

RM46x Microcontroller

Silicon Revision A

Silicon Errata



Literature Number: SPNZ187
September 2012

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

List of Figures

1	Part Marking	5
---	--------------------	---

List of Tables

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

RM46x Microcontroller

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

1 Device and Development-Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all devices and support tools. Each device has one of three prefixes: X, P, or null (no prefix).

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."

X and P 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.

TI device nomenclature also includes a suffix with the device family name. This suffix indicates the silicon revision, the package type and the temperature range (for example, "T" is -40 to 105C).

2 Device Markings

[Figure 1](#) provides examples of the RM46Lx device markings.

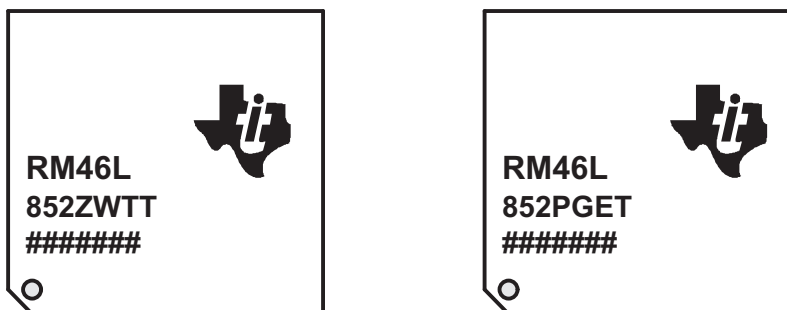


Figure 1. Part Marking

[Table 1](#) shows the applicable silicon revision.

Table 1. Device Revision Codes

DEVICE PART NUMBER DEVICE REVISION CODE	SILICON REVISION	PART NUMBERS/COMMENTS
RM46L852	A	This silicon revision is available as A <i>only</i> . RM46L852

3 Known Design Exceptions to Functional Specifications

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

Table 2. Known Design Exceptions to Functional Specifications

Title	Page
AHB_ACCES_PORT#3 (ARM ID-529470) — DAP AHB-AP Access Port Might Fail To Return Slave Error On AHB Reset.	7
BCG_IP_P_EMIF#241 (EMIF#241) — EMIF Reports Incorrect Time-out Error Data	8
CORTEX-R4#26 (ARM ID-577077) — Thumb STREXD Treated As NOP If Same Register Used For Both Source Operands	9
CORTEX-R4#27 (ARM ID-412027) — Debug Reset Does Not Reset DBGDSCR When In Standby Mode	10
CORTEX-R4#33 (ARM ID-452032) — Processor Can Deadlock When Debug Mode Enables Cleared.....	11
CORTEX-R4#46 (ARM ID-599517) — CP15 Auxiliary ID And Prefetch Instruction Accesses Are UNDEFINED.....	12
CORTEX-R4#54 (ARM ID-639819) — Instruction Causing A Data Watchpoint Match Incorrectly Traced	13
CORTEX-R4#55 (ARM ID-722412) — CPACR.ASEDIS And CPACR.D32DIS Incorrect When Configured With Floating Point	14
CORTEX-R4#57 (ARM ID-737195) — Conditional VMRS APSR_Nzcv, FPSCR May Evaluate With Incorrect Flags	15
CORTEX-R4#58 (ARM ID-726554) — DBGDSCR.Adadiscard Is Wrong When DBGDSCR.Dbgack Set	16
CORTEX-R4#61 (ARM ID-720270) — Latched DTR-Full Flags Not Updated Correctly On DTR Access	17
CORTEX-R4#66 (ARM ID-754269) — Register Corruption During A Load-Multiple Instruction At An Exception Vector.....	18
CORTEX-R4#67 (ARM ID-758269) — Watchpoint On A Load Or Store Multiple May Be Missed.	19
DEVICE#B053 — CPU code execution could be halted on a device warm reset if the core power domain # 2 is disabled by software.	20
DEVICE#B056 — Error flags are incorrectly set in the Error Signaling Module upon a device warm reset for modules in core power domains that are disabled	21
DEVICE#B058 — Device I/Os Could Toggle After Warm Reset.....	22
DEVICE#142 — CPU Abort Not Generated on Write to Unimplemented MCRC Space	23
DEVICE#070 — Some multiplexing controls are not compatible with the emulation device	24
DMA#27 — DMA Requests Lost During Suspend Mode	25
ANALOGIP_021_LPO#16 (F021 LPO#16) — Oscillator Fault Detection Starts too Soon	26
MCRC#18 — CPU Abort Generated on Write to Implemented MCRC Space After Write to Unimplemented MCRC Space	27
MIBSPI#110 — Multibuffer Slave In 3 or 4 Pin Mode Transmits Data Incorrectly for Slow SPICLK Frequencies and for Clock Phase = 1.....	28
MIBSPI#111 — Data Length Error Is Generated Repeatedly In Slave Mode when I/O Loopback is Enabled	29
NHET#52 — WCAP May not Capture the High-Resolution Offset Correctly	30
NHET#53 — Enhanced Input Capture Pins May not Capture Small Pulses Correctly.....	31
SYS#46 — Clock Source Switching Not Qualified With Clock Source Enable And Clock Source Valid	32
SYS#102 — Bit field EFUSE_Abort[4:0] in SYSTASR register is read-clear	33
SYS#111 — The Device may Generate Multiple nRST Pulses.	34
VIM#27 — Unexpected phantom interrupt	35

