TMS320C6411

Digital Signal Processor

Silicon Errata

Silicon Revisions 1.1, 2.0

SPRZ194L
February 2003
Revised August 2005

TEXAS INSTRUMENTS

REVISION HISTORY

This silicon errata revision history highlights the technical changes made to the SPRZ194K revision to make it an SPRZ194L revision.

Scope: Applicable updates to the C64x device family, specifically relating to the C6411 device, have been incorporated. TMS320C6411 silicon revision 2.0 updates have been incorporated.

PAGE(S) NO.	ADDITIONS/CHANGES/DELETIONS
12	Silicon Revision 2.0 Known Design Exceptions to Functional Specifications section: Added Advisory 2.0.5 − "PCI Reads May Get Ahead of PCI Writes When PCI Exceeds the Transfer Request Limit"

Contents

1 Introduction

This document describes the known exceptions to the functional specifications for the TMS320C6411 digital signal processor. [See the *TMS320C6411 Fixed-Point Digital Signal Processor* data sheet (literature number SPRS196).]

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 DSP devices and support tools. Each DSP commercial family member has one of three prefixes: TMX, TMP, or TMS. (i.e., **TMS**320C6411GLZ) 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.”

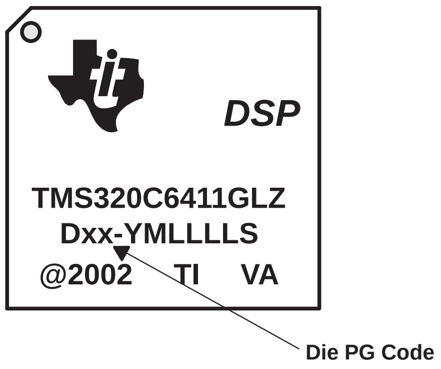
TMS320 is a trademark of Texas Instruments.
All trademarks are the property of their respective owners.

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

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

1.2 Revision Identification

The device revision can be determined by the Die PG code marked on the top of the package. The location of the Die PG code for the GLZ package is shown in Figure 1. Figure 1 shows an example of the types of C6411 package symbolization.



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, Die PG Codes for TMS320C6411 (GLZ)

Silicon revision is identified by a code on the chip. The code is of the format Dxx-YMLLLLS or Cxx-YMLLLLS, etc. If xx is 11, then the silicon is revision 1.1; if xx is 20, then the silicon is revision 2.0, etc.

Table 1. Die PG Codes

Die PG Code (xx)	Silicon Revision	Comments
11	1.1	TMS320C6411
20	2.0	TMS320C6411A

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.

HPI: HWOB Bit in HPIC Register is Writable by CPU

On silicon revision 2.0 and earlier, the HWOB bit in the HPIC register is writable by the CPU, even though it should be writable by the Host only. Writing the incorrect value to the HPIC register will corrupt any data passed through the HPI.

When performing HPIC writes via the CPU, be sure to write the correct value into the HWOB bit.

TCP/VCP: TCP/VCP Memory Address Range Must be Accessed Using Only Doublewords (All C64x Devices, Including C6411)

On silicon revision 2.0 and earlier, the TCP/VCP memory address range (0x5000 0000 to 0x5FFF FFFF) must always be accessed by doubleword requests. Even if the C64x device does not support the TCP/VCP peripherals or if the TCP/VCP coprocessors are not enabled, this memory address range must *always* be accessed via doublewords. A request of any other size will cause the chip to hang. Users should be especially careful when using Code Composer Studio™ Integrated Development Environment (IDE), because Code Composer Studio requests memory data using word-sized accesses. Do not attempt to view this memory map region (0x5000 0000 to 0x5FFF FFFF) in a Code Composer Studio window. (Note: this region can be disabled in the Code Composer Studio memory map to prevent accidental access.)

For all TMS320C64x™ DSP devices, always access the memory address range 0x5000 0000 to 0x5FFF FFFF via doublewords.

