Chip Errata for i.MX RT1160

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
ERR051091	ADC: IO leakage current is observed when pins configured for ADC function
ERR050579	CAAM: When Key Form is set to “form #4: p, q, dp, dq, cr, rrp, rrq” in the format of the PROTINFO field for the RSA Decrypt Protocol, CAAM module can generate an error in some conditions
ERR006941	Core: Asynchronous sampling of SWDIOTMS might cause incorrect operation of SerialWire/JTAG Debug Port
ERR006939	Core: Interrupted loads to SP can cause erroneous behavior
ERR011573	Core: Speculative accesses might be performed to memory unmapped in MPU.
ERR009005	Core: Store immediate overlapping exception return operation might vector to incorrect interrupt
ERR006940	Core: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used
ERR050708	Debug: CoreSight components are not linked to CoreSight ROM table
ERR050458	FLEXIO: Shifter Status/Error flag not generated correctly in Logic Mode
ERR011377	FlexSPI: DLL lock status bit not accurate due to timing issue
ERR050643	GPIO: During initial power-up, a brief pull-up pulse could occur on the port pins
ERR003777	GPT: Possibility of additional pulse on src_clk when switching between clock sources
ERR050606	LPSPI: TCR value does not get resampled when polling the register
ERR050607	LPSPI: TCR[FRAMSZ] can be ignored when TCR[TXMSK]=1b1
ERR050634	OCRAM: No ECC interrupt for CM7
ERR050659	QDC: A possible speed measurement issue when CTRL3[PMEN]=1
ERR050790	ROM: ROM does not support SEMC_DCCR register configuration from DCD
ERR050144	SAI: Setting FCONT=1 when TMR>0 may not function correctly
ERR051122	SNVS: Some fuse trim values are lost in SNVS mode
ERR050396	SOC: Sparse write to CM7 TCM causes data corruption

Table 2. Revision History

Revision	Changes
1.0	Initial revision
1.1	The following errata were added. • ERR051091 • ERR051122 The following erratum was revised. • ERR006941

ERR051091: ADC: IO leakage current is observed when pins configured for ADC function

Description: Under corner conditions, a leakage current of up to 3.5 uA may be observed flowing out from the I/O pads. When a pin is used for an ADC function, the leakage current affects the ADC conversion accuracy.

Workaround: 1. Ensure ADC external input resistance value is as small as possible. This will minimize the voltage deviation generated by the leakage current (3.5 μA). For example, with a 12-bit ADC and 1.8 V reference voltage, an external resistance less than 125 ohms keeps the voltage deviation (due to leakage current) less than 1 LSB.

2. Limit the ADC input voltage to less than 1 V.

ERR050579: CAAM: When Key Form is set to “form #4: p, q, dp, dq, cr, rrp, rrq” in the format of the PROTINFO field for the RSA Decrypt Protocol, CAAM module can generate an error in some conditions

Description: In the format of the PROTINFO field for the RSA Decrypt Protocol, the Key Form can be set 4 options such as “form #1: n, d “, “form #2: p, q, d”, “form #3: p, q, dp, dq, c” and “form #4: p, q, dp, dq, cr, rrp, rrq”. Using the form #4 option can generate errors if the size of p <= 128 Bytes and 128 Bytes < the size of q <= 256 Bytes.

Workaround: To avoid the error generated by CAAM module, please select key form #3 instead of key form #4 in the format of the PROTINFO field for the RSA Decrypt Protocol.

ERR006941: Core: Asynchronous sampling of SWDIOTMS might cause incorrect operation of SerialWire/JTAG Debug Port

Description: Arm Errata 771919: Asynchronous sampling of SWDIOTMS might cause incorrect operation of

SerialWire/JTAG Debug Port

Status

Affects: Cortex-M4, Cortex-M4F

Fault Type: Implementation Category B Rare

Fault Status: Present in: r0p0, r0p1 Open.

Description

The signal SWDIOTMS is bi-directional and can be driven from either the debugger or the SWJ-DP, or pulled up

by an external resistor during the turnaround periods.

The SerialWire protocol is defined with a high PARK bit at the end of the header before the turnaround period

that precedes the ACK from the SWJ-DP. This ensures that the line is high, and the resistor keeps it high during

the ACK period. Therefore, if the SWJ-DP does not respond, the debugger will reliably sample the line

SWDIOTMS high during the missing ACK.

However, during the turnaround period after the ACK or read data there is no PARK bit to guarantee that the

line is high immediately before the turnaround period. In this case, if the pull-up resistor does not pull the line

high within a single SWCLKTCK cycle, the incorrect state of SWDIOTMS might be sampled.

