RM57Lx Microcontroller

Silicon Revision B

Silicon Errata

Literature Number: SPNZ233B
August 2015–Revised June 2018

Contents

List of Figures

List of Tables

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 MCU devices. Each MCU commercial family member has one of three prefixes: X, P, or NULL [blank] (for example, xRM48L952). These prefixes represent evolutionary stages of product development from engineering prototypes (X) through fully qualified production devices (NULL[blank]).

Device development evolutionary flow:

X — Experimental device that is not necessarily representative of the final device's electrical specifications.

P — Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification.

NULL — Fully-qualified production device.

X and P devices are shipped against the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

Production 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 (X or P) 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 an example of the RM57Lx device markings. The device revision can be determined by the symbols marked on the top of the device.

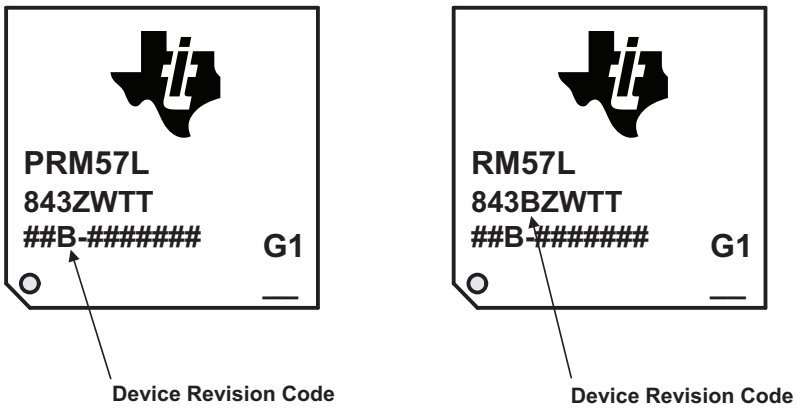

Figure 1. Device Revision Code Identification

Silicon revision is identified by a device revision code. The code is of the format RM57L843x, where "x" denotes the silicon revision. If x is "A" in the device part number, it represents silicon revision A. If x is "B" in the device part number, it represents silicon revision B and so on.

3 Silicon Changes from Previous Device Revision

The following errata have been fixed going from silicon revision A to silicon revision B. For a description of these errata see the Silicon Revision A errata document SPNZ214.

Table 1. Errata Which have been Fixed

Erratum	Considerations When Migrating From Revision A to Revision B Silicon
DEVICE#37	If AJSM is used, may need to update lock/unlock code
DEVICE#39	May leave workaround in place, no change to software required.
DEVICE#41	May leave workaround in place, no change to software required.
DEVICE#42	May leave workaround in place, no change to software required.
DEVICE#45	May leave workaround in place, no change to software required.
DEVICE#46	Software should include handler for Live-Lock event. Previous workarounds will still work on fixed silicon.
DEVICE#55	May leave workaround in place, no change to software required.
DEVICE#57	None
DEVICE#59	May leave workaround in place, no change to software required.
DMAOCP#01	None
LPO#16	May leave workaround in place, no change to software required
PBIST#4	May leave workaround in place, no change to software required.
SSWF021#44	Evaluate impact of additional 384 OSCIN cycles to PLL lock time.
STC#28	None
STC#29	May leave workaround in place, no change to software required.
STC#30	May leave workaround in place, no change to software required.
VIM#28	May leave workaround in place, no change to software required.

4 Known Design Exceptions to Functional Specifications

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

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

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.

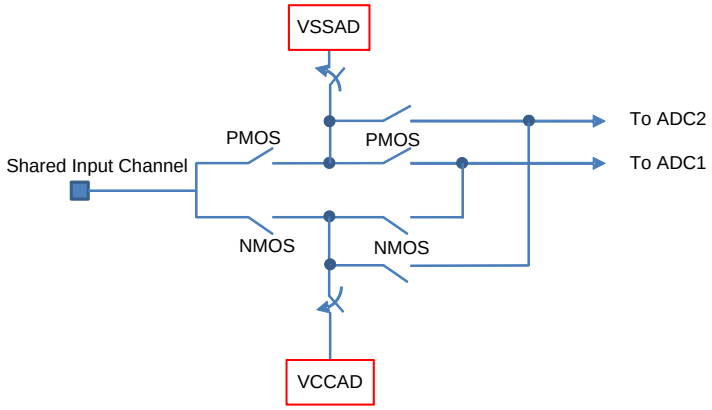

Figure 2. Shared Input Channel in "Open" State

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.

