DM3730, DM3725 Digital Media Processors Silicon Revisions 1.2, 1.1, 1.0

Silicon Errata

Literature Number: SPRZ319F

August 2010–Revised April 2014

Contents

List of Figures

List of Tables

Silicon Errata

SPRZ319F–August 2010–Revised April 2014

DM3730, DM3725 Digital Media Processors (Silicon Revisions 1.2, 1.1, 1.0)

1 Introduction

This document describes the known exceptions to the functional specifications for the DM3730/25 digital media processors. [See the *DM3730/25 Digital Media Processor Data Manual* (literature number SPRS685).]

For additional information, see the latest version of the *AM/DM37x Multimedia Device Technical Reference Manual* (literature number SPRUGN4).

The advisory numbers in the document are not sequential. Some advisory numbers are not applicable.

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 DM3730/25 Device and Development Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all microprocessors and support tools. Each commercial 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."

Production 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.

Sitara is a trademark of Texas Instruments.
Cortex is a trademark of ARM Limited.
ARM is a registered trademark of ARM Ltd.
1-Wire is a registered trademark of Maxim Integrated Products, Inc..
All other trademarks are the property of their respective owners.

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 DM3730/25 Digital Media Processor device markings.

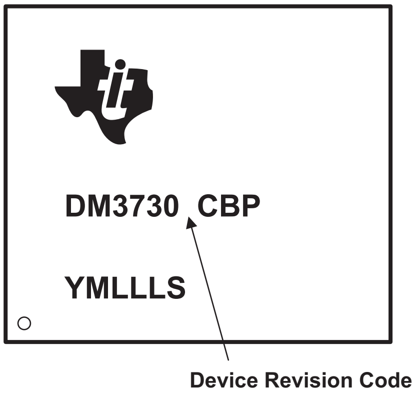


Figure 1. Example, Device Revision Codes for DM3730/25 Digital Media Processor

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) DM3730 is the device part number.

(C) CBP is the package designator.

(D) YM denotes year/month.

(E) LLLL denotes Lot Trace Code.

(F) S denotes Assembly Site Code.

(G) On some "X" 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 DM3730xCBP, where "x" denotes the silicon revision. Table 1 and Table 2 list the information associated with each silicon revision for each device type. For more details on device nomenclature, see the device-specific data manual.

Table 1. DM37x Production Device Revision Codes

DEVICE REVISION CODE (x)	SILICON REVISION	COMMENTS
(blank)	1.2	Silicon revision is new

Table 2. XDM37x Experimental Device Revision Codes

DEVICE REVISION CODE (x)	SILICON REVISION	COMMENTS
B	1.2	Silicon revision is new (also referred to as ES1.2)
A	1.1	Silicon revision 1.1 (also referred to as ES1.1)
(blank)	1.0	Silicon revision 1.0 (also referred to as ES1.0)

Each silicon revision uses a specific revision of Sitara™ ARM® Cortex™-A8 processor. The Sitara ARM Cortex-A8 processor 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 18, minor number counts ROM code version. The ROM code version is coded as hexadecimal readable values, e.g. ROM version 18.07 will be coded as 0000 1807h. Table 3 shows the ROM code revision for each silicon revision of the device

Table 3. Silicon Revision Variables

SILICON REVISION	Sitara ARM CORTEX-A8 VARIANT/REVISION	ROM REVISION
1.2	r3p2	18.07
1.1	r3p2	18.07
1.0	r3p2	18.07

2 All Errata Listed With Silicon Revision Numbers

Table 4. All Usage Notes (Limitations)

Type/Number	Title	1.0	1.1	1.2
Usage Note 2.1	Performance Limitation On LCD Read/Write Access Through RFBI L4 Port	X	X	X
Usage Note 2.2	IVA2: Under Hardware Emulation, IDLE Instruction Must Be Executed From L1P or L2 SRAM	X	X	X
Usage Note 2.3	Domain Woken Up by Wake-Up Dependency Cannot Transition to Inactive State	X	X	X
Usage Note 2.4	VENC: Last Data Line Missing in PAL	X	X	X
Usage Note 2.5	Observability Signals Not Functional in OFF Mode	X	X	X
Usage Note 2.6	Constraints on Module Clocks When Using DVFS	X	X	X
Usage Note 2.7	UART: Cannot Acknowledge Idle Requests in Smartidle Mode When Configured for DMA Operations	X	X	X
Usage Note 2.9	GPIO is Driving Random Values When Device Returns From OFF Mode	X	X	X
Usage Note 2.10	Maximum 12 Bits Output (1 bit sign + 11 bits of value) Is Supported on CAVLD of iVLCD	X	X	X
Usage Note 2.11	Extra Power Consumed When Repeated Start Operation Mode Is Enabled on I2C Interface Dedicated for Smart Reflex (I2C4)	X	X	X
Usage Note 2.12	SGX 192 MHz Limitation With TVOUT @ 54 MHz	X	X	X
Usage Note 2.13	PWDNx Bit in CLock Manager Is Reseting the DPLL4 Mx HSDIVIDER Coefficient Previously Programmed	X	X	X
Usage Note 2.15	MMC IOs Reliability Issue	X	X	
Usage Note 2.16	Downscaling Limitations	X	X	X
Usage Note 2.17	TVOUT Line Shift Issue When Using DSI PLL for DISPC_FCLK	X	X	X
Usage Note 2.18	HDQ™/1-Wire® Communication Constraints	X	X	X
Usage Note 3.4	Avoiding Voltage Drop When Booting at 1 GHz OPP With TPS65950A3	X	X	X
Usage Note 3.5	Undesired McBSP slave mode behavior during reset without CLKR/CLKX	X	X	X
Usage Note 4.1	HS USB Host Subsystem: Some Limitations Exist When Connecting to External Devices	X	X	X
Usage Note 4.2	"Speed Binned" Bit in Control Device Status Register Not Programmed in Devices Built and Tested Prior to June 2011	X	X	X

Table 5. All Design Exceptions to Functional Specifications (Bugs)

Type/Number	Title	1.0	1.1	1.2
Advisory 1.2	I2C Module Does Not Allow 0-Byte Data Requests	X	X	X
Advisory 1.3	Delay Required to Read Some GP, WD, and Sync Timer Registers After Wake-Up	X	X	X
Advisory 1.4	IVA2: C64x+ L1D Cache May Lose Data or Hang DMA Operations Under Certain Conditions	X	X	X
Advisory 1.5	MDR1 Access Can Freeze UART Module When in IrDa Mode	X	X	X
Advisory 1.6	IVA2: Block Cache Operations Word Count (*WC) Must be Less Than or Equal to 0xFF80	X	X	X
Advisory 1.7	I2C: RDR Flag May Be Incorrectly Set	X	X	X
Advisory 1.8	IVA2: EDMA Channel Priority Is Not Correctly Enforced	X	X	X
Advisory 1.9	Inactive State Management: Impossible to Transition to OFF or RETENTION States	X	X	X
Advisory 1.10	Cortex-A8 Errata List	X	X	X
Advisory 1.11	Inappropriate Warm Reset Generation on Smart Reflex I2C Error	X	X	X
Advisory 1.12	IVA2: Back-to-Back SPLOOPS With Interrupts Can Cause Incorrect Operation	X	X	X
Advisory 1.13	IVA2: DSP Generates False Internal Exception for Multiple Writes	X	X	X
Advisory 1.14	GPMC May Stall After 256 Write Accesses in NAND_DATA, NAND_COMMAND, or NAND_ADDRESS Registers	X	X	X
Advisory 1.15	SPI Dummy DMA RX Request Generation	X	X	X
Advisory 1.17	IVA2: ISP/GFX Dependencies	X	X	X
Advisory 1.20	OCP Error Does Not Get Communicated to USBOTG	X	X	X
Advisory 1.21	L3 Interconnect Clock Divisor Default Value Must be Modified Before Configuration of SDRAM Controller	X	X	X
Advisory 1.28	DMA: Drain_IE Reset Value	X	X	X
Advisory 1.29	sDMA: Channel Is Not Disabled After a Transaction Error	X	X	X
Advisory 1.30	SGX Bugs Documented in Imagination Technologies™ Errata	X	X	X
Advisory 1.31	HS USB HOST Controller: Device Aborts the Remote Wake-Up Sequence if AM3715/03 Wakes From OFF/RET to ON in USB TLL Mode	X	X	X
Advisory 1.32	Pending Interrupt to Video Sequencer Prevents IVA2 from Going Into Idle Mode	X	X	X
Advisory 1.33	McSPI Can Generate a Wrong Underflow Interrupt	X	X	X
Advisory 1.34	VENC: TV Detect AC Coupling Mode Not Supported	X	X	X
Advisory 1.35	Unexpected Stalling May Occur During SDMA/IDMA Accesses to DSP L2 Memory	X	X	X
Advisory 1.38	HS USB OTG Failing Access to Registers	X	X	X
Advisory 1.39	HS USB OTG Software Reset Is Not Fully Functional	X	X	X
Advisory 1.40	ROM Code: CKE PAD Is Not Set When Initializing External RAM	X	X	X
Advisory 1.41	ROM Code: SDRC_POWER Register Is Initialized With Hardcoded Value	X	X	X
Advisory 1.42	ROM Code: When sys-clock = 16.8MHz, Then Peripheral Booting via UART3 Is Not Operational	X	X	X
Advisory 1.43	I2C: I2C_STAT:XUDF Is Not Functional in Slave Transmitter Mode	X	X	X
Advisory 1.44	EHCI Controller: Issue in Suspend Resume Protocol	X	X	X
Advisory 1.45	GPIO Pad Spurious Transition (Glitch/Spike) During Wake Up Entering or Exiting System OFF Mode	X	X	X
Advisory 1.46	Voltage Processor TRANXDONE Interrupt Occurs Too Early	X	X	X
Advisory 1.47	HS USB OTG: OTG_SYSCONFIG:AUTOIDLE Bit Is Not Reset Correctly	X	X	X

Table 5. All Design Exceptions to Functional Specifications (Bugs) (continued)

Type/Number	Title	1.0	1.1	1.2
Advisory 1.48	IVA2 Does Not Wake-Up After It Goes to IDLE While DMA Request Is Still Asserted	X	X	X
Advisory 1.49	Software Reset Done While ISP Processing Is Ongoing Can Cause OCP Protocol Violations	X	X	X
Advisory 1.51	Accesses to DDR Stall in SDRC After a Warm-Reset	X	X	X
Advisory 1.52	Standard OTG Compliance Electrical Tests for HOST Mode Will Fail	X	X	X
Advisory 1.54	GPMC Has Incorrect ECC Computation for 4-Bit BCH Mode	X		
Advisory 1.56	HS USB Host: ECHI and OHCI Controllers Cannot Work Concurrently	X		
Advisory 1.57	MMC OCP Clock Not Gated When Thermal Sensor Is Used	X	X	X
Advisory 1.58	Context Save Operation Randomly Failing for CONTROL_PAD_CONF_ETK_D14	X	X	X
Advisory 1.59	I2C4 Does Not Meet I2C Standard AC Timing in FS Mode	X	X	X
Advisory 1.60	I2C1 to 3 SCL Low Period Is Shorter in FS Mode	X	X	X
Advisory 1.61	Missed Dependency With McBSP External Clock Prevents Transition to OSWR	X	X	X
Advisory 1.62	MPU Cannot Exit from Standby	X	X	X
Advisory 1.63	sDMA FIFO Draining Does Not Finish	X	X	X
Advisory 1.64	HSUSB Interoperability Issue With SMSC USB3320 PHY	X		
Advisory 1.65	IVA2 Does Not Wake-Up After It Goes to IDLE While an Interrupt Line Is Not Cleared	X	X	X
Advisory 1.66	IVA2 SEQ (ARM9) JTAG Failures With a Free Running Clock Emulator	X		
Advisory 1.67	Isolation Issue in OFF Mode Impacting the CORE Power Domain	X		
Advisory 1.68	Safe Mode Does Not Work on GPMC_A11 Pad	X		
Advisory 1.69	GPMC_A11 Address Bit Does Not Behave as Expected (256MB NOR Flash Addressability)	X		
Advisory 1.71	DM37x Device May Wake Up From Low Power Modes if IO Wake-Up and Pull Down Are Enabled	X		
Advisory 1.72	Excessive Leakage on BRGC Memories in Cortex A8 When Periphery Is Turned Off	X		
Advisory 1.73	JTAG Pins With bq18jtagfbpbnopm_ssbhv IO Have ESD (Electro Static Discharge) Weakness in Input Buffer	X		
Advisory 1.74	Scan Shift Issues in GWG Memories	X		
Advisory 1.75	DMA4 Channel Fails to Continue With Descriptor Load When Pause Bit Is Cleared Through Config Port Access While in Standby	X	X	X
Advisory 1.76	USB OTG DMA Status Register Read Return Zero	X	X	X
Advisory 1.77	POWERVR SGX™: MMU Lockup on Multiple Page Miss	X		
Advisory 1.78	Device Cannot Reboot if Warm Reset Occurs During MPU DPLL Lock Sequence	X		
Advisory 1.79	USB Host EHCI May Stall When Exiting Smart-Standby Mode	X	X	
Advisory 1.80	USB Host EHCI May Stall When Running High Peak-Bandwidth Demanding Use Cases	X	X	
Advisory 1.81	USB OTG DMA May Stall Under Specific Configuration	X	X	X
Advisory 1.82	USB OTG DMA May Stall When Entering Standby Mode	X	X	X
Advisory 1.84	DPLL3 in Manual Lock Mode Cannot Be Used When CORE Goes to OSWR or OFF State	X	X	X
Advisory 1.85	PRCM DPLL Control FSM Removes SDRC_IDLEREQ Before DPLL3 Locks	X	X	X
Advisory 1.86	PER Domain Reset Issue After Domain-OFF/OSWR Wake-Up	X	X	

Table 5. All Design Exceptions to Functional Specifications (Bugs) (continued)

Type/Number	Title	1.0	1.1	1.2
Advisory 1.87	Self-Refresh Exit Issue After OFF Mode	X	X	
Advisory 1.88	Programming of tCKE	X	X	X
Advisory 1.89	McBSP Used in Slave Mode Can Create a Dead Lock Situation When Doing Power Management	X	X	X
Advisory 1.90	DPLL3 Bypass Condition Does Not Consider State of SGX FCLK	X	X	X
Advisory 1.91	Top Ball Compliancy With LPDDR JEDEC Standard	X	X	X
Advisory 1.92	I2C: In SCCB Mode, Under Specific Conditions, the Module Might Hold the Bus by Keeping SCL Low	X	X	X
Advisory 1.93	I2C: Wrong Behavior When Data With MSB 0 Is Put in the FIFO Before a Transfer With SBLOCK Is Started	X	X	X
Advisory 1.94	I2C: After an Arbitration is Lost the Module Incorrectly Starts the Next Transfer	X	X	X
Advisory 1.95	One Timing Does Not Follow the JEDEC Standard	X	X	X
Advisory 1.96	4-cycle Saturating .M Unit Instructions May Mask 2-cycle Saturating .M Unit	X	X	X
Advisory 1.97	SPLOOP CPU Cross-Path Stall	X	X	X
Advisory 1.98	RTA Feature is Not Supported	X	X	X
Advisory 1.99	HS USB OTG Spurious IRQ When Switching From DMA Mode 1 to DMA Mode 0 or to CPU Mode for Unloading the Last Short Packet of a Transaction	X	X	X
Advisory 1.100	Write to OHCI Registers May Not Complete	X	X	X
Advisory 1.101	Violation of Vix Crossing Specification	X	X	
Advisory 1.102	HS USB OTG Bus Reset	X	X	X
Advisory 1.104	McBSP2 Data Corruption	X	X	X
Advisory 1.105	DSI-PLL Power Command 0x3 Not Working	X	X	X
Advisory 1.106	MPU Leaves MSTANDBY State Before IDLEREQ of Interrupt Controller is Released	X	X	X
Advisory 1.107	Some Power Domains Cannot Go To OFF Mode After Warm Reset	X	X	X
Advisory 1.108	USBHOST Configured In Smart-Idle Can Lead To a Deadlock	X	X	X
Advisory 1.109	USB OTG Software Initiated ULPI Accesses To PHY Registers Can Halt the Bus	X	X	X
Advisory 1.110	At Context Restore, a Certain PER DPLL Programming Can Introduce Wake-up Latencies	X	X	X
Advisory 1.111	USB Peripheral Booting is Not Operational When an MMC Booting has Been Attempted Before	X	X	X
Advisory 1.112	DSS: Color Phase Rotation (CPR) Breaks 2 Pixel Wide Updates When in Stall Mode	X	X	X
Advisory 1.113	USB HOST: Impossible To Attach a FS Device To An EHCI Port - Handoff To OHCI Is Not Functional	X	X	X
Advisory 2.1	USB Host Clock Drift Causes USB Spec Non-compliance in Certain Configurations	X	X	X

3 All Errata Listed by Module

Table 6. Errata List for the Control Module

Title **Page**

Table 7. Errata List for the Cortex-A8

Title **Page**

Table 8. Errata List for the Display Module

Title **Page**

Table 9. Errata List for the DMA4 Module

Title **Page**

Table 10. Errata List for the DPLL Module

Title **Page**

Table 11. Errata List for the GPMC Module

Title **Page**

Table 12. Errata List for the HDQ Module

Title **Page**

Table 13. Errata List for the HS USB Module

Title	Page
Usage Note 4.1 — HS USB Host Subsystem: Some Limitations Exist When Connecting to External Devices	28
Advisory 1.31 — HS USB HOST Controller: Device Aborts the Remote Wake-Up Sequence if AM3715/03 Wakes From OFF/RET to ON in USB TLL Mode	47
Advisory 1.44 — EHCI Controller: Issue in Suspend Resume Protocol	59
Advisory 1.56 — HS USB Host: ECHI and OHCI Controllers Cannot Work Concurrently	69
Advisory 1.64 — HSUSB Interoperability Issue With SMSC USB3320 PHY	74
Advisory 1.79 — USB Host EHCI May Stall When Exiting Smart-Standby Mode	81
Advisory 1.80 — USB Host EHCI May Stall When Running High Peak-Bandwidth Demanding Use Cases	82
Advisory 1.100 — Write to OHCI Registers May Not Complete	101
Advisory 1.108 — USBHOST Configured In Smart-Idle Can Lead To a Deadlock	105
Advisory 1.113 — USB HOST: Impossible To Attach a FS Device To An EHCI Port - Handoff To OHCI Is Not Functional	109
Advisory 2.1 — USB Host Clock Drift Causes USB Spec Non-compliance in Certain Configurations	110

Table 14. Errata List for the HS USB OTG Module

Title	Page
Advisory 1.20 — OCP Error Does Not Get Communicated to USBOTG	45
Advisory 1.38 — HS USB OTG Failing Access to Registers	55
Advisory 1.39 — HS USB OTG Software Reset Is Not Fully Functional	56
Advisory 1.47 — HS USB OTG: OTG_SYSCONFIG:AUTOIDLE Bit Is Not Reset Correctly	64
Advisory 1.52 — Standard OTG Compliance Electrical Tests for HOST Mode Will Fail	68
Advisory 1.76 — USB OTG DMA Status Register Read Return Zero	79
Advisory 1.81 — USB OTG DMA May Stall Under Specific Configuration	83
Advisory 1.82 — USB OTG DMA May Stall When Entering Standby Mode	83
Advisory 1.99 — HS USB OTG Spurious IRQ When Switching From DMA Mode 1 to DMA Mode 0 or to CPU Mode for Unloading the Last Short Packet of a Transaction	100
Advisory 1.102 — HS USB OTG Bus Reset	102
Advisory 1.109 — USB OTG Software Initiated ULPI Accesses To PHY Registers Can Halt the Bus	106

Table 15. Errata List for the I2C Module

Title	Page
Usage Note 2.11 — Extra Power Consumed When Repeated Start Operation Mode Is Enabled on I2C Interface Dedicated for Smart Reflex (I2C4)	21
Advisory 1.2 — I2C Module Does Not Allow 0-Byte Data Requests	32
Advisory 1.7 — I2C: RDR Flag May Be Incorrectly Set	35
Advisory 1.43 — I2C: I2C_STAT:XUDF Is Not Functional in Slave Transmitter Mode	58
Advisory 1.59 — I2C4 Does Not Meet I2C Standard AC Timing in FS Mode	72
Advisory 1.60 — I2C1 to 3 SCL Low Period Is Shorter in FS Mode	72
Advisory 1.92 — I2C: In SCCB Mode, Under Specific Conditions, the Module Might Hold the Bus by Keeping SCL Low	94
Advisory 1.93 — I2C: Wrong Behavior When Data With MSB 0 Is Put in the FIFO Before a Transfer With SBLOCK Is Started	94
Advisory 1.94 — I2C: After an Arbitration is Lost the Module Incorrectly Starts the Next Transfer	95

Table 16. Errata List for the INTC Module

Title **Page**

Table 17. Errata List for the ISP Module

Title **Page**

Table 18. Errata List for the IVA2 Module

Title **Page**

Table 19. Errata List for the IVA2.2 Module

Title **Page**

Table 20. Errata List for the JTAG Module

Title **Page**

Table 21. Errata List for the McBSP Module

Title **Page**

Table 22. Errata List for the MMC Module

Title **Page**

Table 23. Errata List for the MPU

Table 24. Errata List for the PRCM Module

Table 25. Errata List for the ROM Code Module

Table 26. Errata List for the SDRC Module

Table 27. Errata List for the SGX Module

Table 28. Errata List for the SPI Module

Title	Page
Advisory 1.15 — SPI Dummy DMA RX Request Generation	42
Advisory 1.33 — McSPI Can Generate a Wrong Underflow Interrupt	49

Table 29. Errata List for the SR Module

Title	Page
Advisory 1.46 — Voltage Processor TRANXDONE Interrupt Occurs Too Early	64

Table 30. Errata List for the TDL Module

Title	Page
Advisory 1.74 — Scan Shift Issues in GWG Memories	78

Table 31. Errata List for the Timer Module

Title	Page
Advisory 1.3 — Delay Required to Read Some GP, WD, and Sync Timer Registers After Wake-Up	32

Table 32. Errata List for the UART Module

Title	Page
Usage Note 2.7 — UART: Cannot Acknowledge Idle Requests in Smartidle Mode When Configured for DMA Operations	19
Advisory 1.5 — MDR1 Access Can Freeze UART Module When in IrDa Mode	34

Table 33. Errata List for the VENC Module

Title	Page
Usage Note 2.4 — VENC: Last Data Line Missing in PAL	18
Advisory 1.34 — VENC: TV Detect AC Coupling Mode Not Supported	49

Table 34. Errata List for Cautions

Title	Page
Usage Note 3.4 — Avoiding Voltage Drop When Booting at 1-GHz OPP With TPS65950A3	26

4 Usage Notes and Known Design Exceptions to Functional Specifications

4.1 Usage Notes (Limitations)

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 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 microrocessors, see the device-specific data manuals.

Usage Note 2.1 *Performance Limitation On LCD Read/Write Access Through RFBI L4 Port*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

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 processor 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 processor does not support two consecutive data transmits to the LCD (corresponding to two different and consecutive *Write* accesses from the initiator).

