TMS320VC5471

Digital Signal Processor

Silicon Errata

SPRZ189F
June 2001 – Revised May 2003

TEXAS INSTRUMENTS

REVISION HISTORY

This revision history highlights the technical changes made to SPRZ189E to make it an SPRZ189F revision.

Scope: This document has been reviewed for technical accuracy; the technical content is up-to-date as of the specified release date including the following changes.

PAGE(s) NO.	ADDITIONS/CHANGES/DELETIONS
7	Revised “UART (Modem or IrDA) Interrupt Reset” advisory.

Contents

1 Introduction

This document describes the silicon updates to the functional specifications for the TMS320VC5471 silicon. The advisories are applicable to the TMS320VC5471 (257-ball MicroStar BGA™, GHK 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.

1.2 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 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

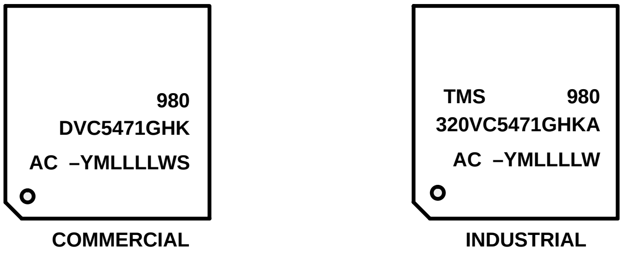
MicroStar BGA and TMS320 are trademarks of Texas Instruments.
All trademarks are the property of their respective owners.

TMX and TMP devices and TMDX development-support tools are shipped with appropriate disclaimers describing their limitations and intended uses. Experimental devices (TMX) may not be representative of a final product and Texas Instruments reserves the right to change or discontinue these products without notice.

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.

The device revision can be determined by the lot trace code marked on the top of the package. The location for the lot trace codes for the GHK package is shown in Figure 1. The location of other markings may vary per device.



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

Figure 1. Example, Typical Lot Trace Code for TMS320VC5471 (GHK)

2 Known Design Marginality/Exceptions to Functional Specifications

Table 1. Summary of Advisories

Description	Revision Affected	Page
WRITA/MVDP	Initial Silicon	6
UART (Modem or IrDA) Interrupt Reset	Initial Silicon	7
Far Branches/Calls/Interrupts from Active Repeat Blocks (BRAF)	Initial Silicon	8
Round (RND) Instruction Clears Pending Interrupts	Initial Silicon	9
Potential Bus Contention Between FLASH Reads and SDRAM Writes	Initial Silicon	9
Read/Reset of the SPI_STATUS Register is Not Working Properly	Initial Silicon	9
SDRAM 16-Bit-Wide Little-Endian Mode	Initial Silicon	10
SDRAM and FLASH Cannot Both be 16-Bit-Wide Busses	Initial Silicon	10
Ethernet Port RX Overrun In Half-Duplex Mode	Initial Silicon	10
Ethernet Byte Length Sensitivity	Initial Silicon	11
Ethernet Hang	Initial Silicon	11
Ethernet CRC Errors	Initial Silicon	12
TX Underrun Errors Erroneously Reported for 65-Byte and 66-Byte Size Ethernet Packets	Initial Silicon	12
Minimum SPI PTV of 1 Can Cause Errors on Rising Edge-Triggered Reads	Initial Silicon	12
SPI RE and WE Status Complete is Indicated Before the Last Bit is Processed	Initial Silicon	13
Reset During 16M/64M SDRAM Accesses Can Prevent the Device From Rebooting	Initial Silicon	14
ARM/WAIT Does Not Operate Correctly	Initial Silicon	15

Advisory *WRITA/MVDP*

Revision(s) Affected: Initial 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: Use one of the following workarounds:

1. Avoid using WRITA/MVDP to write to an area in memory that will be executed as code immediately following the WRITA/MVDP.
2. 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.
3. Avoid DMA reads in an area of program SARAM that is written to by WRITA/MVDP.

Advisory — *UART (Modem or IrDA) Interrupt Reset*

Revision(s) Affected: Initial Silicon

Details: There are two interrupt conditions, which can cause undesirable UART (Modem or IrDA) behavior. These two interrupt conditions can be found in IT_TYPE [bits 5:1] of UART_ISR (FFFF:1028) or UART_IRDA_ISR (FFFF:0828) as defined below:

- 00110 => RX Time Out (priority level 2)
- 00010 => RHR Interrupt (priority level 2)

Workaround: When any UART (Modem or IrDA) interrupt is received, check the IT_TYPE for ether an RX Time Out or RHR Interrupt as defined above.