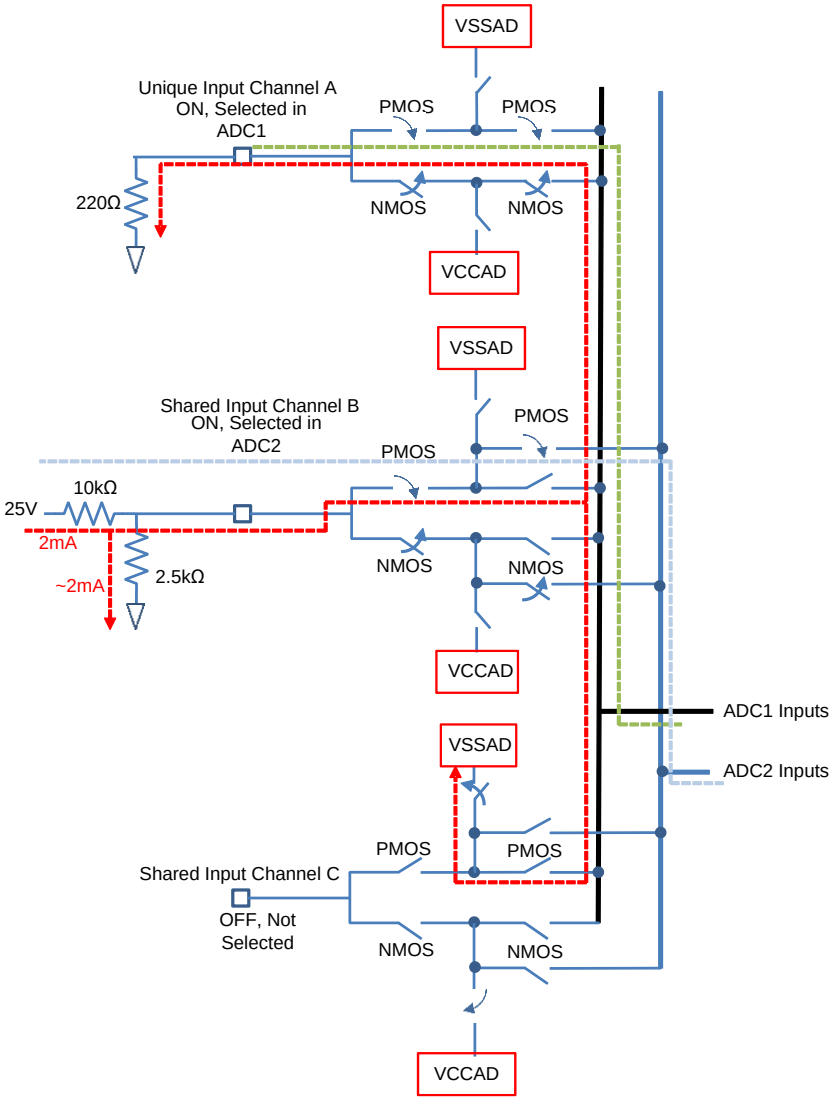

Figure 3. Example ADC1/ADC2 Channel Connection

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#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.

DCAN#27 *During DCAN FIFO Mode, received messages may be placed out of order in the FIFO buffer*

Severity 3-Medium

Expected Behavior In DCAN FIFO mode, the received messages with the same arbitration and mask IDs are supposed to placed in the FIFO in the order which they are received. The CPU will then retrieve the received messages from the FIFO via the IF1/IF2 interface registers.

Issue Some messages may be placed in the FIFO out of the order in which they were received. If the order of the messages is critical to the application for processing, then this behavior will prevent the proper use of the DCAN FIFO mode.

Condition DCAN operates in FIFO mode and uses the CPU to read out the data.

Implication(s) Application can not reliably use DCAN FIFO mode.

