Mask Set Errata for Mask 0N00X

This report applies to mask 0N00X for these products:

- MIMXRT1061DVL6A
- MIMXRT1061DVJ6A
- MIMXRT1062DVL6A
- MIMXRT1062DVJ6A
- MIMXRT106ADVL6A
- MIMXRT106FDVL6A
- MIMXRT106LDVL6A
- MIMXRT1062CVJ5A
- MIMXRT1062CVL5A
- MIMXRT1061CVJ5A
- MIMXRT1061CVL5A

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
ERR050164	BEE: Unaligned access may cause bus error
ERR050235	CCM: Incorrect clock setting for CAN affects UART clock gating
ERR007265	CCM: SoC will enter low power mode before the ARM A9 CPU executes WFI when improper low power sequence is used
ERR011572	Cortex-M7: Write-Through stores and loads may return incorrect data
ERR006223	Failure to resume from WAIT/STOP mode with power gating
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 Bus Idle state.
ERR009595	FlexCAN: Corrupted frame possible if Freeze Mode or Low Power Mode are entered during a Bus-Off state
ERR005829	FlexCAN: FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment during the arbitration process.
ERR009527	FlexCAN: The transmission abort mechanism may not work properly
ERR011377	FlexSPI: DLL lock status bit not accurate due to timing issue
ERR050130	PIT: Temporary incorrect value reported in LMTR64H register in lifetimer mode
ERR050194	QTMR: Overflow flag and related interrupt cannot be generated when the timer is configured as upward count mode

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
ERR050144	SAI: Setting FCONT=1 when TMR>0 may not function correctly
ERR050469	SEMC: 8/16bit write to 8bit PSRAM might cause data corruption
ERR011225	SEMC: CPU AXI write to SEMC NAND memory may cause incorrect data programmed into NAND memory
ERR050538	SOC: Potential boot failure on system reset if SJC_DISABLE fuse is blown
ERR050101	USB: Endpoint conflict issue in device mode
ERR006281	USB: Incorrect DP/DN state when only VBUS is applied
ERR010661	USB: There is an vbus leakage if USBOTG1's vbus is on, and USBOTG2's vbus from on to off for i.MX series

Table 2. Revision History

Revision	Changes
1.1	Initial revision
1.2	The following errata were added. • ERR050469 • ERR050164 • ERR050538

ERR050164: BEE: Unaligned access may cause bus error

Description: BEE only supports 16-byte burst access size. This rule may be violated if master sends unaligned burst, because bus fabric may convert unaligned burst into non-16-byte access and cause BEE to send a bus error.

Workaround: To avoid this issue, the general rule is to set start address to BEE 16-byte aligned and total data number a multiple of 16-byte. These settings will mean the bus fabric will always make 16-byte access requests to the BEE. If access master is CPU, enabling cache is recommended to avoid unaligned accesses.

ERR050235: CCM: Incorrect clock setting for CAN affects UART clock gating

Description: When selecting the CCM CAN clock source with CAN_CLK_SEL set to 2, the UART clock gate will not open and CAN_CLK_ROOT will be off. To avoid this issue, set CAN_CLK_SEL to 0 or 1 for CAN clock selection, or open the UART clock gate by configuring the CCM_CCGRx register.

Workaround: There are two workarounds for this issue:

- • Set CAN_CLK_SEL to 0 or 1 for CAN clock selection.

Or,

- If CAN_CLK_SEL is set to 2, then the CCM must open any of UART clock gate by configuring the CCM_CCGRx register.

ERR007265: CCM: SoC will enter low power mode before the ARM A9 CPU executes WFI when improper low power sequence is used

Description: When software tries to enter low power mode with the following sequence, the SoC will enter low power mode before the ARM A9 CPU executes the WFI instruction:

- Set CCM_CLPCR[1:0] to 2'b00
- ARM CPU enters WFI
- ARM CPU wakeup from an interrupt event, which is masked by GPC or not visible to GPC, such as interrupt from local timer;
- Set CCM_CLPCR[1:0] to 2'b01 or 2'b10
- ARM CPU execute WFI

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

Workaround: Software workaround

1) Software should trigger IRQ #32 (IOMUX) to be always pending manually by setting IOMUX_GPR1_GINT bit

2) Software should then unmask IRQ #32 in GPC before setting CCM low power mode

3) Software should mask IRQ #32 right after CCM low power mode is set (set bit0-1 of CCM_CLPCR)

ERR011572: Cortex-M7: Write-Through 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 get 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.