Usage Note 2.2 *IVA2: Under Hardware Emulation, IDLE Instruction Must Be Executed From L1P or L2 SRAM*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

This limitation was fixed and is kept for silicon errata consistency only. The workaround can be removed from the code.

Usage Note 2.3 *Domain Woken Up by Wake-Up Dependency Cannot Transition to Inactive State*

Revision(s) Affected 1.2, 1.1 and 1.0

Details When a domain is activated up by a wake-up dependency, it cannot transition to an inactive (RET/OFF) state until one of the following conditions are satisfied:

1. If the domain has a sleep dependency with another domain, this domain and the sleep dependency must be also activated (hard-wired or enabled by the software).
2. If the domain is an initiator, it must be taken out of *Standby* mode.

Note: The user should ensure that when a wake-up dependency is enabled for a domain, it is expected to be fully active after the wake-up occurs. This issue happens in a case where a domain has been woken-up, but not activated and used.

This issue can be avoided by ensuring the IVA2 processor is taken out of *Standby* mode prior to any new sleep transition.

Usage Note 2.4 *VENC: Last Data Line Missing in PAL*

Revision(s) Affected 1.2, 1.1 and 1.0

Details 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).

Usage Note 2.5 *Observability Signals Not Functional in OFF Mode*

Revision(s) Affected 1.2, 1.1 and 1.0

Details Signals are routed to observability pins through a buffer powered by VDD2. Consequently, observability is not functional when VDD2 is powered off. This impacts the observability debug feature, but has no functional drawback.

Note: Maintaining the VDD2 supply on the board has no effect since isolation cells are activated regardless of the supplied voltage.

Usage Note 2.6 *Constraints on Module Clocks When Using DVFS*

Revision(s) Affected 1.2, 1.1 and 1.0

Details Module interface timings can depend on the operating point (OPP). The timing closure is done for one instance of a module at a particular voltage. Operating conditions which work on one OPP are not automatically guaranteed on another module's OPP. For more details on the module interface switching characteristics, see the device-specific data manual.

Workaround(s) When Dynamic Voltage and Frequency Scaling (DVFS) is performed, the software should ensure that no timing violations will occur by the two following methods:

- Using software, reconfigure the module accordingly when DVFS is performed to avoid timing violations.
- Use a conservative frequency on the module interface clock corresponding to the performance of the lower OPP used to ensure that DVFS can be performed without additional software management.

Usage Note 2.7 *UART: Cannot Acknowledge Idle Requests in Smartidle Mode When Configured for DMA Operations*

Revision(s) Affected 1.2, 1.1 and 1.0

Details 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.

Usage Note 2.9 *GPIO Is Driving Random Values When Device Returns From OFF Mode*

Revision(s) Affected 1.2, 1.1 and 1.0

Details The GPIO modules in the PER (Peripheral) domain are driving random values when the device returns from OFF mode with the VDD2 voltage shut down. Regardless of the active configuration of a GPIO (input or output), when coming back from OFF mode, GPIO can randomly drive the line high or low.

Root cause: The isolation at the boundary of the peripheral domain is done through isolation latches that are supplied by VDD2. When VDD2 is shut down, the content of the latches is lost. When the device is waking-up from the OFF mode, the padconf is automatically restored by the hardware and the isolation latches are driving the pin until isolation is released (PER is woken-up). There is no reset value for the isolation latches, leading to random values driven by the GPIO.

This is not impacting the GPIO from the WU domain. Only GPIO is implementing this kind of isolation latch, other functions are not impacted by this limitation.

Workaround(s) Workaround 1 allows software to work in software supervised mode while Workaround 2 gets the benefit of hardware dependency. For both workarounds, software is used to configure the internal pull in the padconf register to the desired value.

Workaround1: Change padconf mode to safe mode before initiating the transition to OFF. This avoids the line being driven by the isolation latch and the desired level is maintained by the pull.

Workaround2: Create a WU dependency between WU domain and PER domain (PM_WKDEP_PER[4]: EN_WKUP=1). This will wake up the PER domain, then release the isolation latch before the isolation cell is released at the pad boundary. The reset value of the GPIO module is configuring the GPIO in input, then the pull ensures that the line is driven to the right value until the GPIO module is restored by the applicative software

Usage Note 2.10 *Maximum 12 Bits Output (1 bit sign + 11 bits of value) Is Supported on CAVLD of iVLCD*

Revision(s) Affected 1.2, 1.1 and 1.0

Details While decoding DC coefficients of an Intra16x16 MB, the iVLCD is not able to handle coefficients with range greater than 11 bits. It is observed that the value returned by the iVLCD will be the equivalent of (actual_value & 0x07FF). For example, value of 0x08A7 will be decoded as 0x00A7. However, the bits consumed by the iVLCD is correct and the decoding continues.

As per the standard, normal imagery can generate DC coeffs with range greater than 11 bits (especially at low QP) and some of the test cases are failing due to this issue. Since the error is in DC it will not only affect the MB in question, but propagate to neighboring MBs in the slice. The impact is visually noticeable.

This issue only occurs if either of the following conditions are met:

- For QP<10, Luma DC coefficients of Intra16x16 MB
- For QP<4, Chroma DC coefficients

This special case must be handled with C64x+ software, which increases code size and adds extra MHz.

The estimate of the cost for the work added is: extra 3-4 MHz and 25kB of code.

Usage Note 2.11 *Extra Power Consumed When Repeated Start Operation Mode Is Enabled on I2C Interface Dedicated for Smart Reflex (I2C4)*

Revision(s) Affected 1.2, 1.1 and 1.0

Details On the I2C interface dedicated for Smart Reflex communication with the Power Management IC (I2C4), when Repeated Start operation mode is enabled (PRM_VC_I2C_CFG[SREN]=’1’), the I2C lines (SCL and SDA) are always driving a low state between two I2C commands (i.e. when there is no I2C traffic).

Knowing that there are external Pull-Up attached on these lines, this will impact the power consumption.

Setting PRM_VC_I2C_CFG[SREN] register bit to ’0’ will allow the I2C4 lines driving a high state between two I2C commands.

Usage Note 2.12 *SGX 192 MHz Limitation With TVOUT @ 54 MHz*

Revision(s) Affected 1.2, 1.1 and 1.0

Details The siimultaneous operation of SGX530 @ 192 MHz and TVOUT clock @ 54 MHz is not possible.

SGX functional clock can be selected from four sources:

- Divided version of the CORE.CLK [by 2, 3, 4, 6]
- Divided version of the COREX2.CLK [by 3, 5]
- 96 MHz clock version
- 192 MHz clock version

The frequency combination of SGX530 @ 192 MHz along with TVOUT @ 54 MHz is not possible with the TVout clock divider which maximum value is 31.

TV-Out requires a 54MHz clock provided by the DPLL-PER (DPLL4) , then the DPLL-PER has to be set to 864 MHz to get the 54MHz clock. In the case of setting DPLL-PER at 1728MHz to get 192Mhz on SGX will not allow getting the 54MHz on TVOUT since high-speed divider is limited to 31.

Workaround(s) Only if TVOUT@54MHz is a workaround required. The SGX clock will have to be sourced from CORE-DPLL (SGX clock will scale with L3 clock). When software is using TV-Out @ 54 Mhz, SGX functional clock has to be derived from CORE-DPLL which allows up to 200MHz clock frequency depending on the CORE clock configuration.

Usage Note 2.13 *PWDNx Bit in CLock Manager Is Reseting the DPLL4 Mx HSDIVIDER Coefficient Previously Programmed*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

When PWRDN bit is set, it resets the internal HSDIVIDER divide-by value (Mx). The reset value gets loaded instead of the previous value. The following HSDIVIDERs exhibit this behavior:

- DPLL4: M6 / M5 / M4 / M3 / M2 (CM_CLKEN_PLL[31:26] register bits)
- DPLL3: M3 (CM_CLKEN_PLL[12] register bit).

Workaround(s)

No performance impact by using the following workaround.

It is mandatory to apply the following sequence to ensure the write value will be loaded in DPLL HSDIVIDER FSM:

1. Read HSDIVIDER register.
2. Write HSDIVIDER + 1 (Or any other value different from the Read value).
3. Write HSDIVIDER with the correct value.

The global sequence when using PWRDN bit is:

1. Disable Mx HSDIVIDER clock output related functional clock enable bits (in CM_FCLKEN_xxx / CM_ICLKEN_xxx).
2. Enable PWRDN bit of HSDIVIDER.
3. Disable PWRDN bit of HSDIVIDER.
4. Read current HSDIVIDER register value.
5. Write different value in HSDIVIDER register.
6. Write expected value in HSDIVIDER register.
7. Enable Mx HSDIVIDER clock output related functional clocks (CM_FCLKEN_xxx / CM_ICLKEN_xxx).

Usage Note 2.15 *MMC IOs Reliability Issue*

Revision(s) Affected 1.1, 1.0

Details

Design error in IOs cells used for MMC/SD/SDIO1 interfaces leading to reliability issue when:

- Device is kept under reset with sys_nrespwron = 0.
- OR device is waking up from reset until SW sets IOs power down control bit (CONTROL_PBIAS_LITE.PBIASLITEPWRDNZ0 for MMC/SD/SDIO1).
- OR IOs are in power down mode (programmed in previous registers).

The effect of this reliability issue is IO dysfunction. That is, it stays in power down or does not enter power down anymore. No IO performance degradation is expected.

Impact in customer application is less than 1 dppm over 5 and 7 years product life-time; assuming unreliable conditions described above are reached 100% of the time.

Usage Note 2.16 *Downscaling Limitations*

Revision(s) Affected

1.2, 1.1 and 1.0

Details

Synclost interrupts are observed when performing downscaling and shifting resulting windows on the right of the LCD panel. This interrupt occurs because there is not sufficient time to load the necessary lines buffer before performing a downscaling operation. DISPC video pipeline is expected to be ready with the pixels at the end of HSW+HBP period as shown in Figure 2. This limitation is applicable on 3 and 5 taps configuration.

Workaround(s)

To ensure proper downscaling operation, the following conditions should be met:

- ((PPL-SizeX-PosX) + (HBP+HSW+HFP)) * PCD >= Max(0,Ceil(DS)-2)*(OrgSizeX+1)
- ((PPL-SizeX) + (HBP+HSW+HFP)) * PCD >= Max(0,Ceil(DS)-1)*(OrgSizeX+1)

In case VRFB 90/270 rotation with YUV or RGB16 format, the following conditions should be met:

- ((PPL-SizeX-PosX) + (HBP+HSW+HFP)) * PCD >= Max(0,Ceil(DS)-1)*(OrgSizeX+1)
- ((PPL-SizeX) + (HBP+HSW+HFP)) * PCD >= (Max(0,Ceil(DS)-1)*(OrgSizeX+1))* 2

Where

DS = (OrgSizeY+1)/(SizeY+1)
Ceil(x) is round-up of x

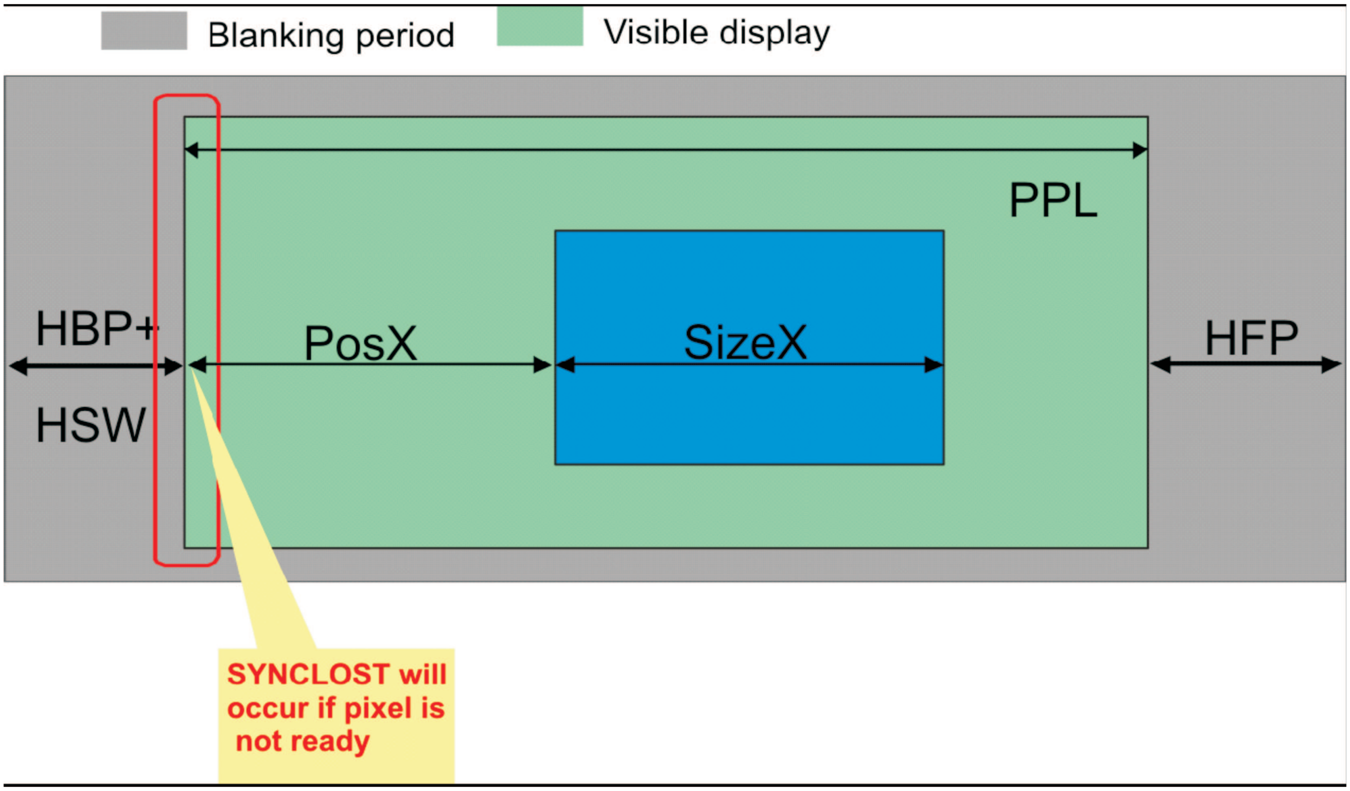


Figure 2. Synclost Interrupt

Usage Note 2.17 *TVOUT Line Shift Issue When Using DSI PLL for DISPC_FCLK*

Revision(s) Affected 1.2, 1.1 and 1.0

Details When using the clock from DSI_PLL (DSI1_PLL_FCLK) as a functional clock for the DISPC, on TVOUT output, the pixel lines are randomly in advance of 294ns.

The failure does not occur when using the DSS1_ALWON_FCLK (from DPLL4).

The failure is random. The faulty lines have "good timing duration," meaning the line duration is 64 us. The synchronization pulse and the burst color are also good.

Workaround(s) When TVOUT is used, the DISPC_FCLK clock source must be DPLL4 to avoid the line shift on TVOUT.

Usage Note 2.18 *HDQ™/1-Wire® Communication Constraints*

Revision(s) Affected 1.2, 1.1 and 1.0

Details HDQ/1-Wire protocols use a return-to-1 mechanism that requires an external pullup resistor on the line. There is a timing limitation on this return-to-1 mechanism that requires a constraint on the external pullup resistor (R) and the capacitive load (C) of the wire.

Workaround(s) There is a constraint in the design for the maximum allowed rise time of the wire. After writing data to the wire, the HDQ/1-Wire module samples the logic value of the wire 1 FSM (finite state machine) clock cycle later. The FSM expects to read back 1 value from the wire. This constraint must be taken into account when calculating the pullup resistor (R) according to the capacitive load (C) of the wire.

The maximum RC (pullup resistor and capacitive load) value should be calculated as:

$R < 1200 \text{ ns} / (10e^{-12} + C)$

For example, if R = 4.7K, C < 245 pF.

Usage Note 3.4 *Avoiding Voltage Drop When Booting at 1-GHz OPP With TPS65950A3*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

TPS65950A3 implements two separate registers for control of VDD1 voltage. First register, VDD1_SEL, can be programmed through I2C1 only while second register, VDD1_SR_CONTROL, can be programmed by SR I2C, I2C4.

By default, the VDD1 DCDC is providing the voltage which is set in VDD1_SEL until AVS is configured on the DM37x side and enabled on TPS65950A3 side. As soon as DCDC_GLOBAL_CFG is set to enable SR_I2C IOs, then DCDC reference switches to VDD1_SR_CONTROL.

The default value (also known as reset value) is corresponding to 1.2-V voltage value.

In the context of 1-GHz boot, the system is started at 1-GHz OPP with AVS disabled (using 1-GHz nominal voltage, refer to DM operating conditions addendum) and AVS will be enabled later on when kernel has booted.

The voltage requirement to run 1-GHz OPP with AVS disabled is higher than 1.2 V (default value for VDD1_SR_CONTROL).

In that context, if user is starting boot at OPP1GHz by rising VDD1_SEL, then system will maintain the right voltage level until AVS is enabled. At that point, as soon as DM37x will perform the access to DCDC_GLOBAL_CFG to enable SR) I2C IOs, then DCDC will take the reset value of VDD1_SR_CONTROL and provide 1.2 V instead of nominal voltage until this VDD1_SR_CONTROL is overwritten. This voltage drop to 1.2 V is unsafe if system is running 1-GHz OPP and can lead to instability.

It is not possible to overwrite SR_I2C reset value before DCDC_GLOBAL_CFG set the IOs to avoid that situation. User should avoid this situation by using a sequence which is not running 1-GHz OPP with 1.2 V on VDD1.

Workaround(s)

Solution#1: System boot at OPP100 or lower OPP (avoid booting at OPP130 and OPP1GHz is a safe option with no re-work).

Solution#2: Boot at 1 GHz but initiate the first AVS calibration for OPP100 or lower OPP. (As soon as 1 OPP calibration happens for one OPP with nominal voltage lower than 1.2 V, then no restriction to perform other OPP calibration.) Sequence can be:

- Boot at 1-GHZ OPP with AVS disabled
- Switch to OPP100
- Run OPP100 AVS calibration
- Run OPP1GHz AVS calibration
- Switch to OPP1GHz
- etc.

Solution#3: Boot at 1-GHz OPP and stay at this OPP until AVS calibration. Right before setting DCDC_GLOBAL_CFG:

- Change OPP to OPP100
- Set DCDC_GLOBAL_CFG (voltage changed to 1.2 V, which is aligned with OPP100)
- Send a bypass command corresponding to 1-GHz nominal voltage through SR_I2C
- Switch back to 1-GHz OPP.

Solution#4: Before enabling OPP1GHz (this means in the bootloader), configure DM37x SR module to launch a bypass command on I2C then set DCDC_GLOBAL_CFG and launch 1-GHz OPP nominal voltage to VDD1_SR_CONTROL. At that point 1-GHz boot can be launched safely.

Usage Note 3.5 *Undesired McBSP slave mode behavior during reset without CLKR/CLKX*

Revision(s) Affected 1.2, 1.1 and 1.0

Details As described in the Technical Reference Manual, the McBSP port requires two clock cycles of CLKR/CLKX during reset to synchronize the interface configuration. When the McBSP is configured in slave mode, the necessary clock is provided by the other device (the interface master). If the master device does not provide the necessary clock before the first frame pulse, then undesired behavior of the McBSP port will occur since it will only be properly initialized during the first two clock cycles of CLKR/CLKX. Impacts of this situation include:

- McBSP port does not receive the first frame of data from the master
- Undefined output (glitch) on McBSP transmit line during the first frame pulse

Workaround(s) If possible, the master should provide CLKR/CLKX before the first frame pulse (during McBSP initialization) to avoid this issue. No other workaround is available.

Usage Note 4.1 HS USB Host Subsystem: Some Limitations Exist When Connecting to External Devices

Revision(s) Affected 1.2, 1.1 and 1.0

Details As shown in Figure 3, the DM37x device includes a high-speed universal serial bus (USB) OTG controller and a high-speed USB host subsystem.

Workaround(s) **Note:** USB Port 3 is not available on CUS package.

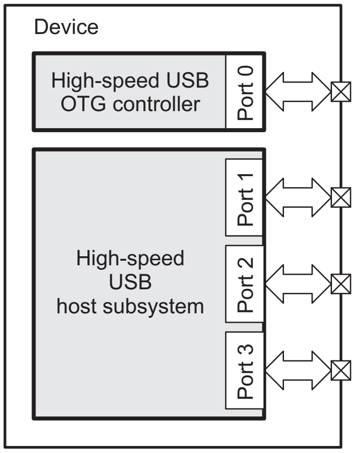


Figure 3. DM37x USB Modules

The high-speed USB OTG controller supports a single USB port which uses a UTMI low-pin interface (ULPI) to connect to an off-chip transceiver (12-pin/8-bit single-data rate mode). As shown in Figure 4, USB Port 0 can be connected to the USB 2.0 PHY included in DM37x companion chips, e.g. TPS65950A3. Alternatively, USB Port 0 can be connected to discrete USB 2.0 PHYs which include a UTMI low-pin interface (ULPI) and are capable if sourcing an output clock.

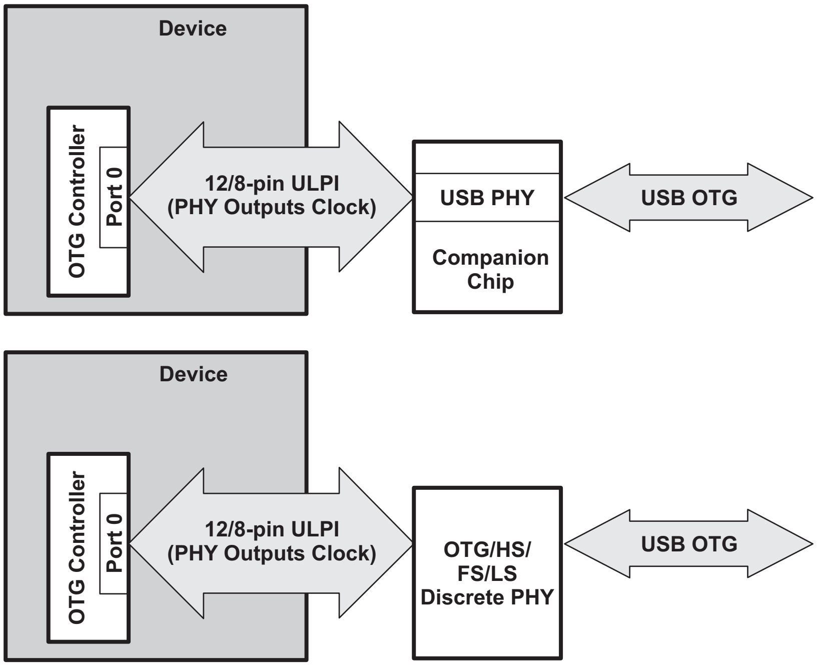


Figure 4. Typical Uses for USB Port 0

As shown in Figure 5, 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.

Both the OHCI and EHCI controllers can operate in parallel. Each of the three USB ports is owned by exactly one of the controllers at any time. Also, when a controller uses the UTMI port, the USB TLL block converts that port to a ULPI transceiver-less link logic (TLL) format.

