TMS320C6412

Digital Signal Processor

Silicon Errata

Silicon Revisions 2.0, 1.2, 1.1, 1.0

SPRZ199J
April 2003
Revised February 2010

Copyright © 2010, Texas Instruments Incorporated

REVISION HISTORY

This silicon errata revision history highlights the technical changes made to the SPRZ199I revision to make it an SPRZ199J revision.

Scope: Clarification of location of pin1 on chip.

PAGE(S) NO.	ADDITIONS/CHANGES/DELETIONS
5	Added note above Figure 1 detailing position of pin1 when device is properly aligned. Updated Figure 1 to clarify position of pin one. Added description of V at bottom of Figure 1.

Contents

1 Introduction

This document describes the known exceptions to the functional specifications for the TMS320C6412 digital signal processor. [See the *TMS320C6412 Fixed-Point Digital Signal Processor* data manual (literature number SPRS219).] Throughout this document, TMS320C64x and C64x refer to TMS320C6412.

For additional information, see the latest version of the *TMS320C6000 DSP Peripherals Overview Reference Guide* (literature number SPRU190).

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

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

1.1 Device and Development-Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all TMS320™ DSP devices and support tools. Each TMS320™ DSP commercial family member has one of three prefixes: TMX, TMP, or TMS. Texas Instruments recommends two of three possible prefix designators for its support tools: TMDX and TMDS. These prefixes represent evolutionary stages of product development from engineering prototypes (TMX/TMDX) through fully qualified production devices/tools (TMS/TMDS).

Device development evolutionary flow:

TMX Experimental device that is not necessarily representative of the final device's electrical specifications

TMP Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification

TMS Fully qualified production device

Support tool development evolutionary flow:

TMDX Development-support product that has not yet completed Texas Instruments internal qualification testing.

TMDS Fully qualified development-support product

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

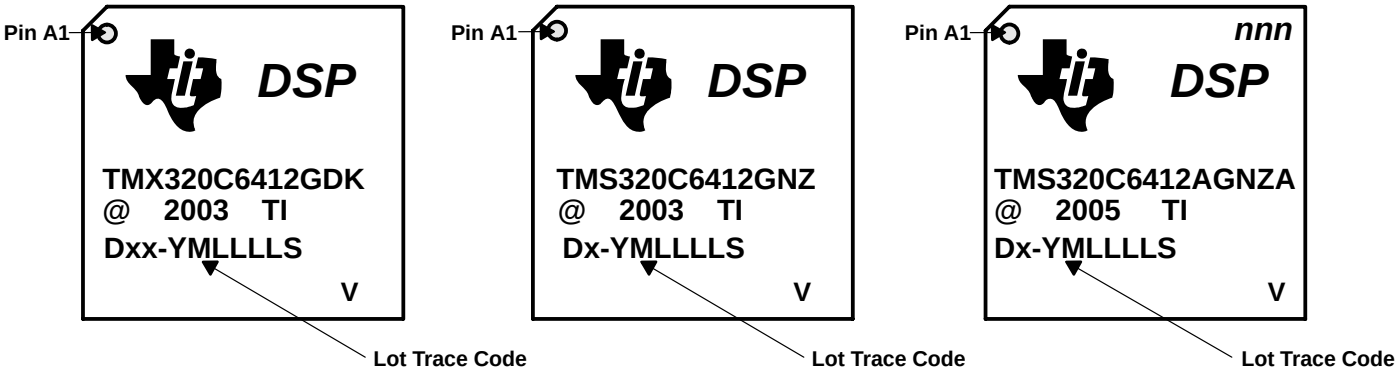
"Developmental product is intended for internal evaluation purposes."

TMS320 is a trademark of Texas Instruments.
All trademarks are the property of their respective owners.

1.2 Revision Identification

The device revision can be determined by the lot trace code marked on the top of the package. The location of the lot trace code for the GDK and GNZ packages are shown in Figure 1. Figure 1 shows ***some*** examples of the types of C6412 package symbolization for −500 MHz and −600 MHz devices.

NOTE: Pin A1 is always located at the top-left corner when you can read the silkscreening/laser-etching and view the TI logo properly.



"nnn" represents the device speed for the −500 MHz devices ***only*** (both commercial and extended temperature). For example:

500 = 1.2 V, 500-MHz Core, 100-MHz EMIF
500 = 1.4 V, 500-MHz Core, 100-MHz EMIF, Extended Temperature [A at end of device part number]

"V" at bottom right corner represents Vendor ID.

