TMS320C6414, TMS320C6415, and TMS320C6416 Digital Signal Processors Silicon Errata

Silicon Revisions 1.0, 1.01, 1.02, 1.03, 1.1, 2.0

SPRZ011T
October 2001
Revised August 2007

REVISION HISTORY

This silicon errata revision history highlights the technical changes made to the SPRZ011S revision to make it an SPRZ011T revision.

Scope: Applicable updates to the C64x device family, specifically relating to the C6414/C6415/C6416 devices, have been incorporated. TMS320C6414/C6415/C6416 silicon revision 2.0 updates have been incorporated.

PAGE(S) NO.	ADDITIONS/CHANGES/DELETIONS
16	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 (C6415/C6416 Only)”

Contents

1 Introduction

This document describes the known exceptions to the functional specifications for the TMS320C6414, TMS320C6415, and TMS320C6416 digital signal processors. [See the *TMS320C6414, TMS320C6415, TMS320C6416 Fixed-Point Digital Signal Processors* data sheet (literature number SPRS146).] Throughout this document, TMS320C64x and C64x refer to TMS320C6414, TMS320C6415, and TMS320C6416.

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:

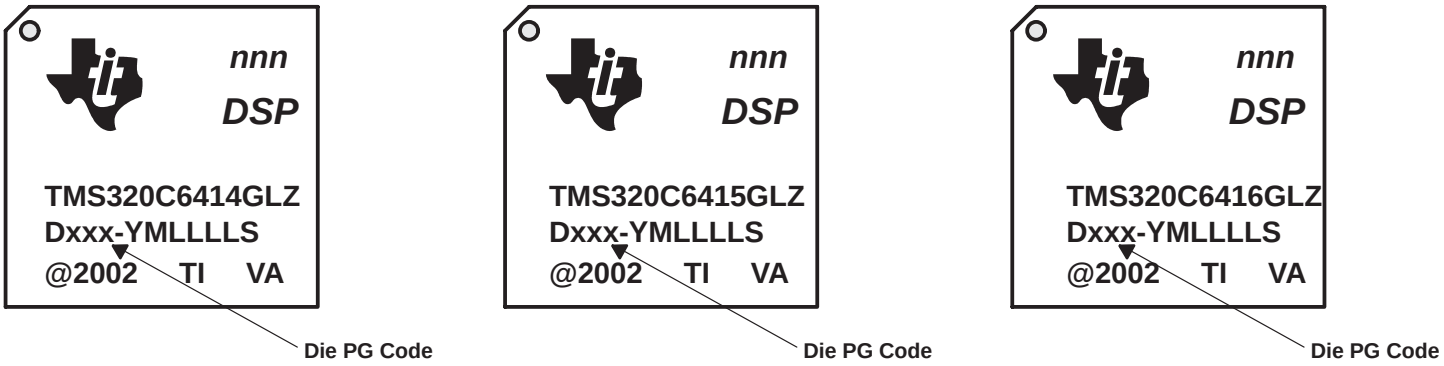
“Developmental product is intended for internal evaluation purposes.”

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 some examples of the types of C6414, C6415, and C6416 package symbolization.



"nnn" represents the device speed. For example:

5E0 = 1.2 V, 500-MHz Core, 100-MHz EMIF
A5E0 = 1.25 V, 500-MHz Core, 100-MHz EMIF, Extended Temperature
Blank = 1.4 V, 600-MHz Core, 133-MHz EMIF
A6E3 = 1.4 V, 600-MHz Core, 133-MHz EMIF, Extended Temperature
7E3 = 1.4 V, 720-MHz Core, 133-MHz EMIF

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 TMS320C6414, TMS320C6415, and TMS320C6416 (GLZ)

Silicon revision is identified by a code on the chip. The code is of the format Dxxx-YMLLLLS or Cxxx-YMLLLLS, etc. If xxx is 101, then the silicon is revision 1.01; if xxx is 102, then the silicon is revision 1.02, etc.

Table 1. Die PG Codes

Die PG Code (xxx)	Silicon Revision	Comments
10	1.0	TMS320C6414, TMS320C6415, and TMS320C6416
101	1.01	TMS320C6414, TMS320C6415, and TMS320C6416 Silicon Revision 1.01 is functionally the same as Revision 1.0
102	1.02	TMS320C6414, TMS320C6415, and TMS320C6416 Silicon Revision 1.02 is functionally the same as Revision 1.0
103	1.03	TMS32C6414C, TMS32C6415C, and TMS32C6416C
11	1.1	TMS32C6414D, TMS32C6415D, and TMS32C6416D
20	2.0	TMS32C6414E, TMS32C6415E, and TMS32C6416E

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)

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.

EMIFA: Dead CLKOUT4/6

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

The EMIFA 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 AECLKIN 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 EMIFA GBLCTL will not be reset to its default value.

To avoid a dead CLKOUT4 and/or CLKOUT6, a valid clock must be provided to EMIFA at all times. This can be implemented externally using AECLKIN or by setting the BEA[17:16] pins (EMIFA_CLK_SEL bits) to select the internal CPU/4 or CPU/6.

Code Composer Studio and TMS320C64x are trademarks of Texas Instruments.

