*TMS570LS12x/11x Microcontroller*

## Silicon Revision A

# Silicon Errata

Literature Number: SPNZ186
September 2012

# Contents


1 Device and Development-Support Tool Nomenclature ..... 4
2 Revision Identification ..... 5
3 Known Design Exceptions to Functional Specifications ..... 6

## List of Figures

1 Device Revision Code Identification ..... 5

## List of Tables

1 Device Revision Codes ..... 5
2 Known Design Exceptions to Functional Specifications ..... 6

# TMS570LS12x/11x Microcontroller

This document describes the known exceptions to the functional specifications for the device.

## 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 devices and support tools. Each 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.

TI device nomenclature also includes a suffix with the device family name. This suffix indicates the silicon revision, the package type and the temperature range (for example, "Q" is -40 to 125C).

## 2 Revision Identification

Figure 1 provides examples of the TMS570LSx device markings. The device revision can be determined by the symbols marked on the top of the device.

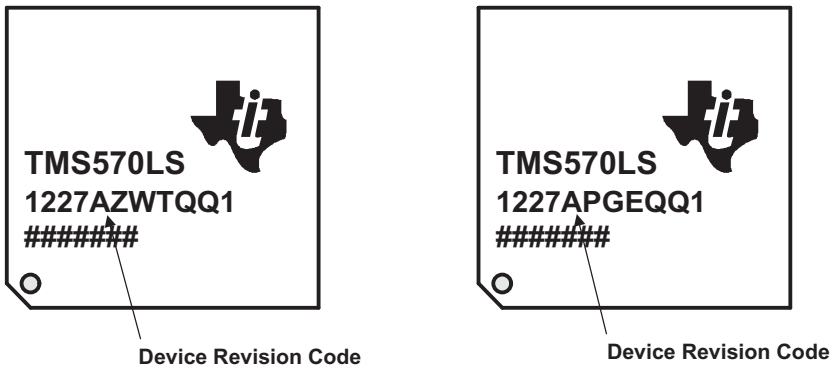


**Figure 1. Device Revision Code Identification**

Silicon revision is identified by a device revision code. The code is of the format TMS570LS1203x, where "x" denotes the silicon revision. If x is "A" in the device part number, it represents silicon version A. Table 1 lists the information associated with each silicon revision.

**Table 1. Device Revision Codes**

| DEVICE PART NUMBER DEVICE REVISION CODE | SILICON REVISION | PART NUMBERS/COMMENTS |
|---|---|---|
| TMS570LS12x/11x | A | This silicon revision is available as X *only*. TMS570LS12x/11x |

## 3 Known Design Exceptions to Functional Specifications

The following table lists the known exceptions to the functional specifications for the device.

**Table 2. Known Design Exceptions to Functional Specifications**


| Title | Page |
|---|---|
| **AHB_ACCES_PORT#3 (ARM ID-529470)** —DAP AHB-AP Access Port Might Fail To Return Slave Error On AHB Reset. | 7 |
| **BCG_IP_P_EMIF#241 (EMIF#241)** —EMIF Reports Incorrect Time-out Error Data | 8 |
| **CORTEX-R4#26 (ARM ID-577077)** — Thumb STREXD Treated As NOP If Same Register Used For Both Source Operands | 9 |
| **CORTEX-R4#27 (ARM ID-412027)** —Debug Reset Does Not Reset DBGDSCR When In Standby Mode | 10 |
| **CORTEX-R4#33 (ARM ID-452032)** —Processor Can Deadlock When Debug Mode Enables Cleared | 11 |
| **CORTEX-R4#46 (ARM ID-599517)** —CP15 Auxiliary ID And Prefetch Instruction Accesses Are UNDEFINED | 12 |
| **CORTEX-R4#55 (ARM ID-722412)** —CPACR.ASEDIS And CPACR.D32DIS Incorrect When Configured With Floating Point | 13 |
| **CORTEX-R4#57 (ARM ID-737195)** —Conditional VMRS APSR_Nzcv, FPSCR May Evaluate With Incorrect Flags | 14 |
| **CORTEX-R4#58 (ARM ID-726554)** —DBGDSCR.Adadiscard Is Wrong When DBGDSCR.Dbgack Set | 15 |
| **CORTEX-R4#61 (ARM ID-720270)** —Latched DTR-Full Flags Not Updated Correctly On DTR Access | 16 |
| **CORTEX-R4#66 (ARM ID-754269)** —Register Corruption During A Load-Multiple Instruction At An Exception Vector | 17 |
| **CORTEX-R4#67 (ARM ID-758269)** —Watchpoint On A Load Or Store Multiple May Be Missed. | 18 |
| **DEVICE#B053** —CPU code execution could be halted on a device warm reset if the core power domain # 2 is disabled by software. | 19 |
| **DEVICE#B056** —Error flags are incorrectly set in the Error Signaling Module upon a device warm reset for modules in core power domains that are disabled | 20 |
| **DEVICE#B058** —Device I/Os Could Toggle After Warm Reset | 21 |
| **DEVICE#142** —CPU Abort Not Generated on Write to Unimplemented MCRC Space | 22 |
| **DEVICE#070** —Some multiplexing controls are not compatible with the emulation device | 23 |
| **DMA#27** —DMA Requests Lost During Suspend Mode | 24 |
| **ERAY#52 (FLEXRAY#52)** —WUS Generates Redundant SIR.WUPA/B Events | 25 |
| **ERAY#58 (FLEXRAY#58)** —Erroneous Cycle Offset During Startup after abort of startup or normal operation | 26 |
| **ERAY#59 (FLEXRAY#59)** —First WUS Following Received Valid WUP May Be Ignored | 27 |
| **ERAY#60 (FLEXRAY#60)** —READY Command Accepted In READY State | 28 |
| **ERAY#61 (FLEXRAY#61)** —Slot Status vPOC!SlotMode Is Reset Immediately When Entering HALT State | 29 |
| **FTU#8** —FlexRay Transfer Unit Not Disabled On Memory Protection Violation (MPV) Error | 30 |
| **FTU#19** —TCCOx Flag Clearing Masked | 31 |
| **FTU#23** —Bus errors on transactions not initiated by FTU could get captured by FTU | 32 |
| **ANALOGIP_021_LPO#16 (F021 LPO#16)** —Oscillator Fault Detection Starts too Soon | 33 |
| **MCRC#18** —CPU Abort Generated on Write to Implemented MCRC Space After Write to Unimplemented MCRC Space | 34 |
| **MIBSPI#110** —Multibuffer Slave In 3 or 4 Pin Mode Transmits Data Incorrectly for Slow SPICLK Frequencies and for Clock Phase = 1 | 35 |
| **MIBSPI#111** —Data Length Error Is Generated Repeatedly In Slave Mode when I/O Loopback is Enabled | 36 |
| **NHET#52** —WCAP May not Capture the High-Resolution Offset Correctly | 37 |
| **NHET#53** —Enhanced Input Capture Pins May not Capture Small Pulses Correctly | 38 |
| **SYS#46** —Clock Source Switching Not Qualified With Clock Source Enable And Clock Source Valid | 39 |
| **SYS#102** —Bit field EFUSE_Abort[4:0] in SYSTASR register is read-clear | 40 |
| **SYS#111** —The Device may Generate Multiple nRST Pulses. | 41 |
| **VIM#27** —Unexpected phantom interrupt | 42 |

