Cortex-M33 AT623 and Cortex-M33 with FPU AT624
Software Developer Errata Notice

This document contains all known errata since the r0p0 release of the product.

Non-Confidential Proprietary notice

LES-PRE-20349

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Web address

http://www.arm.com/.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on this document

If you have comments on content then send an e-mail to errata@arm.com giving:

- The document title.
- The document number: SDEN-756493.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Contents

Introduction

Scope

This document describes errata categorized by level of severity. Each description includes:

- The current status of the erratum.
- Where the implementation deviates from the specification and the conditions required for erroneous behavior to occur.
- The implications of the erratum with respect to typical applications.
- The application and limitations of a workaround where possible.

Categorization of errata

Errata are split into three levels of severity and further qualified as common or rare:

Category A	A critical error. No workaround is available or workarounds are impactful. The error is likely to be common for many systems and applications.
Category A (Rare)	A critical error. No workaround is available or workarounds are impactful. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category B	A significant error or a critical error with an acceptable workaround. The error is likely to be common for many systems and applications.
Category B (Rare)	A significant error or a critical error with an acceptable workaround. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category C	A minor error.

Change control

Errata are listed in this section if they are new to the document, or marked as “updated” if there has been any change to the erratum text. Fixed errata are not shown as updated unless the erratum text has changed. The errata summary table on page 10 identifies errata that have been fixed in each product revision.

14-Jul-2021: Changes in document version 13.0				
ID	Status	Area	Cat	Summary of erratum
2219175	New	Programmer	CatB	A partially completed VLLDM might leave secure floating-point data unprotected

23-Oct-2020: Changes in document version 12.0				
ID	Status	Area	Cat	Summary of erratum
1435973	Updated	Programmer	CatC	Execution priority might be wrong for one cycle after AIRCR, NVIC_ITNS, NVIC_IPR, NVIC_ISER, or NVIC_ICER is changed

05-Jun-2020: Changes in document version 11.0				
ID	Status	Area	Cat	Summary of erratum
No new or updated errata in this document version.				

09-Aug-2019: Changes in document version 10.0				
ID	Status	Area	Cat	Summary of erratum
1113997	New	Programmer	CatC	Group priority of a Non-secure interrupt might be incorrect when AIRCR.PRIS is set
1367266	New	Programmer	CatC	DFSR.EXTERNAL is not set correctly when waking up from sleep
1435973	New	Programmer	CatC	Execution priority might be wrong for one cycle after AIRCR, NVIC_ITNS, NVIC_IPR, NVIC_ISER, or NVIC_ICER is changed
1453380	New	Programmer	CatC	Non-secure HardFault exception might preempt when disabled by AIRCR.BFHFNMINS
1540599	New	Programmer	CatC	Sorting of pending interrupts might be wrong when high latency IRQs are pending

25-Apr-2018: Changes in document version 9.0				
ID	Status	Area	Cat	Summary of erratum
1080541	New	Programmer	CatC	Access permission faults are prioritized over unaligned Device memory faults

17-Jan-2018: Changes in document version 8.0				
ID	Status	Area	Cat	Summary of erratum

No new or updated errata in this document version.

08-Dec-2017: Changes in document version 7.0

ID	Status	Area	Cat	Summary of erratum
No new or updated errata in this document version.				

27-Nov-2017: Changes in document version 6.0

ID	Status	Area	Cat	Summary of erratum
937163	New	Programmer	CatC	Floating-Point state can be incorrectly cleared on some exception return faults
1015127	New	Programmer	CatC	Processor might not wake up to a SEVONPEND event when in WIC-based WFE sleep

23-Oct-2017: Changes in document version 5.0

ID	Status	Area	Cat	Summary of erratum
No new or updated errata in this document version.				

16-May-2017: Changes in document version 4.0

ID	Status	Area	Cat	Summary of erratum
839443	New	Programmer	CatB	Cortex-M33 DWT trace is corrupted when trace buffers are full and can potentially deadlock the PE
832634	New	Programmer	CatC	Cortex-M33 reads an incorrect DAP AP IDR value
836306	New	Programmer	CatC	DebugMonitor stepping does not step the next instruction and keeps executing the DebugMonitor handler
840453	New	Programmer	CatC	Halting debug steps two consecutive exception entry sequences
849231	New	Programmer	CatC	DEMCR.MON_PEND is set because of EDBGRQ even though the current execution priority is higher than DebugMonitor priority
851802	New	Programmer	CatC	Fault on an exception exit is escalated to HardFault when IPSR is corrupted
855792	New	Programmer	CatC	Cortex-M33 DAP does not reset the JTAG TAP on an extended JTAG activation code
857433	New	Programmer	CatC	Pending serious faults are not always indicated
861432	New	Programmer	CatC	DebugMonitor exception exit when IPSR is corrupted fails to step instruction
875823	New	Programmer	CatC	Some T32 unallocated hint encodings UNDEF rather than NOP

02-Feb-2017: Changes in document version 3.0				
ID	Status	Area	Cat	Summary of erratum
795154	New	Programmer	CatC	MTB trace might be incorrect when the processor enters lockup state
796137	New	Programmer	CatC	Conditional trace can be incorrect if FPU is not present
797273	New	Programmer	CatC	ETM Exception Return addresses are incorrect for some lockup exceptions
801633	New	Programmer	CatC	Processor cannot exit sleep state when debug wakeup is not active long enough to be detected
803047	New	Programmer	CatC	AIRCR.BFHFNMINS update does not change fault security state when CPUWAIT is set out of reset
811381	New	Programmer	CatC	Halting debug steps two consecutive exception entry sequences
812148	New	Programmer	CatC	MTB trace might be incorrect when the processor receives NMI while it executes faulting instruction leading to lockup

08-Dec-2016: Changes in document version 2.0				
ID	Status	Area	Cat	Summary of erratum
764623	New	Programmer	CatB	Pended derived NOCP fault on taking a Non-secure interrupt from Secure state fails to stack callee registers
784769	New	Programmer	CatB	CLRONRET functionality will not work as expected
789130	New	Programmer	CatB	Processor might behave incorrectly when the FPU is powered down into retention
770545	New	Programmer	CatC	Integrity signature fault might be ignored during exception unstacking when the core should lockup
778249	New	Programmer	CatC	AIRCR.BFHFNMINS update does not change security state for faults when DHCSR.C_HALT is set or when NMI pended when current execution priority is -2 or -3
787405	New	Programmer	CatC	NOCP Usage fault is not generated for coprocessor instructions when disabled by CPPWR.SUn
788875	New	Programmer	CatC	Debug accesses to FPU registers are incorrect if CPACR.CP10 is disabled

28-Sep-2016: Changes in document version 1.0				
ID	Status	Area	Cat	Summary of erratum
No errata in this document version.				

Errata summary table

The errata associated with this product affect product versions as below.