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

Severity	Medium
Expected Behavior	If an AHB reset occurs (HRESETn is asserted LOW) while the AHB-AP is performing an access on the AHB bus, the AHB-AP should return a slave error back to the JTAG-DP.
Issue	Instead, the AHB-AP might indicate that the access completed successfully and return unpredictable data if the access was a read.
Condition	HRESETn is asserted LOW on a specific cycle while the AHB-AP is completing an access on the AHB bus.
Implication(s)	This will make it harder to identify cases where the AHB bus is being reset. However, this should not affect most usage models because it is not generally possible to debug over a bus when it is likely to be reset.
Workaround(s)	Avoid performing debug operations while the AHB bus might be reset. Resets usually occur following powerdown, and so this can usually be achieved through the use of features that prevent powerdown during debug.

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

Severity	Medium
Expected Behavior	
Issue	Under certain conditions, the EMIF will report a time-out error (vbusp_rstatus=0x3) on the VBUSP read status interface. The data returned for this access will be all zeros. If this access is followed by a read to the EMIF's MMR space, the EMIF will still report a time-out error (vbusp_rstatus=0x3) but with the correct data for the MMR read.
Condition	<p>This issue is only applicable if all of the following are true:</p> <ol style="list-style-type: none"> 1. The EMIF is used for async memory accesses in Extended Wait mode. 2. There is a potential for a time-out error to occur. For example, the async memory will not de-assert the pad_wait_i input. 3. If an async memory read with time-out error is followed by an MMR read. 4. The EMIF's vbusp_rstatus output is used by the infrastructure to zero out the read data when time-out error is reported.
Implication(s)	The EMIF will hold vbusp_rstatus=0x3 until another async access without time-out error or an SDRAM access is performed.
Workaround(s)	If a timeout error occurs, perform a dummy read to SDRAM, or to another async chip select that is not configured to be in Extended Wait mode, or to the same async chip select after disabling the Extended Wait mode on that chip select.

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

Severity	Medium
Expected Behavior	
Issue	The ARM Architecture permits the Thumb STREXD instruction to be encoded with the same register used for both transfer registers (Rt and Rt2). Because of this erratum, the Cortex-R4 processor treats such an encoding as UNPREDICTABLE and executes it as a NOP.
Conditions	This occurs when the processor is in Thumb state and a STREXD instruction is executed which has Rt = Rt2. This instruction was new in ARM Architecture version 7 (ARMv7). It is not present in ARMv6T2 or other earlier architecture versions.
Implications	If this occurs the destination register, Rd, which indicates the status of the instruction, is not updated and no memory transaction takes place. If the software is attempting to perform an exclusive read-modify-write sequence, then it might either incorrectly complete without memory being written, or loop forever attempting to complete the sequence.
Workaround(s)	This can be avoided by using two different registers for the data to be transferred by a STREXD instruction. This may involve copying the data in the transfer register to a second, different register for use by the STREXD.

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

Severity	Medium
Expected Behavior	The debug reset input, PRESETDBGn, resets the processor's debug registers as specified in the ARMv7R Architecture. The debug reset is commonly used to set the debug registers to a known state when a debugger is attached to the target processor.
Issue	When the processor is in Standby Mode and the clock has been gated off, PRESETDBGn fails to reset the Debug Status and Control Register (DBGDSCR).
Conditions	<ul style="list-style-type: none"> • The DBGDSCR register has been written so that its contents differ from the reset values (most fields in this register reset to zero, though a few are UNKNOWN at reset) • The processor is in Standby Mode, and the clocks have been gated off, that is STANDBYWFI is asserted • The debug reset, PRESETDBGn, is asserted and deasserted while the processor clocks remain gated off
Implications	If this occurs, then after the reset the DBGDSCR register contains the values that it had before reset rather than the reset values. If the debugger relies on the reset values, then it may cause erroneous debug of the processor. For example, the DBGDSCR contains the ExtDCCmode field which controls the Data Communications Channel (DCC) access mode. If this field was previously set to Fast mode but the debugger assumes that it is in Non-blocking mode (the reset value) then debugger accesses to the DCC will cause the processor to execute instructions which were not expected.
Workaround(s)	This can be avoided by a workaround in the debug control software. Whenever the debugger (or other software) generates a debug reset, follow this with a write of zero to the DBGDSCR which forces all the fields to their reset values.