EMIF: Dead CLKOUT4/6

On silicon revision 2.0 and earlier, there is a usage condition concerning EMIF which can affect the functionality of CLKOUT4 and CLKOUT6.

The EMIF Global Control register (GBLCTL) controls the logic that outputs the internal CPU/4 or CPU/6 clocks to the CLKOUT4 and CLKOUT6 pins. The GBLCTL is registered with the ECLKIN clock; therefore, without a valid EMIF clock, it is possible to have unknown values in the GBLCTL register. Furthermore, without a valid EMIF clock, the EMIF GBLCTL will not be reset to its default value.

To avoid a dead CLKOUT4 and/or CLKOUT6, a valid clock must be provided to EMIF at all times. This can be implemented externally using ECLKIN or by setting the EMIF_CLK_SEL[1:0] pins to select the internal CPU/4 or CPU/6.

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

On C6411 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

For the C6411 device there is ***no*** current plan to have any further *major* silicon updates; therefore, all silicon revision 2.0 known design exceptions to functional specifications identified will be how the device functionally operates and will ***not*** be changed/corrected.

Advisory 2.0.1 — *L2 Cache: Accesses to Mapped L2 RAM Update L2 LRU Information*

Revision(s) Affected: 2.0 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 2.0.2 — *L2 Cache: L2 Controller Incorrectly Updates LRU for Accesses in L2 Cache*

Revision(s) Affected: 2.0 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).

Advisory 2.0.3 — *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 2.0.4 — *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 2.0.5 — *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 1.1 Known Design Exceptions to Functional Specifications and Usage Notes

3.1 Usage Notes for Silicon Revision 1.1

All usage notes for silicon revision 1.1 still apply and have been moved up to the *Usage Notes for Silicon Revision 2.0* section of this document.

3.2 Silicon Revision 1.1 Known Design Exceptions to Functional Specifications

Advisory 1.1.1 — *EMIF: CLKOUT6 may not be CPU/6 During Reset*

Revision(s) Affected: 1.1

Details: At power up, before the rising edge of $\overline{\text{RESET}}$, there is a chance that CLKOUT6 will not be CPU/6. When this happens, the CLKOUT6 will be CPU/2 until the rising edge of $\overline{\text{RESET}}$. Once the first rising edge of $\overline{\text{RESET}}$ is detected, CLKOUT6 will remain CPU/6 until power is removed from the device. (Internal reference number DSPvd03612).

Workaround: If correct CLKOUT6 is important to the system at all times, two resets can be performed on the DSP. The first will ensure a correct CLKOUT6 and the second can then be used for system reset activities.

Advisory 1.1.2 — *PCI: Slave Writes With Null Data Phases Can Lock Up PCI*

Revision(s) Affected: 1.1

Details: When performing slave writes to the DSP, the PCI port is susceptible to lockup if a word is written without any byte enables asserted. The PCI port will always disconnect a transfer when it detects a null data phase. If this null data phase coincides with a particular internal FIFO state, then the PCI port will also disconnect all future accesses. (Internal reference number DSPvd03620).

Workaround: When transferring data using slave writes to the DSP, ensure at least one byte is enabled in every data phase.

Advisory 1.1.3 — *PCI: Slave Writes Can Corrupt Data*

Revision(s) Affected: 1.1

Details: When performing slave writes to the DSP through the PCI port, it is possible for the data to be corrupted. If the end of the PCI frame coincides with a particular internal FIFO state, then the last word of the burst will overwrite the first word of the burst. The location in memory where the last word should have gone is left unmodified. Only slave writes that burst longer than 4 words are affected. (Internal reference number DSPvd03632).