ID	Cat	Summary	Found in versions	Fixed in version
764623	CatB	Pended derived NOCP fault on taking a Non-secure interrupt from Secure state fails to stack callee registers	r0p0	r0p1
784769	CatB	CLRONRET functionality will not work as expected	r0p0	r0p1
789130	CatB	Processor might behave incorrectly when the FPU is powered down into retention	r0p0	r0p1
839443	CatB	Cortex-M33 DWT trace is corrupted when trace buffers are full and can potentially deadlock the PE	r0p1	r0p2
2219175	CatB	A partially completed VLLDM might leave secure floating-point data unprotected	r0p0, r0p1, r0p2, r0p3, r0p4, r1p0	Open
770545	CatC	Integrity signature fault might be ignored during exception unstacking when the core should lockup	r0p0	r0p1
778249	CatC	AIRCR.BFHFNMINS update does not change security state for faults when DHCSR.C_HALT is set or when NMI pended when current execution priority is -2 or -3	r0p0	r0p1
787405	CatC	NOCP Usage fault is not generated for coprocessor instructions when disabled by CPPWR.SUn	r0p0	r0p1
788875	CatC	Debug accesses to FPU registers are incorrect if CPACR.CP10 is disabled	r0p0	r0p1
795154	CatC	MTB trace might be incorrect when the processor enters lockup state	r0p0	r0p1
796137	CatC	Conditional trace can be incorrect if FPU is not present	r0p0	r0p1
797273	CatC	ETM Exception Return addresses are incorrect for some lockup exceptions	r0p0, r0p1	r0p2
801633	CatC	Processor cannot exit sleep state when debug wakeup is not active long enough to be detected	r0p0	r0p1
803047	CatC	AIRCR.BFHFNMINS update does not change fault security state when CPUWAIT is set out of reset	r0p0	r0p1
811381	CatC	Halting debug steps two consecutive exception entry sequences	r0p0	r0p1
812148	CatC	MTB trace might be incorrect when the processor receives NMI while it executes faulting instruction leading to lockup	r0p0	r0p1

ID	Cat	Summary	Found in versions	Fixed in version
832634	CatC	Cortex-M33 reads an incorrect DAP AP IDR value	r0p0, r0p1	r0p2
836306	CatC	DebugMonitor stepping does not step the next instruction and keeps executing the DebugMonitor handler	r0p0, r0p1	r0p2
840453	CatC	Halting debug steps two consecutive exception entry sequences	r0p0, r0p1	r0p2
849231	CatC	DEMCR.MON_PEND is set because of EDBGRQ even though the current execution priority is higher than DebugMonitor priority	r0p0, r0p1	r0p2
851802	CatC	Fault on an exception exit is escalated to HardFault when IPSR is corrupted	r0p0, r0p1	r0p2
855792	CatC	Cortex-M33 DAP does not reset the JTAG TAP on an extended JTAG activation code	r0p0, r0p1	r0p2
857433	CatC	Pending serious faults are not always indicated	r0p0, r0p1	r0p2
861432	CatC	DebugMonitor exception exit when IPSR is corrupted fails to step instruction	r0p0, r0p1	r0p2
875823	CatC	Some T32 unallocated hint encodings UNDEF rather than NOP	r0p0, r0p1	r0p2
937163	CatC	Floating-Point state can be incorrectly cleared on some exception return faults	r0p0, r0p1, r0p2, r0p3	r0p4
1015127	CatC	Processor might not wake up to a SEVONPEND event when in WIC-based WFE sleep	r0p0, r0p1, r0p2, r0p3	r0p4
1080541	CatC	Access permission faults are prioritized over unaligned Device memory faults	r0p0, r0p1, r0p2, r0p3, r0p4, r1p0	Open
1113997	CatC	Group priority of a Non-secure interrupt might be incorrect when AIRCR.PRIS is set	r0p0, r0p1, r0p2, r0p3, r0p4, r1p0	Open
1367266	CatC	DFSR.EXTERNAL is not set correctly when waking up from sleep	r0p0, r0p1, r0p2, r0p3, r0p4, r1p0	Open
1435973	CatC	Execution priority might be wrong for one cycle after AIRCR, NVIC_ITNS, NVIC_IPR, NVIC_ISER, or NVIC_ICER is changed	r0p0, r0p1, r0p2, r0p3, r0p4, r1p0	Open
1453380	CatC	Non-secure HardFault exception might preempt when disabled by AIRCR.BFHFNMINS	r0p0, r0p1, r0p2, r0p3, r0p4, r1p0	Open
1540599	CatC	Sorting of pending interrupts might be wrong when high latency IRQs are pending	r0p0, r0p1, r0p2, r0p3, r0p4, r1p0	Open

Errata descriptions

Category A

There are no errata in this category.

Category A (rare)

There are no errata in this category.

Category B

764623

Pended derived NOCP fault on taking a Non-secure interrupt from Secure state fails to stack callee registers

Status

Affects: Cortex-M33
Fault Type: Programmer Category B
Fault Status: Present in r0p0. Fixed in r0p1.

Description

Under certain conditions, the floating-point context is stacked when the processor takes an exception. Before stacking the registers, an FPU access check occurs based on the CPACR, NSACR, and CPPWR state. If the core in its current state does not have access to the FPU, then this results in a NOCP Usage Fault. When the processor takes a Non-secure exception from Secure state, the full set of integer and floating-point registers is usually stacked and then set to zero. This avoids any Secure to Non-secure software information leak. Because of this erratum, under certain conditions and if a NOCP Usage Fault occurs, the processor will erroneously clear the floating-point state and fail to stack the integer callee registers when the processor takes an exception.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor configured with the Armv8-M Security Extension and hardware floating-point support.

Conditions

- The processor is in Secure state with CONTROL.FPCA = 1 and FPCCR.LSPEN = 0 (lazy state preservation is disabled).
- The FPU cannot be accessed according to the architecture because either:
    - CPACR.CP10 is 0b00 or 0b01 and the processor is in non-privileged mode, or
    - CPPWR.SU10 is 1.
- The processor takes an exception to Non-secure state with an exception privilege level higher than a Secure Usage Fault.

Implications

Because all the floating-point register state has been set to zero and only some of the integer registers have been written to the Stack on the exception entry, on return to Secure state software will not be able to continue processing the floating-point context after the FPU has been re-enabled. The NOCP UsageFault will occur but the context will have been lost. Integer register state (R4-R11) will also have been lost.

Workaround

You can avoid this erratum by either:

- Not disabling lazy floating-point preservation in Secure state (CONTROL.LSPEN is enabled by default at reset).
- Ensuring that the priority of a Secure UsageFault is always higher than the priority of a Non-secure exception. You can do this by using the AIRCR.PRIS functionality.

Additional information

The scenario considered for this erratum is an OS wanting to perform software based lazy-context switching of the FPU. This means it is possible for the FP state to belong to an out-of-context thread while the integer state belongs to the current thread. In this scenario the NOCP fault (for the out-of-context thread) should not affect the stacking of the integer registers for the current thread. There is no known operating system which does this.

784769

CLRONRET functionality will not work as expected

Status

Affects: Cortex-M33
Fault Type: Programmer Category B
Fault Status: Present in r0p0. Fixed in r0p1.