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

Figure 1. Example, Lot Trace Codes for TMX320C6412 and TMS320C6412 (GDK and GNZ Packages)

Silicon revision is identified by a code on the chip. The code is of the format Dxx-YMLLLLS or Dx-YMLLLLS. If xx is 10, then the silicon is revision 1.0; if x is "A", then the silicon is revision 1.1; if x is "B", then the silicon is revision 1.2, if x is "C", then the silicon is revision 2.0, etc.

Table 1. Lot Trace Codes

Lot Trace Code (xx or x)	Silicon Revision	Comments
C	2.0	TMS320C6412A
B	1.2	TMS320C6412
A	1.1	TMX320C6412, TMS320C6412
10	1.0	TMX320C6412

2 Silicon Revision 2.0 Known Design Exceptions to Functional Specifications and Usage Notes

2.1 Usage Notes for Silicon Revision 2.0

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.

L1P Cache: Incorrect Update of the L1P Tag RAMs (All C64x Devices)

On C6412 silicon revision 2.0 and earlier, when the CPU is executing non-cacheable code from external memory and there is snoop activity from L2 to L1P occurring at the same time, an incorrect update to the L1P Tag RAM can occur.

Snoop activity from L2 to L1P can be generated two ways:

1. EDMA/QDMA activity to L2
2. Block cache invalidates initiated in L2

When there is a non-cacheable L1P fetch that is returned from L2 to L1P, ***and*** there is a snoop from L2 in the very next cycle, then the snoop tag read interferes with the tag/status RAM write for the non-cacheable data. This interference causes the tag RAMs to be incorrectly updated with the tag for that line, rather than discarding the write to the tags. When the NEXT access to that non-cacheable line in L2 occurs, the L1P incorrectly registers this as a hit and transfers data from the L1P rather than the desired external data.

To *avoid* an incorrect update of the L1P tag RAMs, do the following as best practice:

1. While executing code from non-cacheable space, ***do not*** perform either EDMA/QDMA transfers to L2 ***or*** block cache invalidates initiated in L2
2. Mark program code as cacheable as soon as possible.

2.2 Silicon Revision 2.0 Known Design Exceptions to Functional Specifications

Advisory 2.0.1	*PCI: Slave Reads Without Any Byte Enables Issue Target Abort*

Revision(s) Affected: 2.0 and earlier

Details: A slave read transaction that does not assert any byte enables during a data phase will cause the PCI port to respond with a target abort. If a simple master attempts to repeat this transaction until successful, the system will be deadlocked. (Internal reference number DSPvd03632).

Workaround: When doing PCI Slave Reads to the DSP, ensure that at least one byte lane is enabled during every data phase.

Advisory 2.0.2	*I2C: Bus Busy Bit Does Not Reflect the State of the I2C Bus When the I2C is in Reset*

Revision(s) Affected: 2.0 and earlier

Details: The bus busy (BB) bit indicates the status of the I2C bus. The BB bit is set to “1” by a START condition (bus is busy) and set to “0” by a STOP condition (bus is free).The I2C peripheral cannot detect a START or STOP condition when it is in reset (IRS bit set to “0”); therefore, the BB bit will keep the state it was in when the I2C peripheral was placed in reset (when IRS bit is set to “0”) instead of reflecting the actual I2C bus status. The BB bit stays in that state until the I2C peripheral is taken out of reset (IRS bit set to “1”) and a START or STOP condition is detected on the I2C bus. When the device is powered up, the BB bit stays stuck at the default value of “0” until the IRS bit is set to “1” (taking the I2C peripheral out of reset). After the IRS bit is set to “1”, the START or STOP condition can be captured in the BB bit.

Workaround: For multi-master systems, be aware that the BB bit does not reflect the bus status until the I2C peripheral is out of reset (IRS set to “1”) and the first START or STOP condition is detected. Before initiating the first data transfer with the I2C peripheral, follow this sequence:

1. After taking the I2C peripheral out of reset (IRS bit set to “1”), wait a certain period to detect the actual bus status before starting the first data transfer. [The period should be set longer than the total time it takes for the longest data transfer in the application.] Waiting this amount of time after the I2C comes out of reset should ensure at least one START or STOP condition occurred on the I2C bus and captured by the BB bit. After this period, the BB bit will correctly reflect the state of the I2C bus.
2. Poll the BB bit and verify that BB = 0 (bus not busy) before proceeding to next step.
3. Begin data transfers.
4. Do not reset the I2C peripheral between transfers so that the BB bit reflects the actual bus status. If the I2C peripheral must be reset between transfers, repeat steps 1 through 3 ***every*** time the I2C peripheral is taken out of reset (IRS bit set to “1”).