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

On C6414/C6415/C6416 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 C6414/C6415/C6416 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 (C6415/C6416 Only)*

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 (C6415/C6416 Only)

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 and earlier

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 (C6415/C6416 Only)*

Revision(s) Affected: 1.1 and earlier

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 (C6415/C6416 Only)*

Revision(s) Affected: 1.1 and earlier

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 four words are affected. (Internal reference number DSPvd03632).

Workaround: To completely avoid the data corruption, do not perform burst lengths of more than four words or adhere to all of the following guidelines:

1. Limit the PCI burst lengths to 9 words or less
2. Operate the DSP at an 18.0 or 14.8 CPU-to-PCI clock frequency ratio only (e.g., 600-MHz or 490-MHz CPU for 33-MHz PCI)
3. Move the PCI to EDMA priority 0 and move all other EDMA requestors to other queues
4. Perform slave writes to L2 SRAM destinations only
5. Set EDMA_WEIGHT register in the L2 controller to x0000 0002

Following these guidelines ensures that the PCI peripheral and the EDMA servicing logic never align their signals in a way that causes the problematic FIFO state. In cases where the burst length size cannot be controlled, following all guidelines except #1 will minimize the chance of the data corruption.

Advisory 1.1.4 *PCI: Slave Reads Without Any Byte Enables Issue Target Abort (C6415/C6416 Only)*

Revision(s) Affected: 1.1 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 1.1.5 *PCI: Master Transaction Following an Abort Can Erroneously Set MASTEROK (C6415/C6416 Only)*

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) 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 (C6415/C6416 Only)*

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 (C6415/C6416 Only)*

Revision(s) Affected: 1.1 and earlier

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 and earlier

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 and earlier

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.

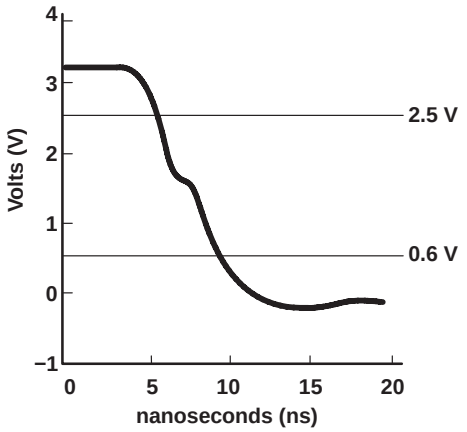


Figure 2. Bad TCK Transition

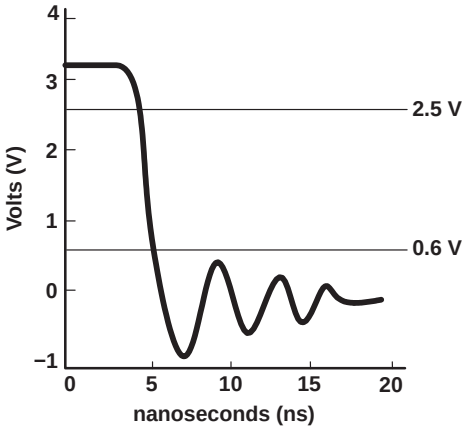


Figure 3. Good TCK Transition

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 and earlier

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.

XDS560 is a trademark of Texas Instruments.

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 and earlier

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.

PLL: PLL May Fail to Oscillate on Startup (Continued)

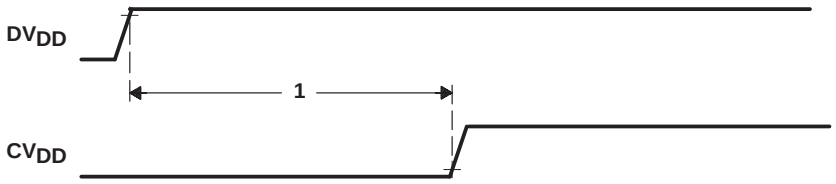


Figure 4. Power-On Sequence Timing

Advisory 1.1.13	*PCI: PCI Power Management Sticky Bits Always "1" (C6415/C6416 Only)*

Revision(s) Affected: 1.1 and earlier

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 and earlier

Details: When using either EMIFA or EMIFB in a system where the 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 and earlier

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.

4 Silicon Revision 1.03 Known Design Exceptions to Functional Specifications and Usage Notes

4.1 Usage Notes for Silicon Revision 1.03

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.

TIMER: Timer Does not Stop During Emulation Halt

On silicon revision 1.03 and earlier, when the CPU is halted by emulation, the timers should stop counting until the CPU is running again. Instead, the timers do not stop but continue to count during emulation halts.

EMIF: Design Specification Change for ARDY Sampling Moving Forward for Silicon Revision 1.1

On silicon revision 1.03 and earlier, ARDY is sampled for the first time **one** ECLKOUT cycle before the last cycle of the programmed strobe period. The new design specification moving forward for silicon revision 1.1 has been changed such that ARDY is sampled for the first time **two** ECLKOUT cycles before the last cycle of the programmed strobe period. Documentation in the EMIF chapter of the *TMS320C6000 DSP Peripherals Overview Reference Guide* (literature number SPRU190) supports the new design specification for silicon revision 1.1.