Description

The Armv8-M architecture includes functionality to clear the floating-point registers to zero on return from exception when FPCCR.CLRONRET is set. Because of this erratum, the clearing of the registers will not take place under certain conditions.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor.

Conditions

When the architectural conditions are met (FPCCR.CLRONRET=1, CONTROL.FPCA=1, and FPCCR_S.LSPACT=0), there are three scenarios where the registers will not be cleared as expected:

1. The processor is configured without the Security Extension (SECEXT=0).
2. The processor is returning to a security state where the CPACR is disabled.
3. The FPU is powered down into retention.

Implications

Secure or privileged floating-point caller state (S0-S15) created within a handler can be exposed to Non-secure or unprivileged code.

Workaround

Software must clear the FPU register state manually before returning from an exception.

789130

Processor might behave incorrectly when the FPU is powered down into retention

Status

Affects: Cortex-M33
Fault Type: Programmer Category B
Fault Status: Present in r0p0. Fixed in r0p1.

Description

When the FPU is powered down into retention it is possible for the processor to behave incorrectly. This can be seen in two ways:

1. On exception entry or during lazy stacking it is not possible to clear the FPU register state. FPUQACTIVE will have been asserted previously but there is a race condition between the FPU powering up and the core attempting to clear the state.
2. When returning from a secure exception to non-secure state, the core can stall indefinitely while the FPU is powered down. This requires multiple faults to occur from secure state.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor that are:

- Configured with the Security Extension.
- Implemented with the ability to power down the FPU into retention (separately from the processor core).

Conditions

There are two conditions under which this erratum can occur:

1. The core has an outstanding secure floating point context, and software disables CPACR_S (FPCA remains set). The system powers down the FPU into retention. One of the following then happens:
    1. The FPU is re-enabled through CPACR_S, and then a non-secure exception is taken. During stacking an earlier fault prevents stacking of the FPU registers (LSPEN=0). The FPU register clearing should still take place but this can happen before the FPU is powered up, in which case it will have no effect.
    2. A non-secure exception is entered and sets up lazy stacking (LSPEN=1, LSPACT_S=1). The FPU is enabled using CPACR_NS and the NSACR allows access. CPACR_S is still disabled and therefore a NOCP fault occurs when attempting to lazy stack. If the fault cannot preempt the current handler, and if the core will not lockup, LSPACT_S is cleared but the state can be left unchanged if the attempt to clear occurs before the FPU is powered up.
2. The core is in secure state with LSPACT_S=1, CPACR_S disabled and FPU powered down into retention. An exception return is triggered to non-secure state (EXCRET[6]=0) and an integrity check fault occurs before the security state switch that prevents it. If EXCRET[4]=0 (non-secure has floating point context) then an LSERR fault will also be raised as LSPACT_S=1. The core will stall while waiting for the FPU to power up but will not request power (if CPACR_NS is enabled)

Implications

The implications for each of the two conditions are as follows:

1. It is possible for secure state to be exposed to non-secure, if the FPU is not powered up when the register is cleared.
2. The core can stall indefinitely until the FPU is powered up.

Workaround

The erratum can be avoided by not powering down the FPU into retention while the core remains powered up.
Note: This workaround can be limited to when the FPU registers contain secure information.

839443

Cortex-M33 DWT trace is corrupted when trace buffers are full and can potentially deadlock the PE

Status

Affects: Cortex-M33
Fault Type: Programmer Category B
Fault Status: Present in r0p1. Fixed in r0p2.

Description

The Cortex-M33 DWT contains buffers for trace generated by the DWT. As defined in the Armv8-M Architecture Reference Manual, some of the buffers contain priority ordering logic for when their packets are forwarded to the ITM. If there is sufficient backpressure from the ITM and the system, then a subset of the DWT buffers might become full and the packet ordering logic does not function as expected. This will lead to packets being read out of order, one packet never being read, and an out-of-protocol packet being continuously output by the DWT. Also, if ITM_TCR.STALLENA is written from 0 to 1 during the trace generation that causes the packet ordering logic to become non-functional, then the ITM might stall the PE indefinitely.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor with DWT and ITM trace support.

Conditions

The following conditions might cause the DWT packet ordering logic to not function as expected:

- DWT trace enabled.
- Sufficient backpressure from the ITM and the system for the DWT to have to buffer packets.
- Generation of the following packets:
    - Two Data Trace PC Value.
    - Two Data Trace Data Address.
    - Three Data Trace Data Value.
    - Four Data Trace Match.
      Note that these packets must be caused by instructions issued into lane 0 of the Cortex-M33 dual-issue pipe or stacking operations because there is separate buffering for packets generated by instructions issued into lane 1 of the Cortex-M33 dual-issue pipe.

The following additional condition might cause the PE to deadlock:

- ITM_TCR.STALLENA is written from 0 to 1 in the middle of generating the above packets.

Implications

Debug tools might not be able to decompress DWT/ITM trace and the PE might deadlock. Both implications require a power-on reset to be recovered from.

Workaround

You can work around this erratum by both:

- Limiting the amount of trace generated by the DWT so that its buffers do not become full. To do this, Arm recommends limiting the programming of the DWT comparators to not generate one of the following packet types:
    - Data Trace PC Value.
    - Data Trace Data Address.
    - Data Trace Data Value.
    - Data Trace Match.
- Writing to ITM_TCR.STALLENA only when DWT trace generation is disabled with ITM_TCR.TXENA == 0.

2219175

A partially completed VLLDM might leave Secure floating-point data unprotected

Status

Affects: Cortex-M33
Fault Type: Programmer Category B
Fault Status: Present in r0p0, r0p1, r0p2, r0p3, r0p4, r1p0. Open.

Description

The VLLDM instruction allows Secure software to restore a floating-point context from memory. Due to this erratum, if this instruction is interrupted or it faults before it completes, then Secure data might be left unprotected in the floating-point register file, including the FPSCR.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor configured with the Armv8-M Security Extension and the Floating-point Extension.

Conditions

This erratum occurs when all the following conditions are met:

- There is no active floating-point context, (CONTROL.FPCA==0)
- Secure lazy floating-point state preservation is not active, (FPCCR_S.LSPACT==0)
- The floating-point registers are treated as Secure (FPCCR_S.TS==1)
- Secure floating-point state needs to be restored, (CONTROL_S.SFPA == 1)
- Non-secure state is permitted to access to the floating-point registers, (NSACR.CP10 == 1)
- A VLLDM instruction has loaded at least one register from memory and does not complete due to an interrupt or fault

Implications

If the floating-point registers contain Secure data, a VLSTM instruction is usually executed before calling a Non-secure function to protect the Secure data. This might cause the data to be transferred to memory (either directly by the VLSTM or indirectly by the triggering of a subsequent lazy state preservation operation). If the data has been transferred to memory, it is restored using VLLDM on return to Secure state.

If the VLLDM is interrupted or it faults before it completes and enters a Non-secure handler, the partial register state which has been loaded will be accessible to Non-secure state.