Workaround: There are several things that can be done to work around this issue. Any one of the following proven workarounds will always prevent the corruption:

- Limit slave-write burst sizes to no more than 4 words
- Alter the system to use master reads instead of slave writes to bring data into the DSP
- Implement an end-to-end data verification scheme to verify and correct any data that was corrupted

If these are not possible or desired, there is an alternate workaround that mitigates the risk, but does not eliminate the possibility, of data corruption. Since the alignment of the end of the PCI frame and the problematic FIFO state will likely only happen for certain burst sizes for a given PCI-to-CPU clock ratio. If the data corruption problem is observed, then altering the CPU speed while keeping the PCI speed constant can alter the timing of the two events (end of PCI frame and FIFO state) such that the problem no longer occurs. If the problem is not observed in a system, a slightly different traffic pattern can cause the problem to appear. It is therefore crucial to ensure that PCI and EDMA traffic are representative of actual system usage conditions when using this workaround.

Advisory 1.1.4 — *PCI: Slave Reads Without Any Byte Enables Issue Target Abort*

Revision(s) Affected: 1.1

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 1.1.5 *PCI: Master Transaction Following an Abort Can Erroneously Set MASTEROK*

Revision(s) Affected: 1.1

Details: If an abort of any kind is received (Master abort, Target abort, or writing “0” to the START bits) and is followed by a master transaction, then the master transaction can set the MASTEROK interrupt bit (PCIIS.6) before the transfer has actually completed. (Internal Reference number DSPvd03633).

Workaround: When processing a MASTEROK interrupt, check the START bits to ensure the transaction has actually completed.

Advisory 1.1.6 *PCI: An Abort Received During a Master Read Can Lock Up PCI*

Revision(s) Affected: 1.1 and earlier

Details: If an abort of any kind is received (Master abort, Target abort, or writing “0” to the START bits) while performing a master read, then the PCI port can be put into a state where no further transactions are possible. The PCI port disconnects all transactions. (Internal reference number DSPvd03434).

Workaround: If this PCI lock-up occurs, reset and reconfigure the PCI port.

Advisory 1.1.7 *PCI: Do Not Write a “1” to the Cache Line Size Register*

Revision(s) Affected: 1.1

Details: If the Cache Line Size Configuration register contains the value of “1”, the PCI port will not function correctly.

Workaround: Do not write a “1” to the Cache Line Size Configuration register.

Advisory 1.1.8 — *HPI: Simultaneous Host and CPU Writes to HPIC Register Conflict*

Revision(s) Affected: 1.1

Details: When a CPU write to the HPIC register and Host write to the HPIC register arrive at the same time, the writes are merged, which may produce unintended consequences. The following shows which write takes precedence for the bits in the HPIC register.

0	HWOB:	Write from Host
1	DSPINT:	Write from DSP (CPU)
2	HINT:	Write from Host
3	HRDY:	N/A
4	FETCH:	Write from Host
5−14	RVSD:	Write from Host
15	RVSD:	Both

Simultaneous access to the HPIC register can cause trouble when trying to set DSPINT or HINT. The side without priority writes a “1” to set, and the side with priority writes a “0” for no change. Thus, only the priority write of “0” takes effect, and the bit is ***not*** set.

Workaround: Repeatedly write to the HPIC register until the correct value is observed.

Advisory 1.1.9 — *EMU: Emulation Prone to Failure Under Certain Situations*

Revision(s) Affected: 1.1

Details: Under certain conditions, the emulation hardware may corrupt the emulation control state machine or may cause it to lose synchronization with the emulator software. When emulation commands fail as a result of the problem, Code Composer Studio Integrated Development Environment (IDE) may be unable to start or it may report errors when interacting with the C64x™ DSP (for example, when halting the CPU, reaching a breakpoint, etc.).

This phenomenon is observed when an erroneous clock edge is generated from the TCK signal inside the C64x DSP. This can be caused by several factors, acting independently or cumulatively:

- TCK transition times (as measured between 2.5 V and 0.6 V) in excess of 3 ns.
- Operating the C64x DSP in a socket, which can aggravate noise or glitches on the TCK input.
- Writing changing bit patterns to the upper 16 bits of EMIFA during emulation transactions. The upper bits of EMIFA are located near TCK on the device and can affect the TCK signal.
- Poor signal integrity on the TCK line from reflections or other layout issues.

