Chip Errata for the i.MX RT1020

This document details the silicon errata known at the time of publication for the i.MX RT1020 multimedia crossover processors.

Table 1 provides a revision history for this document.

Table 1. Document Revision History

Rev. Number	Date	Substantive Changes

Rev. 2	01/2021	• Added the following errata:
		  – ERR050143
		  – ERR050577
		• Removed the following erratum:
		  – ERR007265 (replace ERR007265 with ERR050143)
		• Updated the Figure 1, "Revision Level to Part Marking Cross-Reference,"

Rev. 1	06/2019	• Added the following errata:
		  – ERR011572
		  – ERR050101
		  – ERR050130
		  – ERR050144
		  – ERR050194
		• Updated the Figure 1, "Revision Level to Part Marking Cross-Reference,"

Rev. 0	06/2018	• Initial version

Figure 1 provides a cross-reference to match the revision code to the revision level marked on the device.
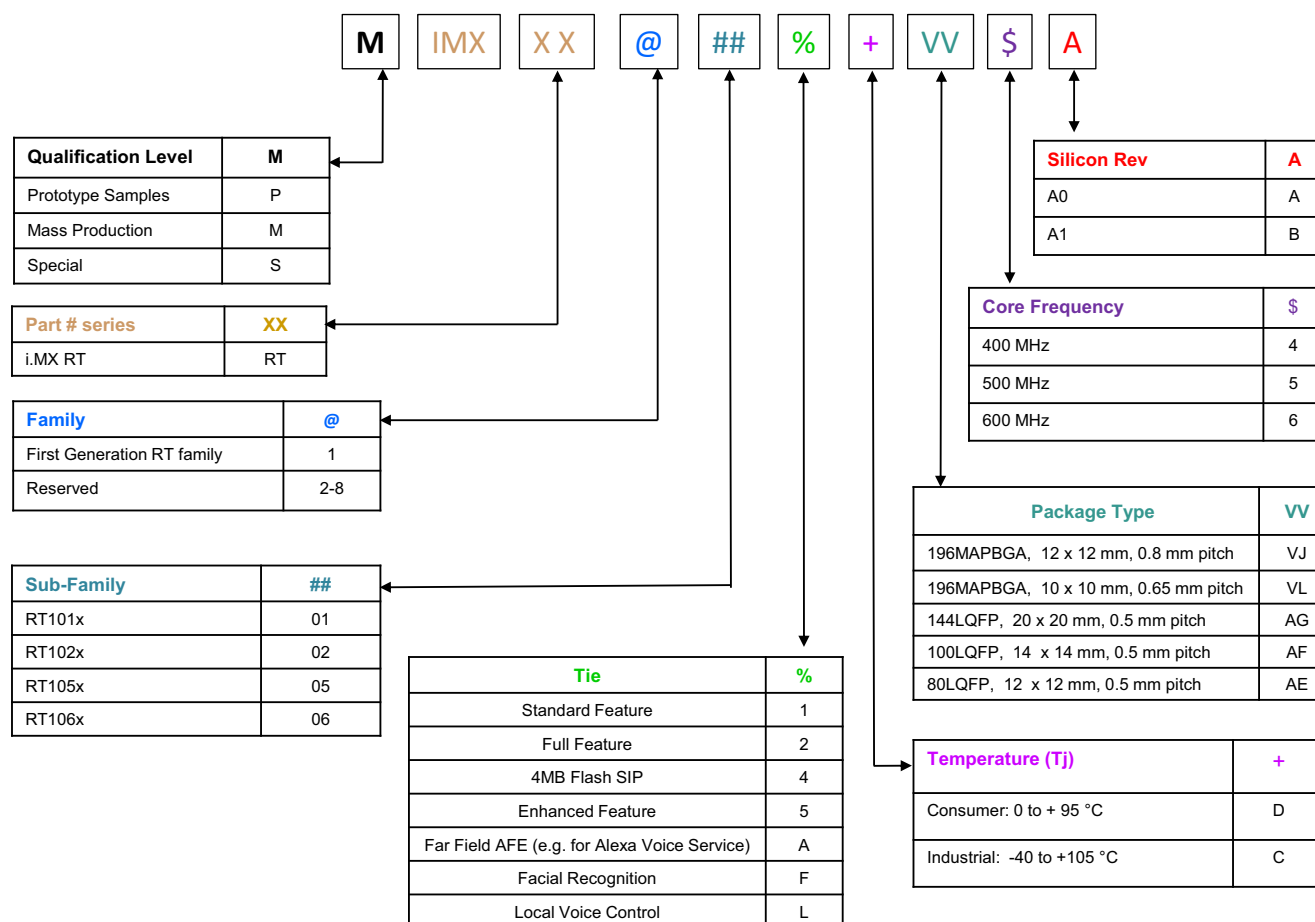

