TMS570LC4357 Microcontroller

Silicon Revision A

Silicon Errata

Literature Number: SPNZ180D
June 2013–Revised May 2016

Contents

1 Device Nomenclature 4
2 Revision Identification 5
3 Known Design Exceptions to Functional Specifications 6
4 Revision History 72

List of Figures

1 Device Revision Code Identification 5
2 Shared Input Channel in "Open" State 9
3 Example ADC1/ADC2 Channel Connection 10
4 First Fail Mode 47
5 Second Fail Mode 48
6 Workarounds 48

List of Tables

1 Device Revision Codes 5
2 Silicon Revision A Known Design Exceptions to Functional Specifications 6
3 Revision History 72

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

1 Device Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all devices. Each commercial family member has one of three prefixes: TMX, TMP, or TMS (for example, **TMS**570LS3137). These prefixes represent evolutionary stages of product development from engineering prototypes (TMX) through fully qualified production devices/tools (TMS).

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.

TMX and TMP devices are shipped against the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

TMS devices 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.

2 Revision Identification

Figure 1 provides examples of the TMS570LCx device markings. The device revision can be determined by the symbols marked on the top of the device.

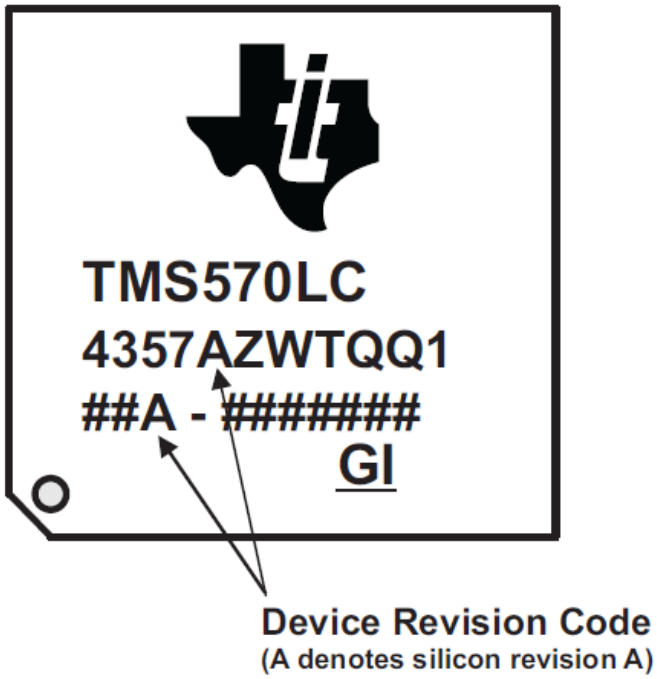

Figure 1. Device Revision Code Identification

Silicon revision is identified by a device revision code. The code is of the format TMS570LC4357x, 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
TMS570LC4357	A	TMS570LC4357

3 Known Design Exceptions to Functional Specifications

Table 2 lists the known exceptions to the functional specifications for silicon revision A of this device.

Table 2. Silicon Revision A Known Design Exceptions to Functional Specifications

Title	Page
ADC#1 — Injecting current into an input channel shared between the two ADCs causes a DC offset in conversion results of other channels	9
AHB_ACCES_PORT#3 (ARM ID-529470) — Debugger may display unpredictable data in the memory browser window if a system reset occurs	12
CCMR5#1 — Self Test error is not generated if a mismatch is detected during the last two test vectors of the self-test for the VIM Compare block	13
CCMR5#2 — Self-test for VIM compare block does not terminate immediately after it detects a self-test failure	14
CORTEX-R5#1 (ARM ID-772721) — In Standby Mode, peripheral port may lose read transactions	15
CORTEX-R5#3 (ARM ID-756523) — Watchpointed access in a store-multiple is not masked	16
CORTEX-R5#5 (ARM ID-771872) — In Standby Mode, with n:1 clocking, AXI-S transactions are not accepted	17
CORTEX-R5#7 (ARM ID-780125) — Processor might deadlock or lose data when configured with cache-ECC	18
DCC#24 — Single Shot Mode Count may be Incorrect	20
DEVICE#31 — RAM ECC memory space is inaccessible through DAP from the debugger	21
DEVICE#32 — DMA cannot continue to access SRAM after the device comes out of global low power mode	22
DEVICE#37 — AJSM Visible Unlock Code has ECC Value not Equal to 0xFFFF	23
DEVICE#39 — Extra WE/OE pulses in asynchronous EMIF access	24
DEVICE#40 — Abort is not generated when writing to unimplemented locations in some peripheral frames	25
DEVICE#41 — Power-on Reset During Bank 7 Sector Erase May Corrupt Other Sectors	26
DEVICE#42 — Reads of Bank7 may be Corrupted by Short Glitches on nPORRST Pin	27
DEVICE#45 — ePWM tripzone mode "high impedance state" does not work	28
DEVICE#46 — No ESM notification when CPU livelock occurs due to hard error in cache memory	29
DEVICE#47 — STC1 (CPU) test cannot be run on segment 1 independently.	30
DEVICE#48 — Interconnect Global Error flag is set after STC1 (CPU) self test completion	31
DEVICE#49 — False interconnect safety checker error flag	32
DEVICE#50 — STC2 (nHET) selftest must run with STCCLKDIV greater than zero	33
DEVICE#51 — Values in the Interconnect status registers are left-shifted left 24 bits	34
DEVICE#54 — Writes and reads to EMAC CPPI memory are byte swapped on big endian device (TMS570)	35
DEVICE#55 — False VIM Compare Error in hardware vector mode	36
DEVICE#56 — nERROR assertion on debugger connect	37
DEVICE#57 — Cannot Set Breakpoints While nRST is Asserted	38
DEVICE#59 — CPU Interconnect Self-test does not set status flags	39
DMAOCP#01 — DMA acts as if it receives an extra DMA request from a peripheral	40
ERAY#52 (FLEXRAY#52) — Wakeup Symbol (WUS) Generates Redundant Wakeup Interrupts (SIR.WUPA/B)	41
ERAY#58 (FLEXRAY#58) — Erroneous Cycle Offset During Startup after abort of startup or normal operation	42
ERAY#59 (FLEXRAY#59) — First Wakeup Symbol (WUS) Following Received Valid Wakeup Pattern (WUP) May Be Ignored	43
ERAY#60 (FLEXRAY#60) — READY Command Accepted In READY State	44
ERAY#61 (FLEXRAY#61) — The Transmission Slot Mode Bit Is Reset Immediately When Entering HALT State	45
ERAY#64 (FLEXRAY#64) — SBESTAT register is not readable	46
ERAY#68 (FLEXRAY#68) — Data transfer overrun for message transfers Message RAM to Output Buffer (OBF) or from Input Buffer (IBF) to Message RAM	47
ERAY#69 (FLEXRAY#69) — Missing startup frame in cycle 0 at coldstart after FREEZE or READY command	49
FTU#08 — FlexRay Transfer Unit Not Disabled On Memory Protection Violation (MPV) Error	50
FTU#19 — TCCOx Flag Clearing Masked	51
GCM#58 — N2HET and HTU require VCLK to be faster than HCLK/4	52
GCM#59 — Oscillator can be disabled while PLL is running	53
LPO#16 — Oscillator Fault Detection Starts too Soon after power-on reset is released	54

Table 2. Silicon Revision A Known Design Exceptions to Functional Specifications (continued)

MIBSPI#110 — Multibuffered SPI in Slave Mode In 3- or 4-Pin Communication Transmits Data Incorrectly for Slow SPICLK Frequencies and for Clock Phase = 1 55
MIBSPI#111 — Data Length Error Is Generated Repeatedly In Slave Mode when I/O Loopback is Enabled 56
MIBSPI#136 — Transfer Complete Interrupt is not generated after completing all the transfers when a Transfer Group is set to end at buffer 128 57
MIBSPI#137 — Spurious RX DMA REQ from a Slave mode MIBSPI 58
MIBSPI#138 — MIBSPI RAM ECC is not read correctly in DIAG mode 59
MIBSPI#139 — Mibspi RX RAM RXEMPTY bit does not get cleared after reading 60
NHET#53 — Enhanced Input Capture Pins May not Capture Small Pulses Correctly 61
NHET#55 —More than one PCNT instruction on the same pin results in measurement error 62
PBIST#4 — PBIST ROM Algorithm Doesn't Execute 64
SSWF021#44 — Change to PLL Lock Time 65
SSWF021#45 — PLL Fails to Start 66
STC#26 — The value programmed into the Self Test Controller (STC) Self-Test Run Timeout Counter Preload Register (STCTPR) is restored to its reset value at the end of each self test run. 67
STC#28 — CPU can hang if a system reset [internal or external] is asserted while the CPU self test is in progress 68
STC#29 — Inadvertent Performance Monitoring Unit (PMU) interrupt request generated if a system reset [internal or external] occurs while a CPU Self-Test is executing. 69
STC#30 — Self Test Controller Fails to Execute Properly 70
VIM#28 — VIM module does not return corrected data to the CPU when a single bit error is detected in the VIM RAM 71

ADC#1

Injecting current into an input channel shared between the two ADCs causes a DC offset in conversion results of other channels

Severity

3 - Medium

Expected Behavior

External circuit connected to one channel must not affect the conversion result of another channel.