Workaround

To avoid this erratum, software can ensure a floating-point context is active before executing the VLLDM instruction by performing the following sequence:

- Read CONTROL_S.SFPA
- If CONTROL_S.SFPA==1 then execute an instruction which has no functional effect apart from causing context creation (such as `VMOV S0, S0`)

Category B (rare)

There are no errata in this category.

Category C

770545

Integrity signature fault might be ignored during exception unstacking when the core should lockup

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

Description

During exception unstacking, the integrity signature is checked if the integer callee state is being popped. This occurs when the processor returns to Secure state and the integer callee state is on the stack. This check ensures that the exception is returning to a legitimate point in the Secure stack.

Under certain conditions, the fault might be ignored and the processor might return to Secure state without handling the fault. This exposes a potential security risk. To trigger this erratum, the processor must be configured for sleep-on-exit and must be about to enter sleep state (that is, there are no other active or pending exceptions).The fault must also trigger lockup because faultmask_s is set.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor configured with the Armv8-M Security Extension.

Conditions

- The processor is returning to Secure state with faultmask_s set.
- The integrity signature is checked (EXC_RETURN.ES==0 or EXC_RETURN.DCRS==0).
    - faultmask_s must not be cleared on exit (EXC_RETURN.ES==0 or the raw execution priority is negative).
- The processor attempts to sleep-on-exit.
    - It means that the current handler is the only active handler and there are no other handlers pending. Sleep-on-exit is enabled.
- An INVIS fault is raised and triggers lockup (because of faultmask_s).
- No other integrity checks must fail.

Implications

It is possible for Secure or Non-secure exceptions to return to arbitrary points in the Secure stack.

- Secure handlers must have modified the stack, SP, or EXC_RETURN value to trigger this erratum because of the type of fault.
- Non-secure handlers can fake EXC_RETURN.S to return to arbitrary locations if a Secure function call within the handler has set faultmask_s.

Note: This behavior is only visible if no other integrity check faults occur. Because the fault is ignored, unstacking continues and the xPSR is unstacked. This IPSR value must be 0x0 to match the mode.

Workaround

You can avoid this erratum by either:

- Disabling SCR_S.SLEEPONEXIT when faultmask_s is set.
- Ensuring a Non-secure handler has no way of setting faultmask_s by calling secure code.

778249

AIRCR.BFHFNMINS update does not change security state for faults when DHCSR.C_HALT is set or when NMI pended when current execution priority is -2 or -3

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

Description

The software must ensure that, when changing AIRCR.BFHFNMINS, the BusFault, HardFault and the NMI are not pending or active. Therefore AIRCR.BFHFNMINS should be set at boot time before any exceptions can happen. Unpredictable behavior can occur if you do not follow this guidance.

An internal buffered version of AIRCR.BFHFNMINS is implemented that is allowed to update only when there are no context switches to reduce the occurrence of unpredictable behavior. Therefore, during exception stacking/unstacking, when the processor is performing a context switch or when an NMI is about to be invoked, the architectural AIRCR.BFHFNMINS is not propagated to the buffered version of AIRCR.BFHFNMINS.

However, the internal buffered version is blocked from being updated to the architectural value when DHCSR.C_HALT is set or when the NMI is pended. Therefore, in the case of DHCSR.C_HALT, a Non-secure exception/thread could block the update from ever taking place regardless of the debug enables or the authentication state. Also, a pended NMI would have a similar effect when the processor is in priority -2 or -3 because it would also block it from being updated.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor configured with the Armv8-M Security extensions.

Conditions

- The processor is in the secure state and DHCSR.C_HALT is set OR NMI is pended when the processor is in priority -2 or -3 (and its not in debug state).
- The internal version of AIRCR.BFHFNMINS is blocked from updating.

Implications

It is possible for Non-secure exceptions to block updates of the architectural AIRCR.BFHFNMINS to the internal buffered version, which is used by the processor when resolving exceptions.

Workaround

This erratum is not expected to require a workaround. This erratum can be avoided by:

- Clearing DHCSR.C_HALT and ICSR.PENDNMICLR when AIRCR.BFHFNMINS is updated.
- Restoring values of DHCSR.C_HALT and ICSR.PENDNMICLR after updating AIRCR.BFHFNMINS.

787405

NOCP Usage fault is not generated for coprocessor instructions when disabled by CPPWR.SUn

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

Description

The CPPWR value viewed by the processor is always 0 when in the Non-secure configuration. This means that it is possible for a coprocessor instruction to complete without generating a NOCP fault, even though the relevant architectural CPPWR.SUn=1. This affects all implemented coprocessors. In Cortex-M33 this is limited to CP0-CP7,CP10-CP11.

Note: PPB accesses return the correct values for CPPWR.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor that are configured without the Armv8-M Security extensions.

Conditions

- The processor executes a coprocessor instruction when CPPWR.SUn is set to '1' and the coprocessor is enabled by CPACR.CPn.
- The processor executes the instruction without generating a NOCP usage fault.

Implications

It is possible to execute coprocessor instructions when you have marked the floating-point state as UNKNOWN to allow the coprocessor to be powered down.

Workaround

Ensure that CPPWR.SUn for a coprocessor is set only when disabled by CPACR.CPn. This ensures the NOCP fault is correctly generated if access is disabled.

Note: This erratum is unlikely to occur because the expectation is that the coprocessor state will be marked as UNKNOWN only while access is disabled in CPACR.

788875

Debug accesses to FPU registers are incorrect if CPACR.CP10 is disabled

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

Description

The FPU clock enable is based on the current CPACR value. If CPACR.CP10 is disabled then debug accesses to FPU registers will not be performed, writes will be ignored, and read data will be incorrect.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor that are configured with hardware floating-point support.

Conditions

- The processor enters halt state.
- CPACR.CP10/11 is disabled for the current security state.
- A debug access is issued to the FPU.

Implications

- Reads of FPU registers return an incorrect value.
- Writes to FPU registers are ignored.

Workaround

You can avoid this erratum by making sure CPACR.CP10 is enabled for the current security state when the processor is halted.

795154

MTB trace might be incorrect when the processor enters lockup state

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

Description

Under certain conditions, the MTB might trace an incorrect destination address when the processor enters lockup state. The source and destination addresses in the generated packet will be identical whereas the destination address should be the lockup address.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor configured with the hardware floating-point support.

Conditions

- Out of order completion of floating-point instructions with respect to non-floating-point instructions must be enabled (ACTLR.DISOOFP = 0).
- A VSQRT/VDIV instruction executes while the processor is executing at a negative priority level and an event occurs which causes the processor to enter lockup state. The event could be the execution of an SVC instruction or a synchronous fault. The erratum occurs if the next instruction in the pipeline is:

- A VSTR to the same destination register as the VSQRT/VDIV.
- A VLSTM and the destination register of the VSQRT/VDIV is S0.

Implications

The erratum causes the trace produced by the MTB to be incorrect for the last packet before the processor enters lockup state. As a consequence, the source and destination addresses in the packet are the same whereas the destination address should be the lockup address.

Workaround

The MTB trace can be corrected if MTB is constantly authenticated to trace.

