

ADV202 Test Modes

by Christine Bako

INTRODUCTION

This application note applies to ADV202 engineering samples that are 0.2 (ES3) or higher, and are branded as follows:

ADV202xxx

SURF®

xxxxxx 0.2 or RevE

This application note outlines a series of recommended procedures and critical information to confirm correct hardware configuration and initialization of the ADV202. It provides test procedures for direct accesses and indirect accesses, tests to confirm correct DMA interface and JDATA mode operation, and addresses common problems the user may encounter. Performing these specific register access tests facilitates debugging systems for hardware and software errors.

DIRECT REGISTER ACCESS FUNCTIONALITY/PLL SETTINGS

Direct Register Access Functionality

Correct direct register access confirms functionality and correct timing on the following pins:

HDATA

$\overline{\text{ACK}}$

$\overline{\text{RD}}$ or $\overline{\text{WR}}$

$\overline{\text{CS}}$

ADDR

Note that direct register timing is specified differently from indirect register accesses. Refer to Figure 3 and Figure 4 in the ADV202 data sheet.

To perform direct register accesses:

- Properly connect the target system to the following pins on the ADV202:

ADDR<3:0> (up to 16 addresses).

$\overline{\text{CS}}$ (Chip Select).

$\overline{\text{WE}}$ (Write Enable).

$\overline{\text{RD}}$ (Read Enable).

$\overline{\text{ACK}}$ (Acknowledged).

HDATA<15:0> (read/write 16-bit data).

HDATA<31:16> (optional extended data bus for 32-bit accesses).

CFG [2:1] are shared with $\overline{\text{DREQ0}}$ and $\overline{\text{DREQ1}}$. The hardware boot mode as set by the state of the CFG pins is ignored on present ADV202s. Refer to Figure 3 and Figure 4 in the ADV202 data sheet for correct timing specifications in normal host mode operation.

- $\overline{\text{IRQ}}$
Connecting $\overline{\text{IRQ}}$ to the host is recommended.
- $\overline{\text{DACK0}}$ and $\overline{\text{DACK1}}$ must be held high during configuration.
- MCLK must be supplied. The PLL registers and MMODE and BUSMODE do not require a clock input to be read or written to, but it is recommended to power up with an MCLK running, otherwise the state of the HDATA pins is in an unknown state and might cause contention with other devices connected to HDATA.
- 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 ADV202 asserts 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 necessary only for asynchronous reads of the direct registers (normal host mode), and is not needed for DMA accesses. Refer to Figure 3 and Figure 4 in the ADV202 data sheet for normal host mode timing.

PLL Settings

The internal PLL of the ADV202 is extremely flexible, but there are rules on what settings are allowed for JCLK given the external MCLK frequency (for details see Page 30 of the ADV202 data sheet). JCLK is the internal frequency derived from the clock input on MCLK pin and set by the PLL registers.

To ensure proper functionality for specific applications, the internal JCLK frequency needs to be in a certain range. The following table summarizes the proper PLL settings for the target application.

For the tests described in this application note, the settings in Table I can be used.

Once the PLL registers are programmed, a delay of approximately 20 μ s is needed after which the ADV202 should undergo a reboot procedure as described later in this document.

Direct Register Read Test

Direct register access is required for all applications. Use the following procedure to check for correct direct register access:

1. Apply MCLK input.
2. Power up the ADV202.
3. Undergo a hard reset by asserting the RESET pin (holding LOW) for several cycles while still applying MCLK input.
4. Read the direct registers for their correct default value using normal host mode access. (See Figure 3 in the ADV202 data sheet.)

Direct Register Error States

Common problems encountered at this stage.

Problem: Direct register read does not come up with correct default values.

Possible cause:

1. $\overline{\text{DACK0}}$ or $\overline{\text{DACK1}}$ is not held high, and therefore some contention on the HDATA bus might occur.
2. Use MCLK input on power-up to prevent HDATA being in an unknown state.
3. Check for the correct PLL settings.

Problem: $\overline{\text{ACK}}$ is never asserted.