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

Severity	Medium
Expected Behavior	The Cortex-R4 processor supports two different debugging modes: Halt-mode and Monitor-mode, or both modes can be disabled. Bits [15:14] in the Debug Status and Control Register (DBGDSCR) control which, if any, mode is enabled. Debug events can be generated by the breakpoint unit, matching instructions executed. Deadlocks should not occur when the debug mode is changed.
Issue	If there are active breakpoints at the time when the debugging mode is changed, this can cause the processor to deadlock.
Conditions	<ul style="list-style-type: none"> • At least one breakpoint is programmed and active • Either halt-mode debugging or monitor mode debugging is enabled and the debug enable input, DBGEN is HIGH • An instruction which matches a breakpoint is fetched and executed • After the instruction is fetched, but before it has completed execution, both halt-mode and monitor-mode debugging are disabled by means of a write to DBGDSCR
Implications	Changing the debugging mode while breakpoints or watchpoints are active is UNPREDICTABLE. Therefore the above conditions cannot be met in normal (recommended) operation of the processor. However, if the conditions do occur, then the processor will deadlock, which is outside the bounds of permitted UNPREDICTABLE behavior.
Workaround(s)	None

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

Severity	Medium
Expected Behavior	<p>The ARMv7-R architecture requires implementation of the following two features in CP15 :</p> <ol style="list-style-type: none"> 1. An Auxiliary ID Register (AIDR), which can be read in privileged modes, and the contents and format of which are IMPLEMENTATION DEFINED. 2. The operation to prefetch an instruction by MVA, as defined in the ARMv6 architecture, to be executed as a NOP.
Issue	CP15 accesses to Auxiliary ID Register (AIDR) or an operation to prefetch an instruction by MVA will generate an UNDEFINED exception on Cortex-R4.
Conditions	<p>Either of the following instructions is executed in a privileged mode:</p> <ul style="list-style-type: none"> • - MRC p15,1,<Rt>,c0,c0,7 ; Read IMPLEMENTATION DEFINED Auxiliary ID Register • - MCR p15,0,<Rt>,c7,c13,1 ; NOP, was Prefetch instruction by MVA in ARMv6
Implications	If software attempts to read the AIDR as part of its process of identifying the processor on which it is running, it will encounter an unexpected UNDEFINED exception. If software attempts to execute an instruction prefetch using the ARMv6 CP15 prefetch operation, it will encounter an unexpected UNDEFINED exception.
Workaround(s)	In the first case, because the contents of the AIDR are IMPLEMENTATION DEFINED, it must always be read in conjunction with the Main ID Register (MIDR). A simple way of avoiding this would be to read and analyze the MIDR first and, if this indicates that the processor is Cortex-R4 skip the read of the AIDR. In the second case, any instruction to prefetch an instruction by MVA should be removed or replaced by a true NOP instruction.

CORTEX-R4#54 (ARM ID-639819) *Instruction Causing A Data Watchpoint Match Incorrectly Traced*

Severity	Medium
Expected Behavior	When tracing a program execution using the ETM, an extra instruction should not be traced.
Issue	When tracing a program execution using the ETM, an extra instruction is traced if a data watchpoint matches and causes a debug exception. The extra instruction that is traced is the instruction which caused the data watchpoint to match.
Conditions	<p>The extra instruction is traced if the following occurs:</p> <ul style="list-style-type: none"> • A hardware watchpoint matches • DBGEN is asserted • Monitor debug-mode is enabled
Implications	Trace analysis tools will incorrectly consider the instruction causing a data watchpoint match to have executed. If any of the ETM address comparators are configured to match on address of the instruction and the exact match bit is set, the comparator will incorrectly fire. This might cause an unexpected trigger or change in any ETM resources which are configured to be sensitive to the address comparator.
Workaround(s)	If a data abort exception is taken and the cause was a data watchpoint, the instruction traced immediately before the entry to the exception handler was not executed and must be discarded.

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

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

Severity	Medium
Expected Behavior	
Issue	In implementations of the VFPv3-D16 architecture that do not also implement the Advanced SIMD functionality, the ASEDIS and D32DIS bits, (bits [31] and [30] respectively of the Coprocessor Access Control Register (CPACR)) are read-as-one/writes ignored (RAO/WI). This indicates that Advanced SIMD functionality and registers D16-D31 are disabled. Because of this erratum, these bits read zero in implementations of Cortex- R4F which include the floating-point unit. When the floating point unit is not included, all bits of the CPACR correctly read-as-zero (RAZ).
Conditions	In an implementation of Cortex-R4F with the floating-point unit included, the CPACR is read.
Implications	If software uses the CPACR to determine if Advanced SIMD functionality and registers D16-D31 are available, it will incorrectly determine that they are. Any attempt to use these features will lead to an unexpected UNDEFINED exception.
Workaround(s)	Because the Cortex-R4F processor never includes Advanced SIMD functionality or registers D16-D31, this can be correctly determined by reading the part number from one of the ID registers rather than examining ASEDIS and D32DIS in the CPACR.

