Mask Set Errata for Mask 2N76P

This report applies to mask 2N76P for these products:

- MPC5775K
- MPC5774K

Mask Specific Information

Major mask revision number	0x2
Minor mask revision number	0x2
JTAG identifier	2981401D

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
ERR009061	ADC: ADC operations may not work when ADC_MCR[ADCLKSEL] = 0
ERR010455	ADC: Calibration routine needs to be repeated after any SoC reset that triggers device built-in self-test
ERR008603	ADC: ENOB, SINAD and THD specifications are not met for Fin > 50kHz
ERR010327	ADC: Incorrect channel under measure is indicated after sampling phase of conversion until conversion completes
ERR010698	AFE: Clarification of the SDPLL loss of lock indicator operation
ERR010537	AFE: Device may not exit the reset sequence if a low voltage detection on VDD_HV_RAW or VDD_HV_DAC supplies occurs when XOSC is the source of PLL0/1 clock.
ERR008578	AFE: The AFE_PLLSTS[LOR], AFE_OSCSTS[STS], AFE_OSCSTS[FAIL], and MC_ME_GS[S_XOSC] bits do not indicate loss of reference status
ERR011420	C55FMC: Extended Module Configuration Register does not accurately reflect the device flash memory configuration
ERR010356	CGM: Auxiliary clock dividers can become stuck if the Sigma Delta PLL or XOSC is selected as input and a reset event causes a glitch on the clock
ERR010393	CGM: CLK_OUT0 and CLK_OUT1 dividers may become stuck if clock selection is changed while dividers with divide by 2 are operational.
ERR010344	CGM: XOSC cannot be selected as source of system clock or auxiliary clock when configured in single-ended bypass mode
ERR010544	CMU: CMU_0 does not monitor XOSC when it is configured in bypass mode with a 40 MHz external clock connected
ERR010484	CMU: CMU_2 does not monitor the CAN_CLK at output of FlexCAN clock multiplexer
ERR011407	CMU: Sudden loss of clock does not signal the Fault Collection and Control Unit

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
ERR008137	CTU: Non zero values of the CTU_IFR and CTU_EFR can cause false interrupts
ERR009696	DEBUG: Nexus trace messages may be corrupt when transferred to the debugger via the Aurora interface
ERR050090	DSPI/SPI: Incorrect data may be transmitted in slave mode
ERR010385	e200z4: Incorrect branch displacement at 16K memory boundaries
ERR007259	e200zx: ICNT and branch history information may be incorrect following a nexus overflow
ERR007305	e200zx: JTAG reads of the Performance Monitor Counter registers are not reliable
ERR010452	eDMA: When master ID replication is enabled, the stored ID and privilege level will change if read by another master.
ERR008748	End to End ECC: An uncorrectable ECC event from a slave to a master can cause invalid data reception
ERR011188	FCCU : Fault NCF[106] may occur when PLL Loss of Lock event occurs during MBIST execution when online STCU-selftest is run.
ERR009956	FCCU: Address feedback error disabled for the e200z4 Safety Core local RAMs and cache memories
ERR010900	FCCU: False indication of a fault state for a single safe clock period can be generated on the error output pin
ERR009420	FCCU: FOSU may give destructive reset when a hardware recoverable fault of width less than one safe clock occurs
ERR007869	FCCU: FOSU monitoring of a fault is blocked for second or later occurrence of the same fault
ERR010483	FCCU: When NCF[42] is generated using the fake fault mechanism the fault cannot be cleared
ERR009595	FlexCAN: Corrupted frame possible if Freeze Mode or Low Power Mode are entered during a Bus-Off state
ERR008341	FlexCAN: Entering Freeze Mode or Low Power Mode from Normal Mode can cause the FlexCAN module to stop operating.
ERR050246	FlexCAN: Receive Message Buffers may have its Code Field corrupted if the Receive FIFO function is used
ERR009527	FlexCAN: The transmission abort mechanism may not work properly
ERR009928	FlexPWM: Half cycle automatic fault clearing does not work in PWM submodule 0 under some conditions
ERR050119	FlexRay: Disabling of FlexRay Message Buffer during the STARTUP Protocol State takes longer than expected three Slots
ERR008770	FlexRAY: Missing TX frames on Channel B when in dual channel mode and Channel A is disabled
ERR008951	I2C: Attempting a start cycle while the bus is busy may generate a short clock pulse
ERR010330	JTAGM: An unexpected interrupt will occur if the Idle Interrupt is enabled after the end of frame transfer
ERR010329	JTAGM: JTAGM Data out registers are writable regardless of state of the Data Transfer Mode bit
ERR007433	JTAGM: Nexus error bit is cleared by successful RWA
ERR010328	JTAGM: Software Reset bit does not auto clear when written with 1 when an external tool is not connected
ERR008935	JTAGM: write accesses to registers must be 32-bit wide
ERR009957	LFAST: Auxiliary clock divider for FlexCAN must be enabled in order to to use LFAST module
ERR010391	LFAST: Receiver pads do not meet input hysteresis specification
ERR007274	LINFlexD: Consecutive headers received by LIN Slave triggers the LIN FSM to an unexpected state

