AM3517, AM3505 Sitara™ Processors
Silicon Revisions 1.1, 1.0

Silicon Errata

Literature Number: SPRZ306E
October 2009–Revised September 2016

Contents

List of Figures

List of Tables

AM3517, AM3505 Sitara™ Processors Silicon Revisions 1.1, 1.0

1 Introduction

This document describes the known exceptions to the functional specifications for the AM3517 and AM3505 Sitara™ARM® processors. (For more detailed information, see *AM3517, AM3505 Sitara Processors*.)

The advisory numbers in the document are not sequential. Some advisory numbers have been moved to the next revision. When items are moved, the remaining advisory numbers are not resequenced.

Throughout this document, unless otherwise specified, AM35x, refers to all AM35x family devices. For additional peripheral information, see the latest version of the *AM35x ARM Microprocessor Technical Reference Manual*.

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 ARM microprocessors and support tools. Each commercial ARM microprocessor platform member has one of three prefixes: X, P, or null (no prefix). 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 (TMDX) through fully qualified production devices/tools (TMDS).

Device development evolutionary flow:

X	Experimental device that is not necessarily representative of the final device's electrical specifications
P	Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification
NULL	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

X and P 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 (X or P) 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 symbols marked on the top of the package. Figure 1 provides an example of the device markings.

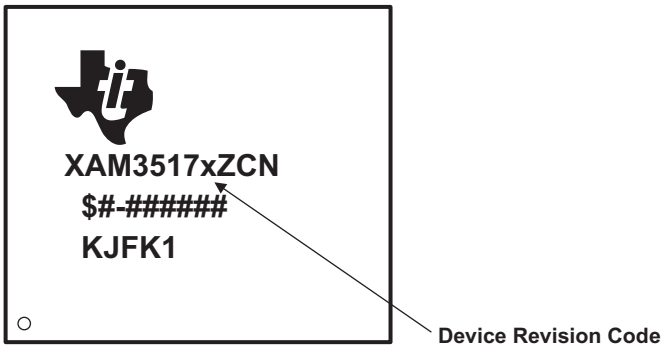


Figure 1. Example, Device Revision Codes for AM35x Microprocessor (ZCN Package)

NOTES:

(A) Non-qualified devices are marked with the letters "X" or "P" at the beginning of the device name, while qualified devices have a "blank" at the beginning of the device name.

(B) "#" denotes an alphanumeric character.

(C) On some "TMX" devices, the device speed may not be shown.

Silicon revision is identified by a code marked on the package. The code is of the format X3517xZCN, where "x" denotes the silicon revision. On TMX devices, if x is "BLANK", then the silicon is revision 1.0. Table 1 and Table 2 list the information associated with each silicon revision on TMX devices.

Table 1. AM35x Device Revision Codes

DEVICE REVISION CODE (x)	SILICON REVISION	COMMENTS
(BLANK)	1.0	This silicon revision is currently available as X *only*. Silicon revision 1.0 (also referred to as ES1.0)
A	1.1	

Each AM35x silicon revision uses a specific revision of the ARM Cortex®-A8 processor. Table 2 lists the ARM Cortex-A8 variant and revision can be read from the Main ID Register.

The ROM code revision can be read from base address 4001 BFFCh. The ROM code version consists of two decimal numbers: major and minor. The major number is always 14, minor number counts ROM code version. The ROM code version is coded as hexadecimal readable values, e.g., ROM version 14.04 will be coded as 0000 1404h. Table 2 shows the ROM code revision for each silicon revision of the device.

Table 2. Silicon Revision Variables

SILICON REVISION	ARM CORTEX-A8 VARIANT/REVISION	ROM REVISION
1.0	r1p7	15.58
1.1	r1p7	15.58

2 Silicon Revision 1.1 Usage Notes and Known Design Exceptions to Functional Specifications

2.1 Usage Notes for Silicon Revision 1.1

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 manual), and the behaviors they describe will not be altered in future silicon revisions.

Note: The peripherals supported on the various AM35x microprocessors are different. The user should only refer to usage notes and advisories pertaining to features supported on the specific device. For a complete list of the supported features of the AM35x microprocessors, see the device-specific data manual.

2.1.1 Cortex-A8 Errata List

This document does not cover the advisories associated with the Cortex-A8 processor. For a list of the advisories associated with the each version of the Cortex-A8 processor, contact your TI representative for the latest copy of the *ARM Core Cortex-A8 (AT400/AT401/AT490)* **Errata Notice**. See Table 2 to determine which version of the Cortex-A8 processor is included in eachAM35x silicon revision.

2.1.2 SGX Errata List

On the AM35x devices, the SGX is considered to be a "black box" for most users; therefore, no detailed information about SGX advisories will be provided in this document. SGX advisories are discussed in the Imagination Technologies™ errata, which is available for API/Code Developers *only*.

2.1.3 Performance Limitation on LCD Read/Write Access Through RFBI L4 Port

Back-to-back accesses for both *Read* and *Write* to the LCD through the L4 interconnect interface of the RFBI module are not supported. A penalty of 1 L4 clock cycle between two consecutive accesses exists.

Read access to the LCD through the RFBI L4 port: The data of a *Read* access is sent back to the initiator of the access only at RECycleTime. RECycleTime is used as a reference event for CS release as well (CS is used as an on-going access notification signal depending on the type of LCD panel connected). This means that any *Read* access to the LCD through the L4 interface of the RFBI will be ended by a CS release (CS going inactive at the end of the access). Therefore, the does not support two consecutive *Read* accesses to the LCD (there is always 1 L4 clock cycle between 2 accesses).

Write access to the LCD through the RFBI L4 port: For a *Write* access, the back-to-back data of a single access that has been split is supported and guaranteed (e.g., one 32-bit write split in two consecutive, back-to-back, 16-bit accesses). In this case, the internal bus CS signal will be kept active until split access completion. However, when there are two different write accesses from the initiator, they will not be seen as back-to-back by the LCD. At the end of the transaction corresponding to one write access from the initiator, the internal bus CS will be released at WECycleTime, and the next command from the initiator will be accepted. Consequently, any *Write* access to the LCD through the L4 interface of the RFBI will be ended by a CS release (CS going inactive at the end of the access). Therefore, the does not support two consecutive data transmits to the LCD (corresponding to two different and consecutive *Write* accesses from the initiator).

2.1.4 VENC: Last Data Line Missing in PAL

The VENC cannot show the image data at the end of the even frames on line 623. This is a minor violation of the ITU-R BT.470-6. Since lines 23 and 336 are reserved for WSS, the VENC can only show 574 lines of data instead of the 576 lines defined in the standard. One half line is missing for lines 23 (reserved for WSS) and 623, and one full line is missing for line 336 (reserved for WSS).

2.1.5 UART: Cannot Acknowledge Idle Requests in Smartidle Mode When Configured for DMA Operations

When configured for DMA operations using *smartidle* mode (SYSC[4:3].IDLEMODE = 0x2), the UART module will not acknowledge incoming idle requests. As a consequence, it can prevent L4 from going to idle.

When there are additional expected transfers, the UART should be placed in *force-idle* mode.

2.1.6 Unexpected RFBI Latency for High Frame Rate

The Remote Frame Buffer (RFBI) state machine architecture can create non-optimized pipelining of the RFBI output data stream. As a result, some idle cycles may be inserted between RFBI accesses to the external panels. The number of idle cycles added depends on theAM35x clock configuration and RFBI configuration. Table 3 shows the minimum additional number of cycles depending on the RFBI configurations.

Table 3. Additional Minimum Cycletime (CS/WE Always Asserted)

RFBI PERFORMANCE	RFBI_CONFIG. CYCLEFORMAT	RFBI_CONFIG. OCPFORMAT	MIN CYCLETIME (Number of OCP cycles)
OCP Slave	1 pixel/ cycle	1 pixel	5
	1 pixel/ 2 cycles	1 pixel	4
	1 pixel/ 3 cycles	1 pixel	4
	2 pixels/ 3 cycles	1 pixel	6
	1 pixel/ cycle	2 pixels	4
	1 pixel/ 2 cycles	2 pixels	4
	1 pixel/ 3 cycles	2 pixels	4
	2 pixels/ 3 cycles	2 pixels	6
Display Controller	1 pixel/ cycle	N/A	4
	1 pixel/ 2 cycles	N/A	3
	1 pixel/ 3 cycles	N/A	3
	1 pixel/ 3 cycles	N/A	6

2.1.7 HS USB Host Subsystem: Some Limitations Exist When Connecting to External Devices

As shown in Figure 2, the device includes a high-speed universal serial bus (USB) OTG controller and a high-speed USB host subsystem.

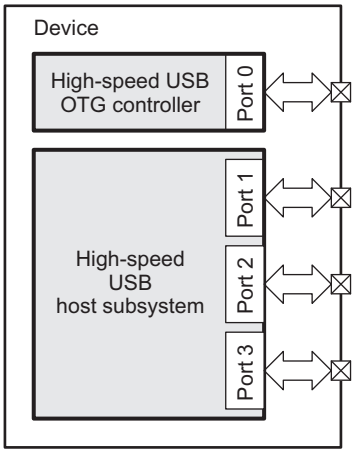


Figure 2. AM35x USB Modules

The high-speed USB OTG controller supports a single USB port which uses a UTMI + interface to connect to the integrated USB OTG PHY.

