TMS320DM357
Digital Media System-on-Chip
Silicon Revision 2.1

Silicon Errata

Literature Number: SPRZ285
November 2008

Contents

List of Figures

List of Tables

TMS320DM357 DMSoC **Silicon Revision 2.1**

1 Introduction

This document describes the known exceptions to the functional specifications for the TMS320DM357 Digital Media System-on-Chip (DMSoC). [See the *TMS320DM357 Digital Media System-on-Chip* data manual (literature number SPRS553).] Throughout this document, TMS320DM35x and DM35x refer to the TMS320DM357 device.

For additional information, see the latest version of the *TMS320DM357 DMSoC Peripheral Reference Guides*.

The advisory numbers in this document are not sequential. Some advisory numbers have been moved to the next revision and others have been removed and documented in the user's guide. When items are moved or deleted, the remaining numbers remain the same and are not resequenced.

This document also contains Usage Notes. Usage Notes highlight and describe particular situations where the device's behavior may not match presumed or documented behavior. This may include behaviors that affect device performance or functional correctness. These notes will be incorporated into future documentation updates for the device (such as the device-specific data sheet), and the behaviors they describe will not be altered in future silicon revisions.

1.1 Device and Development Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all devices and support tools. Each DM357 commercial family member has one of three prefixes: TMX, TMP, or TMS (e.g., **TMS**320DM357ZWT). Texas Instruments recommends two of three possible prefix designators for its support tools: TMDX and TMDS. These prefixes represent evolutionary stages of product development from engineering prototypes (TMX/TMDX) through fully-qualified production devices/tools (TMS/TMDS).

Device development evolutionary flow:

TMX	Experimental device that is not necessarily representative of the final device's electrical specifications
TMP	Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification
TMS	Fully-qualified production device

Support tool development evolutionary flow:

TMDX	Development-support product that has not yet completed Texas Instruments internal qualification testing
TMDS	Fully-qualified development-support product

TMX and TMP devices and TMDX development-support tools are shipped against the following disclaimer:

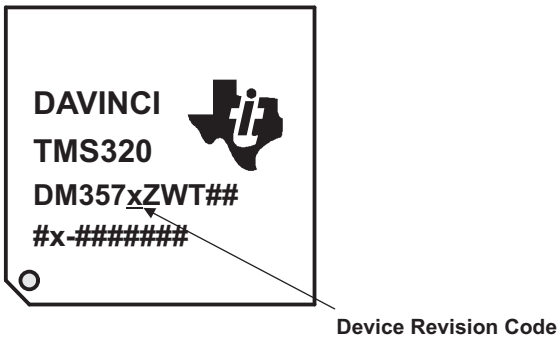
"Developmental product is intended for internal evaluation purposes."

TMS devices and TMDS development-support tools have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

Predictions show that prototype devices (TMX or TMP) have a greater failure rate than the standard production devices. Texas Instruments recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

1.2 Revision Identification

The device revision can be determined by the device revision code marked on the top of the package. The location of the device revision code for the ZWT package is shown in Figure 1. Figure 1 shows some examples of the types of DM357 package symbolization.



A Qualified devices are marked with the letters "TMS" at the beginning of the device name, while nonqualified devices are marked with the letters "TMX" or "TMP" at the beginning of the device name.

B "#" denotes an alphanumeric character. "x" denotes an alpha character only.

Figure 1. Example, Device Revision Codes for TMS320DM357 (ZWT)

Silicon revision is identified by a code on the chip. If x is "blank", then the silicon is revision 2.1 for TMS devices.Table 1 lists the silicon revisions associated with each device revision code for the DM357 device.

Table 1. Device Revision Codes

Device Revision Code (x)	Silicon Revision	Comments
(blank)	2.1	TMS320DM357ZWT

2 Silicon Revision 2.1 Usage Notes and Known Design Exceptions to Functional Specifications

2.1 Usage Notes for Silicon Revision 2.1

Usage Notes highlight and describe particular situations where the device's behavior may not match presumed or documented behavior. This may include behaviors that affect device performance or functional correctness. These notes will be incorporated into future documentation updates for the device (such as the device-specific data sheet), and the behaviors they describe will not be altered in future silicon revisions.

2.1.1 EDMA Transfer Request (TR) Dequeue Priority Limitation

On DM357 Silicon Revision 2.1, if there are multiple events in both Q0 and Q1, then the transfer requests associated with events in Q0 will get submitted to TC0 prior to any transfer requests associated with events in Q1 getting submitted to TC1 — even if TC0 is busy processing earlier transfer requests and TC1 is idling. This can cause delays in submission of requests on Q1. Therefore, it is recommended to reserve the higher priority Q0/TC0 for submission of urgent, small, real-time sensitive transfers (such as audio data transferred to and from ASP) and allocate Q1/TC1 for longer, non-real-time sensitive transfers (such as block memory DDR2 to internal memory and vice versa).

