TMS320VC5410

Digital Signal Processor

Silicon Errata

SPRZ177A
November 2000
Revised October 2001

Contents

1 Introduction . . . 3
1.1 Quality and Reliability Conditions . . . 3
TMX Definition . . . 3
TMP Definition . . . 3
TMS Definition . . . 3
1.2 Revision Identification . . . 4
2 Known Design Marginality/Exceptions to Functional Specifications . . . 5
$\overline{\text{HOLD}}$ in Emulation Mode . . . 5
Write Pending (4) . . . 6
DMA Interrupt Generation in ABU Mode With Address Indexing . . . 6
HPI: Force SAM Mode . . . 6
Far Branches/Calls/Interrupts from Active Repeat Blocks (BRAF) . . . 7
WRITA/MVDP . . . 7
NMI Not Recognized . . . 8
Round (RND) Instruction Clears Pending Interrupts . . . 9
DMGFR Update of DMSFCn During DMA Autoinitialization . . . 9
DMA: Sync Event Miss Following External Transfer . . . 10
HPI Write . . . 10
Bootloader: 8-Bit Parallel Mode Destination Address and Block Size Calculation . . . 11
Bootloader: 8-Bit I/O Mode Data Concatenation . . . 11
Bootloader: 8-Bit Parallel Mode . . . 11
Write Pending (1) . . . 12
Nested FRET . . . 13
Data Bus Holders . . . 13
BCLKS Internal Pullup Missing . . . 13
HPI/DMA Arbitration . . . 13
HPI: HRDY Stuck Low During Reset . . . 14
Write Pending (3) . . . 14
DMPREC . . . 15
CPU Read/DMA Write . . . 16
3 Documentation Support . . . 17

1 Introduction

This document describes the silicon updates to the functional specifications for the TMS320VC5410 silicon. The advisories are applicable to:

- TMS320VC5410 (144-pin LQFP, PGE suffix)
- TMS320VC5410 (176-pin MicroStar BGA™, GGW suffix)

1.1 Quality and Reliability Conditions

TMX Definition

Texas Instruments (TI) does not warranty either (1) electrical performance to specification, or (2) product reliability for products classified as “TMX.” By definition, the product has not completed data sheet verification or reliability performance qualification according to TI Quality Systems Specifications.

The mere fact that a “TMX” device was tested over a particular temperature and voltage ranges should not, in any way, be construed as a warranty of performance.

TMP Definition

TI does not warranty product reliability for products classified as “TMP.” By definition, the product has not completed reliability performance qualification according to TI Quality Systems Specifications; however, products are tested to a published electrical and mechanical specification.

TMS Definition

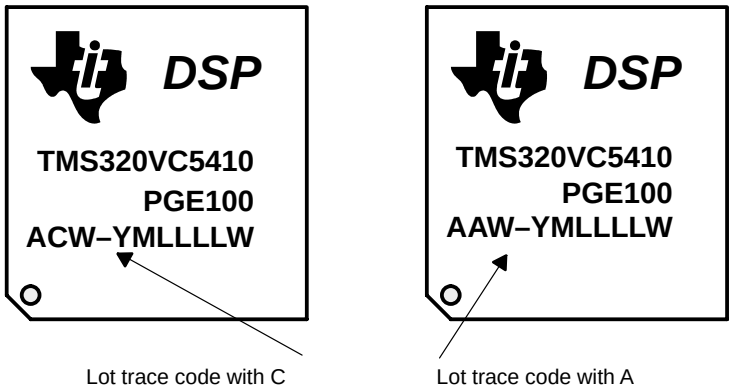
Fully-qualified production device.

MicroStar BGA is a trademark of Texas Instruments.

1.2 Revision Identification

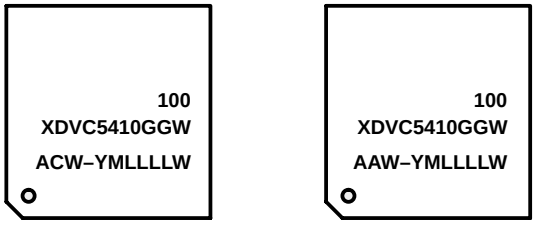
The device revision can be determined by the lot trace code marked on the top of the package. The locations for the lot trace codes for the PGE and the GGW packages are shown in Figure 1 and Figure 2, respectively. The location of other markings may vary per device.