## AHB_ACCES_PORT#3 (ARM ID-529470) *DAP AHB-AP Access Port Might Fail To Return Slave Error On AHB Reset.*

**Severity** Medium

**Expected Behavior** If an AHB reset occurs (HRESETn is asserted LOW) while the AHB-AP is performing an access on the AHB bus, the AHB-AP should return a slave error back to the JTAG-DP.

**Issue** Instead, the AHB-AP might indicate that the access completed successfully and return unpredictable data if the access was a read.

**Condition** HRESETn is asserted LOW on a specific cycle while the AHB-AP is completing an access on the AHB bus.

**Implication(s)** This will make it harder to identify cases where the AHB bus is being reset. However, this should not affect most usage models because it is not generally possible to debug over a bus when it is likely to be reset.

**Workaround(s)** Avoid performing debug operations while the AHB bus might be reset. Resets usually occur following powerdown, and so this can usually be achieved through the use of features that prevent powerdown during debug.

## BCG_IP_P_EMIF#241 (EMIF#241) *EMIF Reports Incorrect Time-out Error Data*

**Severity** Medium

**Expected Behavior**

**Issue** Under certain conditions, the EMIF will report a time-out error (vbusp_rstatus=0x3) on the VBUSP read status interface.The data returned for this access will be all zeros. If this access is followed by a read to the EMIF's MMR space, the EMIF will still report a time-out error (vbusp_rstatus=0x3) but with the correct data for the MMR read.

**Condition** This issue is only applicable if all of the following are true:

1. The EMIF is used for async memory accesses in Extended Wait mode.
2. There is a potential for a time-out error to occur. For example, the async memory will not de-assert the pad_wait_i input.
3. If an async memory read with time-out error is followed by an MMR read.
4. The EMIF's vbusp_rstatus output is used by the infrastructure to zero out the read data when time-out error is reported.

**Implication(s)** The EMIF will hold vbusp_rstatus=0x3 until another async access without time-out error or an SDRAM access is performed.

**Workaround(s)** If a timeout error occurs, perform a dummy read to SDRAM, or to another async chip select that is not configured to be in Extended Wait mode, or to the same async chip select after disabling the Extended Wait mode on that chip select.

## CORTEX-R4#26 (ARM ID-577077) *Thumb STREXD Treated As NOP If Same Register Used For Both Source Operands*

**Severity** Medium

**Expected Behavior**

**Issue** The ARM Architecture permits the Thumb STREXD instruction to be encoded with the same register used for both transfer registers (Rt and Rt2). Because of this erratum, the Cortex-R4 processor treats such an encoding as UNPREDICTABLE and executes it as a NOP.

**Conditions** This occurs when the processor is in Thumb state and a STREXD instruction is executed which has Rt = Rt2. This instruction was new in ARM Architecture version 7 (ARMv7). It is not present in ARMv6T2 or other earlier architecture versions.

**Implications** If this occurs the destination register, Rd, which indicates the status of the instruction, is not updated and no memory transaction takes place. If the software is attempting to perform an exclusive read-modify-write sequence, then it might either incorrectly complete without memory being written, or loop forever attempting to complete the sequence.

**Workaround(s)** This can be avoided by using two different registers for the data to be transferred by a STREXD instruction. This may involve copying the data in the transfer register to a second, different register for use by the STREXD.

## CORTEX-R4#27 (ARM ID-412027) *Debug Reset Does Not Reset DBGDSCR When In Standby Mode*

**Severity**

Medium

**Expected Behavior**

The debug reset input, PRESETDBGn, resets the processor's debug registers as specified in the ARMv7R Architecture. The debug reset is commonly used to set the debug registers to a known state when a debugger is attached to the target processor.

**Issue**

When the processor is in Standby Mode and the clock has been gated off, PRESETDBGn fails to reset the Debug Status and Control Register (DBGDSCR).

**Conditions**

- The DBGDSCR register has been written so that its contents differ from the reset values (most fields in this register reset to zero, though a few are UNKNOWN at reset)
- The processor is in Standby Mode, and the clocks have been gated off, that is STANDBYWFI is asserted
- The debug reset, PRESETDBGn, is asserted and deasserted while the processor clocks remain gated off

**Implications**

If this occurs, then after the reset the DBGDSCR register contains the values that it had before reset rather than the reset values. If the debugger relies on the reset values, then it may cause erroneous debug of the processor. For example, the DBGDSCR contains the ExtDCCmode field which controls the Data Communications Channel (DCC) access mode. If this field was previously set to Fast mode but the debugger assumes that it is in Non-blocking mode (the reset value) then debugger accesses to the DCC will cause the processor to execute instructions which were not expected.

**Workaround(s)**

This can be avoided by a workaround in the debug control software. Whenever the debugger (or other software) generates a debug reset, follow this with a write of zero to the DBGDSCR which forces all the fields to their reset values.