2.1.2 Bus Priority Inversion Can Affect DDR2 Throughput

On DM357 Silicon Revision 2.1, under certain conditions low priority modules can occupy the bus and prevent high priority modules like the VPSS from getting the required DDR2 throughput. The DDR2 memory controller arbitration policy gives preference to accesses to open banks, regardless of the requesting modules' priorities. This is not normally an issue, but can cause a problem if a low priority module performs extremely fast, contiguous accesses. This condition can effectively lock out other modules (even with higher priorities) trying to access the DDR2 memory controller for a long period.

For example, when the ARM is executing the STM (Store Multiple) instruction with the D-cache enabled, the ARM is able to achieve a high-peak transfer rate to cache and the DDR2 burst writes from the cache can stall the OSD (On-Screen Display) subsystem from DDR2 reads to the point that the display can be unusable. This can occur even though the VPSS, by default, has the highest priority. For more details on master peripheral priorities, see the "Bandwidth Management" subsection of the *TMS320DM357 DMSoC ARM Subsystem Reference Guide* (literature number SPRUG25).

The DDR2 memory controller Peripheral Bus Burst Priority Register (PBBPR address 0x2000 0020) contains a user-programmable field to indicate the maximum number of 32-byte DDR2 burst transfers that can go through before the DDR2 memory controller raises the priority of the oldest request in the queue. At reset, this value defaults to 255 (0xFF), meaning that this feature is disabled and a request can remain in the queue indefinitely.

It is recommended that the value of the DDR2 memory controller Peripheral Bus Burst Priority Register (PBBPR address 0x2000 0020) be reduced to limit the length of time that a peripheral can be held off due to the policy giving preference to the open bank. There is a performance trade-off between fast, low-priority peripherals and time-critical high priority peripherals when determining a value for this parameter. A hex value of 0x20 should provide a good ARM (cache enabled) performance and still allow good utilization by the VPSS or other modules.

2.1.3 Audio Serial Port (ASP) Transfers Should be Buffered in Internal Memory

On DM357 Silicon Revision 2.1, Audio Serial Port (ASP) transfers may need to originate and complete from on-chip buffers, either in ARM Internal RAM (TCM). This is due to the fact that there is no tolerance for audio data dropouts that may occur due to the delays in DDR2 accesses from other masters and from unavoidable DDR2 refresh cycles; even if the Q0/TC0 is dedicated to transfers from off-chip memories. On-chip buffers might be needed to ensure immunity from DDR2 latencies. DDR2 latencies are system-dependent, varying between applications, and are impacted by the amount and type of data traffic to DDR2 memories. Once completed, the data can be shuttled between the internal buffer and the DDR2 memory by using EDMA Q1/TC1.

2.1.4 DDR2 VTP I/O Calibration Must be Performed Following Device Power-up and Device Reset

On DM357 Silicon Revision 2.1, the DDR2 memory controller is able to control the impedance of the output I/O. This feature allows the DDR2 memory controller to tune the output impedance of the I/O to match that of the PCB board. Control of the output impedance of the I/O is an important feature because impedance matching reduces reflections, creating a cleaner signal propagation. Calibrating the output impedance of the I/O also reduces the power consumption of the DDR2 memory controller. The calibration is performed with respect to voltage, temperature, and process (VTP). The VTP information obtained from the calibration is used to control the output impedance of the I/O.

VTP I/O calibration ***must*** be performed following device power-up and device reset. If the DDR2 memory controller is reset via the Power and Sleep Controller (PSC), and the VTP input clock is disabled, accesses to the DDR2 memory controller will not complete. To re-enable accesses to the DDR2 memory controller, enable the VTP input clock, then perform the VTP calibration sequence again. The VTP calibration is part of the DDR2 memory controller initialization sequence. For more information on the VTP calibration and the proper DDR2 memory controller initialization sequence, see the *TMS320DM357 DMSoC DDR2 Memory Controller User's Guide* (literature number SPRUG38).

2.1.5 ASP: Initialization Procedure When External Device is Frame-Sync Master

On DM357 Silicon Revision 2.1, if the ASP transmitter expects a frame sync from an external device, care must be taken to ensure that the proper action is employed. After the transmitter comes out of reset (XRST = 1), it waits for a frame sync from the external device. If the first frame sync arrives very shortly after the transmitter is enabled, the CPU or EDMA controller may not have a chance to service the ASP data transmit register (DXR). In this case, the transmitter shifts out the default data in the transmit shift register (XSR) instead of the desired value, which has not yet arrived in the DXR. This causes problems in some applications such that the first data element in the frame is invalid. The data stream appears element-shifted (the first data word may appear in the second channel instead of the first).

To ensure proper operation when the external device is the frame master, you must make sure that the DXR is already serviced with the first word when a frame sync occurs. To do so, you can keep the transmitter in reset until the first frame sync is detected. The software is set up such that it will only take the transmitter out of reset (XRST = 1) promptly after detecting the first frame sync. This ensures that the transmitter does not begin data transfers at the data pin during the first frame-sync period. This also provides almost an entire frame period for the DM357 device to service the DXR with the first word before the second frame sync occurs. The transmitter only begins data transfers upon receiving the second frame sync. At this point, the DXR is already serviced with the first word.