Advisory 1.03.1	*EMIF: BECLKIN Must Be Provided During Reset*

Revision(s) Affected: 1.03 and earlier

Details: If BECLKIN is not clocking during reset, the BEA signals that are used to latch the device boot mode may drive during reset, overriding the system-driven boot mode. If the EMIF-driven value selects ECLKIN as the clock, then the EMIF will hang. (Internal reference number DSPvd02684)

Workaround: Provide an active BECLKIN during reset, even if internally generated EMIF clocks are selected.

Advisory 1.03.2	*EMIF: BUSREQ Signal is Deasserted for 1 Cycle Under Certain Scenario*

Revision(s) Affected: 1.03 and earlier

Details: If an external device requests the EMIF bus (either EMIFA or EMIFB) via the $\overline{\text{HOLD}}$ signal, the EMIF will drive BUSREQ active if a new command is pending for the EMIF (either via CPU access or DMA access). When the external device releases the bus by deasserting the $\overline{\text{HOLD}}$ request, the EMIF may release its BUSREQ signal for 1 cycle even though the command has not completed. The BUSREQ will immediately be reasserted and the command will complete on the pins. (Internal reference number DSPvd02640)

EMIF: BUSREQ Signal is Deasserted for 1 Cycle Under Certain Scenario (Continued)

Workaround: When the external requestor releases the DSP from the Hold state, the external requestor should not assume that the EMIF command is complete if the BUSREQ signal is deasserted immediately. The controller should wait at least 4 EMIF clock cycles to allow pending EMIF transactions to complete before reasserting the $\overline{\text{HOLD}}$ input to the DSP. Command completion will be accurately depicted by the BUSREQ signal after this 4-cycle period.

Advisory 1.03.3 — *EMIF: PDT Feature is not Functional*

Revision(s) Affected: 1.03 and earlier

Details: The PDT feature is not functional and should not be used on silicon revision 1.0. (Internal reference number DSPvd02663)

Workaround: None

Advisory 1.03.4 — *L2: All Odd MAR Reads Return Incorrect Data*

Revision(s) Affected: 1.03 and earlier

Details: All reads from an odd-numbered MAR register always return ‘0’. For example, a read from MAR128 will return the register’s contents but a read from MAR129 will always return ‘0’, regardless of the register’s contents. This does not affect the cache control functionality of the odd MARs, just the ability to determine the current MAR setting. (Internal reference number DSPvd02601)

Workaround: Do not rely on reads from odd-numbered MARs to determine the cacheability of their respective memory regions.

Advisory 1.03.5 — *L1D: Overlapping/Adjacent Accesses Cause Bank Conflict Stalls*

Revision(s) Affected: 1.03 and earlier

Details: Overlapping load or adjacent load/store accesses to L1D cause an unnecessary bank conflict CPU stall. Overlapping accesses are two accesses that request some or all of the same data in a word. Adjacent accesses are those that request separate parts of the same word. When two loads overlap, the L1D causes a memory bank conflict stall to happen, though the loads should happen simultaneously, without a stall. Similarly, adjacent accesses for any combination of loads and stores should not cause a memory bank conflict but do. (Internal reference number DSPvd02666)

Workaround: Avoid overlapping and/or adjacent accesses where a CPU stall is unacceptable.

Advisory 1.03.6	*DMA: DMA Can Be Locked Out of L2*

Revision(s) Affected: 1.03 and earlier

Details: Under certain conditions, a DMA transfer can be denied the use of L2 for a long period of time. This can cause the DMA to miss data transfers that have real-time service requirements (such as certain serial port transfers). There are two scenarios where this problem can happen, both of which occur when a flood of many CPU stores miss L1D. The amount of time that a DMA is locked out of L2 is controlled by the code running on the DSP, and ultimately by the amount of time that L1D write misses are happening in succession.

The first scenario is applicable anytime the CPU issues stores that miss L1D at a rate of at least 1 per cycle. Under this scenario, the DMA will be locked out for the duration of the write flood, regardless of how the rest of the system is configured.

The second scenario is applicable when the CPU issues stores that miss L1D at a rate of at least 1 every other cycle and some L2 is allocated as cache. Under this scenario. the lockout condition occurs when there is an L1P miss that also misses L2, immediately followed by the section of code that produces the flood of stores.

It should be noted that under any scenario, the lockout does not occur for L1D read misses, since the CPU is stalled waiting on data to be returned from L2 before subsequent L1D misses can be generated.

Workaround: If the problem is occurring under the first scenario, implement any one of the following:

- Pre-read all data locations to be written. This will allocate them in L1D, preventing the L1D misses from occurring
- Make sure that any algorithm or code segment that naturally produces a long string of writes inserts gaps in the write sequence such that writes occur at a frequency slower than 1 write miss per cycle. The term “long” is based on the system requirements.

If the problem is occurring under the second scenario, implement any one of the following:

- Pre-read all data locations to be written. This will allocate them in L1D, preventing the L1D misses from occurring
- Make sure that any algorithm or code segment that naturally produces a long string of writes inserts gaps in the write sequence such that writes occur at a frequency slower than 1 write miss every other cycle. The term “long” is based on the system requirements.
- Run the algorithm or code segment from internal SRAM.
- Ensure the entire algorithm or code segment is in cache.