Possible cause:

1. Incorrect timing. Refer to Figure 3 in the data sheet.
2. Check for the correct PLL settings.

Table I. PLL Settings

Video Standard	MCLK Frequency	PLL_HI [address = 0xEh]	PLL_LO [address = 0xFh]
NTSC/PAL ITU.R.BT656	27	0x0008	0x0004
525p SMPTE293M	27	0x0008	0x0004
625p ITU.R.BT1358	27	0x0008	0x0004
1080i SMPTE274M	25	0x0008	0x0006
1080i SMPTE274M	74.25	0x0008	0x0084
1080i SMPTE274M	37.125	0x0008	0x0004

Table II. Direct Register Reset Values

Direct Address	Register Name	Register Value	Writeable
0x5	EIRQIE	0x0000	yes
0x6	EIRQFLG	0x000F	no
0x8	BUSMODE	0x0005	yes
0x9	MMODE	0x0009	yes
0xA	STAGE	undef	yes
0xB	IADDR	undef	yes
0xC	IDATA	undef	yes
0xE	PLL_HI	0x0008	yes
0xF	PLL_LO	0x0006 0x0003 [for ES3 samples]	yes

Direct Register Read/Write and Indirect Register Read/Write

Testing will verify correct functionality of the following:

- Correct direct register reset values
- Correct direct register write accesses
- Correct indirect memory read/write accesses using indirect registers IDATA, IADDR, STAGE.

The following test procedure checks the reset values of the direct registers, performs some direct register writes, and performs some simple indirect register accesses to verify that this interface is functioning properly. These processes are necessary to function correctly before starting any application-specific configuration.

Register Test Procedure (32-bit)

Using HDATA [31:0] with a 32-bit host processor:

1. Write to PLL_HI and PLL_LO the values specified in Table I.
2. Wait for 20 μ s to allow PLL to settle.
3. Initiate a reboot to no boot mode by writing 0x008A to the BOOT register (address 0xD).
4. Check that the reset values from Registers 5, 6, 8, 9, E, and F are correct.
5. Write 0x000A to BUSMODE (address 0x8) to enable 32-bit host interface.
6. Write 0x000A to MMODE (address 0x9) to enable 32-bit indirect access capability.
7. Write 0x001B0000 to IADDR (address 0xB) to set indirect address to internal RAM.
8. Write 0x12345678 to IDATA (address 0xC) to write data to address 0x001B0000.
9. Write 0x9ABCDEF0 to IDATA (address 0xC) to write data to address 0x001B0004.
10. Write 0x0F0F0F0F to IDATA (address 0xC) to write data to address 0x001B0008.
11. Write 0xF0F0F0F0 to IDATA (address 0xC) to write data to address 0x001B000C.
12. Write 0x001B0000 to IADDR (address 0xB) to set indirect address to internal RAM.
13. Read 0x12345678 from IDATA (address 0xC) to read data from address 0x001B0000.
14. Read 0x9ABCDEF0 from IDATA (address 0xC) to read data from address 0x001B0004.
15. Read 0x0F0F0F0F from IDATA (address 0xC) to read data from address 0x001B0008.
16. Read 0xF0F0F0F0 from IDATA (address 0xC) to read data from address 0x001B000C.

REGISTER TEST PROCEDURE (16-BIT)

Using HDATA [15:0] for Data Transfers

In this 16-bit mode, the data readback is half word swapped. This is due to an auto-increment indirect read anomaly on the ADV202. The data is written as expected in big-endian half word order, however the sequence of the read data is in little-endian half word order.

Another method around this anomaly is to always load the IADDR register with the desired address before reading from the IDATA register. Or, the STAGE register could be used during the reads of the IDATA register.

1. Write to PLL_HI and PLL_LO the values specified in Table I.
2. Wait for 20 μ s to allow PLL to settle.
3. Initiate a reboot to no boot mode by writing 0x008A to the BOOT register (address 0xD).
4. Check that the reset values from Registers 5, 6, 8, 9, E, and F are correct.
5. Write 0x0015 to BUSMODE (address 0x8) to enable 16-bit host interface (and JDATA).
6. Write 0x0005 to MMODE (address 0x9) to enable 16-bit indirect access capability.
7. Write 0x001B to STAGE (address 0xA).
8. Write 0x0000 to IADDR (address 0xB) to set indirect address to internal RAM.
9. Write 0x1234 to IDATA (address 0xC) to write data to address 0x001B0000.
10. Write 0x5678 to IDATA (address 0xC) to write data to address 0x001B0002.
11. Write 0x9ABC to IDATA (address 0xC) to write data to address 0x001B0004.
12. Write 0xDEF0 to IDATA (address 0xC) to write data to address 0x001B0006.
13. Write 0x0F0F to IDATA (address 0xC) to write data to address 0x001B0008.
14. Write 0x0F0F to IDATA (address 0xC) to write data to address 0x001B000A.
15. Write 0xF0F0 to IDATA (address 0xC) to write data to address 0x001B000C.
16. Write 0xF0F0 to IDATA (address 0xC) to write data to address 0x001B000E.
17. Write 0x001B to STAGE (address 0xA).
18. Write 0x0000 to IADDR (address 0xB) to set indirect address to RAM location.
19. Read 0x5678 from IDATA (address 0xC) to read data from address 0x001B0002.
20. Read 0x1234 from IDATA (address 0xC) to read data from address 0x001B0000.