The ASP transmitter and receiver on the DM357 device are capable of generating an interrupt upon the detection of frame synchronization. However, on the DM357 device, the receiver and/or transmitter must be out of reset to enable this feature. Therefore, instead of directly using the ASP interrupt to detect the first frame sync, on the DM357 device you can use the GPIO peripheral. This can be achieved by connecting the frame-sync signal to a GPIO pin. The software can either poll the GPIO pin to detect the first frame sync or program the GPIO peripheral to generate an interrupt to the CPU upon detecting the first frame-sync edge. For more information on the GPIO peripheral, see the *TMS320DM357 DMSoC General-Purpose Input/Output (GPIO) User's Guide* (SPRUG31). For details on the initialization sequence when the external device is the frame-sync master, see the *TMS320DM357 DMSoC Audio Serial Port (ASP) User's Guide* (SPRUG35).

2.1.6 SPI Master Mode: CSHOLD Bit Must be Initialized Twice After Reset

On DM357 Silicon Revision 2.1, in addition to the procedure described in Advisory 1.3.32 (SPI Master Mode: Extra Step Required to Use CSHOLD), the SPIDAT1.CSHOLD bit must be initialized twice with the same value after reset and before the first SPI transfer. This is required to clear an internal pipeline stage in the CSHOLD logic.

2.2 Silicon Revision 2.1 Known Design Exceptions to Functional Specifications

Table 2. Silicon Revision 2.1 Advisory List

Advisory 2.1.3 *VPFE: CCDC DC-Subtract Function Does Not Clip Luma to Zero for YUV Modes*

Revision(s) Affected: 2.1

Details: The CCD Controller (CCDC) in the VPFE subsystem includes an optional Luma DC-subtract function for YUV processing. This subtract operation does not clip Luma to zero.

Note: The Optical Black Clamp/DC-subtract function for the *Raw Data* modes ***does*** properly clip to zero for R/G/B and Ye/Cy/G/Mg color spaces.

Workaround(s): Do not use the DC-subtract function for YUV modes.

Advisory 2.1.4 *VPFE: CCDC Register Write Shadowing Does Not Work*

Revision(s) Affected: 2.1

Details: Register write shadowing in the CCDC does not work correctly (CCDCFG.VDLC = 0). Due to this bug, the CCDC PCR.ENABLE and the CCDCFG.YCINSWP fields are always shadowed, regardless of the setting of CCDCFG.VDLC. The SDR_ADDR register will only work correctly when CCDCFG.VDLC = 1, otherwise a delayed write to the SDR_ADDR occurs, creating an unexpected delay in outputting the image to SDRAM. Other registers, listed in the subsection *Programming the CCDC* under "Registering Accessibility During Frame Processing" of the *TMS320DM357 DMSoC Video Processing Front End (VPFE) User's Guide* (literature number SPRUG39), are affected such that shadowing does not occur on the next frame as specified.

Workaround(s): Set CCDCFG.VDLC = 1, such that all registers, except those noted above, are Busy-Writable, which forces register writes to take effect immediately. If changes to the other registers are required, the CCDC should be disabled, and the current frame should be allowed to complete. Register changes can then be made, and the CCDC can then be re-enabled.

Advisory 2.1.5 *VPFE: Pixel Misalignment on CCDC to Preview Engine Path*

Revision(s) Affected: 2.1

Details: There can be a timing glitch between the asynchronous VPFE input pixel clock and the internal VPFE clock that can cause pixel misalignment on the CCDC to PREV path. This misalignment is observed as causing a "pink" effect on the processed image (since the color pattern is misaligned). Once this first frame is adversely affected, the PREV hardware does not reset itself properly for subsequent frames, so the output will keep looking pink for the duration of the Preview mode. A similar issue on the end pixel can occur as well.

Workaround(s): Define the Preview Engine processing frame to start no earlier than the second pixel on a line and to end no later than the second-to-last pixel on a line. This requires the CCDC to be configured to transfer more pixels per line than is needed by the Preview Engine.

Advisory 2.1.11 *VPBE: RGB666 Pin Mux Option Does Not Work*

Revision(s) Affected: 2.1

Details: The intention of the RGB666 pin mux option (PINMUX0 register at 0x01C4 0000) is to allow the users to configure **PWM2/B2/GPIO47** and **PWM1/R2/GPIO46** pins as B2 and R2 functions, respectively. However, due to a hardware limitation, setting the RGB666 bit (PINMUX0.22) ***does not*** enable the B2 and R2 functions.

Workaround(s): Enable the RGB888 pin mux option (PINMUX0.23 = 1).

When the RGB888 pin mux option is enabled, the **B2** and **R2** pin functions are available; however, by enabling the RGB888 pin mux option, the six additional pins lose their GPIO pin function capability (see Table 3).