## CORTEX-R4#33 (ARM ID-452032) *Processor Can Deadlock When Debug Mode Enables Cleared*

**Severity** Medium

**Expected Behavior** The Cortex-R4 processor supports two different debugging modes: Halt-mode and Monitor-mode, or both modes can be disabled. Bits [15:14] in the Debug Status and Control Register (DBGDSCR) control which, if any, mode is enabled. Debug events can be generated by the breakpoint unit, matching instructions executed. Deadlocks should not occur when the debug mode is changed.

**Issue** If there are active breakpoints at the time when the debugging mode is changed, this can cause the processor to deadlock.

**Conditions**

- At least one breakpoint is programmed and active
- Either halt-mode debugging or monitor mode debugging is enabled and the debug enable input, DBGEN is HIGH
- An instruction which matches a breakpoint is fetched and executed
- After the instruction is fetched, but before it has completed execution, both halt-mode and monitor-mode debugging are disabled by means of a write to DBGDSCR

**Implications** Changing the debugging mode while breakpoints or watchpoints are active is UNPREDICTABLE. Therefore the above conditions cannot be met in normal (recommended) operation of the processor. However, if the conditions do occur, then the processor will deadlock, which is outside the bounds of permitted UNPREDICTABLE behavior.

**Workaround(s)** None

## CORTEX-R4#46 (ARM ID-599517) *CP15 Auxiliary ID And Prefetch Instruction Accesses Are UNDEFINED*

**Severity**

Medium

**Expected Behavior**

The ARMv7-R architecture requires implementation of the following two features in CP15 :

1. An Auxiliary ID Register (AIDR), which can be read in privileged modes, and the contents and format of which are IMPLEMENTATION DEFINED.
2. The operation to prefetch an instruction by MVA, as defined in the ARMv6 architecture, to be executed as a NOP.

**Issue**

CP15 accesses to Auxiliary ID Register (AIDR) or an operation to prefetch an instruction by MVA will generate an UNDEFINED exception on Cortex-R4.

**Conditions**

Either of the following instructions is executed in a privileged mode:

- - MRC p15,1,<Rt>,c0,c0,7 ; Read IMPLEMENTATION DEFINED Auxiliary ID Register
- - MCR p15,0,<Rt>,c7,c13,1 ; NOP, was Prefetch instruction by MVA in ARMv6

**Implications**

If software attempts to read the AIDR as part of its process of identifying the processor on which it is running, it will encounter an unexpected UNDEFINED exception. If software attempts to execute an instruction prefetch using the ARMv6 CP15 prefetch operation, it will encounter an unexpected UNDEFINED exception.

**Workaround(s)**

In the first case, because the contents of the AIDR are IMPLEMENTATION DEFINED, it must always be read in conjunction with the Main ID Register (MIDR). A simple way of avoiding this would be to read and analyze the MIDR first and, if this indicates that the processor is Cortex-R4 skip the read of the AIDR. In the second case, any instruction to prefetch an instruction by MVA should be removed or replaced by a true NOP instruction.

## CORTEX-R4#55 (ARM ID-722412) *CPACR.ASEDIS And CPACR.D32DIS Incorrect When Configured With Floating Point*

**Severity** Medium

**Expected Behavior**

**Issue** In implementations of the VFPv3-D16 architecture that do not also implement the Advanced SIMD functionality, the ASEDIS and D32DIS bits, (bits [31] and [30] respectively of the Coprocessor Access Control Register (CPACR)) are read-as-one/writes ignored (RAO/WI). This indicates that Advanced SIMD functionality and registers D16-D31 are disabled. Because of this erratum, these bits read zero in implementations of Cortex- R4F which include the floating-point unit. When the floating point unit is not included, all bits of the CPACR correctly read-as-zero (RAZ).

**Conditions** In an implementation of Cortex-R4F with the floating-point unit included, the CPACR is read.

**Implications** If software uses the CPACR to determine if Advanced SIMD functionality and registers D16-D31 are available, it will incorrectly determine that they are. Any attempt to use these features will lead to an unexpected UNDEFINED exception.

**Workaround(s)** Because the Cortex-R4F processor never includes Advanced SIMD functionality or registers D16-D31, this can be correctly determined by reading the part number from one of the ID registers rather than examining ASEDIS and D32DIS in the CPACR.

## CORTEX-R4#57 (ARM ID-737195) *Conditional VMRS APSR_Nzcv, FPSCR May Evaluate With Incorrect Flags*

**Severity**

Medium

**Expected Behavior**

A conditional VMRS APSR_nzcv, FPSCR instruction should evaluate its condition codes using the correct flags.

**Issue**

Under certain circumstances, a conditional VMRS APSR_nzcv, FPSCR instruction may evaluate its condition codes using the wrong flags and incorrectly execute or not execute.

**Conditions**

The erratum requires the following sequence of instructions in ARMstate: - VMRS<c> APSR_nzcv, FPSCR (formerly FMSTAT<c>), where the condition on the instruction is not always.

- A flag-setting integer multiply or multiply and accumulate instruction (e.g. MULS)
- A single-precision floating-point multiply-accumulate (FP-MAC) instruction (e.g. VMLA), timed such that the accumulate operation is inserted into the pipeline in the cycle in which the VMRS instruction is first attempted to be issued.

  To meet the above timing requirements, the VMRS instruction must be three pipeline stages behind the FPMAC. Depending on the rate in which the instructions are fetched, interlocks within this sequence and dual-issuing, this can be up to three other instructions between this pair, plus the multiply. Out-of-order completion of FP-MAC instructions must be enabled.

**Implications**

If this erratum occurs, the VMRS instruction will pass or fail its condition codes incorrectly, and this will appear in any trace produced by the ETM. This can corrupt the N, Z, C, V flag values in the CPSR which will typically affect the program flow.

**Workaround(s)**

This erratum can be avoided by disabling out-of-order single-precision floating point multiply-accumulate (SPMAC) instruction completion. Set DOOFMACS, bit [16] in the Secondary Auxiliary Control Register. This will have the side-effect of reducing the performance of SP-MAC operations, though the impact will depend on how these instructions are used in your code.