Advisory 1.03.7 *UTOPIA: Utopia Does Not Drive RxCLAV/TxCLAV Low Before Hi-Z (C6415/C6416 Only)*

Revision(s) Affected: 1.03 and earlier

Details: Both RCLAV and XCLAV should be driven inactive low before transitioning to Hi-Z state, but are not. In MPHY mode where there are multiple devices tied to the same bus, it is possible for the master polling the PHYs to perceive that a nonexistent PHY exists because (R/X) CLAV were not first driven low. Since not all master ATM devices have a PHY disable register (PDR), it may not be possible to avoid this situation. (Internal reference number DSPvd02660)

Workaround: Implement one of the following:

- Ensure unused ATM Controller Slaves or PHYs are programmed as NULL PHYs (address 31) so that they do not respond to the master. This can easily be done if they are all TMS320C64x DSP devices on the bus.
- If some of the slave/PHYs are not C64x DSP devices, and the above workaround cannot be implemented in them, an appropriate pulldown resistor should be added to the CLAV bus to ensure the bus is at a logic-low for the next Utopia clock cycle. The pulldown resistor should be able to pull the bus low in one Utopia clock period minus the sum of the DSP's CLK-to-High-Z time and the master's input setup time. The timing constraint can be met by carefully choosing an appropriate resistor size and slowing down the Utopia clock such that the resulting time window is large enough and the resistor is small enough to allow the bus to sufficiently discharge. The following equation governs the resistor and clock periods:

  $$\ln (V_{OH}/V_{IL}) * R * C_{bus} = t_{CLK} - t_{dMax} - t_{isu}$$

  Given the rest of the system parameters, the appropriate value of R can be found. For example:

  System 1: 8 devices on the bus
  C_{bus} = 80 pf, t_{dMax} = 12 ns, t_{isu} = 3 ns, V_{OH} = 3.3 V, V_{IL} = 0.8 V,
  f_{CLK} = 10 MHz, t_{CLK} = 100 ns

  $\ln (V_{OH}/V_{IL}) = 1.5$
  $t_{CLK} - t_{dMax} - t_{isu} = 85$ ns
  $1.5 * R * 80$ pf = 85 ns
  $R = 708\ \Omega$

UTOPIA: Utopia Does Not Drive RxCLAV/TxCLAV Low Before Hi-Z (C6415/C6416 Only) (Continued)

System 2: 2 devices on the bus
C_{bus} = 20 pf, t_{dMax} = 12 ns, t_{isu} = 3 ns, V_{OH} = 3.3 V, V_{IL} = 0.8 V,
f_{CLK} = 20 MHz, t_{CLK} = 50 ns

$\ln (V_{OH}/V_{IL}) = 1.5$
$t_{CLK} - t_{dMax} - t_{isu} = 35$ ns
$1.5 * R * 20$ pf = 35 ns
$R = 1.1$ kΩ

Advisory 1.03.10 — *L1D: Write Hits Stall When Write Miss Buffer is not Empty*

Revision(s) Affected: 1.03 and earlier

Details: If a CPU write hits in L1D when previous writes that missed in L1D are still in the L1D write miss buffer, the CPU will stall until the write miss buffer has been emptied.

Workaround: None

Advisory 1.03.11 — *HPI: Reading HPIA or HPIC After Reading HPID With Postincrement Fails*

Revision(s) Affected: 1.03 and earlier

Details: In HPI16 mode, after reading HPID with postincrement enabled, a read to either HPIA or HPIC will fail. The second halfword of HPIA or HPIC is not correctly asserted onto the HPI bus.

Workaround: Implement any one of the following:

- Perform fixed-mode access to HPID prior to an HPIA or HPIC read.
- Perform HPIC write prior to HPIA read.

Advisory 1.03.12 — *EMIF: Asynchronous Mode Does Not Work With Hold Values Other Than 1*

Revision(s) Affected: 1.03 and earlier

Details: In asynchronous mode, when RDHOLD and/or WRHOLD are set to any value except 1, the EMIF bus does not access memory correctly.

Workaround: Program both RDHOLD and WRHOLD to 1 and alter system parameters to work with hold time of 1 cycle. This can be done by slowing the EMIF clock rate by decreasing the external input frequency.

Advisory 1.03.13 *TCP: TCP is Not Functional for Certain Frame Sizes and Code Rates (C6416 Only)*

Revision(s) Affected: 1.03 and earlier

Details: The TCP does not decode properly for frame sizes (F) of 5 + (8 * n) when the code rate (r) is 1/3. The last set of DMA data is not copied to the systematic and parity RAMs. The TCP therefore does not have all the required data and may return incorrect results.

Workaround: Change the program to avoid using frame sizes of 5 + (8 * n). This can be done by increasing the frame size by one, inserting “dummy” symbols corresponding to encoded bit 0 at the beginning of the frame, and modifying the interleaver table and the TCP-generated hard decisions accordingly.