Table 3. RBG888/RBG666 Pin Mux Options

PINMUX0		PIN FUNCTIONS							
RGB888 (Bit 23)	**RGB666 (Bit 22)**	**PWM2/ B2/ GPIO47**	**PWM1/ R2/ GPIO46**	**R1/ GPIO38**	**B1/ GPIO6**	**G1/ GPIO5**	**C_FIELD/ R0/ GPIO4**	**LCD_FIELD/ B0/ GPIO3**	**G0/ GPIO2**
0	0	GPIO47	GPIO46	GPIO38	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2
0	**1**	**B2**	**R2**	GPIO38	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2
1	x	**B2**	**R2**	*R1*	*B1*	*G1*	*R0*	*B0*	*G0*

Advisory 2.1.12 *VPBE: Restriction on Horizontal Width for RGB888 Video Windows*

Revision(s) Affected: 2.1

Details: When a video window is configured for RGB888 data-input mode, certain horizontal width configurations result in corrupted video windows. The problem occurs at different widths, depending on enabling horizontal zoom (1x, 2x, and 4x) and/or 9/8 horizontal expansion modes.

Workaround(s): To work around this problem, the following constraints must be met for each case:

Case 1a: Video Window 0 (in RGB888 mode)

When 1x, 2x, 4x zoom, or 9/8 expansion is enabled, VIDWIN0**XL** must **NOT** be inside the ranges defined below:

- Z* (179 + 256N – 3) pixels < VIDWIN0**XL** < Z*(179 + 256N + 3) pixels
- Z*(261.5 + 256N – 5.5) pixels < VIDWIN0**XL** < Z*(261.5 + 256N + 5.5) pixels

for all integer values of N where:

Z = 1, 2, 4, or 9/8

XL register is the video window's horizontal display width in pixels (window width).

EXAMPLE:

When 1x zoom (Z = 1) is enabled on video window 0, the prohibited VIDWIN0**XL** ranges are the following:

1. (179 + 256N – 3) < VIDWIN0**XL** < (179 + 256N + 3) for all integer values of N
2. (261.5 + 256N – 5.5) < VIDWIN0**XL** < (261.5 + 256N + 5.5) for all integer values of N

For example,

- VIDWIN0**XL** = 436 will not work because this value falls inside the first range defined above when N = 1 (432 < VIDWIN0**XL** < 438).
- VIDWIN0**XL** = 1026 will not work because this value falls inside the second range defined above when N = 3 (1024 < VIDWIN0**XL** < 1030).
- VIDWIN0**XL** = 1024 will work because this value does not fall inside either ranges defined above for any N.

Case 1b: Video Window 0 (in RGB888 mode)

When (2x)*(9/8) or (4x)*(9/8) is enabled, VIDWIN0**XL** must **NOT** be inside the ranges defined below:

- Z* (179 + 256N – 3) pixels < VIDWIN0**XL** < Z*(179 + 256N + 3) pixels
- Z*(261.5 + 256N – 5.5) pixels < VIDWIN0**XL** < Z*(261.5 + 256N + 5.5) pixels
- VIDWIN0**XL** = (Z/2)*(1 + 72N)

for all integer values of N where:

Z = 2 when (2x)*(9/8) is enabled and

Z = 4 when (4x)*(9/8) is enabled

XL register is the video window's horizontal display width in pixels (window width).

Case 2a: Video Window 1 (in RGB888 mode)

When 1x, 2x, 4x zoom or 9/8 expansion is enabled, VIDWIN1**XL** must **NOT** be inside the ranges defined below:

- Z*(91.5 + 256N – 5.5) pixels < VIDWIN1**XL** < Z*(91.5 + 256N + 5.5) pixels
- Z*(173.5 + 256N – 3.5) pixels < VIDWIN1**XL** < Z*(173.5 + 256N + 3.5) pixels

for all integer values of N where:

Z = 1, 2, 4, or 9/8

XL register is the video window's horizontal display width in pixels (window width).

Case 2b: Video Window 1 (in RGB888 mode)

When (2x)*(9/8) or (4x)*(9/8) is enabled, VIDWIN1**XL** must **NOT** be inside the ranges defined below:

- Z*(91.5 + 256N – 5.5) pixels < VIDWIN1**XL** < Z*(91.5 + 256N + 5.5) pixels
- Z*(173.5 + 256N – 3.5) pixels < VIDWIN1**XL** < Z*(173.5 + 256N + 3.5) pixels
- VIDWIN1**XL** = (Z/2)*(1 + 72N)

for all integer values of N where:

Z = 2 when (2x)*(9/8) is enabled and

Z = 4 when (4x)*(9/8) is enabled

XL register is the video window's horizontal display width in pixels (window width).

Advisory 2.1.13 *USB: Extraneous USB Interrupts Generated*

Revision(s) Affected: 2.1

Details: When the DM357 device is in high-speed (HS) USB peripheral mode and suspended, an extraneous SUSPEND interrupt can be generated if the host issues a RESET. Although an extraneous SUSPEND interrupt is generated, the reset interrupt still arrives as expected at the end-of-reset sequence.