## CORTEX-R4#58 (ARM ID-726554) *DBGDSCR.Adadiscard Is Wrong When DBGDSCR.Dbgack Set*

**Severity** Medium

**Expected Behavior** When the DBGDSCR.ADAdiscard bit is set, asynchronous data aborts are discarded, except for setting the DBGDSCR.ADAbort sticky flag. The Cortex-R4 processor ensures that all possible outstanding asynchronous data aborts have been recognised before it enters debug halt state. The flag is immediately on entry to debug halt state to indicate that the debugger does not need to take any further action to determine whether all possible outstanding asynchronous aborts have been recognized.

**Issue** Because of this erratum, the Cortex-R4 processor also sets the DBGDSCR.ADAdiscard bit when the DBGDSCR.DBGack bit is set. This can cause the DBGDSCR.ADAbort bit to become set when the processor is not in debug halt state, and it is not cleared when the processor enters debug halt state. However, the processor does not discard the abort. It is pending or generates an exception as normal.

**Conditions**

- The processor is not in debug halt state
- The DBGDSCR.DBGack bit is set
- An asynchronous data abort (for example, SLVERR response to a store to Normal-type memory) is recognized

**NOTE:** it is not expected that DBGDSCR.DBGack will be set in any Cortex-R4 system

If this erratum occurs, and the processor subsequently enters debug halt state, the DBGDSCR.ADAbort bit will be set, when in fact no asynchronous data abort has occurred in debug state. Before exiting debug state, the debugger will check this bit and will typically treat it as an error. If no other asynchronous data abort has occurred in debug state, this is a false error.

**Implications**

**Workaround(s)** None

## CORTEX-R4#61 (ARM ID-720270) *Latched DTR-Full Flags Not Updated Correctly On DTR Access*

**Severity** Medium

**Expected Behavior** When the debug Data Transfer Register (DTR) is in non-blocking mode, the latched DTR-full flags (RXfull_l and TXfull_l) record the state of the DTR registers as observed by the debugger and control the flow of data to and from the debugger to prevent race hazards. For example, when the target reads data from DBGDTRRXint, the associated flag RXfull is cleared to indicate that the register has been drained, but the latched value Rxfull_l remains set. Subsequent debugger writes to DBGDTRRXext are ignored because RXfull_l is set. RXfull_l is updated from RXfull when the debugger reads DBGDSCRext such that a debugger write to DBGDTRRXext will only succeed after the debugger has observed that the register is empty. The ARMv7 debug architecture requires that RXfull_l be updated when the debugger reads DBGDSCRext and when it writes DBGDTRRXext. Similarly, TXfull_l must be updated when the debugger reads DBGDSCRext and when it reads DBGDTRTXext.

**Issue** Because of this erratum, RXfull_l and TXfull_l are only updated when the debugger reads DBGDSCRext.

**Conditions** The DTR is in non-blocking mode, that is, DBGDSCR.ExtDCCmode is set to 0b00 and EITHER:

- The debugger reads DBGDSCRext which shows that RXfull is zero, that is, DBGDTRRX is empty.
- The debugger writes data to DBGDTRRXext.
- Without first reading the DBGDSCRext, and before the processor has read from DBGDTRRXint, the debugger performs another write to DBGDTRRXext.

OR

- The debugger reads DBGDSCRext which shows that TXfull is one, that is, DBGDTRTX is full.
- The debugger reads data from DBGDTRTXext,
- The processor writes new data into DBGDTRTXint,
- Without first reading the DBGDSCRext, the debugger performs another read from DBGDTRTXext.

**Implications** The ARMv7 debug architecture requires the debugger to read the DBGDSCRext before attempting to transfer data via the DTR when in non-blocking mode. This erratum only has implications for debuggers that violate this requirement. If the erratum occurs via data transfer, data loss may occur. The architecture requires that data transfer never occur.

Texas Instruments has verified that TI's Code Composer Studios IDE is not affected by this issue.

**Workaround(s)** None

## CORTEX-R4#66 (ARM ID-754269) *Register Corruption During A Load-Multiple Instruction At An Exception Vector*

**Severity** Medium

**Expected Behavior**

**Issue** Under certain circumstances, a load multiple instruction can cause corruption of a general purpose register.

**Conditions** All the following conditions are required for this erratum to occur:

- A UDIV or SDIV instruction is executed with out-of-order completion of divides enabled
- A multi-cycle instruction is partially executed before being interrupted by either an IRQ, FIQ or imprecise abort. In this case, a multi-cycle instruction can be any of the following:
  - LDM/STM that transfers 3 or more registers
  - LDM/STM that transfers 2 registers to an unaligned address without write back
  - LDM/STM that transfers 2 registers to an aligned address with write back
  - TBB/TBH
- A load multiple instruction is executed as the first instruction of the exception handler
- The load multiple instruction itself is interrupted either by an IRQ, FIQ, imprecise abort or external debug halt request. This erratum is very timing sensitive and requires the UDIV or SDIV to complete when the load multiple is in the Issue stage of the CPU pipeline. The register that is corrupted is not necessarily related to the load-multiple instruction and will depend on the state in the CPU store pipeline when the UDIV or SDIV completes.

**Implications** For practical systems, it is not expected that an interruptible LDM will be executed as the first instruction of an exception handler, because the handler is usually required to save the registers of the interrupted context. Therefore, it is not expected that this erratum has any implications for practical systems. If the situation of the erratum occurs it will result in the corruption of the register bank state and could cause a fatal failure if the corrupted register is subsequently read before being written.

**Workaround(s)** To work around this erratum, set bit [7] of the Auxiliary Control Register to disable out-of-order completion for divide instructions. Code performance may be reduced depending on how often divide operations are used.

## CORTEX-R4#67 (ARM ID-758269) *Watchpoint On A Load Or Store Multiple May Be Missed.*

**Severity** Medium

**Expected Behavior** The Cortex-R4 supports synchronous watchpoints. This implies that for load and store multiples, a watchpoint on any memory access will generate a debug event on the instruction itself.

**Issue** Due to this erratum, certain watchpoint hits on multiples will not generate a debug event.

**Conditions** All the following conditions are required for this erratum to occur:

- A load or store multiple instruction is executed with at least 5 registers in the register list
- The address range accessed corresponds to Strongly-Ordered or Device memory
- A watchpoint match is generated for an access that does not correspond to either the first two or the last two registers in the list.

Under these conditions the processor will lose the watchpoint. Note that for a "store multiple" instruction, the conditions are also affected by pipeline state making them timing sensitive.