Figure 1. Revision Level to Part Marking Cross-Reference

For details on the Arm® configuration used on this chip (including Arm module revisions), please see the "Platform configuration" section of the "Arm Cortex®-M7 Platform" chapter of the *i.MX RT1020 Crossover Processor Reference Manual* (IMXRT1020RM).

summarizes errata on the i.MX RT1020.

Table 2. Summary of Silicon Errata

Errata	Name	Solution	Page
ADC			
ERR011164	ADC: ADC_ETC fails to clear the ADC_ETC request signals automatically after receiving DMA ack	No fix scheduled	5
CCM			
ERR006223	CCM: Failure to resume from Wait/Stop mode with power gating	No fix scheduled	6
ERR050143	CCM: SoC will enter low power mode before the Arm CPU executes WFI when improper low power sequence is used	No fix scheduled	7
Cortex-M7			
ERR011572	Cortex-M7: Write-Trough stores and loads may return incorrect data	No fix scheduled	8
FlexCAN			
ERR005829	FlexCAN: FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment during the arbitration process	No fix scheduled	9
ERR006032	FlexCAN: A frame with wrong ID or payload is transmitted into the CAN bus when the Message Buffer under transmission is either aborted or deactivated while the CAN bus is in the Bus Idle state	No fix scheduled	11
ERR009527	FlexCAN: The transmission abort mechanism may not work properly	No fix scheduled	13
ERR009595	FlexCAN: Corrupt frame possible if the Freeze Mode or the Low-Power Mode are entered during a Bus-Off state	No fix scheduled	14
FlexSPI			
ERR011207	FlexSPI: When FLEXSPI_AHBCR[PREFETCHEN] is set, incorrect data can be returned in rare conditions	No fix scheduled	16
ERR011377	FlexSPI: FlexSPI DLL lock status bit not accurate due to timing issue	No fix scheduled	17
PIT			
ERR050130	PIT: Temporary incorrect value reported in LMTR64H register in lifetimer mode	No fix scheduled	18
QTMR			
ERR050194	QTMR: Overflow flag and related interrupt cannot be generated when the timer is configured as upward count mode	No fix scheduled	19
SAI			
ERR011096	SAI: The internal bit clock cannot be generated when BCI = 1	No fix scheduled	20
ERR011150	SAI: Internally generated receive or transmit BCLK cannot be re-enabled if it is first disabled when RCR2[DIV] or TCR2[DIV] > 0	No fix scheduled	21
ERR050144	SAI: Setting FCONT = 1 when TMR > 0 may not function correctly	No fix scheduled	22

Table 2. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
SEMC			
ERR011225	SEMC: CPU AXI writes to SEMC NAND memory may cause incorrect data programmed into NAND memory	No fix scheduled	23
ERR050577	SEMC: Auto-refresh can fail to be triggered during long back to back write (or read)	No fix scheduled	24
SNVS			
ERR011165	SNVS: Invalid ECC check failure	No fix scheduled	25
USB			
ERR006281	USB: Incorrect DP/DN state when only VBUS is applied	No fix scheduled	26
ERR050101	USB: Endpoint conflict issue in device mode	Fix in next revision	27

ERR011164 ADC: ADC_ETC fails to clear the ADC_ETC request signals automatically after receiving DMA ack

Description:

If enable ADC_ETC to trigger DMA transfer, the DMA transfer data send ack out. It does not clear the request signals automatically and continue to trigger DMA.