A TCK edge that can cause this problem might look similar to the one shown in Figure 2. A TCK edge that does not cause the problem will look similar to the one shown in Figure 3. The key difference between the two figures is that Figure 3 has a clean and sharp transition whereas Figure 2 has a “knee” in the transition zone. Problematic TCK signals may not have a knee that is as pronounced as the one in Figure 2. Due to the TCK signal amplification inside the chip, any perturbation of the signal can create erroneous clock edges.

C64x is a trademark of Texas Instruments.

EMU: Emulation Prone to Failure Under Certain Situations (Continued)

As a result of the faster edge transition, there is increased ringing in Figure 3. As long as the ringing does not cross logic input thresholds (0.6 V for falling edges, and 2.5 V for rising edges), this ringing is acceptable.

When examining a TCK signal for this issue, either in board simulation or on an actual board, it is very important to probe the TCK line as close to the DSP input pin as possible. In simulation, it should not be difficult to probe right at the DSP input. For most physical boards, this means using the via for the TCK pad on the back side of the board. Similarly, ground for the probe should come from one of the nearby ground pad vias to minimize EMI noise picked up by the probe.

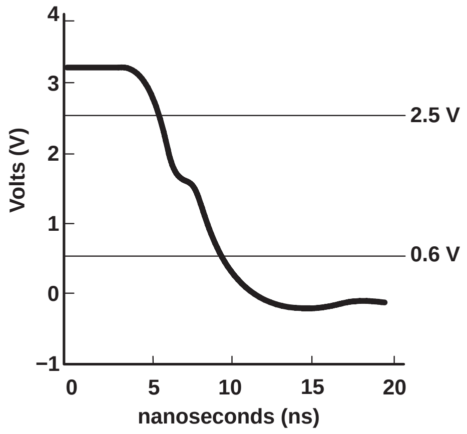


Figure 2. Bad TCK Transition

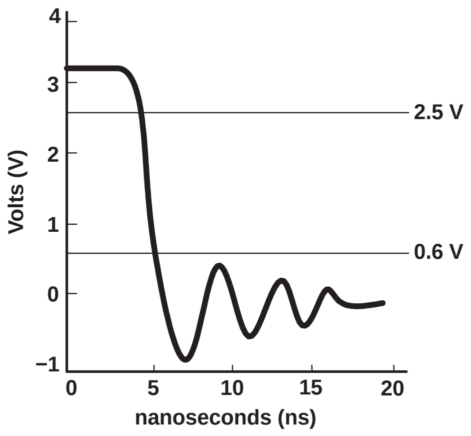


Figure 3. Good TCK Transition

EMU: Emulation Prone to Failure Under Certain Situations (Continued)

Workaround: As the problem may be caused by one or more of the above factors, one or more of the steps outlined below may be necessary to fix it:

- Do not toggle AED[63:48] by either the CPU or EDMA near the time of a breakpoint. This may be difficult to control. Effective implementation may require restricting external accesses to 32 bits or fewer.
- Avoid using a socket
- Ensure the board design achieves rise times and fall times of less than 3 ns with clean monotonic edges for the TCK signal.
- For designs where TCK is supplied by the emulation pod, use a C64x Emulation Adapter Board, part number DSP8102U. To order a C64x Emulation Adapter Board, please contact the TI Product Information Center (PIC).

Advisory 1.1.10 *EMU: Driver-Induced HSRTDX Data Corruption*

Revision(s) Affected: 1.1

Details: There is a known issue with the drivers released in Code Composer Studio version 2.12.10 that causes corruption of HSRTDX data at high speeds. The data corruption occurs when TCLK is operating at around 35 MHz and HSRTDX is configured to utilize EMU1 as the transport channel.