Also, when the DM357 device is in peripheral mode (full-speed [FS] or high-speed [HS]) and being reset to HS mode (RESET with chirping), an extraneous RESET interrupt can be generated during the reset sequence.

Workaround(s): To work around the problem, the following constraints must be met:

1. Software should ignore SUSPEND interrupts when already in a "suspended" state.
2. Software *must* service every USB RESET interrupt. The interrupt flags *must* be cleared before doing any register reads or setups so that any following USB interrupts are not missed.

Advisory 2.1.18 *SPI: Receive Overrun Interrupt and Bit Error Can be Lost*

Revision(s) Affected: 2.1

Details: Receive Overrun Interrupt (RXOVINT) and Bit Error interrupt (BITERRINT) can be lost if: Reading of the SPIFLG register coincides with the setting of these interrupt flag bits. Reading of the upper 16 bits of SPIBUF register coincides with the setting of these interrupt bits.

Workaround(s): Use the interrupt instead of the polling method to check the status of these interrupts.

Access only the lower 16 bits of the SPIBUF register to read received data.

If the polling method must be used, group the error interrupts into one Level (i.e., Level0) and the RX complete interrupt into the other Level (i.e., Level1). Use the SPIINTVECT0 and SPIINTVECT1 registers to find out the interrupt status first and then only read the SPIFLG register to decode the source of the error interrupts.

Advisory 2.1.19 *SPI: RXINTFLG Bit in SPIFLG Register May Not Get Cleared*

Revision(s) Affected: 2.1

Details: The RXINTFLG bit in the SPIFLG register may not get cleared by reading the SPIBUF register when the read coincides with the setting of the RXINTFLG bit due to new data arrival.

Workaround(s): When the above condition occurs, the system is at the verge of receive overrun. Therefore, either optimize the SPIBUF servicing routine to avoid receive overrun or use the EDMA3 to avoid the race condition from occurring.

Advisory 2.1.20 *SPI: A Write to SPIFLG Receiver Overrun Bit Does Not Clear the Flag*

Revision(s) Affected: 2.1

Details: A write to the SPIFLG receiver overrun (SPIFLG.OVRNINTFLG) bit does not clear the flag if the write coincides with the setting of the receive interrupt flag (SPIFLG.RXINTFLG).

Workaround(s): Write to the SPIFLG.OVRNINTFLG bit, then read back the value of the flag. If the flag did not clear, then write to clear the flag again.

Advisory 2.1.21 *SPI: The Receive Overrun Interrupt Flag is Not Set*

Revision(s) Affected: 2.1

Details: The Receive Overrun Interrupt flag is not set if the overrun condition that sets the interrupt flag coincides with a read of the upper bytes (bits 31:24) of the SPIBUF register.

Workaround(s): None.

Advisory 2.1.27 *USB: Some Electrical Parameters Violate USB Specification*

Revision(s) Affected: 2.1

Details: Some electrical characteristics violate the USB 2.0 specification; see Table 4.

Workaround(s): Consider this violation and design your system accordingly.

Table 4. USB Electrical Characteristics in Violation

		USB SPECIFICATION		DM357 DATA MANUAL		UNIT
		MIN	MAX	MIN	MAX	
V_{HSDSC} (1)	USB high-speed disconnect detection threshold (differential signal amplitude)	525	625	525	The lesser of: 1. VOD_DIS (2) - 75 2. 710	mV

(1) V_{HSDSC} violates the USB 2.0 specification.
(2) VOD_DIS = High-speed differential output voltage during a disconnected state.

Advisory 2.1.30 *SPI: SPIINTVECT and SPIFLG Registers are Cleared When Read in Debug Mode*

Revision(s) Affected: 2.1

Details: Both the INTVECT and SPIFLG registers are cleared when refreshing the memory window in debug mode with CCS. These registers should be cleared only by regular CPU reads, not during debug/suspend mode.

Workaround(s): None

Advisory 2.1.31 *SPI: SPI Master Receives Extra Bit When SPICLK Polarity Changes*

Revision(s) Affected: 2.1

Details: If the polarity of the SPICLK pin is changed and the change aligns with the receive edge for the new buffer, then it will be considered as a real SPICLK edge and the receive shift register shifts the data.

Workaround(s): Pre-select the SPIFMTx register by byte writing to just the DFSEL field in the SPIDAT1 register before actually writing to the SPIDAT1 field of the SPIDAT1 register. This additional step needs to be done only when there is going to be an SPICLK polarity change for the new buffer.

Advisory 2.1.32 *SPI Master Mode: Extra Step Required to Use CSHOLD*

Revision(s) Affected: 2.1

Details:

The SPI module chip-select hold (CSHOLD) feature allows the device to instruct the SPI to keep the chip-select pin asserted between transfers. This feature applies in master mode and is enabled by writing a '1' to SPIDAT1.CSHOLD (bit 28).