Figure 1. Example, Typical Lot Trace Code for TMS320VC5410 (PGE)



Lot Trace Code	Silicon Revision	Comments
Blank (no second letter in prefix)	Initial Silicon	This silicon revision was terminated in the factory and no initial silicon devices were shipped.
A (second letter in prefix is A)	Silicon Revision A	
B (second letter in prefix is B)	Silicon Revision B	This silicon revision was terminated in the factory and no Revision B devices were shipped.
C (second letter in prefix is C)	Silicon Revision C	
D (second letter in prefix is D)	Silicon Revision D	

Figure 2. Example, Typical Lot Trace Code for TMS320VC5410 (GGW)



NOTE: Qualified devices in the PGE package are marked with the letters “TMS” at the beginning of the device name, while nonqualified devices in the PGE package are marked with the letters “TMX” or “TMP” at the beginning of the device name. Similarly, qualified devices in the GGW package are marked with the letters “DV” at the beginning of the device name, and nonqualified devices in the GGW package are marked with the letters “XDV” or “PDV” at the beginning of the device name.

2 Known Design Marginality/Exceptions to Functional Specifications

Table 1. Summary of Advisories

Description	Revision Affected	Page
$\overline{\text{HOLD}}$ in Emulation Mode	Initial Silicon, Revisions A, C, and D Silicon	5
Write Pending (4)	Initial Silicon, Revisions A, C, and D Silicon	6
DMA Interrupt Generation in ABU Mode With Address Indexing	Initial Silicon, Revisions A, C, and D Silicon	6
HPI: Force SAM Mode	Initial Silicon, Revisions A, C, and D Silicon	6
Far Branches/Calls/Interrupts from Active Repeat Blocks (BRAF)	Initial Silicon, Revisions A, C, and D Silicon	7
WRITA/MVDP	Initial Silicon, Revisions A, C, and D Silicon	7
NMI Not Recognized	Initial Silicon, Revisions A, C, and D Silicon	8
Round (RND) Instruction Clears Pending Interrupts	Initial Silicon, Revisions A, C, and D Silicon	9
DMGFR Update of DMSFCn During DMA Autoinitialization	Initial Silicon, Revisions A, C, and D Silicon	9
DMA: Sync Event Miss Following External Transfer	Initial Silicon, Revisions A, C, and D Silicon	10
HPI Write	Initial Silicon, Revisions A and C Silicon	10
Bootloader: 8-Bit Parallel Mode Destination Address and Block Size Calculation	Initial Silicon, Revisions A and C Silicon	11
Bootloader: 8-Bit I/O Mode Data Concatenation	Initial Silicon, Revisions A and C Silicon	11
Bootloader: 8-Bit Parallel Mode	Initial Silicon and Revision A Silicon	11
Write Pending (1)	Initial Silicon and Revision A Silicon	12
Nested FRET	Initial Silicon and Revision A Silicon	13
Data Bus Holders	Initial Silicon and Revision A Silicon	13
BCLKS Internal Pullup Missing	Initial Silicon and Revision A Silicon	13
HPI/DMA Arbitration	Initial Silicon and Revision A Silicon	13
HPI: HRDY Stuck Low During Reset	Initial Silicon and Revision A Silicon	14
Write Pending (3)	Initial Silicon and Revision A Silicon	14
DMPREC	Initial Silicon, Revisions A, C, and D Silicon	15
CPU Read/DMA Write	Initial Silicon, Revisions A, C, and D Silicon	16

Advisory *$\overline{\text{HOLD}}$ in Emulation Mode*

Revision(s) Affected: Initial Silicon, Revisions A, C, and D Silicon

Details: $\overline{\text{HOLDA}}$ is not generated when $\overline{\text{HOLD}}$ is asserted while device is stopped in emulation mode.

Workaround: None.

Advisory *Write Pending (4)*

Revision(s) Affected: Initial Silicon, Revisions A, C, and D Silicon

Details: There is a potential for CPU writes within SARAM to be corrupted by DMA accesses to the same block. This problem can affect any type of CPU writes, including MVDD, MVDP, MVPD, MVDK, STM, etc., to a SARAM block in which active DMA read and write accesses occur. When this problem occurs, the CPU write is corrupted and an incorrect value is written. This problem is similar to the *Write Pending (3)* problem which was corrected in a previous revision of silicon; however, the *Write Pending (4)* problem is caused when both DMA writes and DMA reads occur in the SARAM block as the CPU writes.