The erratum produces the following packet 1, followed by packet 2:

packet 1, first word: [address value A, atomic bit]
packet 1, second word: [address value A, start bit]

packet 2, first word: [address value of lockup, atomic bit]
packet 2, second word: [any address after leaving lockup, start bit]

The lockup address is [31:1] 0x77FFFFFF.

In this case, the destination address in the packet 1 should be replaced with the source address of packet 2, and the final trace should look like:

packet 1, first word: [address value A, atomic bit]
packet 1, second word: [address value of lockup, start bit]

packet 2, first word: [address value of lockup, atomic bit]
packet 2, second word: [any address after leaving lockup, start bit]

796137

Conditional trace can be incorrect if FPU is not present

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

Description

When the processor is configured without an FPU and conditional ETM trace is enabled, any conditional VLSTM instructions will be traced as if they were loads rather than stores.

Configurations affected

This erratum affects configurations of the Cortex-M33 processor with the ETM present and the FPU not present.

Conditions

- ETM trace is enabled with the VIEWINST consitions true.
- Conditional tracing is enabled with TRCCONFIGR.COND set to 0b001 or 0b010.
- A conditional VLSTM instruction is traced.

Implications

The VLSTM conditional information will be filtered as if the instruction was a load, rather than a store, so the number of conditional result elements will not align with the instructions expected by a decoder. All subsequent conditional results will be offset until the next exception or periodic synchronization.

Other than the conditional results, the trace stream is not corrupted.

Workaround

Trace tools can work around this erratum if they have knowledge of the FPU presence. When the FPU is not present, tools should set TRCCONFIGR.CONF to enable both load and store conditional tracing (0b011 or 0b111).

797273

ETM Exception Return addresses are incorrect for some lockup exceptions

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Fixed in r0p2.

Description

When the processor is configured with the ETM present and trace is enabled at the time that a lockup condition occurs, the ETM will sometimes report the incorrect 'preferred return address' for the exception. The address will always be either the lockup address or the instruction which will execute next. The scenarios where the address is wrong are either some conditions where an exception stacking operation cannot complete, or some conditions of a lockup occuring at the handler of an exception (when the lockup is traced as a derived lockup, rather than occurring at the handler).

Configurations affected

This erratum affects configurations of the Cortex-M33 processor with the ETM present.

Conditions

A lockup occurs during exception handling or immediately after.

Implications

The trace is consistent and decompressible, but the user will not be able to determine from the exception teturn address exactly how the lockup condition was generated.

Workaround

There is no workaround for this erratum.

801633

Processor cannot exit sleep state when debug wakeup is not active long enough to be detected

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

Description

If a WFI instruction coincides with a DWT event, for example a PC match on the WFI instruction, the processor wakes up if the debug monitor takes priority. Because of this erratum, the NVIC generates a debug wakeup for a DWT hit on a watchpoint and the processor does not wake up. The processor cannot wake up when the debug request is a pulse.

Configurations affected

This erratum affects configurations of the Cortex-M33 processor with DWT present (DBGLVL != 0).

Conditions

- DWT asserts a watchpoint hit for a single cycle when the processor is entering sleep state.
- The wakeup is asserted for a single cycle and the processor remains in sleep state because the wakeup has not been asserted long enough.

Implications

The processor might not wake up from sleep state because of the DWT hit, which would only occur when DWT matches on the WFI instruction or load (or store) before the WFI.

Workaround

For load or store instructions preceding a WFI, a DSB should be placed before the WFI to prevent this erratum from occurring if the watchpoint was hit on the data or address match.
When DWT is set to hit on the PC of the WFI itself, there is no workaround.

803047

AIRCR.BFHFNMINS update does not change fault security state when CPUWAIT is set out of reset

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

Description

The debugger is able to change the state of AIRCR.BFHFNMINS before the processor has come out of reset when the input signal CPUWAIT is asserted. Due to this erratum, AIRCR.BFHFNMINS is not allowed to propagate to the processor and as a consequence, an NMI immediately out of reset will take the wrong security target.

Configurations affected

This erratum affects configurations of the Cortex-M33 processor with the Armv8-M Security Extension implemented and debug access present (DBGLVL !=0).

Conditions

- The processor is stalled in reset state (CPUWAIT=1) and the debugger is updating AIRCR.BFHFNMINS.

Implications

It is possible for the NMI to take the wrong security target immediately out of reset because there were no updates of the architectural AIRCR.BFHFNMINS to the internal buffered version, which is used by the processor when resolving exceptions.

Workaround

There is no workaround for this erratum.

811381

Halting debug steps two consecutive exception entry sequences

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

Description

The Armv8-M architecture specifies that only a single PushStack() update can occur in a Halting debug step sequence. Because of this erratum, two PushStack() updates can be performed before the core completes the Halting debug step and halts.

Configurations affected

This erratum affects configurations of the Cortex-M33 processor with reduced or full set of debug resources (DBGLVL != 0).

Conditions

1. Halting debug stepping is active (DHCSR.C_DEBUGEN == 0b1 and DHCSR.C_STEP == 0b1) and the core starts stepping a Non-secure HardFault exception entry sequence.
2. A stacking fault forces the core to lock up at exception priority -1 after entering the Non-secure HardFault handler.
3. When the core is about to halt to complete the step, there is a one cycle window where an external NMI can preempt and result in a second exception entry sequence being performed before the core halt after entering NMI handler.

OR

1. AIRCR.BFHFNMINS == 0b1 and an external debugger writes to SHCSR and sets Secure HardFault to be pended.
2. Halting debug stepping is active (DHCSR.C_DEBUGEN == 0b1 and DHCSR.C_STEP == 0b1) and the core starts stepping a NMI exception entry sequence.
3. A stacking fault leads to the core locking up at exception priority -2 after entering the NMI handler.
4. The Secure HardFault preempts and results in a second exception entry sequence being performed before the core halts after entering the Secure HardFault handler.

Implications

The processor steps two consecutive PushStack() updates in a step sequence.

Workaround

There is no workaround for the first scenario. For the second scenario, the debugger should not write to SHCSR when performing Halting debug stepping.

812148

MTB trace might be incorrect when the processor receives NMI while it executes faulting instruction leading to lockup

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

Description

Under certain conditions, the MTB might trace the incorrect source address when the processor enters lockup state because of a faulting instruction and receives a coincident NMI.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor

Conditions

- The processor is executing a faulting instruction at a negative priority.
- The processor receives an NMI.
- The processor enters lockup.

Implications

The erratum causes the trace produced by the MTB to be incorrect for the last packet before the processor enters lockup state. As a consequence, the source word of the packet does not contain the source address of the faulting instruction causing the processor to enter lockup.

Workaround

There is no workaround for this erratum.

832634

Cortex-M33 reads an incorrect DAP AP IDR value

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Fixed in r0p2.

Description

The Cortex-M33 DAP contains an AP IDR register which can be used to identify the AP. In the r0p1 release, the AP IDR register reads the following value for bits[7:0]:

- Variant, bits[7:4] == 0x5 - Cortex-M33
- Type, bits[3:0] == 0x1 - AHB Bus

However, the Arm Debug Architecture defines a new type for AHB5 so that the AP IDR register should read the following value for bits[7:0]:

- Variant, bits[7:4] == 0x1 - Cortex-M33
- Type, bits[3:0] == 0x5 - AHB5 Bus

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor with a DAP present.