Issue

This microcontroller (MCU) has two Analog-to-Digital Converters (ADCs). Some of the input channels are unique to ADC1 while some are shared between ADC1 and ADC2. Figure 2 shows a block diagram of an input channel shared between ADC1 and ADC2.

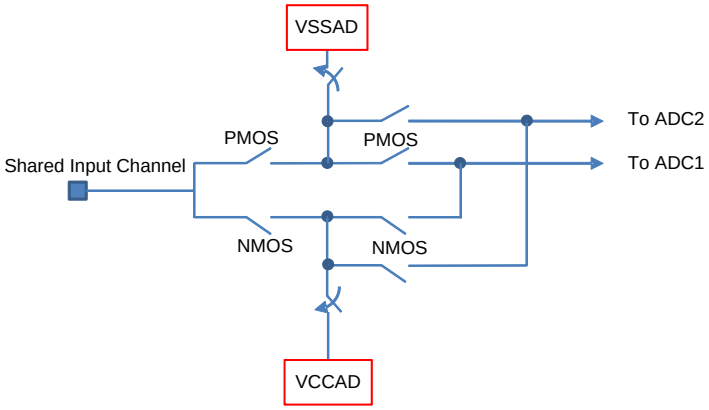

Figure 2. Shared Input Channel in "Open" State

The PMOS and NMOS switches are open indicating that this shared input channel is not currently being sampled either by ADC1 or by ADC2. Also, there are switches to VCCAD and VSSAD that are closed. If any current is injected into this analog input, any leakage through the open PMOS switch will be shunted to VSSAD. These switches to VSSAD and VCCAD are opened as soon as this shared input channel is being sampled by either ADC1 or ADC2.

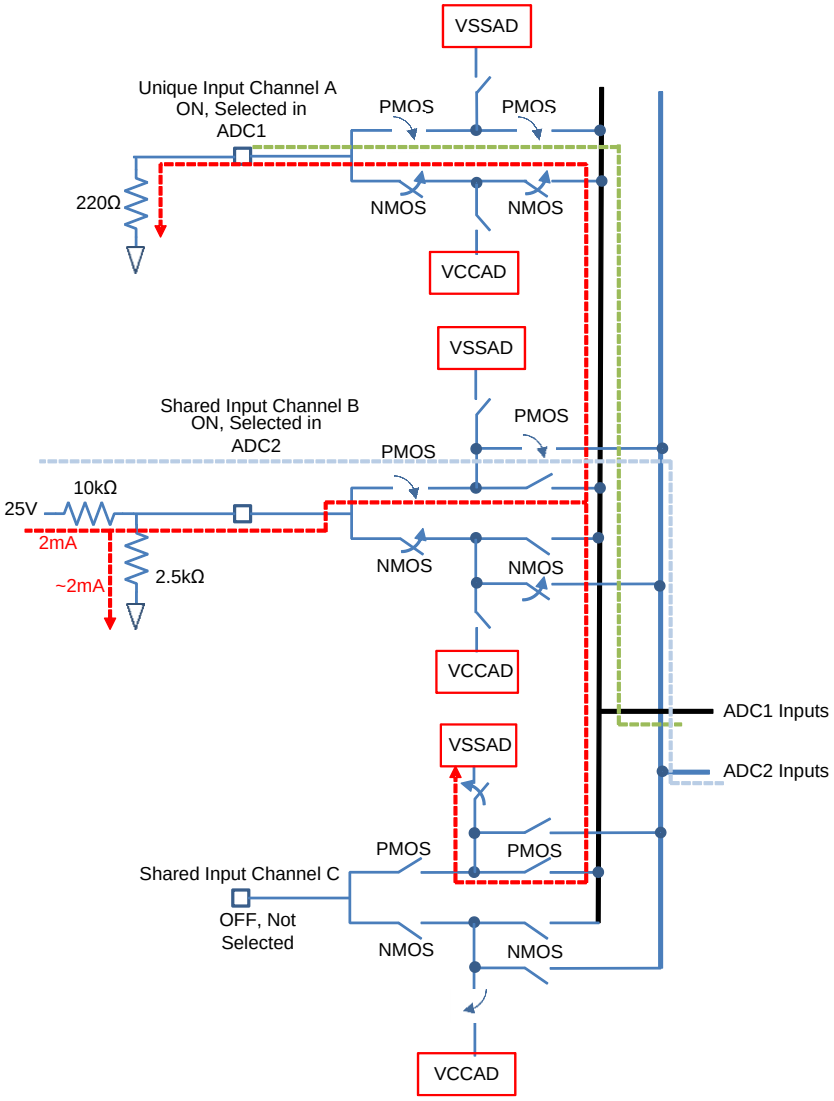

Figure 3. Example ADC1/ADC2 Channel Connection

Figure 3 shows an example where a ADC1 is sampling input channel A which is unique to ADC1, and ADC2 is sampling input channel B, which is a shared-input channel. This is shown by the dashed green and light blue current paths.

Another current path is shown in dashed dark red. This is a current injected into channel B as the input level on terminal B is greater than VCCAD - 0.3V. This is a parasitic current that passes through the "open" PMOS switch, and a part of this current flows to ground through the external 220 ohm resistor connected to input channel A. This causes an offset in the conversion result of channel A being sampled by ADC1.

Conditions

This issue occurs if:

1. Input voltage on a shared input channel being sampled by one ADC is (VCCAD - 0.3V) or higher, and
2. The second ADC samples another channel such that there is some overlap between the sampling windows of the two ADCs

Implications

An offset error is introduced in the conversion result of any channel if a current is being injected into a shared input channel.

Workaround(s)

There are two options to minimize the impact of this issue:

1. Configure the two ADC modules such that their sampling periods do not overlap, or
2. Limit the shared analog input upper limit to be lower than (VCCAD - 0.3V). The PMOS leakage is reduced exponentially if the input is lower than VCCAD - 0.3V.

AHB_ACCES_PORT#3 (ARM ID-529470) *Debugger may display unpredictable data in the memory browser window if a system reset occurs*

Severity 3-Medium

Expected Behavior If a system reset (nRST goes low) occurs while the debugger is performing an access on the system resource using system view, a slave error should be replied to the debugger.

Issue Instead, the response might indicate that the access completed successfully and return unpredictable data if the access was a read.

Condition System reset is asserted LOW on a specific cycle while the debugger is completing an access on the system using the system view. An example would be the debugger like the CCS's memory browser window is refreshing its content using the system view. This is not an issue for a CPU only reset. This is not an issue during power-on reset (nPORRST) either.

Implication(s) Data read using the debugger in system view while a system reset occurs may be corrupt, writes may be lost.

Workaround(s) This is a workaround for users and tools vendors.

Avoid performing debug reads and writes while the device might be reset.

CCMR5#1 *Self Test error is not generated if a mismatch is detected during the last two test vectors of the self-test for the VIM Compare block*

Severity 3-Medium

Expected Behavior When VIM compare logic is put under self-test, it should assert the self-test error signal to the ESM if the self-test fails

Issue The error signal is not generated to the ESM when the self-test fails during the compare mismatch test. However, both STET2 (self test error type) and STE2 (self test error) flags in the CCMSR2 register are still set properly.

Condition The VIM self-test error is not generated when:

1. the self-test is doing the compare mismatch test
2. the failure happens in the last two test vectors of the compare mismatch test

Implication(s) CPU will not be interrupted for the self-test error as the error signal is not asserted to ESM.

Workaround(s) After the application launches the self-test, it can first poll the STC2 bit to find out if the self-test is completed. Once the self-test is completed it will find out if the VIM compare self-test passed or failed by reading the STE2 and STET2 bits.

Application can still find out if the VIM compare self-test passed or failed by reading the STE2 and STET2 bits.

CCMR5#2 *Self-test for VIM compare block does not terminate immediately after it detects a self-test failure*

Severity 4-Low

Expected Behavior CCMR5 module not only compares the outputs between the two CPUs but also compares the outputs between the lockstep VIM modules. The VIM compare block can be put under self-test mode. Normally, the self-test will terminate immediately after it detects a failure.

Issue When a failure is detected in the VIM compare self-test mode, it does not terminate immediately but continues to the end of the self-test. If the user switches from self-test (CCMKEY2=0x6) to normal compare (CCMKEY2=0x0) mode before the self-test is completed then the new mode is not taken.

Condition During VIM compare self-test and if

1. a failure is detected &
2. a mode change immediately after the failure and before the self-test is complete

Implication(s) The switch to a new mode may not be taken.

Workaround(s) Wait until the self-test completes by polling the STC2 bit before reprogramming the VIM compare block to other modes.

CORTEX-R5#1 (ARM ID-772721) *In Standby Mode, peripheral port may lose read transactions*

Severity 4-Low

Expected Behavior AXI peripheral port (address range 0xF8000000-0xFFFFFFFF) read transaction should not be corrupted if the transaction is interrupted by an entry into standby mode using either Wait-For-Interrupt (WFI) or Wait-For-Event (WFE) instruction.