Workaround(s) Use the DMA to read out the FIFO via the IF3 register. Each time a message is received into the FIFO the data is also copied to the IF3 register and a DMA request is generated to the DMA module to read out the data.

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

Severity 3-Medium

Expected Behavior The STC self test should be able to be run on any one interval independently.

Issue STC test cannot be run on interval 1 independently.

Condition Run STC test on interval 1 alone.

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

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

DEVICE#48 *Interconnect Global Error flag is set after a CPU reset*

Severity 3-Medium

Expected Behavior There should be no error generated by a CPU reset.

Issue The Interconnect Global Error (ESM group 1, channel 52) is set intermittently after a CPU reset that is not also a system reset.

Condition The system clock GCLK is faster than the maximum rated speed for HCLK and a CPU only reset is generated. A CPU reset is generated at the end of the STC test or when the CPU writes to a 1 to bit 0 of the CPU Reset Control Register (CPURSTCR)

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#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#60 *nERROR can not be cleared by writing 0x5 to ESMEKR upon system reset*

Severity 4-Low

Expected Behavior Once the ERROR pin outputs low, a write of 0x5 to ESMEKR should release the ESM error pin back to normal state.

Issue The nERROR occurs and is not cleared before the system reset. After system reset, writing 0x5 to ESMEKR can not clear the nERROR to restore the ESM back to normal state.

Condition When nERROR occurs (the nERROR pin outputs low), write 1 to the corresponding bit of ESM status register (ESMSR2, or ESMSR3) to clear the error flag, then reset the device by pushing the nRST button on the PCB board or programming SYSECR[15:14].

Implication(s) The nERROR pin is not cleared by writing 0x5 to ESMEKR, and the ESM must be switched into the error forcing mode before the nERROR is cleared.

Workaround(s) If there is a system reset, the ESM state machine is also reset. It lost the fact that an error event had just happened. So if you now try to reset the nERROR pin by writing 0x5 it will not work. The way to clear the nERROR pin is either through a power on reset or by putting the nERROR pin in diagnostic mode by writing 0xA to the ESMEKR register followed by any non 0xA values to reset the nERROR pin.

EMIF#3 *EMIF generates data abort on register read after time-out error*

Severity 3-Medium

Expected Behavior The EMIF should not cause an abort when accessing EMIF registers.

Issue After an EMIF time-out error when an external asynchronous memory fails to respond, a read to an EMIF register generates data abort.

Condition 1. The EMIF is used for asynchronous memory accesses in Extended Wait mode.

2. A time-out error occurs. For example, the memory does not de-assert the EMIF_nWAIT input.

3. The asynchronous memory access with time-out error is followed by an EMIF register read.

Implication(s) Aborts will be generated on EMIF register reads until the "time-out" status is corrected by a successful EMIF region read.

Workaround(s) If a timeout error occurs, complete a dummy read from the EMIF memory that does not return an error. This can be a synchronous read, a read from another asynchronous chip select that is not configured to be in Extended Wait mode, or to the same asynchronous chip select after disabling the Extended Wait mode on that chip select.

EMIF#5 *SDRAM Initialization Delay Time is Less Than Expected*

Severity

3 - Medium

Expected Behavior

SDRAM devices require a delay of 100 to 200 µs after power-up but, before initialization; during this time, the EMIF must provide a stable clock and either a NOP or COMMAND INHIBIT command to the SDRAM. After this start-up delay, the SDRAM can be configured with a sequence of AUTO_REFRESH commands followed by a LOAD_MODE_REGISTER command and finally a refresh cycle (a PRECHARGE command followed by a REFRESH command). This initialization sequence, from the initial start-up delay to the final refresh cycle, is automatically created by the EMIF when any of the lower 3 bytes of the SDRAM Configuration Register (SDCR) are written.

The Refresh Rate (RR) field of the SDRAM Refresh Control Register (SDRCR) is used by the EMIF SDRAM controller to create the start-up delay. Specifically, the EMIF SDRAM controller should issue 8*RR NOP instructions before the first AUTO_REFRESH command.

Issue

Instead of issuing 8*RR NOP commands during initialization, the EMIF SDRAM controller only issues 1*RR NOP commands.