Projected Impact:

This issue can lead to DMA failure when working with ADC_ETC.

Workarounds:

Configuring two DMA channels for ADC_ETC data transfer. The first DMA channel with low priority triggered by ADC_ETC request is to transfer ADC_ETC data. The second DMA channel with high priority links to the first channel is to clear request of ADC_ETC by writing DMA_CTRL register. Both channel's priority need to be higher than any channel used by other peripherals. This solution is result in DMA to transfer ADC_ETC data twice for one request signal and application need handle the redundant data properly.

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

ERR006223 CCM: Failure to resume from Wait/Stop mode with power gating

Description:

When entering Wait/Stop mode with power gating of the Arm core(s), if an interrupt arrives during the power-down sequence, the system could enter an unexpected state and fail to resume.

Projected Impact:

Device might fail to resume from low-power state.

Workarounds:

Use REG_BYPASS_COUNTER (RBC) to hold off interrupts when the PGC unit is in the middle of the power-down sequence. The counter needs to be set/cleared only when there are no interrupts pending. The counter needs to be enabled as close to the WFI (Wait For Interrupt) state as possible.

The following equation can be used to aid determination of the RBC counter value:

The PREG_BYPASS_COUNT value is equal or greater than 2.

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is in SDK.

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.

Workarounds:

Software workaround:

1. Trigger IRQ #41 (IOMUX), which is always pending by setting IOMUXC_GPR_GPR1[12] 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 CCM_CLPCR[1:0])

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is in SDK

ERR011572   Cortex-M7: Write-Trough stores and loads may return incorrect data

Description:

Arm errata 1259864

If a particular sequence of stores and loads is performed on the Cortex-M7 core to Write-Through memory, and some timing-based internal conditions are met, then a load may not have the last data stored to that address.

This erratum can only occur if the loads and stores are to Write-Through memory. The following methods enable write-through mode of the cache:

1. The Memory Protection Unit (MPU) has been programmed to set this address as Write-Through.
2. The default memory map is being used, and this address is Write-Through in the default memory map.
3. The memory is cacheable, and the CM7_CACR.FORCEWT bit is set.
4. The memory is cacheable, shared, and the CM7_CACR.SIWT bit is set.

Following sequence is required for this erratum to occur:

1. The address of interest must be in the data cache.
2. A Write-Through store is performed to the same double-word as the address of interest.
3. One of the following:
• A linefill is started (to a different cacheline to the address of interest) that allocates to the same set and way as the address of interest.
• An Error Correcting Code (ECC) error is observed anywhere in the data cache.
• A data cache maintenance operation without a following Data Synchronization Barrier (DSB).
4. A store to the address of interest.
5. A load to the address of interest.

If certain specific timing conditions are met, the load get the data from the first store, or from what was in the cache at the start of the sequence instead of the data from the second store.

Under these conditions, a load can return incorrect data.

Workarounds:

There is no direct workaround for this erratum.

Where possible, Arm is recommended that using the MPU to change the attributes on any Write-Through memory to Write-Back memory. If this is not possible, it might be necessary to disable the cache for sections of code that access Write-Through memory.

Proposed Solution:

No fix scheduled

ERR005829 FlexCAN: FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment during the arbitration process

Description:

FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment during the arbitration process. The following conditions are necessary for the issue to occur:

- Only one message buffer is configured to be transmitted.
- The write which enables the message buffer to be transmitted (write on Control/Status word) happens during a specific clock during the arbitration process.
- After this arbitration process occurs, the bus goes to the Idle state and no new message is received on the bus.