Issue The Cortex-R5 processor implements 32-bit AXI and AHB peripheral ports which can be used as an alternative master for all types of memory transactions apart from instruction fetches and certain doubleword accesses. The processor also implements Standby Mode, entered by executing a Wait-For-Interrupt (WFI) or Wait-For- Event (WFE) instruction. In Standby Mode the processor stops executing instructions and also stops the clock to most of the logic. Because of this erratum, if an AXI peripheral port read transaction has been interrupted it can be continually presented on the bus or ignore responses from the slave.

Condition 1. DBGNOCLKSTOP is not asserted, and

2. The CPU initiates a read via the AXI peripheral interface or the AXI virtual peripheral interface at an address which is marked as Normal-type memory, and

3. Before the read has completed, the CPU recognizes and takes an asynchronous exception (interrupt, asynchronous abort or debug entry request), and

4. The CPU subsequently executes WFI or WFE to enter Standby Mode, and

5. Either:

a. The address for the transaction has not been accepted before the CPU gates its clock or,

b. The read data for the transaction is returned while the CPU has its clock gated or,

c. The CPU is reset and the read data is returned after the CPU has restarted.

Implication(s) If this erratum occurs, data read from Normal memory on the AXI peripheral port may be corrupted or the CPU may deadlock.

In systems where interrupt handlers and asynchronous abort handlers return to re-execute the interrupted instruction (i.e. do not switch context), and the handler code does not itself execute WFI or WFE, this erratum cannot occur.

Workaround(s) The erratum can be avoided by ensuring that the whole of the AXI peripheral interface address space is marked

as Device-type memory. If the CPU is configured with an MPU, this can be achieved by programming the MPU region registers appropriately.

Alternatively, the erratum can be worked around by executing a load to a peripheral port address upon entering any interrupt or asynchronous abort handler routine, before any WFI/WFE or the exception return.

CORTEX-R5#3 (ARM ID-756523) *Watchpointed access in a store-multiple is not masked*

Severity 4-Low

Expected Behavior The Cortex-R5 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 but the watchpointed accesses should not perform writes on the bus to update memory.

Issue Due to this erratum this requirement is not met and the processor will incorrectly update memory for a watchpointed access.

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

1. A store multiple instruction is executed with at least 2 registers to be stored
2. The store multiple instruction writes to memory defined as Strongly-Ordered or Device
3. A watchpoint hit is generated for any access in the store multiple apart from the first access
4. The watchpoint hit is generated for an access with address bits[4:0] != 0x0

In these cases the store multiple will continue to perform writes until either the end of the store multiple or the end of the current cache line

Implication(s) Due to this erratum, the memory contents of the watchpointed address are updated before the debug event can be recognized. This means that a debugger:

1. Cannot always assume memory has not been updated when a watchpoint generates a debug event
2. Cannot prevent an access by setting a watchpoint on it

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

CORTEX-R5#5 (ARM ID-771872) *In Standby Mode, with n:1 clocking, AXI-S transactions are not accepted*

Severity 4-Low

Expected Behavior Cortex-R5 processor implements Standby Mode, entered by the execution of a Wait-For-Interrupt (WFI) or Wait-For-Event (WFE) instruction. In Standby Mode, the CPU stops executing instructions and once quiescent, can stop the clock to most of the logic. The CPU will restart its clocks and exit Standby Mode on receipt of wake-up stimulus such as an interrupt, as defined by the ARM Architecture. If the CPU receives a transaction on its AXI-S interface, (address range 0x30000000-0x33FFFFFF) it will restart its clock as necessary to handle the transaction but not exit Standby Mode.

Issue The Cortex-R5 processor implements AXI-slave (AXI-S) interfaces that can be clocked at an integer division of the processor clock by using the appropriate ACLKENSm signal. This n:1 clocking feature allows the processor to communicate with an AXI system operating at a lower frequency. Because of this erratum, when the AXI-S is clocked at an n:1 ratio to the rest of the CPU, the CPU will not respond to transactions received on the AXI-S when in Standby Mode. The CPU will only respond to the transaction after it exits Standby Mode due to an interrupt or other event.

Condition 1. The device is configured in Dual-Core mode where the only master that can possibly access a CPU's AXI-S I/F is by the other CPU

2. The CPU is put into Standby Mode by the execution of a WFI or WFE instruction, and

3. The AXI-slave interface becomes quiescent for at least one cycle, that is, it has no outstanding transactions and no transactions are being presented on the interface, and

4. Before the CPU has returned to Run Mode, a transaction is presented to the AXI-S interface. This only happens when the CPU not in standby mode tries to access the cache of the CPU in standby mode.

Implication(s) While it is not likely that one CPU would access the cache of the other CPU, if it does happen, then this erratum can cause a system deadlock if the generation of the wake up event (such as an interrupt) is predicated on the completion of the AXI-S transaction. If the wake up event is generated independently from the AXI-S transaction then the transaction will be delayed in completing, possibly significantly, which may have an impact on the performance of the system as a whole.

Workaround(s) This erratum can be avoided by making sure that one CPU does not access the cache of the other CPU while that CPU is in standby. This can be done by:

1) Never accessing the cache of the other CPU, or

2) Never putting a CPU in standby by executing the WFI or WFE instructions, or

3) By using semaphores to prevent one CPU from accessing the cache of the other while it is in standby mode.

CORTEX-R5#7 (ARM ID-780125) *Processor might deadlock or lose data when configured with cache-ECC*

Severity 3-Medium

Expected Behavior Cortex-R5 should be able to support ECC diagnostic checking on the cache memory when either write-through or write-back cache scheme is selected.

Issue When Write-back cache is selected and the ECC diagnostic on the data cache memory is enabled, it is possible for the store buffer to enter a state in which no existing writes will proceed. The effect of this state is either: - The pipeline backs-up preventing any instruction execution or - If a specific sequence of accesses is performed, execution resumes but write data is lost.

Condition Either of the following sequences of conditions is required for this erratum to occur:

1) The processor is implemented with data-cache ECC, and cache-ECC is enabled, and

2) The processor accesses a memory location that misses in the L1 data cache and causes a cache line to be read and allocated and

3) The processor performs a write to a Write-Back Cacheable location that hits before and misses after the linefill in step [2]. This write performs its cache lookup in the cycle before the line is reallocated by [2] and

4) The processor subsequently performs a read and a write to the same cache line as the write in step [3]. This read and write can occur in either order. The write is to a different doubleword than the write in step

[3].

or:

1) The processor is implemented with data-cache ECC, and cache-ECC is enabled, and

2) The processor reads a Write-Back Cacheable memory location that misses in the L1 data cache and causes a cache line to be read and allocated but does not detect any ECC errors and

3) The processor performs a write to the same cache line as the read in step [2]. When the address is looked up in the cache it appears to hit because of an ECC error in the tag-RAM and

4) The processor subsequently performs a further write to the same cache line as the read in step [2], but not the same doubleword as the write in step [3].

5) A subsequent speculative cache read also detects an ECC error. This read can be to the same cache set and therefore detect the same error, or a different set that requires a second ECC error for this condition to

be met.

In addition, both sets of conditions require specific timing relationships between the two accesses and are

therefore affected by the timing of transactions on the AXI bus and the status of other ongoing writes in the store

buffer.

Implication(s) Write data may be lost or pipeline backs-up preventing any instructions execution.

Workaround(s) You can avoid this erratum by setting ACTLR.DBWR (bit [14]) to 1. This setting disables an optimization to the internal transfer of bursts of write data to Normal memory. This setting also disables generation of AXI bursts

by the processor for Write-Through and Non-cacheable Normal memory, but not Write-Back memory. Setting

this bit to 1 might reduce the performance of writes to Normal memory by the processor. In benchmarking, the average performance reduction is less than 1% but routines that perform large block writes such as memset or

memcpy show substantially more significant impacts. The impact of the workaround on memset and memcpy is

strongly dependent on the performance and characteristics of the L2 memory system and on the exact instruction sequence used.

If your application permits it you can also avoid this erratum by disabling cache-ECC. Set ACTLR.CEC (bits[5:3]) to b100. This workaround does not reduce the performance of the processor, but disabling ECC has

implications for reliability.

DCC#24 *Single Shot Mode Count may be Incorrect*

Severity 3-Medium

Expected Behavior When the first clock source counts down to zero, the countdown value remaining for the other clock source is accurately captured.

Issue The first issue is that there is an offset in starting and stopping the two counters due to synchronization with VCLK that leads to a fixed offset. The second issue is that the value remaining in the counter that did not reach zero may be latched while the bits are in transition, giving an erroneous value.

Condition When used in single shot mode and the count value captured is not from VCLK.

Implication(s) The cycle count captured may be incorrect.

Workaround(s) Static frequency offset can be removed by making two measurements and subtracting. The sporadic offset can be removed by making multiple measurements and discarding outliers -- an odd filtering algorithm.

DEVICE#31 *RAM ECC memory space is inaccessible through DAP from the debugger*

Severity 4-Low

Expected Behavior Debugger should be able to access the full memory region via DAP.

Issue The entire SRAM ECC space from 0x08400000-0x0847FFFF is inaccessible to the DAP.

Condition Always

Implication(s) User cannot use debugger to view the contents of the SRAM ECC space via DAP.

Workaround(s) Debugger can still use the CPU to access the ECC space.