As shown in Figure 3, the high-speed universal serial bus (USB) host subsystem supports up to three USB ports, each one of which can owned by one of two controllers:

- The EHCI controller, based on the Enhanced Host Controller Interface (EHCI) specification for USB Release 1.0, is in charge of high-speed traffic (480M bit/s) over either a UTMI port or a UTMI low-pin interface (ULPI) port.
- The OHCI controller, based on the Open Host Controller Interface (OHCI) specification for USB Release 1.0a, is in charge of full-speed/low-speed traffic (12/1.5M bit/s, respectively) over a serial interface.

Please note the following functional limitations when using the high-speed USB host subsystem:

- If one port is configured in the ULPI mode, then all other ports must use the same configuration. Therefore, the ports must be configured high-speed mode or full-speed/low-speed mode. For more information, see ECHI and OHCI Controllers Cannot Work Concurrently.
- USB Port 3 cannot operate in ULPI mode; it can only operate in serial mode.

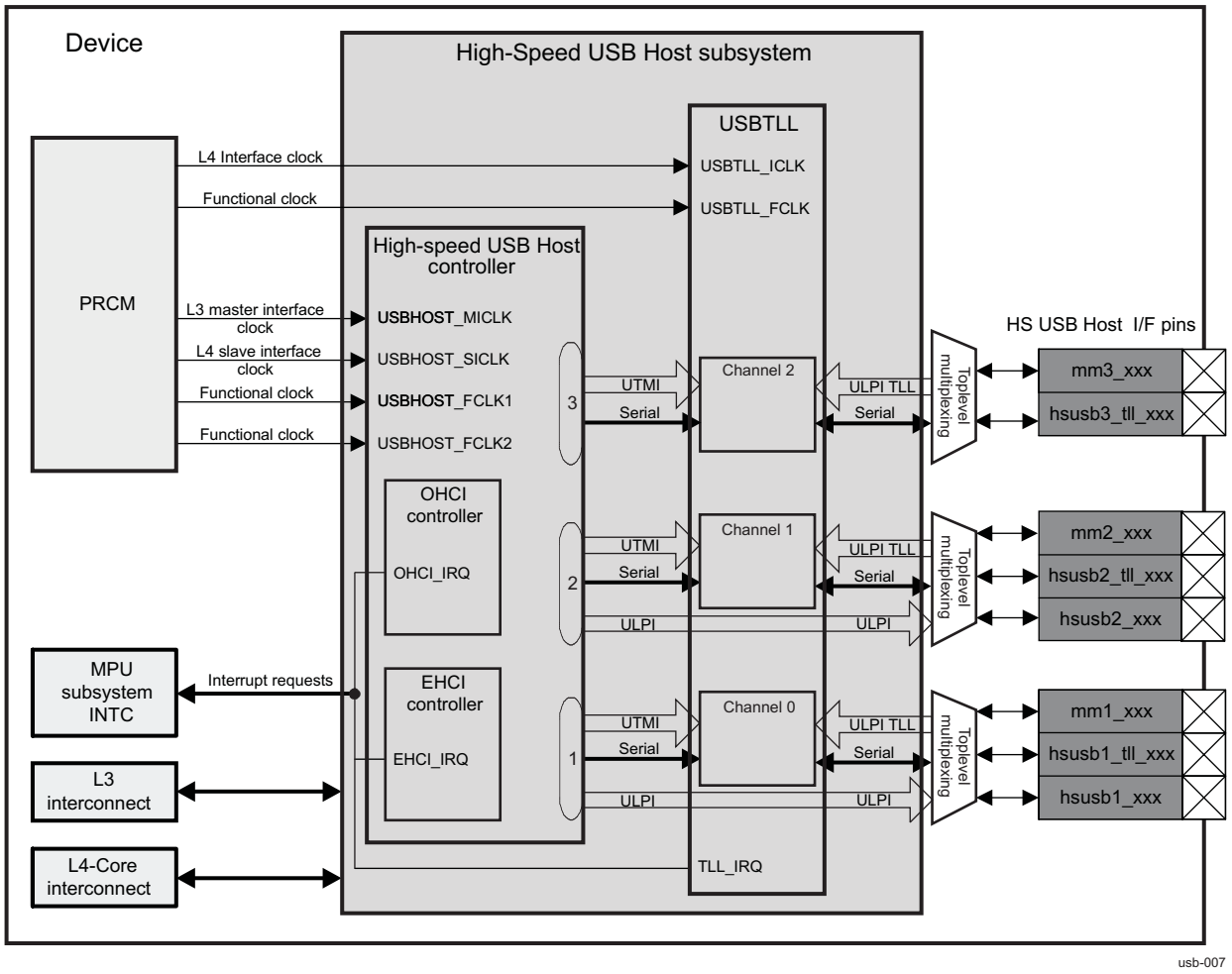
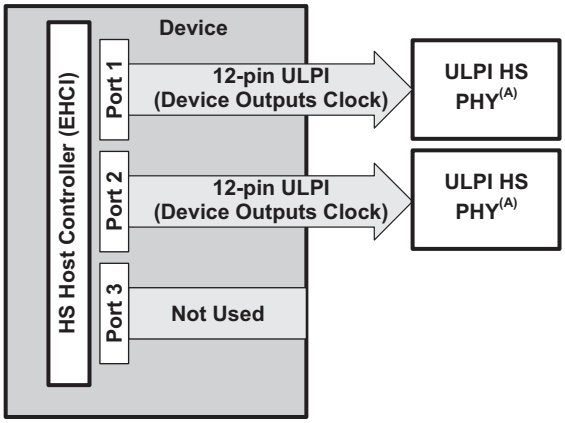


Figure 3. High-Speed USB Host Subsystem Highlight

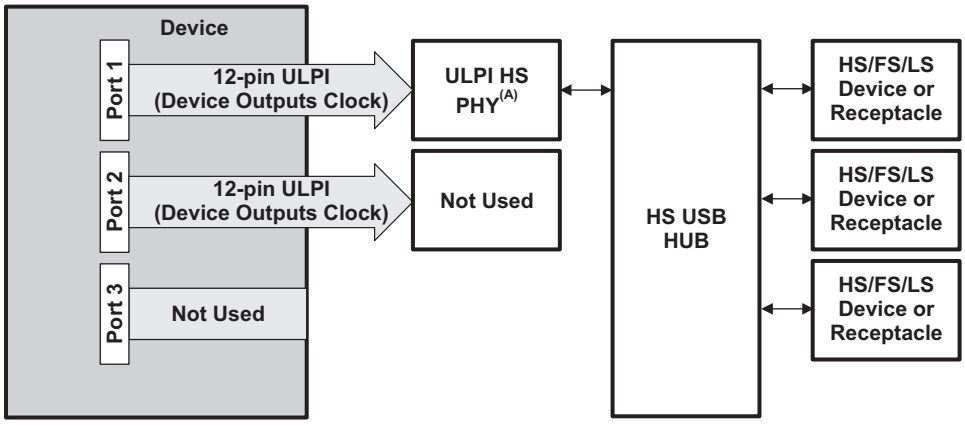
As shown in Figure 4, USB Port 1 and Port 2 can be used to connect to external high-speed PHYs which include a ULPI port. However, in this case, the external USB PHY must be able to accept a source clock generated by the high-speed USB controller. Also, in this usage model the USB ports cannot support full-speed and low-speed operation. Therefore, using this approach, USB Port 1 and Port 2 cannot provide a fully compliant USB 2.0 Type-A receptacle; a high-speed USB hub would be required in this case; please see Figure 5.

Note: USB Port 3 does not support ULPI mode.



A. USB port 3 does not support ULPI mode.

Figure 4. Connecting to High-Speed PHYs Using USB Ports



A. USB port 3 does not support ULPI mode.

Figure 5. Connecting to a High-Speed USB Hub

2.2 Silicon Revision 1.1 Known Design Exceptions to Functional Specifications

Table 4. Silicon Revision 1.1 Advisory List

Table 4. Silicon Revision 1.1 Advisory List (continued)

Advisory 1.1.1 *I2C Module Does Not Allow 0-byte Data Requests*

Revision(s) Affected 1.1 and earlier

Details When configured as the master, the I2C module does not allow 0-byte data transfers.

Note: Programming I2Ci.I2C_CNT[15:0]: DCOUNT = 0 will cause undefined behavior.

Workaround(s) No workaround. ***Do not*** use 0-byte data requests.

Advisory 1.1.2 *Delay Required to Read Some GP, WD, and Sync Timer Registers After Wakeup*

Revision(s) Affected 1.1 and earlier

Details If a General Purpose Timer (GPTimer) is in *posted* mode (TSIRC.POSTED = 1), due to internal resynchronizations, any values read in TCRR, TCAR1, and TCAR2 registers immediately after the timer interface clock (L4) goes from a stopped state to an active state may not return the expected values. This situation is most likely when the wakes up from an idle state.

Notes:

- GPTimer non-posted synchronization mode is not impacted by this advisory.
- This advisory also impacts reads from Watchdog timers WCRR registers.
- All of the watchdog timers support only *posted* internal synchronization mode. There is no capability to change the internal synchronization scheme to *non-posted* mode via software.
- The 32K sync timer CR and 32SYNCNT_REV registers are also impacted by this advisory, since the 32K sync timer is always in *posted* synchronization mode.