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

Severity	Medium
Expected Behavior	A conditional VMRS APSR_nzcv, FPSCR instruction should evaluate its condition codes using the correct flags.
Issue	Under certain circumstances, a conditional VMRS APSR_nzcv, FPSCR instruction may evaluate its condition codes using the wrong flags and incorrectly execute or not execute.
Conditions	<p>The erratum requires the following sequence of instructions in ARMstate: - VMRS<c> APSR_nzcv, FPSCR (formerly FMSTAT<c>), where the condition on the instruction is not always.</p> <ul style="list-style-type: none"> • A flag-setting integer multiply or multiply and accumulate instruction (e.g. MULS) • A single-precision floating-point multiply-accumulate (FP-MAC) instruction (e.g. VMLA), timed such that the accumulate operation is inserted into the pipeline in the cycle in which the VMRS instruction is first attempted to be issued. <p>To meet the above timing requirements, the VMRS instruction must be three pipeline stages behind the FPMAC. Depending on the rate in which the instructions are fetched, interlocks within this sequence and dual-issuing, this can be up to three other instructions between this pair, plus the multiply. Out-of-order completion of FP-MAC instructions must be enabled.</p>
Implications	If this erratum occurs, the VMRS instruction will pass or fail its condition codes incorrectly, and this will appear in any trace produced by the ETM. This can corrupt the N, Z, C, V flag values in the CPSR which will typically affect the program flow.
Workaround(s)	This erratum can be avoided by disabling out-of-order single-precision floating point multiply-accumulate (SPMAC) instruction completion. Set DOOFMACS, bit [16] in the Secondary Auxiliary Control Register. This will have the side-effect of reducing the performance of SP-MAC operations, though the impact will depend on how these instructions are used in your code.

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

Severity	Medium
Expected Behavior	When the DBGDSCR.ADAdiscard bit is set, asynchronous data aborts are discarded, except for setting the DBGDSCR.ADAabort sticky flag. The Cortex-R4 processor ensures that all possible outstanding asynchronous data aborts have been recognised before it enters debug halt state. The flag is immediately on entry to debug halt state to indicate that the debugger does not need to take any further action to determine whether all possible outstanding asynchronous aborts have been recognized.
Issue	Because of this erratum, the Cortex-R4 processor also sets the DBGDSCR.ADAdiscard bit when the DBGDSCR.DBGack bit is set. This can cause the DBGDSCR.ADAabort bit to become set when the processor is not in debug halt state, and it is not cleared when the processor enters debug halt state. However, the processor does not discard the abort. It is pending or generates an exception as normal.
Conditions	<ul style="list-style-type: none"> • The processor is not in debug halt state • The DBGDSCR.DBGack bit is set • An asynchronous data abort (for example, SLVERR response to a store to Normal-type memory) is recognized <hr/> <p>NOTE: it is not expected that DBGDSCR.DBGack will be set in any Cortex-R4 system</p> <hr/> <p>If this erratum occurs, and the processor subsequently enters debug halt state, the DBGDSCR.ADAabort bit will be set, when in fact no asynchronous data abort has occurred in debug state. Before exiting debug state, the debugger will check this bit and will typically treat it as an error. If no other asynchronous data abort has occurred in debug state, this is a false error.</p>
Implications	
Workaround(s)	None

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

Severity	Medium
Expected Behavior	<p>When the debug Data Transfer Register (DTR) is in non-blocking mode, the latched DTR-full flags (RXfull_I and TXfull_I) record the state of the DTR registers as observed by the debugger and control the flow of data to and from the debugger to prevent race hazards. For example, when the target reads data from DBGDTRRXint, the associated flag RXfull is cleared to indicate that the register has been drained, but the latched value RXfull_I remains set. Subsequent debugger writes to DBGDTRRXext are ignored because RXfull_I is set. RXfull_I is updated from RXfull when the debugger reads DBGDSCRext such that a debugger write to DBGDTRRXext will only succeed after the debugger has observed that the register is empty. The ARMv7 debug architecture requires that RXfull_I be updated when the debugger reads DBGDSCRext and when it writes DBGDTRRXext. Similarly, TXfull_I must be updated when the debugger reads DBGDSCRext and when it reads DBGDTRTXext.</p>
Issue	<p>Because of this erratum, RXfull_I and TXfull_I are only updated when the debugger reads DBGDSCRext.</p>
Conditions	<p>The DTR is in non-blocking mode, that is, DBGDSCR.ExtDCCmode is set to 0b00 and EITHER:</p> <ul style="list-style-type: none"> • The debugger reads DBGDSCRext which shows that RXfull is zero, that is, DBGDTRRX is empty. • The debugger writes data to DBGDTRRXext. • Without first reading the DBGDSCRext, and before the processor has read from DBGDTRRXint, the debugger performs another write to DBGDTRRXext. <p>OR</p> <ul style="list-style-type: none"> • The debugger reads DBGDSCRext which shows that TXfull is one, that is, DBGDTRTX is full. • The debugger reads data from DBGDTRTXext, • The processor writes new data into DBGDTRTXint, • Without first reading the DBGDSCRext, the debugger performs another read from DBGDTRTXext.
Implications	<p>The ARMv7 debug architecture requires the debugger to read the DBGDSCRext before attempting to transfer data via the DTR when in non-blocking mode. This erratum only has implications for debuggers that violate this requirement. If the erratum occurs via data transfer, data loss may occur. The architecture requires that data transfer never occur.</p> <p>Texas Instruments has verified that TI's Code Composer Studios IDE is not affected by this issue.</p>
Workaround(s)	None