If the interrupt has been set due to either the RX Time Out or RHR Interrupt, then perform the following maneuvers to reset the interrupt condition:

1. Disable all the interrupts by writing ‘0’ into UART_IER (FFFF:1024) or UART_IRDA_IER (FFFF:0824).
2. Read from UART_RHR (FFFF:1000) or UART_IRDA_RHR (FFFF:0800) until the FIFO is empty. The FIFO will be empty when UART_LSR (FFFF:1014) or UART_IRDA_LSR (FFFF:0814) is read with ‘0’ in RX_FIFO_E [bit 0].
3. If the RHR read does not clear the interrupt, use the RESET_REG (FFFF:2F18), UART_MODEM_RESET [bit 8] or UART_IRDA_RESET [bit 7], to reset the UART (Modem or IrDA) peripheral. The peripheral reset should be active for at least 8 ARM CPU clock cycles. After reset, re-initialize all of the UART (Modem or IrDA) registers to the proper settings needed for the application.
4. Return to normal operation.
5. If the interrupt has been set due to any other interrupt condition, then the interrupt can be reset normally.
6. To reduce the possibility of RX Time Out and/or RHR interrupts, match the RX FIFO size setting to the expected number of words.

If the UART (Modem or IrDA) is unused:

- Set the CLKM_REG (FFFF:2F00) block clock stop bit, UART_MODEM_CLK_STOP [bit 9] or UART_IRDA_CLK_STOP [bit 8]. This will disable the peripheral clock until the UART is actually needed. Also, set the WAKEUP_REG (FFFF:2F08) bit, UART_MODEM_WAKEUP [bit 9] or UART_IRDA_WAKEUP [bit 8]. This will prevent the UART (Modem or IrDA) clock from starting, if any interrupt is detected. This value is copied to CLKM_REG, if an interrupt occurs.

No corrections are planned for future devices.

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

Revision(s) Affected: Initial 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 — *Round (RND) Instruction Clears Pending Interrupts*

Revision(s) Affected: Initial 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 — *Potential Bus Contention Between FLASH Reads and SDRAM Writes*

Revision(s) Affected: Initial Silicon

Details: On reads from FLASH, the output enable to data bus three-state could be slow enough to cause reliability contention with SDRAM write operations.

Workaround: Avoid this reliability contention by using the internal SRAM to run code and intermediate data from the FLASH. Also, all stacks should be located in internal SRAM.

No corrections are planned for future revisions.

Advisory — *Read/Reset of the SPI_STATUS Register is Not Working Properly*

Revision(s) Affected: Initial Silicon

Details: When the external CLKX_SPI prescaled SPI_CLOCK clock is active (which could be every 4, 8, 16, 32, 64, 128, 256 SPI_CLOCK events depending on the state of PTV in SPI_SET (FFFF:2000)) on the same clock cycle as the read/reset, then the reset will not take place as expected. This creates an inconsistent SPI status register reset phenomena.

Workaround: There are three potential workarounds.

1. Perform a read loop until the reset is detected.
2. Use a slower CLKX_SPI clock rate to improve the odds of a reset the first time. This should be incorporated with number 1 above.
3. Turn off the SPI_CLOCK at CLKM_REG (FFFF:2F00) during the read/reset ensuring that the CLKX_SPI will not interfere with the register reset. The WAKEUP_REG register (FFFF:2F08) should be changed to match the CLKM_REG in case an interrupt occurs prior to the read/reset.

No corrections are planned for future revisions.

Advisory — *SDRAM 16-Bit-Wide Little-Endian Mode*

Revision(s) Affected: Initial Silicon

Details: The SDRAM Big-/Little-Endian modes are supported as follows:

- SDRAM 32-bit-wide Big-Endian mode is supported
- SDRAM 16-bit-wide Big-Endian mode is supported
- SDRAM 32-bit-wide Little-Endian mode is supported
- SDRAM 16-bit-wide Little-Endian mode is ***not*** supported

Workaround: If Little-Endian operation is required, use only the 32-bit-wide mode. No corrections are planned for future revisions.

Advisory — *SDRAM and FLASH Cannot Both be 16-Bit-Wide Busses*

Revision(s) Affected: Initial Silicon

Details: For a 32-bit FLASH read following a 32-bit SDRAM write, the first 16-bit word of the 32-bit FLASH read can give incorrect data due to improper address generation. This condition can occur when both the SDRAM and FLASH are configured for 16-bit-wide data busses. Other combinations of 16-bit and 32-bit-wide busses do not exhibit this behavior as described in the following:

- 16-bit-wide FLASH and 16-bit-wide SDRAM busses can fail.
- 16-bit-wide FLASH and 32-bit-wide SDRAM busses, no failures.
- 32-bit-wide FLASH and 16-bit-wide SDRAM busses, no failures.
- 32-bit-wide FLASH and 32-bit-wide SDRAM busses, no failures.