Functionally, the logic is insensitive to the state of SWDIOTMS during these periods, but synthesis tools might

introduce multiple path logic that is sensitive to SWDIOTMS glitches around the clock edges.

Conditions

All write transactions and some read transactions might be vulnerable to this erratum when both:

- Serial Wire mode is being used
- The physical implementation does not prevent glitch generation.

Implications

The SWJ-DP might sample SWDIOTMS incorrectly and enter an UNPREDICTABLE state. At the time of

publication, ARM is not aware of any reports of observed failures due to this erratum.

Workaround: Check the following points after implementation:

1) Ensure that the evaluation of NextState in DAPSwjWatcher.v is not sensitive to SWDITMSSync1 when

State_cdc_check has the value 10'b1100100000 (SWJ_SSLP).

2) Ensure that the following logic in DAPSwDpProtocol.v is implemented using AND gates or a CDC-safe mux

for each bit:

assign ResetCountD = DBGDI & ~DBGDOEN ? (ResetCountReg+6'd1) : {6{1'b0}};

3) Ensure that the ResetCountReg flops in DAPSwDpProtocol.v are implemented using metastability-hardened

cells if possible.

4) Ensure that the evaluation of NxtState in DAPSwDpProtocol.v is insensitive to DBGDI when State has any of

the following values:

- 5'b01000 (SWDP_SLEPARKH)
- 5'b01010 (SWDP_SLETRNH2)
- 5'b01011 (SWDP_SLETRNH1)
- 5'b01100 (SWDP_SLETRNH0)
- 5'b10011 (SWDP_SLEPARKW)
- 5'b10100 (SWDP_SLETRNW3)
- 5'b10101 (SWDP_SLETRNW2)
- 5'b10110 (SWDP_SLETRNW1)
- 5'b10111 (SWDP_SLETRNW0)

5) Ensure that the following flops in DAPSwDpProtocol.v are implemented with CDC-safe recirculation muxes:

- SerBank
- SerDir
- SerAddr
- ShiftReg
- Parity
- ErrorChk
- WriteErr
- WbufReq

6) Ensure that the following flops in DAPJtagDpProtocol are implemented with CDC-safe recirculation muxes:

- JTAGcurr

ERR006939: Core: Interrupted loads to SP can cause erroneous behavior

Description: Arm Errata 752770: Interrupted loads to SP can cause erroneous behavior

This issue is more prevalent for user code written to manipulate the stack. Most compilers will not be affected by this, but please confirm this with your compiler vendor. MQX™ and FreeRTOS™ are not affected by this issue.

Affects: Cortex-M4, Cortex-M4F

Fault Type: Programmer Category B

Fault Status: Present in: r0p0, r0p1 Open.

If an interrupt occurs during the data-phase of a single word load to the stack-pointer (SP/R13), erroneous behavior can occur. In all cases, returning from the interrupt will result in the load instruction being executed an additional time. For all instructions performing an update to the base register, the base register will be erroneously updated on each execution, resulting in the stack-pointer being loaded from an incorrect memory location.

The affected instructions that can result in the load transaction being repeated are:

1) LDR SP,[Rn],#imm

2) LDR SP,[Rn,#imm]!

3) LDR SP,[Rn,#imm]

4) LDR SP,[Rn]

5) LDR SP,[Rn,Rm]

The affected instructions that can result in the stack-pointer being loaded from an incorrect memory address are:

1) LDR SP,[Rn],#imm

2) LDR SP,[Rn,#imm]!

Conditions:

1) An LDR is executed, with SP/R13 as the destination.

2) The address for the LDR is successfully issued to the memory system.

3) An interrupt is taken before the data has been returned and written to the stack-pointer.

Implications:

Unless the load is being performed to Device or Strongly-Ordered memory, there should be no implications from the repetition of the load. In the unlikely event that the load is being performed to Device or Strongly-Ordered memory, the repeated read can result in the final stack-pointer value being different than had only a single load been performed.

Interruption of the two write-back forms of the instruction can result in both the base register value and final stack-pointer value being incorrect. This can result in apparent stack corruption and subsequent unintended modification of memory.

Workaround: Most compilers are not affected by this, so a workaround is not required.

However, for hand-written assembly code to manipulate the stack, both issues may be worked around by replacing the direct load to the stack-pointer, with an intermediate load to a general-purpose register followed by a move to the stack-pointer.

If repeated reads are acceptable, then the base-update issue may be worked around by performing the stack pointer load without the base increment followed by a subsequent ADD or SUB instruction to perform the appropriate update to the base register.