The procedure for a frame size F = 5 + (8 * n) is as follows:

1. Add three symbols to beginning of frame. All three symbols should have the value 0x80, corresponding to encoder input/output 0.
2. Generate an interleaver table of size F as per system requirements.
3. Modify the interleaver table such that the entry at index 0 is 0, and all other entries are moved up by one index. The pseudo-C code is as follows:

```
for (i=F; i>0; i--) {
  int_table[i] = int_table[i-1] + 1;
}
int_table[0] = 0;
```

4. Configure the TCP and EDMA for decoding of frame size F1 = F + 1.
5. Execute TCP.
6. Discard first bit [least significant bit (LSB) of the first hard decision word returned by TCP]. The pseudo-C code is as follows:

```
// shift right all complete 32-bit words
for (i=0; i<F1/32; i++) {
  hard_decisions[i] = hard_decisions[i] >> 1;
  hard_decisions[i] = hard_decisions[i] & 0x7FFFFFFF;
  temp_bit = hard_decisions[i+1] & 0x1;
  temp_bit = temp_bit << 31;
  hard_decisions[i] = hard_decisions[i] | temp_bit;
};
// shift right the last (incomplete) 32-bit word.
hard_decisions[i] = hard_decisions[i] >> 1;
```

Advisory 1.03.14 *EMIF: Asynchronous Write Followed by Read Can Occur With No $\overline{CE}$ Deassertion*

Revision(s) Affected: 1.03 and earlier

Details: An asynchronous write followed immediately by a read can occur with no $\overline{CE}$ deassertion. When this occurs, $\overline{AOE}$ is driven active on the same cycle that the data lines are put into the high-impedance state. Due to finite turn-on time for external devices, it is unlikely that this will have any device impact.

For interfaces to standard asynchronous memory, there are no logical implications, since most asynchronous memories do not have a requirement for $\overline{CE}$ to be deasserted between a write and read access. Custom logic should detect new commands based on transitions on $\overline{AOE}$, $\overline{AWE}$, and $\overline{ARE}$ as long as $\overline{CE}$ is asserted. Logic should not depend on $\overline{CE}$ signal transitions.

Workaround: None

Advisory 1.03.15 *EMIF: $\overline{CE}$ Not Asserted Correctly Under Certain Scenario in Synchronous Mode*

Revision(s) Affected: 1.03 and earlier

Details: In programmable Synchronous mode, when the $\overline{CE}$ extension bit (CEEXT in CExSEC) is turned on and read latency (SYNCRL in CExSEC) is 2 or 3, $\overline{CE}$ is not asserted at the start of the memory access. Instead, $\overline{CE}$ assertion is delayed until $\overline{SOE}$ gets asserted. As a result, memories will miss the command that was issued before $\overline{SOE}$ and $\overline{CE}$ were active. This does not happen for interfaces that use a read latency (SYNCRL in CExSEC) of 0 or 1.

Workaround: For interfaces that use a read latency greater than 1, the CEEXT feature should not be used.

Advisory 1.03.16 *EDMA: Transfer Completion Code Not Set for Large Transfers*

Revision(s) Affected: 1.03 and earlier

Details: For transfers where the element count is large (ELECNT ≥ 0x8000), all data is transferred as programmed, but the transfer does not register as complete when it is finished. The bit specified by TCC and TCINT does not get set in CIPR, and the linked EDMA parameter set is not loaded.

Workaround: Break up large transfers into smaller transfers of fewer than 0x8000 elements.

Advisory 1.03.17 *TCP: Frame Sizes Larger Than 20730 Cause Error (C6416 Only)*

Revision(s) Affected: 1.03 and earlier

Details: When running the TCP in shared processing mode, if the frame size is greater than 20730, the TCP will generate an error interrupt. The TCP is stopped and the frame is not decoded.

Workaround: None

Advisory 1.03.18 *VCP: Output Pointer is Incorrect (C6416 Only)*

Revision(s) Affected: 1.03 and earlier

Details: When the VCP coprocessor decodes frame of size (N * 64) ± (K − 1) the output FIFO pointer generated is incorrect. This causes the reading of the following decoded frames to be incorrect. Therefore, you cannot chain several frames without manually performing a start between them. In addition, this results in unpredictable behavior while using convergent mode.

Workaround: Perform a START via the VCPEXE register between frames. Do *not* use convergent mode.

Advisory 1.03.19 *TCP: TCP Does Not Generate EDMA Event With SUM = 01 (C6416 Only)*

Revision(s) Affected: 1.03 and earlier

Details: When the TCP coprocessor is set up to transfer output parameters via EDMA (OUTF bit in TCPIC0 is 1) and SUM = 01 (the default setting) of the corresponding EDMA parameters, no XEVT is generated. Thus, the EDMA transfer of the output parameters does not occur.

Workaround: Set SUM = 00 for the EDMA options of the output parameters transfer.

Advisory 1.03.20	*UTOPIA Does Not Regenerate EDMA Events as Specified (C6415/C6416 Only)*

Revision(s) Affected: 1.03 and earlier

Details: The UTOPIA peripheral does not regenerate EDMA events as indicated in the Slave-Transmit Queue section and the Slave-Receive Queue section of the *TMS320C6000 DSP Peripherals Overview Reference Guide* (literature number SPRU190). This exception does not affect functional correctness, but does impact UTOPIA data transfer efficiency. On the transmit side, UXEVT should be reasserted when there is space for at least one cell packet, even if a previous cell packet is still being written to the UTOPIA peripheral. However, UXEVT is not generated until the previous cell is completely written to the UTOPIA peripheral.

Similarly, on the receive side, UREVT should be reasserted when the next complete cell is available, even if a previous cell is still being read. UREVT is actually reasserted only when the previous cell has been completely read.