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

- On silicon revision 1.0 only, 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 Advisory 1.55.
- USB Port 3 cannot operate in ULPI mode; it can only operate in serial or ULPI TLL mode. Furthermore, USB Port 3 is not available in the CUS package.

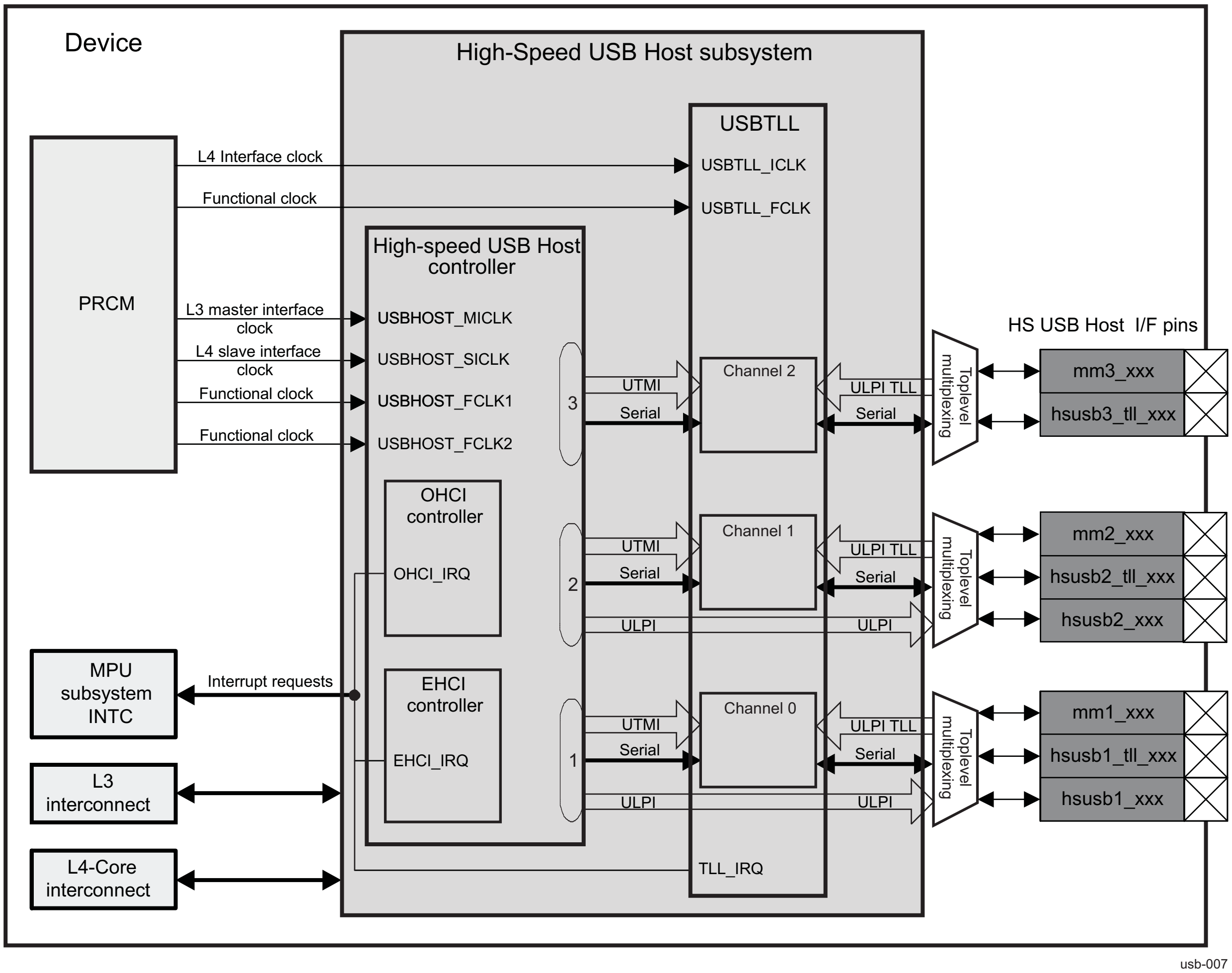
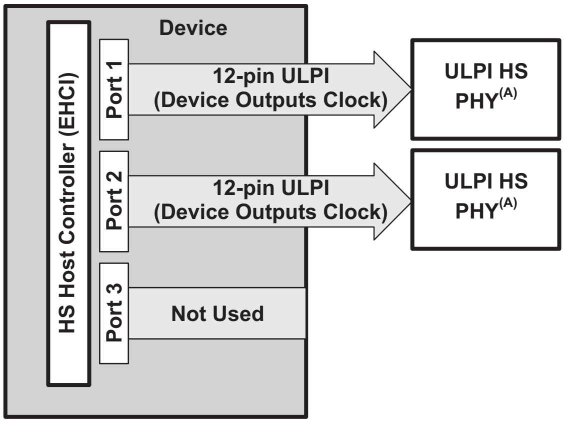


Figure 5. High-Speed USB Host Subsystem Highlight

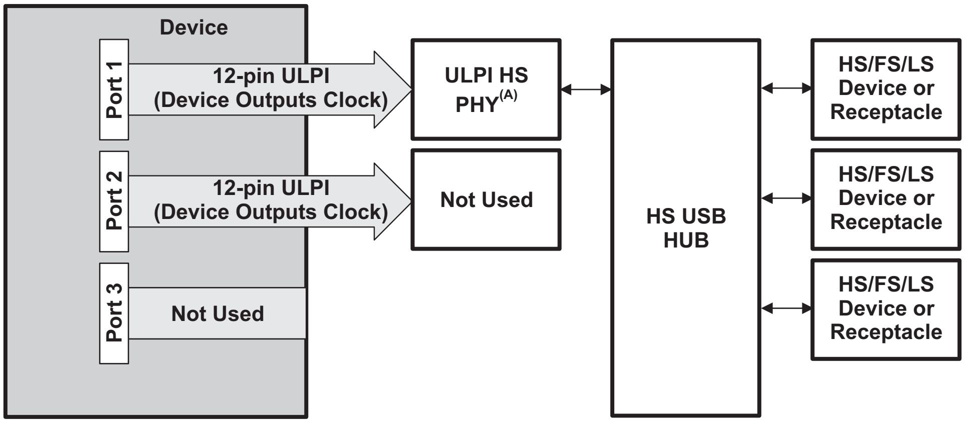
As shown in Figure 6, USB Port 1 and Port 2 can be used to connect to external high-speed PHYs which include a ULPI port. However, in this case, the external USB PHY must be able to accept a source clock generated by the DM37x high-speed USB controller. Also, in this usage model the USB ports cannot 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 7.

Note: USB Port 3 does not support ULPI mode. USB Port 3 is not available on CUS package.



A. USB port 3 does not support ULPI mode.
Also, USB port 3 not available on CUS package.

Figure 6. Connecting to High-Speed PHYs Using USB Ports



A. USB port 3 does not support ULPI mode. Also, USB port 3 not available on CUS package.

Figure 7. Connecting to a High-Speed USB Hub

Usage Note 4.2 *"Speed Binned" Bit in Control Device Status Register Not Programmed in Devices Built and Tested Prior to June 2011*

Revision(s) Affected 1.2, 1.1 and 1.0

Details The Speed Binned bit (bit 9) of the Control Device Status Register (0x4800 244C) may not accurately reflect the maximum speed capabilities of the device. Some 1-GHz capable DM37x devices built and tested before June 2011 may have a '0' for this bit.

Workaround(s) Use the device marking on the top of the package to determine the speed capabilities of the device. Check the device markings against the Device Nomenclature information in the device-specific data manual.

4.2 Known Design Exceptions to Functional Specifications (Bugs)

Advisory 1.2 — *I2C Module Does Not Allow 0-Byte Data Requests*

Revision(s) Affected 1.2, 1.1 and 1.0

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) There is no workaround for this issue. ***Do not*** use 0-byte data requests.

Advisory 1.3 — *Delay Required to Read Some GP, WD, and Sync Timer Registers After Wake-Up*

Revision(s) Affected 1.2, 1.1 and 1.0

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 processor 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-wake-up 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 1.4 *IVA2: C64x+ L1D Cache May Lose Data or Hang DMA Operations Under Certain Conditions*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

Under certain conditions, parallel loads with predication to the same cache line may cause victims to be dropped and/or DMA to hang.
All of the following conditions ***must*** be true in order for this problem to occur:

1. Two LD instructions in parallel.
2. Both are LDs to the same cache line (upper 26 address bits are the same).
3. The LD using T1 is predicated and the predicate is *false.*
4. The LD using T2 is either not predicated, or is predicated and the predicate is *true.*
5. The cache line is absent from the cache.
6. The two other lines in the same L1D set are valid.
7. The LRU cache line in the set is dirty.

Results:

- L1D informs L2 to expect a victim for the affected set.
- L2 stalls DMAs with addresses that correspond to that set (DMA includes accesses from IDMA and EDMA).

  Note: DMA includes accesses from IDMA, EDMA, and any external masters, such as PCI or other CPUs.

- L1D processes the true-predicated request correctly.
- L1D does not send the indicated victim.

Impact:

- If the load instruction reads a cacheable location:
  - The updated data in the LRU line gets dropped.
  - DMA accesses whose addresses match the affected set hang.
- If the load instruction reads a non-cacheable location:
  - L1D retains the updated data from the LRU line.
  - DMA reads may see stale data if the LRU line's address is in L2 memory.

Workaround(s)

Use Code Gen patch 6.0.3 (available on update advisor) to recompile your source code and avoid this issue. Libraries supplied by TI will be re-released using the 6.0.3 compiler patch. Customer-generated libraries from TI's third-party supplier may also need to be recompiled.

For existing object code and libraries, an available Perl script can determine locations of parallel predicated loads that may fail. The script is available at the same update advisor location as the Code Gen patch.

Advisory 1.5 *MDR1 Access Can Freeze UART Module When in IrDa Mode*

Revision(s) Affected 1.2, 1.1 and 1.0

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 1.6 *IVA2: Block Cache Operations Word Count (*WC) Must be Less Than or Equal to 0xFF80*

Revision(s) Affected 1.2, 1.1 and 1.0

Details When performing any block cache operation, such as "Writeback", "Writeback with Invalidate", or "Invalidate", for any memory controller or memory range (e.g., L1P, L2, L1D) the word count programmed ***must be*** less than or equal to 0xFF80. If a value greater than 0xFF80 is desired, then this must be broken into multiple operations. The following registers are affected: L2WWC, L2WIWC, L2IWC, L1PIWC, L1DWIWC, L1DWWC, and L1DIWC.

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

Advisory 1.7 *I2C: RDR Flag May Be Incorrectly Set*

Revision(s) Affected 1.2, 1.1 and 1.0

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 8 when receiving an I2C RDR interrupt.

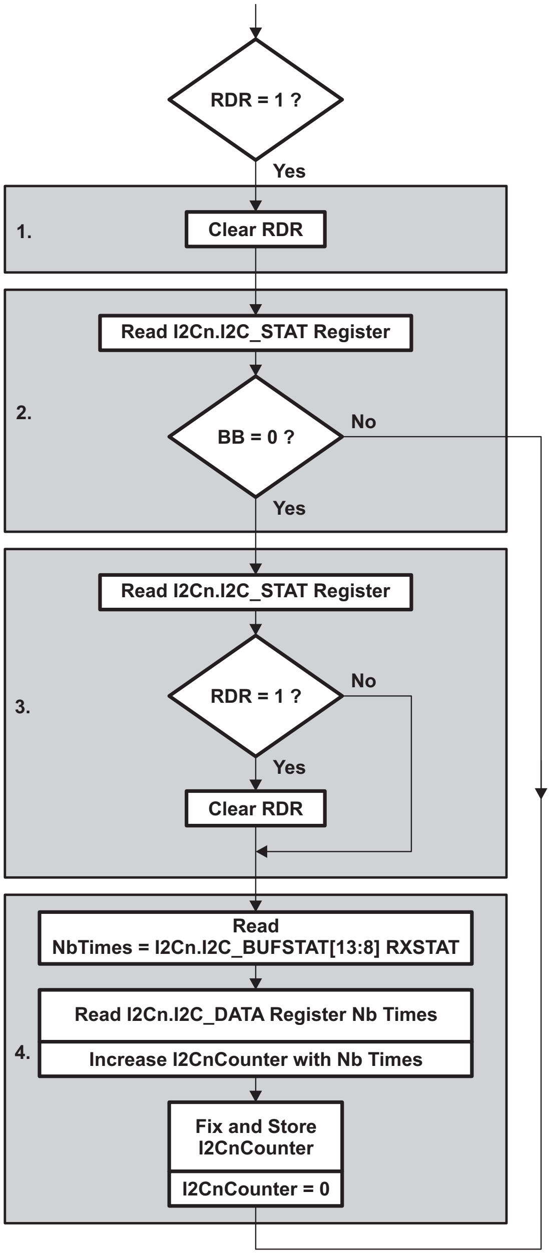

RDR = 1 ?
Yes
1.
Clear RDR
Read I2Cn.I2C_STAT Register
2.
BB = 0 ?
No
Yes
Read I2Cn.I2C_STAT Register
3.
RDR = 1 ?
No
Yes
Clear RDR
Read
NbTimes = I2Cn.I2C_BUFSTAT[13:8] RXSTAT
Read I2Cn.I2C_DATA Register Nb Times
4.
Increase I2CnCounter with Nb Times
Fix and Store
I2CnCounter
I2CnCounter = 0

Figure 8. I2C Flowchart

Advisory 1.8 *IVA2: EDMA Channel Priority Is Not Correctly Enforced*

Revision(s) Affected 1.2, 1.1 and 1.0

Details The EDMA channel controller (TPCC) has 2 event queues: Q0 and Q1. Each queue can be mapped to one of the two Transfer Controllers (TPTC): TPTC0 or TPTC1. As explained in the *AM/DM37x Multimedia Device Technical Reference Manual* (literature number SPRUGN4), the events in each event queue will be extracted as soon as the corresponding TPTC is available for a new Transfer Request (TR) to be programmed into the TPTC. However, due to an issue in the IVA2.2 subsystem, the requests queued in Q1 cannot be submitted to their TPTC as long as Q0 is not empty.

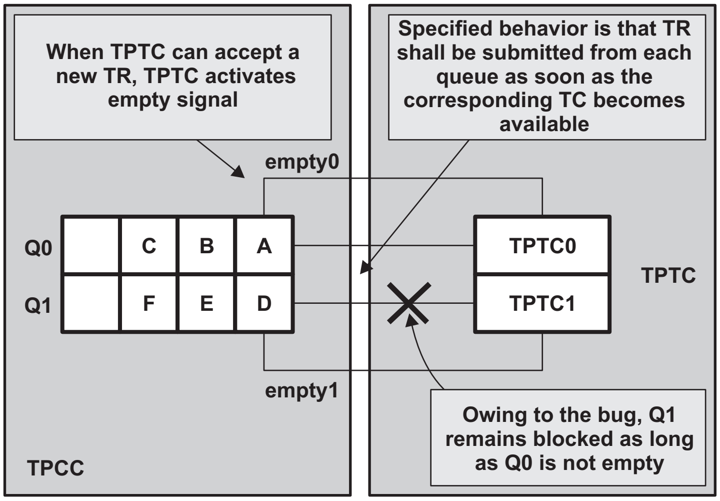

When TPTC can accept a new TR, TPTC activates empty signal
Specified behavior is that TR shall be submitted from each queue as soon as the corresponding TC becomes available
empty0
Q0
C
B
A
Q1
F
E
D
TPTC0
TPTC1
TPTC
empty1
Owing to the bug, Q1 remains blocked as long as Q0 is not empty
TPCC

Figure 9. TR Submission Scheme

Workaround(s) Infrequent short transfers (e.g., latency critical synchronized transfers to/from peripheral) must be placed in Q0, and longer transfers (e.g., block copy) in Q1. For example, when a latency critical transfer is placed in EDMA and a longer transfer in QDMA, the following programming model will minimize the impact of this limitation:

- EDMA Event is queued to Q0 (using IVA_TPCC.DMAQNUMn register, n from 0 to 7)
- QDMA Event is queued to Q1 (using IVA_TPCC.QDMAQNUM register)
- Queue to TC Mapping is Q0:TPTC1 Q1:TPTC0 (using IVA_TPCC.QUETCMAP register)
- Queue Priority is Q0 = 0x0, Q1 = 0x7 (using IVA_TPCC.QUEPRI register)

Advisory 1.9 *Inactive State Management: Impossible to Transition to OFF or RETENTION States*

Revision(s) Affected 1.2, 1.1 and 1.0

Details If a power domain meets the conditions to go to an INACTIVE state (that is, 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 or OFF state automatically from INACTIVE state, even if software updates the POWERSTATE bit to 0x1 (RET) or 0x0 (OFF). The domain must be transitioned to the ON state before it can transition to the RET or OFF states.

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

1. Do not use autostate with PM_PWSTCTRL_XXX.POWERSTATE=0x3 (ON) to put power state in INACTIVE. Using autostate from ON to RET (or OFF) transition is not impacted..
2. Perform wake-up event (software must force wake-up) to transition to an active state before transitioning to the RET or OFF states.

Advisory 1.10 *Cortex-A8 Errata List*

Revision(s) Affected 1.2, 1.1 and 1.0

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 3 to determine which version of the ARM Cortex-A8 processor is included in each AM3715/03 silicon revision.

Advisory 1.11 *Inappropriate Warm Reset Generation on Smart Reflex I2C Error*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

The Voltage Controller generates an I2C access error when an I2C command is sent to PowerIC during a voltage domain sleep transition. When the access error is generated, an interrupt is generated for software error handling. PRCM generates a warm_reset when an access error is generated by the Voltage Controller during voltage domain wake-up transition. This is to recover the full system (processor + Peripherals + Power IC) and avoid a deadlock (processor wake-up transition stalling due to I2C access error and VDD1/ VDD2 not supplied). Since both VDD1 and VDD2 Voltage Controllers share the same interrupt line, an access error for VDD2 Voltage controller is also propagated to VDD1 Voltage controller.

Note: This bug occurs only with the I2C module used by the Smart Reflex module. Other instances of the I2C module are not impacted by this issue.

In the following scenario:

1. VDD1 is performing a wake-up transition while VDD2 is performing a sleep transition.
2. An I2C access error is generated on VDD2 sleep request and VDD1 and VDD2 share the same error line, so this access error on VDD2 is broadcasted to both VDD1 and VDD2.
3. The condition for a warm_reset generation is met on VDD1, and Warm_reset is asserted inappropriately.

Workaround(s)

This issue was detected in simulation, but it is a corner case which is not expected to occur in production since the I2C access error should never occur if the PCB is safe.

Advisory 1.12 *IVA2: Back-to-Back SPLOOPS With Interrupts Can Cause Incorrect Operation*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

Back-to-back software pipeline loops (SPLOOPs) with interrupts can cause incorrect operation during C64x+ execution when:

1. The DSP code contains two contiguous SPLOOP/D/W.
2. DSP is interrupted when executing the first SPLOOP/D/W.
3. There are less than two execute packets between the SPKERNEL of the first SPLOOP/D/W and the SPLOOP/D/W instruction of the second.

When this issue occurs, the first SPLOOP/D/W terminates abruptly, i.e. without completing the loop, even though the termination condition is false. The failure mechanism can be seen as a hang, or by the first SPLOOP/D/W draining for the interrupt and starting the second SPLOOP/D/W without taking the interrupt or returning to complete the first SPLOOP/D/W.

Workaround(s)

This issue can be worked around by ensuring there are at least two execute packets between the SPKERNEL of the first SPLOOP/D/W and the SPLOOP/D/W instruction of the second.

A fix is implemented in the compiler included in version CGT6.0.6 of the Code Generation Tools. Starting with this revision of the CGT, the compiler ensures there are two cycles between SPKERNEL and SPLOOP/D/W instructions by adding the appropriate number of NOP instructions following the SPKERNEL instruction.

Advisory 1.13 *IVA2: DSP Generates False Internal Exception for Multiple Writes*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

A false internal exception can be generated by C64x+ if an interrupt happens during DSP code flow and the instructions that will be annulled during pipeline flush are dependant. This issue occurs in the exception detection logic. It examines the DSP instructions during the pipeline flush, even though they have been annulled. The hardware does not detect this and incorrectly assumes that multiple write instructions to the same register with 2 different conditional registers will be executed.

The DSP generates an incorrect internal exception in the following scenario: the CPU is draining the pipeline as part of an interrupt context switch. During this time, it annuls instructions in the pipeline. The first annulled execute packet changes the value of one or more predicate registers. The second annulled execute packet has two or more predicated instructions that use the predicates written in the previous cycle. The values held in the predicate registers appear to cause the instructions in the second annulled execute packet to write to the same register. The conflicting writes would not happen if the first execute packet was not annulled.

Example:

```
ZERO A0
ZERO A1
------------> (interrupt occurs here)
MVK 1, A0; (annulled)
[!A0] MVK 2, A1; (annulled)
|| [!B0] MVK 3, A1; (annulled)
```

Invalid exception triggers as it appears that the last two MVK will both write A1.

Even if this issue appears in a DSP code, it does not affect the code flow and does not produce an unexpected exit routine value.

Workaround(s)

The CPU only recognizes the incorrect exception while it drains the pipeline for an interrupt. As a result, the CPU begins exception processing upon reaching the interrupt handler. The NRP (NMI Return Pointer Register) and NTSR (NMI Task State Register) will reflect the state of the machine upon arriving at the interrupt handler.

Therefore, to identify the incorrect resource conflict exception in software, verify the following conditions at the beginning of the exception handler prior to normal exception processing:

1. Exception occurred during an interrupt context switch.
   - In NTSR, verify that INT=1, SPLX=0, IB=0, CXM=00.
   - Verify that NRP points to an interrupt service fetch packet. That is, (NRP & 0xFFFFFE1F) == (ISTP & 0xFFFFFE1F).
2. The exception is a resource conflict exception. In IERR, verify that RCX == 1 and all other IERR bits == 0.
3. The exception is an internal exception. In EFR, verify that IXF == 1 and all other EFR bits == 0.

Upon matching the above conditions, suppress the exception as follows:

- Clear EFR.IXF by writing 2 to ECR.
- Resume the interrupt handler by branching to NRP.

The above workaround identifies and suppresses all cases of the incorrect resource conflict exception. It resumes normal program execution when the incorrect exception occurs, and has minimal impact on the execution time of program code. The interrupted code sequence runs as expected when the interrupt handler returns.

The workaround also suppresses a particular valid exception case that is indistinguishable from the incorrect case. Specifically, the code will suppress the exception generated by two instructions with different delay slots (e.g., LDW and DOTP2) writing to the same register in the same cycle, where the conflicting writes occur during the interrupt context switch.

Example of sequence with incorrectly suppressed exception:

```
LDW *A0, A1
DOTP2 A3, A2, A1
NOP
-----------------> interrupt occurs
NOP
NOP ; Both LDW and DOTP2 write to A1 this cycle
```

The workaround will not suppress these valid resource conflict exceptions if the multiple writes occur outside an interrupt context switch. That is, the workaround will not suppress the exception generated by the code above when it executes without an interfering interrupt.