DEVICE#32 *DMA cannot continue to access SRAM after the device comes out of global low power mode*

Severity 5-None

Expected Behavior The device can enter global low power mode in the middle of a DMA access to the SRAM. After low power mode is exited the DMA should resume the access to the SRAM from where it left off.
Issue

Issue When the device is exited out of low power mode, the DMA will not get the proper data from the SRAM. Instead, an error is generated by the interconnect to the ESM module.

Condition When the on-going DMA transfer is interrupted by the device going into low power mode and resumes the transfer after the low power mode is exited.

Implication(s)

Implication(s) Wrong data is transferred.

Workaround(s) Application to ensure either that DMA is first disabled or wait until the transfer is completed before entering LPM. After LPM mode is exited the DMA can be re-enabled to resume transfers. The issue happening to DMA can also apply to other bus masters. It is recommended that all masters are first disabled to ensure that there are no on-going transfers or bus transactions before low power mode is entered.

DEVICE#37 *AJSM Visible Unlock Code has ECC Value not Equal to 0xFFFF*

Severity 4-Low

Expected Behavior The visible unlock code should have mostly 1's and an ECC value of all 1's to allow the maximum number of choices that can be created by programming 0's over the existing unlock value.

Issue The visible unlock code used has only about half of the bits as 1's and an ECC value of 0x172C.

Condition None

Implication(s) The number of possible valid customer lock codes that can be created is significantly reduced.

Workaround(s) A small computer program or script can be used to generate random 128-bit key values and the corresponding ECC. It then creates a list of choices in which the 128-bit key and the 16-bit ECC can be created by only programming zeros over the existing values.

DEVICE#39 *Extra WE/OE pulses in asynchronous EMIF access*

Severity 3-Medium

Expected Behavior The number of WE/OE pulses in accessing asynchronous EMIF should match the data size. For example, there should be only one WE pulse for writing a 16-bit data value to a 16-bit wide EMIF

Issue Extra WE/OE pulses are generated in asynchronous EMIF accesses. Write: Extra WE pulses are generated for all writes (8/16/32/64bit) . There would be at most two extra WE pulses. Read: All reads (8/16/32/64 bit) behaves like 64 bits reads at the EMIF port..

Condition All asynchronous EMIF writes and non-64 bit asynchronous EMIF reads.

Implication(s) The extra WE/OE cycles make it impossible to use the EMIF to program/erase external flash memory or access external peripherals with FIFO interfaces.

Reading and writing to asynchronous RAM or reading from FLASH works, but is less efficient because of the redundant read or write cycles.

Workaround(s) The WE signal can be externally gated (OR gate) with the DQM signals to suppress the extra WE pulses. Extra write cycles will be wasted.

There is no way to suppress the extra OE signals by hardware. However, user can left shift the address (2 bit for 16 bit EMIF and 3 bit for 8 bit EMIF) and use external hardware to only allow the first OE pulse to pass. Extra read cycles will be wasted.

DEVICE#40 *Abort is not generated when writing to unimplemented locations in some peripheral frames*

Severity 4-Low

Expected Behavior Writes to unimplemented spaces in some peripheral frames will result in an abort according to the spec.

Issue Writes to the unimplemented locations within the peripheral frames on the following peripherals do not generate an abort.

- CPPI memory Slave (Ethernet RAM)
- All ETPWMs
- All ECAPs
- All EQEPs
- EMIF registers
- CRC1 and CRC2
- All MIBSPI RAM
- All DCAN RAMs
- NHET RAMs
- HTU RAMs
- FlexRay TU RAMs
- HTU registers
- DMA RAM
- RTP RAM
- Flash wrapper
- eFuse Farm Controller
- Power Domain Control
- SCM
- EPC
- NMPUs
- DMA
- L2RAMW

Condition Writing to unimplemented locations when the peripheral frame is configured as "device" memory.

Implication(s) Writes to unimplemented locations within these peripheral frames will have no affect and reads will correctly generate an abort. Improperly executing software that writes an incorrect location may not be easily caught because of the lack of an abort.

Workaround(s) Configure all peripheral frames as "strongly ordered" memory to enable the abort. This has a performance penalty when writing to peripheral registers or peripheral memories.

DEVICE#41 *Power-on Reset During Bank 7 Sector Erase May Corrupt Other Sectors*

Severity 3-Medium

Expected Behavior A power-on reset (nPORRST) during sector erase will not affect sectors other than the one being erased.

Issue A power-on reset while doing a sector erase may corrupt other sectors. Typically sector zero is corrupted when trying to erase one of the other sectors.

Condition Doing sector erase of bank 7

Implication(s) Data in the other sector will be partially erased. This will likely result in ECC errors when trying to read data from that sector.

Workaround(s) Before starting the sector erase, write to the FSM_SECTOR1 register (0xFFF872C0) only enabling the sector desired to be erased.

DEVICE#42 *Reads of Bank7 may be Corrupted by Short Glitches on nPORRST Pin*

Severity 3-Medium

Expected Behavior Glitches on nPORRST less than 475ns in length are ignored by the device

Issue Glitches on nPORRST may reset bank 7 of the flash without causing a CPU reset

Condition Low going voltage glitch on the nPORRST pin

Implication(s) The CPU will read bad data from flash bank 7 if the read happens during the glitch. This will most likely cause an ECC error, but an ECC error is not guaranteed.

Workaround(s) Good layout practices keeping the nPORRST trace protected from induced noise will prevent occurrence of this problem. Otherwise reading each location twice and comparing the value read should guarantee a proper read.

DEVICE#45 *ePWM tripzone mode "high impedance state" does not work*

Severity 3-Medium

Expected Behavior EPPWMx output pins can be configured into one of the following four modes in response to a tripzone (fault) condition. High-impedance Force to high state Force to low state No action taken (Ignore any fault condition)

Issue The tripzone action high-impedance" behaves like no action taken on EPWMx pins.

Condition Always

Implication(s) If the application tries to protect the system based on tripzone high-impedance" mode, the PWM output will continue and the protection will fail.

Workaround(s) Do not define the tripzone action as high-impedance. Instead, set it as force to high state or force to low state according to the application requirements.

DEVICE#46 *No ESM notification when CPU livelock occurs due to hard error in cache memory*

Severity 2-High

Expected Behavior There should be a non-maskable ESM event to the system when CPU livelock (keep fetching code/data from memory) due to hard error in cache memory.

Issue CPU livelock event signal is not hooked up to ESM. No non-maskable ESM event is generated when CPU livelock occurs.

Condition Hard error in Cache

Implication(s) No indication to system when CPU is locked up due to hard error in cache.

Workaround(s) There are three options. They all have limitations.

(1) Enable CPU cache correctable event on ESM.

(2) Enable abort on cache correctable errors.

(3) Use watchdog timer.

Option (1) and (2) will generate ESM or abort on cache soft errors too. There will be a longer delay in reporting the error with option 3. All needs to be setup by software before cache is enabled.

DEVICE#47 *STC1 (CPU) test cannot be run on segment 1 independently.*

Severity 3-Medium

Expected Behavior The STC self test should be run any one segment independently.

Issue STC test cannot be run on segment 1 independently.

Condition Run STC test on segment 1 alone.

Implication(s) User cannot run STC test on segment 1 alone.

Workaround(s) Run segments zero and one together to make sure segment one is also tested.

DEVICE#48 *Interconnect Global Error flag is set after STC1 (CPU) self test completion*

Severity 3-Medium

Expected Behavior There should be no error after STC1 (CPU) self test.

Issue The Interconnect Global Error (ESM group 1, channel 52) is set intermittently after STC1 (CPU) self test.

Condition System clock GCLK is faster than the maximum rated speed for HCLK.

Implication(s) The ESM group 1 channel 52 flag, Interconnect Global Error, will be set.

Workaround(s) (1)Clear the ESM group 1 channel 52 flag if set after the STC1 (CPU) self-test.

(2)Slow down GCLK to the maximum spec of HCLK the STC1 (CPU) self-test.

DEVICE#49 *False interconnect safety checker error flag*

Severity 3-Medium

Expected Behavior CPU Interconnect Subsystem - Global error flag (ESM group 1 channel 52) should not be set for untaken speculative data fetches.

Issue CPU Interconnect Subsystem - Global error flag (ESM group 1 channel 52) may be set for untaken speculative data fetches.

Condition All the following conditions have to occur.

(1)The memory attribute is configured as normal.

(2)There is a speculative fetch outside the valid memory range.

(3)Multiple bus masters are configured for use.

Implication(s) CPU Interconnect Subsystem - Global error flag (ESM group 1 channel 52) is falsely set.

Workaround(s) Enable CPU MPU to block any access (real and speculative) outside the valid memory range.

DEVICE#50 *STC2 (nHET) selftest must run with STCCLKDIV greater than zero*

Severity 3-Medium

Expected Behavior Self test controller 2 (STC2), testing the two nHETs, should function at any STCCLKDIV value.

Issue STC2 test will give false errors when STCCLKDIV is zero.

Condition When executing STC test on nHET.

Implication(s) STC self test on nHET will fail when the value of STCCLKDIV is zero.

Workaround(s) Use non-zero STCCLKDIV value to run STC2 selftest.