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

Severity	Medium
Expected Behavior	
Issue	Under certain circumstances, a load multiple instruction can cause corruption of a general purpose register.
Conditions	<p>All the following conditions are required for this erratum to occur:</p> <ul style="list-style-type: none"> • A UDIV or SDIV instruction is executed with out-of-order completion of divides enabled • A multi-cycle instruction is partially executed before being interrupted by either an IRQ, FIQ or imprecise abort. In this case, a multi-cycle instruction can be any of the following: <ul style="list-style-type: none"> – LDM/STM that transfers 3 or more registers – LDM/STM that transfers 2 registers to an unaligned address without write back – LDM/STM that transfers 2 registers to an aligned address with write back – TBB/TBH • A load multiple instruction is executed as the first instruction of the exception handler • The load multiple instruction itself is interrupted either by an IRQ, FIQ, imprecise abort or external debug halt request. This erratum is very timing sensitive and requires the UDIV or SDIV to complete when the load multiple is in the Issue stage of the CPU pipeline. The register that is corrupted is not necessarily related to the load-multiple instruction and will depend on the state in the CPU store pipeline when the UDIV or SDIV completes.
Implications	For practical systems, it is not expected that an interruptible LDM will be executed as the first instruction of an exception handler, because the handler is usually required to save the registers of the interrupted context. Therefore, it is not expected that this erratum has any implications for practical systems. If the situation of the erratum occurs it will result in the corruption of the register bank state and could cause a fatal failure if the corrupted register is subsequently read before being written.
Workaround(s)	To work around this erratum, set bit [7] of the Auxiliary Control Register to disable out-of-order completion for divide instructions. Code performance may be reduced depending on how often divide operations are used.

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

Severity	Medium
Expected Behavior	The Cortex-R4 supports synchronous watchpoints. This implies that for load and store multiples, a watchpoint on any memory access will generate a debug event on the instruction itself.
Issue	Due to this erratum, certain watchpoint hits on multiples will not generate a debug event.
Conditions	<p>All the following conditions are required for this erratum to occur:</p> <ul style="list-style-type: none"> • A load or store multiple instruction is executed with at least 5 registers in the register list • The address range accessed corresponds to Strongly-Ordered or Device memory • A watchpoint match is generated for an access that does not correspond to either the first two or the last two registers in the list. <p>Under these conditions the processor will lose the watchpoint. Note that for a "store multiple" instruction, the conditions are also affected by pipeline state making them timing sensitive.</p>
Implications	Due to this erratum, a debugger may not be able to correctly watch accesses made to Device or Strongly-ordered memory. The ARM architecture recommends that watchpoints should not be set on individual Device or Strongly-ordered addresses that can be accessed as part of a load or store multiple. Instead, it recommends the use of the address range masking functionality provided to set watchpoints on an entire region, ensuring that the watchpoint event will be seen on the first access of a load or store multiple to this region. If this recommendation is followed, this erratum will not occur.
Workaround(s)	None

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

DEVICE#B053	<i>CPU code execution could be halted on a device warm reset if the core power domain # 2 is disabled by software.</i>
Severity	Medium
Expected Behavior	The CPU code execution must start from the reset vector (address 0x00000000) upon a device warm reset and is not affected by the state of any switchable device power domain.
Issue	CPU code execution could be halted upon a warm reset if the core power domain # 2 has been disabled by software prior to the device warm reset.
Conditions	The behavior is not dependent on any particular operating condition.
Implications	CPU code execution is halted so that a system hang occurs. An external monitor must be present to prevent the system from entering an unsafe state when this happens.
Workaround(s)	The application must not disable the core power domain # 2 in software via the Power Management Module (PMM) registers, even if the modules inside this core power domain are not used in the application.