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
ERR006428	LINFlexD: Data reception could terminate abruptly in LIN Slave mode when Time-out counter mode is enabled
ERR008933	LINFlexD: Inconsistent sync field may cause an incorrect baud rate and the Sync Field Error Flag may not be set
ERR008970	LINFlexD: Spurious bit error in extended frame mode may cause an incorrect Idle State
ERR008080	LINFlexD: TX pin gets set to High-Z when in IDLE state
ERR006350	LINFlexD: WLS feature cannot be used in buffered mode
ERR010365	MC_ME: An invalid mode (Clock Usage) configuration error can occur if the Sigma-Delta PLL and PLL1 are switched on or off independently
ERR008228	MC_ME: Wakeup from STOP mode may lead to a system hang scenario
ERR010317	MEMU: The Memory Error Management Unit may not report a consecutive error from the same address for some memories if the valid bit has been cleared for the initial error
ERR007833	M_CAN: Incorrect frame transmission after recovery from Restricted Operation Mode
ERR007832	M_CAN: Change of CAN operation mode during start of transmission causes incorrect behaviour
ERR007834	M_CAN: Erroneous Interrupt flag after setting / resetting INIT during frame reception
ERR009070	M_CAN: FD frame abort may cause Protocol exception event and extended Bus Integration state
ERR008045	M_CAN: Frame transmission in DAR mode
ERR009069	M_CAN: Incorrect activation of MRAF interrupt
ERR009226	M_CAN: Message loss if message RAM access is not granted prior to the next received message
ERR007828	M_CAN: Message reception and transmission directly after detection of Protocol Exception Event
ERR011469	M_CAN: Message transmitted with wrong arbitration and control fields
ERR050016	M_CAN: Retransmission in DAR mode due to lost arbitration at the first two identifier bits
ERR008299	M_CAN: Setting the Configuration Change Enable (CCE) bit during a transmission scan can halt CAN tranmissions
ERR007594	M_CAN: Transmitted bit in control field is falsified when using extreme bit time configurations
ERR011456	M_CAN: Tx FIFO message sequence inversion
ERR011457	M_CAN: Unexpected High Priority Message (HPM) interrupt
ERR010331	NAL: Trace connections to the device are lost if a device reset occurs
ERR008340	NPC: EVTO_B toggles instead of remaining asserted when used by the DTS if Nexus is not enabled
ERR010479	NPC: Nexus enable required for mode changes when a debugger is attached
ERR010340	NZxC3: ICNT and HIST fields of a Nexus message are not properly reset following a device reset
ERR010638	NZxC3: Nexus messaging becomes corrupted if more than one master (including a device core) is active and the core is subsequently disabled.
ERR007905	PIT: Accessing the PIT by peripheral interface may fail immediately after enabling the PIT peripheral clock
ERR050130	PIT: Temporary incorrect value reported in LMTR64H register in lifetimer mode
ERR010486	PLLDIG: Device may not exit the reset sequence if a reset occurs whilst PLL1 is running with Modulation enabled.
ERR009827	PMC: Device may not exit the reset sequence if a low voltage detect event occurs during execution of offline or online self-test
ERR010259	PMC: Device may not exit the startup reset sequence under certain power sequencing conditions until the STCU watchdog time-out interval elapses

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
ERR011156	PMC: During online PMC self-test if RESET_B is asserted then a spurious destructive or power on reset may occur and be reported as an LVD/HVD event in the RGM_DES register.
ERR008580	PMC: If VDD_HV_PMU is applied before VDD_HV_IO in internal regulation mode, an over-voltage condition on VDD can occur
ERR010414	PMC: Low Voltage Detect (LVD) self test may incorrectly indicate a self test fail if the supply is outwith its operating range during power up
ERR009498	PMC: Performing the PMC self-test with PLL0/PLL1 selected as system clock source can cause an unexpected reset
ERR008745	PMC: Temperature Event Status Register flag bits may be incorrectly set after reset
ERR010345	PMC: The Power Management Controller (PMC) module Analog Front End (AFE) Low Voltage Detect (LVD) interrupt may not work as a valid source of interrupt
ERR011032	PMC: Unexpected events when an LVD occurs on a supply with its LVD masked
ERR011306	SAR ADC: Incorrect value of ADC power down exit delay evaluated by the formula given in PDEDR [PDED] field description
ERR007788	SIUL2: A transfer error is not generated for 8-bit accesses to non-existent MSCRs
ERR007791	SIUL2: Transfer error not generated if reserved addresses within the range of SIUL BASE + 0x100 to 0x23F are accessed
ERR010879	SPT: Adaptive scaling using count unoccupied bits from bit 23 down mode can introduce harmonics in the FFT output
ERR011329	SPT: After completion of a command sequence there is a time window in which work register access by CPU can cause the system to become non-responsive
ERR010352	SPT: AHB error status can be asserted for the incorrect SPT DMA when performing back to back AHB accesses
ERR008223	SPT: FFT instruction with quadrature extension enabled may give wrong parity error indication
ERR010085	SPT: FIR operation writes unnecessary extra data into OPRAM when asynchronous PDMA is concurrently accessing the same OPRAM bank
ERR008224	SPT: SDMA_WR_ERR is generated in certain SDMA transfer modes
ERR010568	SPT: Spurious CS_AHB_ERR can be generated in the SPT_DMA_ERR_STATUS register
ERR008222	SPT: When a parity error occurs the Twiddle RAM (TRAM) or Operand RAM (ORAM) contoller may give erroneous address to MEMU
ERR010997	SPT: Work register contents not deterministic after execution of STOP command
ERR008910	SSCM: NXEN status in SSCM_STATUS registers is a single bit field
ERR007615	SSCM: Accesses to reserved register addresses in modules present in MC_CGM memory space will not generate bus aborts
ERR008749	STCU2: Device may not come out of reset when offline STCU is enabled
ERR010683	STCU2: Online LBIST of partition 6 incorrectly generates reset when run with AFE LVD unmasked
ERR010088	STCU2: Unexpected STCU self-test timeout can occur when a short sequence for external reset is triggered during execution of online self-test
ERR010326	SWT: Out of lockstep fault for instruction and data cache memories is incorrectly indicated when watchdog is configured in fixed address execution service mode
ERR010880	SWT: SWT may not get serviced in Fixed and Incremental address mode
ERR010619	WGM: Access to look-up table address register or data register when WGM clock is inactive can cause system to become non-responsive
ERR008238	WGM: WGM peripheral configuration is not controllable by MC_ME_PCTL[238]

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
ERR008310	XBIC: Crossbar Integrity Checker may miss recording information from an initial fault event in the case of back-to-back faults
ERR008730	XBIC: XBIC may store incorrect fault information when a fault occurs
ERR010436	ZipWire: SIPI can have only one initiator with one outstanding write frame at time

Table 2. Revision History

Revision	Changes
0	Initial revision
1	The following errata were removed. • e9682: Reference manual updated: added note to the SPI_MCR[CLR_RXF] bit field description. • e7991: Reference manual updated to describe correct usage • e8004: Reference manual updated to describe correct usage • e10199: Reference manual updated: SDPLL selection as system clock is now reserved The following errata were added. • e10484 • e10483 • e10365 • e9957 • e10345 The following errata were revised. • e10356: Added details for scenario with XOSC selected
2	The following errata were removed. • e8731: Not applicable to this silicon revision (All LINFlexD clocks made synchronous) The following errata were added. • e10452 • e10544 The following errata were revised. • e10345: Corrected to show destructive reset reaction as still functional • e10340: Major improvements to content and wording • e10483: Minor corrections to wording • e9595: Additions to workaround and corrections to step numbering • e10344: Added information regarding XOSC selection as PLL0 and PLL1 input
3	The following errata were added. • e10537 • e10486 • e10414 • e10564 • e10455 • e10568 • e10393

Table continues on the next page...