Workaround(s) The software must wait at least 2 timer interface clock cycles + 1 timer functional clock cycle after L4 clock-wakeup before reading TCRR, TCAR1, or TCAR2 registers for GP Timers in *posted* internal synchronization mode, and before reading the WCRR register of the Watchdog timers. The same workaround must be applied before reading CR and 32KSYNCNT_REV registers of the synctimer module.

Advisory 1.1.3 *Race Condition May Cause I2C Slave to Nack a Transfer*

Revision(s) Affected 1.1 and earlier

Details If the I2C module is configured as a slave, in *autoidle* mode (I2C_SYSC.AUTOIDLE = 1) and the ARDY (I2C.12C_STAT[2]) condition and the START condition are detected in the module at the same time, internal clock gating will be incorrectly applied. This will cause the I2C to NACK (I2C.12C_STAT[1]) the transfer for which the START (I2C.I2C_STA[6]) condition was received. Subsequent transfers will be ACKed as expected.

Workaround(s) **Workaround 1**: Software ***must*** set SYSC_AUTOIDLE to 0. In this case, the failure condition never occurs.

Workaround 2: Ensure that the external I2C master always resends a NACKed transfer via software. If a transfer was NACKed because of this race condition, the next transfer will always be ACKed.

Advisory 1.1.4 *MDR1 Access Can Freeze UART Module When in IrDa Mode*

Revision(s) Affected 1.1 and earlier

Details Because of a glitchy structure inside the UART module, accessing the MDR1 register may create a dummy underrun condition and freeze the UART IrDa transmission. Only IrDa modes *Slow Infrared* (SIR), *Medium Infrared* (MIR), and *Fast Infrared* (FIR) are impacted. Even if the bug condition occurs in *UART* mode or *IrDa CIR* mode, it will have no effect. Therefore, UART1 and UART2 are immune to this problem, and only UART3 may exhibit this issue when used in one of the IrDa modes– SIR, MIR, or FIR.

Workaround(s) To ensure this problem does not occur, the following software initialization sequence must be used each time MDR1 must be changed to one of the three failing *IrDa* modes:

1. If needed, setup the UART by writing the required registers, except MDR1.
2. Set appropriately the MDR1.MODE_SELECT bit field.
3. Wait for 5 L4 clock cycles + 5 UART functional clock cycles.
4. Read RESUME register to resume the halted operation.

Advisory 1.1.5 *I2C: RDR Flag may be Incorrectly Set*

Revision(s) Affected 1.1 and earlier

Details Under certain rare conditions, the I2C_STAT[13].RDR bit may be set as well as the corresponding interrupt fire, even when there is no data in the receive FIFO, or the I2C data transfer is still ongoing. These spurious RDR events must be ignored by the software.

Workaround(s) Software must filter out unexpected RDR pulses, using the flowchart illustrated in Figure 6 when receiving an I2C RDR interrupt.

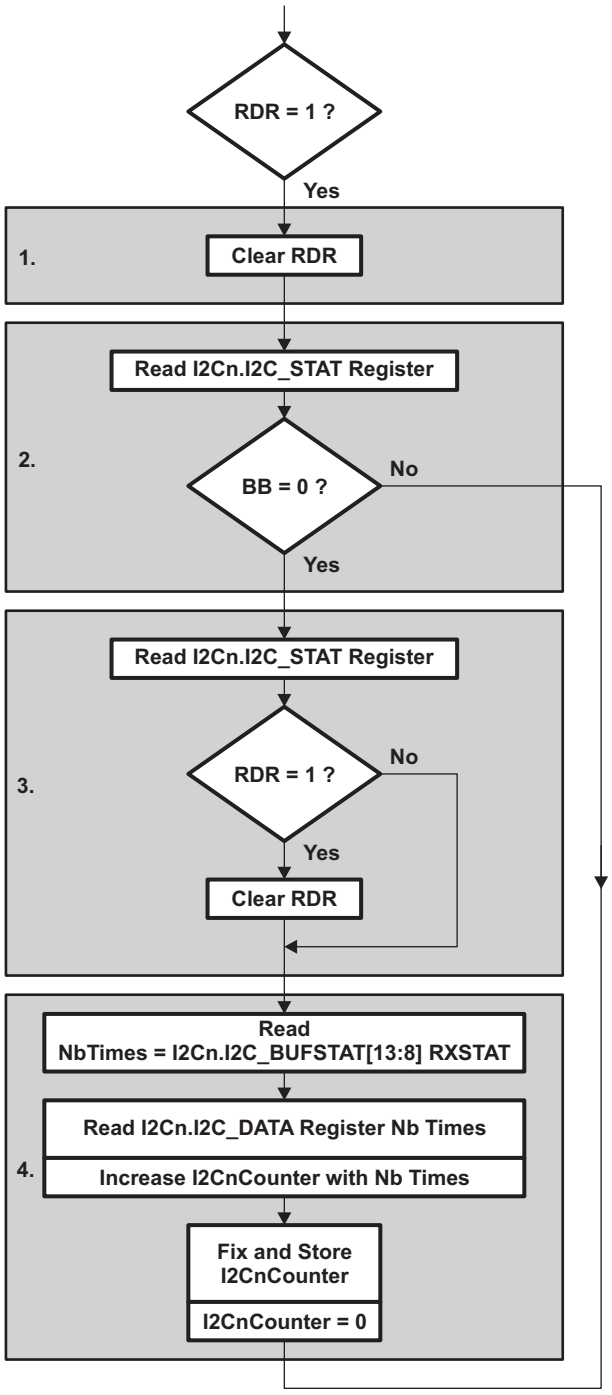


Figure 6. I2C Flowchart

Advisory 1.1.6 *GPMC may Stall After 256 Write Accesses in NAND_DATA, NAND_COMMAND, or NAND_ADDRESS Registers*

Revision(s) Affected

1.1 and earlier

Details

The GPMC may stall if the following conditions are met:

1. GPMC_CONFIG[0].NANDFORCEPOSTEDWRITE=1.
2. Software performs more than 256 continuous write accesses in NAND_COMMAND_x, NAND_ADDRESS_x or NAND_DATA_x registers.
3. GPMC_STATUS[0].EMPTYWRITEBUFFERSTATUS is always 0 (buffer not empty) during write accesses. This means the software has to write fast enough in GPMC registers in order to never have the write buffer empty.

This mechanism is CS independent. If the software performs 128 write accesses in NAND_DATA_0 followed by 128 write accesses in NAND_DATA_1 then the bug will occur.

Workaround(s)

Accesses performed through the "prefetch and write posting engine" of the GPMC are not impacted by this limitation, and software should use this mechanism for the best performance.

If the prefetch and write posting engine is not used, when GPMC_CONFIG[0].NANDFORCEPOSTEDWRITE=1 and after 255 write accesses in NAND_COMMAND_x, NAND_DATA_x or NAND_ADDRESS_x registers, the software has to wait until GPMC_STATUS[0].EMPTYWRITEBUFFERSTATUS=1 before sending the next 255 write accesses.

Advisory 1.1.7 *SPI Dummy DMA RX Request Generation*

Revision(s) Affected 1.1 and earlier

Details A dummy DMA RX request is generated as soon as the SPI is configured in *slave* mode and a SPI clock edge is detected. The dummy DMA RX request is generated during module configuration, when the module is not performing an SPI transfer. The dummy DMA RX request occurs as soon as the SPI interface signal sensitivity is changed compared to the default value.

The dummy DMA RX request is generated because the mechanism to avoid dummy data capture on a CS glitch is done regardless of channel activation.

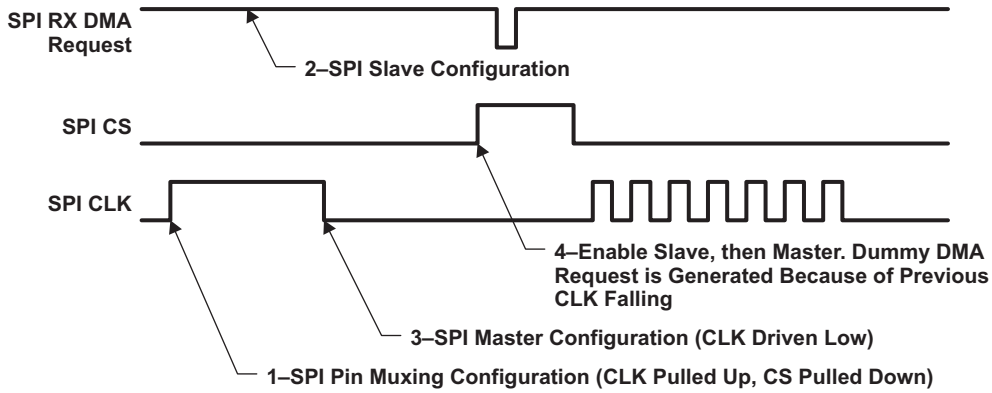


Figure 7. SPI Dummy DMA RX Generation

Workaround(s) Avoid conditions where the SPI is in *slave* mode and the SPI clock toggles (see examples below).