**Implications** Due to this erratum, a debugger may not be able to correctly watch accesses made to Device or Strongly-ordered memory. The ARM architecture recommends that watchpoints should not be set on individual Device or Strongly-ordered addresses that can be accessed as part of a load or store multiple. Instead, it recommends the use of the address range masking functionality provided to set watchpoints on an entire region, ensuring that the watchpoint event will be seen on the first access of a load or store multiple to this region. If this recommendation is followed, this erratum will not occur.

**Workaround(s)** None

## DEVICE#B053 ***CPU code execution could be halted on a device warm reset if the core power domain # 2 is disabled by software.***

**Severity** Medium

**Expected Behavior** The CPU code execution must start from the reset vector (address 0x00000000) upon a device warm reset and is not affected by the state of any switchable device power domain.

**Issue** CPU code execution could be halted upon a warm reset if the core power domain # 2 has been disabled by software prior to the device warm reset.

**Conditions** The behavior is not dependent on any particular operating condition.

**Implications** CPU code execution is halted so that a system hang occurs. An external monitor must be present to prevent the system from entering an unsafe state when this happens.

**Workaround(s)** The application must not disable the core power domain # 2 in software via the Power Management Module (PMM) registers, even if the modules inside this core power domain are not used in the application.

## DEVICE#B056 *Error flags are incorrectly set in the Error Signaling Module upon a device warm reset for modules in core power domains that are disabled*

**Severity** High

**Expected Behavior** Once a core power domain is disabled (turned off) no error must be indicated to the ESM from a module in this power domain.

**Issue** When a core power domain is disabled, a device warm reset causes error signals to be sent to the ESM that appear to be from the power domain that has been disabled. This is caused due to the incorrect power domain isolation implemented on these error signals. All such error signals are connected to ESM group1 channels.

**Conditions** The behavior is not dependent on any particular operating condition.

**Implications** If the application has enabled the generation of interrupts or the assertion of the nERROR output when any flag gets set in the ESM module's group1 status registers, then the CPU will receive an interrupt and/or the nERROR pin will be driven for an error condition ascribed to a module that is in a core power domain that has been disabled.

**Workaround(s)** When a core power domain is disabled, the application must ensure that any error signal related to a module within this power domains is not configured to generate an interrupt to the CPU or assert the device nERROR output. Alternately, the application could clear the ESM group1 status flags upon each device warm reset condition before it enables interrupts and nERROR drive features for the ESM group1 channels.

## DEVICE#B058 *Device I/Os Could Toggle After Warm Reset*

**Severity** Medium

**Expected Behavior** All device I/Os are tri-stated (inputs) after a device warm reset. No toggle is expected on any of the device I/Os without software configuration.

**Issue** All core power domains get enabled on a system reset. After the core domains are turned ON, the domain outputs are unstable for a few cycles after the system reset is released. This could cause the I/Os to toggle.

**Conditions** The behavior is not dependent on any particular operating condition.

**Implications** Any external circuit connected to the device I/Os could see activity on the connections to the device after a devcie warm reset.

**Workaround(s)** There are no software or hardware workarounds for this issue.

## DEVICE#142 *CPU Abort Not Generated on Write to Unimplemented MCRC Space*

| | |
|---|---|
| **Severity** | Low |
| **Expected Behavior** | A write to the illegal address region of the MCRC module will generate an abort |
| **Issue** | A CPU Abort does not get generated per the following condition: |
| **Conditions** | When a normal mode write to an illegal address region of MCRC register space is followed by a debug mode access. |
| **Implications** | When debugging, either a breakpoint on the instruction after the illegal write, or single stepping through the illegal write will not generate an abort. |
| **Workaround(s)** | None |

## DEVICE#070 *Some multiplexing controls are not compatible with the emulation device*

**Severity** Medium

**Expected Behavior** For the common functions, it is expected that the derivative parts will maintain compatibility to the superset emulation part.

**Issue** There are multiple issues with the multiplexing controls:

1. For GIOB[2], to take input from the alternate path,the PINMMR29[16] has to cleared. This is incompatible with the emulation device, where the PINMMR29[16] needs to be set to select the alternate path as input for GIOB[2].
2. PINMMR29[24] is cleared by default to select MII interface for the EMAC. This is incompatible with the emulation device, where the PINMMR29[24] was set by default and selects the RMII interface.

**Conditions** The behavior is not dependent on any particular operating condition.

**Implications** The application must manage the configuration of the PINMMR registers based on the actual part being used. The code running on the emulation device cannot be run as-is on this device.

**Workaround(s)** There are no software workarounds.

## DMA#27 *DMA Requests Lost During Suspend Mode*

**Severity** Medium

**Expected Behavior**

**Issue** DMA requests from a peripheral can be lost.

**Condition** This only happens when:

- The device is suspended by a debugger
- A peripheral continues to generate requests while the device is suspended
- The DMA is setup to continue the current block transfer during suspend mode with DMA DEBUG MODE set to ‘01’
- And the request trigger type TTYPE is set to frame trigger

**Implication(s)** When the DMA comes out of the suspend mode to resume the transfer, the data transfers corresponding to the third and subsequent requests will be lost.

**Workaround(s)** Either use TTYPE = Block transfer when DMA DEBUG MODE is '01' (Finish Current Block Transfer) or use DMA DEBUG MODE = '00' (Ignore suspend) when using TTYPE = Frame transfer to complete block transfer even after suspend/halt is asserted.

## ERAY#52 (FLEXRAY#52) *WUS Generates Redundant SIR.WUPA/B Events*

| | |
|---|---|
| **Severity** | Low |
| **Expected Behavior** | If a sequence of wakeup symbols (WUS) is received and all are separated by appropriate idle phases then a valid wakeup pattern (WUP) should be detected after *every second WUS.* |
| **Issue** | Instead, the FlexRay module detects a valid wakeup pattern after the second WUS and then after *each following WUS*. |
| **Condition** | A sequence of wakeup symbols (WUS) is received. All separated by appropriate idle phases. |
| **Implication(s)** | More SIR.WUPA/B events are seen than expected especially when an application program frequently resets the appropriate SIR.WUPA/B bits |
| **Workaround(s)** | Ignore redundant SIR.WUPA/B events. |

## ERAY#58 (FLEXRAY#58) *Erroneous Cycle Offset During Startup after abort of startup or normal operation*

**Severity**

Low

**Expected Behavior**

Correct cycle offset inspite of abort of startup or normal operation by a READY command.