ERR011573: Core: Speculative accesses might be performed to memory unmapped in MPU.

Description: Arm errata 1013783-B

Cortex-M7 can perform speculative memory accesses to Normal memory for various reasons. All other types of memory should never be subject to speculative accesses.

The memory attributes for a given address are defined by the settings of the MPU when it is enabled. Regions that are not mapped in the MPU do not have any explicit attributes and should not be subject to any speculative accesses.

Because of this erratum, Cortex-M7 can incorrectly perform speculative accesses to such unmapped regions.

Conditions:

To trigger this erratum, the data cache must be enabled and the MPU must be enabled with the default memory map disabled. That is:

- CCR.DC = 1; data cache is enabled.
- MPU_CTRL.ENABLE = 1; MPU is enabled.

• If MPU_CTRL.PRIVDEFNA = 1, then this erratum cannot occur from privileged mode.

• If MPU_CTRL.HFNMIENA = 1, then this erratum cannot occur from the NMI or HF handlers or exception handlers when FAULTMASK = 1.

In these situations, a PLD instruction targeting an unmapped region might result in an incorrect speculative access. The PLD instruction itself could be speculative because of branch prediction. Even a literal data value that corresponds to a PLD encoding could theoretically cause this issue. This makes it difficult to scan code to check if these conditions apply.

Therefore, Arm recommends that any software with the MPU and data cache configured as mentioned in the conditions above uses the workaround below.

Implications:

Processor execution is not directly affected by this erratum. The data returned from the speculative access is never used and if the access is inferred by the program, then an abort will be taken as required.

The only implications of this erratum are the access itself which should not have been performed. This might have an impact on memory regions with side-effects on reads or on memory which never returns a response on the bus.

Workaround: Instead of leaving memory unmapped, software should use MPU region 0 to cover all unmapped memory and make this region execute-never and inaccessible. That is, MPU_RASR0 should be programmed with:

• MPU_RASR0.ENABLE = 1; MPU region 0 enable.

• MPU_RASR0.SIZE = b11111; MPU region 0 size = 2^32 bytes to cover entire memory.

• MPU_RASR0.SRD = b00000000; All sub-regions enabled.

• MPU_RASR0.XN = 1; Execute-never to prevent instruction fetch.

• MPU_RASR0.AP = b000; No read or write access for any privilege level.

• MPU_RASR0.TEX = b000; Attributes = Strongly-ordered.

• MPU_RASR0.C = b0; Attributes = Strongly-ordered.

• MPU_RASR0.B = b0; Attributes = Strongly-ordered.

Note that the MPU supports addressing hitting in multiple regions with the highest numbered region taking priority.

Therefore, use of MPU region 0 in this way does not affect the existing organization and use of MPU regions.

ERR009005: Core: Store immediate overlapping exception return operation might vector to incorrect interrupt

Description: Arm Errata 838869: Store immediate overlapping exception return operation might vector to incorrect interrupt

Affects: Cortex-M4, Cortex-M4F

Fault Type: Programmer Category B Rare

Fault Status: Present in: r0p0, r0p1 Open.

The Cortex-M4 includes a write buffer that permits execution to continue while a store is waiting on the bus. Under specific timing conditions, during an exception return while this buffer is still in use by a store instruction, a late change in selection of the next interrupt to be taken might result in there being a mismatch between the interrupt acknowledged by the interrupt controller and the vector fetched by the processor.

Configurations Affected

This erratum only affects systems where writeable memory locations can exhibit more than one wait state.

Workaround: For software not using the memory protection unit, this erratum can be worked around by setting DISDEFWBUF in the Auxiliary Control Register.

In all other cases, the erratum can be avoided by ensuring a DSB occurs between the store and the BX instruction. For exception handlers written in C, this can be achieved by inserting the appropriate set of intrinsics or inline assembly just before the end of the interrupt function, for example:

ARMCC:

...

__schedule_barrier();

__asm{DSB};

__schedule_barrier();

}

GCC:

...

__asm volatile (“dsb 0xf” ::: “memory”);

}

ERR006940: Core: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

Description: Arm Errata 776924: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

Affects: Cortex-M4F

Fault Type: Programmer Category B

Fault Status: Present in: r0p0, r0p1 Open.

On Cortex-M4 with FPU, the VDIV and VSQRT instructions take 14 cycles to execute. When an interrupt is taken a VDIV or VSQRT instruction is not terminated, and completes its execution while the interrupt stacking occurs. If lazy context save of floating point state is enabled then the automatic stacking of the floating point context does not occur until a floating point instruction is executed inside the interrupt service routine.