DEVICE#B056	<i>Error flags are incorrectly set in the Error Signaling Module upon a device warm reset for modules in core power domains that are disabled</i>
Severity	High
Expected Behavior	Once a core power domain is disabled (turned off) no error must be indicated to the ESM from a module in this power domain.
Issue	When a core power domain is disabled, a device warm reset causes error signals to be sent to the ESM that appear to be from the power domain that has been disabled. This is caused due to the incorrect power domain isolation implemented on these error signals. All such error signals are connected to ESM group1 channels.
Conditions	The behavior is not dependent on any particular operating condition.
Implications	If the application has enabled the generation of interrupts or the assertion of the nERROR output when any flag gets set in the ESM module's group1 status registers, then the CPU will receive an interrupt and/or the nERROR pin will be driven for an error condition ascribed to a module that is in a core power domain that has been disabled.
Workaround(s)	When a core power domain is disabled, the application must ensure that any error signal related to a module within this power domains is not configured to generate an interrupt to the CPU or assert the device nERROR output. Alternately, the application could clear the ESM group1 status flags upon each device warm reset condition before it enables interrupts and nERROR drive features for the ESM group1 channels.

DEVICE#B058	<i>Device I/Os Could Toggle After Warm Reset</i>
Severity	Medium
Expected Behavior	All device I/Os are tri-stated (inputs) after a device warm reset. No toggle is expected on any of the device I/Os without software configuration.
Issue	All core power domains get enabled on a system reset. After the core domains are turned ON, the domain outputs are unstable for a few cycles after the system reset is released. This could cause the I/Os to toggle.
Conditions	The behavior is not dependent on any particular operating condition.
Implications	Any external circuit connected to the device I/Os could see activity on the connections to the device after a devcie warm reset.
Workaround(s)	There are no software or hardware workarounds for this issue.

DEVICE#142	<i>CPU Abort Not Generated on Write to Unimplemented MCRC Space</i>
Severity	Low
Expected Behavior	A write to the illegal address region of the MCRC module will generate an abort
Issue	A CPU Abort does not get generated per the following condition:
Conditions	When a normal mode write to an illegal address region of MCRC register space is followed by a debug mode access.
Implications	When debugging, either a breakpoint on the instruction after the illegal write, or single stepping through the illegal write will not generate an abort.
Workaround(s)	None

DEVICE#070	<i>Some multiplexing controls are not compatible with the emulation device</i>
Severity	Medium
Expected Behavior	For the common functions, it is expected that the derivative parts will maintain compatibility to the superset emulation part.
Issue	<p>There are multiple issues with the multiplexing controls:</p> <ol style="list-style-type: none"> 1. For GIOB[2], to take input from the alternate path, the PINMMR29[16] has to be cleared. This is incompatible with the emulation device, where the PINMMR29[16] needs to be set to select the alternate path as input for GIOB[2]. 2. PINMMR29[24] is cleared by default to select MII interface for the EMAC. This is incompatible with the emulation device, where the PINMMR29[24] was set by default and selects the RMII interface.
Conditions	The behavior is not dependent on any particular operating condition.
Implications	The application must manage the configuration of the PINMMR registers based on the actual part being used. The code running on the emulation device cannot be run as-is on this device.
Workaround(s)	There are no software workarounds.

DMA#27	<i>DMA Requests Lost During Suspend Mode</i>
Severity	Medium
Expected Behavior	
Issue	DMA requests from a peripheral can be lost.
Condition	<p>This only happens when:</p> <ul style="list-style-type: none"> • The device is suspended by a debugger • A peripheral continues to generate requests while the device is suspended • The DMA is setup to continue the current block transfer during suspend mode with DMA DEBUG MODE set to '01' • And the request trigger type TTYPE is set to frame trigger
Implication(s)	When the DMA comes out of the suspend mode to resume the transfer, the data transfers corresponding to the third and subsequent requests will be lost.
Workaround(s)	<p>Either use TTYPE = Block transfer when DMA DEBUG MODE is '01' (Finish Current Block Transfer) or use DMA DEBUG MODE = '00' (Ignore suspend) when using TTYPE = Frame transfer to complete block transfer even after suspend/halt is asserted.</p>

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

Severity	Low
Expected Behavior	The oscillator fault detection circuit allows approximately 200ms after the release of nPORRST for the oscillator to become valid before monitoring for oscillator faults.
Issue	Sometimes the oscillator fault detection starts monitoring the oscillator shortly (approximately 100us) after the release of nPORRST.
Condition	This occurs only after power on.
Implication(s)	If the oscillator is slow to start, an oscillator fault condition may be detected and the device will switch to using the HFLPO as a clock source.
Workaround(s)	<p>This condition is avoided if nPORRST is held low long enough for the oscillator to stabilize.</p> <p>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. Refer to the device Technical Reference Manual for information on how to recover from an oscillator failure.</p>

MCRC#18	<i>CPU Abort Generated on Write to Implemented MCRC Space After Write to Unimplemented MCRC Space</i>
Severity	Low
Expected Behavior	A write to the legal address region of the MCRC module should not generate an abort
Issue	An Unexpected CPU Data Abort is generated in the following condition:
Conditions	When a normal mode write to an illegal address region of MCRC register space is followed by a write to a legal address region of the MCRC register space.
Implications	A write to an illegal address region of the MCRC register space generates a data abort as expected. The next write to a legal address region of the MCRC register space generates an unexpected second data abort.
Workaround(s)	None