**Issue**

The state INITIALIZE_SCHEDULE may be one macrotick too short during an integration attempt. This leads to an early cycle start in state INTEGRATION_COLDSTART_CHECK or INTEGRATION_CONSISTENCY_CHECK.

**Condition**

An abort of startup or normal operation by a READY command near the macrotick border. The erratum is limited to applications where READY command is used to leave STARTUP, NORMAL_ACTIVE, or NORMAL_PASSIVE state

**Implication(s)**

As a result the integrating node calculates a cycle offset of one macrotick at the end of the first even/odd cycle pair in the states INTEGRATION_COLDSTART_CHECK or INTEGRATION_CONSISTENCY_CHECK and tries to correct this offset.

If the node is able to correct the offset of one macrotick (pOffsetCorrectionOut >> gdMacrotick), the node enters NORMAL_ACTIVE with the first startup attempt.

If the node is not able to correct the offset error because pOffsetCorrectionOut is too small (pOffsetCorrectionOut <= gdMacrotick), the node enters ABORT_STARTUP and is ready to try startup again. The next (second) startup attempt is not affected by this erratum.

**Workaround(s)**

With a configuration ofpOffsetCorrectionOut >> gdMacrotick*(1+cClockDeviationMax) the node will be able to correct the offset and therefore also be able to successfully integrate.

## ERAY#59 (FLEXRAY#59) *First WUS Following Received Valid WUP May Be Ignored*

| | |
|---|---|
| **Severity** | Low |
| **Expected Behavior** | The FlexRay controller protocol engine should recognize all wakeup symbols (WUS). |
| **Issue** | However, the protocol engine may ignore the first wakeup symbol (WUS) following the below stated state transition therefore signalling the next SIR.WUPA/B at the third WUS instead of the second WUS. |
| **Condition** | The erratum is limited to the reception of redundant wakeup patterns. When the protocol engine is in state WAKEUP_LISTEN and receives a valid wakeup pattern (WUP), it transfers into state READY and updates the wakeup status vector CCSV.WSV[2:0] as well as the status interrupt flags SIR.WST and SIR.WUPA/B. |
| **Implication(s)** | Delayed setting of status interrupt flags SIR.WUPA/B for redundant wakeup patterns. |
| **Workaround(s)** | None |

## ERAY#60 (FLEXRAY#60) *READY Command Accepted In READY State*

| | |
|---|---|
| **Severity** | Low |
| **Expected Behavior** | The FlexRay module should ignore a READY command while in READY state. |
| **Issue** | However, the FlexRay module does not ignore a READY command while in READY state. |
| **Conditions** | The erratum is limited to the READY state. |
| **Implications** | Flag CCSV.CSI is set. Cold starting needs to be enabled by the Protocol Operation Controller (POC) command ALLOW_COLDSTART (SUCC1.CMD = "1001"). |
| **Workaround(s)** | None |

## ERAY#61 (FLEXRAY#61) *Slot Status vPOC!SlotMode Is Reset Immediately When Entering HALT State*

| | |
|---|---|
| **Severity** | Low |
| **Expected Behavior** | According to the FlexRay protocol specification, the slot mode should not be reset to SINGLE slot mode before the following state transition from HALT to DEFAULT_CONFIG state. |
| **Issue** | However, when the protocol engine is in NORMAL_ACTIVE or NORMAL_PASSIVE state, a HALT or FREEZE command issued by the CPU resets vPOC!SlotMode immediately to SINGLE slot mode (CCSV.SLM[1:0] = "00"). |
| **Conditions** | The erratum is limited to the HALT state. |
| **Implications** | The slot status vPOC!SlotMode is reset to SINGLE when entering HALT state. |
| **Workaround(s)** | None |

## FTU#8 *FlexRay Transfer Unit Not Disabled On Memory Protection Violation (MPV) Error*

**Severity** Medium

**Expected Behavior** On memory protection violation (MPV) errors the FTU should get disabled.

**Issue** The FTU does not get disabled under some conditions.

**Conditions** If an MPV error occurs during the following transfer scenarios:

- During header transfer from system memory to FlexRay RAM, when FTU is configured to transfer header and payload
- During payload transfer from FlexRay RAM to system memory, when FTU is configured to transfer header and payload
- During a transfer from FlexRay RAM to system memory, when FTU is configured to transfer payload only

**Implications** The MPV error flag in the Transfer Error Interrupt Flag (TIEF) register is set, but the TUE flag in the Global Control Set/Reset (GCS/R) register does not get cleared. As a result, the FTU does not get disabled.

**Workaround(s)** This errata can be avoided in the following ways:

- For transfers from system memory to FlexRay RAM, transfer the payload only
- Generate an MPV interrupt and clear the TUE flag in the Global Control Set/Reset (GCS/R) register in the interrupt service routine

## FTU#19 *TCCOx Flag Clearing Masked*

**Severity** Low

**Expected Behavior** According to the documentation, when TOOFF (Transfer Occurred Offset) is read, the corresponding message flag in TCCOx must be cleared.

**Issue** In some conditions, the read of TOOFF register would not be up to date and would not reflect the last buffer completed.

**Conditions** There may be a timing condition when TCCOx flag clearing could be masked due to the state machine clearing of TTCCx (Trigger transfer to communication controller) within the same cycle as software reading TOOFF.

**Implications** The TCCOx flag is not being cleared.

**Workaround(s)** After reading the TOOFF to determine the highest buffer completed, clear the corresponding flag in TCCOx.