Conditions

Debug tools read the Cortex-M33 DAP AP IDR value.

Implications

Debug tools will not be able to detect that they are connected to an AHB5 AP.

Workaround

Given that the r0p0 and r0p1 value is uniquely identifiable and the value is documented in the Integration and Implementation Manual, debug tools will still be able to identify this as a Cortex-M33 DAP and could infer that it therefore supports AHB5.

836306

DebugMonitor stepping does not step the next instruction and keeps executing the DebugMonitor handler

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Fixed in r0p2.

Description

When performing DebugMonitor stepping and executing an out-of-order FPU instruction belonging to a subset of FPU instructions inside the DebugMonitor handler, then, when returning from the DebugMonitor handler and if the DebugMonitor exception priority is higher than the current execution priority, the DebugMonitor exception might be taken through the tail-chain mechanism without stepping any instruction. Without any debugger intervention to disable debug monitor stepping, it would appear like the DebugMonitor handler keeps executing back-to-back without stepping any intruction.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor.

Conditions

- Debug monitor stepping is active:
    - DHCSR.C_DEBUGEN == 1'b0.
    - DEMCR.MON_EN == 1'b1.
    - Execution priority is below the DebugMonitor exception priority.
- Out-of-order execution of floating-point instruction is active:
    - ACTLR.DISOOFP == 1'b0.
- The DebugMonitor handler executes one of the following VFP instructions:
    - VDIV.
    - VSQRT.
    - VMLA, VMLS, VNMLA, VNMLS.
    - VFMA, VFMS, VFNMA, VFNMS.
- This VFP instruction retires after the exception return instruction.

Implications

If no workaround is applied and if the DebugMonitor exception priority is always higher than the current execution priority, then the DebugMonitor exception will keep tail-chaining and execute its handler as long as DebugMonitor stepping is active.

Workaround

If the DebugMonitor handler contains one of the VFP instructions listed in the Conditions section, then always execute a VMOV after this instruction to work around this erratum.

840453

Halting debug steps two consecutive exception entry sequences

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Fixed in r0p2.

Description

The Armv8-M architecture specifies that only a single PushStack() update can occur in a Halting debug step sequence. Because of this erratum, two PushStack() updates can be performed before the core completes the Halting debug step and halts.

Configurations affected

This erratum affects configurations of the Cortex-M33 processor with reduced or full set of debug resources, DBGLVL! =0, and FPU present, FPU==1.

Conditions

1. Halting debug stepping is active, DHCSR.C_DEBUGEN == 0b1 and DHCSR.C_STEP == 0b1.
2. The core starts stepping an exception entry sequence whose stack frame contains the floating-point registers context. Lazy stacking is not active.
3. Following the entry in the exception handler, there is a one-cycle window where an asynchronous interrupt can preempt. This results in a second exception entry sequence being performed before the core halts.

Implications

The processor steps two consecutive PushStack() updates in a step sequence.

Workaround

This erratum can be avoided by activating lazy stacking before the first exception is taken, with FPCCR.LSPEN == '1'.

849231

DEMCR.MON_PEND is set because of EDBGRQ even though the current execution priority is higher than DebugMonitor priority

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Fixed in r0p2.

Description

During stacking, an EDBGRQ is asserted and the DEMCR.MON_PEND is set even though the current execution priority is higher than the priority of the DebugMonitor.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor where at least one IRQ has IRQLATENCY=0.

Conditions

- Halting debug is disabled, DHCSR.C_DEBUGEN=0.
- DebugMonitor is enabled, DEMCR.MON_EN=1.
- An external debug request is asserted, EDBGRQ=1.
- A registered IRQ has been preempted and the IPSR has been updated before the exception entry has happened.
- The current execution priority is higher than the DebugMonitor priority.

Implications

The DebugMonitor is pended earlier although it will not be taken until it has sufficient priority, that is, when the DebugMonitor priority is higher than the current execution priority.

Workaround

There is no workaround.

851802

Fault on an exception exit is escalated to HardFault when IPSR is corrupted

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Fixed in r0p2.

Description

During an exception exit of a HardFault/NMI handler where the IPSR has been corrupted, a SecureFault is asserted. The active bit of the HardFault/NMI exception is cleared correctly on the exit, however, the SecureFault is escalated to HardFault even though its priority is higher than the current execution priority.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor with IRQLATENCY!=0.

Conditions

- IPSR has to be corrupted so it is not pointing to the currently active exception although it is still pointing to an active exception (that is, a registered IRQ).
- The active exception is an NMI or a HardFault.
- An exception exit happens with the NMI or HardFault active bit cleared correctly and then generates a fault while unstacking (that is, an invalid exception return through SFSR.INVER).
- The fault is escalated when it had the highest priority.

Implications

A fault on an exception exit is escalated to HardFault when IPSR was corrupted because of a previous exception exit. It can only happen when the IPSR points to a registered IRQ where its active is set.

Workaround

There is no workaround.

855792

Cortex-M33 DAP does not reset the JTAG TAP on an extended JTAG activation code

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Fixed in r0p2.

Description

The Cortex-M33 DAP does not reset the JTAG TAP on an extended JTAG activation code sequence (state transition from G2 to dormant in the Arm Debug Interface Architecture Specification).

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor with a DAP present.

Conditions

An extended JTAG activation code on SWDIOTMS is applied and causes a state transition from G2 to dormant as defined in the Arm Debug Interface Architecture Specification.

Implications

The JTAG TAP cannot be reset into the Test-Logic-Reset state when the SWJ-DP is in dormant state. This means that when sharing a wire with another 1149.7 device, all devices cannot be placed into their reset state simultaneously.

Workaround

Push the JTAG TAP into Test-Logic-Reset either:

- Using the nTRST pin if available.
- Switching to the JTAG protocol and sending at least five SWCLKTCK cycles with SWDIOTMS HIGH.

857433

Pending serious faults are not always indicated

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Fixed in r0p2.

Description

When an exception is traced, if a serious fault is also pending, then this should be indicated in the exception type. Because of this erratum, if the exception handler is in a Secure region and Secure trace is not permitted, then the exception will be traced but the exception type will indicate that no serious fault is pending.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor with the Security Extension present and the ETM present.

Conditions

- Trace is active in a Non-secure region of code.
- Secure non-invasive debug is disabled.
- An exception is taken at a higher priority than a pending BusFault, HardFault, MemManage fault, or SecureFault.
- The exception handler is in Secure state.

Implications

The information about the pending serious fault will not be traced at the expected time. Depending on the authentication and filtering of trace, the serious fault itself will be traced correctly if it is handled later. The pending serious fault can also be traced along with a further exception, provided that the handler is not in a trace prohibited region.

Workaround

There is no workaround.

861432

DebugMonitor exception exit when IPSR is corrupted fails to step instruction

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Fixed in r0p2.