Conditions

This problem is a fundamental limitation in the EMIF module.

Implications

The impact is dependent on the particular SDRAM device that is being used with the EMIF module. In many cases there will be no apparent issue even when the SDRAM specification is violated. Therefore, check and, if necessary, correct this problem even if no SDRAM-related issues have been observed.

There are two possible implications to the delay of 1*RR instead of 8*RR:

- For systems operating with an EMIF clock frequency < 40 MHz, the RR counter value computed by software may be too small if the calculation was originally based on an 8*RR EMIF clock cycle delay.
- For systems operating with an EMIF clock frequency ≥ 40 MHz, the RR counter value does not have the range necessary to create a 200-µs delay.

Workaround(s)

- For systems operating with an EMIF clock frequency < 40 MHz, recalculate the correct RR value based on a 1*RR EMIF clock cycle delay. Update the SDRAM initialization code with the corrected value.
- For systems operating with an EMIF clock frequency ≥ 40 MHz, initialize the SDRAM while the EMIF clock source is still < 40 MHz. This means that the SDRAM should be initialized while the device is still operating from the OSCIN clock source, before it switches to the faster PLL-based clock source.

For a system operating with an EMIF clock frequency ≥ 40 MHz, the SDRAM should be initialized as part of the device initialization code and in this order relative to the PLL and clock tree configuration:

1. Configure the PLL so that it begins to lock onto its final operating frequency. Leave the clock source for the device in the default (OSCIN) setting. The PLL output divider should be set to the maximum divide value (/32) when programming PLL Control Register 1 (PLLCTL1).
2. Enable the EMIF_CLK pin as an output. This pin should output a clock with frequency OSCIN/2 based on the default clock divider settings.
3. Configure the EMIF SDRAM Timing Register (SDTIMR). Use the final EMIF clock frequency (≥ 40 MHz) to calculate the values for the fields of the SDTIMR as described in the device technical reference manual. (Do not use the frequency OSCIN/2 when computing these values.)

4. Configure the EMIF SDRAM Self Refresh Exit Timing Register (SDSRETR) also based on the final (≥ 40 MHz) EMIF clock frequency.
5. Configure the EMIF SDRAM Refresh Control Register (SDRCR) with a value of RR that produces the delay required by the SDRAM device (typically 200 µs). For example, with a 16-MHz OSCIN frequency, the default frequency of the EMIF clock will be 8 MHz, with a period of 125 ns. At this frequency, an RR count of 1600 will produce an initialization delay of 200 µs.
6. Write to the SDRAM Configuration Register (SDCR) with values that match the parameters of the SDRAM device. Use a single 32-bit write to ensure that the EMIF begins the SDRAM initialization sequence after the write is complete.
7. Read from the SDRAM at any address (for example, the base address of 0x80000000).
8. The EMIF automatically stalls the CPU until it completes the initialization process and performs the requested read.
9. Reprogram the SDRCR with the correct RR value for runtime operation. Use the final (≥ 40 MHz) EMIF clock frequency when computing this value. Refer to the technical reference manual for instructions on computing this value.
10. Write a value of 0x80 to the most significant byte (bits 31:24) of the SDCR. Make sure that a **byte-write** is performed and that only the most significant byte of the SDCR is written. Avoid writing to any other byte of the SDCR so that a new initialization sequence is not started as a side effect of the write.
11. Ensure that no other reads or writes of the EMIF are performed until Step 12, Step 13, and Step 14 are complete.
12. Switch the device clock source to the PLL. In particular, the Primary System Module GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC).
13. Program the desired clock divider values into the Primary System Module Clock Control Register (CLKCNTL).
14. Step up the PLL output frequency to its final value by reducing the PLLDIV value in the System Module PLL Control Register 1 (PLLCTL1).

With the device now operating with the final (≥ 40 MHz) EMIF clock frequency, any access to the SDRAM will bring the SDRAM out of the self refresh state and normal SDRAM activity may begin.

This list consists of only the steps that are related to proper SDRAM initialization; it is not a comprehensive list of device initialization steps. Many other steps in the device initialization can be performed before Step 1 and between Step 1 and Step 2. However, Step 2 through Step 14 should be performed as a sequence without introducing any other initialization steps between them. Following this recommendation reduces the risk of unrelated EMIF activity (including activity on the asynchronous chip selects) during the critical time when the SDRAM is in the self-refresh (SBY) state and the EMIF clock is being switched.