## FTU#23 *Bus errors on transactions not initiated by FTU could get captured by FTU*

| | |
|---|---|
| **Severity** | Medium |
| **Expected Behavior** | FTU captures bus errors only on transactions initiated by the FTU. |
| **Issue** | In case a bus error is caused, the FTU captures the error independent of whether it or another bus master (on the same switched central resource (SCR)) actually initiated the read/write. Bus errors caused by bus masters on other SCRs do not cause this issue. |
| **Conditions** | This errata applies only in the case a bus error is caused by any bus master on the same SCR as the FTU. |
| **Implications** | FTU captures bus errors on transactions initiated by any bus master on the same SCR as the FTU. |
| **Workaround(s)** | - |

## ANALOGIP_021_LPO#16 (F021 LPO#16) *Oscillator Fault Detection Starts too Soon*

**Severity** Low

**Expected Behavior** The oscillator fault detection circuit allows approximately 200ms after the release of nPORRST for the oscillator to become valid before monitoring for oscillator faults.

**Issue** Sometimes the oscillator fault detection starts monitoring the oscillator shortly (approximately 100us) after the release of nPORRST.

**Condition** This occurs only after power on.

**Implication(s)** If the oscillator is slow to start, an oscillator fault condition may be detected and the device will switch to using the HFLPO as a clock source.

**Workaround(s)** This condition is avoided if nPORRST is held low long enough for the oscillator to stablize.

This condition can be corrected by software. The software should check for oscillator fault after detecting a power-on reset. If an oscillator fault has occured, the software can attempt to restart the oscillator. Refer to the device Technical Reference Manual for information on how to recover from an oscillator failure.

## MCRC#18 *CPU Abort Generated on Write to Implemented MCRC Space After Write to Unimplemented MCRC Space*

| | |
|---|---|
| **Severity** | Low |
| **Expected Behavior** | A write to the legal address region of the MCRC module should not generate an abort |
| **Issue** | An Unexpected CPU Data Abort is generated in the following condition: |
| **Conditions** | When a normal mode write to an illegal address region of MCRC register space is followed by a write to a legal address region of the MCRC register space. |
| **Implications** | A write to an illegal address region of the MCRC register space generates a data abort as expected. The next write to a legal address region of the MCRC register space generates an unexpected second data abort. |
| **Workaround(s)** | None |

## MIBSPI#110 *Multibuffer Slave In 3 or 4 Pin Mode Transmits Data Incorrectly for Slow SPICLK Frequencies and for Clock Phase = 1*

**Severity**

Medium

**Expected Behavior**

The SPI must be able to transmit and receive data correctly in slave mode as long as the SPICLK is slower than the maximum frequency specified in the device datasheet.

**Issue**

The MibSPI module, when configured in multi-buffered slave mode with 3 functional pins (CLK, SIMO, SOMI) or 4 functional pins (CLK, SIMO, SOMI, nENA), could cause incorrect data to be transmitted.

**Conditions**

This issue can occur under the following condition:

- Module is configured to be in mult-buffered mode, AND
- Module is configured to be a slave in the SPI communication, AND
- SPI communication is configured to be in 3-pin mode or 4-pin mode with nENA, AND
- Clock phase for SPICLK is 1, AND
- SPICLK frequency is VCLK frequency / 12 or slower

**Implications**

Under the above described condition, the slave MibSPI module can transmit incorrect data.

**Workaround(s)**

The issue can be avoided by setting the CSHOLD bit in the control field of the TX RAM. The nCS is not used as a functional signal in this communication, hence setting the CSHOLD bit does not cause any other effect on the SPI communication.

## MIBSPI#111 *Data Length Error Is Generated Repeatedly In Slave Mode when I/O Loopback is Enabled*

**Severity** Medium

**Expected Behavior** After a data length (DLEN) error is generated and the interrupt is serviced the SPI should abort the ongoing transfer and stop.

**Issue** When a DLEN error is created in Slave mode of the SPI using nSCS pins in IO LoopBack Test mode, the SPI module re-transmits the data with the DLEN error instead of aborting the ongoing transfer and stopping.

**Conditions** This is only an issue for a IOLPBK mode Slave in Analog Loopback configuration, when the intentional error generation feature is triggered using CTRL_DLENERR(IOLPBKTSTCR.16).

**Implications** The SPI will repeatedly transmit the data with the DLEN error when configured in the above configuration.

**Workaround(s)** After the DLEN_ERR interrupt is detected in IOLPBK mode, disable the transfers by clearing the SPIEN bit of SPIGCR1 register (bit 24) and then re-enable the transfers by setting SPIEN.

## NHET#52 *WCAP May not Capture the High-Resolution Offset Correctly*

**Severity** High

**Expected Behavior** The N2HET is a complex timer that operates by executing a set of timing instructions each loop resolution period. Normally counters and capture registers implemented by such a scheme would be limited in resolution to the timer's loop period; but the N2HET also includes dedicated hardware timer extensions that enable operation with resolution as high as 1/128 of the timer loop period. The WCAP instruction is designed to support time-stamping with high resolution. When an event occurs, the WCAP instruction is supposed to capture both the loop-resolution counter value and the high-resolution offset measuring the interval between the start of the loop and actual event occurrence, and return the combined result. This result can be interpreted as the number of timer loops represented as fixed point number with 25 integral bits and 7 fractional bits.

**Issue** WCAP may not capture the high-resolution offset correctly.

**Conditions** If the edge event that the WCAP instruction is time-stamping happens to align with the start of the internal loop resolution period, then the WCAP does not properly capture the 7-bit fractional offset correctly

**Implications** Under the above described condition, the fractional offset of the previous event captured by WCAP is what is stored in the WCAP data field. Because of this boundary condition, the WCAP instruction should not be used in high-resolution mode

**Workaround(s)** A similar instruction, PCNT, is known to operate correctly at high resolution. While WCAP returns the absolute time-stamp of an event, PCNT returns the interval between an event and the previous event on the same pin.

Instead of WCAP, if an absolute time-stamp is required it is recommended to use a PCNT instruction followed by an ADD instruction that accumulates the time between successive edge measurements to produce a result that is a WCAP equivalent.