Table 2. Revision History (continued)

Revision	Changes
	• e10331 • e10619 • e10479 The following errata were revised. • e7305: Corrected PMC4 to PMC3
4	The following errata were removed. • e9556: 266 MHz PLL frequency not supported for online self-test • e10564: Further design analysis resolved that erratum does not apply The following errata were added. • e10683 The following errata were revised. • e8748: Removed Zipwire/LFAST as erratum does not affect this bus master • e9827: Erratum also applies to online self-test execution scenario
5	The following errata were added. • e10698 • e11420 • e11407 • e11188 • e10900 • e11469 • e50016 • e11456 • e11457 • e10638 • e11156 • e11032 • e11306 • e10879 • e11329 • e10997 • e10880 • e10436 The following errata were revised. • e7869: Simplification of workaround.
6	The following errata were added. • ERR050119 • ERR050090 • ERR050246 • ERR050130 The following erratum was revised. • ERR010479

ERR009061: ADC: ADC operations may not work when ADC_MCR[ADCLKSEL] = 0

Description: Any Successive Approximation Register (SAR) Analog to Digital Converter (ADC) operations including calibration, conversions or self test with the Module Configuration Register Analog Clock frequency Selector field (ADC_MCR[ADCLKSEL]) = 0 may never complete or lead to incorrect results.

Workaround: Use the Clock Generation Module (CGM) auxiliary clock divider for ADC_CLK to achieve desired ADC clock frequency for all operations including calibration. See the Reference Manual for the exact CGM registers to perform the configuration. This will lead to all ADC instances to run at the same clock frequency as configured in CGM auxiliary divider. Ensure in any write access to the ADC_MCR register the ADCLKSEL bit is always set to 1.

ERR010455: ADC: Calibration routine needs to be repeated after any SoC reset that triggers device built-in self-test

Description: The Successive Approximation Register Analog-to-Digital Converter (SAR-ADC) Calibration, BIST Control and Status register (ADC_CALBISTREG) and Offset and Gain User register (ADC_OFSGNUSR) are used to configure the settings used in SAR-ADC's calibration routines and should retain any user programmed values after short and long functional resets.

However it is found that the values of these registers along with calibration routine generated internal values are reset after any SoC reset that triggers offline or online built-in self-test (BIST), which includes long functional reset from an external source (RESET_B) when RGM_FESS[SS_EXR] = 0b0.

The value of these registers goes to the reset value (at POR) even if self-test bypass is configured.

Workaround: The user should ensure that if the SAR-ADC is used in the application then the high accuracy mode ADC calibration routine should be re-run after any reset that causes BIST to be executed. If non-default calibration routine settings are used then the ADC_CALBISTREG and ADC_OFSGNUSR registers should be re-programmed before re-running the calibration routine.

ERR008603: ADC: ENOB, SINAD and THD specifications are not met for Fin &gt; 50kHz

Description: The Analog to Digital Converter (ADC) specifications for ENOB (Effective Number of Bits), SINAD (Signal-to-noise and distortion) and THD (Total Harmonic Distortion) are not met with input signal frequency above 50kHz. Below 50kHz, these specifications are valid.

With input frequencies between 50kHz and 125kHz, ENOB will be > 9.5 bits, SINAD will be > 59 dB and THD will be > 59 dB.

Workaround: Expect lower ENOB, SINAD & THD values at input frequencies above 50kHz.

ERR010327: ADC: Incorrect channel under measure is indicated after sampling phase of conversion until conversion completes

Description: The Main Status Register Channel under measure address field (ADC_MSR[CHADDR]) indicates which ADC channel is currently performing a conversion. This field indicates the correct channel during the sampling phase of conversion, but will display an incorrect value in the subsequent phases until conversion is complete.

Workaround: User must only consider ADC_MSR[CHADDR] to be valid when the ADC is in the sample phase of conversion. The Main Status Register Status of the ADC field shows when the ADC is in the sample phase (ADC_MSR[ADCSTATUS] = 0b100).

ERR010698: AFE: Clarification of the SDPLL loss of lock indicator operation

Description: The description of the SDPLL Status Register SDPLL loss of lock indicator (PLLSTS[LCKLOSS]) flag refers to a register field PLLCTRL1[PWRDN] which is not visible in SDPLL Control Register 1. It should instead state the following:

SDPLL loss of lock indicator. This bit is set when the SDPLL falls out of lock. This is determined by looking for a falling edge of PLLSTS[LOCK] while SDPLL is powered on (after completion of initialization sequence). When this bit is high, it can create the SDPLL loss of lock interrupt if PLLCTRL3[LCKLOIE] is set. Clear this bit by writing a 1 to this position.

Workaround: Understand that reference to “PLLCTRL1[PWRDN] equal to 0” really means that "when the SDPLL is powered up."

ERR010537: AFE: Device may not exit the reset sequence if a low voltage detection on VDD_HV_RAW or VDD_HV_DAC supplies occurs when XOSC is the source of PLL0/1 clock.

Description: If a low voltage detection (LVD) event occurs on the VDD_HV_RAW or VDD_HV_DAC supplies then the default reaction is not to perform a destructive reset of the device (reset reaction for LVD on AFE supplies is enabled by writing to MCB_AFE_LVD_MASK). However it has been found that when the XOSC is used as the source for PLL0 or PLL1 clock then following an LVD event on these supplies there is a chance that the device may become unresponsive if no reset reaction is enabled, or may not exit the reset sequence if the reset reaction is enabled. This occurence is considered to be extremely rare, but is theoretically possible.

Workaround: Use an external watchdog circuit to determine when the MCU is not responding because of brownout conditions on external power supplies and adhere to the Safety Manual guidelines if the stuck in reset condition occurs. Asserting the device power-on reset signal (VREG_POR_B) when the MCU becomes unresponsive or when any power supply is outwith the operating range will bring the device back into normal operation.

ERR008578: AFE: The AFE_PLLSTS[LOR], AFE_OSCSTS[STS], AFE_OSCSTS[FAIL], and MC_ME_GS[S_XOSC] bits do not indicate loss of reference status

Description: The Sigma Delta PLL (SDPLL) status register's Loss of Reference bit AFE_PLLSTS[LOR], Oscillator Status Register Fail bit and Status bit (AFE_OSCSTS[FAIL], AFE_OCSTS[STS]), and Mode Entry Global Status External Oscillator Status bit (MC_ME_GS[S_XOSC]) do not provide the correct status of the SDPLL loss of reference when SDPLL reference clock is removed in the system. The incorrect bit states are listed below:

AFE_PLLSTS[LOR] improperly indicates 0 AFE_OSCSTS[FAIL] improperly indicates 0 AFE_OSCSTS[STS] improperly indicates 1 AFE_ME_GS[S_XOSC] improperly indicates 1

Workaround: Use the Clock Monitor Unit 0 (CMU_0) to detect the status of the oscillator clock.