(Internal reference number: 1474)

Advisory 2.0.3	*I2C: Addressed-As-Slave (AAS) Bit is not Cleared Correctly*

Revision(s) Affected: 2.0 and earlier

Details:

The addressed-as-slave (AAS) bit indicates that the I2C peripheral has recognized its own slave address on the I2C bus. In normal (proper) operation for both 7- and 10-bit addressing modes, the AAS bit has the capability to know that its own I2C peripheral has been addressed as a slave and is now capable of transferring/receiving. Also, in normal operation for both addressing modes, the AAS bit is subsequently cleared by receiving a STOP condition *or* by a slave address different from the I2C peripheral's own slave address.

Currently, in the 7-bit addressing mode, the AAS bit *is* cleared when receiving a NACK, a STOP condition, or a repeated START condition. The AAS bit currently is *not* cleared by receiving a slave address different from the I2C peripheral's own slave address.

Currently, in the 10-bit addressing mode, the AAS bit *is* cleared when receiving a NACK, a STOP condition, or by a slave address different from the I2C peripheral's own slave address. The AAS bit, in 10-bit addressing mode, is *not* cleared by a repeated START condition.

For either address mode, the AAS bit is properly set when addressed as a slave.

The only divergence from normal operation is how the AAS bit is cleared in either address modes.

Workaround: None.

Note the AAS bit is set correctly when the I2C peripheral is addressed as a slave for both addressing modes. Also, take into account that when the AAS bit is cleared, the I2C peripheral is no longer addressed as a slave, regardless of addressing mode.

(Internal reference number: 1483)

Advisory 2.0.4	*EMAC: Multi-Channel Mode Ignores Writes to Interrupt Acknowledgement Register*

Revision(s) Affected: 2.0 and earlier

Details: This bug occurs in both TX and RX, when more than one channel is in use. Under multi-channel conditions, the EMAC can “miss” processor writes to the interrupt acknowledgement registers. The specific writes affected are to the RX[0−7]INTACK registers on multi-channel RX, and the TX[0−7]INTACK registers on multi-channel TX.

Use of the EMAC in single-channel mode is not affected. When operating either TX or RX in multi-channel mode (but not the other), only the multi-channel operation is affected.

Workaround: One workaround is to use EMAC only in single-channel mode.

When using the EMAC in multi-channel mode, application software must verify writes to the RX[0−7]INTACK/TX[0−7]INTACK registers by checking that the corresponding bits in the MACINVECTOR register have been cleared. Alternatively, an application my be coded that is tolerant of receiving duplicate interrupts from the EMAC.

The ideal workaround approach is dependent on the software architecture. When using descriptor rings as circular queues, verifying the writes to the ACK registers is the cleanest and safest approach.

When using the CSL HAL to develop driver software, programmers will need to mind this advisory. However, the CSL Ethernet MAC (EMAC) interface functions have the proper workarounds in place.

Advisory 2.0.5

EMIF: Data Corruption can Occur in SDRAM When HOLD Feature is Used

Revision(s) Affected: 2.0 and earlier

Details: When using EMIFA in a system where the HOLD feature is used, data can be corrupted in the SDRAM. When the SDRAM refresh counter within the EMIF expires around the same time a $\overline{\text{HOLD}}$ request is asserted, the DSP starts a refresh of the SDRAM. Before the t_{RFC} specification is met, the DSP generates a DCAB command and asserts $\overline{\text{HOLDA}}$, thus violating t_{RFC} specification for SDRAM.

Workaround: Since both the DSP and the other processor can act as a master, external arbitration logic is needed. There are three possible workarounds:

1. Program the arbitration logic to take care of SDRAM refresh. Disable refresh on DSP. Since the DSP is no longer responsible for refresh of SDRAM, the arbitration logic ensures t_{RFC} specification is not violated.
2. Use one of the DSP internal timers to provide an output signal to the arbitration logic that indicates refresh is pending. The arbitration logic would then be responsible for de-asserting $\overline{\text{HOLD}}$ and starting its own timer to estimate when the refresh operation has completed. Once the timer within the arbitration logic expires, the arbitration logic should assert $\overline{\text{HOLD}}$ if needed.
3. Use two of the DSP internal timers to output two signals that indicate the start and end of a refresh operation to the arbitration logic. The arbitration logic would then be responsible for de-asserting $\overline{\text{HOLD}}$ between the start and end of a refresh operation.