Workaround: No workaround exists. No corrections are planned for future revisions.

Advisory — *Ethernet Port RX Overrun In Half-Duplex Mode*

Revision(s) Affected: Initial Silicon

Details: An error in the Ethernet State Machine (ESM) may eventually cause an RX overrun condition at the receiving Ethernet port when operating in half-duplex mode with a busy Ethernet port (90% utilization) receiving data and attempting to transmit out of that same Ethernet port. At high levels of receiver utilization in half-duplex mode, the transmit side of the port gets virtually no throughput. At the beginning of a transmit packet, the state machine will properly monitor both the RX and TX operations. However, if the transmit packet is partially sent, the ESM can get hung in a state where it will no longer monitor RX activity. Thus the receiving side of the port is getting all of the bandwidth, effectively blocking all transmit packets, and RX overrun will eventually occur.

Workaround: Disable the ESM. No corrections are planned for future revisions.

Advisory — *Ethernet Byte Length Sensitivity*

Revision(s) Affected: Initial Silicon

Details: Ethernet throughput and packet loss tests reveal a regular pattern of four good packet sizes followed by four bad packet sizes as shown in the modulo 8 (or 4) example below:

- Packet size = 197 to 200, good throughput.
- Packet size = 201 to 204, mediocre throughput.
- Packet size = 205 to 208, good throughput.
- Packet size = 209 to 212, mediocre throughput.

This behavior is apparent at 100 Mbs full-duplex and less prevalent at 10 Mbs half-duplex. The root cause of the problem has been identified in the buffer manager FIFO which can continually produce underrun errors. Once a bad state is entered, the port remains in that bad state for all successive transmit packets. Each transmit packet is truncated, causing CRC errors.

The buffer memory underflow (UFLW) bit 0 of the EIM_SE_SR_E0 (FFFF:0118) status register is set.

Workaround: A software workaround exists.

Once the underrun state is detected, the UFLW status can be used through software polling to force a 65-byte packet from the CPU port to the ENET0 port to clear this condition.

After this event, the Ethernet port will behave properly, but can still revert back to improper behavior. Thus, constant surveillance and correction are needed to maintain optimum Ethernet performance.

No corrections are planned for future revisions.

Advisory — *Ethernet Hang*

Revision(s) Affected: Initial Silicon

Details: The Ethernet port hangs with the following:

- 10 Mbs half-duplex, utilization below 80% good, above 80% bad.
- 10 Mbs full-duplex, utilization below 80% good, above 80% bad.
- 100 Mbs half-duplex, utilization below 10% good, above 25% bad.
- 100 Mbs full-duplex, any amount of utilization can cause a hang.

Investigation of the hardware has uncovered that this issue is related to the Ethernet byte length sensitivity advisory. Basically, 50% of the time the random packet size can hit the poor performing modulo 8 area. Under heavy Ethernet traffic, this will eventually cause TX underrun errors. As different sized packets continue to pass through the ESM, the Ethernet port will eventually hang.

Workaround: A software workaround exists. See Ethernet Byte Length Sensitivity advisory workaround.

No corrections are planned for future revisions.

Advisory — *Ethernet CRC Errors*

Revision(s) Affected: Initial Silicon

Details: Ethernet CRC errors are predominantly produced as a result of the Ethernet byte length sensitivity errata. Once this failure mechanism commences, the FIFO buffer counter gets out of sync with the data, eventually leading to a loss of eight data bytes transferred, which causes CRC errors. Once this error state is entered, the port continues to send TX packets with eight bytes missing causing a large number of CRC errors.

Workaround: A software workaround exists. See Ethernet Byte Length Sensitivity advisory workaround.

No corrections are planned for future revisions.

Advisory — *TX Underrun Errors Erroneously Reported for 65-Byte and 66-Byte Size Ethernet Packets*

Revision(s) Affected: Initial Silicon

Details: Spurious TX underrun errors are reported for 65- and 66-byte packets running through the ESM. The software workaround for the Ethernet Byte Length Sensitivity advisory relies on the TX underrun status interrupt. The small packet underrun errors cause false error detection.

Workaround: The TX underrun errors do not signify a real Ethernet error and should be ignored for these small packet sizes.

No corrections are planned for future revisions.