Workaround: This problem only occurs when both DMA reads and DMA writes occur within the same SARAM block as the CPU write. To prevent the problem, the DMA transfers should be modified such that only one type of DMA access occurs within the SARAM block that CPU writes occur, either the source (DMA read) or destination (DMA write) should be moved to a separate memory block.

Advisory *DMA Interrupt Generation in ABU Mode With Address Indexing*

Revision(s) Affected: Initial Silicon, Revisions A, C, and D Silicon

Details: When using indexed addressing with the DMA in ABU mode, some of the indexing modes do not work correctly.

Index value ≤ –1:	HALF interrupt does not work correctly. FULL interrupt does not work correctly.
Index value = +1:	HALF interrupt works correctly. FULL interrupt works correctly.
Index value ≥ +2:	HALF interrupt does not work correctly. FULL interrupt works correctly.

Workaround: In ABU mode, use only +1 as an index for indexed addressing.

Advisory *HPI: Force SAM Mode*

Revision(s) Affected: Initial Silicon, Revisions A, C, and D Silicon

Details: The HPIC reserved bit position 1 is uninitialized.

Workaround: This bit should be written with a logic 1.

Advisory *Far Branches/Calls/Interrupts from Active Repeat Blocks (BRAF)*

Revision(s) Affected: Initial Silicon, Revisions A, C, and D Silicon

Details: When a block repeat is interrupted by a far call, far branch, or interrupt to another page; and a program memory address in the called routine happens to have the same lower 16 bits as the block-repeat end address (REA), a branch to the 16-bit block-repeat start address (RSA) is executed on the current page until the block-repeat counter decrements to 0. The XPC is ignored during these occurrences.

Workaround: Use one of the following workarounds:

1. If the called routine must be on a different page and has a program memory address that has the same lower 16 bits as the REA, save ST1 and clear the BRAF in the vector table before entering the called routine with the following two instructions:

```
PSHM ST1
RSBX BRAF
```

   Then, restore ST1 before returning from the called routine. In the case of an interrupt service routine, these two instructions can be included in the delay slots following a delayed-branch instruction (BD) at the interrupt vector location. Then, the ST1 is restored before returning from the routine. With this method, BRAF is always inactive while in the called routine. If BRAF was not active at the time of the call, the RSBX BRAF has no effect.
2. Put the called routine on the same page as the interruptible block-repeat code. This can be achieved automatically by placing the interrupt vector table and the interrupt service routines or other called routines on the overlay pages. If this approach is used, far branches/calls are not necessary and the situation is completely avoided.
3. Avoid putting the called routine on other pages where a program memory address has the same lower 16 bits as the REA.
4. Use the BANZ instruction as a substitute for the block repeat.

Advisory *WRITA/MVDP*

Revision(s) Affected: Initial Silicon, Revisions A, C, and D Silicon

Details: If a WRITA or MVDP instruction executing from a SARAM block performs a write to any SARAM block that is immediately followed by any read (including DMA or instruction fetch) of the same address that is written to, the read data may be corrupted.

Workaround: Avoid using WRITA/MVDP to write to an area in memory that will be executed as code immediately following the WRITA/MVDP.

Rearrange the code so that a read access does not immediately follow the WRITA/MVDP instruction with an address that is identical to the last address written to. Use a dummy write if necessary.