Advisory 1.14 *GPMC May Stall After 256 Write Accesses in NAND_DATA, NAND_COMMAND, or NAND_ADDRESS Registers*

Revision(s) Affected

1.2, 1.1 and 1.0

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 1.15 *SPI Dummy DMA RX Request Generation*

Revision(s) Affected 1.2, 1.1 and 1.0

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.

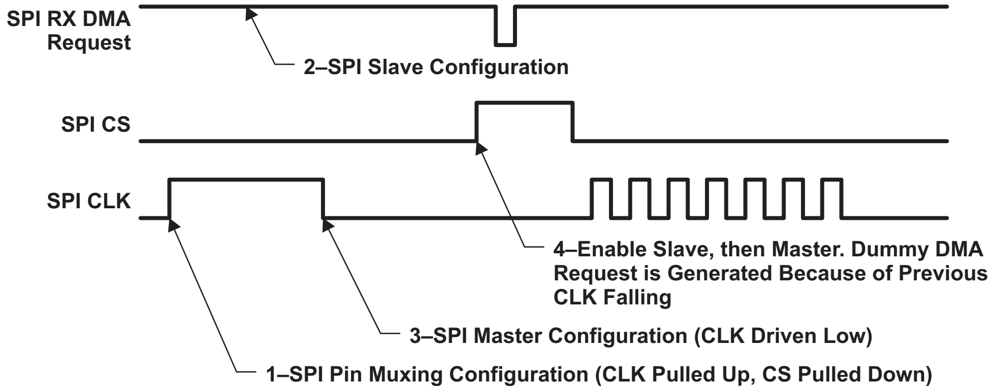

SPI RX DMA Request
2–SPI Slave Configuration
SPI CS
SPI CLK
4–Enable Slave, then Master. Dummy DMA Request is Generated Because of Previous CLK Falling
3–SPI Master Configuration (CLK Driven Low)
1–SPI Pin Muxing Configuration (CLK Pulled Up, CS Pulled Down)

Figure 10. SPI Dummy DMA RX Generation

Workaround(s)

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

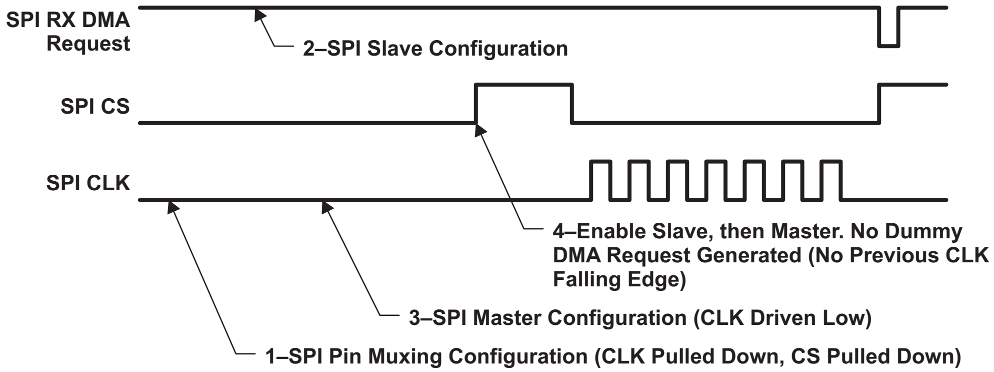

SPI RX DMA Request
2–SPI Slave Configuration
SPI CS
SPI CLK
4–Enable Slave, then Master. No Dummy DMA Request Generated (No Previous CLK Falling Edge)
3–SPI Master Configuration (CLK Driven Low)
1–SPI Pin Muxing Configuration (CLK Pulled Down, CS Pulled Down)

Figure 11. Dummy DMA RX Generation (No Clock Edge)

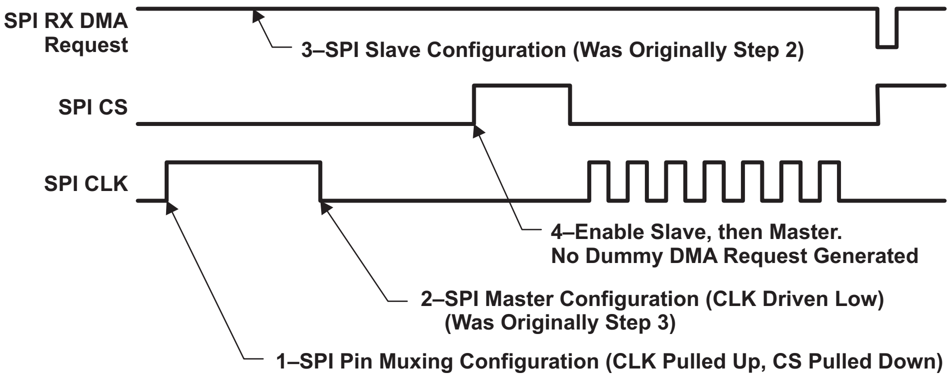

SPI RX DMA Request
3–SPI Slave Configuration (Was Originally Step 2)
SPI CS
SPI CLK
4–Enable Slave, then Master.
No Dummy DMA Request Generated
2–SPI Master Configuration (CLK Driven Low)
(Was Originally Step 3)
1–SPI Pin Muxing Configuration (CLK Pulled Up, CS Pulled Down)

Figure 12. Dummy DMA RX Generation (Slave Mode After Clock Line Driven to Default Value)

Advisory 1.17 *IVA2: ISP/GFX Dependencies*

Revision(s) Affected 1.2, 1.1 and 1.0

Details Scenario:

- IVA2 is idled.
- Interrupt is propagated to the IVA2.
- IVA2 INTC (WUGEN) generates a Wake-Up event to the PRCM for IVA2 Wake-Up.
- A Wake-Up dependency is defined between IVA and ISP, or IVA and GFX.
- PRCM wakes-up IVA2, which initiates wake-up for ISP/GFX.
- Interrupt is propagated to IVA2 Core.
- At this point IVA2 Core could initiate a transfer to ISP/GFX even though the ISP/GFX module may not have finished its wake-up sequence, thus resulting in OCP transfer fail.

Workaround(s) The IVA2 software should look at the ISP/GFX clock activity status bit to verify that the ISP/GFX domain is ON before performing any accesses. Next, 10 NOPs should be inserted for additional margin.

Advisory 1.20 *OCP Error Does Not Get Communicated to USBOTG*

Revision(s) Affected 1.2, 1.1 and 1.0

Details The OCP master interface of the sub-chip has a design limitation. This interface connects to an OCP slave on one side and to the AHB master interface of Mentor USB OTG core, MUSBMHDRC, on the other side. It is a wrapper that converts AHB master read/write requests to equivalent OCP read/write requests, and the OCP response/data from OCP slave is expected to be converted to equivalent AHB response/data. In the design, the AHB hresp is always set to OKAY. As a result, an OCP response is never translated to an equivalent AHB response. If an OCP error response is received on the master interface, it will be sent as AHB OKAY response and not as AHB Error response.

The DMA controller will continue with the DMA read/write transfer as it is unaware of the OCP error response that occurred for the given read/write transfer. The bus error bit in DMA_CNTL register will never be set.

Workaround(s) In an OCP error scenario, halt the DMA and terminate the DMA transfer.

Advisory 1.21 *L3 Interconnect Clock Divisor Default Value Must be Modified Before Configuration of SDRAM Controller*

Revision(s) Affected 1.2, 1.1 and 1.0

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 scenario:

- GP device External Fast boot is used.

Workaround(s) For External Boot or External Fast Boot which are both impacted by the issue, CM_CLKSEL_CORE[1:0].CLKSEL_L3 (bit 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 1.28 *DMA: Drain_IE Reset Value*

Revision(s) Affected 1.2, 1.1 and 1.0

Details The Drain_IE bit in the DMA4_CICRi[12] register is not initialized after POR, and 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 1.29 *sDMA: Channel Is Not Disabled After a Transaction Error*

Revision(s) Affected 1.2, 1.1 and 1.0

Details During a destination synchronized transfer on the write port (or source sync with SDMA.DMA4_CCRi[25] BUFFERING_DISABLE = 1), if a transaction error is reported at the last element of the transaction, the channel is not automatically disabled by 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 1.30 *SGX Bugs Documented in Imagination Technologies™ Errata*

Revision(s) Affected 1.2, 1.1 and 1.0

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 1.31 *HS USB HOST Controller: Device Aborts the Remote Wake-Up Sequence if AM3715/03 Wakes From OFF/RET to ON in USB TLL Mode*

Revision(s) Affected 1.2, 1.1 and 1.0

Details AM3715/03 configures the TLL and the USBHOST. PRCM is programmed for the USB and TLL save and restore. USBHOST suspends the bus and a suspend interrupt is generated by the TLL. 37x is programmed to go to OFF/RET mode.

Before USBHOST is switched OFF the contents of the USBHOST are saved. Before CORE is to switched OFF/RET TLL contents are saved. 37x transitions to OFF/RET state.

External device initiates a remote wake-up. This is seen by the PRCM through daisy chain and this wakes AM3715/03. CORE reset is released, TLL restores starts. Upon the completion of the TLL restore, ALT interrupt is generated by the TLL PHY to the device. This breaks the remote wake-up protocol and as a result device aborts the remote wake-up sequence.

Workaround(s) Disable the "interrupt enable fall" register ULPI_USB_INT_EN_FALL_i[4]: IDGND_FALL; from the external device side before entering the suspend state.

Advisory 1.32 *Pending Interrupt to Video Sequencer Prevents IVA2 from Going Into Idle Mode*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

All the interrupt events either masked or un-masked should be cleared before transitioning the IVA2 domain to standby. This condition is mandatory to allow the idle transition, otherwise the IVA2 is kept active.

This issue may occur if the video sequencer is not used (that is, kept under reset or already idled) while the IVA2 idle sequence is performed. In this case, if an interrupt is propagated to the video sequencer, it will not be handled by the interrupt handler which is inactive but would avoid the IVA2 standby transition. The interrupt is not processed because the video sequencer is inactive, thus avoiding the standby transition of the whole IVA2 domain.

The following is an example of this issue: When performing an MP3 low-power use case, the video sequencer is not used, kept under reset, and only the DSP domain of the IVA2 sub-system is used. Any interrupt (even masked) to the video sequencer avoids standby transition for the entire IVA2 domain.

Workaround(s)

Any write/read to the video sequencer IRQ Register (SEQ_IRQCLR/SEQ_IRQSTATE, etc.) from the DSP or L3 interconnect resolves this issue and allows the standby transition.

See below sample code example which can be inserted in C source code:

```
// Definition of Constant Values
asm("IVA2_SEQ_BASE_ADDRESS .set 0x00090000");
asm("SEQ_IRQSTATE_OFFSET .set 0x0000004C");
asm("EFI_READ32_REQ_CMD .set 0x02");
//;; Required Code Fragment for Dummy read register in Sequencer
// push registers to be used
asm(" STW A3, *SP--");
asm(" STW A4, *SP--");
asm(" STW B4, *SP--");
asm(" SUB SP, 4, SP"); // secure 1 word for local frame generation
//;; Sequencer ReadRegister() sequence
asm(" ZERO B4");
asm(" MVKH IVA2_SEQ_BASE_ADDRESS, B4");
asm(" DINT");
asm(" ADDK SEQ_IRQSTATE_OFFSET, B4");
asm(" NOP");
asm(" EFSW.L1X B4, EFI_READ32_REQ_CMD");
asm(" NOP 9");
asm(" NOP 8");
asm(" EFRW.S1 A3");
asm(" STW A3, *B15[1]"); // if required to save the read data
asm(" NOP 2");
asm(" RINT");
// pop registers
asm(" ADD SP, 4, SP");
asm(" LDW *++SP, B4");
asm(" LDW *++SP, A4");
asm(" LDW *++SP, A3");
```

Advisory 1.33 *McSPI Can Generate a Wrong Underflow Interrupt*

Revision(s) Affected 1.2, 1.1 and 1.0

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 (that is, 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 two different chips.

Workaround(s) The following initialization sequence will solve this issue:

1. On the master side: Set MCSPI_MODULCTRL:SINGLE. Perform the following steps by doing three 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 1.34 *VENC: TV Detect AC Coupling Mode Not Supported*

Revision(s) Affected 1.2, 1.1 and 1.0

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 1.35 *Unexpected Stalling May Occur During SDMA/IDMA Accesses to DSP L2 Memory*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

Note: This exception does not apply if there are no IDMA/SDMA accesses to DSP L2 memory (both cache/SRAM and shared SRAM) during run-time.

Background Information:

The C64x+ Megamodule in the IVA2.2 subsystem has a Master Direct Memory Access (MDMA) bus interface and a Slave DMA (SDMA) bus interface (see Figure 13). The MDMA interface provides DSP access to resources outside the C64x+ Megamodule such as external DDR memory, flash memory, and On-chip Memory (OCM) RAM. The MDMA interface is typically used for C64x+ CPU/cache accesses to memory beyond the Level 2 (L2) memory level. These accesses can be in the form of cache line allocates, write-backs, and non-cacheable loads and stores to/from system memories.

The SDMA interface allows other master peripherals in the system to access DSP memories. The memories which may be accessed are Level 1 Data (L1D), Level 1 Program (L1P), L2 cache/RAM, and shared L2 RAM. Examples of masters who may access these memories are EDMA transfer controllers, the system DMA, and the MPU.

Note: The IVA2.2 video hardware accelerator accesses L2 shared RAM through the shared L2 interface (SL2IF). Accesses through the shared L2 SRAM are not susceptible to this issue.

The DSP Internal DMA (IDMA) is a DMA engine within the C64x+ Megamodule which is used to move data between internal DSP memories (L1 and L2) and/or DSP peripheral configuration bus. The IDMA engine shares resources with the SDMA interface.

The C64x+ Megamodule has a L1D cache and L2 cache both implementing write-back data caches. This means that it holds updated values for external memory as long as possible. It writes these updated values to external memory either when it needs to make room for new data, or when specifically requested to do so by the application. These write-backs are called "victims." The L1D sends its victims to L2. The caching architecture has pipelining which means multiple requests could be pending between L1, L2 and MDMA.

Additional details on the C64x+ Megamodule and its MDMA and SDMA ports can be found in the *TMS320C64X+ Megamodule Reference Guide* (literature number SPRU871).

Exception Description:

Ideally the MDMA and SDMA/IDMA paths operate independently with minimal interference. Normally MDMA accesses may stall for extended periods of time (clock cycles) due to expected system level delays (e.g. due to bandwidth limitations and/or DDR memory refreshes). However, due to the exception, there are cases where SDMA and/or IDMA accesses to L2/L1 experience an additional/unexpected stall during the normal stalls seen by the MDMA interface. For latency sensitive traffic, the SDMA stall can result in missing real time deadlines.

Note: SDMA/IDMA accesses to L1P/D will not see any unexpected stall if there are no SDMA/IDMA accesses to L2 (both cache/SRAM and shared SRAM). This is because unexpected SDMA/IDMA stalls to L1 happen only when they are pipelined behind L2 accesses which experience stalls due to the exception.

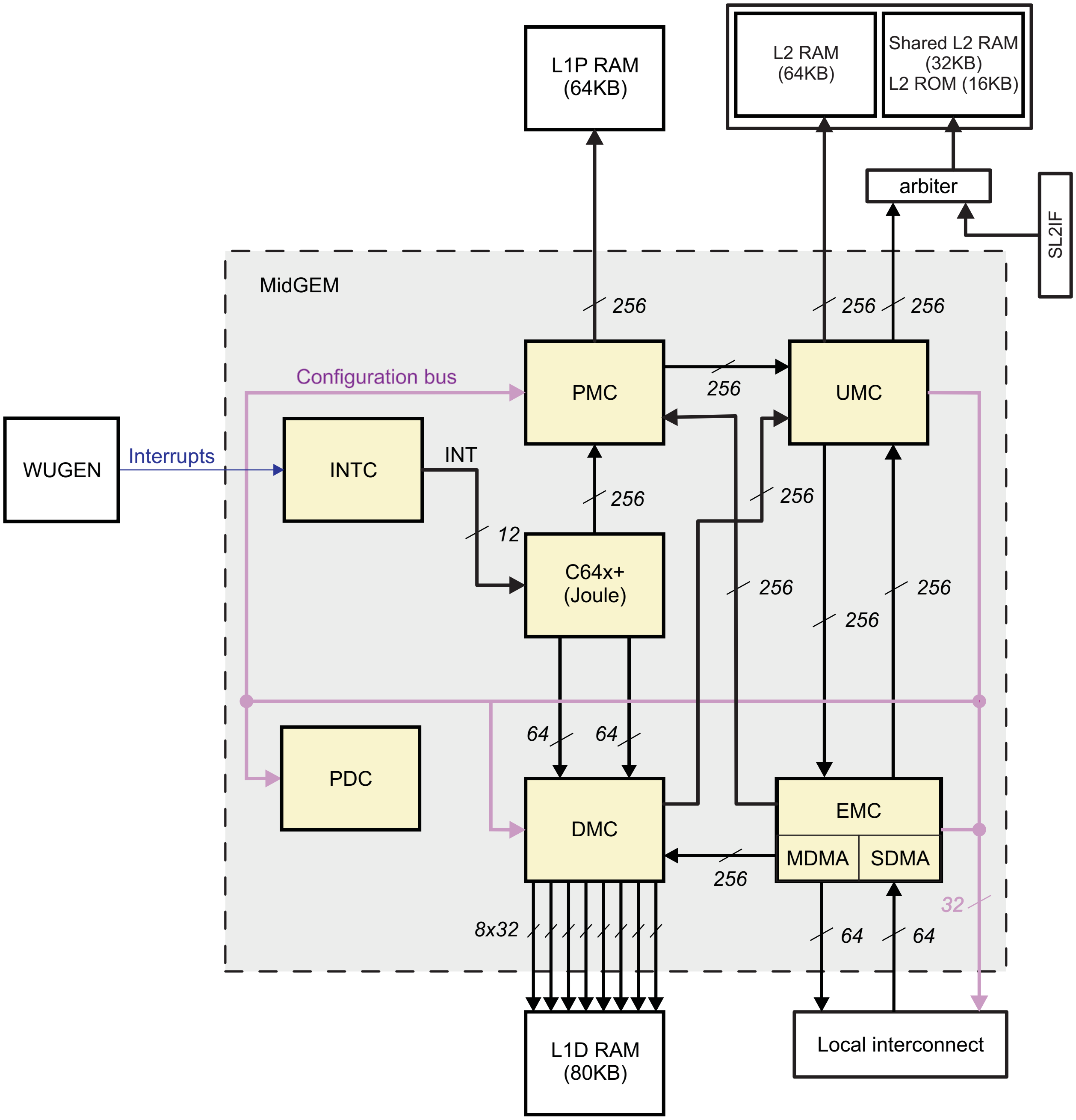


Figure 13. C64x+ Megamodule Block Diagram

In the following scenarios, SDMA/IDMA stalls may possibly occur. Note that each of these scenarios describes expected normal DSP functionality. The only issue is that the SDMA/IDMA access potentially exhibits additional unexpected stalling during each of these scenarios:

1. Bursts of writes to non-cacheable MDMA space (external DDR memory, flash memory, and OCM RAM). The DSP buffers up to 4 non-cacheable writes. When this buffer fills, SDMA/IDMA is blocked until the buffer is non-full. Thus, bursts of non-cacheable writes longer than three writes can stall SDMA/IDMA traffic.

2. Various combinations of L1 & L2 cache activity:
   (a) L1D read miss generating victim traffic to L2 (Cache or SRAM) or external memory. The SDMA/IDMA may be stalled while servicing the read miss and the victim. If the read miss also misses L2 cache, the SDMA/IDMA may be stalled until data is fetched from external memory to service the read miss.
   (b) L1D read request missing L2 (going external) while another L1D request becomes pending. The SDMA/IDMA may be stalled until the external memory access is complete.
   (c) L2 victim traffic to external memory during any pending L1D request. SDMA/IDMA may be stalled until external memory access and the pending L1D request completes.

The duration of the IDMA/SDMA stalls is highly dependent on the quantity/characteristics of the L1/L2 cache traffic and MDMA traffic present in the specific system/application and is not simple to quantify. In case # 2a, 2b and 2c from above, stalling may or may not occur depending on specific state of the cache request pipelines and target locations of the traffics. These stalling mechanisms may also interact in various ways to cause combined stalls of larger duration. Due to these factors, it is not straightforward to predict if the stalling will occur and the duration of the stalling.

The occurrence of such IDMA/SDMA stalling and/or any system impact of such stalling will be most likely in systems with very excessive context switching and/or L1/L2 cache miss/victim traffic and in systems with very heavily loaded EMIF, again making it difficult to predict occurrence and precise duration of individual stalls.

Determining if a real-time deadline miss is due to this exception:

The steps outlined below can be used to determine if the issue described here is the culprit of real-time deadline misses for existing applications. Examples of situations in which real-time deadlines may be being missed include loss of McBSP samples and low peripheral throughput.

Step 1. Determine if the transfer missing the real-time deadline is directly accessing L2 or L1D memory. If not, then this issue is not the source of the problem.

Step 2. Next, identify all SDMA transfers to/from L2 memory. An EDMA transfer to/from L2 from/to a McBSP would be an example of such a transfer. Another example would be an system DMA block transfer to/from L2.

If there are no SDMA transfers going into L2, then this issue is not the source of the problem.

Step 3. Redirect all SDMA transfers to L2 memory to other memories. This can be done several ways:

- Temporarily move all the L2 SDMA transfers to L1D SRAM.
- If not all L2 SDMA transfers can be moved to L1D memory, temporarily direct some of the transfers to OCM RAM or DDR memory and keep the rest in L1D memory. Still, there should be no L2 SDMA transfers.

If real-time deadlines are still being missed after implementing any of the options in Step 3, then the issue described here is likely not the cause of the problem. If real-time deadline misses are solved using any of the options in Step 3, then it is very likely this issue is the source of the problem. Refer to "Workaround Description" for ways to work around this issue.

Workaround(s)

Method 1

Entirely eliminate the exception by removing all SDMA/IDMA accesses to L2 SRAM. The EDMA/QDMA, IDMA, system DMA, and other chip masters cannot access L2. There are no issues with the CPU itself accessing code/data in L2. Also, this issue only pertains to SDMA accesses to L2, accesses through the shared L2 interface (SL2IF) are still allowed. Note that it is recommended to always configure the L2 cache/SRAM block as 100% cache.

Method 2

Issues such as dropped McBSP samples can be worked around by moving latency sensitive buffers outside the C64x+ Megamodule. For example, rather than placing buffers for the McBSP into L1/L2, those buffers can instead be placed in other memory such as OCM RAM.