21. Read 0xDEF0 from IDATA (address 0xC) to read data from address 0x001B0006.
22. Read 0x9ABC from IDATA (address 0xC) to read data from address 0x001B0004.
23. Read 0x0F0F from IDATA (address 0xC) to read data from address 0x001B000A.
24. Read 0x0F0F from IDATA (address 0xC) to read data from address 0x001B0008.
25. Read 0xF0F0 from IDATA (address 0xC) to read data from address 0x001B000E.
26. Read 0xF0F0 from IDATA (address 0xC) to read data from address 0x001B000C.

Common Errors

Problem: Random data or incorrect indirect register reads or can not overwrite direct register values.

Possible cause:

1. Incorrect timing, check for $\overline{\text{ACK}}$ hold time specification.
2. $\overline{\text{DACK0}}$ and $\overline{\text{DACK1}}$ are not held high during the test.
3. Check for the correct PLL settings.
4. Supply MCLK input on power-up and after ADV202 comes out of reset.

Problem: In 16-bit mode, half words are swapped. This applies to indirect memory accesses only, not to direct register accesses, when the ADV202 is set to MMODE = 0x0005.

In this case the readback value from indirect memory is half word swapped.

This is due to an auto-increment anomaly on the ADV202.

DMA AND JDATA TESTS

After successfully running these tests, the design is functional in

- Direct register access.
- Indirect memory access.
- Firmware load.
- ADV202 external DMA interface.
- JDATA interface.

DREQ/DACK READ TEST

`eval_edma_singlerd0_loop.sea`

Application ID = 0xFF04

This test uses the ADV202 to generate a data pattern which can be read using the External DMA Channel 0 ($\overline{\text{DREQ0}}$ and $\overline{\text{DACK0}}$ pins). It is important for the user to maintain the $\overline{\text{DACK0}}$ pin high, until the EDMA interface has been enabled by the Host [as in (15)]. This test should be performed prior to using the ADV202 in DREQ/DACK DMA modes [EDMOD0 and DMMOD0 set to 1, 2, 5, or 6].

The 64 word pattern is as follows:

```
00010203, 04050607, 08090A0B, 0C0D0E0F,
10111213, 14151617, 18191A1B, 1C1D1E1F, ...
F0F1F2F3, F4F5F6F7, F8F9FAFB, FCDFEFFF
```

DREQ/DACK TEST PROCEDURE

1. Write to PLL_HI and PLL_LO the values specified in Table I.
2. Wait for 20 μs to allow PLL to settle.
3. Initiate a reboot to no boot host mode by writing 0x008A to the BOOT register (address 0xD). This boot mode allows the firmware to be loaded over the host interface.
4. Write 0x000A to BUSMODE (address 0x8) to enable 32-bit host interface and configure for 32-bit wide HDATA bus.
5. Write 0x000A to MMODE (address 0x9) to enable 32-bit indirect register access capability.
6. Write 0x00050000 to IADDR (address 0xB) to set indirect address to start of program memory.
7. For each 32-bit value in the firmware application file [7.75 kB], write this value to IDATA (address 0xC) to load the program.
Firmware application file:
`eval_edma_singlerd0_loop.sea`
8. Initiate a reboot by writing 0x008D to the BOOT register (address 0xD).
9. Write 0x000A to BUSMODE (address 0x8) to enable 32-bit host interface and configure 32-bit wide HDATA bus after the SOFTRST in (8).
10. Write 0x0009 to MMODE (address 0x9) to enable 16-bit indirect access width and indirect address increment enabled.