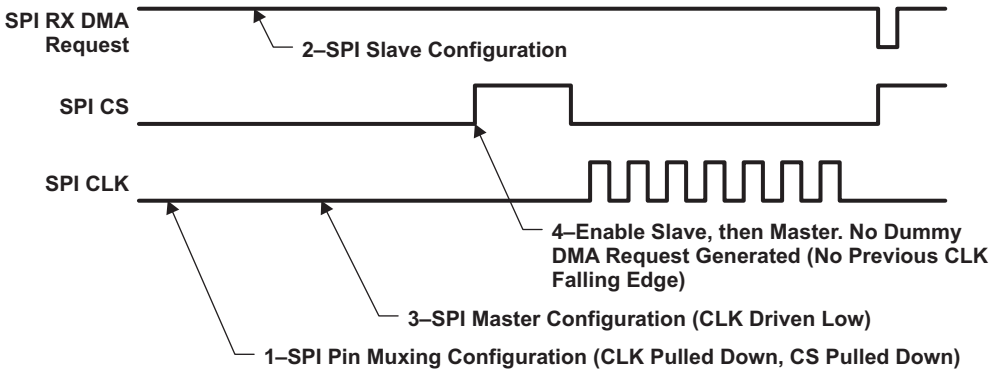


Figure 8. Dummy DMA RX Generation (No Clock Edge)

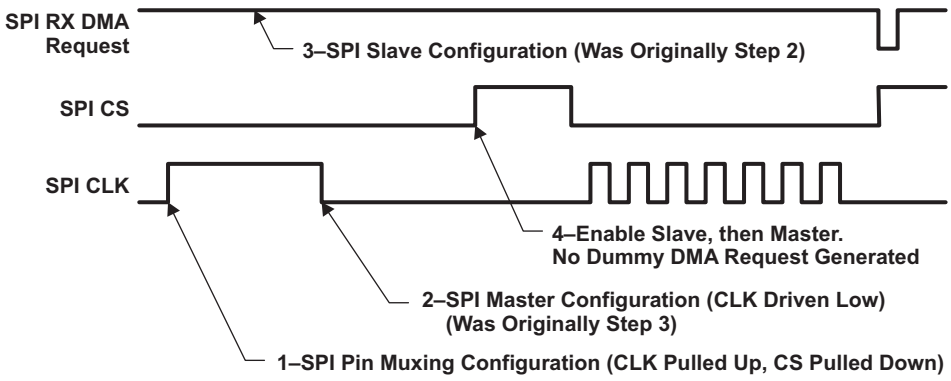


Figure 9. Dummy DMA RX Generation (Slave Mode After Clock Line Driven to Default Value)

Advisory 1.1.9 *PRM_CLKSRC_CTRL Registers Reset on Warm Reset*

Revision(s) Affected 1.1 and earlier

Details The PRM_CLKSRC_CTRL registers are reset on a Warm Reset; however, they should be reset on Cold Reset only. These parameters depend on the device environment only, but the registers must be re-programmed.

Workaround(s) If the default values are not used, the registers must be reprogrammed after warm reset is released.

Advisory 1.1.10 *L3 Interconnect Clock Divisor Default Value Must be Modified Before Configuration of SDRAM Controller*

Revision(s) Affected 1.1 and earlier

Details The SDRAM output clock is gated when an incorrect L3 CLK_SEL ratio is set. This operation is generally transparent and handled at boot time by the ROM code. The workaround should be implemented for the following scenarios:

- GP device External Fast boot is used.

Workaround(s) For External Fast Boot which are both impacted by the issue, CLKSEL_L3 (bits 1:0) = 10b must be set before performing SDRC configuration. In all other cases, this programing is handled by the ROM code and no workaround is necessary.

Advisory 1.1.11 *DMA: Drain_IE Reset Value*

Revision(s) Affected 1.1 and earlier

Details The Drain_IE bit in the DMA4_CICRi[12] register is not initialized after POR; its default value can be either '0' or '1' while the documentation specifies '0'.

Workaround(s) Prior to the DMA setup, the software must write 0x0 to this bit to disable the Drain_IE interrupt in the CICR register.

Advisory 1.1.12 *SDMA: Channel is Not Disabled After a Transaction Error*

Revision(s) Affected 1.1 and earlier

Details During a destination synchronized transfer on the write port (or source sync with SDMA.DMA4_CCRin[25] BUFFERING_DISABLE = 1), if a transaction error is reported at the last element of the transaction, the channel is not automatically disabled by the DMA.

Workaround(s) Whenever a transaction error is detected on the write side of the channel, the software must disable the channel by writing a '0' to DMA4_CCRi[7]: ENABLE bit.

Advisory 1.1.13 *McSPI Can Generate a Wrong Underflow Interrupt*

Revision(s) Affected 1.1 and earlier

Details

In the case where:

- A McSPI module is configured as master and is connected to another McSPI module configured as a slave (on the same chip, or on a different chip)
- The CS polarity is changed from the reset state (i.e., changed from CS inactive low to CS inactive high) on the master and slave sides
- The slave is enabled and then the master is enabled according to the programming guide

then the slave McSPI will generate a false underflow as soon as the first channel is enabled on the master McSPI side. This is because the master McSPI sets the right clock and chip select polarities only when the first channel is enabled. As the CS polarity is changed on the master side, this will generate a low-to-high transition on the CS signal. The slave McSPI will detect this transition and will try to load its shift register, which will result in an underflow interrupt being generated because there is no data to load.

If the slave is an external SPI device, then there is no issue. Only slave McSPI modules will be impacted.

If the CS polarity is not changed from its reset state, then there is no issue.

This issue will only occur when performing loopback tests on the same chip between two McSPI modules or when communicating through SPI between two McSPI modules on 2 different chips.

Workaround(s)

The following initialization sequence will solve this issue:

1. On the master side: Set MCSPI_MODULCTRL:SINGLE. Perform the following 3 steps by doing 3 different OCP accesses:
   - Configure channel I in MCSPI_CH(I)_CONF
   - Set MCSPI_CH(I)_CONF:FORCE
   - Reset MCSPI_CH(I)_CONF:FORCE
   - Reset MCSPI_MODULCTRL:SINGLE bit. The SPI bus polarity is now updated.
2. On the slave side : Configure channel 0 : write MCSPI_CH0_CONF Enable channel 0 : set MCSPI_CH0_CTRL:EN
3. On the master side : Enable channel I : set MCSPI_CH(I)_CTRL:EN

Advisory 1.1.14 *TV Detect AC Coupling Mode Not Supported*

Revision(s) Affected 1.1 and earlier

Details The TV detect in AC coupling mode is not implemented accurately and is not functional; therefore, TV detection in AC mode is impossible.

Workaround(s) Use DC coupling mode only.

Advisory 1.1.16 *I2C : I2C_STAT:XUDF is Not Functional in Slave Transmitter Mode*

Revision(s) Affected 1.1 and earlier

Details When configured in slave transmitter mode, the I2C_STAT:XUDF will not be set as expected if an underflow occurs. Only slave transmitter mode is impacted. Master transmitter mode is not impacted. The impact is rather low as the user can rely on I2C_STAT:XRDY interrupt status bit instead.

Workaround(s) Use the I2C_STAT:XRDY interrupt bit instead of XUDF when in slave transmitter mode.

Advisory 1.1.18 *EHCI controller- Issue in suspend resume protocol.*

Revision(s) Affected: 1.1 and earlier

Details: This issue concerns HSUSBHOST controller of AM35x and more specifically the EHCI controller (Only HS ports are impacted). The issue occurs after a Suspend when trying to Resume the bus (Host initiating the Resume) and also when doing RemoteWakeup (Device initiating the Resume).

The HSUSBHOST exits the resume/wakeup sequence without checking the USB bus LineState to ensure that it is in High Speed Idle state after having put the PHY in High Speed state. This error scenario will occur when there is a delay in PHY indicating the controller that the bus has switched to High Speed, and during that time the host is about to send an SOF packet.

Both TLL and PHY modes are impacted.

Workaround(s): The clear of the “ForceResume” bit in PORTSC register must be done at the beginning of the Port- SOF counter boundary, knowing that PortSOF counter is out of sync after a Suspend with GlobalSOF counter (PortSOF counter is stopped during Suspend while GlobalSOF counter continues counting). The FrameIndex counter is based on GlobalSOF counter (SOF counters are counting the 125us delay of a micro-frame). Only FrameIndex counter can be accessed by Software.

Two Workarounds can be listed for this issue (See flow diagrams and SW programming sequences below):

- Workaround 1:
  - Advantage: No limitation on the number of HS port usage.
  - Disadvantage: This workaround applies only to the to Suspend/Resume case and Suspend/RemoteWakeup case is not supported. This is because where within the PortSOF counter the HW will set the Resume bit is not known.
- Workaround 2:
  - Advantage: This workaround applies to both Suspend/Resume and Suspend/RemoteWakeup cases.
  - Disadvantage: Limitation to have only one HS port used (No limitation on the number of FS port usage as OHCI controller is not concerned). This is because EHCI controller has to be stopped by SW before clearing portSC.FPR.

Workaround 1: Any Write access to the “Port Suspend” and “Port Force Resume” bits in the PORTSC register must be done at the beginning of a micro-frame. This ensures that the Clear of “ForceResume” bit is done at the beginning of PortSOF counter.

Workaround 1 SW implementation:

- Suspend/Resume case SW implementation:
  1. Read FRINDEX register.
  2. Keep polling FRINDEX register to make sure that the register value has incremented from the value read in Step (1).
  3. Set the PORTSC.suspend bit.
  4. Wait for the required suspend time.
  5. When software is ready to issue resume, Read FRINDEX register.
  6. Keep polling FRINDEX register to make sure that the register value has incremented from the value read in Step (5).
  7. Set PORTSC.FPR bit.
  8. Wait for at least 20ms (as specified by the USB 2.0 Spec).
  9. Read FRINDEX register.
  10. Keep polling FRINDEX register to make sure that the register value has

incremented from the value read in Step (9).