ERR011420: C55FMC: Extended Module Configuration Register does not accurately reflect the device flash memory configuration

Description: The C55 Flash Memory Controller Extended Module Configuration Register (C55FMC_MCRE) is a read-only register intended to provide the device flash memory configuration in terms of the number of flash memory blocks of each size. The value of this register should be specific to each part number (depending on flash configuration) however it has been found that this register always reads 0x0700_6109 (denoting a 4MB flash device) regardless of the part number and flash configuration.

Workaround: Please use the System Integration Unit Lite 2 MCU ID Registers (SIUL2_MIDR1 and SIUL2_MIDR2) to determine the amount of flash memory available on the device.

ERR010356: CGM: Auxiliary clock dividers can become stuck if the Sigma Delta PLL or XOSC is selected as input and a reset event causes a glitch on the clock

Description: The Clock Generation Module Auxiliary Clock Dividers (MC_CGM_ACx_DCy) provide a divided system clock to peripheral modules. If the 160 MHz Sigma Delta PLL (SDPLL_CLK160) or 40 MHz XOSC is selected as the input to one of these dividers through the associated Select Control Register (MC_CGM_ACx_SC = 3 for SDPLL_CLK160 or MC_CGM_ACx_SC = 1 for XOSC) and there is a glitch on the XOSC input (either the crystal or the 40 MHz bypass) or a reset event other than power-on reset (POR) that causes a glitch on the SDPLL clock, the auxiliary clock divider can become stuck following the reset. A stuck divider will not output any clock unless the workaround is followed or until the device goes through the power-on reset sequence. This only affects auxiliary clock dividers that are configured to divide the input clock by 2 (Divider Division Value in the Divider Configuration Register (MC_CGM_ACx_DCy[DIV]) = 1)

Workaround: Stuck dividers can be reset and recovered if the associated Auxiliary Clock Selector which selects SDPLL or XOSC as input is switched twice after the device completes the reset sequence. This is only necessary after a reset sequence that did not originate at phase POWERUP, as the power-on reset sequence resets the divider input selection. The following software sequence is recommended to be executed after each non-POR reset for auxiliary clock dividers that use SDPLL_CLK160 or XOSC as input and are configured (before the reset) to divide by 2 (MC_CGM_ACx_DCy[DIV]) = 1).

- Disable divider by setting divider enable to 0 (MC_CGM_ACx_DCy[DE] = 0)

• Initially set MC_CGM_AC_SCx[SELCTL] = 0 to select IRC clock

• Then set MC_CGM_AC_SCx[SELCTL] = 3 to re-select SDPLL or MC_CGM_AC_SCx[SELCTL] = 1 to re-select XOSC

• Re-enable divider by setting divider enable to 1 (MC_CGM_ACx_DCy[DE] = 1)

• The divider ratio can then be set so that divide by 2 is achieved (MC_CGM_ACx_DCy[DIV] = 1)

ERR010393: CGM: CLK_OUT0 and CLK_OUT1 dividers may become stuck if clock selection is changed while dividers with divide by 2 are operational.

Description: If clock out functionality is enabled on either CLK_OUT0 and/or CLK_OUT1 and there is a divide by 2 divider operational on these clocks (via CGM_AC14_DC0[DE] and/or CGM_AC9_DC0[DE] = 0b1), then if the clock selection for CLK_OUT is changed via the MCB_CLKOUT_SEL register or a Phase 1,2 or 3 reset occurs then the dividers may become stuck, causing no clock to be output from the divider. This will not clear until a power on reset occurs. This is only true if the divider is using divide by 2 (CGM_AC14_DC0[DIV] and/or CGM_AC9_DC0[DIV] = 0b1).

Workaround: In order to avoid the divider becoming stuck when performing a device reset or changing clock selection when using divide by 2, the user is recommended to follow the steps below:

1. Configure the divider value to divide by 3 by setting CGM_AC14_DC0[DIV] and/or CGM_AC9_DC0[DIV] = 0b10, or any value not equal to 0b0 or 0b1

2. Wait for the divider update to complete, shown by the divider update status bit CGM_DIV_UPD_STAT[17]=0b0 and/or CGM_DIV_UPD_STAT[22]=0b0

3. Disable the divider by setting CGM_AC14_DC0[DE] and/or CGM_AC9_DC0[DE] = 0b0

4. Wait for CGM_DIV_UPD_STAT[17]=0b0 and/or CGM_DIV_UPD_STAT[22]=0b0

5. Change the clock selection required in the MCB_CLKOUT_SEL register

6. Wait for CGM_DIV_UPD_STAT[17]=0b0 and/or CGM_DIV_UPD_STAT[22]=0b0

7. Enable the divider by setting CGM_AC14_DC0[DE] and/or CGM_AC9_DC0[DE] = 0b1

8. Wait for CGM_DIV_UPD_STAT[17]=0b0 and/or CGM_DIV_UPD_STAT[22]=0b0

9. Configure the divider value to the desired division factor by setting CGM_AC14_DC0[DIV] and/or CGM_AC9_DC0[DIV] accordingly

10. Wait for CGM_DIV_UPD_STAT[17]=0b0 and/or CGM_DIV_UPD_STAT[22]=0b0

11. If usage of CGM_AC14_DC0[DIV] =0b1 and/or CGM_AC9_DC0[DIV]=0b1 is only required temporarily then it is advised to re-configure the divider after use to CGM_AC14_DC0[DIV] and/or CGM_AC9_DC0[DIV] = 0b10 or any value not equal to 0b0 or 0b1

If it is expected that a device reset can occur whilst the dividers are enabled it is recommended not to use the divide by 2 option to ensure the clock out dividers do not hang.

If the clock monitoring unit (CMU) is being used to monitor CLK_OUT0 (via CMU_6) then this must also be disabled before CGM_AC14 is disabled during the steps above.

ERR010344: CGM: XOSC cannot be selected as source of system clock or auxiliary clock when configured in single-ended bypass mode

Description: When the RADAR Analog Front End (AFE) XOSC is configured in single-ended mode bypass mode by setting the Crystal oscillator bypass control bit in the Oscillator Control Register (AFE_OSCCTRL[EN_EXT] = 0b1) and a mode is entered with the XOSC control bit cleared in the relevant Mode Entry Module Mode Configuration register (MC_ME_mode_MC[XOSCON] = 0b0) then

- The XOSC cannot be selected as thesource of system clock via System clock switch control in the relevant mode configuration register (MC_ME_mode_MC[SYSCLK]= 0b0001)
- The XOSC cannot be used as the clock source of peripherals modules through the Auxiliary Clock Source Selection Control field in the Clock Generation Module Auxiliary Clock Select Control Registers (CGM_ACn_SC[SELCTL] = 0b001)

Mode transition will not complete if these configurations are attempted.

