2KPROG is programmed to LRCP, then the layer information, the resolution, the component, and the position are decoded from the JPEG2000 stream in that order.

- LRCP : Layer-Resolution-Component-Position
- RLCP : Resolution-Layer-Component-Position
- RPCL : Resolution-Position-Component-Layer
- PCRL : Position-Component-Resolution-Layer
- CPRL : Component-Position-Resolution-Layer

QFACT, STEPSIZES, and LOAD_SS

One step size is assigned to each subband by default. After the host boot procedure, the ADV212 checks the LOAD_SS parameter. If the LOAD_SS parameter is set to 0, the default step sizes are written automatically to the STEPSIZES parameters.

The QFACT parameter provides a simple method to scale the quantization factors (inverse step sizes) by a fraction that is determined by dividing the QFACT parameter by 256 (if QFACT = 0, the quantization factor is set to 1). If custom step sizes are not to be used at startup (LOAD_SS = 0 after initialization), the QFACT parameter is applied to the default step sizes after they are written to the STEPSIZES parameters. Future changes to QFACT require the same procedure as changing the STEPSIZES parameters.

To use custom step sizes at startup, follow this procedure:

1. After writing the firmware to memory and before re-booting with 0x008D, write the new step sizes to the STEPSIZES parameters using the standard JPEG2000 step-size format.
2. Write 0x0001 to the LOAD_SS parameter.

To change the STEPSIZES or QFACT parameters during the encode capture process, follow this procedure:

1. Poll the LOAD_SS parameter (indirect address 0x00057F36) until it reads 0x0000.
2. Write the new step sizes to the STEPSIZES parameters using the standard JPEG2000 step-size format.
3. Write the new quantization factor to the QFACT parameter.
4. Write 0x0001 to the LOAD_SS parameter.

Just before the start of the next input video field, the program loads the new step sizes and/or QFACT, and then resets the LOAD_SS parameter to 0.

If custom step sizes are to be used at startup, these parameters must be provided and the LOAD_SS parameter must be written to 1.

COD_STYLE

COD_STYLE [2:0]

ADV212 compressed data output formats are described in "ADV202 Output Formats and Attribute Data Format".

COD_STYLE [7:4]

PPT = Packed packet headers for a tile. For more detail, see the ISO/IEC15444-1 specification.

PLT = Packet length header for a tile. For more detail, see the ISO/IEC15444-1 specification.

SOP = Start of packet header. For more detail see the ISO/IEC15444-1 specification.

EPH = End of packet header. For more detail see the ISO/IEC15444-1 specification.

Visual Weighting: LOAD_VW, VW_Y, VW_CB, and VW_CR

Initially, all wavelet subbands are weighted only by their contribution to the mean-square-error (MSE) of the decoded image. This is also called the L2Norm of the subband. However, you can reduce image compression artifacts by applying an additional weighting factor to each subband based on that subband's contribution to visual perception.

The weighting factor is typically a floating-point number between 0 and 1, but can also be larger than 1. However, the ADV212 requires a signed, twos-complement, 16-bit log2 representation of this value with 9,7 precision. A factor that is less than 1 results in a negative log2 value, while a factor that is larger than 1 results in a positive log2 value.

The LOAD_VW parameter is used to load the appropriate visual weighting table on startup. If this parameter is set to 0, no weighting is used. At any time, the user can have the firmware load either a predefined visual weighting table or a custom table for the VW_Y, VW_CB, VW_CR parameters:

1. Poll the LOAD_VW parameter [indirect register address 0x00057F37] until it reads 0x0000.
2. Write the visual weight to the VW_Y, VW_CB, and VW_CR parameters using the format previously described.
3. Load the visual weights.

   To load the custom visual weights, write 0x0001 to the LOAD_VW parameter.

   To load the predefined visual weights recommended in the ISO/IEC15444-3 standard for motion JPEG2000, write 0x0002 to the LOAD_VW parameter.

After the firmware loads the visual weights, LOAD_VW is reset to 0.

LTARGET [1:16]

These 16 values contain the desired target size for each layer in the code stream and are used with multilayer rate control programmed with RCTYPE.

LTARGET [1] represents a 32-bit word for layer 1, LTARGET [2] represents a 32-bit word for layer 2 etc.

RCTYPE = 3

LTARGET represents the target byte length of the layer. This includes bytes in all layers up to and including the specified layer.

RCTYPE = 4

LTARGET represents the target quality for each layer.

Code Example

A source code example for ADV212 standard definition encode/decode application can be found at:

ftp://ftp.analog.com/pub/Digital_Imaging/ADV212_Eval_P160SD_FPGA_SYSTEM/P160SD_system/projects/Spartan3SLC400/video_pipe/edk71/. See the latest revision.

The main function is contained in the TestApp.c file.

ADV212 configuration and initialization is contained in the ADV212.c file.

The code contains configuration information for a 32-bit normal host mode interface to transfer compressed data to and from the ADV212.

NOTES