11. Initialize external DMA register while this register is disabled:
Write 0xFFFF1408 to IADDR (address 0xB).
Write 0x0000300A to IDATA (address 0xC).
DMA is thus configured for single DMA transfers with $\overline{\text{DREQ0}}$ pulse width configured to six JCLK cycles. See Figure 9 in the ADV202 data sheet.
12. Initialize external interrupt and enable SWIRQ0:
Write 0x00000400 to EIRQIE (address 0x5).
13. Wait for $\overline{\text{IRQ}}$ to be asserted or poll the software flag register (address 0x7) for 0xFF04. This indicates initialization complete.
14. Clear interrupt flag:
Write 0x00000400 to EIRQFLG (address 0x6). This will cause the $\overline{\text{IRQ}}$ to go high and indicates the ADV202 is ready to run the program.
15. Enable external DMA register:
Write 0xFFFF1408 to IADDR (address 0xB).
Write 0x0000300B to IDATA (address 0xC).
 $\overline{\text{DREQ0}}$ should then be asserted within three MCLK cycles.
16. Host can now perform reads with $\overline{\text{DREQ0}}$, $\overline{\text{DACK0}}$ handshake after first $\overline{\text{DREQ0}}$ pulse is generated by the ADV202. The internal ADV202 processor generates the 64-word pattern as mentioned earlier, starting with 0x000010203.

DREQ/DACK READ/WRITE TEST USING BOTH DMA CHANNELS

eval_edma_singlewrrd0_2chn_loop

Application ID = 0xFF06

This test can be used to verify DREQ/DACK write and read transfers to/from the ADV202 and should be performed prior to using the ADV202 in HIPI mode or prior to using the ADV202 in DREQ/DACK DMA modes [EDMOD0/1 set to 1, 2, 5, or 6].

It uses external DMA Channel 0 ($\overline{\text{DREQ0}}/\overline{\text{DACK0}}$) for writing and Channel 1 ($\overline{\text{DREQ1}}/\overline{\text{DACK1}}$) for reading. It is important for the user to maintain the $\overline{\text{DACK0}}$ and $\overline{\text{DACK1}}$ pins high, until the EDMA interfaces have been enabled by the host [as under (16)]. This test supports up to 1,024 word write transfers before the buffer must be read out of DMA Channel 1.

Test Procedure

1. Write to PLL_HI and PLL_LO the values specified in Table I.
2. Wait for 20 μs to allow PLL to settle.
3. Initiate a reboot to no boot host mode by writing 0x008A to the BOOT register (address 0xD). This boot mode allows the firmware to be loaded over the host interface.
4. Write 0x000A to BUSMODE (address 0x8) to enable 32-bit host interface and configure for 32-bit wide HDATA bus.
5. Write 0x000A to MMODE (address 0x9) to enable 32-bit indirect register access capability.
6. Write 0x00050000 to IADDR (address 0xB) to set indirect address to start of program memory.
7. For each 32-bit value in the firmware application file [7.75 kB], write this value to IDATA (address 0xC) to load the program.
Firmware application file:
eval_edma_singlewrrd0_2chn_loop.sea
8. Initiate a reboot by writing 0x008D to the BOOT register (address 0xD).
9. Write 0x000A to BUSMODE (address 0x8) to enable 32-bit host interface and configure for 32-bit wide HDATA bus after the SOFTRST in (8).
10. Write 0x0009 to MMODE (address 0x9) to enable 16-bit indirect access width and indirect address increment enabled.
11. Initialize external DMA Registers 0 and 1 while this register is disabled:
Write 0xFFFF1408 to IADDR (address 0xB).
Write 0x0000300A to IDATA (address 0xC).
Write 0x0000300C to IDATA (address 0xC).
12. Initialize FIFO mode register in order to set the data flow direction for input and output data:
Write 0xFFFF1418 to IADDR (address 0xB).
Write 0x00000010 to IDATA (address 0xC).
13. Initialize external interrupt and enable SWIRQ0:
Write 0x00000400 to EIRQIE (address 0x5).
14. Wait for $\overline{\text{IRQ}}$ to be asserted or poll the software flag register (address 0x7) for 0xFF06. This indicates initialization complete.