When data is written to the SPIDAT1 register with the CSHOLD bit set to '1', the master is supposed to keep the SPI_ENx pin asserted after the transfer completes. When data is written to the SPIDAT1 register with CSHOLD set to '0', the master is supposed to de-assert the SPI_ENx pin after the transfer completes.

For example, assume that the device needs to send two 16-bit words (0x1234 and 0x5678) to an SPI slave that requires its chip select to remain asserted between the transfers. This is a common requirement when communicating with SPI memory devices.

According to the SPI specification, the following sequence should produce the expected result as illustrated in Figure 2:

- Write 0x10001234 to SPIDAT1 for transmission of 0X1234 (CSHOLD = 1)
- Write 0x00005678 to SPIDAT1 for transmission of 0x5678 (CSHOLD = 0)

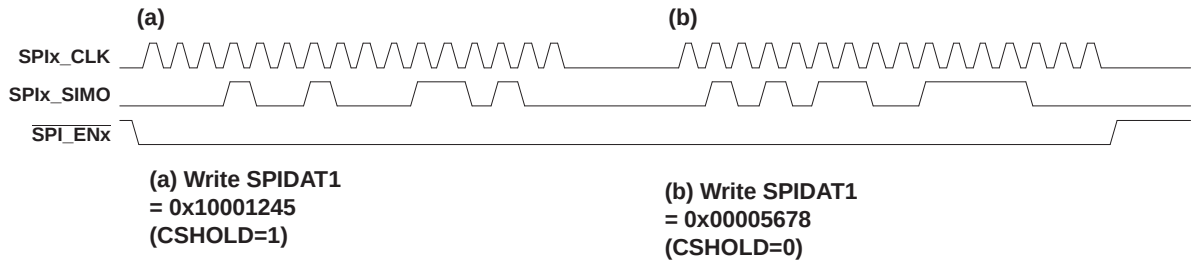


Figure 2. Expected CSHOLD Behavior

Instead, what actually occurs is that SPI_ENx is momentarily de-asserted at the beginning of the second write, as illustrated in Figure 3.

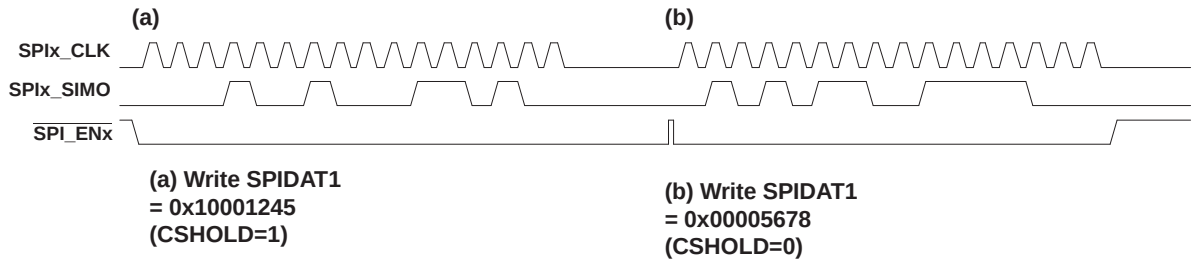


Figure 3. Actual CSHOLD Behavior–32-Bit Writes to SPIDAT1

Both Figure 2 and Figure 3 assume that SPIDAT1 is written using a single 32-bit write instruction. If SPIDAT1 is instead written using an 8-bit or 16-bit instruction to write to the CSHOLD field, followed by a 16-bit write to the transmit shift register field of SPIDAT1, then what actually occurs is illustrated in Figure 4. This is the same case illustrated in Figure 3 except that the de-assertion of SPI_ENx lasts for the duration between writing a '0' to the CSHOLD field and writing new data to the transmit shift register.

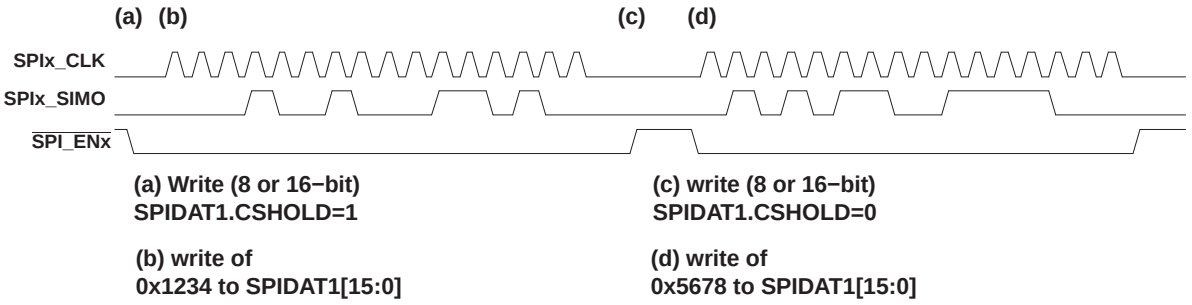


Figure 4. Actual CSHOLD Behavior–Halfword Writes to SPIDAT1

Workaround(s): For each word in the sequence of words during which SPI_ENx should be held low, write to the SPIDAT1 register with the CSHOLD bit set to '1'. Follow this by a write to only the CSHOLD field of SPIDAT1, setting CSHOLD = 0 to de-assert SPI_ENx. See Figure 5 for an illustration.