Description

During an exception exit of a DebugMonitor handler when stepping is enabled, the stack frame is corrupted so that the IPSR value read back is 12 (the exception number for DebugMonitor). The core will behave as if it is still in the DebugMonitor handler and disable stepping.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor.

Conditions

- Halting debug is not enabled, DHCSR.C_DEBUGEN=0.
- DebugMonitor stepping is enabled, DEMCR.MON_EN=1 and DEMCR.MON_STEP=1.
- An exception exit happens from the DebugMonitor handler.
- The IPSR value restored from the stack has to be corrupted so it is still pointing to the DebugMonitor even though it is no longer active.
- Instructions will not be stepped while the IPSR remains set to the DebugMonitor.

Implications

When the DebugMonitor is set to stepping and Halting debug is not enabled, the instructions will not be stepped when the IPSR is corrupted. The IPSR would be corrupted from exiting the DebugMonitor while remaining set to 0xC. The consequence is that monitor stepping is disabled.

Workaround

There is no workaround.

875823

Some T32 unallocated hint encodings UNDEF rather than NOP

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Fixed in r0p2.

Description

The Armv8-M architecture defines a hint instruction class within the T32 opcode space, which includes the implemented instructions NOP, YIELD, WFE, WFI, and SEV. Any unimplemented encoding in this space is treated as a reserved hint, which should behave like a NOP.

However, for the encodings with hint[3:0] = 0000 and option[3:0] = 0101-0111, which correspond to reserved hints, the PE will generate an UNDEFINSTR UsageFault rather than the architecturally specified NOP.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor.

Conditions

The processor executes the following instruction encodings. The value of the bits enclosed in parentheses "( )" are ignored for the purposes of generating the UNDEFINSTR UsageFault.

```
1111 0011 1010 (1)(1)(1)(1) 1000 (0)000 0000 0101
1111 0011 1010 (1)(1)(1)(1) 1000 (0)000 0000 0110
1111 0011 1010 (1)(1)(1)(1) 1000 (0)000 0000 0111
```

Implications

Because these encodings are unallocated encodings that a compiler will not generate, they will not be encountered in code generated by a toolchain. Manual insertion of these encodings in assembly will result in the erroneous behavior described above when the encodings are executed.

Workaround

There is no workaround.

937163

Floating-point state can be incorrectly cleared on some exception return faults

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0, r0p1, r0p2 and r0p3. Fixed in r0p4.

Description

The Armv8-M architecture defines integrity checks which are performed before the exception return unstacking occurs. These check the validity of the EXC_RETURN value and raise a fault if they fail. Because of this erratum it is possible for the floating-point state to be incorrectly cleared when one of these faults occurs.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor with the Floating Point Unit and the Security Extension included.

Conditions

The floating-point state will be incorrectly cleared when all the following conditions are met:

1. One of the following exception return integrity checks fails:
    - SFSR.INVER.
    - UFSR.INVPC (exiting a handler that is not active).
    - UFSR.INVPC (EXC_RETURN[1]!=0).
    - SFSR.LSERR (when attempting to clear because of FPCCR.CLRONRET).
2. The floating-point state would have been unstacked if there had been no fault (that is, EXC_RETURN[4]==0, FPCCR.LSPACT==0 and access is permitted to the FPU).

Implications

The floating-point state can be incorrectly cleared if software causes one of the faults mentioned above. The scenario that could be problematic is when a Secure exception calls a Non-secure function, which in turn attempts to return from the exception. This erratum allows the Non-secure function to clear the Secure floating-point context. Note that doing so will always cause a Secure fault to be raised and no Secure state is ever leaked to Non-secure.

Workaround

This erratum is not expected to require a workaround.

1015127

Processor might not wake up to a SEVONPEND event when in WIC-based WFE sleep

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0, r0p1, r0p2 and r0p3. Fixed in r0p4.

Description

The Armv8-M architecture includes a feature which allows an event to be sent when an interrupt state changes from inactive to pending (SEVONPEND). The Cortex-M33 processor also includes a Wakeup Interrupt Controller (WIC) to enable the processor to enter a low-power state. Because of this erratum, when in WIC-based WFE sleep, it is possible that the processor will fail to wake up as expected because a pended interrupt does not generate the expected event.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor with the Security Extension and a WIC included.

Conditions

- WIC-based sleep enabled (SCR.SLEEPDEEP==1, WICENACK==1).
- Only one of the banked SCR.SEVONPEND bits is set.
- The processor enters WFE sleep in the security state where the associated banked SCR.SEVONPEND field is not set.

Implications

The WIC will not wake up to an event generated by a pending interrupt targeting the alternate security state where the associated SCR.SEVONPEND bit is 1. However, it is expected that a system will not be affected by this behavior since software cannot depend on a wake-up event controlled by the alternate security state.

Workaround

This erratum is not expected to require a workaround.

1080541

Access permission faults are prioritized over unaligned Device memory faults

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0, r0p1, r0p2, r0p3, r0p4, and r1p0. Open.

Description

A load or store which causes an unaligned access to Device memory will result in an UNALIGNED UsageFault exception. However, if the region is not accessible because of the MPU access permissions (as specified in MPU_RBAR.AP), then the resulting MemManage fault will be prioritized over the UsageFault.

Configurations affected

This erratum affects all configurations of the Cortex-M33 processor with the MPU enabled.

Conditions

The MPU is enabled and:

- A load/store access occurs to an address which is not aligned to the data type specified in the instruction.
- The memory access hits one region only.
- The region attributes (specified in the MAIR register) mark the location as Device memory.
- The region access permissions prevent the access (that is, unprivileged or write not allowed).

Implications

The MemManage fault caused by the access permission violation will be prioritized over the UNALIGNED UsageFault exception because of the memory attributes.

Workaround

There is no workaround.
However, it is expected that no existing software is relying on this behavior since it was permitted in Armv7-M.

1113997

Group priority of a Non-secure interrupt might be incorrect when AIRCR.PRIS is set

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0, r0p1, r0p2, r0p3, r0p4, and r1p0. Open.

Description

When the processor is configured with Security extension and AIRCR.PRIS is 1, the Armv8-M architecture requires that the priorities of Non-secure interrupts are modified to ensure that Secure interrupts are prioritized over Non-secure interrupts. The Armv8-M architecture requires that lower priority numbers take precedence over higher priority numbers.

Because of this erratum, a Non-secure interrupt with higher priority number might be handled in the wrong order compared to another Non-secure or Secure interrupt.

Configuration affected

This erratum only affects the processor with Security extension configured and Verilog parameter IRQLVL set to a value less than 8. IRQLVL is the number of priority bits physically implemented within the 8-bit priority value. For IRQLVL less than 8, the least significant bits of the priority field are zeroed.

The erratum can occur if any of the following is true:

- The configuration includes an interrupt number of 240 or higher.
- The configuration uses the IRQLATENCY parameter to define interrupts of lower and higher interrupt latency.
- The Secure software configuration defines some Secure and some Non-secure interrupts.