Method 3

Employ any combination of the following as appropriate/possible to reduce the system impact of the exception:

1. Improve system tolerance on DMA side (IDMA/SDMA/MDMA):
   - (a) Understand and minimize latency critical SDMA/IDMA accesses to L2 or L1P/D.
   - (b) Directly reduce critical real-time deadlines if possible at peripheral/IO level (e.g. increase word size and/or reduce bit rates on serial ports).
   - (c) Reduce DSP MDMA latency by:
     - (i) Increase priority of DSP access to external memory such that MDMA latency of MDMA accesses causing stalls is minimized (NOTE: other masters such as system DMA may have real-time deadlines which dictate higher priority than DSP).
     - (ii) Do not perform external memory access using ready handshaking during DSP run-time (devices using ready handshaking potentially insert very excessive latency to external memory accesses).
2. Minimize offending scenarios on DSP/Caching side:
   - (a) If DSP performing non-cacheable writes is causing issue, insert non-cacheable reads every few writes to allow write buffer to drain.
   - (b) Avoid caching from slow memories such as asynchronous memory. Instead, page the data via the EDMA from the off-chip async memory to L2 SRAM or SDRAM space before accessing the data from the DSP. NOTE: paging cannot occur while real-time deadlines must be met.
   - (c) Use explicit cache commands to trigger cache write-backs during appropriate times (L1D Writeback All, L2 Writeback All). Conversely, do not use these commands at inappropriate times (when real-time deadlines must be met).
   - (d) Restructure program data and data flow to minimize the offending cache activity
     - (i) Define the read-only data as "const". The const C keyword tells the compiler that the array will not be written to. By default, such arrays get allocated to the ".const" section as opposed to BSS. With a suitable linker command file, the developer can link the .const section off-chip, while linking .bss on-chip. Because programs initialize .bss at run time, this has the added benefit of reducing the program’s initialization time and total memory image.
     - (ii) Explicitly allocate lookup tables and writeable buffers to their own sections. The #pragma DATA_SECTION(label, "section") directive tells the compiler to place a particular variable in the specified COFF section. The developer can explicitly control the layout of the program with this directive and an appropriate linker command file.
     - (iii) Avoid directly accessing data in slow memories (e.g. flash), copy at initialization time to faster memories
   - (e) Modify troublesome code
     - (i) Rewrite to use DMAs to minimize data cache writebacks. If the code accesses a large quantity of data externally, consider using DMAs to bring in the data, using double buffering and related techniques (minimizing cache write-back traffic and thus chance for occurrence of the exception).
     - (ii) Re-block the loops. In some cases, restructuring loops can increase reuse in the cache, and reduce total traffic to external memory.
     - (iii) Throttle the loops. If restructuring the code is impractical, then it becomes reasonable to slow it down. What this does is reduce the likelihood that

consecutive SDMA/IDMA blocks "stack up" in the cache request pipelines resulting in a very long stall.

Advisory 1.38 *HS USB OTG Failing Access to Registers*

Revision(s) Affected 1.2, 1.1 and 1.0

Details This issue only happens under these conditions:

- More than one endpoint is being used with at least one RX endpoint.
- As soon as one RX endpoint is used with double packet buffering enabled.
- The module's OCP master interface is completing DMA transfer.
- The UTMI/ULPI interface is receiving/transmitting data.
- The processor reads any USB register using the OCP slave interface.
- L4_ICLK = L3_ICLK / 2

When all conditions are met, the access to the register is failing: reads return 0, and writes fail.

Workaround(s) Disabling the DMA AutoClr feature solves the issue (RXCSRH[7]AUTOCLEAR = 0).

Advisory 1.39 *HS USB OTG Software Reset Is Not Fully Functional*

Revision(s) Affected 1.2, 1.1 and 1.0

Details Upon performing a software reset by setting the OTG_SYSCONFIG:SOFTRESET bit,

- The hsusb_stp pin will remain low instead of going high as expected.
- The reset command will not be sent to the PHY as it should be (the module should send 0x69 or 0x61 and nothing is actually sent).

There is no real impact on the USB link functionality except for the two items above. Only software resets through the OTG_SYSCONFIG:SOFTRESET bit field are impacted (cold or warm resets are not impacted).

Workaround(s) Use the ULPI RegAddr and ULPI RegData registers to manually send the reset command to the PHY. As soon as this is done, LINK and PHY can begin negotiating and functioning normally.

Advisory 1.40 *ROM Code: CKE PAD Is Not Set When Initializing External RAM*

Revision(s) Affected 1.2, 1.1 and 1.0

Details By default, the CKE pad is in safe mode after Power On Reset. A pull-up is connected to it, so its default level is high. The ROM Code never configures it.

In case the booting image contains a Configuration Header which configures a CHRAM section, the ROM Code can configure external RAM according to the parameters located in CHRAM section. In case the platform boots from NAND, it can allow for example the ROM Code to directly copy the booting image into external RAM. Because the ROM Code does not configure this pad, the CKE signal does not reach the external RAM.

There is no functional impact at boot time. There is no limitation at run-time. The CHRAM section defines the setting for register SDRC_POWER at physical address 0x6D00 0070. The software designer must ensure that any special feature involving CKE signaling is OFF (at boot time only).

- SDRC_POWER_REG must turn OFF all Auto_cnt feature (CLKCTRL field = 0).
- SDRC_POWER_REG must disable the Power Down mode of the target memory, via CKE (PWDENA field = 0).

Workaround(s) There is no workaround for this issue. The functionality "configuration & boot from external RAM" remains operational following the basic rules indicated above. After booting, the application can properly configure the CKE pad and configure a more optimized refresh policy.

Advisory 1.41 *ROM Code: SDRC_POWER Register Is Initialized With Hardcoded Value*

Revision(s) Affected 1.2, 1.1 and 1.0

Details In case the booting image contains a Configuration Header which contains itself a CHRAM section, the ROM Code can configure external RAM according to the parameters located in CHRAM section. In case the platform boots from NAND, it can allow for example the ROM Code to directly copy the booting image into external RAM. The CHRAM section contains all the values to initialize the SDRC module. When setting the SDRC_POWER_REG register, the ROM Code does not read the value contained in the Configuration header but applies instead a hard coded value set to 0xC1.

This value configures the SDRC as follows:

- High power/High Bandwidth Mode (HPHB)
- Power down mode feature disabled
- Enable clock
- No auto-count feature turned on
- Enter self-refresh when hardware idle request
- Enter self-refresh when a warm reset is applied
- Auto-count = 0

There is no chance to configure SDRC_POWER register with a different value than the one described above.

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

Advisory 1.42 *ROM Code: When sys-clock = 16.8MHz, Then Peripheral Booting via UART3 Is Not Operational*

Revision(s) Affected 1.2, 1.1 and 1.0

Details After Power on Reset, the following occur:

- ROM Code detects the system clock.
- ROM Code configures the CORE DPLL and PER DPLL.
- The DPLL configuration depends on the system clock.
- For a 16.8 MHz system clock, the PER DPLL is wrongly configured. DPLL locks to 828 MHz instead of 864 MHz.
- For all other clocks, the configuration is correct.

The baud rate of UART3 depends on DPLL PER frequency. The UART baud rate programmed by ROM Code is not 115200 bauds as required by UART Host. Thus the data sent by the Host is not understood by ROM Code. Therefore Peripheral Booting via UART3 is not operational at sys-clock 16.8MHz. There is no possible workaround.

USB does not used this clock.

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

Advisory 1.43 *I2C: I2C_STAT:XUDF Is Not Functional in Slave Transmitter Mode*

Revision(s) Affected 1.2, 1.1 and 1.0

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 1.44 *EHCI Controller: Issue in Suspend Resume Protocol*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details: This issue concerns HSUSBHOST controller of AM3715/03 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 RemoteWake-Up (device initiating the Resume).

The HSUSBHOST exits the resume/wake-up 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 the PHY telling the controller that the bus has switched to High Speed, and during the time the host is about to send an SOF packet.

TLL mode is impacted.

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

Two workarounds can be listed for this issue (see the flow diagrams and software programming sequences below):

- Workaround 1:
  - Advantage: No limitation on the number of HS port usage.
  - Disadvantage: This applies to the Suspend/Resume case only; the Suspend/RemoteWakeup case is not supported. This is because we do not know where within the PortSOF counter the hardware will set the Resume bit.
- Workaround 2:
  - Advantage: This workaround applies to both the Suspend/Resume and Suspend/RemoteWakeup cases.
  - Disadvantage: There is a limitation to have only one HS port used (no limitation on the number of FS ports used, since the OHCI controller is not concerned). This is because EHCI controller has to be stopped by software before clearing portSC.FPR.

Workaround 1 detailed description:

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 the ForceResume bit is done at the beginning of the PortSOF counter.

Workaround 1 software implementation:

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

10. Keep polling the FRINDEX register to make sure that the register value has incremented from the value read in Step (9)
11. Clear the PORTSC.FPR bit

- Suspend/RemoteWake-Up case software implementation: NOT APPLICABLE

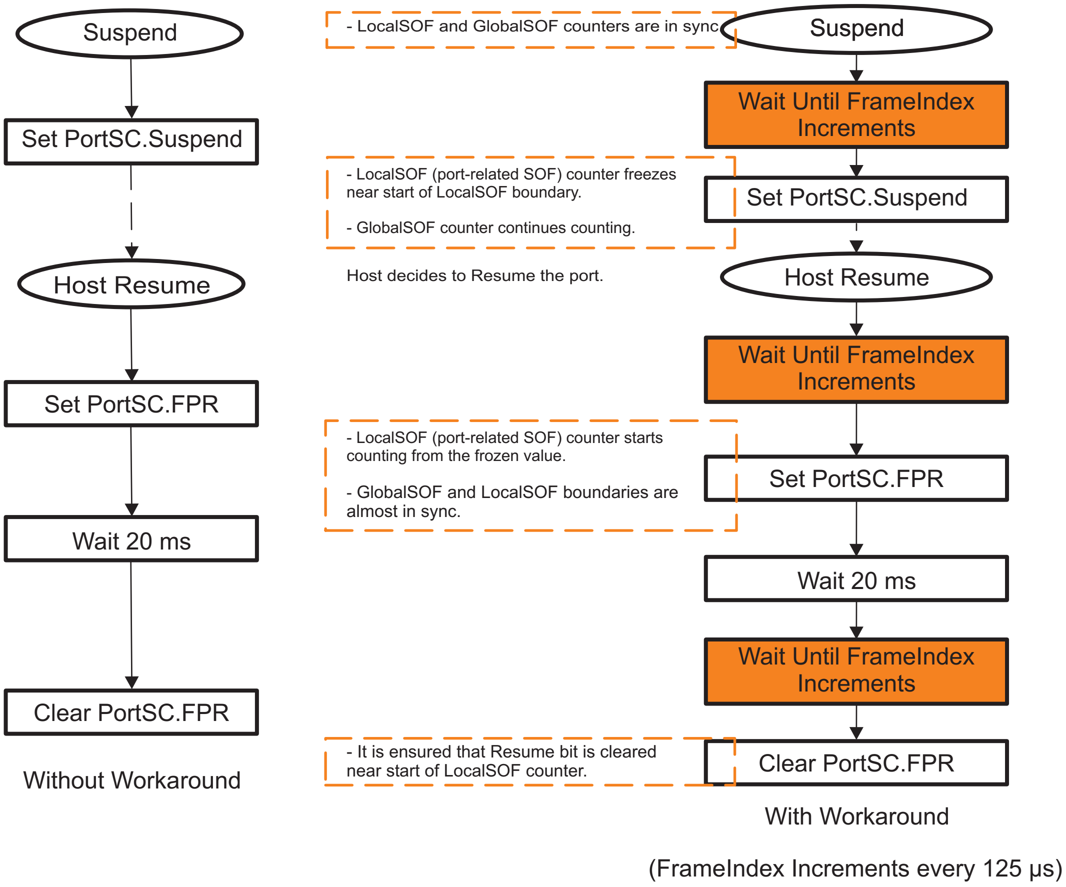

Suspend
Set PortSC.Suspend
Host Resume
Set PortSC.FPR
Wait 20 ms
Clear PortSC.FPR
Without Workaround
- LocalSOF and GlobalSOF counters are in sync.
Suspend
Wait Until FrameIndex Increments
- LocalSOF (port-related SOF) counter freezes near start of LocalSOF boundary.
- GlobalSOF counter continues counting.
Set PortSC.Suspend
Host decides to Resume the port.
Host Resume
Wait Until FrameIndex Increments
- LocalSOF (port-related SOF) counter starts counting from the frozen value.
- GlobalSOF and LocalSOF boundaries are almost in sync.
Set PortSC.FPR
Wait 20 ms
Wait Until FrameIndex Increments
- It is ensured that Resume bit is cleared near start of LocalSOF counter.
Clear PortSC.FPR
With Workaround
(FrameIndex Increments every 125 µs)

Figure 14. Workaround 1 Implementation Diagram

Workaround 2 detailed description:

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 the Resume bit.

Workaround 2 software implementation:

- Suspend/Resume case software implementation:
  1. Set the PORTSC.suspend bit
  2. Wait for the required suspend time
  3. When software is ready to issue resume, set the 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 hardware
  7. Clear the PORTSC.FPR bit
  8. Set the USBCMD.RS bit to 1
- Suspend/RemoteWake-Up case software implementation:
  1. Set the PORTSC.suspend bit
  2. Wait for the required suspend time
  3. Wake-Up event happens and software driver handles the wake-up 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 the PORTSC.FPR bit
8. Set the USBCMD.RS bit to 1

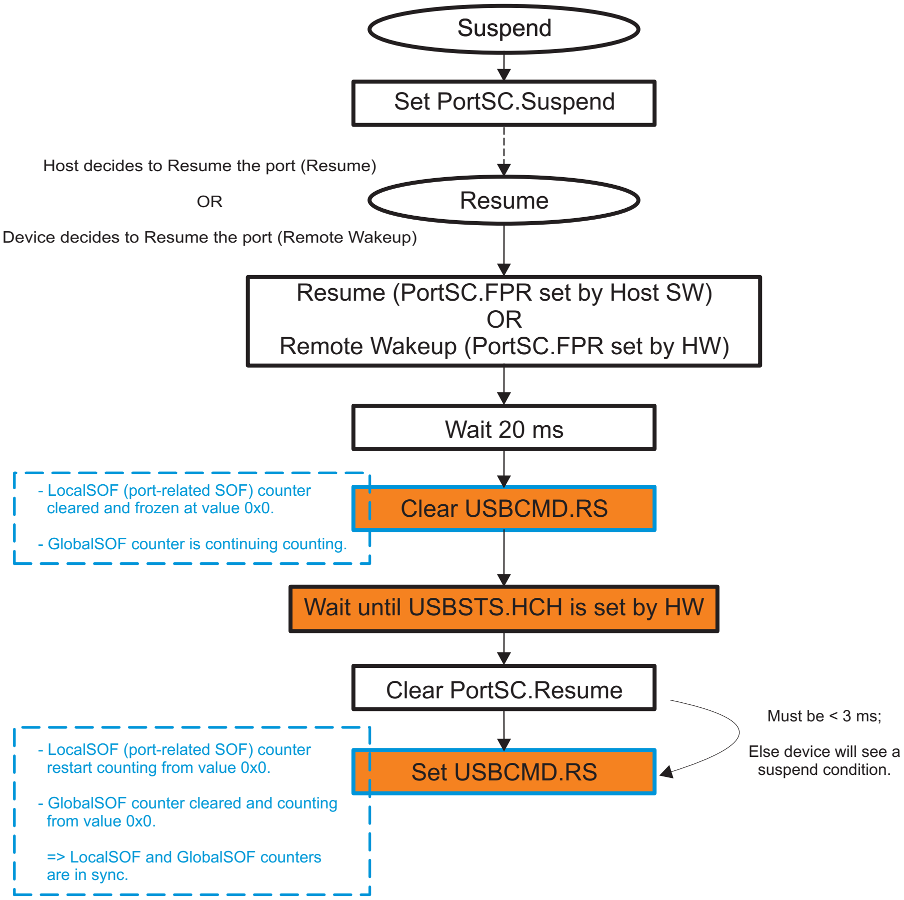

Suspend
Set PortSC.Suspend
Host decides to Resume the port (Resume)
OR
Device decides to Resume the port (Remote Wakeup)
Resume
Resume (PortSC.FPR set by Host SW)
OR
Remote Wakeup (PortSC.FPR set by HW)
Wait 20 ms
- LocalSOF (port-related SOF) counter cleared and frozen at value 0x0.
- GlobalSOF counter is continuing counting.
Clear USBCMD.RS
Wait until USBSTS.HCH is set by HW
Clear PortSC.Resume
Must be < 3 ms;
Else device will see a suspend condition.
- LocalSOF (port-related SOF) counter restart counting from value 0x0.
- GlobalSOF counter cleared and counting from value 0x0.
=> LocalSOF and GlobalSOF counters are in sync.
Set USBCMD.RS

Figure 15. Workaround 2 Implementation Diagram

Advisory 1.45 *GPIO Pad Spurious Transition (Glitch/Spike) During Wake Up Entering or Exiting System OFF Mode*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details: When the device is entering or exiting the system OFF mode, a spurious transition (approximately one nanosecond duration) may occur on the pads internally muxed to a GPIO. This is valid for all GPIO modules 1 to 6 and is linked to several parameters, such as the use of the padconf OFF override feature, the use of SW or HW supervised power domain transitions, the GPIO module used, and the GPIO pin used. This issue is due to a combination of internal races between the signals controlling the pad and the use of non glitch free multiplexers on the pin-muxing and padconf OFF override logic.

Workaround(s): For the use of the padconf OFF override feature for all GPIO modules 1 to 6:

- Do not configure the OFF override to the same output drive as the GPIO. This can create glitches. (for example, do not configure the padconf OFF override to output drive 1 if the GPIO is already set to output drive 1).
- If you need to configure the padconf OFF override to perform a different function than the one the GPIO is set to (for example, setting it to output drive 0 during OFF when the GPIO is set as an input):
  - If the GPIO is driving 1 (resp. 0) and the OFF override is set to drive 0 (resp. 1), then there is no problem, because the voltage level will transition
  - If the GPIO is driving 1 or 0 and the OFF override is set to the input mode (whatever the pull direction), there can be a glitch, there is no workaround to avoid it.
  - If the GPIO is in input pull up (resp. down) mode, and the OFF override is set to drive 1 (resp. 0), there can be a glitch, there is no workaround to avoid it.
  - If the GPIO is in input pull up (resp. down) mode, and the OFF override is set to drive 0 (resp. 1), then there is no problem, because the voltage level will transition

For the OFF mode entry-exit glitch/spikes not related to the padconf OFF override feature: This only applies to GPIO modules 2 to 6. GPIO module 1 is not impacted. There is only one workaround that is 100% safe and that also takes into account :

- Use PER automatic HW transitions (no SW controlled transitions) and enable a wakeup dependency between the PER and WU domains
- If the GPIO drives 1 (resp. 0), first enable the pull up (resp. down), then switch the GPIO to input mode. There is nothing to do if the GPIO was an input.
- Store the GPIO config before going to OFF
- Upon wakeup, the PER is woken up automatically and the GPIO modules are reset to input mode. If the GPIO was in output mode, the pull will now drive the line.
- Restore the GPIO config. If the GPIO was in output mode before being set to OFF, then the pull is released and the GPIO drives the line.

NOTE 1: For low input impedance devices connected to GPIO modules 2 to 6:

The workaround above implies that the level on the line is maintained by a pull for some time and not by a strong drive. The device pull is a mirror current that has a nominal value of 100 µA (18K Ω) with a dispersion across PVT conditions: min 250µA (7.2K Ω), max 50 µA (36K Ω). If the IC connected to the GPIO has a high impedance input stage (>>36K Ω, which is always guaranteed for CMOS logic), then there is no problem. If the IC connected to the GPIO has a low impedance input stage (in the range of the internal pull 7.2-36K Ω), then the voltage between the device and the IC may drop below 1.8V or rise above 0V.

- Action1: It is your responsibility to ensure that no component used on the board has input impedance of 50K Ω or lower
- Action2: If you are using risky components (see action1), an alternative option should be identified or a specific electrical study is needed to validate the option (contact

your TI representative)

NOTE 2: The use of a pull for momentarily maintaining the level on the line is not something new introduced by this workaround. In any case, with or without any workaround, because of the GPIO modules 2 to 6 being reset to input mode when coming back from OFF, there is going to be one moment when the level on the line is maintained by the pull.

A global workaround would be:

- Do not use the padconf OFF override feature for GPIO modules 1 to 6
- Apply the entire workaround to GPIO modules 2 to 6

Advisory 1.46 *Voltage Processor TRANXDONE Interrupt Occurs Too Early*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details: VPx_TRANXDONE_ST interrupt is generated too early (immediately in worst case) if the HW counter (in Voltage Processor) used to wait for voltage ramp duration is incorrect. The HW counter could be incorrect in the following scenarios:

- The voltage update is done through Voltage Processor force-update command.
- Smart-Reflex was previously enabled.

This limitation impacts the low-to-high OPP transitions as software waits for TRANXDONE interrupt generation before changing the frequencies. The risk is that high frequency is configured while high voltage is not yet fully ramped. High-to-low OPP transitions are not functionally impacted by this HW limitation.

Workaround(s): Upon Low-to-High OPP transitions, software shall add a delay (software wait loop) when receiving the VPx_TRANXDONE_ST interrupt and before increasing the frequencies.

This delay needs to be calculated according to:

- PMIC slew rate for voltage ramp-up
- Delta between ‘low’ voltage and ‘high’ voltage
- Delay already introduced by software execution (ISR, call to set_module_frequency...)

Advisory 1.47 *HS USB OTG: OTG_SYSCONFIG:AUTOIDLE Bit Is Not Reset Correctly*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details: Upon any source of HW reset (at boot time or when coming back from OFF mode for example), the reset value of the OTG_SYSCONFIG:AUTOIDLE bit is 1 (autoidle feature enabled). The behavior should be the same when applying a software reset to the module through the OTG_SYSCONFIG:SOFTRESET bit. Because of an implementation bug, and because the AUTOIDLE and SOFTRESET bits are in the same OTG_SYSCONFIG register, the value defined for the AUTOIDLE bit while performing the software reset gets over-written into it after software reset is over.

For example: if the AUTOIDLE bit is 0 and a software reset is performed. AUTOIDLE bit should be reset to 1.

- If the software reset is performed by writing 0x2 in OTG_SYSCONFIG, AUTOIDLE will be un-expectedly reset to 0.
- If the software reset is performed by writing 0x3 in OTG_SYSCONFIG, AUTOIDLE will be reset to 1 as expected.

Workaround(s): There is no workaround. When performing the software reset, just write the AUTOIDLE bit at the value you expect it to be after reset.

Advisory 1.48 *IVA2 Does Not Wake-Up After It Goes to IDLE While DMA Request Is Still Asserted*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details: IVA2 may go to Retention during a brief period between "a completion of an EDMA write transfer" and "a deassertion of DMA request (MCBSP_DMA for example)" . The deassertion of DMA request could be delayed by making consecutive accesses (Write or Read) to the low-speed peripherals (e.g., consecutive non-posted writes to GPTimerx from DSP or from MPU if an EDMA write transfer and this GPTimerx use same L4 thread, i.e. McBSP1 and GPTimer1) while EDMA is transferring some data.

The DMA request is then de-asserted while the IVA2 has gone to retention but WUGEN memorizes that the DMA request was asserted. When a new DMA is asserted while IVA is in IDLE state then the WUGEN can't see it because it considers that the DMA is still asserted so it doesn't serve it and doesn't wake-up the IVA2.