11. Clear PORTSC.FPR bit.

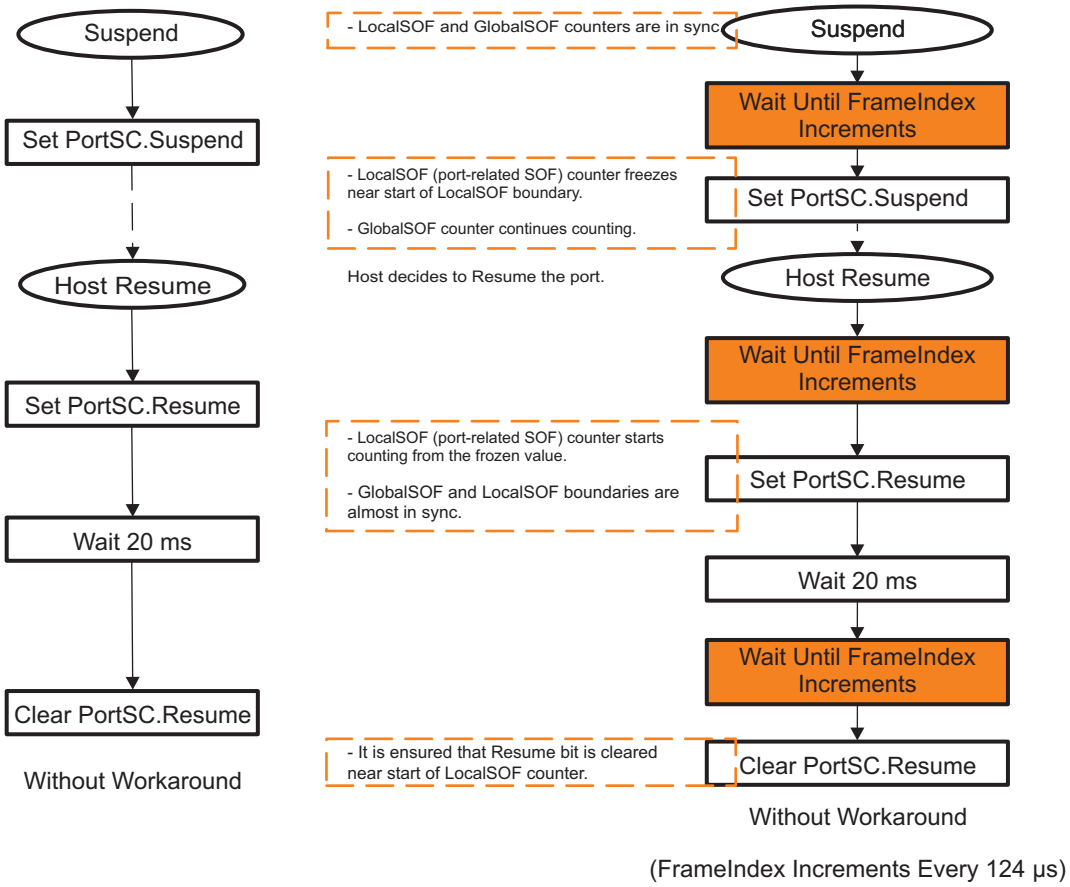


Figure 10. Workaround 1 Implementation Diagram

Workaround 2:

GlobalSOF and LocalSOF counters are in sync as soon as Run/Stop bit is set: Alignment on FrameIndex SOF is required only for the clear of Resume bit.

Workaround 2 SW implementation:

- Suspend/Resume case SW implementation:
  1. Set the PORTSC.suspend bit.
  2. Wait for the required suspend time.
  3. When software is ready to issue resume, set PORTSC.FPR bit.
  4. Wait for at least 20ms (as specified by the USB 2.0 Spec).
  5. Clear the USBCMD.RS bit to 0.
  6. Wait until USBSTS.HCH is set to 1 by HW.
  7. Clear PORTSC.FPR bit.
  8. Set the USBCMD.RS bit to 1.
- Suspend/RemoteWakeup case SW implementation:
  1. Set the PORTSC.suspend bit.
  2. Wait for the required suspend time.
  3. Wakeup event happens and software driver handles the wakeup interrupt.
  4. Wait for at least 20ms (as specified by the USB 2.0 Spec).

5. Clear the USBCMD.RS bit to 0.
6. Wait until USBSTS.HCH is set to 1 by HW
7. Clear PORTSC.FPR bit.
8. Set the USBCMD.RS bit to 1.

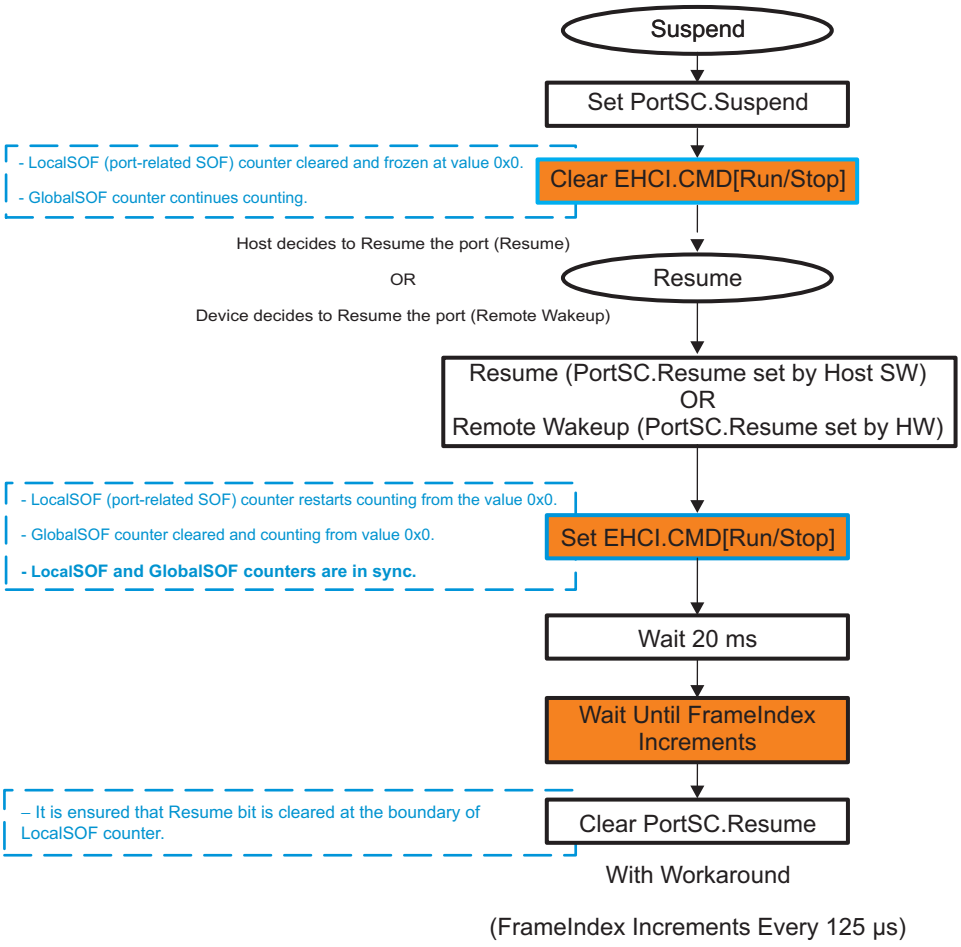


Figure 11. Workaround 2 Implementation Diagram

Advisory 1.1.20 *USBOTG Bytewise (8 bit/16 bit) Read Access Limitation*

Revision(s) Affected: 1.1 and earlier

Details: The USBOTG core supports transfers to and from the FIFOs in 8-bit, 16-bit, or 32-bit alignment as required, provided the data accessed is contiguous. The bytewise (8 bit/16 bit) read access is ***not*** supported at the IP Subsystem (IPSS) bridge which always posts a 32-bit read to the internal targets.

Note: Bytewise (8 bit/16 bit) writes are ***not*** affected.

Workaround(s): As a result of this read behavior, software must comprehend the following items to avoid functional issues when using the USBOTG:

1. When reading data from the Core Endpoint 0 FIFO, software ***must not*** use 8-bit or 16-bit read accesses. *Only* 32-bit read accesses are to be used as follows:
   - Read the COUNT0 register (number of bytes), add 3, then divide-by-4 to round up the number of valid bytes in the FIFO to the minimum number of 32-bit words (NumWords) that must be read to return all valid data bytes.
   - Use 32-bit reads to read NumWords from the Endpoint 0 FIFO.
   - Ignore any extra data reads (only COUNT0 bytes are valid).

   This needs to read from FIFO into 32-bit buffer, then do byte transfers from buffer to destination array to avoid unaligned word transfer processor exceptions.
2. Use CDMA for data I/O to all non-control endpoints. This eliminates the need for software to access the endpoint FIFOs per item 1., as the CDMA interface to the FIFO memory is unaffected by this bridge read behavior.
3. The following USBOTG interrupt-related registers have a read side affect where any active interrupt indicated by the register is cleared when the register is read:
   - INTRTX 16-bit register (offset 0x02)
   - INTRRX 16-bit register (offset 0x04)
   - INTRUSB 8-bit register (offset 0x0A)

   As a result, software should not read other registers (indicated by an "*" in the tables below) within the same 32-bit word address containing the registers listed in issue 3. above to avoid the read side affect (accidentally clearing an interrupt event) and potentially missing interrupt events.