Workaround: None

Advisory 1.03.21	*PCI: Idle Cycles are Required Between Slave Reads (C6415/C6416 Only)*

Revision(s) Affected: 1.03 and earlier

Details: For correct PCI behavior, at least two Idle cycles are required between consecutive slave reads. A slave read is one in which the host is the master and the DSP is the slave. If the Idle cycle requirement is not met, the DSP's PCI port will disconnect-retry the transaction forever.

Workaround: Ensure that there are at least two Idle PCI cycles between consecutive slave reads of the DSP.

Advisory 1.03.23	*EMU: Device May Not Operate Properly Without an Active TCK*

Revision(s) Affected: 1.03 and earlier

Details: The DSP may not operate properly without at least one TCK edge after $\overline{\text{RESET}}$ goes high. If the device does not initialize properly, it may operate unpredictably after reset. Note that operating with an emulator will correct this problem as the emulator provides TCK.

Workaround: For the reset and initialization logic to work properly, the logic must see at least one rising edge on TCK after $\overline{\text{RESET}}$ goes high and before the first L2 memory access. This TCK edge should occur as soon as possible after the rising edge of $\overline{\text{RESET}}$ to ensure it is before any L2 accesses, regardless of the device's boot mode. There are several ways to accomplish this:

- Always operate with an emulator
- Route a continuously running clock to TCK
- Route a logic signal to TCK that generates the rising edge at the correct time

Advisory 1.03.24	*CACHE: L2 Does Not Clean or Flush the Cache on Every Invocation*

Revision(s) Affected: 1.03 and earlier

Details: The L2CLEAN and L2FLUSH operations do not properly clean/flush the cache on every invocation. Rather, every second invocation does not flush/clean the contents of the L2 cache. No data is lost, but the cache will not be in the expected state when the clean/flush completes. This advisory does not apply to cleans or flushes initiated with the xxxBAR and xxxWC registers, only those cleans or flushes initiated with L2FLUSH and L2CLEAN.

Workaround: Write a "1" to the L2CLEAN or L2FLUSH register twice in succession.

Advisory 1.03.26	*JTAG: JTAG Does Not Function in Heterogeneous Environments*

Revision(s) Affected: 1.03 and earlier

Details: The JTAG port does not work properly if non-C64x devices are in the scan chain with the C64x device. This advisory applies to emulation mode only and does ***not*** impact boundary-scan mode.

Workaround: Place all C64x devices in a separate scan chain.

Advisory 1.03.27	*EMU: Advanced Emulation Features are Not Functional*

Revision(s) Affected: 1.03 and earlier

Details: JTAG real-time data exchange (RTDX™) software, high-speed RTDX, BIOS real-time analysis (RTA) via RTDX, hardware breakpoints, multiprocessor global breakpoints, multiprocessor homogeneous debug, and advanced event triggering (AET) do not work.

Workaround: Use normal software breakpoint emulation to debug and evaluate systems.

Advisory 1.03.28	*HPI: Host HPIA Writes can Block CPU HPIC Writes*

Revision(s) Affected: 1.03 and earlier

Details: Host HPIA writes that arrive at the same time as CPU HPIC writes prevent the HPIC writes from taking effect. It will appear that the HPIC write never happened, causing problems such as the inability to clear the DSPINT bit to allow future interrupts, or being unable to set bit fields in the HPIC register.

Workaround: From the CPU, repeatedly write HPIC until a read confirms that the register has updated to the correct value.

Advisory 1.03.29	*PCI: PCI Burst Lengths Limited to 16 Words (C6415/C6416 Only)*

Revision(s) Affected: 1.03 and earlier

Details: Due to the buffering between the PCI port and the internal EDMA hardware servicing the port, the PCI port disconnects transactions that attempt to burst more than 16 words.

Workaround: These transaction disconnects do not cause any functional problems, they are merely a performance limitation that will be resolved in a future silicon revision.

Advisory 1.03.30 *TCP: SNR Stopping Criteria Nonfunctional for Small Frames (C6416 Only)*

Revision(s) Affected: 1.03 and earlier

Details: The TCP has a limiting feature for processing delay such that if the decoded data has reached a sufficient quality, the TCP will stop decoding and the decoded data will be ready for use by the CPU. For frame sizes of less than 64, the TCP may stop sooner than desired, if the signal-to-noise ratio (SNR) stopping criteria is enabled. When this happens, the quality of the decoded data may be compromised. (Internal reference number DSPvd02668).

Workaround: When using the TCP with frame sizes less than 64, disable the SNR stopping criteria feature.

Advisory 1.03.31 *VCP: Normalization Error in SM Max/Min (C6416 Only)*

Revision(s) Affected: 1.03 and earlier

Details: If noise in the data is significant, there is a possibility that the state metrics will not be normalized correctly. Discrepancies might exist in the state metrics maximum/minimum computation caused by a sign operation error. (Internal reference number DSPvd02917).

Workaround: None

Advisory 1.03.32 *VCP: Sliding Windows Boundaries (C6416 Only)*

Revision(s) Affected: 1.03 and earlier

Details: When the frame length exceeds the VCP's capacity to decode in one window, computations are performed in multiple windows attached together. Under this condition, if noise in the data is significant at the boundary between two windows, the data will potentially be decoded differently in the two windows. This results in invalid data over the boundary. (Internal reference number DSPvd02919).