15. Clear interrupt flag:
Write 0x00000400 to EIRQFLG (address 0x6). This will cause the $\overline{\text{IRQ}}$ to go high and indicates the ADV202 is ready to run the program.
16. Enable external DMA registers:
Write 0xFFFF1408 to IADDR (address 0xB).
Write 0x0000300B to IDATA (address 0xC).
Write 0x0000300D to IDATA (address 0xC).
Both DMA channels are configured for single DMA transfers with $\overline{\text{DREQ0}}$ pulse width configured to six JCLK cycles. See Figure 9 in the ADV202 data sheet. $\overline{\text{DREQ0}}$ should then be asserted within three MCLK cycles.
17. Host can now perform a write access with a $\overline{\text{DREQ0}}/\overline{\text{DACK0}}$ handshake after the first $\overline{\text{DREQ0}}$ pulse is generated by the ADV202. After data is written to the ADV202, $\overline{\text{DREQ1}}$ will be asserted, indicating that data can be read out. In this loop-back test, the data read out is the same as the data which was written into Channel 0. There is a 1,000 word buffer, so the host may write up to 1,000 words before doing the first read. The test will run continuously in an infinite loop.
7. For each 32-bit value in the firmware application file [7.75 kB], write this value to IDATA (address 0xC) to load the program.
8. Firmware application file:
eval_edma_jdatard0_loop.sea
9. Initiate a reboot by writing 0x008D to the BOOT register (address 0xD).
10. Write 0x0015 to BUSMODE (address 0x8) to enable 16-bit host and JDATA port.
11. Write 0x0009 to MMODE (address 0x9) to enable 16-bit indirect access width and indirect address increment enabled.
12. Initialize external interrupt and enable SWIRQ0:
Write 0x00000400 to EIRQIE (address 0x5).
13. Wait for $\overline{\text{IRQ}}$ to be asserted or poll the software flag register (address 0x7) for 0xFF05. This indicates initialization complete.
14. Clear interrupt flag:
Write 0x00000400 to EIRQFLG (address 0x6).
This will cause the $\overline{\text{IRQ}}$ to go high and indicates the ADV202 is ready to run the program.
15. Host can initiate JDATA reads by driving HOLD to the low state. VALID will go to high state in the following MCLK cycle and data transfers will begin on the JDATA port if HOLD and VALID are set to active high polarity. The data pattern is 0x00, 0x01, 0x02, ...0xFF, 0x00, 0x01, and so on.

JDATA READ TEST

eval_edma_jdatard0_loop

Application ID = 0xFF05

This test uses the ADV202 to generate a byte stream that can be read via the JDATA port which includes JDATA[7:0], VALID ($\overline{\text{DREQ0}}$ pin), and HOLD ($\overline{\text{DACK0}}$ pin). It is important for the user to maintain the HOLD ($\overline{\text{DACK0}}$) pin high, until the JDATA interface has been enabled by the ADV202. The data pattern is a sequential stream of byte values from 0x00 to 0xFF. This test should be used prior to using the ADV202 in a JDATA mode application.

1. Write to PLL_HI and PLL_LO the values specified in Table I.
2. Wait for 20 μ s to allow PLL to settle.
3. Initiate a reboot to no boot host mode by writing 0x008A to the BOOT register (address 0xD). This boot mode allows the firmware to be loaded over the host interface.
4. Write 0x000A to BUSMODE (address 0x8) to enable 32-bit host interface and configure for 32-bit wide HDATA bus.
5. Write 0x000A to MMODE (address 0x9) to enable 32-bit indirect register access capability.
6. Write 0x00050000 to IADDR (address 0xB) to set indirect address to start of program memory.

JDATA READ/WRITE TEST

eval_edma_jdatawrrd0_loop

Application ID = 0xFF07

This test can be used to verify JDATA writes and reads to and from the ADV202. The host sets up the number of words (number of bytes times four) to be written, enables the JDATA interface, and begins writing data. When the programmed number of words is reached, the ADV202 disables the JDATA interface direction, and JDATA can be read from the part after the JDATA interface has been re-enabled by the host. This continues in an infinite loop. This assumes a 16-bit host interface in JDATA mode. This test should be used prior to using the ADV202 in a JDATA mode application.

1. Write to PLL_HI and PLL_LO the values specified in Table I.
2. Wait for 20 μ s to allow PLL to settle.
3. Initiate a reboot to no boot host mode by writing 0x008A to the BOOT register (address 0xD). This boot mode allows the firmware to be loaded over the host interface.

