TMS320VC5420

Digital Signal Processor

Silicon Errata

SPRZ169B
April 2000
Revised October 2001

TEXAS INSTRUMENTS

Contents

1 Introduction

This document describes the silicon updates to the functional specifications for the TMS320VC5420. The updates are applicable to:

- TMS320VC5420 (144-pin LQFP, PGE suffix)
- TMS320VC5420 (144 MicroStar BGA™, GGU 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 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.

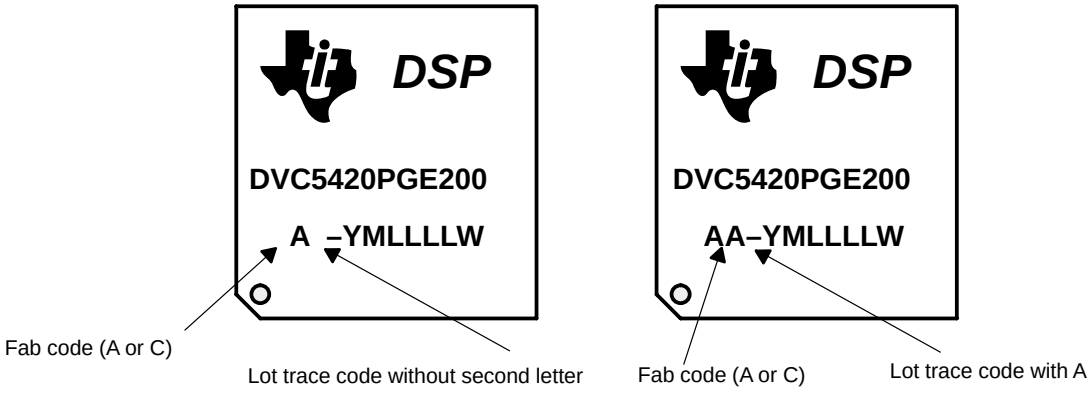
TMS Definition

Fully-qualified production device.

1.2 Revision Identification

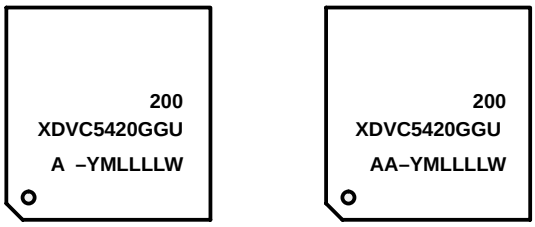
The device revision can be determined by the lot trace code marked on the top of the package. The locations for the lot trace codes for the PGE and the GGU packages are shown in Figure 1 and Figure 2, respectively. The location of other markings may vary per device.

Figure 1. Example, Typical Lot Trace Code for TMS320VC5420 (PGE)



Lot Trace Code	Silicon Revision
Blank No second letter in the Lot Trace Code	Indicates Original Silicon
A as second letter in the Lot Trace Code	Indicates Silicon Revision A
B as second letter in the Lot Trace Code	Indicates Silicon Revision B
C as second letter in the Lot Trace Code	Indicates Silicon Revision C
D as second letter in the Lot Trace Code	Indicates Silicon Revision D
E as second letter in the Lot Trace Code	Indicates Silicon Revision E
F as second letter in the Lot Trace Code	Indicates Silicon Revision F. Revision F is functionally equivalent to Revision E

Figure 2. Example, Typical Lot Trace Code for TMS320VC5420 (GGU)



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

Advisory *Write Pending #3*

Revision(s) Affected: All revisions

Details: This problem involves the write pending feature of the CPU (see background below for an explanation of write pending) and DMA accesses. When the DMA accesses an SARAM block that has a CPU write pending, the write pending feature is nullified. The result is that subsequent CPU reads of the address fail because the previous value of the memory location is read instead of the most recently written value.