Workaround: None

Advisory 1.03.33 *TCP/VCP: Using Concurrent/Different Priority Levels can Corrupt Data (C6416 Only)*

Revision(s) Affected: 1.03 and earlier

Details: The EDMA has a specific Hub Interface Unit (HIU), which interfaces the EDMA to the peripherals. Typically, a single EDMA port interfaces to a single peripheral via a single HIU. The TCP and VCP, however, are both interfaced to the EDMA behind a single HIU. In this respect, the EDMA views the TCP and VCP as a single peripheral (i.e., single HIU), whereas reality is that they are two distinct peripherals, which do not monitor commands for the other module.

TCP/VCP: Using Concurrent/Different Priority Levels can Corrupt Data (C6416 Only) (Continued)

The EDMA Coprocessor HIU is optimized for pipelining commands to a single peripheral. If transfers are enabled to the TCP and VCP on different priority levels, then it is possible for both transfers to be active simultaneously, interleaving their commands to the single HIU. When this happens, there is a chance that the command responses from the TCP and VCP will become confused. This can corrupt the data involved with either or both transfers. (Internal reference number DSPvd03608).

Workaround: If the system is using TCP and VCP in parallel, all EDMA transfers to or from the TCP and VCP must be set to the same priority level.

Advisory 1.03.34

PCI Speed: PCI Does not Operate Correctly Unless CPU is Running 8 Times Faster (C6415/C6416 Only)

Revision(s) Affected: 1.03 and earlier

Details: If the ratio of the CPU clock frequency to the PCI clock frequency is less than 8:1, the PCI port does not behave correctly. The port may return incorrect data, refuse all transactions, or lock-up completely. (Internal reference number DSPvd03436, DSPvd03603, DSPvd03535, DSPvd03536).

Workaround: Always clock the CPU at least 8 times faster than the PCI port. For example, in a system with a 33-MHz PCI bus, always run the CPU faster than 266 MHz.

Advisory 1.03.35

EMU: Interrupts Received While Halted may not Execute Correctly

Revision(s) Affected: 1.03 and earlier

Details: If the CPU is halted by emulation, and an interrupt arrives at the Interrupt Pending Register (IPR), then any subsequent write of a CPU control register through emulation will corrupt the CPU state. As a result, the CPU will branch to the reset vector instead of the correct interrupt service routine (ISR). (Internal reference number DSPvd02763).

Workaround: Either ensure no interrupts happen while halted or disable interrupts before software breakpoints.

Advisory 1.03.36 *EMU: Interrupts Received Near Software Breakpoints may Falsely Trigger the Breakpoint*

Revision(s) Affected: 1.03 and earlier

Details: If an interrupt arrives causing an instruction that is tagged with a software breakpoint (SWBP) to be annulled, the SWBP will halt the CPU anyway. The emulator will show the program counter (PC) at the instruction with the SWBP, though that instruction has not taken effect, and the CPU will next execute the appropriate interrupt service routine. (Internal reference number DSPvd02764).

Workaround: None

Advisory 1.03.37 *PCI: Non-Word PCI Accesses in Big-Endian Mode Corrupt Data (C6415/C6416 Only)*

Revision(s) Affected: 1.03 [Big-Endian mode *not* functional on earlier silicon revisions]

Details: When the PCI is configured in Big-Endian mode, non-word accesses to the PCI port corrupts data. The PCI port correctly executes 32-bit word accesses.

Workaround: When in Big-Endian mode, use *only* 32-bit word accesses (all four byte enables asserted) over PCI.

Advisory 1.03.38 *VCP: SYMX/R Bits Set to a Value of Less Than 2 Can Cause Error (C6416 Only)*

Revision(s) Affected: 1.03 and earlier

Details: The VCP may generate incorrect results or hang if the SYMX/R bits in the VCPIC5 register are configured with a value of less than 2 (0010).

Workaround: For correct operation, the SYMX/R bits should always be configured (set) to the maximum value of 15 (1111).

5 Silicon Revisions 1.0, 1.01, and 1.02 Known Design Exceptions to Functional Specifications

Advisory 1.0.7	*EMIF: CLKOUT6 is Not Functional*

Revision(s) Affected: 1.02 and earlier

Details: The internally generated CPU/6 EMIF clock may come out of reset as either a CPU/2 clock or a CPU/6 clock. In addition, duty cycle is not 50 percent. This problem affects both the EMIF internal clock and the CLKOUT6 pin, rendering it unusable. (Internal reference number DSPvd02679)

Workaround: Use CLKOUT4 if the CPU is running at or below 533 MHz, or use an external clock on ECLKIN.

Advisory 1.0.13	*PCI: PCI is Not Functional When DSP is in Big-Endian Mode (C6415/C6416 Only)*

Revision(s) Affected: 1.02 and earlier

Details: The PCI port does not operate when the DSP is in big-endian mode.

Workaround: Use the PCI port with the DSP in the little-endian mode only.

Advisory 1.0.18	*RESET: Asserting $\overline{\text{RESET}}$ Does Not Wake Up CPU From PD3 Mode*

Revision(s) Affected: 1.02 and earlier

Details: When in PD3 power-down mode, the CPU does not respond to $\overline{\text{RESET}}$ being asserted. A power cycle is needed to wake up the device, and there is a chance that multiple power cycles will be required to successfully wake up the device.