- The XOSC however can be used as the clock source of PLL0 and PLL1 CLKIN in the Clock Generation Module Auxiliary Clock Select Control Registers (For PLL0 CLKIN MC_CGM_AC3_SC[SELCTL] = 0b001 and

for PLL1 CLKIN by MC_CGM_AC4_SC[SELCTL] = 0b001)

Workaround: If the application uses the XOSC in single-ended bypass mode then the user should not attempt to configure the XOSC as system clock or a peripheral module clock. Instead, the onboard PLLs should be selected as the source for these clocks and the associated clock dividers configured to provide the desired clock frequencies.

ERR010544: CMU: CMU_0 does not monitor XOSC when it is configured in bypass mode with a 40 MHz external clock connected

Description: The AFE crystal oscillator (XOSC) can be configured in XOSC bypass mode by setting the Crystal oscillator bypass control bit in the AFE Oscillator Control Register (AFE_OSCCTRL[EN_EXT] = 0b1) and clearing the XOSC control bit in the MCU operating mode Configuration Register in the Mode Entry module (MC_ME_[mode]_MC[XOSCON] = 0b0). A 40 MHz external single-ended clock must then be provided to the MCU. In this configuration Clock Monitor Unit 0 (CMU_0) does not monitor the XOSC clock because the Mode Entry module indicates that the XOSC is powered down. As a result NCF[27] and NCF[28] are non-functional as fault inputs for CMU_0.

Workaround: There are several possible workarounds to achieve clock monitoring on the XOSC in single-ended bypass mode:

• Expose XOSC on the clock out signal CLK_OUT0 and monitor using CMU_6

• If the Sigma-Delta PLL (SDPLL) is enabled it uses the XOSC as reference input to achieve lock. NCF[89] (SDPLL – Loss of Lock) or NCF[90] (SDPLL – Loss of Reference) can be used to monitor the status of XOSC

• If system PLL0 is configured to use the the XOSC as reference input then NCF[25] (PLL0 – Loss of lock) can be used to monitor the status of XOSC

• If system PLL1 is configured to use the the XOSC as reference input then NCF[26] (PLL1 – Loss of lock) can be used to monitor the status of XOSC

ERR010484: CMU: CMU_2 does not monitor the CAN_CLK at output of FlexCAN clock multiplexer

Description: Clock Monitor Unit 2 (CMU_2) does not monitor the CAN_CLK at the output of the 'from FlexCAN' controlled multiplexer, as the reference manual shows. This multiplexer allows selection of CAN_CLK from either Auxiliary Clock 2 or the XOSC_CLK. CMU_2 actually monitors the output of the Auxiliary Clock 2 divider (output of MC_CGM_AC2_DC0) placed before the multiplexer.

Workaround: Only configure CMU_2 according to the corrected position of monitoring as detailed in the errata description.

ERR011407: CMU: Sudden loss of clock does not signal the Fault Collection and Control Unit

Description: The Clock Monitor Unit (CMU) detects when the frequency of a monitored clock drops below a programmed threshold and asserts the Frequency Less than Low Threshold (FLL) signal if this occurs. The FLL signal is routed to the Fault Collection and Control Unit (FCCU) providing a mechanism to react to the clock fault but due to the monitoring implementation the FLL signal will not be triggered when the monitored clock suddenly stops.

Workaround: Each of the CMU monitored clocks has been analysed for the system level failure effect upon loss of the monitored clock and the safety mechanisms present to detect this. From this analysis it is concluded that loss of all the monitored clocks can be detected by other existing safety mechanisms in the system.

Further, since the CMU monitored clocks are derived from one of the system clock sources (IRC_CLK, XOSC_CLK, PLL0_PHI_CLK, PLL1_PHI_CLK or SDPLL_CLK) if the loss of the monitored clock is caused by the loss of the source clock then this will be detected and reported to FCCU by existing source clock loss detection mechanisms.

CMU_5 is an exception because it is possible to source the monitored LFAST_CLK from external LFAST_REF_CLK input instead of a system clock source. In the externally provided LFAST_CLK use-case the loss of LFAST_REF_CLK can only be detected by loss of the LFAST frame transmit/receive functionality which leads to interruption in the LFAST communication.

ERR011394: CTE: Timing table is not executed when the CTE time base clock is set to 80 MHz and the table execution duration is 1 clock cycle

Description: The CTE time base (datapath) clock is derived from the 80 MHz CTE module clock. It is possible to configure the time base clock frequency to an undivided 80 MHz by setting CTE Control Register 1 CTE Clock divider field to 0 or 1 (CTE_CNTRL1[CTECK_DV] = 0 or 1). If this is done and the Timing Table (TT) execution duration is set to 1 clock cycle by programming TT0/TT1 Execution Duration Register to 1 (CTE_LUT_DUR = 1 or CTE_LUT_DUR1 = 1 depending on whether TT0 or TT1 is to be executed) then the CTE will not execute the timing table and no timing signals will be generated.

Workaround: To execute a timing table for 1 clock cycle the CTE time base clock frequency must be less than 80 MHz. This can be achieved by setting CTE_CNTRL1[CTECK_DV] field to any value greater than 1 which results in a divided CTE time base clock less than 80 MHz.

ERR008137: CTU: Non zero values of the CTU_IFR and CTU_EFR can cause false interrupts

Description: The reset values for the Cross Triggering Unit's (CTU) Interrupt Flag Register (CTU_IFR) and Error Flag Register (CTU_EFR) are 0x0000 but after the CTU timer clock (CTU Clock) is enabled in the SOC the values of these registers will get updated to:

CTU_IFR = 0x1FE

CTU_EFR = 0x400

So if any of the trigger interrupts(T0_IE to T7_IE) or the Interrupt Error Enable (IEE) bits in the CTU Interrupt/DMA register (CTU_IR) are enabled before these registers are cleared the CTU will incorrectly generate the corresponding interrupt to the system.

Workaround: Customer needs to first clear the CTU_IFR and CTU_EFR registers before enabling any interrupts in the CTU_IR register.