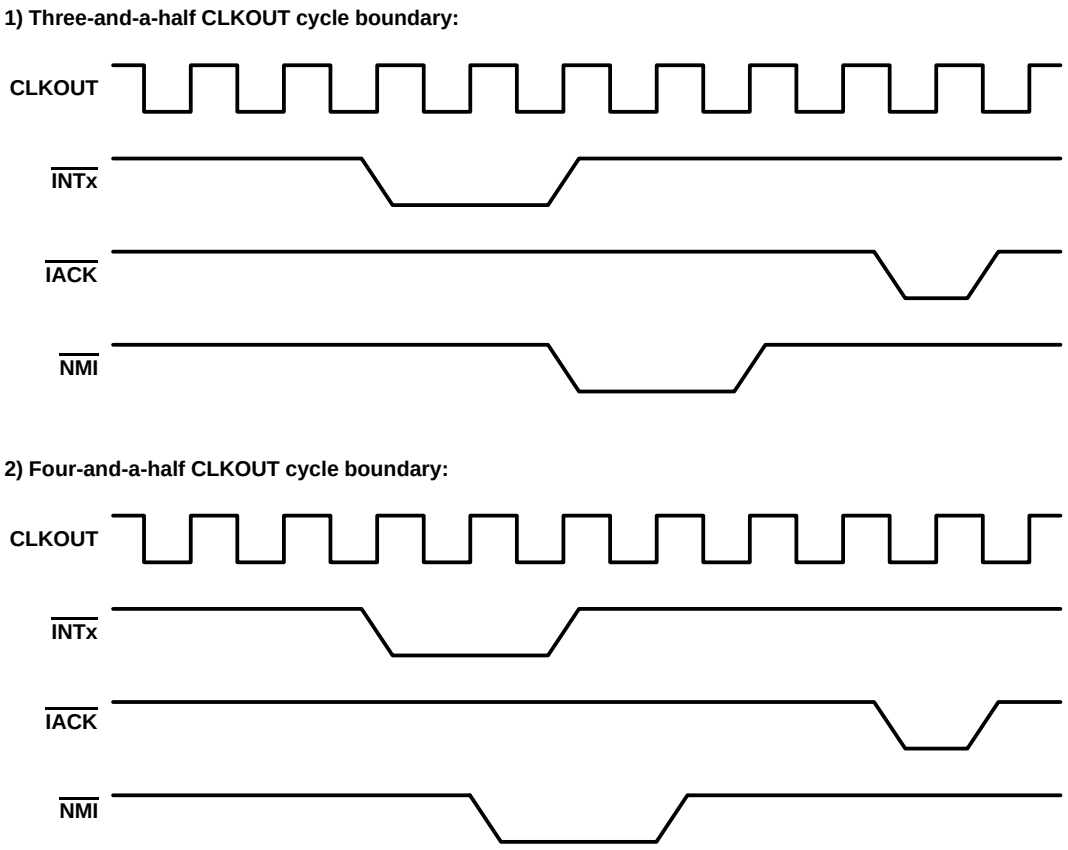
Avoid DMA reads in an area of program SARAM that is written to by WRITA/MVDP.

Advisory *NMI Not Recognized*

Revision(s) Affected: Initial Silicon, Revisions A, C, and D Silicon

Details: NMI could be missed if the leading (falling) edge of $\overline{\text{NMI}}$ occurs between three-and-a-half and four-and-a-half CLKOUT cycles before the leading (falling) edge of $\overline{\text{IACK}}$. This can occur regardless of the timing of the actual interrupt. The critical timings involved are $\overline{\text{NMI}}$ and $\overline{\text{IACK}}$, as shown in Figure 3.

Figure 3. NMI Critical Timings



Workaround: None.

Advisory *Round (RND) Instruction Clears Pending Interrupts*

Revision(s) Affected: Initial Silicon, Revisions A, C, and D Silicon

Details: The RND (round) instruction opcode is decoded incorrectly and will write to the interrupt flag register (IFR) with the data from the data write bus (E bus). Therefore, it could cause the pending interrupt to be missed.

Workaround: Do not use the RND instruction. Replace the RND instruction with an ADD instruction as follows:

For this instruction ...	Use ...
`RND src[,dst]`	`ADD #1,15,src[,dst]`

Advisory *DMGFR Update of DMSFCn During DMA Autoinitialization*

Revision(s) Affected: Initial Silicon, Revisions A, C, and D Silicon

Details: When autoinitialization is used on the 5410 DMA controller, the Global Frame Count Reload Register (DMGFR) updates the entire Sync and Frame Count Register (DMSFCn) with its contents instead of only the frame count field as defined in the documentation. As a result, the first-pass block transfer on an autoinitialized channel works correctly, but once the channel context registers are reloaded, the DSYN and DBLW fields of the DMSFCn may be corrupted.

Workaround: Load the DMGFR with the desired contents for the entire DMSFCn (all 16 bits).