Write Pending #3 (Continued)

Background Info: The 54x CPU includes a feature called write-pending that minimizes the impact of memory writes on CPU performance. When a CPU write and read occur to the same memory block, the write pending feature allows the CPU to continue execution without causing a pipeline stall. In this situation, the address and data to be written are temporarily latched in registers until the memory is available to be written. If a read of the same address occurs before the write actually completes in memory, the write pending feature causes the read to access the temporary registers instead of memory. When a write is pending, a subsequent write to any address in this memory block forces the pending write to occur by stalling the CPU, and then the new write is pended. So the write pending feature operates as a 1 level FIFO for writes to memory that is in use.

There are many sequences of instructions that cause a write pending condition, but in general this occurs whenever the pipeline stages for a CPU write and a CPU read of the same resource occur at the same time. In single access RAM (SARAM) this condition can occur whenever the execute (E) phase of an instruction that writes to memory takes place at the same clock cycle as the access (A), or read (R) phase of an instruction that reads from the same memory block. The condition can also occur in SARAM when the E phase of the write instruction takes place at the same clock cycle as the fetch (F) phase of any instruction being executed from the same block of memory. Therefore, the write pending condition occurs very often in SARAM. For example, if a segment of code is executing from a block of SARAM and performs a write to an address in the same block, the write will be pended until a subsequent write occurs, or until execution moves (branch, call, return, int, etc.) to a different memory block.

The write pending condition occurs less frequently in dual access RAM (DARAM) due to the fact that DARAM can support two accesses in a single cycle. However it is still possible to cause a write pending condition in DARAM, because the write operation during the E phase of a write instruction occurs in the same half–cycle as the read operation during the A phase of a dual operand read. See the dual access memory section in Chapter 7 of the *TMS320C54x DSP Reference Set, Volume 1: CPU and Peripherals* (literature number SPRU131) for a detailed explanation.

Workaround: The best possible workaround is to avoid ever causing a write pending condition in an SARAM block that will be accessed by the DMA. To do this, write conflicts with both program reads (fetches) and data reads must be avoided. To avoid conflicts with program fetches, an SARAM block should not be used to execute code that writes to the same block. Once this is avoided, conflicts with data reads can be avoided by insuring that writes to an SARAM block are not immediately followed by read instructions of the same block. This can be practically achieved by following all writes to SARAM block by NOPs or other instructions that do not read the SARAM block. At least 3 instruction cycles (e.g., 3 NOPs) should be inserted between a write instruction to SARAM and a subsequent read of the same block.

Another, less desirable, workaround is to avoid DMA accesses to any SARAM block that is being accessed by the CPU. For example, the CPU could operate on an SARAM block to prepare a buffer for transmission, and then the DMA can be enabled to access that block once the CPU has finished.

Advisory *DMPREC*

Revision(s) Affected: All revisions

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. Or,
- 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	*NMI*

Revision(s) Affected: All revisions

Details: An NMI can be ignored if the internal CPU interrupt logic is not adequately prepared.

Workaround: Avoid generating an NMI during the time when other interrupts are being serviced. Alternatively, use one of the other external interrupts, appropriately enabled, to serve the NMI function.

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

Revision(s) Affected: All revisions

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:

- 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.

- 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 bug is completely avoided.

- Avoid putting the called routine on other pages where a program memory address has the same lower 16 bits as the REA.

- Use the BANZ instruction as a substitute for the block repeat.

Advisory — *HPI HINT*

Revision(s) Affected: All revisions

Details: The HPI will become locked up, with HRDY stuck low, if both the host processor and the 5420 CPU write a one (1) to HINT at the same time.

Workaround: Do not perform redundant operations to the HINT bit. Both the HOST and the CPU should check to see if HINT is set before trying to write a one (1) to this bit.

For ...	IF ...	Then ...
the HOST	HINT is ***not*** set ...	Do not try to clear HINT by writing a one (1) to it, because the CPU may try to set it.
the CPU	HINT is ***already*** set...	Do not try to set HINT again by writing a one (1) to it, since the HOST may try to clear it.