ERR009696: DEBUG: Nexus trace messages may be corrupt when transferred to the debugger via the Aurora interface

Description: When Nexus tracing is enabled and the Aurora interface is used to transfer those Nexus trace messages to the debugger, it may happen that sometime Aurora frames that contain the trace messages are corrupt and the Aurora frames can not be correctly decoded by the debugger. Consequently the trace messages and their content are lost.

Most likely this situation occurs when the Aurora interface is heavily loaded, i.e. many Nexus trace messages are generated by the Nexus clients within the device and need to be transferred via the Aurora interface.

Workaround: Limit the number of Nexus trace messages to be transferred via the Aurora interface by configuration of the Nexus clients within the device.

ERR050090: DSPI/SPI: Incorrect data may be transmitted in slave mode

Description: If the Serial Peripheral Interface (SPI or the Deserial/Serial Peripheral Interface) is operating in slave mode, incorrect or stale data may be transmitted in next transaction without underflow interrupt generation if the set up time of the Peripheral Chip Select (PCS) to the SPI Serial Clock (SCLK) is short and the transmit FIFO may become empty after one transaction.

This can occur if the PCS to SCK is less than:

4 X IPG_CLOCK_PERIOD + 4 X DSPI_CLOCK_PERIOD + 0.5 x SCK_CLOCK_PERIOD

Where:

IPG_CLOCK is the internal bus clock ("system" clock)

DSPI_CLOCK is the protocol clock.

SCK_CLOCK is the Line-Side Serial Communication Clock.

Workaround: When operating in slave mode, software must ensure that the time interval between PCS assertion to start of SCK Clock is greater than 4 X IPG_CLOCK_PERIOD + 4 X DSPI_CLOCK_PERIOD + 0.5 x SCK_CLOCK_PERIOD.

To meet this requirement, the Master SPI can either lengthen the PCS to SCK assertion time or decrease the frequency of the communication interface, or both.

ERR010385: e200z4: Incorrect branch displacement at 16K memory boundaries

Description: The branch target address will be incorrectly calculated in the e200z4 core under the following conditions (all conditions must be matched):

- The first full instruction in a 16 Kbyte section/page of code is a 32-bit long branch with a branch displacement value with the lower 14 bits of the displacement exactly 0x3FFE
- And this branch instruction is located at byte offset 0x0002 in the section/page
- And the preceding instruction is a 32-bit length instruction which is misaligned across the 16K boundary
- And both instructions are dual-issued

Under these conditions, the branch target address will be too small by 32Kbytes.

Workaround: After software is compiled and linked, code should be checked to ensure that there are no branch instructions located at address 0x2 of any 16K memory boundary with the lower 14 bits of the displacement equal to 0x3FFE if preceded a 32-bit instruction that crosses the 16K memory boundary. If this sequence occurs, add a NOP instruction or otherwise force a change to the instruction addresses to remove the condition.

A tool is available on nxp.com that can be run to examine code for this condition, search for branch_displacement_erratum_10385_checker.

ERR007259: e200zx: ICNT and branch history information may be incorrect following a nexus overflow

Description: If an internal Nexus message queue over-flow occurs when the e200zx core is running in branch history mode (Branch Method bit [BTM] in the Development Control register 1 [DC1] is set [1]), the instruction Count (ICNT) and branch history (HIST) information in the first program trace message following the Program Correlation message caused by an over-flow of the internal trace buffers, will contain incorrect ICNT and HIST information.

This can also occur following an overflow of the internal Nexus message queues in the traditional branch mode (BTM in the DC1 is cleared [0]). Traditional branch mode Nexus messages do not include HIST information, since all branches generate a trace message.

Workaround: There are two methods for dealing with this situation.

1) Avoid overflows of the Nexus internal FIFOs by reducing the amount of trace data being generated by limiting the range of the trace area by utilizing watchpoint enabled trace windows or by disabling unneeded trace information, or by utilizing the stall feature of the cores.

2) After receiving an overflow ERROR message in Branch History mode, the ICNT and HIST information from the first Program Trace Synchronization message and the next Program Trace message with a relative address should be discarded. The address information is correct, however, the ICNT and previous branch history are not correct. All subsequent messages will be correct.

In traditional branch mode, the ICNT information should be discarded from the Program Trace Sync message and the next direct branch message.

ERR007305: e200zx: JTAG reads of the Performance Monitor Counter registers are not reliable

Description: Reads of the Performance Monitor Counter (PMC0, PMC1, PMC2, and PMC3) registers through the IEEE 1149.1 or IEEE 1149.7 (JTAG) interfaces may return occasional corrupted values.

Workaround: To ensure proper performance monitor counter data at all times, software can be modified to periodically read the PMCx values and store them into memory. JTAG accesses could then be used to read the latest values from memory using Nexus Read/Write Access or the tool could enable Nexus data trace for the stored locations for the information to be transmitted through the Nexus Trace port.

ERR010452: eDMA: When master ID replication is enabled, the stored ID and privilege level will change if read by another master.

Description: When master ID replication is enabled (DMA_DCHMIDn[EMI]=1), the DMA_DCHMIDn[PAL] and DMA_DCHMIDn[MID] fields should reflect the privilege level and master ID respectively of the master that wrote the DMA_TCDn_CSR[DONE:START] byte. However, if a different master reads the DMA_TCDn_CSR[DONE:START] byte, the master ID and privilege level will incorrectly change to this read access.

Workaround: Only allow the intended master ID replication core to access the DMA_TCDn_CSR[DONE:START] byte.

ERR008748: End to End ECC: An uncorrectable ECC event from a slave to a master can cause invalid data reception

Description: When an Uncorrectable Error Correction Code (ECC) is received by a master, incorrect data can result in the transaction after the ECC error. There are three scenarios impacting the device:

Case 1 DMA:

If an End to End ECC transaction is received during back to back requests from DMA to a slave in the cross bar (RAM/Flash) then the DMA can receive incorrect data

Case 2 Master A Ethernet and Master B FlexRay:

When an Uncorrectable ECC is received by one master during a burst transaction to a slave and an access is made by the other master to the same slave, there is potential that Master B will receive wrong data or cause either of the masters to stall.

Case 3 Signal Processing Toolbox (SPT):

If an End to End ECC transaction is received during back to back requests from SPT to a slave in the cross bar (RAM/Flash) then the SPT can receive incorrect data

Workaround: Use one of the two following workarounds:

1. Use the NCF for Uncorrectable ECC Errors on all the slaves to determine that such a fault has occurred and take appropriate action.