DEVICE#51 *Values in the Interconnect status registers are left-shifted left 24 bits*

Severity 4-Low

Expected Behavior Reading the interconnect status register is expected to return the proper value in bits 7 through 0 as documented.

Issue With the issue, the values of the interconnect registers bits 7 through 0 appear in bit locations 31 through 24.

Condition Any read of the interconnect registers (addresses 0xFA000000 through 0xFA00002B)

Implication(s) Byte reads from the specified byte location returns zero.

Workaround(s) Software is needed to right-shift the register by 24 bits.

DEVICE#54 *Writes and reads to EMAC CPPI memory are byte swapped on big endian device (TMS570)*

Severity 3-Medium

Expected Behavior Writes and reads to EMAC CPPI packet buffer descriptors correctly regardless of the endiannes of the device.

Issue CPU reads/writes to EMAC CPPI memory are byte-swapped on big endian device (TMS570).

Condition EMAC CPPI memory access on big endian device (TMS570).

Implication(s) EMAC does not work as expected.

The software for other Hercules devices will not work correctly.

Workaround(s) The software must apply byte swapping before writing data to the CPPI memory, or after reading the CPPI memory. This is already done if using HalCoGen to develop the software.

DEVICE#55 *False VIM Compare Error in hardware vector mode*

Severity 2-High

Expected Behavior No VIM compare errors are generated when servicing interrupts.

Issue A VIM compare error may be generated when interrupt is serviced in hardware vector mode.

Condition When using hardware vector mode with GCLK frequency greater than 200MHz on lock-step CPUs.

Implication(s) The VIM compare error is a group 2 error. A non-maskable, high priority ESM interrupt will be generated and the nERROR pin will toggle.

Workaround(s) Use software vector mode

DEVICE#56 *nERROR assertion on debugger connect*

Severity 4-Low

Expected Behavior No errors should be detected when connecting to the device by JTAG

Issue Sometimes a CPU compare error (ESM Group 2 channel 2) is generated when the debugger connects to device.

Condition Upon a debugger initially connecting to the device.

Implication(s) The nError pin will toggle upon initial connection with the debugger.

Workaround(s) Clear the nERROR by writing 0x5 to the ESMEKR key register in the ESM module and ignore the nERROR pin toggle which happens immediately upon the debugger connecting.

DEVICE#57 *Cannot Set Breakpoints While nRST is Asserted*

Severity 3-Medium

Expected Behavior The debug logic on the Cortex R5 can be setup while a debugger holds the device in a system reset

Issue The debug logic in the Cortex R5 is being reset by a system reset instead of only a power-on reset (nPORRST).

Condition While debugging with a JTAG based emulator

Implication(s) Debug system are unable to hold a part in system reset and then setup a breakpoint to stop the CPU upon execution of the first instruction. This makes it difficult to put a part that has code programmed in it back into the reset state for debugging.

Workaround(s) None

DEVICE#59 *CPU Interconnect Self-test does not set status flags*

Severity 4-Low

Expected Behavior The CPU subsystem interconnect can be placed under self-test. When the self-test is complete, it is expected that the PT_OK (positive test has succeeded) and NT_OK (negative test has succeeded) bits in the STC_STATUS register to be set.

Issue The PT_OK and NT_OK status bits are not set even on successful completion.

Condition None

Implication(s) The software incorrectly determines that the interconnect self-test has failed.

Workaround(s) Ignore the NT_OK and the PT_OK status bits. ESM group 3 channel 12 flag will be set upon self-test failure.

DMAOCP#01 *DMA acts as if it receives an extra DMA request from a peripheral*

Severity 2-High

Expected Behavior DMA does one transfer for each request from a peripheral

Issue When a new DMA request comes in the same cycle as the cycle that the DMA finishes processing a request on that channel, the DMA will do two transfers for that one request.

Condition When a new request comes in the same cycle as the DMA finishes the previous request.

Implication(s) The peripheral may not be able to properly handle the second transfer

Workaround(s) None

ERAY#52 (FLEXRAY#52) *Wakeup Symbol (WUS) Generates Redundant Wakeup Interrupts (SIR.WUPA/B)*

Severity 4-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 The FlexRay module detects a valid wakeup pattern (WUP) 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 4-Low

Expected Behavior Correct cycle offset in spite 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 issue 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 Wakeup Symbol (WUS) Following Received Valid Wakeup Pattern (WUP) May Be Ignored*

Severity 4-Low

Expected Behavior The FlexRay controller protocol engine should recognize all wakeup symbols (WUS).

Issue The FlexRay controller protocol engine may ignore the first wakeup symbol (WUS) following the below stated state transition, therefore it sets the wakeup status interrupt flags (SIR.WUPA/B) at the third WUS instead of the second WUS.

Condition The issue is limited to the reception of redundant wakeup patterns. When the protocol engine is in WAKEUP_LISTEN state and receives a valid wakeup pattern (WUP), it transfer into READY state 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 4-Low

Expected Behavior The FlexRay module should ignore a READY command while in READY state.

Issue The FlexRay module does not ignore a READY command while in READY state.

Condition The Protocol Operation Controller (POC) issues a READY command while in READY state.

Implication(s) The coldstart inhibit bit CCSV.CSI is set whenever the POC enters READY state.

Workaround(s) None.

ERAY#61 (FLEXRAY#61) *The Transmission Slot Mode Bit Is Reset Immediately When Entering HALT State*

Severity 4-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. The mode can be changed in DEFAULT_CONFIG or CONFIG state only.

Issue Transmission slot mode bit is immediately reset to SINGLE slot mode (CCSV.SLM[1:0] = "00").

Condition The protocol engine is in NORMAL_ACTIVE or NORMAL_PASSIVE state, and a HALT or FREEZE command is issued by the CPU

Implication(s) The transmission slot mode is reset to SINGLE when entering HALT state.

Workaround(s) None.

ERAY#64 (FLEXRAY#64) *SBESTAT register is not readable*

Severity 3-Medium

Expected Behavior The Single Bit Error Status Register (SBESTAT) is supposed to indicate the status and the location of the single bit error when a single bit error is detected.

Issue When a single bit error is detected, reading the SBESTAT does not show any status. The SBESTAT register will instead read zero.

Condition When a single bit error is detected.

Implication(s) In the SBESTAT register there is the Single Bit Error (SBE) flag and other bits to indicate the location of the single bit error. With this issue, the application cannot assess the location of the error. However, the error is corrected and the single bit error is still asserted to the ESM group 1 channel 72, if enabled.

Workaround(s) None

ERAY#68 (FLEXRAY#68) *Data transfer overrun for message transfers Message RAM to Output Buffer (OBF) or from Input Buffer (IBF) to Message RAM*

Severity Medium

Expected Behavior Data transfers should not overrun the expected receive buffer.

Issue 1) A message buffer transfer from Message RAM to OBF When the message buffer has its payload configured to maximum length (PLC = 127), the OBF word on address 00h (payload data bytes 0 to 3) is overwritten with unexpected data at the end of the transfer.

2) A message buffer transfer from IBF to Message RAM After the Data Section of the selected message buffer in the Message RAM has been written, one additional write access overwrites the following word in the Message RAM which might be the first word of the next Data Section

Conditions The problem occurs under the following conditions:

1) A received message is transferred from the Transient Buffer RAM (TBF) to the message buffer that has its data pointer pointing to the first word of the Message RAM's Data Partition located directly after the last header word of the Header Partition of the Last Configured Buffer as defined by MRC.LCB.

2) The Host triggers a transfer from / to the Last Configured Buffer in the Message RAM with a specific time relation to the start of the TBF transfer described under 1).

Implications 1) When a message is transferred from the Last Configured Buffer in the Message RAM to the OBF and PLC = 127 it may happen, that at the end of the transfer the OBF word on address 00h (payload data bytes 0 to 3) is overwritten with unexpected data (see Figure 4).

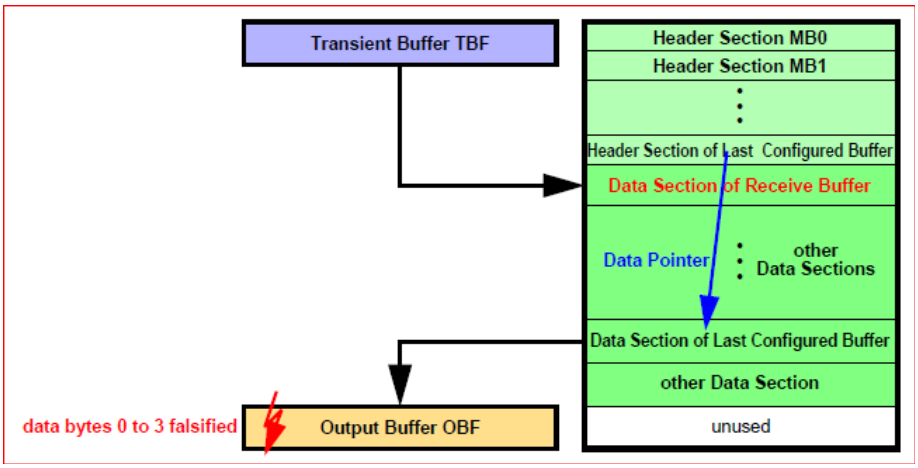

Figure 4. First Fail Mode

2) When a message is transferred from IBF to the Last Configured Buffer in the Message RAM, it may happen, that at the end of the transfer of the Data Section one additional write access overwrites the following word, which may be the first word of another message's Data Section in the Message RAM (see Figure 5).