Table 5. INTRTX REGISTER (16-Bit) EXAMPLE

32-BIT ALIGNED REGISTER ADDRESS	INTRTX (16-BIT REGISTER)	FADDR* (8-BIT REGISTER)	POWER* (8-BIT REGISTER)
0x00	0x02	0x1	0x0

Table 6. INTRRX REGISTER (16-Bit) EXAMPLE

32-BIT ALIGNED REGISTER ADDRESS	INTRTXE* (16-BIT REGISTER)	INTRRX (16-BIT REGISTER)
0x04	0x06	0x04

Table 7. INTRUSB REGISTER (8-Bit) EXAMPLE

32-BIT ALIGNED REGISTER ADDRESS	INTRUSBE* (8-BIT REGISTER)	INTRUSB (8-BIT REGISTER)	INTRRXE* (16-BIT REGISTER)
0x08	0xB	0xA	0x08

Typically, software will never need to read the FADDR, INTRTXE, INTRRXE, and INTRUSBE registers as it is software that configures these registers and thus, their values are already known. FADDR, INTRTXE, INTRRXE, and INTRUSBE registers are not read by software, so INTRRX and INTRUSB registers will not be unintentionally read.

The POWER register is queried by software to determine bus states and operating modes. The POWER register is typically only read at initialization time, so unintentional reads of INTRTX during USB transmits will not happen.

Advisory 1.1.21 *VPFE: CCDCFG.VDLC Must Always Be "1"*

Revision(s) Affected: 1.1 and earlier

Details: In the CCDC module, CCDCFG.VDLC must always be set to "1" due to the fact that there is no synchronization on the vd_rising_edge signal.

Workaround(s): None. CCDCFG.VDLC must always be one.

Advisory 1.1.22 *Warm Reset on EMIF4 Clears Register Configuration*

Revision(s) Affected: 1.1 and earlier

Details: During warm reset, the EMIF4 contents would be lost and software would need to reprogram the EMIF4 configuration registers prior to accessing the DDR memory (DDR2/mDDR) interface; therefore, self-refresh during a warm reset is ***not*** supported.

Workaround(s): After resuming from a warm reset, software would need to program the EMIF4 controller prior to accessing the DDR memory (DDR2/mDDR).

However during a cold reset, the following scenario is possible for the DDR memory (DDR2/mDDR):

- CKE pin to be pulled low during the power-on sequence
- AM35x device is "on" and data is put in DDR memory (DDR2/mDDR)
- DDR2 is put in self-refresh by pulling CKE pin low
- AM35x device power is removed – DDR memory (DDR2/mDDR) remains in self-refresh (via an external pulldown to VSS)
- AM35x device power is turned "on"
- DDR memory (DDR2/mDDR) remains in self-refresh because CKE pin stays low (driven or 3-state with pull), default state of pinmux is cke_safe, CKE pin stays in this reset state until the next step (CKE pin low or pulled low).
- SW configures the DDR memory (DDR2/mDDR) EMIF4 controller for self-refresh (and for proper mode/configuration); because CKE pin is forced low the memory will see none of this thus, ignoring all commands.
- The DDR memory (DDR2/mDDR) EMIF4 controller is now in sync with memory (and both are in self-refresh mode).
- SW configures the CKE pin to be controlled now directly by the EMIF4.
- SW wakes up memory via the controller.

Advisory 1.1.23 *CONTROL_REVISION Register Not Aligned With Silicon Revision*

Revision(s) Affected 1.1 and earlier

Details CONTROL_REVISION register contains the same value (0x00000010) for each silicon revision.

Workaround(s) Use CONTROL_IDCODE, which is upgraded for each silicon revision and documented in the *AM35x ARM Microprocessor Technical Reference Manual*.

Advisory 1.1.24 *Inactive State Management: Impossible to Transition to RETENTION States*

Revision(s) Affected 1.1 and earlier

Details If a power domain meets the conditions to go to an INACTIVE state (i.e., POWERSTATE is programmed to 0x3 (ON) and clock can be shut off), then the domain will transition to INACTIVE state. However, the domain cannot go to RET state automatically from INACTIVE state, even if software updates the POWERSTATE bit to 0x1 (RET) . The domain must be transitioned to the ON state before it can transition to the RET states.

Workaround(s) The following two conditions must be met:

1. Do not use autostate.
2. Perform wake-up event (software must force wakeup) to transition to an active state before transitioning to the RET states.

Advisory 1.1.25 *DSS 90 and 270-Degree Rotation DMA Optimization Does Not Function Properly*

Revision(s) Affected 1.1 and earlier

Details The DMA optimization functionality has been implemented in the display controller to reduce bandwidth. The access to the memory in 90- and 270- degree-rotation can be programmed to fetch two pixels per access. When this feature is used, the resampling (even if the ratio is 1) must be enabled to store the pixel in the lines buffer. This feature can be used with RGB16 and YUV422 pixels formats. Due to the address generation and the horizontal scaling issue, the DMA optimization does not work properly with RGB16 and YUV422 format for 90-degree and 270-degree rotations. The image is not correctly displayed on the panel.

Workaround(s) No workaround. The bit VIDDMAOPTIMIZATION (bit 20 of DISPC_VID1_ATTRIBUTES or DISPC_VID2_ATTRIBUTES register) must be left to its default value: 0x0.

Advisory 1.1.26 *I2C: Data Lost on Transmission from Memory to I2C Interface*

Revision(s) Affected 1.1 and earlier

Details The I2C is configured as master transmitter. After servicing a XRDY/XDR interrupt (FIFO empty), from the data sent on OCP, one, two or several bytes sent from the memory to the I2C interface are lost. The bytes lost are always the first transmitted on the OCP, when servicing the XRDY/XDR interrupts. The occurrence of the bug is related to the coincidence of the moment when data is sent on the OCP and the moment when the most significant bit of a byte is sent on the I2C, always when starting servicing the XRDY/XDR interrupt. Ideally, no data should be lost when transmitted from the OCP to the I2C. However, one, two or several bytes at the beginning of a transmission from the OCP to I2C are lost, if the moment when they are put on the OCP coincides with the transmission of the most significant bit of a byte on the I2C.

Workaround(s) A workaround exists for the interrupt mode of operation. Before servicing the XRDY/XDR interrupt, you must wait for the XUDF status bit to be set. For the data transmission using DMA, there is no available software workaround.

Advisory 1.1.27 *I2C: Wrong RDR Interrupt After Disabling the Module with I2C_EN*

Revision(s) Affected 1.1 and earlier

Details When the I2C_CON:I2C_EN bit is reset during the I2C module reconfiguration, some synchronization signals are not properly reset. This can generate a wrong RDR interrupt when the next transfer begins (before RRDY interrupt and stop condition).

Workaround(s) During the reconfiguration of the module, perform a software reset of the module (I2C_SYSC:SRST) instead of just reseting the I2C_CON:I2C_EN bit.

Advisory 1.1.28 *UART Not Asserting its TX DMA Request When RX FIFO is Not Empty*

Revision(s) Affected 1.1 and earlier

Details As long as the UART RX FIFO is not empty, the UART will not assert its TX DMA request. This means that in scenario using the DMA in both RX and TX, no UART transmission will occur until the RX fifo is emptied. This can cause deadlock situation if the software leaves some bytes in the RX FIFO.

Workaround(s) Software must make sure to always empty the RX FIFO when using the UART in full duplex mode.

Advisory 1.1.29 *HS USB: ECHI and OHCI Controllers Cannot Work Concurrently*

Revision(s) Affected 1.1 and earlier

Details An issue in the USBHOST memory access arbiter prevents EHCI and OHCI Host Controllers from working simultaneously. As a result one cannot connect a HS and a FS USB devices on the USBHOST.

Workaround(s) No workaround exists for the generic use-case. For low-throughput requirement a SW arbitration scheme can be implemented.

Advisory 1.1.31 *MPU Cannot Exit from Standby*

Revision(s) Affected 1.1 and earlier

Details MPU interrupt controller is not able to sort the input interrupt under idlereq pulse application. The sequence which creates this situation is: 1. When there is a pulse of idlereq (one or two clock cycles) applied after coming out of idle state. 2. There is change in input interrupt.

Impact: The input interrupt can not be sorted until the internal OCP clock starts running. Interrupt to CPU will be delayed till the OCP clock starts running. The OCP clock can start running if Another Idlereq pulse greater than two clock cycles.

Workaround(s) Disabling auto-gating (INTCPS_SYSCONFIG[0]:AUTOIDLE=0) feature which will allow the change in idlereq to be sampled. This can be done right before executing the idle instruction to avoid power consumption impact.

Advisory 1.1.32 *sDMA FIFO Draining Does Not Finish*

Revision(s) Affected 1.1 and earlier

Details There is an issue when sDMA channel is disabled on the fly, sDMA enters standby even through FIFO Drain is still in progress. SW WA is to put sDMA in NoStandby before a logical channel is disabled, then put it back to SmartStandby after the channel finishes FIFO draining. The issue only happens when FIFO draining is used and sDMA is configured SRC sync, BufferingEnabled and SmartStandby.