Workaround: Use EMU0 as the transport channel when using HSRTDX; however, if using EMU0 is not possible or if EMU0 exhibits similar corruption, then it is suggested that the TCLK frequency be lowered to a frequency no greater than 30 MHz.

To lower the TCLK frequency when using an XDS560™ emulator:

1. Invoke CCSetup
2. Right-click on the appropriate XDS560-enabled system in the left-most pane and select “Properties”
3. Click on the “Board Properties” Tab
4. Select a value of “User Defined” for the “TCLK” property
5. For the JTAG clock rate, enter a value that corresponds to twice the desired TCLK frequency. For example, if a 25-MHz TCLK is desired, then enter “50” into the “JTAG Clock Rate (2 * JTAG Clock Freq.)” property field.

Advisory 1.1.11 — *EMU: Data Corruption with RTDX and Real-Time Emulation Memory Read*

Revision(s) Affected: 1.1

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™ 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”).

Advisory 1.1.12 — *PLL: PLL May Fail to Oscillate on Startup*

Revision(s) Affected: 1.1

Details: A power-on reset sequence has been defined. Power up the I/O power supply (DV_{DD}) before the Core power supply (CV_{DD}) to ensure proper PLL start-up and operation.

Workaround: The following is the processor power-up sequence and timing:

1. DV_{DD} power supply
2. CV_{DD} power supply

timing requirements for power-on sequence[†] (see Figure 4)

NO.			−5E0 A−5E0 −6E3		UNIT
			MIN	MAX	
1	$t_{d(DVDDR-CVDD)}$	Delay time, DV_{DD} supply ready to CV_{DD} supply ramp start	0.5	200	ms

† Delay time from DV_{DD} is referenced to the supply reaching its minimum operating voltage.

RTDX and DSP/BIOS are trademarks of Texas Instruments.

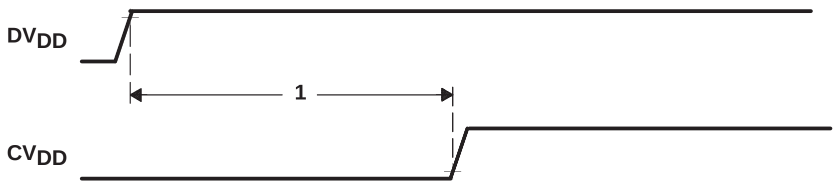


Figure 4. Power-On Sequence Timing

Advisory 1.1.13 — *PCI: PCI Power Management Sticky Bits Always “1”*

Revision(s) Affected: 1.1

Details: The Power Management Control/Status Register (PMCSR) sticky bits, PME_STATUS and PME_EN, are always set to ’1’ indicating a power management event (PME) has occurred, even though the C64x devices do not support PMEs. A host write to clear the PME_STATUS and PME_EN bits will not be successful. The host’s failure to clear the sticky bits may cause some PCI BIOS software to throw fatal exceptions and/or refuse to boot.

Workaround: None.

Advisory 1.1.14 — *EMIF: Data Corruption can Occur in SDRAM When HOLD Feature is Used*

Revision(s) Affected: 1.1

Details: When using the EMIF in a system where the $\overline{\text{HOLD}}$ feature is used, data can be corrupted in the SDRAM that is on the same EMIF where $\overline{\text{HOLD}}$ is used. 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.

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

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 1.1.15 — *EMIF: PDT Write Transfers Fail When PDTWL Equals 3*

Revision(s) Affected: 1.1

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 1.1.16 — *PLL: PLL May Stop Oscillating if I/O Supply Falls Out of Specification*

Revision(s) Affected: 1.1

Details: If the I/O power supply (DV_{DD}) drops below 2.5 V, the PLL may stop oscillating and the DSP will not operate. If this occurs, DV_{DD} and core power supply (CV_{DD}) must be completely powered down and the DSP must be powered up.

Workaround: Do not allow the I/O power supply to drop below 2.5 V.

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

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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