Chip Errata for i.MX RT1010

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
ERR050143	CCM: SoC will enter low power mode before the ARM CPU executes WFI when improper low power sequence is used
ERR011573	Core: Speculative accesses might be performed to memory unmapped in MPU.
ERR006223	Failure to resume from WAIT/STOP mode with power gating
ERR011377	FlexSPI: DLL lock status bit not accurate due to timing issue
ERR050606	LPSPI: TCR value does not get resampled when polling the register
ERR050607	LPSPI: TCR[FRAMSZ] can be ignored when TCR[TXMSK]=1b1
ERR050130	PIT: Temporary incorrect value reported in LMTR64H register in lifetimer mode
ERR050538	SOC: Potential boot failure on system reset if SJC_DISABLE fuse is blown
ERR050327	SWRESET on flexspi keyblob fetching state machine will impact software driver

Table 2. Revision History

Revision	Changes
0	Initial revision
1	The following errata were added. • ERR011573 • ERR050538 • ERR050606 • ERR050607 The following erratum was revised. • ERR011377

ERR050143: CCM: SoC will enter low power mode before the ARM CPU executes WFI when improper low power sequence is used

Description: When software tries to enter the low power mode with the following sequence, SoC enters the low power mode before the ARM CPU executes the WFI instructions.

• Set CCM_CLPCR[1:0] to 2'b00

• ARM CPU enters WFI

• ARM CPU wakes up from an interrupt event, which is masked by GPC or not visible to GPC, such as an interrupt from local timer.

• Set CCM_CLPCR[1:0] to 2'b01 or 2'b10

• ARM CPU executes WFI

Before the last step, SoC enters the WAIT mode if CCM_CLPCR[1:0] is set to 2'b01, or enters the STOP mode if CCM_CLPCR[1:0] is set to 2'b10.

Workaround: Software workaround

1) Trigger IRQ #41 (IOMUX), which is always pending by setting IOMUX_GPR1_GINT bit

2) Unmask IRQ #41 in GPC before setting the CCM low power mode

3) Mask IRQ #41 right after the CCM low power mode is set (set bit0-1 of CCM_CLPCR)

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.

ERR006223: Failure to resume from WAIT/STOP mode with power gating

Description: When entering WAIT/STOP mode with power gating of the core(s), if an interrupt arrives during the power down sequence, the system could enter an unexpected state and fail to resume.

Workaround: Use REG_BYPASS_COUNTER (RBC) to hold off interrupts when the PGC unit is in the middle of power down sequence. The counter needs to be set/cleared only when no interrupts pending. To use the work around effectively, the counter needs to be enabled as close to WFI as possible.

Following equation can be used to aid determination of RBC counter value.

RBC_COUNT * (1 / 32K RTC Frequency) >= (25 + PDNSCR_SW2ISO) * (1 / IPG_CLK Frequency)

PDNSCR_ISO2SW = PDNSCR_ISO = 1 (counts in IPG_CLK clock domain)

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.

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.

ERR050130: PIT: Temporary incorrect value reported in LMTR64H register in lifetimer mode

Description: When the Programmable interrupt timer (PIT) module is used in lifetimer mode, timer 0 and timer 1 are chained and the timer load start value (LDVAL0[TSV] and LDVAL1[TSV]) are set according to the application need for both timers. When timer 0 current time value (CVAL0[TVL]) reaches 0x0 and subsequently reloads to LDVAL0[TSV], then timer 1 CVAL1[TVL] should decrement by 0x1.

However this decrement does not occur until one cycle later, therefore a read of the PIT upper lifetime timer register (LTMR64H) is followed by a read of the PIT lower lifetime timer register (LTMR64L) at the instant when timer 0 has reloaded to LDVAL0[TSV] and timer 1 is yet to be decremented in next cycle then an incorrect timer value in LTMR64H[LTH] is expected.

Workaround: In lifetimer mode if the read value of LTMR64L[LTL] is equal to LDVAL0[TSV] then read both LTMR64H and LTMR64L registers one additional time to obtain the correct lifetime value.

ERR050538: SOC: Potential boot failure on system reset if SJC_DISABLE fuse is blown

Description: By default, the JTAG/SWD clock is pulled high reset. When the SJC_DISABLE fuse is blown the clock is low. The fuses are reloaded during a system reset, so there is a window between the system reset assertion and fuse loading completion, during which the clock is high. When the fuses are loaded the clock will go low, causing a transition from high to low. There is another clock transition from low to high on a subsequent system reset. The clock toggles can cause the system to think JTAG/SWD is active. This causes a security violation leading to HAB boot failure.

Workaround: Configure the appropriate IOMUXC_SW_PAD_CTL register's PUE and PUS fields to enable a pull resistor on one of the following signals:

- Pull JTAG_TCK/SWD_CLK low
- Pull JTAG_TRST low
- Pull JTAG_TMS/SWD_DIO high

The IOMUXC registers retain state on a system reset, so this only needs to be done one time after each POR.

ERR050327: SWRESET on flexspi keyblob fetching state machine will impact software driver

Description: In this version, OTFAD keyblob done/error bit can be cleared by flexspi SWRESET.

When keyblob done/error status bit is 0, AHB bus access can be blocked since FlexSPI needs to wait OTFAD finishes its key blob processing.

But in some software drivers, SW reset is used after every flash erase/program. This makes FlexSPI lost its keyblob processing result and AHB bus error shows up in following AHB bus access.

Workaround: Using AHBCR[CLRAHBXBUF] instead of SW reset after page program or erase.

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.