Advisory 2.0.6

EMIF: PDT Write Transfers Fail When PDTWL Equals 3

Revision(s) Affected: 2.0 and earlier

Details: During a PDT write transfer, the $\overline{\text{PDT}}$, $\overline{\text{PDTA}}$, PDTDIR, $\overline{\text{SDWE}}$, and $\overline{\text{SDCAS}}$ signals will not be driven to their appropriate states when a non-PDT write is followed by a PDT write to a different bank. The incorrect behavior of $\overline{\text{PDT}}$, $\overline{\text{PDTA}}$, PDTDIR, $\overline{\text{SDWE}}$, and $\overline{\text{SDCAS}}$ can result in data corruption, as well as bus contention. This only occurs when PDTWL is set equal to 3 in the PDTCTL register.

Workaround: When performing both non-PDT writes and PDT writes to the same CE space, do not set PDTWL equal to 3.

Advisory 2.0.7

PCI: AC Timings Differ From Data Sheet Specifications

Revision(s) Affected: 2.0 and earlier

Details: The timing parameters in Table 2 and Table 3 differ from the ones specified in the *TMS320C6412 Fixed-Point Digital Signal Processor* data manual (literature number SPRS219B or higher). Table 2 and Table 3 list the differences between the data manual values and the actual values on silicon revisions 2.0 and earlier.

Table 2. Timing Requirements for PCI Inputs

NO.			SPRS219B (OLD VALUES)				ACTUAL VALUES REV. 2.0 AND EARLIER				UNIT
			−500		−600 −720		−500		−600 −720		
			33 MHz		66 MHz		33 MHz		66 MHz		
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
5	$t_{h(IV-PCLKH)}$	Hold time, input valid after PCLK high	0		0		0.7		0.7		ns

Table 3. Switching Characteristics Over Recommended Operating Conditions for PCI Outputs

NO.	PARAMETER		SPRS219B (OLD VALUES)				ACTUAL VALUES REV. 2.0 AND EARLIER						UNIT
			−500		−600 −720		−500		A−500		−600 −720		
			33 MHz		66 MHz		33 MHz				66 MHz		
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
1	$t_{d(PCLKH-OV)}$	Delay time, PCLK high to output valid	2	11	2	6	2	11	1.7	11	2	6.6	ns

Workaround: These parameters will be fixed in the next ***major*** silicon revision.

Advisory 2.0.8

EMIF: PDT Transfers Fail When Acessing the Same SDRAM Page as Non-PDT Transfers

Revision(s) Affected: 2.0 and earlier

Details: When PDT and non-PDT transfers occur to the same SDRAM page, $\overline{\text{PDTA}}$, $\overline{\text{PDT}}$, and PDTDIR may not be driven to their appropriate state. The incorrect behavior of these signals can result in PDT data corruption.

Workaround: Place all PDT transfers, whether reads or writes, in a memory range that is an aliased version of the physical SDRAM. For example, if SDRAM is in CE0 and is 128 Mbytes (MB) in depth, then the functional addressable space is 0x8000 0000 through 0x87FF FFFF and all normal CPU and non-PDT DMA transfers should access this memory range. The “aliased” view of the SDRAM is at address 0x8800 0000 through 0x8FFF FFFF and must be used for all PDT transfers. Similarly, if SDRAM is 64 MB in depth, the functional addressable view is 0x8000 0000 through 0x83FF FFFF and the “aliased” view is 0x8400 0000 through 0x87FF FFFF. The aliased view accesses the same underlying physical address as the functional view.

The address space for the “aliased” view can be created by bit-wise ORing the “logical address” (functional address) in use as follows.

- For 128 MB, OR with 0x0800 0000
- For 64 MB, OR with 0x0400 0000
- For 32 MB, OR with 0x0200 0000
- For 16 MB, OR with 0x0100 0000

This workaround is ONLY applicable if the CE space has less than or equal to 128 MB of SDRAM connected to it. If a CE space is full (maximum addressable space is 256 MB), then that CE space cannot support PDT transfers.

Advisory 2.0.9	*PCI: Slave Reads With a Long Latency Can Return Bad Data*

Revision(s) Affected: 2.0 and earlier

Details:

When an external master attempts to read memory from the DSP, it is issued a “Retry”. The PCI will then go prefetch a FIFO’s worth (32 words) of data. If this data takes an exceptionally long time to fetch (approximately 32K PCI cycles, ~1 ms @ 33-MHz PCI), the PCI port can mishandle the return data and “delete” the first word of a PCI frame. The data returned to the master is address-shifted by one 32-bit word. For example, if {0,1,2,3,4, ...} is expected and {1,2,3,4,5 ...} is returned.

Data will only take that long to fetch if the access is to a very slow external memory, or there are large, slow DMAs using the same priority queue as PCI (which, by default, is medium). For performance reasons, the separation of DMA traffic is recommended.