4. Write 0x000A to BUSMODE (address 0x8) to enable 32-bit host interface and configure for 32-bit wide HDATA bus.
5. Write 0x000A to MMODE (address 0x9) to enable 32-bit indirect register access capability.
6. Write 0x00050000 to IADDR (address 0xB) to set indirect address to start of program memory.
7. For each 32-bit value in the firmware application file [7.75 kB], write this value to IDATA (address 0xC) to load the program.
Firmware application file:
eval_edma_jdatawrdd0_loop.sea
8. Initiate a reboot by writing 0x008D to the BOOT register (address 0xD).
9. Write 0x0015 to BUSMODE (address 0x8) to enable 16-bit host and JDATA port.
10. Write 0x0009 to MMODE (address 0x9) to enable 16-bit indirect access width and indirect address increment enabled.
11. Initialize external interrupt and enable SWIRQ0:
Write 0x00000400 to EIRQIE (address 0x5).
12. Configure the external DMA register:
Write 0xFFFF to STAGE (address 0xA).
Write 0x1408 to IADDR (address 0xB).
13. Wait for \overline{IRQ} to be asserted or poll the software flag register (address 0x7) for 0xFF07.
14. Clear interrupt flag:
Write 0x00000400 to EIRQFLG (address 0x6).
Initialization is now complete and the host can enter the number of words to be written.
15. Use COMMAND FIFO to initialize number of words to be written:
Write 0x4144 to COMMAND (address 0x4).
Write 0x5632 to COMMAND (address 0x4).
Write 0x0000 to COMMAND (address 0x4).
Write 0x0001 to COMMAND (address 0x4).
Write 0x0000 to COMMAND (address 0x4).
Write 0xFFFF to COMMAND (address 0x4).
The value of the last write is the number of 32-bit words to be written through the JDATA port. Since JDATA port width is eight bits, this value is four times the number of JDATA transfers. A small number such as 0x80 should be used initially until it's confirmed that the test is running properly.
16. Wait for \overline{IRQ} to go low again or poll SWFLG for 0x5555.
This indicates that the part is ready to accept data after the host enables the DMA interface. This is done now by writing the value 0x061B to the IDATA register (address 0xC). It is important to note that the data value is written to the EDMOD0 register.
This example sets VALID/HOLD to be both active high. This means that data will be transferred when $\overline{DREQ0}$ is high and $\overline{DACK0}$ is low. If opposite polarities are desired, the value written should be 0x001B. When polarities are changed, it is safer to first write the register with EDMA_En bit set to zero, then drive the HOLD pin to the active state (to suppress unwanted transfers) and then enable the EDMA interface.
17. Clear interrupt flag (EIRQFLG:Addr 0x6) and begin JDATA write. Write hex value 0x0400 to Direct Address 0x6. This will cause the \overline{IRQ} signal to go to high (inactive) state. JDATA writes can now begin. The number of bytes written should correspond to the number of words ($\times 4$) programmed in Step 7. Once the programmed number of words is accepted by the ADV202, it will disable the JDATA port and reinitialize the configuration for JDATA reading (to check the written data). This will be indicated by an IRQn low transition (or by polling for 0xaaaa in SWFLG).
18. Wait for \overline{IRQ} transition low (or poll SWFLG for value 0xaaaa). This indicates that the part is ready to provide data after the host has enabled the EDMA interface. This is done by writing the value 0x061B to the IDATA register (Addr:0xc). This is similar to Step 8, however, in this case it enables the EDMOD0 for reading. Again, data is transferred when $\overline{DREQ0}$ is high and $\overline{DACK0}$ is low. For opposite polarities, please follow the procedure described in Step 8.
19. Clear interrupt flag (EIRQFLG:Addr 0x6) and begin JDATA reads. Write hex value 0x0400 to Direct Address 0x6. This will cause the \overline{IRQ} signal to go to high (inactive) state. JDATA reads can now begin. The number of bytes read should correspond to the number of words ($\times 4$) programmed in Step 7. Once the programmed number of words is read from the ADV202, it will disable the JDATA port and reinitialize the configuration for JDATA writing (loops back to Step 8). This will be indicated by an \overline{IRQ} low transition (or by polling for 0x5555 in SWFLG).

Common Errors in JDATA Mode

Common test errors and their remedies:

Problem: Incorrect register values in JDATA mode.

For ES3 samples it is required to gate pins \overline{RD} and \overline{CS} , so that the \overline{RD} connected to the ADV202 only goes low when \overline{CS} is low as well. This will ensure that \overline{RD} only goes low when the ADV202 is being accessed and no bus contention will occur.

Problem: No initial assertion of VALID.

Under normal operations, initially the VALID will be asserted as soon as data is available in the CODE FIFO and should be followed by a HOLD deassert to start data transmission. On ES3 samples it is required to deassert HOLD first, regardless of the state of the CODE FIFO.

This is an ES3 bug and has been fixed on RevE or ES5 samples.

After successfully performing these tests, the ADV202 is ready to be used for specific applications.