Workaround(s): **Workaround 1**: Add a dummy Read access to the same peripheral for each EDMA write transfer for one EDMA request to the peripheral by using EDMA chaining function. When the deassertion of the EDMA request is delayed by other accesses (Write or Read) to the low-speed peripheral, WUGEN sets IVA SS in Standby State without waiting of the actual completion of the EDMA transfer.

If a dummy Read access to the same peripheral is added after each EDMA write transfer to the peripheral then WUGEN sets IVA SS in Standby State after the dummy Read access completion. At this time, the EDMA write transfer should be finished and also the EDMA request is deasserted absolutely. A race between IVA goes to idle and the deassertion of the EDMA request can be avoided.

Note: A dummy Read access to the peripheral should be added for each EDMA write transfer for one EDMA request. Therefore, there is possibility to decrease the performance.

Workaround 2: Enable DMATRUECOMPEN, ITCINTEN and TCINTEN. All EDMA write request is posted-write in default. Enabling DMATRUECOMPEN changes EDMA write request into a non-posted write. The write request which is affected by DMATRUECOMPEN can be enabled/disabled by using ITCINTEN and TCINTEN That is, if ITCINTEN=1 and DMATRUECOMPEN=1 then intermediate write events can be non-posted write requests. And also if TCINTEN=1 and DMATRUECOMPEN=1 then final write event can be a non-posted write request.

Therefore, even the service for EDMA write transfer is delayed by other accesses (Write or Read) to the low-speed peripheral, WUGEN sets IVA SS in Standby State with waiting the completion of this EDMA write transfer by enabling DMATRUECOMPEN, ITCINTEN and TCINTEN.

Note: TCC interrupt is common for both intermediate requests (enabled with ITCINTEN) and final request (enabled with TCINTEN). Therefore, there is no way to get only the interrupt for final request when both ITCENTEN and TCINTEN are enabled.

To generate the interrupt only with final event, additional use of PaRAMs is required. The method depends on the transfer synchronization dimention (that is, ASync or ABsync):

- Case1: In case of Async transfer and Bcnt !=1 or Ccnt !=1, and would like to get the interrupt only with the final event, 3 or more linked PaRAMs are needed.
- Case2: In case of ABsync transfer and Ccnt !=1, and would like to get the interrupt only with the final event, 2 or more linked PaRAM are needed.
- Case3: In other case (that is, Async transfer and Bcnt =1, Async transfer and Ccnt =1, or ABsync transfer and Ccnt =1), there is no need to care about the PaRAM setting to get the interrupt of final event.

Advisory 1.49 *Software Reset Done While ISP Processing Is Ongoing Can Cause OCP Protocol Violations*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details: The ISP applies a software reset (ISP_SYSCONFIG[1]:SOFT_RESET or CCP2_SYSCONFIG[1] SOFT_RESET) immediately without checking if there is ongoing OCP traffic. Incomplete OCP transactions may be generated when the reset occurs in the middle of an L3 OCP burst. Incomplete OCP transactions stalls the L3 interconnect or cause a timeout condition.

Workaround(s): Software needs to first stop all ISP traffic before doing a software reset. Idea is to block the OCP interface to avoid transfer are on going when reset is applied. To do that, MMU mapping error mechanism is used. MMU is programmed to generate an IRQ when an address which is unmapped in MMU table hit the MMU. Then all MMU tables are trashed so that IRQ will happen on the next OCP access (and access will block the OCP port). Once the IRQ happen, no transfer is on going by construction and reset can be applied.

Following sequence should be applied:

1. Configure the MMU in manual mode so that it requests translation entries from the CPU by triggering an IRQ.
2. When an access arrives, the MMU will stall the OCP port on a clean OCP boundary.
3. The CPU doesn't provide the translation entry: that ensure that the OCP traffic has been stopped.
4. Wait 1000 cycles to make sure all responses have been returned.
5. Issue a software reset.

Advisory 1.51 *Accesses to DDR Stall in SDRC After a Warm-Reset*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details: In some cases, user is not able to access DDR memory after warm-reset.

This situation occurs while the warm-reset happens during a read access to DDR memory. In that particular condition, DDR memory does not respond to a corrupted read command due to the warm reset occurrence but SDRC is waiting for read completion.

SDRC is not sensitive to the warm reset, but the interconnect is reset on the fly, thus causing a misalignment between SDRC logic, interconnect logic, and DDR memory state.

Root cause description: A corrupted read transaction is issued to a closed row: (address0, bank0) instead of the expected read access, violating protocol.

Failure signature: Once the failure occurs and system has restarted, memory content is not accessible. SDRC registers can be accessed successfully, until 1st access to memory location is performed. After 1st access to memory is done, SDRC is stuck.

Workaround(s): Steps to perform before a software reset is triggered, if user needs to generate a software reset and keep DDR memory content:

1. Set SDRC_POWER[SRFRONIDLEREQ]=1 //enable self-refresh on idle request
2. Set PRCM CM_ICLKEN1_CORE[SDRC]=0 //put SDRC in idle
3. Wait until PRCM CM_IDLEST1_CORE[SDRC]=1 //wait until SDRC goes to idle
4. Generate software reset

Steps to perform after warm reset occurs:

if HW warm reset is the source, apply below steps before any accesses to SDRAM

1. Reset SMS and SDRC
   (a) set SMS_SYSCONFIG[1].SOFTRESET=0x1, wait until SMS_SYSSTATUS[0].RESETDONE = 0x1
   (b) set SDRC_SYSCONFIG[1].SOFTRESET=0x1, wait until SDRC_SYSSTATUS[0].RESETDONE = 0x1
2. Re-initialize SMS, SDRC and memory (if software warm reset is the source, SDRAM can be accessed reliably with no additional operation since user ensure above sequence before applying the warm reset).

Note: DDR memory content is lost upon HW warm-reset (WDT, secure violation, …). SDRC_POWER[SRFRONRESET] value does not matter.

Advisory 1.52 *Standard OTG Compliance Electrical Tests for HOST Mode Will Fail*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details: Below is the sequence performed to observe the issue:

- Perform Enumeration as a HOST.
- Perform USB Reset, Read the device descriptor, check for OTG defined specific test-VID/PID.
- Set testmode according the found PID (Program the testmode register in the USBOTGHS.)

Now the DP-DM data lines are not in the expected states.Hence programming of TestMode J results in the Data-Lines permanently toggle between J and K state.

Workaround(s): Program the PHY OPMODE to 10 using the ulpi extended register access before programming the TESTMODE registers in HOST Mode.

Advisory 1.54 — *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 is using a wrong polynomial, 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 prevents having correct error location.

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

Advisory 1.56 — *HS USB Host: ECHI and OHCI Controllers Cannot Work Concurrently*

Revision(s) Affected: 1.0

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 software arbitration scheme can be implemented.

Advisory 1.57 *MMC OCP Clock Not Gated When Thermal Sensor Is Used*

Revision(s) Affected: 1.2, 1.1 and 1.0

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 overconsumption (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 overconsumption due to OCP logic being un-necessarilly clocked.

Advisory 1.58 *Context Save Operation Randomly Failing for CONTROL_PAD_CONF_ETK_D14*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details: Context save operation saves the pad configuration register contents to the scratch pad memory, that will be used at the context restore operation after OFF mode exit. Sometimes randomly after context save operation, CONTROL_PAD_CONF_ETK_D14 register is not saved into scratch pad memory. This register contains pad configuration information for 2 pads: etk_d14 and etk_d15. This is impacting only CONTROL_PAD_CONF_ETK_D14, all other padconf registers are saved and restored as expected.

The failure signature is this register not saved: the scratch pad memory keep value saved during previous OFF transition. It is not a register content corruption. When waking-up from OFF mode, CONTROL_PAD_CONF_ETK_D14 is restored with old value (or non-initialized RAM value for first time).

CaseA: Issue occurs randomly when CORE_L4_ICLK/4 (default) (CONTROL_PADCONF_OFF[2]. WKUPCTRLCLOCKDIV=0, default). Caused by the polling for context save completion. Issue does not occur if a delay is added just before SAVEDONE bit polling.

CaseB: Issue occurs always When CORE_L4_ICLK/2 (CONTROL_PADCONF_OFF[2]. WKUPCTRLCLOCKDIV= 1). No workaround available.

Root cause description:

Case A: Failure occurs when OCP port is accessed for context save operation at same time as last save access of the SAR mechanism is performed. The OCP access is delaying the completion of the save of CONTROL_PAD_CONF_ETK_D14 register. In that case SAR never ended and SAVEDONE bit is wrongly returned.

Case B: Wake-up control clock is CORE_L4_ICLK/2, SAR is wrongly completed before saving last context.

For caseA and caseB, result is current CONTROL_PAD_CONF_ETK_D14 is not saved in scratchpad @0x480025F8 and previous value is kept in scratchpad.

Workaround(s): Wake-up control clock = CORE_L4_ICLK/4 AND Do not access any SCM registers before context save completed following recomended sequence below.

1. Ts = 2*(1/core_l4_iclk_freq)* X*200 (X=4 when CONTROL_PADCONF_OFF[2]. WKUPCTRLCLOCKDIV= 0)
   - Example: 21.2us for 83MHz CORE_L4_ICLK
2. Enable padconf save operation: CONTROL_PADCONF_OFF[1]:STARTSAVE='1'
3. Wait for Ts Delay + 10% (additional step)
4. Poll SAVEDONEBIT='1'
5. Read padconf_OFF save for etk_d14/15 @0x480028DC is equal to CONTROL_PADCONF_ETK_D14 @0x480025F8
   (a) If padconf_OFF save equal CONTROL_PADCONF_ETK_D14 then Sequence is completed successfully.
   (b) If padconf_OFF save NOT equal to CONTROL_PADCONF_ETK_D14 then Clear SAVEDONE bit and Go back to step 1 (restart a new save).

Advisory 1.59 *I2C4 Does Not Meet I2C Standard AC Timing in FS Mode*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details: I2C4_SCL low period is fixed by hardware then cannot be modified by software. Due to IO cell influence, I2C4_SCL AC timing is shorter than expected. As a result the standard AC timing (SCL minimum low period) in FS mode is not met. Please see the DM3730/25 Data Manual for exact I2C4 AC timings.

Workaround(s): There is no workaround for I2C4.

I2C4 is dedicated for SmartReflex and expected to connect with Power IC. Design review was done and concluded that there is no problem when the DM3730/25 device is interfaced with TPS65950.

Advisory 1.60 *I2C1 to 3 SCL Low Period Is Shorter in FS Mode*

Revision(s) Affected: 1.2, 1.1 and 1.0

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 1.61 *Missed Dependency With McBSP External Clock Prevents Transition to OSWR*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details: In case of PER domain, due to a missed dependency, if the McBSP external clock is the only enabled permanent clock of PER PD, then CORE PD is prevented from transitioning to idle and therefore into OSWR.

Such issue disappears if at least one other permanent clock of PER PD is enabled.

Note: A permanent clock is a functional clock that can stay active while the corresponding entity managing it, CM, can go into idle. That is, a permanent clock can stay running while CM internal FSMs are clock-gated and even when CORE domain goes to retention.

For WD reset and secure violation reset (SSM) PRM_VOLTSETUP2[15:0]:OFFMODESETUPTIME is ignored and formula is: RSTTIME1*32KHz.

Workaround(s): Enable a permanent clock in PER domain when McBSP external clock is used.

To minimize power consumption impact, a 32KHz clock should be selected and enabled (fclken = ‘1’) without enabling the module (iclken = ‘0’). In such way, only small 32KHz clock tree contributes to power consumption increasing.

Advisory 1.62 *MPU Cannot Exit from Standby*

Revision(s) Affected: 1.2, 1.1 and 1.0

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 1.63 *sDMA FIFO Draining Does Not Finish*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details: There is an issue when sDMA channel is disabled on the fly, sDMA enters standby even through FIFO Drain is still in progress. software workaround 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 1.64 *HSUSB Interoperability Issue With SMSC USB3320 PHY*

Revision(s) Affected: 1.0

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 Device HOST controller exit from suspend mode (while it is expected to keep in low power mode).

Device 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 for this issue.

Advisory 1.65 *IVA2 Does Not Wake-Up After It Goes to IDLE While an Interrupt Line Is Not Cleared*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details: IVA is allowed to transition to sleep as soon as the last pending IRQ was handled and the clear command was sent. If latency is added to clear the interrupt source due to system activity, then IVA2 will transition to sleep before the interrupt source is cleared, meaning that the interrupt line is not de-asserted when IVA goes to sleep but will be released later on.

In that particular condition (interrupt line active when IVA goes to sleep), then this interrupt line is no longer able to generate a WU event to the IVA. The IVA will ignore the activity on that line until it is woken-up by another source.

Workaround(s): Whenever IVA clears interrupt, IVA needs to read back the same register. This ensures the interrupt is always cleared before IDLE entry. This workaround is implemented in TI DSP Bridge for GP-Timer and mailbox interrupt. Similar sequence needs to be applied by user in the ISR to all other possible interrupt sources.

Advisory 1.66 *IVA2 SEQ (ARM9) JTAG Failures With a Free Running Clock Emulator*

Revision(s) Affected 1.0

Details JTAG on IVA2 SEQ is unreliable if the IVA2 SEQ functional clock is less than 8 times the select JTAG frequency.

There is no impact to normal device operation. In emulation mode certain JTAG clock/IVA2 SEQ functional clock combinations do not work.

Workaround(s) Reduce the JTAG clock frequency [slower emulation speed] or increase the IVA2 SEQ functional clock frequency before the connection or use REVD XDS560 (in adaptive mode),or LAUTERBACH (in adaptive mode).

Advisory 1.67 *Isolation Issue in OFF Mode Impacting the CORE Power Domain*

Revision(s) Affected 1.0

Details

There is excessive current leakage in core-voltage domain in OFF mode. During OFF mode three HSDIVIDER block instances in DPLL overconsume 10uA each. Overall, the total overconsumption reach 30uA.

No functional issue, impacts off-mode leakage extra consumption 30uA.

Workaround(s)

There is no workaround for this issue.

Advisory 1.68 *Safe Mode Does Not Work on GPMC_A11 Pad*

Revision(s) Affected 1.0

Details

The safe mode selected in the MUXMODE field of the CONTROL.CONTROL_PADCONF register (muxmode= 7) is not supported on GPMC_A11 pad.

No functional issue, but it is impacting the off-mode leakage by an extra consumption of 270uA.

Workaround(s)

Software need to set GPMC_A11 muxmode to 1 (meaning mode 1), instead of 7 (safe mode) to avoid the leakage.

Advisory 1.69 *GPMC_A11 Address Bit Does Not Behave as Expected (256MB NOR Flash Addressability)*

Revision(s) Affected: 1.0

Details

The GPMC_A11 address bit, which is only used to extend Nor flash access area, is not functional, so the Maximum address space is 256Mb of Nor flash memory. It is not possible to access upper 256Mb of Nor flash memory.

Workaround(s)

There is no workaround for this issue.

Advisory 1.71 *DM37x Device May Wake Up From Low Power Modes if IO Wake-Up and Pull Down Are Enabled*

Revision(s) Affected: 1.0

Details

Spurious wake up events may happen when PAD is in weak Pull down state (using PIPD) and VDD2 is off. The DM3730/25 device may wake up from off mode and standby when VDD2 is OFF, if pull down and wake-up capabilities are enabled.

This issue is present on modules having IO cells with configurable wake-up capabiliy and Pull up / Pull down control setting avaliable by the MUXMODE field in CONTROL.CONTROL_PADCONF_X.

A spurious wake-up events could be generated and then causing additional software overhead.

Impacted pins list:

UART: uart1_tx, uart1_rx, uart1_cts, uart1_rts, uart2_cts, uart2_rts, uart2_tx, uart2_rx, uart3_cts_rctx, uart3_rts_sd, uart3_rx_irrx, uart3_tx_irtx

SYSTEM: sys_clkout2,sys_nirq

HDQ: hdq_sio

GPMC: gpmc_a1, gpmc_a2, gpmc_a3, gpmc_a4, gpmc_a5, gpmc_a6, gpmc_a7, gpmc_a8, gpmc_a9, gpmc_a10, gpmc_d8, gpmc_d9, gpmc_d10, gpmc_d11, gpmc_d12, gpmc_d13, gpmc_d14, gpmc_d15 , gpmc_ncs1, gpmc_ncs2, gpmc_ncs3, gpmc_ncs4, gpmc_ncs5, gpmc_ncs6, gpmc_ncs7, gpmc_clk, gpmc_nbe0_cle, gpmc_nbe1, gpmc_nwp, gpmc_wait1, gpmc_wait2, gpmc_wait3

DSS: dss_pclk, dss_hsync , dss_vsync, dss_acbias, dss_data6, dss_data7, dss_data8, dss_data9, dss_data10, dss_data11, dss_data12, dss_data13, dss_data14, dss_data15, dss_data16, dss_data17 , dss_data18, dss_data19, dss_data20, dss_data21, dss_data22, dss_data23

CAMERA: cam_d2, cam_d3, cam_d4, cam_d5, cam_d10, cam_d11,cam_xclka, cam_pclk, cam_fld, cam_xclkb, cam_wen, cam_strobe

BOOT: sys_boot0, sys_boot1, sys_boot2, sys_boot3, sys_boot4, sys_boot5, sys_boot6

HS USB OTG: hsusb0_clk, hsusb0_stp, hsusb0_dir, hsusb0_nxt, hsusb0_data0, hsusb0_data1, hsusb0_data2, hsusb0_data3, hsusb0_data4 , hsusb0_data5, hsusb0_data6, hsusb0_data7

SD/MMC: sdmmc2_clk, sdmmc2_cmd, sdmmc2_dat0, sdmmc2_dat1, sdmmc2_dat2, sdmmc2_dat3, sdmmc2_dat7, sdmmc2_dat6, sdmmc2_dat5, sdmmc2_dat4

McBSP: mcbsp1_clkr, mcbsp1_fsr, mcbsp_clks, mcbsp1_dx, mcbsp1_dr, mcbsp1_fsx, mcbsp1_clkx, mcbsp2_fsx, mcbsp2_clkx, mcbsp2_dr, mcbsp2_dx, mcbsp3_dx, mcbsp3_dr, mcbsp3_clkx , mcbsp3_fsx, mcbsp4_clkx, mcbsp4_dr, mcbsp4_dx, mcbsp4_fsx

McSPI: mcspi1_clk, mcspi1_simo, mcspi1_somi, mcspi1_cs0, mcspi1_cs1, mcspi1_cs2, mcspi1_cs3, mcspi2_clk, mcspi2_simo, mcspi2_somi, mcspi2_cs0, mcspi2_cs1

TRACE: etk_clk, etk_ctl, etk_d0, etk_d1, etk_d2, etk_d3, etk_d4, etk_d5, etk_d6, etk_d7, etk_d8, etk_d9, etk_d10, etk_d11, etk_d12, etk_d13, etk_d14 , etk_d15

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

Advisory 1.72 *Excessive Leakage on BRGC Memories in Cortex A8 When Periphery Is Turned Off*

Revision(s) Affected: 1.0

Details The leakage is in the output stage of BRGC memories; it occurs when BRGC periphery is turned OFF. Because of the DM3730/25 power configuration, the leakage occurs when MPU power domain is switched OFF or OSWR (Open SWitch Retention), and VDD1 supply is ON.

There is no impact to functionality in any mode of operation.

The Leakage estimation at 1.1v is included between 6mA and 35uA depending of memory contents.

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

Advisory 1.73 *JTAG Pins With bq18jtagfbpbnopm_ssbhv IO Have ESD (Electro Static Discharge) Weakness in Input Buffer*

Revision(s) Affected: 1.0

Details ESD test using HBM (Human Body Model) to characterize the pad IC robustness shows weakness on AA10 and AA20 JTAG pins.

TLP (Transmission Line Pulse) test shows circuit failure at 150mA and 7V, on the AA10 and AA20 pins.

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

Advisory 1.74 *Scan Shift Issues in GWG Memories*

Revision(s) Affected: 1.0

Details Very rarely high Vmin in scan shift captured in Cortex A8 and IVA2.2. The Vmin is the lowest voltage at which a device under test can operate for a given TDL (Lower the Vmin the better the margin is).

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

Advisory 1.75 *DMA4 Channel Fails to Continue With Descriptor Load When Pause Bit Is Cleared Through Config Port Access While in Standby*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details The channel hangs when the Pause bit (DMA4_CDPi [7]) is cleared through config port while in Standby.

Workaround(s) The software Workaround is to configure DMA4 to be in No-Standby or Force-Standby before clearing the Pause bit (DMA4_CDPi [7]). The DMA4 can be reverted back to Smart-Standby mode after detecting DMA4_CSR[15] of corresponding channel to be 0 or ensuring DMA4_CCR[7] bit of corresponding channel to be 0, this ensures descriptor load completion or channel termination.

Advisory 1.76 *USB OTG DMA Status Register Read Return Zero*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details When a DMA interrupt occurs, the reading of the DMA Interrupt register may fail and return 0x00, which means that software cannot determine which channel generate the irq. This issue happens if the register is read in the exact cycle that internal bus is being accessed from USB.

When a DMA transfer is complete, a DMA interrupt is generated. However, when Software reads the DMA interrupt register, the register read may be incorrect and the interrupt will be internally cleared. When the issue happens the side effect of this is that would never terminate the DMA transfer.

Workaround(s) On a DMA interrupt reception, software should read the interrupt register and go and read of the byte counts for each DMA channel in use. This way, software can identify which channel created the interrupt by checking a byte count equal to zero (for an active channel byte count equal zero meaning it is the channel that generate the interrupt).

Advisory 1.77 *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 requesters).

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): If the core is forced to an idle state (in terms of TA and 3D processing) before a cache invalidate is issued by setting , the occurrence of the lockup is substantially reduced.

Advisory 1.78 *Device Cannot Reboot if Warm Reset Occurs During MPU DPLL Lock Sequence*

Revision(s) Affected: 1.0

Details

The device lockup occurs when a warm reset event happens within a specific time window during the MPU DPLL relock sequence. The window is 2.5 system clock cycles before the lock sequence completion and 1.5 system clock cycles after the lock sequence completion. The total window cycle is 4 system clock cycles.

For 26MHz system clock this translates to a window of ~150ns during which if the warm reset is asserted, it can cause this issue. This causes the MPU DPLL clock to go low and prevents the MPU from rebooting.

In the case that warm reset happens within the 4 clock cycles window around the lock sequence completion then the MPU will hang. In such case a cold reset is the only option to recover.

Workaround(s)

There is no workaround for this issue.

Advisory 1.79 *USB Host EHCI May Stall When Exiting Smart-Standby Mode*

Revision(s) Affected: 1.1 and 1.0

Details:

When the USBHOST module is set to smart-standby mode, and it is ready to enter the standby state (that is, 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 restoring their state after the USBHOST has entered smart-standby state.

Software workaround sequence:

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

Advisory 1.80 *USB Host EHCI May Stall When Running High Peak-Bandwidth Demanding Use Cases*

Revision(s) Affected: 1.1 and 1.0

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 bursts, 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-byte 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 running 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 occurrence 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 1.81 *USB OTG DMA May Stall Under Specific Configuration*

Revision(s) Affected: 1.2, 1.1 and 1.0

Details The USB OTG module is AHB native, while the interconnect is OCP native. This issue impacts the AHB/OCP wrapper around the USB OTG module. It can only occur if at least one Read DMA channel and one Write DMA channel are active concurrently. This issue does not occur with only one DMA channel active at a time.