Workaround(s) Put sDMA in NoStandby before a logical channel is disabled, then put it back to SmartStandby right after the channel finishes FIFO draining. This issue can be avoided when one of the conditions (sDMA FIFO drain function enabling, SmartStandby, or On-the-fly channel disabling) is removed.

Advisory 1.1.33 *HSUSB Interoperability Issue with SMSC USB3320 PHY*

Revision(s) Affected 1.1 and earlier

Details After suspend sequence, USB3320 USB PHY goes correctly in low-power mode:

- DP Line goes High and DM line remains Low (J state)
- Rbias Voltage = 0 V
  Whereas OMAP HOST controller exit from suspend mode (while it is expected to keep in low power mode).
  OMAP Host state (exited from low power mode) is inconsistent with PHY state (low power mode) resulting in a lockup situation. Resuming the port has no effect as HOST controller has already exited from low-power mode.

Root cause: Delay in assertion of DIR causes USBHOST ULPI interface to exit ULPI Low Power mode. USB3320 USB PHY assert DIR signal 3 clock cycle after STP signal is de-asserted.

Workaround(s) There is no workaround.

Advisory 1.1.35 *USB Host EHCI may Stall When Exiting Smart-standby Mode*

Revision(s) Affected 1.1 and earlier

Details When the USBHOST module is set to smart-standby mode, and it is ready to enter the standby state (i.e., all ports are supported and all attached devices are in suspend mode), it may incorrectly assert the Mstandby signal too early while there are ongoing residual OCP transactions. If this condition occurs, the internal state machine may go to an undefined state and the USB link may be stuck upon the next resume.

Workaround(s) The software should explicitly disable (pause) the USB HOST OCP initiator activity by disabling the schedules (USBCMD[5]ASE = 0, USBCMD[4]PSE=0) just before suspending the connected ports and restore their state after the USBHOST has entered smart-standby state.

Software workaround sequence:

- Read USBCMD register and save it;
- Clear USBCMD[5]ASE and USBCMD[4]PSE bits;
- Wait for the USBSTS[15]ASS and USBSTS[14]PSS bits to reflect this change;
- Suspend the connected ports;
- Wait for the ports suspend to take effect (~3ms);
- Restore the USBCMD register;

Advisory 1.1.36 *USB Host EHCI May Stall when Running High Peak-bandwidth Demanding UseCases*

Revision(s) Affected 1.1 and earlier

Details The USB host module is AHB native. Therefore there is an AHB2OCP bridge allowing to connect it to the OCP L3 interconnect. Both AHB and OCP masters are able to generate single accesses (R/W) as well as busrts, depending on the configuration,as well as address ranges. Under some specific L3 latency conditions, when a USB host write is followed by a USB host single read (not burst read), then the read can be lost in the AHB/OCP bridge. When this happens, the internal state machines of the module go into an undefined state and the EHCI stalls; ongoing transfers are stopped, and new transfers cannot be scheduled anymore. This situation will only occur when both following conditions happen simultaneously:

- The module is performing a write followed by a single AHB read.
  - This can happen when processing control messages (Transfer descriptions in memory are updated (written) by the host when being processed and an 8 bytes command is fetched by the host (2 single AHB reads))
  - This can also happen for any OUT transfer (bulk, isochronous, interrupt) depending on data payload size and maximum Tx packet size parameter (TxMaxP)
- Congestion peaks occur in the system, generating back pressure at the host boundary with the interconnect
  - This can typically happen when high priority initiators like Display Subsystem and/or Camera are runnung heavy use cases in parallel of USB transfers. This issue does not impact IN transfers.

Workaround(s) Define ((payload size) modulo (MaxP size)) >=16 bytes. If ((payload size) modulo (MaxP size)) < 16 bytes, then dummy data can be added to the buffer in order to achieve ((payload size)modulo (MaxP size)) >= 16 bytes. However this is not always possible, typically for control transfers, for which payload size is fixed to 8 bytes.

In this case, it is only possible to reduce the failure occurence by:

- removing un-necessary control commands (like get_device_state upon suspend exit)
- avoiding enumeration during peak-bandwidth demanding use cases Once the issue has occurred, the only way to recover will be to reset the USB host module and re-enumerate.

Advisory 1.1.39 *DSS RGB16 Wrong Image Generated with 5-Tap Resizer on Odd HPPL*

Revision(s) Affected 1.1 and earlier

Details When RGB16 image with odd number of Horizontal Pixels per Line (HPPL) is used for image resizing with 5-Tap configuration, the output image is shifted by 1 Pixel to left line by line. This issue is specific to below condition:

1. 5-tap vertical resize configuration.
2. Input image format is RGB16.
3. Horizontal width of the original image size is odd pixels.

If one of the above conditions change to below, the issue do not appear and will get correct output image.

1. Change 5-tap configuration to 3-tap configuration.
2. Original image size being Even HPPL.
3. Input image format changed from RGB16 to RGB24 unpacked.

Workaround(s) With RGB16 image format use image size with even number of Horizontal Pixels per Line (HPPL) for 5-tap resizer configuration.

Advisory 1.1.40 *Limitation With Single DMA Read Access in USBHOST*

Revision(s) Affected 1.1 and earlier

Details There is a bug in the AHB2OCP Bridge of USBHOST which causes the single DMA read access that is preceded by a DMA write transaction to be dropped.

Workaround(s) Limit single access read.

Advisory 1.1.41 *DPLL3 Recall and Long Relock Time*

Revision(s) Affected 1.1 and earlier

Details Temperature drift is impacting DPLL3 relock time. High temperature drift (positive or negative delta t > 20C) can extend relock time from 40 up to 150 clock cycles.

This could cause issue in low power scenarios when DPLL3 is set in idle most of the time and needs a fast wakeup to meet application timing.

Workaround(s) The issue is fixed by disabling DPLL3 automatic control (bypass or stop mode using CM_AUTOIDLE_PLL[2:0] register bits). Automatic control should be enabled again before doing a transition to retention or off modes.

Advisory 1.1.42 *Incorrect FSR Capture for McBSP1 When Receiver Configured as Master*

Revision(s) Affected 1.1 and earlier

Details The McBSP1 FSR signal could be captured on the same clock edge as it was generated resulting in a right bit shift of the received data. This is possible when McBSP is configured as follows:

- 6 pin
- Receiver Master
- Full Cycle (default)
- VDDSHV = 3.3V

Workaround(s) If the receiver must be configured as master set the receiver to Half Cycle Mode. Half Cycle Mode is selected by setting McBSP1.MCBSPLP_RCR_REG[11] = 0x0.

Advisory 1.1.43 *DDR2 External Strobe Enable*

Revision(s) Affected 1.1 and earlier

Details When implementing AM35x with DDR2, glitches on external strobe gating signal (STRBENx and STRBEN_DLYx) can cause a misalignment in the DDR PHY FIFO pointers, resulting in erroneous reads from DDR2. The glitches are caused by package coupling from signals bonded out close to STRBEN, and can be more pronounced during high levels of switching on DDR data signals. Series termination resistors can suppress this noise, but is sometimes not practical on small form factor boards. This errata does not apply when using mDDR.

Workaround(s) When using DDR2, internal strobe gating should be used in most cases by setting DDR_PHY_CTRL_1.CONFIG_EXT_STRBEN to 0. Also, memory buffer drive should be set to full drive strength (set SDRAM_CONFIG.REG_SDRAM_DRIVE to 0).

Advisory 1.1.44 *mDDR Internal Strobe Enable*

Revision(s) Affected 1.1 and earlier

Details When implementing AM35x with mDDR, internal strobe gating (DDR_PHY_CTRL_1.CONFIG_EXT_STRBEN = 0) does not work for all worst case silicon conditions. This errata does not apply when using DDR2.

Workaround(s) When using mDDR, set DDR_PHY_CTRL_1.CONFIG_EXT_STRBEN to 1 to enable external strobe gating. Ensure board routing recommendations in the AM35x Data Manual are followed when routing strobe enable signals STRBENx and STRBEN_DLYx.

Advisory 1.1.45 *V_{IL} Specification Relaxation for AC timings*

Revision(s) Affected 1.1 and earlier

Details As a result of some timing degradations for the 3.3V and 1.8V I/Os at weak process and high temperature, the V_{IL} specification for AC I/O timings are changing to 0.4V.

Workaround(s) Ensure that V_{IL} for companion devices meet this specification.

Advisory 1.1.46 *Cortex-A8 Errata List*

Revision(s) Affected 1.1 and earlier

Details This errata document does not cover the advisories associated with the ARM Cortex-A8 processor. For a list of the advisories associated with each version of the ARM Cortex-A8 processor, contact your TI representative for a copy of the *ARM Core Cortex-A8 (AT400/AT401) Errata Notice*. See Table 2 to determine which version of the ARM Cortex-A8 processor is included in each device silicon revision.

Advisory 1.1.47 *SGX Bugs Documented in Imagination Technologies™ Errata*

Revision(s) Affected 1.1 and earlier

Details SGX is considered as a black box for most users; therefore, no detailed information about SGX bugs is provided in this document. SGX bugs are detailed in errata supported by Imagination Technologies™, which is available only for API/code developers.

The reference of the issues impacting SGX instances are: To be updated further to imagination review.