Advisory *DMA: Sync Event Miss Following External Transfer*

Revision(s) Affected: Initial Silicon, Revisions A, C, and D Silicon

Details: When a DMA channel that accesses external memory is followed by another channel that uses a sync event, the sync event on the latter channel may be missed. The channel affected is the channel serviced after the external channel. The channel-servicing order will be affected by which channels are enabled and the relative priority of the channels.

Workaround: Use one of the following workarounds:

1. Use a channel that does not require sync events to follow the external transfer channel (preferably configured as high priority).
2. Create a dummy channel following the external transfer channel with the following characteristics:
   - Internal-to-internal transfer
   - No sync event
   - High priority (causes this channel to always be serviced)
   - ABU mode (prevents the need to reset the channel after initialization)

Advisory *HPI Write*

Revision(s) Affected: Initial Silicon, Revisions A and C Silicon

Details: Some bits of data being written to the HPI by the host can be corrupted if the end of the HPI write occurs at the same time as multiple memory interface signals are switching. Specifically, this can happen if either the external memory address or data bus is switching at the rising edge of whichever signal is being used to write data into the HPI ($\overline{HCS}$ or $\overline{HDS1}$/$\overline{HDS2}$). The situation is most serious when either of these buses is switching many bits to the same state. HD4 and HD6 are the two most likely bits to be corrupted. The most common occurrence of this situation is when many address lines are switching from zero to one, and ones are being written to HD4 and/or HD6. The result is that HD4 and/or HD6 will receive a zero instead of a one.

Workaround: Avoid having HPI and external memory interface cycles at the same time.

Alternatively, information written to the HPI can be read back to verify that it was written correctly.

This has been fixed in Revision D of the silicon.

Advisory *Bootloader: 8-Bit Parallel Mode Destination Address and Block Size Calculation*

Revision(s) Affected: Initial Silicon, Revisions A and C Silicon

Details: When the bootloader is used in 8-bit parallel mode, the destination address and/or block size may be incorrectly generated because of a sign-extension error. This depends on the contents of the boot table and may not occur in all cases.

Major causes include:

Cause 1. External data bus line D15 is high while the bootloader reads the bus.

Cause 2. The extended address of one of the sections in the boot table falls between xx8000h and xxFFFFh.

Workaround: Cause 1 can be avoided by pulling down or driving the D15 pin low during bootload.

There is no workaround for Cause 2; it has been fixed in Revision D of the silicon.

Advisory *Bootloader: 8-Bit I/O Mode Data Concatenation*

Revision(s) Affected: Initial Silicon, Revisions A and C Silicon

Details: When the bootloader is used in 8-bit I/O mode, the data read from the boot table may be concatenated incorrectly due to a sign-extension error. This depends on the contents of the boot table and may not occur in all cases.

Workaround: This has been fixed in Revision D of the silicon.

Advisory *Bootloader: 8-Bit Parallel Mode*

Revision(s) Affected: Initial Silicon and Revision A Silicon

Details: In 8-bit parallel boot mode, the bootloader may incorrectly interpret the end-of-data blocks because of a failure to mask the upper bits in the accumulator when the end-of-block marker (0000h) is evaluated.

Workaround: D8–D15 should be pulled down or driven low during bootload in this mode. This problem has been fixed in Revision C of the silicon.

Advisory *Write Pending (1)*

Revision(s) Affected: Initial Silicon and Revision A Silicon

Details: A bus conflict can occur when both the peripheral (HPI/McBSP/DMA) and the CPU, using the "store" instruction, access the same block of single-access RAM (SARAM). When the DMA bus requests to access a SARAM block while the CPU executes dual-store or back-to-back store instructions to store values to the same SARAM block, an incorrect value is written into the memory for the second store instruction (same value as the first store value).The following instructions are affected:

DST	SACCD	ST||LD	ST||MAC
ST	STRCD	ST||ADD	ST||MACR
STH	SRCCD	ST||SUB	ST||MAS
STL			ST||MASR

See related advisories: *Write Pending (3)* and *Write Pending (4)*

Workaround: Choose one of the following workarounds (listed in order of ease-of-use):

1. Use DARAM for instances where the CPU and the DMA need to have simultaneous access to the same memory block.
2. If SARAM must be used, direct the CPU and the DMA to different memory blocks (see below).
3. Prevent the CPU from performing two consecutive writes by either inserting extra cycles (NOPs) between single-store instructions or converting any of the double-store or parallel-store instructions above to an equivalent with two single-store instructions separated by a NOP.