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

MIBSPI#110	<i>Multibuffer Slave In 3 or 4 Pin Mode Transmits Data Incorrectly for Slow SPICLK Frequencies and for Clock Phase = 1</i>
Severity	Medium
Expected Behavior	The SPI must be able to transmit and receive data correctly in slave mode as long as the SPICLK is slower than the maximum frequency specified in the device datasheet.
Issue	The MibSPI module, when configured in multi-buffered slave mode with 3 functional pins (CLK, SIMO, SOMI) or 4 functional pins (CLK, SIMO, SOMI, nENA), could cause incorrect data to be transmitted.
Conditions	<p>This issue can occur under the following condition:</p> <ul style="list-style-type: none"> • 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
Implications	Under the above described condition, the slave MibSPI module can transmit incorrect data.
Workaround(s)	The issue can be avoided by setting the CSHOLD bit in the control field of the TX RAM. The nCS is not used as a functional signal in this communication, hence setting the CSHOLD bit does not cause any other effect on the SPI communication.

MIBSPI#111	<i>Data Length Error Is Generated Repeatedly In Slave Mode when I/O Loopback is Enabled</i>
Severity	Medium
Expected Behavior	After a data length (DLEN) error is generated and the interrupt is serviced the SPI should abort the ongoing transfer and stop.
Issue	When a DLEN error is created in Slave mode of the SPI using nSCS pins in IO LoopBack Test mode, the SPI module re-transmits the data with the DLEN error instead of aborting the ongoing transfer and stopping.
Conditions	This is only an issue for a IOLPBK mode Slave in Analog Loopback configuration, when the intentional error generation feature is triggered using CTRL_DLENERR(IOLPBKTSTCR.16).
Implications	The SPI will repeatedly transmit the data with the DLEN error when configured in the above configuration.
Workaround(s)	After the DLEN_ERR interrupt is detected in IOLPBK mode, disable the transfers by clearing the SPIEN bit of SPIGCR1 register (bit 24) and then re-enable the transfers by setting SPIEN.

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

Severity	High
Expected Behavior	The N2HET is a complex timer that operates by executing a set of timing instructions each loop resolution period. Normally counters and capture registers implemented by such a scheme would be limited in resolution to the timer's loop period; but the N2HET also includes dedicated hardware timer extensions that enable operation with resolution as high as 1/128 of the timer loop period. The WCAP instruction is designed to support time-stamping with high resolution. When an event occurs, the WCAP instruction is supposed to capture both the loop-resolution counter value and the high-resolution offset measuring the interval between the start of the loop and actual event occurrence, and return the combined result. This result can be interpreted as the number of timer loops represented as fixed point number with 25 integral bits and 7 fractional bits.
Issue	WCAP may not capture the high-resolution offset correctly.
Conditions	If the edge event that the WCAP instruction is time-stamping happens to align with the start of the internal loop resolution period, then the WCAP does not properly capture the 7-bit fractional offset correctly
Implications	Under the above described condition, the fractional offset of the previous event captured by WCAP is what is stored in the WCAP data field. Because of this boundary condition, the WCAP instruction should not be used in high-resolution mode
Workaround(s)	<p>A similar instruction, PCNT, is known to operate correctly at high resolution. While WCAP returns the absolute time-stamp of an event, PCNT returns the interval between an event and the previous event on the same pin.</p> <p>Instead of WCAP, if an absolute time-stamp is required it is recommended to use a PCNT instruction followed by an ADD instruction that accumulates the time between successive edge measurements to produce a result that is a WCAP equivalent.</p> <p>An example for using PCNT as a work around for WCAP:</p> <pre>WCAP {next = ERRPIN, cond_addr = JMP, hr_lr=high, reg=T, event=RISE, pin=16, data=0}</pre> <p>work around:</p> <pre>SAVE: MOV32 {remote=TRISE, type=REMTOREG, reg=R} TRISE: PCNT {next=TIMESTAMP, hr_lr=high, type=RISE2RISE, pin=16, data=0} TIMESTAMP: ADD {src1=REM, src2=R, rdest=REM, remote=TRISE, dest=NONE, next=CHK, data=0, hr_data=0} CHK: BR {cond_addr=JMP, event=RISE, pin=20,next=ERRPIN}</pre>

NHET#53	<i>Enhanced Input Capture Pins May not Capture Small Pulses Correctly</i>
Severity	High
Expected Behavior	The PCNT and WCAP instructions can capture pulse length or time stamp of small pulses that have two edges within a single loop resolution.
Issue	The high resolution value may be captured incorrectly.
Conditions	If the second edge event that the PCNT or WCAP instruction is using happens to align with the start of the internal loop resolution period, then the instruction does not properly capture the high resolution value correctly.
Implications	Because of this boundary condition, the PCNT and WCAP instructions should not be used on small pulses.
Workaround(s)	None