For more detailed information on the EDMA peripheral, EDMA performance, and EDMA performance data, see the following reference guide and application notes:

- *TMS320C6000 DSP Enhanced Direct Memory Access (EDMA) Controller Reference Guide* (literature number SPRU234)
- *TMS320C64x EDMA Architecture* Application Report (literature number SPRA994)
- *TMS320C6000 EDMA IO Scheduling and Performance* Application Report (literature number SPRAA00)
- *TMS320C64x EDMA Performance Data* Application Report (literature number SPRAA02)

PCI slave reads that start in less than 32K PCI cycles will not return bad data.

Workaround:

Do not use PCI to directly read from exceptionally slow external memories.

Do not put any other DMA activity on the same priority level as PCI.

If system considerations force a user to put other DMA activity on PCI’s level, put only DMA traffic that will complete within the 32K PCI clock cycle limit.

Advisory 2.0.10

PCI Reads May Get Ahead of PCI Writes When PCI Exceeds the Transfer Request Limit

Revision(s) Affected: 2.0 and earlier

Details: When the PCI port exceeds its allocation of EDMA Transfer Requests (TRs), it can re−order read and write memory transactions. This can cause PCI slave reads to return “stale” data.

When the PCI port reaches its limit for the number of outstanding TRs, it must wait for previously issued TRs to complete. While the PCI port is waiting, any pending write data and/or read requests will be held pending in internal buffers. When a TR allocation is available, any pending read requests are serviced before any pending write data, regardless of the order in which the write data and read request were received. This can potentially allow the read request to get ahead of the write data, and return “stale” results.

Workaround: TI recommends that good EDMA resource allocation be used to prevent the problem. For detailed information on EDMA resource allocation, refer to the *TMS320C64x EDMA Architecture Application Report* (literature number SPRA994).

The following guidelines can help prevent PCI from running out of available TRs:

- Do not place large or slow transfers on EDMA priority levels at or above the PCI’s priority level.
- Increase PCI’s TR allocation limit. For detailed information, refer to the *TMS320C6000 DSP Peripheral Component Interconnect (PCI) Reference Guide* (literature number SPRU581).

The following guidelines can ensure correct read data:

- Do not read any of the previous 32 words written
- Write 32 words of “dummy” data after writing “real” data

3 Silicon Revision 1.2 Known Design Exceptions to Functional Specifications

3.1 Usage Notes for Silicon Revision 1.2

Silicon Revision 1.2 applicable usage note(s) have been found on a later silicon revision; for more detail, see the *Usage Notes for Silicon Revision 2.0* section of this document.

Advisory 1.2.2	*EMU: Data Corruption With RTDX and Real-Time Emulation Memory Read*

Revision(s) Affected: 1.2 and earlier

Details: This advisory impacts emulation accesses including JTAG Real-Time Data Exchange (RTDX™), HSRTDX, and real-time emulation memory reads.

If using one or more of the aforementioned impacted emulation functions, you may experience data corruption when performing emulation read requests 2 cycles after L1D has stalled due to a snoop stall, or when the last CPU access is a falsely predicated read request where the predication bit is zero and is ignored by L1D.

When the above condition is true, L1D incorrectly interprets the CPU read request as an emulation request, and assumes the predication (normally true) is intended for the emulation request. As a result, L1D ignores the emulation request.

Because the CPU still expects a data return to the active pipeline cycle, it reads the last data from the read bus, which can cause an update halt and create RTDX corruption (DSPvd03642).

Workaround: For workaround suggestions on how to reduce (minimize) the chances of receiving corrupted data, please see the release notes provided with CCS C6000 2.12.10 [Code Composer Studio™ Integrated Development Environment (IDE) TMS320C64x Silicon Revision 1.1 Chip Support Package (CSP)]. These workaround suggestions are discussed in release note #12 — “SDSsq27324: DSP/BIOS™ Real-Time Analysis (RTA) Update Halt and Real-Time Data Exchange (RTDX) Data Corruption”.

RTDX, Code Composer Studio, and DSP/BIOS are trademarks of Texas Instruments.

Advisory 1.2.8

EMIF: Programmable Synchronous Interface AC Timings Differ From Data Sheet Specifications

Revision(s) Affected: 1.2 and earlier

Details: The timing parameters in Table 4 and Table 5 differ from the ones specified in the *TMS320C6412 Fixed-Point Digital Signal Processor* data manual (literature number SPRS219B or higher). Table 4 and Table 5 list the differences between the data manual values and the actual values on silicon revisions 1.2 and earlier.