When the bug occurs, an AHB USB DMA read will be lost in the wrapper, which will lead to a stall of all DMA active channels. The whole USB OTG DMA will appear as stalled

The following settings do **not** have any influence on this bug:

- DMA mode 0 or 1
- Double packet buffering being set or not.
- Autoclear/autoset being set or not
- Isochronous, bulk, interrupt or control transfers
- The fact that other Tx or Rx EP may be handled by interrupt at the same time.
- The fact that the module is used in host or device mode.

Workaround(s) This issue can be circumvented when three conditions are met at the same time:

- Use only burst mode 3 (DMA_CTRL[10:9]DMA_BRSTM=0x3 This should be the default configuration as it will increase the burst lengths, which in turn will improve the throughput/bandwidth)
- The DMA address for the Tx DMA channels should be 32 bytes aligned (DMA_ADDR[4:0]=0x0)
- The payload size is constrained so that:
  - DMA_COUNT > 16 for short packets
  - (DMA_COUNT % TxMaxP) == 0 OR > 16 for large packets.

(There is no constraint on the DMA address for Rx EP.)

Note: Another workaround would be to never have an Rx and a Tx DMA channel active simultaneously. Wait for the completion of all Rx DMA channels before starting any Tx DMA channel, or Tx and Rx, respectively.

Advisory 1.82 *USB OTG DMA May Stall When Entering Standby Mode*

Revision(s) Affected 1.2, 1.1 and 1.0

Details If the OTG module is in SmartStandby Mode (OTG_SYSCONFIG.MIDLEMODE = 0x2), when an OTG DMA channel is enabled (DMA_CNTL.DMA_EN) near a USB SUSPEND condition, the MStandby signal may assert while there is residual OCP traffic initiated by the DMA.

This illegal traffic causes the OTG DMA to stall.

Workaround(s) Ensure the module cannot enter standby mode while DMA transfers are active.

Advisory 1.84 *DPLL3 in Manual Lock Mode Cannot Be Used When CORE Goes to OSWR or OFF State*

Revision(s) Affected 1.2, 1.1 and 1.0

Details This issue occurs when the DPLL3 is in Manual Lock mode (CM_AUTOIDLE_PLL[AUTO_CORE_DPLL] = 0).

Due to a wrong isolation cell on the dpll3clkon signal between CM and PRM (TIE-LOW implemented instead of ISO_LATCH), the DPLL3 Lock state signal is not taken into account by the PRM when CORE goes OSWR (Open Switch Retention) or to OFF state. Consequently, PRM will consider that the DPLL3 is not locked and the DPLL3 input clock will be stopped.

The following impacts occur:

- CORE OSWR mode: DPLL3 cannot be kept Locked when CORE OSWR state if DPLL3 is set in Manual Lock mode.
- CORE OFF mode: Architecture will not support transition to OFF mode if DPLL3 is set in Manual Lock mode.

Workaround(s) CORE OSWR and CORE OFF mode: DPLL3 must be set in Automatic mode when CORE goes to CORE OSWR or OFF state. However, DPLL3 will need to relock after Wake-Up.

Advisory 1.85 *PRCM DPLL Control FSM Removes SDRC_IDLEREQ Before DPLL3 Locks*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

When an internal SWAKEUP event occurs while CORE DPLL is going to bypass, PRCM may report that DPLL3 is locked when it is not. This is due to timing path of an internal signal which is not fitting to one L3 clock cycle.

In that case, PRCM DPLL control FSM deasserts SDRC_IDLEREQ signal before DPLL3 Lock state is set.

The consequence is that SDRC is released from IDLE with the bypass clock (which is too low) instead of the locked frequency. DLL may/may not lock based on Process Voltage Temperature conditions.

This issue is seen when DPLL3 Automatic mode is enabled:
CM_AUTOIDLE_PLL[AUTO_CORE_DPLL] = 1 or = 5.

Workaround(s)

Workaround 1

Initialization (Device boot up):

1. Disable DPLL3 automatic mode by default (CM_AUTOIDLE_PLL[AUTO_CORE_DPLL] = 0).
2. Issue will not be faced as DPLL3 is always locked.

Workaround 2

Before CORE Voltage Domain Sleep Trasition to RETENTION or OFF mode:

1. Disable Smart Reflex.
2. Reduce DPLL3 M2 Frequency to get L3 running at OPP50 Frequency (by changing M2 Divider value). This increases the period duration of one L3 clock cycle.
3. Set VDD2 Voltage for OPP100. This reduces the timing duration of the critical path signal, which will now fit to one L3 clock cycle.
4. Enable DPLL3 Automatic Stop mode. This ensures proper transition to RETENTION or OFF mode.

In summary, L3 = OPP50 + VDD2 = OPP100 combination must be used:

- In case of OPP100 (L3=200MHz, VDD2=1.1375V):
  1. Lower the frequency to 100MHz.
  2. Keep the voltage as it is (1.1375V).
  3. Enable DPLL3 Automatic Stop mode.
- In case of OPP50 (L3=100MHz, VDD2=0.93V):
  1. Keep the frequency as it is (100MHz).
  2. Increase CORE voltage to 1.1375V.
  3. Enable DPLL3 Automatic Stop mode.

Workaround 3

After CORE Voltage Domain Wake-Up Transition from RETENTION or OFF mode:

Before Enabling Smart Reflex:

1. Disable DPLL3 Automatic mode.
2. Restore previous DPLL3 M2 Frequency and CORE Voltage values.

Important Notes:

- Case 2 and Case 3 must be executed if Voltage Domain transition. These cases cannot be executed if only Power Domain transition is targeted.
- Due to another bug (see), the following two scenarios need to taken into account:
  – Target state is OSWR and voltage transitioning is happening: Case 2 and Case 3 can be applied.
  – Target state is OSWR and voltage transitioning does not happen: Case 2 and Case 3 cannot be applied; Target state must be changed from OSWR to CSWR.

Advisory 1.86 *PER Domain Reset Issue After Domain-OFF/OSWR Wake-Up*

Revision(s) Affected 1.1 and 1.0

Details

After Peripheral Power Domain (PER-PD) is woken up from OFF/OSWR state while Core Power Domain (CORE-PD) is kept active, write or read access to some internal memory (UART3/4 FIFO and sidetone memory inside McBSP2/3) does not work correctly.

This leads to a corruption of transmit or receive data of UART3/4, or output from McBSP2/3 sidetone modules. Other logics behave properly.

The cause of the issue is that memory control logic of UART3/4 FIFO and McBSP2/3 sidetone memory are not reset and remain uninitialized when PER-PD is woken up from OFF or OSWR state. The logics are properly reset when CORE-PD is coming back from OFF/OSWR state. They also get reset by warm-reset, but not reset by a module level software reset.

Workaround(s)

Do not allow PER-PD to go to OSWR/OFF as long as Core-PD is not switched to OFF/OSWR.

When both CORE-PD and PER-PD goes into OSWR/OFF, PER-PD should be brought to active before CORE-PD.

This can be done by configuring a wake-up dependency between PER-PD and WKUP-PD (PM_WKDEP_PER.EN_WKUP = 0x1) so that CORE-PD and PER-PD will wake up at the same time.

Even with the above configuration, there is a small possibillity that on transition to OFF mode,a wake-up event is generated at a time when PER-PD entered OFF/OSWR but CORE-PD is not. This timing window is very small(about 8 us at most) but to ensure the correct operation in this corner case, following procedure is recommended.

After waking up from CORE OFF/OSWR configuration:

1. Check these conditions to see if the previous Core-PD transition to OFF/OSWR was aborted (thus not actually reached OFF/OSWR state):
   - PM_PREPWSTST_PER[1:0].LASTPOWERSTATEENTERED = 0x0 (PER domain was previously OFF)
   - PM_PREPWSTST_CORE[1:0].LASTPOWERSTATEENTERED = 0x3 (CORE domain was previously ON)
2. If this corner case is detected, check the UART3/4 FIFO and/or McBSP2/3 sidetone functionally using their internal loopback features.
   - For UART3/4, the internal loop back mode can be enabled by MCR_REG[4].LOOPBACK_EN.
   - For McBSP2/3 sidetone feature, digital loop back can be used. Two different words should be used (typically 0x55.. and 0xAA..) in the loop back test to confirm correct behavior.
3. If an error is detected by the above loopback tests, execute one of these recovery sequences:
   - Generate warm-reset.
   - Put the Core-PD and PER-PD to OSWR/OFF.

Advisory 1.87 *Self-Refresh Exit Issue After OFF Mode*

Revision(s) Affected 1.1 and 1.0

Details When device is waking up from OFF mode, then SDRC state machine is sending inappropriate sequence on the DDR interface: CKE0 toggling (Self Refresh Exit) then on the next clock cycle (running at system clock frequency), Self Refresh entry Command on CS0 is issued. Right after that, the exact same sequence happen on CS1 (CKE1 followed by SR entry CMD). This is violating the JEDEC standard which is defining tXSR timing as the minimal timing between CKE toggling and next DDR command.

A violation of this tXSR timing can result in partial memory corruption. Only a limited area in the memory will be corrupted (1 or 2 row) but the row impacted is random. This can result in system instability happening after an OFF mode transition.

Due to a race condition between IO isolation release and CKE0 control, CKE0 toggling is not propagated to the IO for system clock frequencies 26Mhz and 38.4 Mhz. All other system clock frequencies: 12Mhz, 13Mhz and 19.2Mhz are impacted. 26MHz is impacted for rare hot silicon only.

Workaround(s) **Hardware workaround:**

Only 26Mhz and 38.4Mhz system clock frequencies are supported (No internal division allowed) when OFF mode is used. Using 26Mhz or 38.4Mhz will mask CKE0 toggling then DDR on CS0 won't be affected but DDR on CS1 will still be affected. Additionally, to this system clock frequency range limitation, the system should ensure that the VDD2 power domain will always be at 100% OPP when device wake-up from OFF mode for 26MHz.

Options for memory on CS1:

- **Production Option 1**: Remove the memory on CS1 and remap all the memory resources on CS0.
- **Production Option 2**: Assuming the CS1 memory is always put in self refresh and not in deep power down mode while core power domain is OFF, the Txsr violation does not create any other side effect than memory content corruption. So if the software does not rely on CS1 memory content after core power domain OFF mode by saving critical data in CS0 memory, then memory can still be used safely during active mode. This option is only valid for memory in self refresh mode while core power domain is OFF. If the memory is put in deep power down mode before transitioning the core power domain to OFF mode, then electrical issues may happen on memory depending on its design.
- **Production Option 3**: Avoid device OFF mode transition and all standby modes including core power domain transitioning to OFF mode. Power consumption impact assessment is needed in that case.

Software workarounds:

A software workaround can be used to decrease the probability for the occurrence of the failure. This workaround can be used for development purposes and will provide a significant improvement for system stability. However, it cannot be guaranteed across DM37x and memory process distribution and Voltage/ Temperature conditions. That is why it is not an option for production.

Software Case 1. When two DDR are connected to the DM37x device, then a software workaround should be applied on top of the hardware workaround to handle a correct SR or DPD exit when coming back from OFF mode with CS1 memory either in SR or DPD:

- After pad configuration save operation completion (CONTROL_GENERAL_PURPOSE_STATUS[0]:savedone bit = 1), software override the value saved in scratchpad memory by writing mode7=0x007 to @0x4800 2834 before transitioning to OFF.

- Isolation is maintaining the value on CKE1 pin during OFF mode.
- When wake-up from OFF mode occurs, the pad configuration restoration will automatically restore the updated value that will drive high the CKE1 pin when the isolation is released. This increases the timing between CKE1 rise time and the next valid command; that is, reduces the timing violation.

Note: Due to that issue, ROM code can access only to CS0 after a wake-up from OFF mode since CS1 will be available only after applicative software is launched (and software workaround applied). User should ensure that all the data stored in DDR and used by ROM code for context restoration will be put on CS0 only.

Advisory 1.88 *Programming of tCKE*

Revision(s) Affected	1.2, 1.1 and 1.0
Details	Following resolution of the "SDRC Timings are Not Aligned With JEDEC Standard" advisory that was in DM37x, the solution eats up two clock cycles of CKE low state. It affects the timing parameter tCKE, min. high and low pulse width requirement.
Workaround(s)	In order to meet the tCKE timing parameter requirements of the LPDDR memory, SDRC needs to be programmed with a value of the required tCKE value plus 2. For example, if tCKE requirement of the memory is 2, SDRC.SDRC_ACTIM_CTRLB_p needs to be programmed with 4. This in turn reduces the max tCKE value that can be programmed by 2.

Advisory 1.89 *McBSP Used in Slave Mode Can Create a Dead Lock Situation When Doing Power Management*

Revision(s) Affected

1.2, 1.1 and 1.0

Details

When using McBSP in Slave mode and doing Power management (McBSP going to IDLE state), if external CLKX clock is not provided by external peripheral component, McBSP cannot exit IDLE state. Consequence is that McBSP registers cannot be accessed anymore (except MCBSPLP_STATUS_REG[CLKMUXSTATUS] register).

Note: There is a similar limitation on the CLKR when the module is configured as a receiver.

Workaround(s)

The following workarounds are valid for CLKX as well as CLKR. There are three possible workarounds to avoid this situation.

Workaround 1

Do not use power management features of McBSP (SIDLEMODE field in MCBSPLP_SYSCONFIG_REG register set to NO-IDLE all the time).

Impact: As McBSP transition to IDLE is avoided (SIDLEMODE=NO-IDLE); PRCM cannot transition the chip to Low Power mode

Workaround 2

Keep external CLKX clock always running during application (for example during an Audio Playback).

- At the *beginning* of the application (e.g. Audio Playback): Keep CLKX always running by setting external peripheral component register accordingly.
- At the *end* of the application (e.g. Audio Playback): Unload McBSP2 drivers (and set XRST bit from SPCR2 to '0'). Then stop CLKX by software (by setting external peripheral component register accordingly).

Impact: Serial clock kept active during whole application (e.g. Audio Playback).

Workaround 3

If SIDLEMODE is used (other value than NO-IDLE) ensure McBSP registers can be accessed by reading/polling the MCBSPLP_STATUS_REG[CLKMUXSTATUS] register. If registers cannot be accessed (CLKMUXSTATUS=1), take corrective actions in software, such as (re)enabling CLKX.

Advisory 1.90 *DPLL3 Bypass Condition Does Not Consider State of SGX FCLK*

Revision(s) Affected 1.2, 1.1 and 1.0

Details DPLL3 bypass condition does not consider the state of SGX FCLK. Therefore, DPLL3 can be put in bypass mode even if SGX FCLK is still active, thus causing unexpected SGX FCLK frequency drop.

There is no issue when SGX FCLK source is DPLL4 (Selected through CM_CLKSEL_SGX).

Workaround(s) There are two possible workarounds. Recommendation is to use Workaround 1 as it is more efficient in terms of power consumption.

Workaround 1

Clear the CM_AUTOIDLE_PLL[AUTO_CORE_DPLL] bit (that is, Auto Control Disabled) when SGX FCLK is enabled (CM_FCLKEN_SGX[EN_SGX]='1') and set it back to expected automatic PRCM control configuration when SGX FCLK is disabled. Then L3 gating will still be possible while DPLL3 is kept in locked state.

Workaround 2

DPLL3 can also be forced to stay in locked state by enabling the L3 interface clock for SGX (CM_ICLKEN_SGX[EN_SGX]='1'). However, this will be less efficient in terms of power consumption as the L3 clock tree is enabled.

Advisory 1.91 *Top Ball Compliancy With LPDDR JEDEC Standard*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

JEDEC standard defined the usage of the different pins; some of them should be unconnected.

Within DM37x some of these "unconnected pins" have been used.

This is the case for CBP package pins AB14 and AB20 that are connected to VSS, and AC20 that is used for GPMC_A11. These pins may cause electrical issues according to the usage of these pins on memory side.

Workaround(s)

For AB14 and AB20, nothing can be done within DM37x. Need to check if these pins are connected to VSS or unconnected within memories.

For AC20, when GPMC_A11 is not used, it is recommended to configure the pad in high impedance state to be close to a not connected state.

For ES1.0, this is achieved by programming CONTROL_PADCONF_SDRC_CKE1[31:16] to 0x0001. This means that:

- The muxmode is set to undefined mode to have the default setting of the IO, which is input. Safe_mode cannot be used due to another bug on ES1.0 (see).
- PU/PD and input are disabled to not have a source of power consumption and also to isolate the IO.

Due to and , this pad will drive a strong '1' during reset (safe-mode), users need to check what can be the impact within memory.

Starting from ES1.1, safe_mode can be used, program CONTROL_PADCONF_SDRC_CKE1[31:16] to 0x0007. This means that:

- The muxmode is set to safe_mode.
- PU/PD and input are disabled to not have a source of power consumption and also to isolate the IO.

Issue has been already identified with a memory manufacturer. These three pins are connected to VSS within the memory, though not documented.

Without above workaround, it will imply power consumption and reliability issue on the GPMC_A11.

With above workaround:

- ES1.0: This consumption will appear only during reset, without any reliability impact for regular applications.
- Starting from ES1.1: No extra consumption, no reliability issue.

Advisory 1.92 — *I2C: In SCCB Mode, Under Specific Conditions, the Module Might Hold the Bus by Keeping SCL Low*

Revision(s) Affected 1.2, 1.1 and 1.0

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

Revision(s) Affected 1.2, 1.1 and 1.0

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

Revision(s) Affected 1.2, 1.1 and 1.0

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 1.95 *One Timing Does Not Follow the JEDEC Standard*

Revision(s) Affected 1.2, 1.1 and 1.0

Details tRFC is specified as Auto-refresh to Auto-refresh/Active period. This parameter is also used as minimum Self-refresh period in JEDEC standard. tRFC period is programmable in SDRC controller to meet the standard. In the device, tRFC is not used to determine minimum Self-Refresh period,but the minimum self-refresh period depends on the delay with the next SDRC command and can be min. 1 clock period. In the device, the minimum self-refresh period can be controlled through tCKE (tRFCmin=tCKE).

There is no impact; the situation is tolerated by memory components (checked with memory vendors) for both 1 cycle and 2 cycles.

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

Advisory 1.96 *4-cycle Saturating .M Unit Instructions May Mask 2-cycle Saturating .M Unit Instruction Saturation Bit Update*

Revision(s) Affected 1.2, 1.1 and 1.0

Details If a 4-cycle instruction that can saturate is writing back its result in the same cycle that a 2-cycle instruction can saturate, then only the saturation condition for the 4-cycle instruction is recorded in the CSR.SAT bit and SSR.M1 or SSR.M2 control registers.

In other words, if the 4-cycle instruction does not saturate, but the 2-cycle instruction does saturate, the fact that the 2-cycle instruction is lost, and is not recorded in the CSR.SAT and SSR.Mx bit-fields.

The following list includes 4-cycle instructions that can mask the 2-cycle saturation condition:

- SMPY32
- CMPYR
- CMPYR1
- DDOTPH2R
- DDOTPL2R

The following list includes 2-cycle instructions that can set the saturation flags:

- SMPY
- SMPYLH
- SMPYHL
- SSHVL
- SSHVR

The 2-cycle instruction must be started two cycles after the 4-cycle instruction in order for the conflict situation to occur. For example:

- CMPYR .M1 A0,A1,A2
- NOP
- SSHVL .M1 A8,A9,A10

If the CMPYR instruction does not saturate, the saturation status of the SSHVL instruction will not be recorded. Note also that the 2-cycle and 4-cycle instructions must be issued on the same .M unit for this interference to occur. If the SSHVL instruction were on the .M2 unit, the saturated bit would be recorded correctly.

Workaround(s) Ensure that no code has scheduled any of the instructions from the 4-cycle list to write back in the same cycle as the instructions in the 2-cycle list.

The compiler added a workaround option --enable:c64p_csr_sat_workaround. Available in C6x compiler versions 6.1.16 and later, 7.0.2 and later, 7.1.0B2 and later, and 7.2.0A and later.

Advisory 1.97 *SPLOOP CPU Cross-Path Stall*

Revision(s) Affected 1.2, 1.1 and 1.0

Details This erratum affects the IVA2 c64x core. When the following three rules are met, a stall is seen when a SPKERNEL instruction is executed:

1. Cross-path instruction rule: An instruction reading a register via the cross path in the first cycle after SPKERNEL instruction.
2. Data dependence rule: An instruction in the SPLOOP body that writes to the previous cross-path read register. This instruction can be anywhere in the SPLOOP body.
3. Functional unit rule: No instruction in parallel with the SPKERNEL instruction that uses the same functional unit as the cross-path read instruction (see Rule 1).

This results in a 1-CPU cycle stall for each iteration of the loop.

The following three examples of code are affected by this issue:

```
Example 1:
SPLOOP 1
MV .S1 A0, A1; stalls every iteration due to MV after loop
SPKERNEL
MV .S2X A1, B2
```

```
Example 2:
SPLOOP 14
MV .S1 A0, A1; stalls every iteration due to MV after loop
NOP 9
NOP 9
NOP 9
NOP 9
SPKERNEL
MV .S2X A1, B2
```

```
Example 3:
SPLOOP 1
MV .S1 A0, A1; stalls every iteration due to MV after loop
SPKERNEL
||NEG .L2 B3, B4 ; Qualifies for rule 3, function unit rule
MV .S2X A1, B2
```

The following three examples are not affected by this issue:

```
Example 1:
;No stalls: No cross path in instruction after SPKERNEL
SPLOOP 1
MV .S1 A0, A1
SPKERNEL
MV .S1 A1, A2
```

```
Example 2:
;No stalls: A1 not written to in loop body
SPLOOP 1
MV .S1 A0, A2
SPKERNEL
MV .S2X A1, B2
```

```
Example 3:
;No stalls: Instruction in parallel with SPKERNEL prevents bug since it's in
the same unit as the instruction that uses the cross-path
SPLOOP 1
MV .S1 A0, A1
SPKERNEL
||NEG .S2 B3, B4; masks the bug
MV .S2X A1, B2
```

Workaround(s)

Unfortunately, the way SPLOOP code is scheduled is beyond the application developer's control and is controlled by the compiler. Therefore there are no direct workarounds for non-assembly source code. The fix is included in the following compiler releases (versions that are listed refer to c6x compiler within Code Composer Studio):

- 6.0.25 or later
- 6.1.15 or later
- 7.0.2 or later
- 7.1.0B2 or later
- 7.2.0A or later

Advisory 1.98 *RTA Feature is Not Supported*

Revision(s) Affected 1.2, 1.1 and 1.0

Details RTA (retention till access) feature is not supported and leads to device stability issues when enabled.

These modules are embedding memories with RTA support:

- MPU subsystem (cache memories)
- IVA2.2 SS: internal memories/cache, eDMA, HW accelerators
- OCM RAM
- SGX
- ISP
- Display subsystem
- HS USB OTG
- McBSP

Workaround(s) CONTROL_MEM_RTA_CTRL[0]:HD_MEM_RTA_SEL default value is '1' meaning RTA enabled. It must be reset to '0' as soon as device is booted for correct operation. RTA should be disabled at boot time and then consistently kept disabled.