Lazy context save is enabled by default. When it is enabled, the minimum time for the first instruction in the interrupt service routine to start executing is 12 cycles. In certain timing conditions, and if there is only one or two instructions inside the interrupt service routine, then the VDIV or VSQRT instruction might not write its result to the register bank or to the FPSCR.

Workaround: A workaround is only required if the floating point unit is present and enabled. A workaround is not required if the memory system inserts one or more wait states to every stack transaction.

There are two workarounds:

1) Disable lazy context save of floating point state by clearing LSPEN to 0 (bit 30 of the FPCCR at address 0xE000EF34).

2) Ensure that every interrupt service routine contains more than 2 instructions in addition to the exception return instruction.

ERR050708: Debug: CoreSight components are not linked to CoreSight ROM table

Description: There are CoreSight components integrated at 0xE004_4000 to 0xE004_8FFF including SWO, TSGEN, TPIU, ATB funnel and CTI. There is no CoreSight ROM integrated, so they are not discoverable through a ROM table. These components can still be used by accessing the corresponding addresses shown in RM.

Workaround: These components can still be used by accessing the corresponding addresses shown in RM. It's a compliance issue that they are not discoverable through a ROM table.

ERR050458: FLEXIO: Shifter Status/Error flag not generated correctly in Logic Mode

Description: Some shifters will not generate status or error flags correctly when configured for logic mode (SHIFTCTLn[SMOD] = 0b111). Shifters 0, 1, 2, and 3 behave correctly. All other shifters are affected.

Workaround: In logic mode, if the Status/Error flags are required, use shifter 0, 1, 2, or 3. If the Status/Error flag is not required, then any shifter could be used in logic mode.

ERR011377: FlexSPI: DLL lock status bit not accurate due to timing issue

Description: After configuring DLL and the lock status bit is set, still may get wrong data if immediately read/write from FLEXSPI based external flash due to timing issue

Workaround: Adding a delay time (equal or more than 512 FlexSPI root clock cycle) after the DLL lock status is set.

ERR050643: GPIO: During initial power-up, a brief pull-up pulse could occur on the port pins

Description: By default (reset state), the GPIO pins are in the high-Z state and typically stay high-Z until the application code changes its state. The internal pull-up and internal pull-down resistors are disabled by default.

During power up, the internal pull-up resistor on the GPIO_AD, GPIO_LPSR, and DISP_B2 pins may enable during the early part of the IO ramp up, resulting in a brief pull-up current pulse on some port pins that drops to zero before the VDD supplies reach the minimum operating voltage.

Workaround: A pulldown resistor (~10K) can be added to the GPIO pin(s) to minimize the peak voltage where the application is sensitive to potential pulses.

ERR003777: GPT: Possibility of additional pulse on src_clk when switching between clock sources

Description: There is a possibility of an extra pulse on SCLK in the GPT when switching between the clock sources.

Workaround: Changing the clock source should only be done when the GPT is disabled. A way to accomplished this is as follows:

Disable GPT—Write 1'b0 to EN bit of GPTCR

Disable interrupts—Write 6'b000000 in Bits [5:0] of GPTIR

Configure Output Mode to unconnected/ disconnected—Write zeros in OM3, OM2, OM1 in GPTCR

Disable Input Capture Modes—Write zeros in IM1,IM2 in GPTCR

Change clock source CLKSRC in GPTCR

Clear Status register—Write 003F in GPTSR

Set ENMOD in GPTCR

ENABLE GPT—Write 1'b1 to EN bit of GPTCR. The GPTSR should not be read immediately after changing the clock source (a wait of at least one SCLK is required).

ERR050606: LPSPI: TCR value does not get resampled when polling the register

Description: Reading the Transmit Command Register will return the current state of the command register.

Following a write to the TCR (Transmit Command register), if the user continuously reads the TCR (polls the register), then the read content no longer represents the contents of the Transmit Command register if it updates due to internal logic following the first read. The same value shall continue to be read.

Workaround: After reading the Transmit Command Register must always access a different register in between subsequent reads from TCR.

ERR050607: LPSPI: TCR[FRAMSZ] can be ignored when TCR[TXMSK]=1b1

Description: TCR (Transmit Command Register) is used to write new command word to the LPSPI transmit FIFO.

TCR[FRAMESZ] configures the frame size of the data to be transmitted in number of bits equal to (FRAMESZ + 1). When TCR[TXMSK] is set, transmit data is masked (no data is loaded from transmit FIFO and output pin is tristated). In master mode, the Transmit Data Mask bit will initiate a new transfer which cannot be aborted by another command word; the Transmit Data Mask bit will be cleared by hardware at the end of the transfer. TCR[CONTC] controls the continuous transfer mode. TCR[CONTC]=1b1 enables continuous transfer. In master mode, continuous transfer will keep the PCS asserted at the end of the frame size, until a command word is received that starts a new frame.