For example:

1. Message buffer 13 is deactivated on RxIntermission (write 0x0 to the CODE field from the Control/Status word) [First write to CODE]
2. Reconfigure the ID and data fields
3. Enable the message buffer 13 to be transmitted on BusIdle (write 0xC on CODE field) [Second write to CODE]
4. CAN bus keeps in Idle state
5. No write on the Control/Status from any message buffer happens.

During the second write to CODE (step 3), the write must happen one clock before the current message buffer 13 to be scanned by arbitration process. In this case, it does not detect the new code (0xC) and no new arbitration is scheduled.

The problem can be detected only if the message traffic ceases and the CAN bus enters into Idle state after the described sequence of events.

There is no issue if any of the conditions below holds:

- Any message buffer (either Tx or Rx) is reconfigured (by writing to its CS field) just after the Intermission field.
- There are other configured message buffers to be transmitted.
- A new incoming message sent by any external node starts just after the Intermission field.

Projected Impact:

FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment.

Workarounds:

To transmit a CAN frame, the CPU must prepare a message buffer for transmission by executing the following standard 5-step procedure:

1. Check if the respective interrupt bit is set and clear it.

Chip Errata for the i.MX RT1020, Rev. 2, 01/2021

2. If the message buffer is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control/Status word to request an abortion of the transmission. Wait for the corresponding IFLAG to be asserted by polling the IFLAG register or by the interrupt request if enabled by the respective IMASK. Then read back the CODE field to check if the transmission was aborted or transmitted. If backwards compatibility is desired (MCR[AEN] bit negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the message buffer, but then the pending frame might be transmitted without notification.

3. Write the ID word.

4. Write the data bytes.

5. Write the DLC, Control, and CODE fields of the Control/Status word to activate the message buffer.

6. The workaround consists of executing two extra steps:

7. Reserve the first valid mailbox as an inactive mailbox (CODE = 0b1000). If RX FIFO is disabled, this mailbox must be message buffer 0. Otherwise, the first valid mailbox can be found using the "RX FIFO filters" table in the FlexCAN chapter of the chip reference manual.

8. Write twice INACTIVE code (0b1000) into the first valid mailbox.

NOTE

The first mailbox cannot be used for reception or transmission process.

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

ERR006032 **FlexCAN: A frame with wrong ID or payload is transmitted into the CAN bus when the Message Buffer under transmission is either aborted or deactivated while the CAN bus is in the Bus Idle state**

Description:

The FlexCAN module may transmit an incorrect message if one or more Message Buffers (MBs) are configured for transmission while FlexCAN is in the Bus Idle state, and the MB selected for transmission is either aborted or deactivated at the exact moment it starts to be transmitted. This will cause FlexCAN to transmit a syntactically correct message, but with either incorrect ID or data field. The CRC information will be calculated over the incorrect data (in case data is affected) and all other fields of the frame will be correct.

The probability of the problem occurring is limited to one CAN bit during the transmission of one frame, however under a very specific combination of simultaneous events:

1. Bug event may take place in one specific CAN bit per frame.

2. The CAN bus must be in Bus Idle state.

3. The CPU must be triggered to configure one or more MBs for transmission while in the Bus Idle state.

4. The CPU must be triggered to remove the MBs just configured, by abortion or deactivation, in a short period after starting the configuration in step 3.

In summary, the probability of occurrence is very low, in the order of 1 per 10 million. Moreover, the procedure of configuring a MB followed by abortion or deactivation of the same MB in a short intervals is unlikely to occur in normal applications.

In practice, there is no issue if the CPU guarantees that any MB configured for transmission will be aborted or deactivated just in the next frame.

Workarounds:

The user can avoid the error by preventing to make MB configurations for transmission when the CAN bus is in the Bus Idle state.

There are bits in a FlexCAN debug register can be used to determine when the CAN bus is in the Idle state.