NCF[17] - System RAM uncorrectable ECC error

NCF[23] - Flash (c55fmc) uncorrectable ECC error can be monitored to determine if this scenario occurred

2. Clear Pending Read Enable bit for the masters:

Signal Processing Toolbox (SPT) - IAHB_BE5[PRE_SPT= Bit 29] - Set to 0

Ethernet+FlexRay – IAHB_BE2[PRE_FEC – Bit 21] - Set to 0

DMA – IAHB_BE1[PRE_DMA- Bit 31] - Set to 0

However, this configuration may reduce the performance timing of the system.

ERR011188: FCCU : Fault NCF[106] may occur when PLL Loss of Lock event occurs during MBIST execution when online STCU-selftest is run.

Description: If a PLL loss of lock (LOL) event occurs during the Direct Memory Access (DMA) RAM Memory BIST (MBIST) portion of the online STCU Self Test then the Self Test will be aborted and the Fault Collection and Control Unit (FCCU) will report a Non-Critical Fault NCF[7] (STCU NCF). This will be indicated by NCF[7] being set in the FCCU_NCF_S0 register. It has been found that in addition to this the FCCU may unexpectedly also report a DMA RAM Alarm fault. This will be indicated by NCF[106] being set in the FCCU_NCF_S3 register.

Workaround: If the application makes use of the DMA module and the DMA RAM Alarm NCF[106] occurs then the software reaction to this fault should first ensure that the alarm is genuine by ensuring that no STCU online self test was in operation and was aborted due to a PLL LOL event. This can be determined by checking the STCU2_ERR_STAT[LOCKESW] bit. If STCU2_ERR_STAT[LOCKESW] is not equal to 0b1 then the alarm is genuine and software can take appropriate action to this alarm, otherwise it can be cleared (by writing 0x400 to FCCU_NCF_S3) and ignored. If STCU online self test is not used in the application then no action need be taken and if NCF[106] occurs it can be considered genuine and be dealt with accordingly.

It should be noted that following a failed STCU online self test due to the conditions described the STCU2_ERR_STAT[LOCKESW] bit will remain set until a subsequent self test is run. This needs to be considered as if no action to the failing self test (e.g re-run self test) is taken then this would mean that a genuine NCF[106] occurrence could be interpreted as a false flag.

ERR009956: FCCU: Address feedback error disabled for the e200z4 Safety Core local RAMs and cache memories

Description: The additional, dedicated measures against addressing and control faults (such as address/ control feedback) have been disabled. This applies to the following memories of the e200z4 Safety Core 0:

Memory type	Name

Data SRAM	Core0 DTCM
Instruction CACHE	Core0 ICACHE
Data CACHE	Core0 DCACHE
Instruction TAG	Core0 ITAG
Data TAG	Core0 DTAG

This, in turn, has disabled the Fault Collection and Control Unit (FCCU) Failure Input Channels 95, 96 and 97 (NCF[95], NCF[96], NCF[97]) which indicate an addressing error (address-feedback) inside the above memories.

Workaround: Checks for address errors should be performed using the MBIST(Memory Built-In Self Test) features of the device. FCCU input channel 95,96,97 will not be asserted

ERR010900: FCCU: False indication of a fault state for a single safe clock period can be generated on the error output pin

Description: The error out pin from the Fault Collection and Control Unit (FCCU) may pulse to a logic low (0b0) when the following conditions are fulfilled:

- software changes the error out protocol from a toggling protocol to a not-toggling protocol, and programs the FCCU_CFG.FCCU_SET_AFTER_RESET bit to 0b1
- software switches the Fault Collection and Control Unit (FCCU) state machine from CONFIG to NORMAL state

The duration of the glitch is equal to a single clock period of the Internal RC oscillator and there is a 50% of probability of the pulse occurring.

Workaround: Split the configuration of the FCCU in 2 phases.

During the first phase, software should do the following:

1) move the FCCU to the CONFIG state

2) configure the FCCU including the error out protocol, but without setting the FCCU_CFG.FCCU_SET_AFTER_RESET flag to 0b1 (leave as 0b0)

3) exits to the NORMAL state

During the second phase, software should do the following:

4) move the FCCU to the CONFIG state

5) set the FCCU_CFG.FCCU_SET_AFTER_RESET flag to 0b1

6) exit to the NORMAL state

Note: The default (after reset) error out protocol is the Dual Rail. Since this is a toggling protocol, the software must execute the above steps each time the user wants to switch to a non-toggling error out protocol.

ERR009420: FCCU: FOSU may give destructive reset when a hardware recoverable fault of width less than one safe clock occurs

Description: The Fault Collection and Control Unit Output Supervision Unit (FOSU) may issue a destructive reset in the following conditions:

• An input fault is programmed as hardware recoverable (FCCU_RF_CFG0, FCCU_RF_CFG1)

• The reaction programmed is only Error Output pin (EOUT) signaling (FCCU_EOUT_SIG_EN0,FCCU_EOUT_SIG_EN1)

• The source fault coming on the Recoverable Fault (RF) line is asserted only for less than one safe clock duration

Workaround: 1) If a fault must be configured as hardware recoverable, may last less than one safe clock cycle and requires EOUT signalling, program long/short functional reset for that HW recoverable fault besides eout signalling.

2) Alternatively if a fault assertion may last less than one safe clock cycle and only EOUT signaling is preferred as reaction, the said fault shall be configured as software recoverable

ERR007869: FCCU: FOSU monitoring of a fault is blocked for second or later occurrence of the same fault

Description: The Fault Collection and Control Unit (FCCU) Output Supervision Unit (FOSU) will not monitor the FCCU for the second or later occurrence of a given fault in the following cases:

1. Reset is programmed as the only reaction for the fault.

2. Assertion of the fault coincides with the long/short functional reset reaction to a fault previously asserted.

Workaround: Enable either Alarm state (NCFTOEx) or at least one other type of Fault-state reaction: Non-maskable Interrupt (NMI) or error out (EOUT) signaling reaction for the faults that have a reset reaction enabled only. Restrictions of combining reset reaction with additional reactions may be written in the chip specific sub-section of the FCCU chapter.

ERR010483: FCCU: When NCF[42] is generated using the fake fault mechanism the fault cannot be cleared

Description: Fault Collection and Control Unit (FCCU) Non-critical fault input 42 (NCF[42]) which signifies an “Alarm indicating the flash detected an error in the ECC correction logic through an EDC” is not cleared even when the corresponding bit in the Noncritical Fault Status 1 (FCCU_NCF_S1) register is cleared twice using write-one-to-clear (w1c) technique, as specified in the reference manual. The fault only fails to clear when it is simulated using the FCCU fake fault mechanism.

Workaround: The fault can be cleared by performing a read transaction on the onboard flash memory in between the two w1c operations on FCCU_NCF_S1.