Table 4. Timing Requirements for Programmable Synchronous Interface Cycles for EMIFA Module

NO.			SPRS219B (OLD VALUES)				ACTUAL VALUES REV. 1.2 AND EARLIER				UNIT
			−500		−600		−500		−600		
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
7	$t_{h(EKOxH-EDV)}$	Hold time, read AEDx valid after AECLKOUTx high	1.5		1.5		2.0		2.0		ns

Table 5. Switching Characteristics Over Recommended Operating Conditions for Programmable Synchronous Interface Cycles for EMIFA Module

NO.	PARAMETER		SPRS219B (OLD VALUES)				ACTUAL VALUES REV. 1.2 AND EARLIER						UNIT
			−500		−600		A-500		−500		−600		
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
1	$t_{d(EKOxH-CEV)}$	Delay time, AECLKOUTx high to $\overline{ACEx}$ valid	1.3	6.4	1.3	4.9	0.8	6.5	1.0	6.5	1.0	5.0	ns
2	$t_{d(EKOxH-BEV)}$	Delay time, AECLKOUTx high to $\overline{ABEx}$ valid		6.4		4.9		6.5		6.5		5.0	ns
3	$t_{d(EKOxH-BEIV)}$	Delay time, AECLKOUTx high to $\overline{ABEx}$ invalid	1.3		1.3		0.8		1.0		1.0		ns
4	$t_{d(EKOxH-EAV)}$	Delay time, AECLKOUTx high to AEAx valid		6.4		4.9		6.5		6.5		5.0	ns
5	$t_{d(EKOxH-EAIV)}$	Delay time, AECLKOUTx high to AEAx invalid	1.3		1.3		0.8		1.0		1.0		ns
8	$t_{d(EKOxH-ADSV)}$	Delay time, AECLKOUTx high to $\overline{ASADS}/\overline{ASRE}$ valid	1.3	6.4	1.3	4.9	0.8	6.5	1.0	6.5	1.0	5.0	ns
9	$t_{d(EKOxH-OEV)}$	Delay time, AECLKOUTx high to $\overline{ASOE}$ valid	1.3	6.4	1.3	4.9	0.8	6.5	1.0	6.5	1.0	5.0	ns
10	$t_{d(EKOxH-EDV)}$	Delay time, AECLKOUTx high to AEDx valid		6.4		4.9		6.5		6.5		5.0	ns
11	$t_{d(EKOxH-EDIV)}$	Delay time, AECLKOUTx high to AEDx invalid	1.3		1.3		0.8		1.0		1.0		ns
12	$t_{d(EKOxH-WEV)}$	Delay time, AECLKOUTx high to $\overline{ASWE}$ valid	1.3	6.4	1.3	4.9	0.8	6.5	1.0	6.5	1.0	5.0	ns

Workaround: These parameters will be improved in the next ***major*** silicon revision.

Advisory 1.2.9 *EMIF: Synchronous DRAM AC Timings Differ From Data Sheet Specifications*

Revision(s) Affected: 1.2 and earlier

Details: The timing parameters in Table 6 and Table 7 differ from the ones specified in the *TMS320C6412 Fixed-Point Digital Signal Processor* data manual (literature number SPRS219B or higher). Table 6 and Table 7 list the differences between the data manual values and the actual values on silicon revisions 1.2 and earlier.

Table 6. Timing Requirements for Synchronous DRAM Cycles for EMIFA Module

NO.			SPRS219B (OLD VALUES)				ACTUAL VALUES REV. 1.2 AND EARLIER				UNIT
			−500		−600		−500		−600		
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
6	$t_{su(EDV-EKO1H)}$	Setup time, read AEDx valid before AECLKOUTx high	2.1		0.6		2.1		0.7		ns
7	$t_{h(EKO1H-EDV)}$	Hold time, read AEDx valid after AECLKOUTx high	2.5		1.8		3.0		2.3		ns

Table 7. Switching Characteristics Over Recommended Operating Conditions for Synchronous DRAM Cycles for EMIFA Module