This debug register is located at: FlexCAN Debug 1 Register (CAN_DBG1) - Base + 0x0058

The CAN Finite State Machine (CFSM) bits of CAN_DBG1 register monitor the FlexCAN's internal state. The CFSM is the six least significant bits of the CAN_DBG1 register. The CAN Bit Number (CBN) is the five bits long field at bit position 3 to 7 in the CAN_DBG1 register that indicates the current bit number in a given CFSM state value.

CAN_DBG1.CFSM = 0x0000_003F

CAN_DBG1.CBN = 0x1F00_0000

There are several internal states values that need to be looked for, listed below with their corresponding CFSM value.

RXINTERMISSION - 0x2F

TXINTERMISSION - 0x14

BUSIDLE - 0x02

The following procedure must be performed to configure a MB transmission:

1. Disable all interrupt

2. Read CAN_DBG1.CFSM and CAN_DBG1.CBN fields.

3. Check if CFSM value is either BUSIDLE, RXINTERMISSION or TXINTERMISSION.

For the later two values, also check if CBN value is 3, to determine the paired conditions RXINTERMISSION bit 3 or TXINTERMISSION bit 3, and processed as described below.

3.1. If CAN_DBG1 fields indicate BUSIDLE, wait N CPU clocks.

3.2. Else if CAN_DBG1 fields indicate either RXINTERMISSION bit 3 or TXINTERMISSION bit 3 wait until CFSM is different from either RXINTERMISSION or TXINTERMISSION.

3.3. Check again CAN_DBG1 fields, if they indicate BUSIDLE, wait for DELAY time.

4. Write 0x0 into Code field of CS word.

5. Enable all interrupts.

6. Write the ID word.

7. Write the DATA words.

8. Write 0xC into Code field of CS word.

NOTE

DELAY = {2 x (MAXMB + 1) + 18} x peripheral_clock_period + 3 x PE_clock_period + 1 x CAN_bit_period

The Number of the Last Message Buffer (MAXMB) are the 7 least significant bits in the Module Configuration Register (CAN_MCR: base + 0x0).

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

ERR009527   **FlexCAN: The transmission abort mechanism may not work properly**

Description:

The Flexible Controller Area Network (FlexCAN) is not able to abort a transmission frame and the abort process may remain pending on the following cases:

1. If a pending abort request occurs while the FlexCAN is receiving a remote frame.
2. When a frame is aborted during an overload frame after a frame reception.
3. When an abort is requested while the FlexCAN has just started a transmission.
4. When the Freeze Mode request occurs and the FlexCAN has just started a transmission.

Workarounds:

Use the Mailbox Inactivation mechanism instead of the transmission abort mechanism. The Abort Enable (AEN) bit of the Module Configuration Register can be kept cleared and the abort code value "0b1001" cannot be written into the CODE field of the Message Buffer Control and Status word.

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

ERR009595   **FlexCAN: Corrupt frame possible if the Freeze Mode or the Low-Power Mode are entered during a Bus-Off state**

Description:

If the Freeze Enable bit (FRZ) of the Module Configuration Register (MCR) is asserted and the Freeze Mode is requested by asserting the Halt bit (HALT) of the MCR register during the Bus Off state, the transmission after exiting the Bus-Off condition will be corrupted. The issue occurs only if a transmission is pending before the Freeze Mode request. In addition, the same issue can happen if the Low-Power Mode is requested instead of the Freeze Mode.

Workarounds:

The workaround depends on whether the Bus-Off condition occurs prior to requesting Freeze Mode or the Low-Power Mode.

Procedure to enter the Freeze Mode:

1. Set the Freeze Enable bit (FRZ) in the Module Control Register (MCR).
2. Check if the Module Disable bit (MDIS) in the MCR register is set. If yes, clear the MDIS bit.
3. Poll the MCR register until the Low-Power Mode Acknowledge (LPMACK) bit in the MCR is cleared (timeout for software are implementation is two CAN bits length).
4. Read the Fault Confinement State (FLTCONF) field in the Error and Status 1 Register (ESR1) to check if FlexCAN is in bus-off state. If yes, go to the step 5A. Otherwise, go to step 5B.