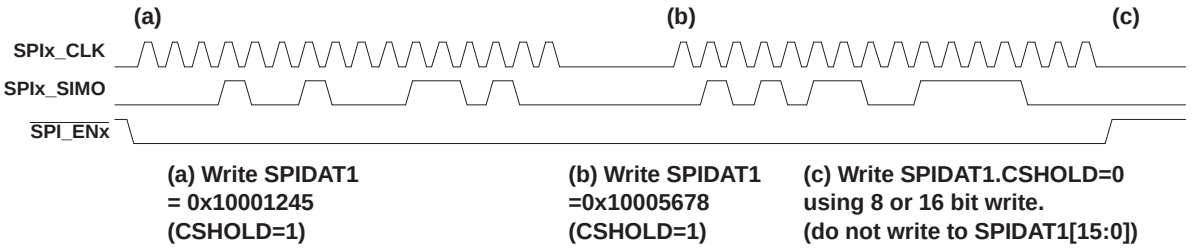


Figure 5. Workaround Assuming 32-Bit Writes to SPIDAT1 Followed by a Write Only to CSHOLD

Alternatively, only write to the SPIDAT1 CSHOLD field before and after the transfer to toggle the SPI_ENx pin. During the transfer, write only to the data field of SPIDAT1[15:0] using 16-bit (halfword) write commands. For an illustration, see Figure 6.

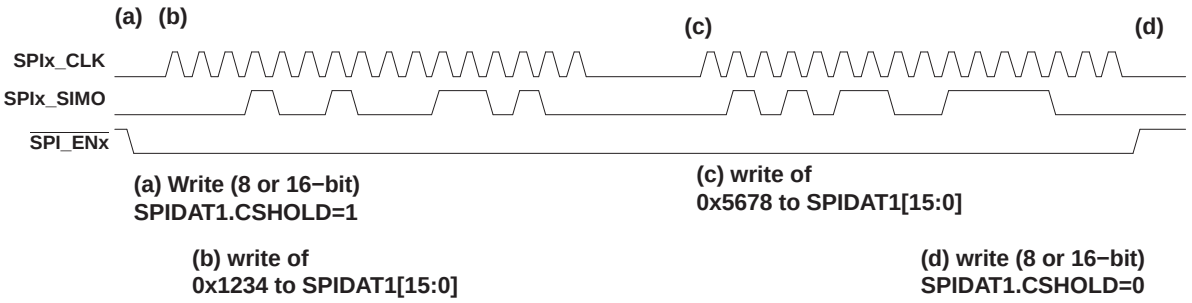


Figure 6. Workaround Assuming Halfword Writes to SPIDAT1

Advisory 2.1.34 *VPBE: Restriction When 6/5 Vertical Expansion Filter is Enabled*

Revision(s) Affected: 2.1

Details: The video windows are corrupted when the 6/5 vertical Expansion Filter is enabled under the following configuration: both video window 0 and video window 1 are enabled, 6/5 expansion is enabled, expansion filter is enabled, and the windows are at specific vertical coordinates.

Additionally, the video windows are corrupted when 6/5 expansion is enabled, expansion filter is enabled, and 4x or 2x vertical zoom is enabled for video window 0.

Workaround(s): To work around this problem, when the 6/5 vertical Expansion Filter is enabled, the following constraints ***must be*** met:

- (VIDWIN1YP + VIDWIN1YL - VIDWIN0YP) + 1 = Z*(6N)
  Where: Z = 1, 2, or 4 is the vertical zoom factor for video window 0 or 1, N is an integer, and YP register is a window's vertical position (upper-left pixel offset from base pixel).
- Additionally, when 6/5 vertical Expansion Filter is enabled, 4x or 2x vertical zoom cannot be enabled in video window 0.

Advisory 2.1.37 *VPFE: Preview Engine Hangs When the Video Port is Enabled in CCDC*

Revision(s) Affected: 2.1

Details: For RGB Bayer pattern input data, the CCDC can be setup in such a way that it can output raw data to DDR2 while concurrently sending the same data via its video port to the Preview Engine, H3A and Histogram modules. The Preview Engine and Histogram modules can alternatively get input data from DDR2 while the H3A module can only get input data from the CCDC via the video port. In other words, the following con-current data path should work.

CCDC → H3A

CCDC → DDR2 → Preview Engine → DDR2

The problem is that these two data paths can not work con-currently. Once the video port is enabled in the CCDC by setting FMTCFG.VPEN = 1, the Preview Engine hangs after a while, i.e., the second data path breaks when the first is enabled. When the Preview engine hangs, its PCR.BUSY bit stays at 1 and the interrupt is never triggered. A VPSS reset is needed to bring the Preview Engine back to normal functional mode.