Figure 4 shows EMIF activity as the preceding device initialization sequence is performed, assuming an OSCIN frequency of 16.0 MHz and a PLL frequency of 180 MHz.

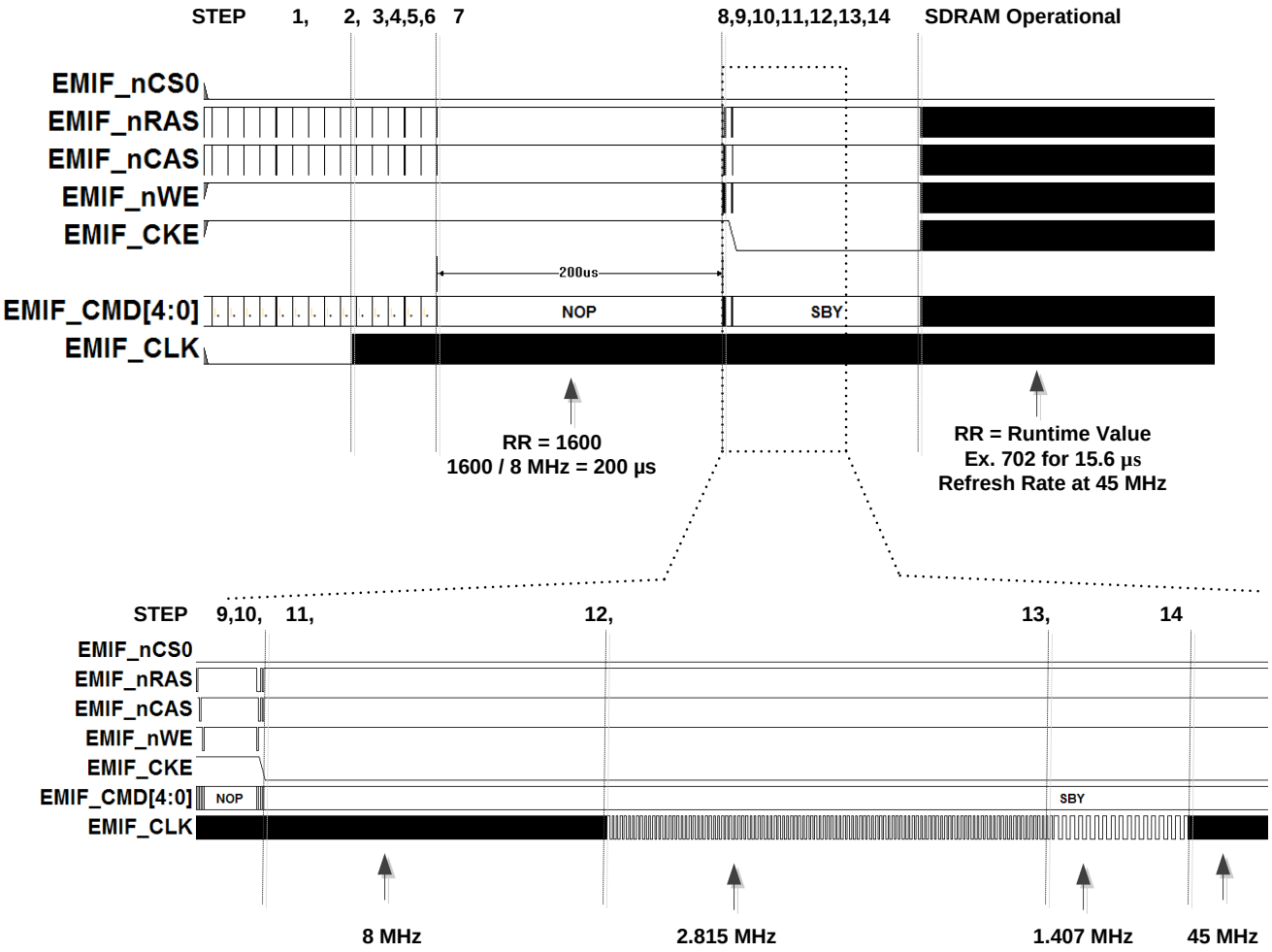

Figure 4. Correct SDRAM Initialization Procedure for Final EMIF Clock Frequency ≥ 40 MHz

NOTE: EMIF_CMD[4:0] is a bus consisting of [EMIF_nCS0, EMIF_nWE, EMIF_CKE, EMIF_nRAS, EMIF_nCAS].