SYS#46	<i>Clock Source Switching Not Qualified With Clock Source Enable And Clock Source Valid</i>
Severity	Low
Expected Behavior	An attempt to switch to a clock source which is not valid yet should be discarded.
Issue	Switching a clock source by simply writing to the GHVSRC bits of the GHVSRC register may cause unexpected behavior. The clock will switch to a source even if the clock source was not ready.
Conditions	The clock source is enabled (CSDIS register) but the clock source is not yet valid (CSVSTAT register).
Implications	Unexpected behavior stated above.
Workaround(s)	Always check the CSDIS register to make sure the clock source is turned on and check the CSVSTAT register to make sure the clock source is valid. Then write to GHVSRC to switch the clock.

SYS#102	<i>Bit field EFUSE_Abort[4:0] in SYSTASR register is read-clear</i>
Severity	Medium
Expected Behavior	The documentation states that EFUSE_Abort[4:0] of the SYSTASR register should be write-clear in privilege mode.
Issue	However, these bits are implemented as read-clear.
Conditions	Always.
Implications	Software implementation for error handling needs to take care of this.
Workaround(s)	None

SYS#111	<i>The Device may Generate Multiple nRST Pulses.</i>
Severity	Medium
Expected Behavior	When an event on the device causes a system reset to be asserted, the device drives the system reset (nRST) terminal low for 8 VCLK periods. Then the nRST terminal is released and gets pulled High, allowing the device to start its warm boot sequence.
Issue	It is possible that the device may generate multiple system resets instead of releasing the system reset (nRST) at the appropriate time and starting the warm boot sequence.
Conditions	This issue occurs when either the external reset pulse or the 8 VCLK internally generated reset pulse is the same length as the delay caused by the glitch filter on the nRST signal. The delay time of the glitch filter is a function of device process, the temperature and the core voltage. This has a range from 500ns to 2 μ s. The length of the 8 VCLK reset extension is a function of the crystal speed, the I/O voltage, and the resistance and capacitance on the nRST pin. There is no problem when reset is properly asserted with the PORRST pin for 1ms or more as the nRST pin will be held low much longer than 2 μ s.
Implications	Usually only one or two extra reset pulses are generated. However, it is possible, in rare cases, that a long string of reset pulses could be generated. This would prevent the device from booting up in a timely manner.
Workaround(s)	In the case of an externally generated system reset, this issue can be avoided by asserting the system reset signal for more than 2 μ s. If the issue is seen when there is an internal system reset condition then the issue can be avoided by using an 8MHz or slower crystal. Another workaround is to avoid the use of internal system resets such as software reset, oscillator fault reset, watchdog reset, or debug reset. This erratum has been fixed on newer devices by extending the time of the internally generated nRSTpulse to 32 VCLK cycles. This extends the reset pulse to a minimum time of 3.2 μ s (with a 20MHz crystal), which exceeds the maximum delay time of the glitch filter.

VIM#27	<i>Unexpected phantom interrupt</i>
Severity	High
Expected Behavior	When responding to an interrupt and a subsequent interrupt is received, the corresponding VIM request should be flagged as pending in the VIM status registers. When the CPU is ready to service the subsequent interrupt, the correct service routine address should be fetched by the CPU.
Issue	On rare occasions the VIM may return the phantom interrupt vector instead of the real interrupt vector.
Condition	This condition is specific to software and hardware vectored modes. This is not applicable for legacy interrupt servicing mode. The ratio of GCLK to VCLK must be 3:1 or greater for hardware vectored mode. The ratio of GCLK to VCLK must be 5:1 or greater for software vectored mode. A subsequent interrupt request must occur when the VIM is finishing acknowledging a previous interrupt.
Implication(s)	The subsequent interrupt request vectors to the phantom interrupt routine instead of the correct service routine.
Workaround(s)	This issue can be avoided if the clock ratio GCLK:VCLK is 1:1 or 2:1. Otherwise the VIM status register can be checked within the phantom interrupt routine to determine the source of the subsequent interrupt.

IMPORTANT NOTICE

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

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

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

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

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

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

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

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

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

Products

Audio	www.ti.com/audio
Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DLP® Products	www.dlp.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
OMAP Applications Processors	www.ti.com/omap
Wireless Connectivity	www.ti.com/wirelessconnectivity

Applications

Automotive and Transportation	www.ti.com/automotive
Communications and Telecom	www.ti.com/communications
Computers and Peripherals	www.ti.com/computers
Consumer Electronics	www.ti.com/consumer-apps
Energy and Lighting	www.ti.com/energy
Industrial	www.ti.com/industrial
Medical	www.ti.com/medical
Security	www.ti.com/security
Space, Avionics and Defense	www.ti.com/space-avionics-defense
Video and Imaging	www.ti.com/video

TI E2E Community

e2e.ti.com