Advisory — *WRITA/MVDP*

Revision(s) Affected: All revisions

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: Use one of the following workarounds:

- 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	*HPI/FIFO Bus Conflict*

Revision(s) Affected: All revisions

Details: A FIFO transmit will corrupt the internal DMA bus for both HPI reads and writes. If a FIFO receive is done after the FIFO transmit, the bus condition will be cleared and the HPI can successfully access the bus.

Workaround: Use one of the following workarounds.

- In HPI mux mode operation, HPIC can be used to interrupt the host before a FIFO transmit; however, this is not supported for HPI non-mux mode.
- A partial workaround for HPI non-mux mode is to keep an unused DMA channel busy with dummy FIFO receives to an ABU in a low-priority mode. However, this will only reduce the chance of conflict. The HPI could potentially attempt an access during the time when the transmit channel goes inactive and the receive channel goes active.

  No corrections are planned for future revisions.

Advisory	*Boundary Scan*

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

Details: Boundary scan cannot be performed and when EXTEST or SAMPLE/PRELOAD modes are selected, the scan path will break, causing board tests to fail. The bypass function of the scan path operates correctly. The affected PG revisions can be included in the scan chain if the boundary scan ATPG tools are configured to ALWAYS keep these devices in BYPASS mode.

Workaround: The BSDL files provided for Revision E can be used to set up the description of the board under test. If Revision D or lower devices are present on the board, the ATPG tool should generate the board test such that the devices are not operated in EXTEST or SAMPLE/PRELOAD. Most ATPG tools will provide the capability to define which devices on the board should be tested and which devices should be disabled (in BYPASS). If these devices are configured in BYPASS, tests on other boundary scan devices on the board can be performed.

This problem is corrected in Revision E silicon.

Advisory *Round (RND) Instruction Clears Pending Interrupts*

Revision(s) Affected: Initial Silicon and Revision A 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: Replace the RND instruction with an ADD instruction as follows:

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

Advisory *Write Pending #1*

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

Details: Consecutive writes to SARAM using the DST instruction or back-to-back store instructions such as STL or STH can be corrupted when the DMA also attempts to access the same SARAM block. The DMA accesses can be from any source such as HPI or McBSP, and can be either reads or writes. The corruption that occurs is that the data written to one location is not the data specified by the program, but the same data as the previous write.

Workaround: Avoid using the DST instruction and back-to-back STORE instructions such as STL or STH when the DMA is accessing the same SARAM block.

This problem is corrected in Revision B silicon.

Advisory *Far Return (FRET)*

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

Details: The Far Return (FRET) instruction interferes with a memory write operation when the write is followed by the FRET instruction. The FRET command will corrupt the previous memory write.

Workaround: Arrange code so that memory writes are not followed by the FRET instruction. NOPs can also be placed between memory writes and FRET instructions.

This problem is corrected in Revision B silicon.

Advisory	*Write Pending #2*

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

Details: A bus conflict can occur when the CPU is required to arbitrate simultaneous write-pending and read operations from the same SARAM block. When this arbitration occurs, incorrect data is written into memory when using the following instructions: DELAY, FCALL, LTD, MACD, MVDD, MVDK, MVDM, MVDP, MVMD, MVKD, PORTR, PORTW, POPD, POPM, PSHD, PSHM, SRCCD, STRCD, ST(#lk,SMEM), and STM(#lk,MMR). The write-pending bug is a write from such instructions as STL(A,*AR2) and STL(A,SMEM).

Workaround: Use one of the following workarounds.

- Avoid using the SARAM block if possible.
- Rearrange the code so the described instructions do not immediately follow the instruction which caused the write pending status.
- Use different SARAM blocks for the stack pointer if having problems with the call instruction.

This problem is corrected in Revision B silicon.

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 select "**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 TMS320VC5420, please refer to:

- *TMS320VC5420 Fixed-Point Digital Signal Processor* data sheet*,* literature number SPRS080
- *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 generation of 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