NO.	PARAMETER		SPRS219B (OLD VALUES)				ACTUAL VALUES REV. 1.2 AND EARLIER						UNIT
			−500		−600		A-500		−500		−600		
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
1	$t_{d(EKO1H-CEV)}$	Delay time, AECLKOUTx high to $\overline{ACEx}$ valid	1.3	6.4	1.3	4.9	0.9	6.5	1.1	6.5	1.1	5.0	ns
2	$t_{d(EKO1H-BEV)}$	Delay time, AECLKOUTx high to $\overline{ABEx}$ valid		6.4		4.9		6.5		6.5		5.0	ns
3	$t_{d(EKO1H-BEIV)}$	Delay time, AECLKOUTx high to $\overline{ABEx}$ invalid	1.3		1.3		0.9		1.1		1.1		ns
4	$t_{d(EKO1H-EAV)}$	Delay time, AECLKOUTx high to AEAx valid		6.4		4.9		6.5		6.5		5.0	ns
5	$t_{d(EKO1H-EAIV)}$	Delay time, AECLKOUTx high to AEAx invalid	1.3		1.3		0.9		1.1		1.1		ns
8	$t_{d(EKO1H-CASV)}$	Delay time, AECLKOUTx high to $\overline{ASDCAS}$ valid	1.3	6.4	1.3	4.9	0.9	6.5	1.1	6.5	1.1	5.0	ns
9	$t_{d(EKO1H-EDV)}$	Delay time, AECLKOUTx high to AEDx valid		6.4		4.9		6.5		6.5		5.0	ns
10	$t_{d(EKO1H-EDIV)}$	Delay time, AECLKOUTx high to AEDx invalid	1.3		1.3		0.9		1.1		1.1		ns
11	$t_{d(EKO1H-WEV)}$	Delay time, AECLKOUTx high to $\overline{ASDWE}$ valid	1.3	6.4	1.3	4.9	0.9	6.5	1.1	6.5	1.1	5.0	ns
12	$t_{d(EKO1H-RAS)}$	Delay time, AECLKOUTx high to $\overline{ASDRAS}$ valid	1.3	6.4	1.3	4.9	0.9	6.5	1.1	6.5	1.1	5.0	ns
13	$t_{d(EKO1H-ACKEV)}$	Delay time, AECLKOUTx high to ASDCKE valid	1.3	6.4	1.3	4.9	0.9	6.5	1.1	6.5	1.1	5.0	ns
14	$t_{d(EKO1H-PDTV)}$	Delay time, AECLKOUTx high to $\overline{APDT}$ valid	1.3	6.4	1.3	4.9	0.9	6.5	1.1	6.5	1.1	5.0	ns

EMIF: Synchronous DRAM AC Timings Differ From Data Sheet Specifications (Continued)

Workaround: These parameters will be improved in the next ***major*** silicon revision.

Advisory 1.2.11 *Interrupts: NMI not Functional if Both PCI/HPI Modules are Disabled*

Revision(s) Affected: 1.2 and earlier

Details: The Non-Maskable Interrupt (NMI) will *not* propagate to the interrupt module if and only if both the PCI and the HPI peripheral modules are disabled on the device.

The following boot configuration applies to this condition wherein the PCI and HPI peripherals are disabled and the EMAC peripheral and GP0[15:9] pins are enabled:

- PCI_EN = 0
- PCI_EEAI = 0
- HD5 = 1 (HPI_WIDTH = 32 bits)
- MAC_EN = 1 (EMAC/MDIO enabled)

The NMI will work properly and propagate to the interrupt module for all other values of the PCI_EN, PCI_EEAI, HD5, and MAC_EN pins (see Table 8).

Workaround: Both a software and hardware workaround exist:

Software Workaround: A software workaround exists to capture the NMI from the “0011” bootmode configuration (PCI/HPI disabled while EMAC/MDIO and GP0[15:9] are enabled).

The following software steps could be part of the boot code:

1. Write a value of 0xC010 0C01 to the Peripheral Configuration Lock register (PCFGLOCK) [address 0x01B3 F018], which will enable the NMI to be recognized by the interrupt module.
2. Read bit 1 from the PCFGLOCK register. It should return a “1”.

Hardware Workaround: For the hardware workaround, any of the boot configurations in Table 8 can be used.

Table 8. Boot Configurations with NMI Functioning Properly

PCI_EN	PCI_EEAI	HD5	MAC_EN
0	0	0	0
0	0	0	1†
0	0	1	0
1	1	x	x
1	0	x	x

† When the EMAC/MDIO peripherals are used, HPI16 mode ***must*** be enabled.

For more detailed information on configuration of the device, see the *Device Configuration* section of the device-specific data sheet.

Advisory 1.2.12 *L2 Cache: Accesses to Mapped L2 RAM Update L2 LRU Information*

Revision(s) Affected: 1.2 and earlier

Details:

CPU accesses to L2 RAM addresses incorrectly cause updates to the “Least-Recently Used” (LRU) state information in the L2 cache. This may cause an increased number of L2 cache misses in some systems.

The L2 cache implements a 4-way set-associative cache. The cache uses the (LRU) information to determine what “way” within each set is least recently used. When the CPU accesses data in L2 cache, the L2 controller determines what “way” holds the data in that set, and marks that “way” as most-recent. When allocating a new line in the cache, the L2 controller evicts the line in the set from the least-recently used “way” under the assumption that more-recently accessed data is more relevant.