The RTA control register is in core power domain and will recover its default value (RTA enabled) as soon as core power domain is waking up from OFF mode. SW is in charge of disabling the RTA feature as soon as core power domain is recovering from OFF state.

By construction, this workaround cannot be applied before ROM code completion. ROM code execution will be executed with RTA ON but this will not cause any stability issue. ROM code execution (including secure RAM restore in the case of HS device) is performed in a specific controlled framework that is not exhibiting any issue with RTA ON.

Advisory 1.99 *HS USB OTG Spurious IRQ When Switching From DMA Mode 1 to DMA Mode 0 or to CPU Mode for Unloading the Last Short Packet of a Transaction*

Revision(s) Affected 1.2, 1.1 and 1.0

Details For unloading the last short packet of a transaction received in the RXFIFO, the USB OTG module will generate an IntrRx interrupt and the software will switch from DMA mode 1 to DMA mode 0 or from DMA mode 1 to CPU mode. When this occurs, it is possible that the HS USB OTG module will generate a spurious IntrRx IRQ. The issue only occurs in Rx mode. (Tx mode is not impacted). The issue also occurs in device or host mode. All other transitions between mode1/mode0/CPU mode are not impacted.

Workaround(s) When switching from DMA mode 1 to CPU mode for unloading the last short packet in the FIFO:

- Do not clear the RXCSRH[5]:DMAReqEnab and RXCSRH[3]:DMAReqMode bits indicating that DMA mode 1 and DMA request is enabled
- Unload the FIFO with CPU accesses
- Clear the RXCSRL[0]:RxPktRdy bit. If needed, depending on SW programming, update RXCSRH[5]:DMAReqEnab and RXCSRH[3]:DMAReqMode.

When switching from DMA mode 1 to DMA mode 0:

- Do not clear the RxCSRH[5]DMAReqEnab and RxCSRH[3]DMAReqMode bits indicating that DMA mode 1 and DMA request is enabled
- Reprogram DMA_CNTL[2]:DMA_MODE to mode 0 and update the required value in the DMA_COUNT register
- Once the DMA interrupt is generated, clear the RXCSRL[0]:RxPktRdy bit. If needed, depending on SW programming, update RXCSRH[5]:DMAReqEnab and RXCSRH[3]:DMAReqMode).

Advisory 1.100 *Write to OHCI Registers May Not Complete*

Revision(s) Affected 1.2, 1.1 and 1.0

Details When the L3 clock frequency is higher than 168MHz and when UHH_SYSCONFIG[0]:AUTOIDLE is set, a write to any OHCI register may not complete. An internal clock is cut too soon, potentially leaving the last write access unfinished.

The bug can materialize under different scenarios such as:

- The system is hanging when coming back from a suspend because the resume command written to HCCONTROL[7:6]:HCFS does not complete
- The system is twice detecting the same IRQ from OHCI because the write to the HCINTERRUPTSTATUS register does not complete

EHCI and UHH registers are not impacted by this bug.

Workaround(s) One workaround consists in adding a dummy read to an OHCI register (for example, HCREVISION) after each write to any register. This dummy read will re-enable the clock, which will allow the previous write to complete.

A second workaround, easier to implement at OS level, consists in disabling the USB HOST autoidle feature in UHH_SYSCONFIG[0] AUTOIDLE. By doing this, the writes can correctly complete.

Advisory 1.101 *Violation of Vix Crossing Specification*

Revision(s) Affected 1.1 and 1.0

Details JEDEC standard specifies the voltage levels associated with the two crossing points of the DDR CK and NCK signals. The Vix specification range is Min: 0.4xVDDQ (V), Max: 0.6xVDDQ (V). This means Min: 0.72 V, Max: 1.08 V for DM37x specification.

Due to the skew on sdrc_clk and sdrc_nclk, these crossing points are not following JEDEC requirements. No functional issue has been reported related to this issue, only the violation versus the JEDEC specification

For an 8pF load at drive strength 0, the worst-case Vix-hi crossing is exactly at the spec limit and Vix-low is 40mV below the specification. For an 8pF load at drive strength 1, the worst-case Vix-hi crossing is 100mV above the limit and Vix-low is 120mV below the specification.

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

Advisory 1.102 *HS USB OTG Bus Reset*

Revision(s) Affected 1.2, 1.1 and 1.0

Details When the HS USB OTG module is acting as a host (A-Dev) and the software initiates a suspend transition (by setting POWER[1]:Suspend_Mode), the bus will enter the suspend state. If the software then tries to issue a USB reset (POWER[3]:Reset), the HS USB OTG module will not initiate the USB reset on the bus, and will remain in the suspend state.

This issue exists only when HS USB OTG is acting as a host. In device mode, suspend to reset detection is possible.

Workaround(s) Resume the bus (POWER[2]:Resume) before resetting it (POWER[3]:Reset).

Advisory 1.104 *McBSP2 Data Corruption*

Revision(s) Affected 1.2, 1.1 and 1.0

Details When McBSP2 is configured in transmit slave mode, the OCP clk is used for the audio and transmit buffers, CLKx clock is used for the Transmit Shift Register. If McBSP2 is requested to enter idle mode, it can acknowledge this idle and in that case the logic between audio and transmit buffers will switch to CLKx.

A short Idle Request pulse to McBSP2 causes a glitch inside McBSP2 logic. This leads to incorrect McBSP2 operations such as:

- FIFO level corruption
- FIFO pointer corruption

The glitch does not result in system freeze, only the on going frame will be corrupted.

This issue only occurs with McBSP2, as the other McBSP do not have this buffer structure and so clock switching mechanism.

Workaround(s) 100% valid workaround is to use McBSP2 in No-Idle mode. This workaround has impact to power consumption, especially for scenarios like low-power MP3 playback.

If this workaround cannot be taken, there is no 100% valid workaround. It is recommended to minimize the issue occurrence rate as much as possible:

- Avoid repeated short idle request generation. One use case example that can lead to such idle request generation is a small data transfer by sDMA to a peripheral FIFO.
- Short idle request generation due to a random system event cannot be avoided, but this possibility is very low in most systems.

Advisory 1.105 *DSI-PLL Power Command 0x3 Not Working*

Revision(s) Affected 1.2, 1.1 and 1.0

Details The DSI-PLL power commands allows user to disable/enable the clock supply to HSDIVIDER and to DSI complex IO. This setting is done in bitfield DSI_CLK_CTRL.PLL_PWR_CMD.

Setting to 0x3 to PLL_PWR_CMD gate the clock supplied to DSI complex IO but also to the HSDIVIDER module. This not the normal behavior, only the clock supplied to DSI complex IO should be gated.

Workaround(s) Software should not set 0x3 in PLL_PWR_CMD and keep a setting to 0x2 instead (command to change to ON state for both PLL and HSDIVISER).

Advisory 1.106 *MPU Leaves MSTANDBY State Before IDLEREQ of Interrupt Controller is Released*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

Due to a race condition when WFI instruction is executed and an interrupt is triggered at the same time, MPU leaves MSTANDBY state while Idle Request of Interrupt Controller (INTC) is still asserted (INTC Idle Request signal is generated by the PRCM, while MSTANDBY signal is an output from MPU). This is due to the fact that there is no handshaking mechanism between MPU Subsystem and PRCM.

Any access to the Interrupt Controller registers during the time where INTC Idle Request is still asserted will generate a Data Abort (The worst case window where this can happen is 4xL3 clock cycles after Wakeup from WFI).

Workaround(s)

Do a dummy read to any register from the Clock Manager (CM) after Wakeup from WFI (memory area of INTC registers must be defined as strongly-ordered in the MMU). This dummy read will add the required latency (> 4xL3 clock cycles) to avoid the read of INTC registers in the critical window where Data Abort can happen. Dummy read of any register from the SDRC or PRM will also work fine.

Advisory 1.107 *Some Power Domains Cannot Go To OFF Mode After Warm Reset*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

After a Warm Reset, due to the wrong value of the internal signals indicating USBHOST, SGX, CAM, DSS and PER power domains status, the clock manager FSM in the PRM cannot initiate any new sleep transition for these power domains.

As a consequence, USBHOST, SGX, CAM, DSS and PER Power Domains cannot go to off mode after a warm Reset. Thus, device will not reach OFF mode.

This is a corner case and always occurs in the following scenario:

- Device is waking up from OFF mode with an I/O wake-up event.
- Warm reset is generated before clearing the PM_WKST_WKUP.ST_IO bit.

It will never occur if wake-up event is something other than an I/O wake-up event (for example: GPTimer). Because in the other cases the wake-up events will be cleared by the warm reset (in the I/O wakeup case,the PM_WKST_WKUP.ST_IO bit is in the PRM domain, and then not sensitive to a warm reset).

Workaround(s)

When rebooting from a warm reset, toggle USBHOST, SGX, CAM, DSS and PER sleep dependencies on MPU, and then clear them after a minimum of two SYS_CLK cycles. This unlocks the CM_FSM and allows any new power transition. The sleep dependencies are:

- CM_SLEEPDEP_USBHOST = 0x2 (USB HOST domain sleep dependency with MPU domain is enabled)
- CM_SLEEPDEP_SGX = 0x2 (SGX domain sleep dependency with MPU domain is enabled)
- CM_SLEEPDEP_CAM = 0x2 (CAM domain sleep dependency with MPU domain is enabled)
- CM_SLEEPDEP_DSS = 0x2 (DSS domain sleep dependency with MPU domain is enabled)
- CM_SLEEPDEP_PER = 0x2 (PER domain sleep dependency with MPU domain is enabled)
- Wait a minimum of two SYS_CLK cycles
- CM_SLEEPDEP_USBHOST = 0x0 (USB HOST domain sleep dependency with MPU domain is disabled)
- CM_SLEEPDEP_SGX = 0x0 (SGX domain sleep dependency with MPU domain is disabled)
- CM_SLEEPDEP_CAM = 0x0 (CAM domain sleep dependency with MPU domain is disabled)
- CM_SLEEPDEP_DSS = 0x0 (DSS domain sleep dependency with MPU domain is disabled)
- CM_SLEEPDEP_PER = 0x0 (PER domain sleep dependency with MPU domain is disabled)

Advisory 1.108 *USBHOST Configured In Smart-Idle Can Lead To a Deadlock*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

In the following configuration:

- USBHOST module is set to smart-idle mode.
- PRCM asserts idle_req to the USBHOST module. (This typically happens when the system is going to a low power mode: all ports have been suspended, the master part of the USBHOST module has entered the standby state, and software has cut the functional clocks.)
- A USBHOST interrupt occurs before the module is able to answer idle_ack, typically a remote wakeup IRQ.

Then the USB HOST module will enter a deadlock situation where it is no more accessible nor functional. The only way to recover will be to perform a software reset of the module.

Workaround(s)

The best workaround consists of switching the module to force idle mode right before cutting the module's FCLK.

- The bus has reached the suspend state.
- Write SYSCONFIG:Idlemode= ForceIdle and read it back to ensure the write has been taken into account in case of posted writes.
- Cut the FCLK.
- Idle_req will be asserted and idle_ack answered within one L3 clock cycle.

Upon resume or remote wakeup, switch back the module to smart-idle.

This workaround reduces the failure window to only one L3 clock cycle. The probability for an interrupt to fire at this exact time is considered extremely low, and the case has never been hit on board.

Advisory 1.109 *USB OTG Software Initiated ULPI Accesses To PHY Registers Can Halt the Bus*

Revision(s) Affected 1.2, 1.1 and 1.0

Details Software initiated read/write accesses to the PHY ULPI registers (that is, when using USB OTG ULPIReg registers) may be wrongly mixed up with USB OTG Tx traffic. Due to this corruption the USB OTG will detect a Vbus error or a Babble error (logged in IntrUSB register), and halts the communication.

Hardware initiated accesses to the PHY ULPI registers are not impacted because they occur during connect, disconnect, suspend, and resume, when there is no USB traffic. Only software initiated accesses are impacted, and only when there is on-going USB Tx transfers.(including SOF in host mode) Software initiated accesses during connect, disconnect, suspend, and resume are not impacted. Host and peripheral modes are impacted.

Workaround(s) If possible do not use software initiated reads/writes to access the PHY ULPI registers. Depending on the PHY, there may be a possibility to access these registers by I2C.

If the USB OTG is host, another workaround consist in putting the bus in suspend mode with low-power mode disabled when willing to read/write an ULPI PHY register:

- Disable PHY low-power mode (Power:EnableSuspendM=0).
- Execute USB SUSPEND.
- Do the ULPI register read/write.
- Execute USB RESUME.
- Restore Power:EnableSuspendM.

If the USB OTG is peripheral, another workaround consists in using the 1-2 us time frame after reception of SOF, before any transfer begins on the bus:

- Enable the SOF interrupt.
- Program the ULPIReg registers except the ULPIRegControl:D0 bit.
- When the SOF interrupt fires, set the ULPIRegControl:D0 bit to make the access.
- Disable the SOF interrupt.

Advisory 1.110 *At Context Restore, a Certain PER DPLL Programming Can Introduce Wake-up Latencies*

Revision(s) Affected 1.2, 1.1 and 1.0

Details After OFF mode transition, among many restorations, the ROM Code restores the CM_AUTOIDLE_PLL register, and after that, it tries to relock the PER DPLL.

In case the restoration data stored in scratchpad memory contains a field CM_AUTOIDLE_PLL.AUTO_PERIPH_DPLL = 1, then the way the ROM Code restores and locks the PER DPLL does not respect the PER DPLL programming scheme.

In that case, the DPLL might not lock. Meanwhile, when trying to lock the PER DPLL, the ROM Code does not hang. Only extra latencies are introduced at wake-up.

Workaround(s) When saving the context-restore structure in scratchpad memory, in order to respect the PER DPLL programming scheme, it is advised to store 0 in the CM_AUTOIDLE_PLL.AUTO_PERIPH_DPLL field of the saved structure.

After wake-up, the application should store in CM_AUTOIDLE_PLL register the right desired value.

Advisory 1.111 *USB Peripheral Booting is Not Operational When an MMC Booting has Been Attempted Before*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

After Power on Reset, ROM Code reads the sys-boot pins to determine which booting sequence it must perform. The Sys-boot pins define the following scenarios, (among many others):

- Boot from MMC#1 or #2, and if it fails, attempt booting from USB.
- These scenarios can be used in manufactory, at the programming stage, when the embedded eMMC / eSD memory is blank.

The USB Peripheral Booting fails if each of the following conditions is respected:

- The sys-boot pins configuration defines a booting scenario when MMC#1 or MMC#2 booting is attempted before any USB booting.
- A TPS659xx PMIC is present on the platform.
- The USB transceiver involved in the USB booting phase is the one integrated in the PMIC.

Workaround(s)

Choosing another sys-boot pin configuration forcing the USB peripheral booting first followed by the memory booting is the simplest solution. Note that USB Booting will be performed only after Power-On Reset. If it is a necessity to boot first from a memory, and if UART3 is available on board:

- Choose a sys-boot configuration where UART3 peripheral booting is attempted.
- Download via UART a very small piece of code which writes a Software Booting Configuration in Scratchpad memory forcing a USB boot at software reset and then perform a software reset by software.

A hardware workaround consisting in the use of another USB transceiver than the one integrated in the PMIC is also possible.

Advisory 1.112 *DSS: Color Phase Rotation (CPR) Breaks 2 Pixel Wide Updates When in Stall Mode*

Revision(s) Affected 1.2, 1.1 and 1.0

Details There is a deadlock in the internal transaction pipeline when all these conditions are met:

- CPR is enabled (DSS.DISPC_CONFIG[15] CPR bit set to 1)
- Stall mode is enabled (DSS.DISPC_CONTROL[11] STALLMODE bit set to 1)
- There are updates of two or less pixels per line (DSS.DISPC_SIZE_LCD[10:0] PPL < 2)

Workaround(s) In stall mode when CPR is enabled, use PPL >= 2 (number of pixels per line = PPL+1). Enlarge the update area to never have updates two pixels wide or less.

Advisory 1.113 *USB HOST: Impossible To Attach a FS Device To An EHCI Port - Handoff To OHCI Is Not Functional*

Revision(s) Affected 1.2, 1.1 and 1.0

Details It is not possible to perform USB transactions with an FS device on port USB_B1/B2.

EHCI is able to detect the device, reset it, and find that it is a FS device. SW will then hand off the port to OHCI by setting the PORTSC[5]:PO (PORT OWNER) bit.

OHCI is able to detect the connection, but then cannot communicate with the device.

When the PHY switches to FsLsSerialMode, the ULPI DIR signal will go to 1 forever. In HS mode, if DIR is 1, then the ULPI DATA switches to input only mode. This input only configuration is wrongly kept after switching to FsLsSerialMode, so USB transactions cannot occur.

Attaching a FS device directly to an OHCI port is working fine.

Workaround(s) Only attach FS devices to OHCI ports or use an HS hub to interface between EHCI ports and LS/FS/HS devices.

Advisory 2.1 *USB Host Clock Drift Causes USB Spec Non-compliance in Certain Configurations*

Revision(s) Affected 1.2, 1.1 and 1.0

Details

Certain combinations of SYS_CLKIN frequency and DPLL5 clock configurations (for example, M and N values) can cause the USB clock coming from the EHCI controller to have long term frequency drift. This may exceed the input clock specification of the USB PHY and may create a violation of the USB signal rate specification. Depending on the tolerance of the USB PHY, this may result in operational errors on the USB port.

The USB Host uses the EHCI controller for high speed USB operations, and communicates to an external USB PHY over ULPI. The EHCI controller masters the ULPI clock (HSUSBx_CLK) across this interface. This clock is sourced from DPLL5, which is configured using multiply (M) and divider (N) values, as well as a post divider (DIV_120M). The values of M, N, and DIV_120M affect the PLL output characteristics, one of which is long term frequency drift. Because of the stringent USB specification for a transmit signal rate of +/-500 ppm, it is possible to configure the PLL to generate the correct frequency; however the ULPI clock may have enough long term frequency drift to cause an external USB PHY to generate a USB transmit signal rate outside of the specification. To avoid this issue, recommended values for both DPLL5 configuration and device clock input frequency are provided below.

The USB EHCI logic uses a 120 MHz clock to operate its internal state machines and produces a 60 MHz ULPI clock that becomes the clock reference for an external USB PHY. Typically, this USB PHY will use this 60 MHz reference clock and multiply it by 8 to create a 480 MHz USB data clock. DPLL5 sources the 120 MHz clock that the USB EHCI logic uses, so any characteristics of this 120 MHz clock will influence the performance of other clocks or signal rates that are derived from it.

In the case of the devices affected, the major contributor to the long term frequency drift is the update rate of the VCO. The PLL was designed to update its frequency at an update rate = SYS_CLK/(N+1). Slower update rates allow the output clock to drift farther from its desired frequency between updates, resulting in an inaccurate 60 MHz clock, and causing potential problems for the USB PHY that is using this clock. The recommended solutions shown below maximize this update rate, and thus minimize the long term frequency drift of the output clock. This can be used by an external USB PHY to produce a USB transmit signal rate that is within USB specification.

The PLL also has varied output performance depending on its VCO frequency. The VCO self calibrates at power-on to use one of three operating points. When the VCO is configured to operate in the lower portion of the specified frequency range, the self calibration logic may select either of two lower power operating points. The lowest power operating point has a high noise output and the highest power operating point has the lowest noise output. If the VCO is configured to operate in the upper portion of the specified frequency range, the self-calibration logic would always select the highest power operating point which provides the lowest noise output. The recommended solutions shown below configure the VCO (VCO output = SYS_CLK*M/N) close to its 1 GHz maximum output frequency, so the self-calibration logic will always choose the lowest noise operating point.

Workaround(s)

For new designs, and for existing designs with system clock frequencies of 12 MHz, 19.2 MHz or 38.4 MHz, use the following table to configure the DPLL5 optimally. This will result in the lowest long term frequency drift on the ULPI 60 MHz clock, and will eliminate the issue discussed in this alert.

Table 35. DPLL5 Configurations for New Designs and Designs With 12 MHz, 19.2 MHz or 38.4 MHz Clock Frequencies

System Clock (SYS_CLK) [1]	DPLL5 M Value (CM_CLKSEL4_PLL.PERIPH2_DPLL_MULT)	DPLL5 N Value (CM_CLKSEL4_PLL.PERIPH2_DPLL_MULT) [2]	DPLL5 DIV_120M Value (CM_CLKSEL5_PLL.DIV_120M)
12 MHz	80	0	8
19.2 MHz	50	0	8
38.4 MHz	25	0	8

Notes:

(1) The device system clock (SYS_CLK) is derived from the input clock crystal or oscillator (sys_xtalin) and could be divided by 2 using PRM_CLKSRC_CTRL.SYSCLKDIV. The table above refers to the system clock which is SYS_CLK = (sys_xtalin frequency) / SYSCLKDIV.

(2) The N value in the table is the value of the register. The actual divide value is N+1. For example, if N=11, the divide value is 12.

For existing designs with system clock frequencies of 13MHz or 26MHz, you can eliminate this issue by changing the input clock crystal or oscillator to 12MHz, 19.2MHz or 38.4MHz as shown above. This is the preferred solution to eliminate the issue discussed in this alert. Be sure to evaluate all system clock requirements when changing input clock frequencies.

If you cannot change the system clock frequency, use the following values for DPLL5 configuration. The first two options (13 MHz, M = 443, N = 5, M2 = 8 and 26 MHz input, M = 443, N = 11, M2 = 8) are optimal for these clock input frequencies, but because of the low update rate, and the inexact 120 MHz frequency, they could result in enough long term frequency drift on HSUSBx_CLK to cause an external USB PHY to create an occasional out of spec transmit signal rate on the USB bus. The third option (26 MHz input, M = 480, N = 12, M2 = 8) has an advantage of creating an exact 120 MHz frequency but uses a a slightly lower update rate (2.0 MHz vs 2.17 MHz). The choice between these two options with a 26 MHz input should be based on characterization on the end system.

Table 36. DPLL5 Configurations With 13 MHz and 26 MHz Clock Frequencies

System Clock (SYS_CLK)	DPLL5 M Value (CM_CLKSEL4_PLL.PERIPH2_DPLL_MULT)	DPLL5 N Value (CM_CLKSEL4_PLL.PERIPH2_DPLL_MULT)	DPLL5 DIV_120M Value (CM_CLKSEL5_PLL.DIV_120M)
13 MHz	443	5	8
26 MHz	443	11	8
26 MHz	480	12	8

Even in this configuration, this issue may or may not produce data transfer errors on the USB bus. The USB PHY may be able to generate signal rates that are in spec, or the attached downstream USB device may be able to tolerate the occasional out of spec signal rate. For example, data transfer errors on bulk packets may result in an occasional lost packet that will be retransmitted. It is important to evaluate and characterize the system performance of the USB data path to determine if this issue affects the overall operation of the product.

Revision History

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

This silicon errata revision history highlights the technical changes made from the previous to the current revision.

See	Additions/Modifications/Deletions
Advisory 2.1	Updated workaround for "USB Host Clock Drift Causes USB Spec Non-compliance in Certain Configurations"

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