5A. Set the Soft Reset bit ((SOFTRST) in the MCR.

6A. Poll the MCR register until the Soft Reset (SOFTRST) bit is cleared (timeout for software are implementation is two CAN bits length.

7A. Poll the MCR register until the Freeze Acknowledge (FRZACK) bit is set (timeout for software are implementation is two CAN bit length).

8A. Reconfigure the MCR.

9A. Reconfigure all the Interrupt Mask Registers (IMASKn).

5B. Set the Halt Flex CAN (HALT) bit in the MCR.

6B. Poll the MCR register until the Freeze Acknowledge (FRZACK) bit is set (timeout for software are implementation is 178 CAN bits length).

NOTE

The time between step 4 and step 5B must be less than 1353 CAN bit periods.

Procedure to enter the Low-Power Mode:

1. Enter the Freeze Mode (execute the procedure A).
2. Request the Low-Power Mode.

Chip Errata for the i.MX RT1020, Rev. 2, 01/2021

3. Poll the MCR register until the Low-Power Mode Acknowledge (LPMACK) bit in the MCR is set (timeout for software are implementation is two CAN bit length).

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

ERR011207 FlexSPI: When FLEXSPI_AHBCR[PREFETCHEN] is set, incorrect data can be returned in rare conditions

Description:

When prefetching is enabled (FLEXSPI_AHBCR[PREFETCHEN]) for non-cacheable space, there are conditions where write-read order might not be guaranteed. The problem can occur if data is written and then read back using AHB interface, while a region containing the data location is in the process of being loaded into the FlexSPI's AHB Rx buffer.

Workarounds:

There are two workarounds:

- If FlexSPI space is not cached (configured as device or strongly-ordered type in the MPU), then FLEXSPI_AHBRXBUFnCR0[PREFETCHEN] can be cleared.
- If the write is critical and the following read is to the same address, FlexSPI_STS0[SEQIDLE] bit can be checked to make sure the write has completed (SEQIDLE is 1) before issuing the subsequent read.

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

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

Description:

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

Workarounds:

Add delay time (100 NOP) again after the DLL lock status is set.

Proposed Solution:

No fix scheduled

Software Status:

No software workaround available

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.

Workarounds:

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

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

ERR050194   **QTMR: Overflow flag and related interrupt cannot be generated when the timer is configured as upward count mode**

Description:

1. Overflow flag and related interrupt cannot be generated successfully in upward count mode.
2. When TMR_CTRL[OUTMODE] is set to 110b, OFLAG output is not cleared on counter rollover when the timer counts upward.

Workarounds:

For item 1, using compare interrupt instead of overflow interrupt by setting compare value to 0xFFFF. The compare interrupt has the same timing effect as overflow interrupt in this way.

For item 2, there is no workaround.

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

ERR011096   SAI: The internal bit clock cannot be generated when BCI = 1

Description:

When SAI transmitter or receiver is configured for the internal bit clock with BCI = 1, the bit clock cannot be generated for either of the following two configurations:

1. SYNC = 00 and BCS = 0
2. SYNC = 01 and BCS = 1

Projected Impact:

The SAI bit clock cannot be generated properly.

Workarounds:

When SAI transmitter or receiver is configured for the internal bit clock with BCI = 1, using one of the following two configurations:

1. SYNC = 01 and BCS = 0
2. SYNC = 00 and BCS = 1

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

ERR011150 **SAI: Internally generated receive or transmit BCLK cannot be re-enabled if it is first disabled when RCR2[DIV] or TCR2[DIV] > 0**

Description:

The receive or transmit bit clock (BCLK) is internally generated and enabled with DIV > 0. When it is disabled, due to software or Stop mode entry, the BCLK is enabled again. Then the clock cannot be generated.