An example for using PCNT as a work around for WCAP:

WCAP {next = ERRPIN, cond_addr = JMP, hr_lr=high, reg=T, event=RISE, pin=16, data=0}

work around:

SAVE: MOV32 {remote=TRISE, type=REMTOREG, reg=R}

TRISE: PCNT {next=TIMESTAMP, hr_lr=high, type=RISE2RISE, pin=16, data=0}

TIMESTAMP: ADD {src1=REM, src2=R, rdest=REM, remote=TRISE, dest=NONE, next=CHK, data=0, hr_data=0}

CHK: BR {cond_addr=JMP, event=RISE, pin=20,next=ERRPIN}

## NHET#53 *Enhanced Input Capture Pins May not Capture Small Pulses Correctly*

| | |
|---|---|
| **Severity** | High |
| **Expected Behavior** | The PCNT and WCAP instructions can capture pulse length or time stamp of small pulses that have two edges within a single loop resolution. |
| **Issue** | The high resolution value may be captured incorrectly. |
| **Conditions** | If the second edge event that the PCNT or WCAP instruction is using happens to align with the start of the internal loop resolution period, then the instruction does not properly capture the high resolution value correctly. |
| **Implications** | Because of this boundary condition, the PCNT and WCAP instructions should not be used on small pulses. |
| **Workaround(s)** | None |

## SYS#46 *Clock Source Switching Not Qualified With Clock Source Enable And Clock Source Valid*

**Severity** Low

**Expected Behavior** An attempt to switch to a clock source which is not valid yet should be discarded.

**Issue** Switching a clock source by simply writing to the GHVSRC bits of the GHVSRC register may cause unexpected behavior. The clock will switch to a source even if the clock source was not ready.

**Conditions** The clock source is enabled (CSDIS register) but the clock source is not yet valid (CSVSTAT register).

**Implications** Unexpected behavior stated above.

**Workaround(s)** Always check the CSDIS register to make sure the clock source is turned on and check the CSVSTAT register to make sure the clock source is valid. Then write to GHVSRC to switch the clock.

## SYS#102 *Bit field EFUSE_Abort[4:0] in SYSTASR register is read-clear*

| | |
|---|---|
| **Severity** | Medium |
| **Expected Behavior** | The documentation states that EFUSE_Abort[4:0] of the SYSTASR register should be write-clear in privilege mode. |
| **Issue** | However, these bits are implemented as read-clear. |
| **Conditions** | Always. |
| **Implications** | Software implementation for error handling needs to take care of this. |
| **Workaround(s)** | None |

## SYS#111 *The Device may Generate Multiple nRST Pulses.*

**Severity** Medium

**Expected Behavior** When an event on the device causes a system reset to be asserted, the device drives the system reset (nRST) terminal low for 8 VCLK periods. Then the nRST terminal is released and gets pulled High, allowing the device to start its warm boot sequence.

**Issue** It is possible that the device may generate multiple system resets instead of releasing the system reset (nRST) at the appropriate time and starting the warm boot sequence.

**Conditions** This issue occurs when either the external reset pulse or the 8 VCLK internally generated reset pulse is the same length as the delay caused by the glitch filter on the nRST signal. The delay time of the glitch filter is a function of device process, the temperature and the core voltage. This has a range from 500ns to 2µs. The length of the 8 VCLK reset extension is a function of the crystal speed, the I/O voltage, and the resistance and capacitance on the nRST pin. There is no problem when reset is properly asserted with the PORRST pin for 1ms or more as the nRST pin will be held low much longer than 2us.

**Implications** Usually only one or two extra reset pulses are generated. However, it is possible, in rare cases, that a long string of reset pulses could be generated. This would prevent the device from booting up in a timely manner.

**Workaround(s)** In the case of an externally generated system reset, this issue can be avoided by asserting the system reset signal for more than 2µs. If the issue is seen when there is an internal system reset condition then the issue can be avoided by using an 8MHz or slower crystal. Another workaround is to avoid the use of internal system resets such as software reset, oscillator fault reset, watchdog reset, or debug reset. This erratum has been fixed on newer devices by extending the time of the internally generated nRSTpulse to 32 VCLK cycles. This extends the reset pulse to a minimum time of 3.2us (with a 20MHz crystal), which exceeds the maximum delay time of the glitch filter.

## VIM#27 *Unexpected phantom interrupt*

**Severity** High

**Expected Behavior** When responding to an interrupt and a subsequent interrupt is received, the corresponding VIM request should be flagged as pending in the VIM status registers. When the CPU is ready to service the subsequent interrupt, the correct service routine address should be fetched by the CPU.

**Issue** On rare occasions the VIM may return the phantom interrupt vector instead of the real interrupt vector.

**Condition** This condition is specific to software and hardware vectored modes. This is not applicable for legacy interrupt servicing mode. The ratio of GCLK to VCLK must be 3:1 or greater for hardware vectored mode. The ratio of GCLK to VCLK must be 5:1 or greater for software vectored mode. A subsequent interrupt request must occur when the VIM is finishing acknowledging a previous interrupt.

**Implication(s)** The subsequent interrupt request vectors to the phantom interrupt routine instead of the correct service routine.

**Workaround(s)** This issue can be avoided if the clock ratio GCLK:VCLK is 1:1 or 2:1. Otherwise the VIM status register can be checked within the phantom interrupt routine to determine the source of the subsequent interrupt.

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license 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 significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components which meet ISO/TS16949 requirements, mainly for automotive use. Components which have not been so designated are neither designed nor intended for automotive use; and TI will not be responsible for any failure of such components to meet such requirements.

| **Products** | | **Applications** | |
|---|---|---|---|
| Audio | www.ti.com/audio | Automotive and Transportation | www.ti.com/automotive |
| Amplifiers | amplifier.ti.com | Communications and Telecom | www.ti.com/communications |
| Data Converters | dataconverter.ti.com | Computers and Peripherals | www.ti.com/computers |
| DLP® Products | www.dlp.com | Consumer Electronics | www.ti.com/consumer-apps |
| DSP | dsp.ti.com | Energy and Lighting | www.ti.com/energy |
| Clocks and Timers | www.ti.com/clocks | Industrial | www.ti.com/industrial |
| Interface | interface.ti.com | Medical | www.ti.com/medical |
| Logic | logic.ti.com | Security | www.ti.com/security |
| Power Mgmt | power.ti.com | Space, Avionics and Defense | www.ti.com/space-avionics-defense |
| Microcontrollers | microcontroller.ti.com | Video and Imaging | www.ti.com/video |
| RFID | www.ti-rfid.com | | |
| OMAP Applications Processors | www.ti.com/omap | **TI E2E Community** | e2e.ti.com |
| Wireless Connectivity | www.ti.com/wirelessconnectivity | | |

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