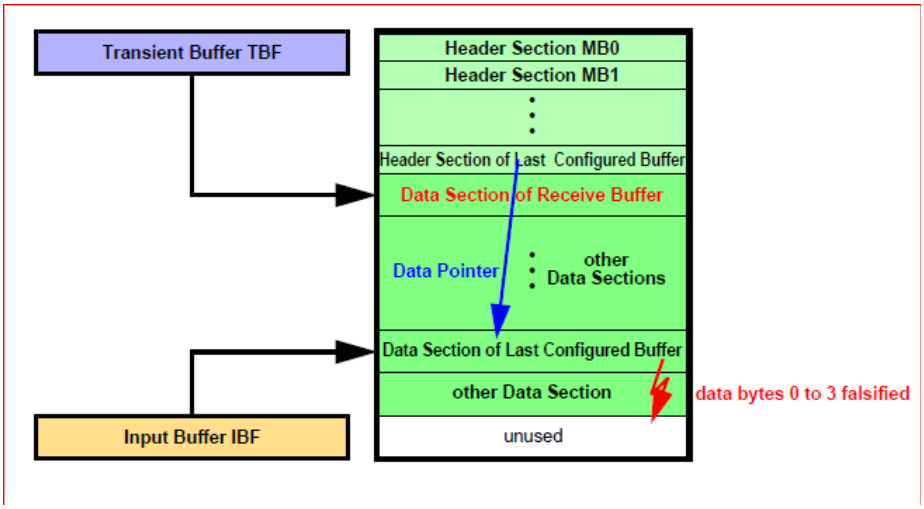

Figure 5. Second Fail Mode

Workaround(s)

1) Leave at least one unused word in the Message RAM between Header Section and Data Section.

OR

2) Ensure that the Data Section directly following the Header Partion is assigned to a transmit buffer.

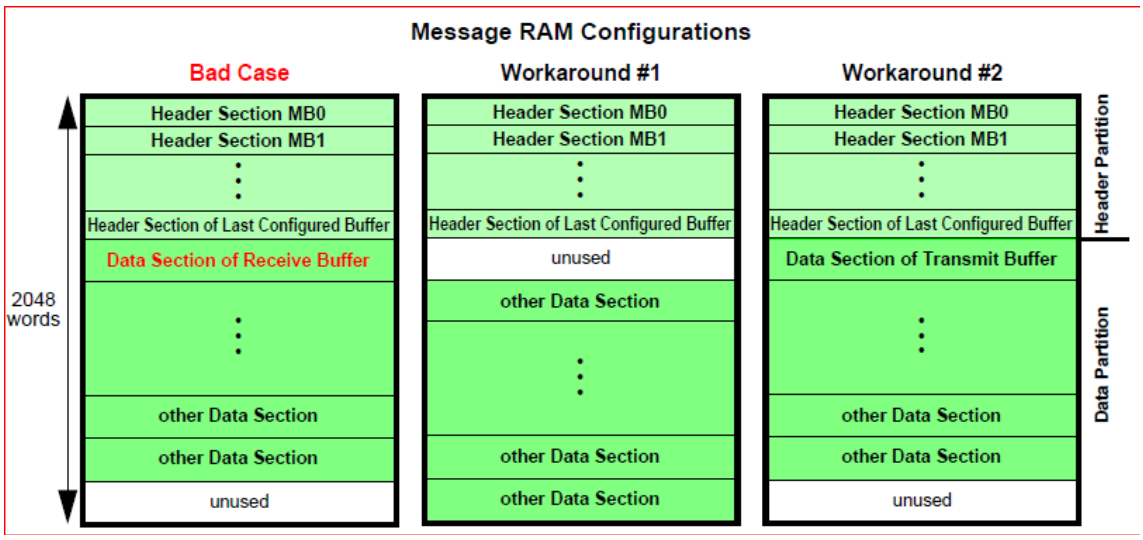

Figure 6. Workarounds

ERAY#69 (FLEXRAY#69) *Missing startup frame in cycle 0 at coldstart after FREEZE or READY command*

Severity 3-Medium

Expected Behavior When a coldstart node re-enters startup, it listens to its attached channels and attempts to receive FlexRay frames. If no communication is received, the node commences a coldstart attempt which begins with the transmission of a collision avoidance symbol (CAS). Only the coldstart node that transmits the CAS transmits the startup frames in the first four cycles (from cycle 0 to cycle 3) after the CAS.

Issue The FlexRay may not transmit its startup frame in the first cycle after CAS (cycle 0) when it is restarted as the leading coldstarter (was stopped by FREEZE or READY).

Condition The issue is limited to the following condition:

1. FlexRay has been stopped by FREEZE or READY command.
2. FlexRay is configured with startup frames with 1 to 7 slots
3. A coldstart after hardware reset is not affected.

Implication(s) Startup frame is not sent in cycle 0 after entering COLDSTART_COLLISION_RESOLUTION state from COLDSTART_LISTEN state.

Workaround(s) Configure the FlexRay to use a static slot greater or equal 8 for the startup / sync message.

FTU#08 *FlexRay Transfer Unit Not Disabled On Memory Protection Violation (MPV) Error*

Severity 3-Medium

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

Issue The FTU does not get disabled under some conditions.

Condition 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

Implication(s) The MPV error flag in the Transfer Error Interrupt Flag (TEIF) register is set, but the Transfer Unit Enabled (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 erratum 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 4-Low

Expected Behavior 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.

Condition 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.

Implication(s) 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.

GCM#58 *N2HET and HTU require VCLK to be faster than HCLK/4*

Severity 4-Low

Expected Behavior N2HET and HTU rely on both VCLK and VCLK2. These modules require phase alignment between VCLK and VCLK2 (i.e. every VCLK rising edge corresponds to a VCLK2 rising edge). If a reset occurs, the phase relationship is expected to be re-established after a reset. Note that the Technical Reference Manual requires VCLK2 to be an integer multiple of VCLK.

Issue If VCLK is equal to HCLK/4 or slower, then the phase alignment between the rising edge of VCLK and the rising edge of VCLK2 can be lost after any system reset except power-on reset. Once lost, the alignment can only be re-established with a power-on reset. (A system reset is a reset that affects the entire system (e.g. system control registers). Some causes for a system reset are (1) power-on reset, (2) an oscillator or PLL failure (when the response to the failure has been configured to generate a reset), (3) a software reset, (4) a watchdog reset, (5) a debug reset , or (6) a reset generated from outside and propagated through nRST.)

Condition This issue may occur if both conditions (below) occur:

1. VCLK is HCLK/4 or slower
2. Any system reset except power-on reset occurs

If both conditions (above) occur, VCLK may not be able to re-align its phase to VCLK2 without a power-on reset.

Implication(s) This issue breaks the required phase alignment between VCLK and VCLK2. Since N2HET and HTU use both VCLK and VCLK2, the functionality of the modules is not guaranteed when the clocks are not properly aligned.

In a typical application, VCLK is unlikely to be configured as slow as HCLK/4 based upon performance requirements.

Workaround(s) The recommended workaround is to configure VCLK no slower than HCLK/3. (Since VCLK2 is required to be an integer multiple of VCLK, VCLK2 also must be no slower than HCLK/3.)

None if either VCLK or VCLK2 must be HCLK/4 or slower per application requirement. Otherwise, the recommendation is not to configure VCLK or VCLK2 at HCLK/4 or slower ratio.

GCM#59 *Oscillator can be disabled while PLL is running*

Severity 4-Low

Expected Behavior No clock source can be disabled if it is being used

Issue The oscillator can be disabled if the PLL is the only thing using it as a clock source

Condition The oscillator may be disabled if:

1. no clock domain relies upon the oscillator
2. no clock domain relies upon any PLL

Implication(s) This issue allows the oscillator to be disabled while used by the PLL. When the oscillator disables, the PLL will slip. The system behaves exactly like it would in case of a PLL slip. The response includes:

1. setting the RF SLIP flag (GBLSTAT.8)
2. switching Clock Source 1 from the PLL (if enabled). This autonomous switch prevents use of the PLL until the fault is cleared.
3. the device generates an ESM error (if enabled)
4. Cause a reset if the Reset-On-Slip Failure bit is set in PLLCTRL1.

If the software now uses the PLL as a clock source, there will be a long delay (mS) for the oscillator and the PLL to restart and provide a clock. Additionally, the SLIP flag(s) must be cleared in order for the PLL to propagate to the clock domains.

Normally this is not an issue as the software should not attempt to disable the oscillator when it is being used by the PLL. Also, once the PLL is stable and used as a clock source, the oscillator can no longer be disabled.

Workaround(s) Since the PLL is a secondary clock source dependent on the Oscillator input, the user software should not disable the Oscillator while the PLL is enabled while neither of them are sources for any of the clock domains.

LPO#16 *Oscillator Fault Detection Starts too Soon after power-on reset is released*

Severity 4-Low

Expected Behavior The oscillator fault detection circuit allows approximately 200ms after the release of power-on reset 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 power-on reset.