Projected Impact:

The SAI bit clock cannot be generated.

Workarounds:

If the receive or transmit BCLK is internally generated and a DIV value is greater than 0, the SAI must be reset before the BCLK is re-enabled. This is achieved by writing the SR bit in the respective RCSR or TCSR register first to 1, and then immediately write it to 0.

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

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.

Workarounds:

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

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

ERR011225 SEMC: CPU AXI writes to SEMC NAND memory may cause incorrect data programmed into NAND memory

Description:

When SEMC NAND memory region is Normal type, non-cacheable, cacheable write-through, or writeback, non-allocate, and not hit, CM7 AXI writes to the region could program incorrect data to the NAND memory.

Projected Impact:

CPU cannot perform AXI write to SEMC NAND memory when it is the Normal memory type.

Workarounds:

1. Set SEMC NAND memory region to Device type or Strongly-ordered type in MPU, and CPU only perform 32-bit write to SEMC NAND memory region or;
2. Use eDMA to perform 64-bit AXI write to SEMC NAND memory region or;
3. Use IP command to program SEMC NAND memory.

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

ERR050577   SEMC: Auto-refresh can fail to be triggered during long back to back write (or read)

Description:

Auto-refresh command can fail to be triggered during long time back to back write (or read) when SDRAM controller burst length is greater than 1. Missing an auto-refresh command can lead to corruption of SDRAM.

Workarounds:

Configure burst length to 1 by writing register SDRAMCR0.

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

ERR011165　　　　SNVS: Invalid ECC check failure

Description:

When setting LPMKCR[ZMK_ECC_EN] bit, it may generate ZMK ECC Check Failure Violation even the ZMK and its ECC values are correct.

Projected Impact:

ZMK is not usable in case the ZMK ECC check is enabled.

Workarounds:

Not enable ZMK ECC check

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

ERR006281 USB: Incorrect DP/DN state when only VBUS is applied

Description:

When VBUS is applied without any other supplies, incorrect communication states are possible on the data (DP/DN) signals. If VDDHIGH_IN is supplied, the problem is removed.

Projected Impact:

This issue primarily impacts applications using charger detection to signal power modes to a PMIC in an undercharged battery scenario where the standard USB current allotment is not sufficient to boot the system.

Workarounds:

Apply VDDHIGH_IN if battery charge detection is needed. Otherwise, disable charger detection by setting the EN_B bit in USB_ANALOG_USBx_CHRG_DETECTn to 1.

Proposed Solution:

No fix scheduled

Software Status:

Software workaround is not in SDK.

ERR050101 USB: Endpoint conflict issue in device mode

Description:

An endpoint conflict occurs when the USB is working in device mode and an isochronous IN endpoint exists.

When the endpointA IN direction is an isochronous IN endpoint, and the host sends an IN token to endpointA on another device, then the OUT transaction may be missed regardless the OUT endpoint number. Generally, this occurs when the device is connected to the host through a hub and other devices are connected to the same hub.

The affected OUT endpoint can be either control, bulk, isochronous, or an interrupt endpoint.

After the OUT endpoint is primed, if an IN token to the same endpoint number on another device is received, then the OUT endpoint may be unprimed (Cannot be detected by SW), which causes this endpoint to no longer respond to the host OUT token, and thus, no corresponding interrupt occurs.

Workarounds:

Do not connect to a hub in the case when ISO IN endpoint(s) is used. When the hub(s) must be connected in this scenario, the endpoint number(s) of the ISO IN endpoint(s) should be different from the endpoint number(s) of any type of IN endpoint(s) used in any other device(s) connected to the same host.

Proposed Solution:

Fix in next revision

Table continues on the next page...

and service marks licensed by Power.org. M, M Mobileye and other Mobileye trademarks or logos appearing herein are trademarks of Mobileye Vision Technologies Ltd. in the United States, the EU and/or other jurisdictions.

arm