The RAM block boundaries on the 5410 are:

```
0060–07FF   DARAM 2K
0800–0FFF   DARAM 2K
1000–17FF   DARAM 2K
1800–1FFF   DARAM 2K
2000–3FFF   SARAM 8K
4000–5FFF   SARAM 8K
6000–7FFF   SARAM 8K
8000–9FFF   SARAM 8K
A000–BFFF   SARAM 8K
C000–DFFF   SARAM 8K
E000–FFFF   SARAM 8K
```

This problem has been fixed in Revision C of the silicon.

Advisory *Nested FRET*

Revision(s) Affected: Initial Silicon and Revision A Silicon

Details: Only one level of the far return can be implemented. When there are nested far calls or interrupts, only the innermost far return is executed correctly, while the outer or remaining far return(s) fail. The effect on the XPC is: while the first XPC that is restored is correct, the next XPC that is restored is not correct, but in fact, it is the XPC of the deepest nested context and these two values alternate for the next FRET(E) instructions executed.

Workaround: This problem has been fixed in Revision C of the silicon.

Advisory *Data Bus Holders*

Revision(s) Affected: Initial Silicon and Revision A Silicon

Details: The bus holders on the data bus do not function properly when enabled. The HPI data bus holders are unaffected and operate properly.

Workaround: The data bus holders should be disabled by clearing the BH bit in the BSCR. This problem has been fixed in Revision C of the silicon.

Advisory *BCLKS Internal Pullup Missing*

Revision(s) Affected: Initial Silicon and Revision A Silicon

Details: The internal pullup resistor for the BCLKS pin does not function. This does not impair the device functionally. It is an issue only for power consumption on the BCLKS pin while in IDLE modes.

Workaround: An external pullup can be used to control this pin if desired. This problem has been fixed in Revision C of the silicon.

Advisory *HPI/DMA Arbitration*

Revision(s) Affected: Initial Silicon and Revision A Silicon

Details: If the HPI8 makes a request to the DMA controller to use the DMA bus when the DMA bus is in the process of transitioning from idle to busy (performing a transfer), the HPI data (read or write) may be corrupted. If the HPI8 is not used, the DMA transfers operate correctly. If the HPI8 is the only user of the DMA bus, the HPI8 transfers operate correctly.

Workaround: This does not occur when the DMA bus does not enter an idle state. A DMA channel can be configured to perform dummy transfers to keep the DMA bus from going idle. These dummy transfers can be configured as low priority to avoid delays to higher-priority actual data.

This problem has been fixed in Revision C of the silicon.

Advisory *HPI: HRDY Stuck Low During Reset*

Revision(s) Affected: Initial Silicon and Revision A Silicon

Details: During reset, HRDY may be initially stuck low.

Workaround: Either of the following will correctly reset the HRDY logic:

1. Perform two initial dummy accesses (read or write) ignoring HRDY. A suggested method is to read both bytes of HPIC.
2. Do two device resets and perform the HPI8 activity during the second reset. In this case, the two dummy accesses are not necessary.

This problem has been fixed in Revision C of the silicon.

Advisory *Write Pending (3)*

Revision(s) Affected: Initial Silicon and Revision A Silicon

Details: This advisory is an extension of the *Write Pending (1)* situation that is also present on the 5410 silicon.

If the CPU attempts to perform a write to a given SARAM block at the same time the DMA system attempts to access the block, the CPU write will be pended to allow the higher-priority DMA access to proceed. The CPU write-pending data will be lost because the SARAM will not latch the CPU data while the DMA access is performed. Since the HPI8 always uses the DMA system for data transfers, HPI8 accesses appear to be DMA accesses to the SARAM; therefore, HPI8 accesses can also contribute to this situation.

This situation is not confined to a specific set of instructions as is the *Write Pending (1)* problem.

Workaround: Prevent the above described combination of accesses to SARAM from occurring in the same cycle by using either of the following workarounds:

1. Using DARAM for memory that must be accessed by the CPU and the DMA at the same time.
2. Directing CPU and DMA activity to different SARAM blocks.

This problem has been fixed in Revision C of the silicon.