Conditions

- AIRCR.PRIGROUP is set to a value that makes the Non-secure group priority field use all implemented IRQLVL bits.
- AIRCR.PRIS is equal to 1.
- A Non-secure interrupt IRQ has its least significant group priority bit, NVIC_IPRn.PRI_Nm[8-IRQLVL], set to 1, where the interrupt number is 4n+m

The least significant bit of the group priority might be treated as zero because of this erratum. This increases the apparent group priority to the next higher level.

This might cause incorrect prioritization when this value is compared against:

- Another Non-secure IRQ in the opposite half of the IRQ range of 480 IRQs, if neither IRQ has IRQLATENCY set for low latency
- Another Non-secure IRQ with the opposite IRQLATENCY setting.
- A Secure IRQ whose Secure group priority matches the erroneous priority level of the Non-secure IRQ adjusted by applying the PRIS function. This can only be true for low priority Secure IRQs with a priority number of 0x80 or higher, so the erratum does not affect Secure IRQs with priority numbers 0x00 to 0x7F.

Implications

The erroneously increased group priority level might become equal to the real group priority level of another interrupt, and therefore might prevent preemption that should have occurred, or might alter the tail-chaining order of multiple pending interrupts based on the subsidiary prioritizations of sub-priority or interrupt number instead of the actual group priority difference.

Workaround

There is no workaround for this erratum.

1367266

DFSR.EXTERNAL is not set correctly when waking up from sleep

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0, r0p1, r0p2, r0p3, r0p4, and r1p0. Open.

Description

An external debug event which causes the processor to enter Debug state or the debug monitor should set DFSR.EXTERNAL. It has been found that this field is not set if the event occurs while the processor is asleep.

Configurations affected

This erratum affects all configurations of the Cortex-M33.

Conditions

- The processor is asleep.
- An external debug event occurs which would cause the processor to enter Debug state (when Halting debug is enabled) or to take a DebugMonitor exception (when Halting debug is not enabled or not present).

Implications

DFSR.EXTERNAL status field is not set as expected. As a result, software cannot determine information about the debug event from the DFSR register.

Workaround

There is no workaround.

1435973

Execution priority might be wrong for one cycle after AIRCR, NVIC_ITNS, NVIC_IPR, NVIC_ISER, or NVIC_ICER is changed

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0, r0p1, r0p2, r0p3, r0p4, and r1p0. Open.

Description

Programmable registers are used in the *Nested Vectored Interrupt Controller* (NVIC) to determine execution priority, interrupt priority, and in turn exception pre-emption, fault escalation and other NVIC behavior.

As a result of this erratum, when at least one interrupt is configured as higher-priority and specific programmable register values are changed, there is a one-cycle window where execution and interrupt priority might be wrong, leading to incorrect NVIC behavior such as exception pre-emption and fault escalation.

Configuration affected

This erratum affects all configurations including high latency interrupts, (Verilog parameter IRQTIER != 0).

Conditions

This erratum occurs when the following conditions are met:

- At least one high latency IRQ is active or pending. The corresponding bit in the Verilog parameter IRQTIER is 1
- A write is carried out to the AIRCR register or to one of the following registers associated with an active or pending high latency interrupt
    - NVIC_ITNSn : Interrupt security target
    - NVIC_IPRn : Interrupt priority
    - NVIC_ISERn : Interrupt set enable
    - NVIC_ICERn : Interrupt clear enable

Implications

This erratum can lead to the following incorrect NVIC behavior:

- The handler of a pending and enabled interrupt might be entered erroneously.
- Synchronous fault escalation to HardFault might be wrong when a synchronous fault is raised in the cycle after the write to the NVIC-related register.
- The UFRDY bits in FPCCR might be wrong when the write to the NVIC-related register is followed immediately by a VLSTM instruction.
- Instruction stepping, asynchronous debug events, and breakpoints might be incorrectly triggered or missed in the cycle after the write to the NVIC-related register.

Workaround

There is no workaround for this erratum.

Typical applications do not need a workaround for this erratum because registers related to interrupt priority are typically programmed during boot-up and then remain static.

1453380

Non-secure HardFault exception might preempt when disabled by AIRCR.BFHFNMINS

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0, r0p1, r0p2, r0p3, r0p4, and r1p0. Open.

Description

When the processor implements the Security Extension and AIRCR.BFHFNMINS is 1, the Non-secure banked version of SHCSR.HARDFAULTPENDED can be set to 1. This Non-secure pended HardFault might not preempt per architecture because it does not have enough priority (that is, the processor is in HardFault handler mode). If AIRCR.BFHFNMINS is subsequently changed to 0 with the Non-secure HardFault still pending, then the architecture requires that the Non-secure HardFault should never preempt regardless of execution priority.

Because of this erratum, the pended Non-secure HardFault exception preempts when AIRCR.BFHFNMINS is 0 and current execution priority is larger than -1 (Non-secure HardFault having higher priority).

Configuration affected

This erratum affects all processor configurations where the Security Extension is implemented.

Conditions

The Non-secure banked version of SHCSR.HARDFAULTPENDED is set to 1 when AIRCR.BFHFNMIS is changed from 1 to 0.

Implications

Non-secure HardFault handler might be erroneously executed when the programmer explicitly disables the Non-secure HardFault by changing AIRCR.BFHFNIMS from 1 to 0. However, the sequence to trigger this is highly unlikely to be present in rational programs as it is implausible for any program to explicitly pend the Non-secure HardFault and then disable it.

Workaround

There is no workaround for this erratum.

1540599

Sorting of pending interrupts might be wrong when high latency IRQs are pending

Status

Affects: Cortex-M33
Fault Type: Programmer Category C
Fault Status: Present in r0p0, r0p1, r0p2, r0p3, r0p4, and r1p0. Open.

Description

The NVIC contains a pending tree which sorts all pending and enabled interrupts based on priorities. If DHCSR.C_DEBUGEN and DHCSR.C_MASKINTS are 1, DHCSR.S_SDE is 0 and halting debug is allowed, then Non-secure PendSV, Non-secure SysTick, and Non-secure IRQs should be masked off and they should not affect the sorting of pending and enabled secure interrupts.
If multiple high latency IRQs are pending and enabled with different security targets and priorities, then Non-secure IRQs which should be masked off might cause the pending tree output to be a pending Secure interrupt without highest priority. This is because of incorrect masking before doing priority comparisons in the tree.

Configuration affected

This erratum affects all processor configurations where the Security Extension is implemented.

Conditions

- HaltingDebugAllowed() == True.
- DHCSR is programmed to mask only Non-secure IRQ.
- There is at least one Non-secure high latency IRQ pending and enabled with higher priority than any pending and enabled Secure IRQ.
- There is at least one Secure high latency IRQ pending and enabled with higher priority than any pending and enabled Secure low latency IRQ.
- There is at least one Secure low latency IRQ pending and enabled.

Implications

1. ICSR.VECTPENDING might contain a pending and enabled Secure low latency IRQ that does not have the highest priority.
2. The handler of a Secure pending and enabled low latency IRQ might be entered before that of a pending and enabled Secure high latency IRQ with higher priority.

Workaround

There is no workaround for this erratum.