When the CPU accesses data in L2 SRAM, either via program fetches or data accesses, the L2 cache is incorrectly updated by the L2 controller. The L2 controller updates the LRU as if the access was to “way 0” in the cache. This causes the LRU history to *not* reflect the actual sequence of accesses to L2 cache. As a result, the L2 may not choose the actual least-recently used line during an eviction.

Only CPU accesses to L2 RAM cause this update to LRU information in L2 cache. DMA accesses to L2 RAM *do not* trigger updates to the L2 LRU information.

Workaround:

Perform one of the following three workarounds:

- Choose an L2 cache size that fits your cached working set. The advisory primarily impacts programs that are significantly larger than the L2 cache size.
- Explicitly remove cached contents from L2 when finished with them. The L2 cache allocates “invalid” lines within each set before consulting the LRU. Programs may do this using “block invalidate” or “block writeback-invalidate” commands in the L2 cache.
- Lay out buffers in L2 RAM so they *do not* conflict with buffers or code held in L2 cache. L2 RAM addresses map onto L2 sets in the same manner as external memory addresses.

For more detailed information on the organization and manipulation of the L2 cache, see the *TMS320C64x DSP Two-Level Internal Memory Reference Guide* (literature number SPRU610).

Advisory 1.2.13	*L2 Cache: L2 Controller Incorrectly Updates LRU for Accesses in L2 Cache*

Revision(s) Affected: 1.2 and earlier

Details:

The L2 cache implements a 4-way set-associative cache. The cache uses the “Least-Recently Used” (LRU) information to determine what “way” within each set is least recently used. When the CPU accesses data in L2 cache, the L2 controller determines what “way” holds the data in that set, and marks that “way” as most-recent. When allocating a new line in the cache, the L2 controller evicts the line in the set from the least-recently used “way” under the assumption that more-recently accessed data is more relevant.

For this advisory, CPU accesses which hit L2 cache *do not* correctly update the LRU information for the set accessed. Instead of storing the LRU information back to the set being accessed, the L2 controller stores the information to 3 adjacent sets.

LRU information is stored in groups of 4 sets. The 3 adjacent sets affected by the current set are defined as follows:

- Group 1 contains sets 0, 1, 2, 3
- Group 2 contains sets 4, 5, 6, 7
- Etc.

For example, during an access to set 5, the L2 controller incorrectly stores the LRU information to sets 4, 6, 7.

As a result of this issue, repeated misses to the same set with no intervening accesses to adjacent sets will allocate from the same “way”. This can make the L2 cache appear to “thrash”. A series of misses to consecutive sets in L2 cache may appear to allocate with reduced associativity; that is, L2 could appear to behave as a 2-way or direct-mapped cache.

Workaround:

Perform one of the following three workarounds:

- Choose an L2 cache size that fits your cached working set. The advisory primarily impacts programs that are significantly larger than the L2 cache size.
- Explicitly remove cached contents from L2 when finished with them. The L2 cache allocates “invalid” lines within each set before consulting the LRU. Programs may do this using “block invalidate” or “block writeback-invalidate” commands in the L2 cache.
- Offset external buffers that are accessed as part of the same working set so that accesses to the buffers are at least 4 L2 sets apart (512 bytes). This will prevent the buffers from ”thrashing” each other in L2 cache.

For more detailed information on the organization and manipulation of the L2 cache, see the *TMS320C64x DSP Two-Level Internal Memory Reference Guide* (literature number SPRU610).

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

4.1 Usage Notes for Silicon Revision 1.1

Silicon Revision 1.1 applicable usage note(s) have been found on a later silicon revision; for more detail, see the *Usage Notes for Silicon Revision 2.0* section of this document.

4.2 Silicon Revision 1.1 Known Design Exceptions to Functional Specifications

All known design exceptions to functional specifications for silicon revision 1.1 still apply and have been moved up to the *Silicon Revision 2.0 Known Design Exceptions to Functional Specifications section and Silicon Revision 1.2 Known Design Exceptions to Functional Specifications* section of this document.

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

5.1 Usage Notes for Silicon Revision 1.0

Silicon Revision 1.0 applicable usage note(s) have been found on a later silicon revision; for more detail, see the *Usage Notes for Silicon Revision 2.0* section of this document.

5.2 Silicon Revision 1.0 Known Design Exceptions to Functional Specifications

All known design exceptions to functional specifications for silicon revision 1.0 still apply and have been moved up to the *Silicon Revision 2.0 Known Design Exceptions to Functional Specifications* section and *Silicon Revision 1.2 Known Design Exceptions to Functional Specifications* section of this document.

IMPORTANT NOTICE

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

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

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

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

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

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

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

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

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

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

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2010, Texas Instruments Incorporated