Condition This issue occurs due to an uninitialized internal signal at power on; the signal is initialized if the oscillator is active when power-on reset is released. Once initialized, the oscillator fault detection works robustly.

Note: The nature of the issue makes it difficult to reproduce on a small number of units.

Implication(s) This fault errantly detects an oscillator fault. The system behaves exactly like it would in case of a real oscillator fault. The response includes:

1. setting the OSC FAIL flag (GBLSTAT.0)
2. switching Clock Source 0 from the oscillator to the HFLPO.
3. switching Clock Source 1 from the PLL to HFLPO.

These autonomous switches prevent use of the oscillator or PLL until the fault is cleared.

4. the device generates an ESM error (if enabled)
5. Cause a reset if the Reset-On-Oscillator Failure bit is set in PLLCTRL1.

Workaround(s) This issue is avoided if power-on reset is held low long enough for the oscillator to stabilize.

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 occurred, the software can attempt to restart the oscillator as described in the Technical Reference Manual.

MIBSPI#110 *Multibuffered SPI in Slave Mode In 3- or 4-Pin Communication Transmits Data Incorrectly for Slow SPICLK Frequencies and for Clock Phase = 1*

Severity 3-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 transmit incorrect data.

Condition This issue can occur under the following condition:

- Module is configured to be in multi-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

Implication(s) 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 3-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.

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

Implication(s) 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.

MIBSPI#136 *Transfer Complete Interrupt is not generated after completing all the transfers when a Transfer Group is set to end at buffer 128*

Severity 4-Low

Expected Behavior If transfer group TG0 is configured with 128 buffers and when the transfer is complete it should generate a TG_COMPLETE interrupt

Issue Transfer Complete Interrupt is not generated after the transfer is complete even though the transfer complete flag in register TGINTFLG (offset 0x84) is set.

Condition This erratum is only valid when:

1. A MibSPI instance that does not support the extended buffer feature is used.
2. The number of buffers intended by the application for Transfer Group 0 (TG0) is 128 or 0x80.
3. The last buffer number in a transfer group is set to 128 by setting the first buffer of the next transfer group (PSTART) to "0x00". (Note that PSTART-1 of next transfer group becomes last buffer number for the current transfer group)

Implication(s) Transfer Complete interrupt is not generated to indicate the completion of Muti-buffer SPI transmission.

Workaround(s) For MIbSPI instances that do not support the extended buffer feature, setup the next unused transfer group with its first buffer (PSTART) configured as 0x80. This enables the transfer group complete interrupt for the current transfer group to be generated after transferring up to 128 buffers.

If all available transfer groups are required for actual transfers, program LPEND field of LTGPEND register as 0x80.

MIBSPI#137 *Spurious RX DMA REQ from a Slave mode MIBSPI*

Severity 4-Low

Expected Behavior The MIBSPI should not generate DMA requests when it has not received data from the SPI master

Issue A spurious DMA request is generated even when the SPI slave is not transferring data.

Condition This erratum is only valid when all below conditions are true:

- The MIBSPI is configured in standard (not multi-buffered) SPI mode as a slave.
- SPIINT0.16 (DMA_REQ_EN) bit is set to enable DMA requests.
- The nSCS (Chip Select) pin is in active state, but no transfers are active.
- The SPI is disabled by clearing SPIGCR1.24 (SPIEN) bit from '1' to '0'.

The above sequence triggers a false request pulse on the Receive DMA Request as soon as SPIEN bit is cleared from '1' to '0'.

Implication(s) The SPI generates a false DMA request to the DMA module when the data is not yet available for the DMA module to retrieve.

Workaround(s) Whenever the SPI is to be disabled by clearing SPIEN bit, clear the DMA_REQ_EN bit to '0' first and then clear the SPIEN bit.

MIBSPI#138 *MIBSPI RAM ECC is not read correctly in DIAG mode*

Severity 4-Low

Expected Behavior ECC values of MIBSPI RAM should be read correctly

Issue Read operation to ECC address space of MIBSPI RAM in DIAG mode does not return correct ECC value for the first 128 buffers if the Extended Buffer support is implemented but the Extended Mode is disabled for the particular MibSPI instance.

Condition Always

Implication(s) Need to enable Extended Buffer Mode to access ECC bits corresponding to the first 128 buffers of RXRAM.

Workaround(s) Enable Extended Buffer Mode to access ECC bits corresponding to the first 128buffers of RXRAM. In this mode, RXRAM ECC bits corresponding to the first 128 buffers can be accessed at a different location 0xC00-0xDFF.

MIBSPI#139 *Mibspi RX RAM RXEMPTY bit does not get cleared after reading*

Severity 3-Medium

Expected Behavior The MibSPI RXEMPTY flag is auto-cleared after a CPU or DMA read.

Issue Under a certain condition, the RXEMPTY flag is not auto-cleared after a CPU or DMA read.

Condition The TXFULL flag of the latest buffer that the sequencer read out of transmit RAM for the currently active transfer group is 0, AND

A higher priority transfer group interrupts the current transfer group and the sequencer starts to read the first buffer of the new transfer group from the transmit RAM, AND

Simultaneously, the host (CPU/DMA) is reading out a receive RAM location that contains valid received data from the previous transfers.

Implication(s) The fake RXEMPTY '1' suspends the next Mibspi transfer with BUFMODE 6 or 7.

With other BUFMODEs, a false "Receive data buffer overrun" will be reported for the next Mibspi transfer.

Workaround(s) 1. If at all possible, avoid transfer groups interrupting one another.

2. If dummy buffers are used in lower priority transfer group, select appropriate "BUFMODE" for them (like SKIP/DISABLED) unless there is a specific need to use the "SUSPEND" mode.

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

Severity 2-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.

Condition 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.

Implication(s) Because of this boundary condition, the PCNT and WCAP instructions should not be used on small pulses.

Workaround(s) None

NHET#55 *More than one PCNT instruction on the same pin results in measurement error*

Severity

3 - Medium

Expected Behavior

It should be possible to use more than one Period/Pulse Count (PCNT) instruction to measure a single pin, as long as only one of the PCNT instructions is configured for high resolution (hr_lr=HIGH). For example, consider the following code fragments.

Code Fragment 1 - Should Be OK, But Fails Due to This Issue

```
PC1    PCNT { hr_lr=HIGH, type=RISE2FALL, pin=2};
PC2    PCNT { hr_lr=LOW,  type=FALL2FALL, pin=2};
```

Code Fragment 2 - Should Be OK, But Fails Due to This Issue

```
PC1    PCNT { hr_lr=LOW,  type=RISE2FALL, pin=2};
PC2    PCNT { hr_lr=HIGH, type=FALL2FALL, pin=2};
```

Code fragments 1 and 2 should work properly because only one of the two PCNT instructions are configured for hr_lr=HIGH, and there is one hi-res structure available.

Issue

There are two issues.

1. A measurement error is introduced into the result of the PCNT instruction with hr_lr=HIGH. Normally this instruction would return a result to within ±½ high resolution clock periods of the actual result, due to quantization noise. However another PCNT instruction on the same pin causes an error of up to ±1 loop resolution period. Note that this error is greater than the normal loop resolution period error of ±½ loop resolution period; because the high-resolution bits also contribute to the error in this case.
2. A measurement error is introduced into the result of the PCNT instruction with hr_lr=LOW. The PCNT instruction with hr_lr=LOW should return a value with 0's in bit positions 6:0 (the high-resolution portion of the measurement result). This is the case when both PCNT instructions are set for hr_lr=LOW (Code Fragment 3) but for Code Fragments 1 and 2 the loop resolution PCNT returns a non-zero in bit positions 6:0.

Conditions

This problem occurs when both conditions are true:

1. More than one PCNT selecting the same pin number is executed during the same loop resolution period.
2. One of the PCNT instructions is configured for high resolution (hr_lr=HIGH).

Please also note that the N2HET assembler defaults to high resolution for PCNT if the hr_lr field is not specified as part of the instruction. Therefore unless the instruction is coded explicitly with 'hr_lr=LOW as an option, the assembler will create N2HET machine code with hr_lr=HIGH.'

Implications

The impact is greatest when workaround option 1 cannot be applied due to the number of timer pins required by the application. If Option 1 cannot be applied, then the PCNT measurements on this pin are reduced to ±½ loop resolution period.

Workaround(s)

Option 1 - Use the HR Share feature and make both measurements with hr_lr=HIGH. First, set the appropriate HRSHARE bit in the HETHRSH register. In the following example this means setting HETHRSH bit 1 - "HRSHARE3/2". This bit causes the input of device pin 2 to drive the N2HET pin inputs 2 and 3. Then modify the N2HET code sequence to use pin 3 for one of the PCNT instructions:

Code Fragment 1 Modified for HR Share

```
PC1    PCNT { hr_lr=HIGH, type=RISE2FALL, pin=2};
PC2    PCNT { hr_lr=HIGH, type=FALL2FALL, pin=3};
```

This option exceeds the original measurement resolution objective because both PCNT measurements are made with high-resolution. The disadvantage of this workaround is that it requires the high-resolution structure of pin 3, leaving pin 3 only useable as a GPIO pin rather than as a timer pin.