The SGX errata document is not delivered by TI but Imagination technologies.

Workaround(s) There is no workaround for this issue.

Advisory 1.1.48 *I2C1 to 3 SCL Low Period Is Shorter in FS Mode*

Revision(s) Affected: 1.1 and earlier

Details: Due to IO cell influence, I2C1 to 3 SCL low period can be shorter than expected. As a result, I2C AC timing (SCL minimum low period) in FS mode may not meet the timing configured by software.

Workaround(s): I2C1 to 3, SCL low period is programmable and proper adjustments to the SCLL/HSSCLL values can avoid the issue.

Advisory 1.1.49 *I2C: In SCCB Mode, Under Specific Conditions, the Module Might Hold the Bus by Keeping SCL Low*

Revision(s) Affected 1.1 and earlier

Details In SCCB mode, if the XRDY/RRDY are not served during the address phase, the module starts to hold the bus by keeping SCL low (FIFO empty or full). Then, after serving these interrupts, the module does not continue the current transfer and blocks in this state.

This bug appears only in applications where the module is used in SCCB mode. The bug is not appearing at all if interrupts are served before the address phase starts (quickly enough to avoid entering this context).

Workaround(s) There is no workaround for this issue.

Advisory 1.1.50 *I2C: Wrong Behavior When Data With MSB 0 Is Put in the FIFO Before a Transfer With SBLOCK Is Started*

Revision(s) Affected 1.1 and earlier

Details Expected behavior: Before starting a new transfer from another I2C device, one byte of data is written to TX FIFO. The module is addressed on a 10bit address as a Slave transmitter (one of his addresses) and I2C clock blocking is enabled. After, repeated start condition SBLOCK is activated again for the second part of the address.

Observed behavior: Given the addressing and SBLOCK conditions defined above, if the data put in FIFO has its MSB 0, the module makes a glitch on SDA bus on the eighth bit (SDA is to '0' for a short period) that can be interpreted as an illegal start/stop condition.

Workaround(s) The scenario described is a corner case, it may very seldom happen in applications. To avoid the situation, before a transfer is started on the I2C bus, all interrupts should have been cleared (part of the guideline given in the spec), or when I2C is a transmitter, no data should be placed in the FIFO, without receiving the request to do so from the slave.

Advisory 1.1.51 *I2C: After an Arbitration is Lost the Module Incorrectly Starts the Next Transfer*

Revision(s) Affected 1.1 and earlier

Details Expected behavior: After a transfer on the I2C bus, where the module lost the arbitration during address phase, a new transfer as a Master is programmed in I2C_CON by setting MST bit to '1', having the start bit STT in the I2C_CON register still unset. The STT bit can be set after a significant delay, to point the moment in which the transfer starts on the I2C bus. The module should only start the transfer on I2C after setting this STT start bit in I2C_CON.

Observed behavior: The module starts the transfer on I2C before setting STT, immediately after setting MST bit in the I2C_CON to '1'.

Workaround(s) The MST and STT bits inside I2C_CON should be set to '1' at the same moment (avoid setting the MST to '1' while STT = '0').

Advisory 1.1.52 *USB DEVICE Aborts Remote Wake-Up Sequence When Device Wakes From OFF/RET to ON in USB TLL Mode*

Revision(s) Affected 1.1 and earlier

Details When the device is programmed to go to OFF/RET mode, the power, reset, and clock management module (PRCM) powers down the High-speed USB Host Subsystem. In response, the High-speed USB Host Controller suspends the bus and the USBTLL issues a suspend interrupt.

Before USB Host Controller is switched OFF its register contents are automatically saved to voltage domain retention memory. Also, before the CORE voltage domain is switched to OFF/RET, the USBTLL contents are saved. Once these registers are saved, the device transitions to the OFF/RET state.

When an external device initiates a remote wakeup, the PRCM wakes the device. The CORE domain reset is released and the USBTLL registers are restored. Upon the completion of the USBTLL register restore an ALT interrupt is erroneously generated by the USBTLL to the external device. This breaks the USB remote wakeup protocol and as a result the external device aborts the remote wakeup sequence.

Workaround(s) To workaround this issue the IdGnd fall interrupts should be disabled by the external device at boot-up. This interrupt can be disabled by clearing the IDGND_FALL bit of the ULPI_USB_INT_EN_FALL_i registers.

Advisory 1.1.53 *MMC OCP Clock Not Gated When Thermal Sensor Is Used*

Revision(s) Affected: 1.1 and earlier

Details: Once enabled (CM_FCLKEN3_CORE.EN_TS = 0x1,), the PRCM provide the 32Khz clock to the clock tree made of Thermal sensor+ MMC1/2/3.

As soon as 32Khz clock is provided to MMC module then debouncing operation is started. Once debouncing is completed, then interface clock is ungated at module level and OCP clock is enabled internally to the MMC module (while MMC module is not used). This is creating unexpected over-consumption (100uA/MMC instance measured). The auto gating stays inefficient as long as module is not enabled.

Workaround(s): Each time thermal sensor is used, the MMC instances which are not used should be enabled then disabled (write CM_ICLKEN1_CORE.EN_MMCn = 0x1, CM_FCLKEN1_CORE.EN_MMCn = 0x1, wait until CM_IDLEST1_CORE.ST_MMCn = 0x0 then write CM_ICLKEN1_CORE.EN_MMCn = 0x0, CM_FCLKEN1_CORE.EN_MMCn = 0x0) to avoid over-consumption due to OCP logic being un-necessarily clocked.

Advisory 1.1.54 *USB 2.0 On-The_Go (OTG) Session Request Protocol (SRP) is Not Supported*

Revision(s) Affected: 1.1 and earlier

Details: The USB 2.0 On-The-Go (OTG) Session Request Protocol (SRP) allows a USB-peripheral to request the USB-host to enable Vbus and start a session. On this device, the SRP protocol is not supported.

The OTG Host Negotiation Protocol (HNP), which allows USB-devices to swap roles between host and peripheral, is supported.

Workaround(s): None.

Advisory 1.1.55 *EMAC : Transmit Statistics Registers Do Not Increment While Connected in Half-Duplex Mode*

Revision(s) Affected: 1.1 and earlier

Details: When connected in half-duplex mode, the relevant transmit registers in the Network Statistics Registers do not correctly increment when data is transmitted by the EMAC. Full-duplex modes are unaffected by this errata and receive registers increment as expected in both full and half-duplex modes. This issue is limited to the Network Statistics Register since the network interface itself is unaffected and all network traffic is processed normally.

Workaround(s): None.

3 Silicon Revision 1.0 Usage Notes and Known Design Exceptions to Functional Specifications

3.1 Usage Notes for Silicon Revision 1.0

Silicon revision 1.0 applicable Usage Notes have been found on a later silicon revision. For more details, see Section 2.1.

Note: The peripherals supported on the various AM35x microprocessors are different. The user should only refer to usage notes and advisories pertaining to features supported on the specific device. For a complete list of the supported features of the AM35x microprocessors, see the device-specific data manual.

3.2 Silicon Revision 1.0 Known Design Exceptions to Functional Specifications

Some silicon revision 1.0 applicable advisories have been found on a later silicon revision. For more details, see Section 2.2.

Table 8. Silicon Revision 1.0 Advisory List

Advisory 1.0.34 *PowerVR SGX™: MMU Lockup on Multiple Page Miss*

Revision(s) Affected 1.0

Details The PowerVR SGX Bus interface contains an MMU address translation function which works on 4KB page allocations. The page table entries are setup and located in external DRAM memory and are fetched on demand as requests are made, there is a cache within the PowerVR SGX which keeps the most recently used entries. When an internal requester makes a request in virtual space the address is tested with the cached entries.

If there is a hit then the physical high order address bits are returned and combined with the lower address bits of the virtual address bits to form the external physical address access. If there is a cache miss then the MMU must make an external fetch to memory to update the on chip cache, when this memory fetch completes the original memory access can proceed. The PowerVR SGX has 7 parallel memory request sources that feed into the MMU translation logic, so at any one time there can be a maximum of 7 parallel request sources that can all exhibit a page miss at the same time. The MMU contains a 3 bit counter to keep track of cache-miss requests (7 requestors). However there are scenarios where the request sources generate multiple cache miss-requests causing the 3 bit counter to overflow and miss service requests. Ultimately multiple passes through normal and miss phases together with various contributing memory latency can result in the counter not returning to zero and the MMU will lockup in the MMU miss phase. This will block all further accesses and the core will lock up. This could result in lock up of the PowerVR SGX image processing pipeline and may result in system hang.

Workaround(s) The experiments have shown that if the core is forced to an idle state (in terms of TA and 3D processing) before a cache invalidate is issued, then the occurrence of the lockup is substantially reduced.

Advisory 1.0.38 *GPMC Has Incorrect ECC Computation for 4-bit BCH Mode*

Revision(s) Affected 1.0

Details The GPMC supports 4- or 8-bit error detection BCH code. 4-bit error mode uses a wrong polynomial, and as a result for this mode the GPMC will:

- On page write, generate incorrect ECC parity
- On page read, generate an incorrect syndrome.

This bug causes an incorrect error location.

Workaround(s) No workaround.

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from D Revision (June 2012) to E Revision	Page
• Deleted Advisory 1.1.19. This advisory did not apply to this device	32

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license 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 significant portions 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. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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