Workaround: To wake the CPU from PD3 power-down mode, power cycle the device until it properly resets.

Advisory 1.0.19	*RESET: Improper Reset May Damage the Device*

Revision(s) Affected: 1.02 and earlier

Details: The TMS320C64x device uses electronically blowable fuses for certain memory and device configuration purposes, which may be damaged if the device is not held in reset during and after the power-on ramp. In addition, since the reset signal must propagate through the I/O buffers to the core logic, the I/O supply must be powered on enough to allow the reset signal to propagate through before the core supply reaches a voltage where fuse damage can occur (1.0 V). However, if the I/O supply reaches a voltage where the I/O buffers are operational before the core is powered on enough to force all outputs to the high-impedance state, board-level bus contention may result.

Workaround: To protect the fuses and prevent bus contention, a reset supervisor must be used in conjunction with a controlled power-on ramp sequence. A reset supervisor such as the Texas Instruments' TPS3110 ensures that $\overline{\text{RESET}}$ is asserted low or put in a high-impedance mode so that it may be pulled down for the entire power-on ramp. The I/O rail must start to ramp before or at the same time as the core rail, and the core rail must be no more than 300 mV behind at any time until the core rail has reached its operating voltage. Ramping the power in this manner ensures that the device's input buffers allow the $\overline{\text{RESET}}$ signal to propagate to the core (which is necessary to prevent fuse damage) while providing a valid signal to the device's output buffers before they can drive (which is necessary to prevent bus contention). Power supplies such as the Power Trends™ PT6932 sequence the power ramp in this manner. See Figure 5 and Figure 6 for the core and I/O power-on ramps and a recommended reset supervisor circuit example.

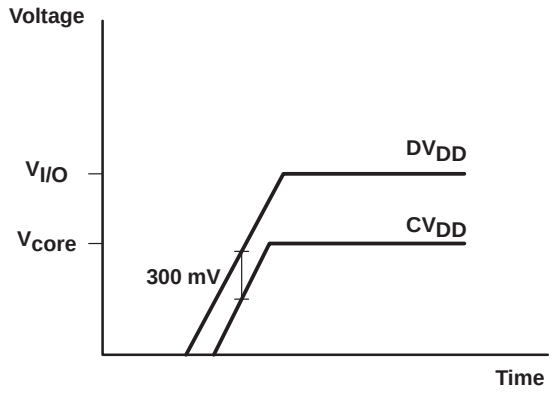
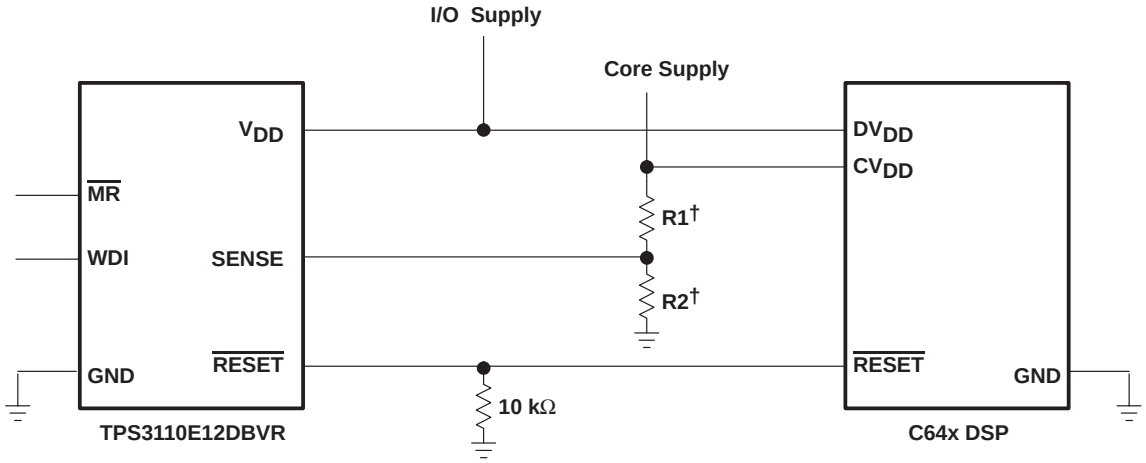


Figure 5. Core and I/O Power-On Ramps

Power Trends is a trademark of Texas Instruments.

RESET: Improper Reset May Damage the Device (Continued)



$$V_{(SENSE-TH)} = 0.551V \times \frac{R1 + R2}{R2}$$

† R1 and R2 should be chosen such that $V_{(SENSE-TH)}$ is set below the data sheet recommended operating conditions voltage for CV_{DD}.

Figure 6. Example Reset Supervisor Circuit

Advisory 1.0.26	*PD1 Does Not Gate Off CPU Clock*

Revision(s) Affected: 1.02 and earlier

Details: Power-down mode 1 (PD1) does not gate off the CPU clock. Instead, an IDLE instruction is executed on the CPU. As a result, PD1 does not reduce power consumption significantly below the IDLE level.

Workaround: None

IMPORTANT NOTICE

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

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

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

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

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

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

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

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

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

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

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
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
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
Low Power Wireless	www.ti.com/lpw	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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