TMS320C6418

Digital Signal Processor

Silicon Errata

Silicon Revision 1.1

SPRZ224A
August 2004
Revised November 2004

REVISION HISTORY

This silicon errata revision history highlights the technical changes made to the SPRZ224 revision to make it an SPRZ224A revision.

Scope: Applicable updates to the C64x device family, specifically relating to the C6418 devices, have been incorporated.

PAGE(S) NO.	ADDITIONS/CHANGES/DELETIONS
11	Silicon Revision 1.1 Known Design Exceptions to Functional Specifications: Added the following *new* silicon revision 1.1 advisories: Advisory 1.1.6, EMIF: PDT Transfers Fail When Acessing the Same SDRAM Page as Non-PDT Transfers

Contents

1 Introduction

This document describes the known exceptions to the functional specifications for the TMS320C6418 digital signal processor. [See the *TMS320C6418 Fixed-Point Digital Signal Processor* data manual (literature number SPRS241).] Throughout this document, TMS320C64x and C64x refer to the TMS320C6418 devices.

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:

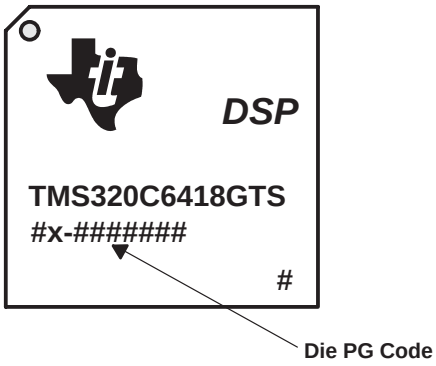
"Developmental product is intended for internal evaluation purposes."

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 GTS package is shown in Figure 1. Figure 1 shows some examples of the types of C6418 package symbolization.



NOTES: A. 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.
B. “#” denotes an alphanumeric character. “x” denotes an alpha character only.

Figure 1. Example, Die PG Codes for TMS320C6418 (GTS)

Silicon revision is identified by a code on the chip. The code is of the format #x-#######. If x is “A”, then the silicon is revision 1.1, etc. Table 1 lists the silicon revisions associated with each die PG code for the C6418 devices.

Table 1. Die PG Codes

Die PG Code (x)	Silicon Revision	Comments
A	1.1	Initial silicon revision − TMX320C6418GTS

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

2.1 Usage Notes for Silicon Revision 1.1

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

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

On C6418 silicon revision 1.1, 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 1.1 Known Design Exceptions to Functional Specifications

Advisory 1.1.1	*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™ Integrated Development Environment (IDE) TMS320C64x Silicon Revision 1.1 Chip Support Package (CSP)]. These workaround suggestions are discussed in release note #12 — “SDSsq27324: DSP/BIOS™ Real-Time Analysis (RTA) Update Halt and Real-Time Data Exchange (RTDX) Data Corruption”.

Advisory 1.1.2 — *Interrupts: NMI not Functional if HPI Module is Disabled*

Revision(s) Affected: 1.1

Details: The Non-Maskable Interrupt (NMI) will *not* propagate to the interrupt module when the HPI peripheral module is disabled on the device. At reset, the TOUT0/$\overline{\text{HPI\_EN}}$ pin determines whether the HPI peripheral module is enabled/disabled.

- If the TOUT0/$\overline{\text{HPI\_EN}}$ = 1 at reset, the internal clock to the HPI module is disabled (NMI does *not* work and will not propagate to the interrupt module).
- If the TOUT0/$\overline{\text{HPI\_EN}}$ = 0 at reset, the internal clock to the HPI module is enabled (NMI works as documented).

The $\overline{\text{HPI\_EN}}$ bit (DEVSTAT.8) [address 0x01B3 F004] can be used to read the status (enabled/disabled) of the HPI module. The $\overline{\text{HPI\_EN}}$ bit in the DEVSTAT register is automatically set to 0 when the HPI module is enabled, and set to 1 when the HPI module is disabled.

Workaround: Both a software and hardware workaround exist:

Software Workaround: A software workaround exists to capture the NMI when the HPI module is disabled ($\overline{\text{HPI\_EN}}$ [DEVSTAT.8] = 1).

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

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

Hardware Workaround: For the hardware workaround, ***ensure*** the HPI peripheral is enabled (TOUT0/$\overline{\text{HPI\_EN}}$ = 0 [default]). For more detailed information on configuration of the device, see the *Device Configuration* section of the device-specific data sheet.

or

If the system in which the DSP is being used requires that the HPI is disabled, then the software workaround ***must*** be used.

Advisory 1.1.3 — *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 1.1.4	*L2 Cache: Accesses to Mapped L2 RAM Update L2 LRU Information*

Revision(s) Affected: 1.1

Details:

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

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

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

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

Workaround:

Perform one of the following three workarounds:

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

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

Advisory 1.1.5	*L2 Cache: L2 Controller Incorrectly Updates LRU for Accesses in L2 Cache*

Revision(s) Affected: 1.1

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

Revision(s) Affected: 1.1

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.

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 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.

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.

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