Option 2 - Use only loop resolution mode PCNT instructions (as in Code Fragment 3). This will work properly while leaving pin 3 available for timing functions, but the resolution on both the period and duty cycle measurements are reduced to loop resolution.

Code Fragment 3 - OK

```
PC1    PCNT { hr_lr=LOW, type=RISE2FALL, pin=2};
PC2    PCNT { hr_lr=LOW, type=FALL2FALL, pin=2};
```

PBIST#4 *PBIST ROM Algorithm Doesn't Execute*

Severity 3-Medium

Expected Behavior PBIST controller checks memories with the specified algorithm as documented in the TRM

Issue The possibility that the PBIST algorithms will not execute only occurs when PBIST is initially run after power on reset. Once the PBIST controller starts working, it will continue to work until the next power cycle.

Condition The possibility that the PBIST algorithms will not download only occurs when PBIST is initially executed after power on reset. Once the PBIST ROM starts working, it will continue to work until the next power cycle.

Implication(s) The PBIST test may return with a pass status, even though the algorithm was not properly executed

Workaround(s) This condition does not occur often, but when it does occur the execution time is very short. Successive attempts eventually succeed. One workaround is to measure the execution time of the PBIST algorithm. Using a software loop with interrupts disabled is sufficient. If the execution time is much shorter than the normal execution time and the status indicates PBIST passed, ignore the results and rerun the PBIST test. Normal execution time is dependent on clock speeds and which memory and algorithms are selected. The normal execution time can be derived from PBIST times given in the datasheet, or as measured in initial code development.

TI recommends to use 120% of the normal time as a time out value and less than 80% of the normal execution time as an indication that the PBIST controller did not execute properly.

A more sophisticated workaround is to use the errata_PBIST_4() function provided in HALCoGen version 3.08.00 or later.

SSWF021#44 *Change to PLL Lock Time*

Severity 4-Low

Expected Behavior The time for the PLL to lock would be the same for all revisions of a part.

Issue The PLL lock time has been increased on newer revision parts by 384 OSCIN cycles.

Condition None

Implication(s) The PLL takes longer to lock. If the software has a timeout loop waiting for the PLL to lock, software developed on this revision of silicon may timeout on future revisions of silicon.

Workaround(s) If there is a timeout loop in the software waiting for the PLL to lock, the timeout value should be large enough to allow for the greater lock time required in newer versions of the silicon. The lock time increases:

from 128 + 1024*NR OSCIN cycles

to 512 + 1024*NR OSCIN cycles

For typical PLL settings, the input divider is larger than 1 so that the percentage increase in lock time is small.

SSWF021#45 *PLL Fails to Start*

Severity 2-High

Expected Behavior When the PLL control registers are properly initialized and the appropriate clock source disable bit is cleared, after the prescribed number of OSCIN cycles, the PLL should be locked and the appropriate CSVSTAT bit should be set.

Issue On rare occasions the PLL does not start properly. The fail has one of three signatures:

1. CSVSTAT is set, but the ESM flag for PLL slip is set.
2. CSVSTAT is not set and the ESM flag for PLL slip is set.
3. CSVSTAT is set, the ESM flag for PLL slip is not set, but the PLL as measured by the DCC is not running.

Condition This issue applies to both PLLs (if the device has more than one PLL). This condition occurs only from a power-on. Once the PLL has locked, the PLL stays locked. Once properly locked, the PLL can be disabled and re-enabled with no issues.

Implication(s) If the PLL is used as the main clock source when it has not properly started, the CPU may stop executing instructions.

Workaround(s) While the main clock is being driven by the oscillator, the software loop checking that the PLL has locked (CSVSTAT = 1) should also check if the ESM flag for PLL slip has been set. When the CSVSTAT bit is set, the PLL frequency should be measured with the DCC before using the PLL as a clock source. If either the ESM flag for PLL slip is set, or the PLL has an incorrect frequency, the PLL should be disabled and the lock procedure should be repeated; TI recommends allowing a minimum of five attempts.

A more detailed explanation of the workaround with associated source code can be found in the application note:

Hercules PLL Advisory SSWF021#45 Workaround

STC#26 *The value programmed into the Self Test Controller (STC) Self-Test Run Timeout Counter Preload Register (STCTPR) is restored to its reset value at the end of each self test run.*

Severity 4-Low

Expected Behavior Once the Self-Test Run Timeout Counter Preload Register (STCTPR) is written, the value written into the register will be maintained until it is overwritten or a system or power on reset occurs and it will be used to preload the timeout counter for each self test run.

Issue The STCTPR is reset to the reset default value (0xFFFFFFFF) at the end of each CPU self test run and the value previously written to the STCTPR register is lost.

Condition Execution of any CPU self test with a STCTPR value other than the default value (0xFFFFFFFF).

Implication(s) Subsequent self test runs will use a maximum timeout value of 0xFFFFFFFF if not re-written to the desired value.

Workaround(s) The Timeout preload value in STCTPR register needs to be programmed to the required time out value before starting each self test if a timeout count other than 0xFFFFFFFF is desired.

STC#28 *CPU can hang if a system reset [internal or external] is asserted while the CPU self test is in progress*

Severity 4-Low

Expected Behavior If a system reset (internal or external) is asserted, the CPU should be reset cleanly and restart execution once the reset is released.

Issue The CPU may inadvertently enter debug mode.

Condition This condition can occur when a system reset (internal or external) is asserted and the CPU is executing a CPU self test.

Implication(s) Entering debug mode will cause the CPU to stop executing (hang) on the first instruction after reset.

Workaround(s) None

STC#29 *Inadvertent Performance Monitoring Unit (PMU) interrupt request generated if a system reset [internal or external] occurs while a CPU Self-Test is executing.*

Severity 4-Low

Expected Behavior If an internal or external system reset is asserted the CPU should be reset cleanly with no inadvertent interrupt requests.

Issue An unexpected PMU interrupt request may be generated.

Condition This condition can occur when am internal or external system reset is asserted and the CPU is executing a CPU self test.

Implication(s) The interrupt request signal from the performance monitoring unit (PMUIRQ) may inadvertently be set. This signal will generate an interrupt to the Vector Interrupt Module (VIM) and later become an interrupt to the CPU. Therefore, it is possible to see an unexpected interrupt after the CPU comes out of the system reset.

Workaround(s) Clear VIM interrupt request 22 by writing 0x00400000 to location 0xFFFFFE20 before enabling this interrupt.

STC#30 *Self Test Controller Fails to Execute Properly*

Severity 2-High

Expected Behavior The STC tests the logic of the CPUs then generates a CPU reset on completion

Issue The STC may hang, causing the CPUs to stop executing instructions. No CPU reset is generated. The root cause is that the ROM in the STC is not read properly. This is the same root cause as in erratum PBIST#4.

Condition This only occurs after a power-on. If the STC powers on properly it will execute properly.

Implication(s) The CPU will hang and not respond to interrupts.

Workaround(s) Use the function "errata_PBIST_4()" provided in HALCoGen 3.08.00 or later before running PBIST or LBIST. Then run the PBIST ROM tests on all ROMs before using the STC controller to execute LBIST.

VIM#28 *VIM module does not return corrected data to the CPU when a single bit error is detected in the VIM RAM*

Severity 2-High

Expected Behavior In hardware vectored interrupt mode the CPU will read the vector address from the IRQVECADDR bus of the VIC port. The IRQVECADDR is provided by the VIM module. In software vectored mode, the CPU will read the vector address from the IRQVECREG register in the VIM Module. The VIM module performs ECC correction/detection when reading the vector address from the VIM RAM and loads the corrected vector address onto the IRQVECADDR bus and into the IRQVECREG register if a single bit error is detected.

Issue When a single bit error is detected on the VIM RAM, the VIM does not load the corrected data onto the IRQVECADDR bus and into the RQVECREG register. However, if the CPU directly reads from the VIM RAM, the VIM will return the corrected data to the CPU if a single error is detected.

Condition When an interrupt is generated in either hardware vectored interrupt or software vectored interrupt mode.

Implication(s) When there is a single bit error in the vector address, the un-corrected vector address can cause the CPU to jump to an incorrect address. CPU behavior can be unpredictable.

Workaround(s) Set the "PAR_ECC_CTL.EDAC_MODE" register field to "DETECTION-ONLY" mode. This causes a single bit error to generate an uncorrectable error and causes the VIM to revert to the FALL_BACK ADDRESS for the interrupt vector. A single bit error can be treated as an uncorrectable error, or the FALL_BACK routine can determine the correct vector of the interrupt routine by following these steps:

- In the FALL_BACK routine, read the address in the uncorrectable error address register, UERRADDR to determine the address of the single or multi-bit error.

- Using the address of the VIM ram error, retrieve the vector address that was originally loaded into the VIM RAM (usually from a table in flash)

- Service the interrupt

- Clear the UERR bit in ECCSTAT register by writing a 1 to it.

- Optionally rewrite the correct vector address to the VIM RAM (to correct any soft error).

This procedure will work for single-bit or multi-bit errors.

4 Revision History

This silicon errata revision history highlights the technical changes made from the previous to the current version of this document.

Table 3. Revision History

Advisory Changes in Advisory List	Advisory ID
Added advisory(s)	SSWF021#45
Removed advisory(s)	None
Modified advisory(s)	None
Other	None

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 as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

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