Advisory — *Minimum SPI PTV of 1 Can Cause Errors on Rising Edge-Triggered Reads*

Revision(s) Affected: Initial Silicon

Details: If the Prescale Clock Divisor (PTV) value in the SPI setup register is 1 (SPI_SET register bits [2:0] = 000) and the SPI is reading with a rising edge clock trigger, the DI serial read data will be incorrectly clocked into the SPI.

Workaround: Do not use a PTV value of 1. For a minimum, use a PTV value of 2 (SPI_SET register bits [2:0] = 001). This gives a minimum cycle time for CLKX_SPI of 168.4 ns with a minimum MCU cycle time of 21.05 ns; CLKX_SPI = (MCU Clock / (4*2)).

No corrections are planned for future revisions.

Advisory *SPI RE and WE Status Complete is Indicated Before the Last Bit is Processed*

Revision(s) Affected: Initial Silicon

Details: The SPI performs a transmission or reception of the last bit properly, but it incorrectly sets the SPI Status Register Write End (WE: SPI_STATUS bit 1) or Read End (RE: SPI_STATUS bit 0) bits before completion of the last bit. The status is not complete until the serial port clock (CLKX_SPI) runs through its last period and external enable (MCUEN) deactivation.

Depending on the control settings of the SPI, it can take up to $2^1/_2$ CLKX_SPI clock cycles to complete the SPI process after WE or RE are set to 1. Also, with a minimum setting of the prescale clock divisor (PTV), this could be as many as 640 MCU clocks (4 * 64 * $2^1/_2$).

Workaround: None.

No corrections are planned for future revisions.

Advisory	*Reset During 16M/64M SDRAM Accesses Can Prevent the Device From Rebooting*

Revision(s) Affected: Initial Silicon

Details: When the SD_SIZE bits in the SDRAM_CONFIG register have been programmed for 16M or 64M sizes, the “Full Page Burst” option will be incorrectly selected. If a device reset occurs during an SDRAM access, the SDRAM (which does not have a hardware reset) continues to burst data. After the reset, the MCU will never receive its first instruction because the external boot memory (i.e., FLASH) shares the SDRAM bus.

Proper SDRAM initialization will occur when the SD_SIZE is either 128M or 256M.

Workaround: Temporarily changing the SDRAM Size bits before and after the SDRAM initialization procedure will not affect the SDRAM size, since it is not a parameter of the SDRAM initialization procedure. The SDRAM initialization procedure should perform the following steps:

1. Load the SDRAM_CONFIG register with the desired CAS Latency.
2. If SD_SIZE bit 20 in the SDRAM_CONFIG register is ‘0’ binary, then temporarily rewrite this bit to ‘1’ binary while the SDRAM initialization is in progress. Changing this bit will not affect the SDRAM Size, but will allow proper setting of the Burst Length.
3. Load the SDRAM_INIT_CONF register with an appropriate number of refresh cycles to be performed once the SDRAM is in the IDLE state. In addition, load this register with a clock cycle count larger than the delay required by the SDRAM. Typically, this delay is about 100 μs. However, consult the SDRAM data sheet for the exact value required.
4. Start the SDRAM initialization procedure by activating the SDRAM_INIT bit in SDRAM_CNTL. There is no need to reset this bit, since it is self-clearing.
5. Wait for completion of the SDRAM initialization procedure by checking the READY bit in the SDRAM_CNTL register. When READY is active, the SDRAM initialization is complete and the SDRAM is ready for normal operation.
6. If the SD_SIZE bit 20 in the SDRAM_CONFIG register was adjusted from ‘0’ binary to ‘1’ binary for the SDRAM initialization, then change it back to ‘0’ binary.

No corrections are planned for future revisions.

Advisory — *ARM/WAIT Does Not Operate Correctly*

Revision(s) Affected: Initial Silicon

Details: ARM/WAIT is improperly synchronized to ARM_MCLK. Metastability issues can occur. Also, if ARM/WAIT is active at any time during a wait state counter cycle, another entire wait state cycle is restarted at the end of the present cycle versus the intended behavior of extending the bus access by a single ARM_MCLK cycle at a minimum.

Workaround: None.

Do not use ARM/WAIT and tie the pin high with an external pullup. Increase the wait state setting for external memory spaces connected to slow external devices through the WS bits in the external memory control register (CS0_REG–CS4_REG: bits 4-0).

No corrections are planned for future revisions.

3 Documentation Support

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

For further information regarding the TMS320VC5471, please refer to:

- *TMS320VC5471 Fixed-Point Digital Signal Processor* data manual, literature number SPRS180
- *TMS320VC547x CPU and Periphera*ls *Reference Guide,* literature number SPRU038
- *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.

TMS320C54x is a trademark of Texas Instruments.

IMPORTANT NOTICE

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265