Advisory *DMPREC*

Revision(s) Affected: Initial Silicon, Revisions A, C, and D Silicon

Details: When updating the DE bits of the DMPREC register while one or more DMA channel transfers are in progress, it is possible for the write to the DMPREC to cause an additional transfer on one of the active channels.

The problem occurs when an active channel completes a transfer at the same time that the user updates the DMPREC register. When the transfer completes, the DMA logic attempts to clear the DE bit corresponding to the complete channel transfer, but the register is instead updated with the CPU write (usually an ORM instruction) which can set the bit and cause an additional transfer on the channel. Refer to the example below for further clarification:

Example:

DMPREC value = 00C1h, corresponding to the following channel activity:

Channel 0 – enabled and running.	(DE0 = 1)
Channel 1 – disabled.	(DE1 = 0)
Channel 2 – disabled.	(DE2 = 0)
Channel 3 – disabled.	(DE3 = 0)
Channel 4 – disabled.	(DE4 = 0)
Channel 5 – disabled.	(DE5 = 0)

If the following conditions occur simultaneously:

Channel 0 transfer completes and DMA logic clears DE0 internally.

User code attempts to enable another channel (e.g., ORM #2, DMPREC)

The user code will re-enable channel 0 (DMPREC value written = 00C3h), and an additional, unintended transfer will begin on channel 0.

Workaround: There are a few use conditions under which this problem does not occur. If all active DMA channels are configured in ABU mode or in auto initialization mode, then the problem does not occur because the channels remain enabled until they are disabled by user code. The problem is also avoided in applications that use only one DMA channel at a time.

Systems that use multiple DMA channels simultaneously in multiframe mode, without autoinitialization are most likely to have this problem. In such systems one of the following methods can be used to avoid the problem:

- Always wait for all channels to complete existing transfers before re-enabling any channels, and always enable all channels at the same time.
- Before enabling a channel, check the progress of any on-going transfers by reading the element and frame counts of each active channel. If any active channel is within two element transfers of completing a block transfer, then wait until the active channel completes the block transfer before writing to the DMPREC register. Otherwise, if all active channels have more than two element transfers left in a block transfer, it is safe to update the DMPREC register.

Advisory	*CPU Read/DMA Write*

Revision(s) Affected: Initial Silicon, Revisions A, C, and D Silicon

Details: A CPU read of onchip memory can be corrupted if it reads the address that was last written by the CPU, while external DMA writes are occurring. The following characteristics apply to this problem:

- This problem happens when external DMA writes are active (i.e., whenever the destination of a DMA transfer is an external memory address (program, data, or IO)).
- The source of the DMA transfer is not relevant to the problem.
- The DMA and CPU do not have to be accessing the same block of memory for the problem to occur.
- This can affect CPU reads of either SARAM or DARAM memory blocks.

Workaround: The following workarounds apply:

1. Rearrange code such that CPU reads of the last address written by the CPU do not occur. For example in sections of code where this is a problem, always precede a CPU read with a write of a different address.
2. Do not use the DMA controller to perform external writes.

3 Documentation Support

For device-specific data sheets and related documentation, visit the TI web site at: **http://www.ti.com**

To access documentation on the web site:

1. Go to **http://www.ti.com**
2. Open the "**Products**" dialog box and choose "**Digital Signal Processors**"
3. Scroll to **"C54X™ DSP Generation"** and click on **"DEVICE INFORMATION"**
4. Click on a device name and then click on the documentation type you prefer.

For further information regarding the TMS320VC5410, please refer to:

- *TMS320VC5410 Fixed-Point Digital Signal Processor* data sheet*,* literature number SPRS075
- *TMS320C54x™ DSP Functional Overview,* literature number SPRU307

The five-volume *TMS320C54x DSP Reference Set,* literature number SPRU210, consisting of:

- *Volume 1: CPU and Peripherals,* literature number SPRU131
- *Volume 2: Mnemonic Instruction Set,* literature number SPRU172
- *Volume 3: Algebraic Instruction Set,* literature number SPRU179
- *Volume 4: Applications Guide,* literature number SPRU173
- *Volume 5: Enhanced Peripherals,* literature number SPRU302

The reference set describes in detail the TMS320C54x™ DSP products currently available and the hardware and software applications, including algorithms, for fixed-point TMS320™ DSP family of devices.

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.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265