The 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 gets 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.

Workaround: There is no direct workaround for this erratum.

Where possible, Arm recommends that you use 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.

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)

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 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 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:

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

b) The CAN bus must be in Bus Idle state.

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

d) 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 ©.

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 interval 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. Said differently, there is no issue if any MB configured for transmission lasts active for a minimum of 1 CAN frame.

Workaround: The user can avoid the error by preventing to make Message Buffer (MB) configurations for transmission when the CAN bus is in the Bus Idle state.

To do so, there are bits in a FlexCAN debug register that can be used to determine when the CAN bus is in 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 6 least significant bits of the CAN_DBG1 register. The CAN Bit Number (CBN) is the 5 bits long field at bit positions 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 for transmission:

1) Disable all interrupts.

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 proceed 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*(MAXMB+1) + 18 } * peripheral_clock_period + 3 * PE_clock_period + 1 * CAN_bit_period

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

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

Description: In the Flexible Controller Area Network (FlexCAN) module, 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 Low-Power Mode is requested instead of Freeze Mode.

Workaround: The workaround depends on whether the bus-off condition occurs prior to requesting Freeze mode or low power mode.

A) Procedure to enter Freeze Mode:

1. Set the Freeze Enable bit (FRZ) in the Module Control Register (MCR).

2. Check if the Module Disable bit (MDIS) in MCR register is set. If yes, clear the MDIS bit.

3. Poll the MCR register until the Low-Power Mode Acknowledge (LPMACK) bit in MCR is cleared (timeout for software implementation is 2 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 step 5A. Otherwise, go to step 5B.

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

6A. Poll the MCR register until the Soft Reset (SOFTRST) bit is cleared (timeout for software implementation is 2 CAN Bits length).

7A. Poll the MCR register until the Freeze Acknowledge (FRZACK) bit is set (timeout for software implementation is 2 CAN Bits length).

8A. Reconfigure the Module Control Register (MCR).

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

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

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

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

B) Procedure to enter in Low-Power Mode:

1. Enter in Freeze Mode (execute the procedure A).

2. Request the Low-Power Mode.

3. Poll the MCR register until the Low-Power Mode Acknowledge (LPMACK) bit in MCR is set (timeout for software implementation is 2 CAN Bits length).

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.

Workaround: 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.

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 may 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.

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 in the following cases:

a) If a pending abort request occurs while the FlexCAN is receiving a remote frame.

b) When a frame is aborted during an overload frame after a frame reception.

c) When an abort is requested while the FlexCAN has just started a transmission.

d) When Freeze Mode request occurs and the FlexCAN has just started a transmission.

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

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.

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.

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.

Workaround: 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.

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.

ERR050469: SEMC: 8/16bit write to 8bit PSRAM might cause data corruption

Description: When SEMC is configured as 8bit PSRAM port, and the memory type is Normal type in MPU configuration for the PSRAM region, 8bit or 16bit writes to the region might cause data corrupted.

Workaround: 8bit PSRAM port of SEMC works with following cases.

1). 32bit aligned write access

2). 8bit or 16bit write access with device memory attribution

ERR011225: SEMC: CPU AXI write 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 write-back non-allocate and not hit, CM7 AXI write to the region could program incorrect data to the NAND memory.

Workaround: 1) Set SEMC NAND memory region to Device type or Strongly-ordered type in MPU and CPU only perform 32bit write to it or

2) Use eDMA to perform 64bit AXI write to SEMC NAND memory region or

3) Use IP command to program SEMC NAND memory

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.

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.

Workaround: 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.

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.

Workaround: Apply VDDHIGH_IN if battery charger detection feature is needed

ERR010661: USB: There is an vbus leakage if USBOTG1's vbus is on, and USBOTG2's vbus from on to off for i.MX series

Description: When two USB ports are used in OTG mode simultaneously, the VBUS voltage for the second port (the port not selected by the ANATOP_HW_ANADIG_REG_3P0[VBUS_SEL] field) will be powered by the first port (the one that is selected by the VBUS_SEL field). oltage will not drop after cable unplug, then port can't detect the cable detach.

Workaround: Only have one port works as OTG, keep the others as host. Set the vbus_sel bit to select the host port.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support