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.

GCM#60 *VCLK at HCLK/2 Instead of HCLK/1*

Severity 3-Medium

Expected Behavior When using a clock configuration of HCLK = VCLK2 = VCLK, a write to VCLK2R field of CLKCNTL to zero, then a read of CLKCNTL and a write of the VCLKR field to zero will set the proper clock ratios.

Issue Sometimes, the VCLKR field of CLKCNTL changes to zero, but the VCLK frequency remains HCLK/2.

Condition This happens only after an externally asserted nRST. It does not occur after an nPORRST or an internally generated nRST such as from a software reset, internal watchdog reset or oscillator fault reset. It is not an issue when the final VCLK ratio is VCLK = HCLK/2.

Implication(s) The VCLK frequency will be HCLK/2 instead of equal to HCLK/1.

Workaround(s) When using HCLK=VCLK2=VCLK, set the VCLK2R field = 0 first, then set VCLKR = 0, then VCLKR = 1, and finally VCLKR = 0 again. The sequence should be done with a read of CLKCNTL between each write to allow additional clocks for synchronization. The following C code is shown as an example:

```
systemREG1->CLKCNTL  = (systemREG1->CLKCNTL & 0xF0FFFFFFU) | (uint32)((uint32)0U << 24U);

systemREG1->CLKCNTL  = (systemREG1->CLKCNTL & 0xFFF0FFFFU) | (uint32)((uint32)0U << 16U);

systemREG1->CLKCNTL  = (systemREG1->CLKCNTL & 0xFFF0FFFFU) | (uint32)((uint32)1U << 16U);

systemREG1->CLKCNTL  = (systemREG1->CLKCNTL & 0xFFF0FFFFU) | (uint32)((uint32)0U << 16U);
```

L2FMC#5 *Incorrect data read from flash ECC data memory region, flash OTP memory region, or data flash memory region when configured as "normal" type memory*

Severity 3-Medium

Expected Behavior Reads from flash ECC data memory region, flash OTP memory region, or data flash memory region return correct data regardless of which memory type the target location is configured as.

Issue The flash interface module has pipeline logic to determine whether an address being accessed has already been prefetched into the local pipeline. This logic only compares bits 21:5 of the addresses instead of the full 31:5 address bits. This causes an incorrect pipeline hit identification inside the flash interface module upon accesses to the Flash data space ECC memory, flash OTP memory, and the data flash memory regions. In this case, the read returns the instruction opcode instead of the actual data from the memory being accessed. The issue occurs when an instruction accesses a location in flash data ECC memory, or flash OTP memory, or data flash bank memory such that the address bits 21:5 of the instruction being executed and the target memory location are the same.

Condition - The prefetch buffer is enabled inside the flash interface module

- The memory being accessed: flash data ECC memory or flash OTP memory or data flash bank memory, is configured to be "normal" type

- Address bits 21:5 of instruction used to access these regions and the address bits 21:5 of location being accessed are matching

Implication(s) The data being read is not correct.

For example, if an instruction at address 0x00080310 reads a location 0xF0080310, the value returned will be the instruction at address 0x00080310 instead of the data at address 0xF0080310.

Workaround(s) The issue only occurs when the memory location being accessed is configured as a "normal" type memory.

As a workaround, the application must configure the CPU's MPU to configure the flash data ECC memory region, the flash OTP memory region, and the data flash memory region to be either "device" type or "strongly ordered" type memory regions.

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};
```

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.

5 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)	DCAN#27, DEVICE#60, EMIF#3, EMIF#5, GCM#60, L2FMC#5
Removed advisory(s)	CORTEX-R5#5: does not apply to lockstep R5 implementation
Modified advisory(s)	None
Other	None

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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