If command word is written with TCR[TXMSK]=1 and TCR[FRAMESZ]>32 and the next command word with TCR[CONTC]=0 is in the FIFO then at the end of any 32-bit word of the first command, the frame will terminate early and negate PCS.

Workaround: There are two workarounds:

1. Do not write a 2nd command word after writing command word with TXMSK=1 and FRAMESZ>32 until the first one has completed.

OR

2. Divide the command word into multiple command words with TXMSK=1 and FRAMESZ=32 (or remainder) using a continuous transfer.

ERR050634: OCRAM: No ECC interrupt for CM7

Description: OCRAM space from 0x2020_000 to 0x2023_FFFF cannot trigger ECC interrupt for CM7 core.

Workaround: This memory space cannot be used as ECC memory for CM7 core, while it can be used as normal memory.

ERR050659: QDC: A possible speed measurement issue when CTRL3[PMEN]=1

Description: When CTRL3[PMEN]=1, and reading POSD occurs simultaneously with any edge of phase A or phase B signal, the captured position difference value in POSDH register may not match the time period captured in POSDPERH register, which causes inaccuracy in speed measurement.

Workaround: No workaround.

ERR050790: ROM: ROM does not support SEMC_DCCR register configuration from DCD

Description: The ROM does not allow for writes to the SEMC_DCCR register. When the SEMC_DCCR is not written, the maximum SDRAM frequency at boot time is 166MHz.

Workaround: Use the DCD to configure SDRAM for a maximum operating frequency of 166MHz. If faster operation is required, then application code can be used to write the SEMC_DCCR value and then increase the SEMC_CLK_ROOT frequency (up to 200MHz). Please refer to the SDK for examples showing booting with a 166MHz SDRAM clock, and then increasing the clock to 200MHz.

ERR050144: SAI: Setting FCONT=1 when TMR>0 may not function correctly

Description: When FCONT=1 the transmitter will recover after a FIFO error when the FIFO is no longer empty and starting again from the same word in the following frame where the error occurred.

Configuring TMR > 0 will configure one or more words in the frame to be masked (nothing transmitted during that slot). If anything other than the last word(s) in the frame are masked when FCONT=1 and a FIFO Error Flag is set, then the transmitter will not recover and will set FIFO Error Flag during each frame.

Workaround: To avoid this issue, set FCONT in TCR4 to be 0.

ERR051122: SNVS: Some fuse trim values are lost in SNVS mode

Description: Some trim values will be lost when entering in SNVS mode. This may lead to missed and/or false triggers of voltage, clock, and temperature tampers. GPIO_SNVS_xx pins will automatically switch to tamper function in SNVS mode, even for the devices that do not support tamper and have configured the pins for their GPIO13 functions.

Workaround: 1. Disable voltage, clock, and temperature tampers before entering in SNVS mode. The tampers can be re-enabled after exiting SNVS.

2. Do not use GPIO_SNVS_xx pins as GPIO functions in SNVS mode.

ERR050396: SOC: Sparse write to CM7 TCM causes data corruption

Description: For some bus masters, writing access to CM7 TCM is handled by a NIC301 block which does not support sparse write conversion.

It results in a data corruption when sparse writing to CM7 TCM happens.

The affected bus masters includes CAAM, ENET_1G, ENET_QOS, GC355, LCDIFv2, PXP, uSDHC, ENET, and USB.

Workaround: For CAAM, ENET_1G, ENET_QOS, GC355, LCDIFv2, and PXP: Do not write TCM, use OCRAM or external RAM instead.

For uSDHC: If CM7 TCM is the destination for writing, IOMUXC_GPR_GPR28[AWCACHE_USDHC] should be cleared. If IOMUXC_GPR_GPR28[AWCACHE_USDHC] is set, write buffers must be placed in OCRAM or external RAM.

For ENET: If CM7 TCM is the destination for writing, IOMUXC_GPR_GPR28[CACHE_ENET] should be cleared. If IOMUXC_GPR_GPR28[CACHE_ENET] is set, write buffers must be placed in OCRAM or external RAM.

For USB: If CM7 TCM is the destination for writing, IOMUXC_GPR_GPR28[CACHE_USB] should be cleared. If IOMUXC_GPR_GPR28[CACHE_USB] is set, write buffers must be placed in OCRAM or external RAM.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.