The Preview Engine is configured in one shot mode and its data source is from memory.

Workaround(s): To workaround this issue implement one of the following recommendations:

1. Disable the CCDC video port (FMTCFG.VPEN = 0) while using the Preview Engine in one shot mode getting data from memory.
2. When H3A is used, configure the Preview Engine to get input from video port.

Advisory 2.1.40 *VPBE: VENC Default Luma Interpolation Filter Does Not Clip to Zero*

Revision(s) Affected: 2.1

Details: The Video Encoder (VENC) in the VPBE subsystem includes an optional 2x interpolation function for the luma signal. The default filter used for this interpolation (VMISC.YUPF = 0), *does not* clip the luma to zero.

Workaround(s): ***Do not*** use the default setting for luma 2x interpolation filter (VMISC.YUPF = **0**).
Instead, use the alternate setting for luma 2x interpolation filter (VMISC.YUPF = **1**).

Advisory 2.1.43 *USB (Device Mode): Calculated CRC Value Does Not Match Host CRC Value*

Revision(s) Affected: 2.1

Details:

The USB Controller can occasionally calculate a bad CRC for a received data packet. This error is rare and only occurs when ***ALL*** of the following conditions are met:

- USB Controller is in Device Mode of Operation and is receiving data
- Received data packet has a good CRC value of 0x7FF2
- A timing violation caused by a synchronization error (race condition)

The timing synchronization error is caused by a race condition between two control signals in the PHY Clock and System Clock domains. When these two synchronized control signals are crossing a clock boundary and the received data packet has a good CRC value of 0x7FF2, a race condition may occur causing one of the control signals to be latched a few pico-seconds ahead of the other control signal.

The issue has been observed on both Bulk (Non-Isochronous) and Isochronous transfers and may potentially exist on Control and Interrupt transfers since the data paths for all these transfers are the same or are very similar.

When the problem occurs in Non-Isochronous transfer types, the data that was "in-flight" to the USB Controller's FIFO from the Host is discarded by the USB Controller. Due to the error condition, the USB Controller also refrains from sending an ACK packet to the Host, as mandated by the USB transfer protocol. This forces the Host to re-transmit the data packet, anticipating an error in data transmission. The problem is usually corrected when the Host re-transmits the data packet.

When this problem occurs in Isochronous transfer mode for either High- or Full-speed, the USB Controller flags the device application S/W that a CRC error existed but retains the received data within the FIFO as well as captures the received data packet size value minus one byte from the actual data size. Since the magnitude of the actual timing violated due to the synchronization problem is only in pico-seconds, the entire data sent from the Host is routed into the USB device receive FIFO (i.e., even though the received data counter is one byte less, the full data packet is available for the USB driver).

Workaround(s):

Case 1a: Non-Isochronous Transfers (High-Speed): For non-Isochronous transfers operating in High-Speed mode, the Host and Device H/W perform the necessary re-transmission; thererfore, the issue should be transparent to the Host driver. The issue will also be transparent to the USB device driver since the H/W flushes the received data and forces the Host driver to re-transmit by not sending an ACK packet. For this reason, no interrupt is generated by the H/W to signify an error condition to the device-side application S/W.

Although quite rare, when both the Host and Device are operating in High-Speed mode and all the three consecutive transmissions did not occur without an error, the Host will use a PING packet at a later time to check if the endpoint is ready for accepting data. Upon the Host receiving an ACK packet in response to the PING packet, the Host re-initiates the previously failed transmission again. This process continues until the transfer takes place without error. For this reason, the Non-Isochronous High-Speed transfer is immune to this issue except for a throughput reduction for the time it takes for the re-transmission.

Case 1b: Non-Isochronous Transfers (Full-Speed): For non-Isochronous transfers operating in Full-Speed mode, it is recommended that the Host driver be constructed in such a way that it invokes the transfer multiple times prior to forcing a reset to the USB device. When the transfer is repeated, it is expected for the transfer to complete "error-free".

If the Host driver is not "set up" to invoke multiple failed transfers then, the Host driver will reset the USB driver, re-enumerate, and continue from where it left off.

Case 2: Isochronous Transfers (High- and Full-Speed): For Isochronous Transfers operating in either High- or Full-Speed modes, upon receiving a CRC error, the USB controller flags the device application S/W that a CRC error existed but will retain the received data within the FIFO as well as capture the received data packet size value minus one byte from the actual data packet size. Since the magnitude of the actual timing violated due to the synchronization problem is only in pico-seconds, the entire data sent from the Host is routed into the USB device receive FIFO (i.e., even though the received data counter is one byte less, the full data packet is available for the USB driver) and the USB driver should ignore the received CRC error and read one more additional byte from the receive FIFO. This one-byte counter difference is transparent to the Host H/W and S/W.

Due to the rare occurrence of this issue and its very minimal impact on applications, there are no plans to correct this issue in future silicon revisions.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Clocks and Timers	www.ti.com/clocks	Digital Control	www.ti.com/digitalcontrol
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265