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).

ERR008341: FlexCAN: Entering Freeze Mode or Low Power Mode from Normal Mode can cause the FlexCAN module to stop operating.

Description: In the Flexible Controller Area Network (FlexCAN) module, if the Freeze Enable bit (FRZ) in the Module Configuration Register (MCR) is asserted and the Freeze Mode is requested by asserting the Halt bit (HALT) in MCR, in some cases, the Freeze Mode Acknowledge bit (FRZACK) in the MCR may never be asserted.

In addition, the Low-Power Mode Acknowledge bit (LPMACK) in the MCR may never be asserted in some cases when the Low-Power Mode is requested.

Under the two scenarios described above, the loss of ACK assertion (FRZACK, LPMACK) causes a lock condition. A soft reset action is required in order to remove the lock condition.

The change from Normal Mode to Low-Power Mode cannot be done directly. Instead, first change mode from Normal to Freeze Mode, and then from Freeze to Low-Power Mode.

Workaround: To avoid the lock condition, the following procedures must be used:

A) Procedure to enter in Freeze Mode:

1. Set both the Freeze Enable bit (FRZ) and the Halt bit (HALT) 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 Freeze Mode Acknowledge bit (FRZACK) in MCR is set or the timeout is reached (see NOTE below).

4. If the Freeze Mode Acknowledge bit (FRZACK) is set, no further action is required. Skip steps 5 to 8.

5. If the timeout is reached because the Freeze Mode Acknowledge bit (FRZACK) is still cleared, then set the Soft Reset bit (SOFTRST) in MCR.

6. Poll the MCR register until the Soft Reset bit (SOFTRST) bit is cleared.

7. Reconfigure the Module Control Register (MCR)

8. Reconfigure all the Interrupt Mask Registers (IMASKn).

After Step 8, the module will be in Freeze Mode.

NOTE: The minimum timeout duration must be equivalent to:

a) 730 CAN bits if the CAN FD Operation Enable bit (FDEN) in MCR is set (CAN bits calculated at arbitration bit rate),

b) 180 CAN bits if the FDEN bit is cleared.

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.

ERR050246: FlexCAN: Receive Message Buffers may have its Code Field corrupted if the Receive FIFO function is used

Description: If the Code Field of a Receive Message Buffer is corrupted it may deactivate the Message Buffer, so it is unable to receive new messages. It may also turn a Receive Message Buffer into any type of Message Buffer as defined in the Message buffer structure section in the device documentation.

The Code Field of the FlexCAN Receive Message Buffers (MB) may get corrupted if the following sequence occurs.

1- A message is received and transferred to an MB (i.e. MBx)

2- MBx is locked by software for more than 20 CAN bit times (time determines the probability of erratum to manifest).

3- SMB0 (Serial Message Buffer 0) receives a message (i.e. message1) intended for MBx, but destination is locked by the software (as depicted in point 2 above) and therefore NOT transferred to MBx.

4- A subsequent incoming message (i.e. message2) is being loaded into SMB1 (as SMB0 is full) and is evaluated by the FlexCAN hardware as being for the FIFO.

5-During the message2, the MBx is unlocked. Then, the content of SMB0 is transferred to MBx and the CODE field is updated with an incorrect value.

In case a customer does use Rx FIFO only or dedicated MB only, (i.e. either Rx MB or Rx FIFO is used) the problem doesn't occur. In case a customer does use the Enhanced Rx FIFO and dedicated MB at the same application, the problem does not occur also. So bottom line the problem does only occur if the FlexCAN is programmed to receive in the Legacy FIFO and dedicated MB at the same application.

Workaround: This defect only applies if the Receive FIFO (Legacy Rx FIFO) is used. This feature is enabled by RFEN bit in the Module Control Register (MCR). If the Rx FIFO is not used, the Receive Message Buffer Code Field is not corrupted.

If available on the device, use the enhanced Rx FIFO feature instead of the Legacy Rx FIFO. The Enhanced Rx FIFO is enabled by the ERFEN bit in the Enhanced Rx FIFO Control Register (ERFCR).

The defect does not occur if the Receive Message Buffer lock time is less than or equal to the time equivalent to 20 x CAN bit time.

The recommended way for the CPU to service (read) the frame received in a mailbox is by the following procedure:

1. Read the Control and Status word of that mailbox.

2. Check if the BUSY bit is deasserted, indicating that the mailbox is not locked. Repeat step 1) while it is asserted.

3. Read the contents of the mailbox.

4. Clear the proper flag in the IFLAG register.

5. Read the Free Running Timer register (TIMER) to unlock the mailbox

In order to guarantee that this procedure occurs in less than 20 CAN bit times the MB receive handling process in software (step 1 to step 5 above) should be performed as a 'critical code section' (interrupts disabled before execution) and should ensure that the MB receive handling occurs in a deterministic number of cycles.

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.

ERR009928: FlexPWM: Half cycle automatic fault clearing does not work in PWM submodule 0 under some conditions

Description: When

a) the EXT_SYNC signal is selected to cause initialization by setting the Submodule 0 Control 2 Register FlexPWM_SUB0_CTRL2[INIT_SEL] = 11 and

b) a specific FAULTx input is associated with the submodule 0 outputs using the Submodule 0 Fault Disable Mapping Register (FlexPWM_SUB0_DISMAP) and

c) the respective bit for that FAULTx is 0 in the FFULL bitfield of the Fault Status Register FlexPWM_FSTS and

d) the respective bit for that FAULTx is 1 in the FAUTO bitfield of the Fault Control Register FlexPWM_FCTRL,

then the PWM outputs of submodule 0 will only be re-enabled at the cycle boundary (full cycle) and will not be re-enabled at the cycle midpoint (half cycle).

Workaround: When the EXT_SYNC signal is used to cause initialization in submodule 0 and the submodule 0 PWM outputs are disabled by a specific FAULTx input, use full cycle automatic fault clearing for the specific FAULTx input by setting the corresponding bit of the Fault Status Register FlexPWM_FSTS[FFULL] to 1.

ERR050119: FlexRay: Disabling of FlexRay Message Buffer during the STARTUP Protocol State takes longer than expected three Slots

Description: Disabling of FlexRay Message Buffer takes longer than the expected three Slots. This is observed, when software application tries to disable the Message Buffer during the FlexRay STARTUP protocol state (VPOC!State = POC:startup) when vPOC!StartupState = “initialize schedule” or “integration consistency check”.

In this scenario, FlexRay Communication Controller keeps the specific Message buffer search results until the availability of next cycle start/segment start/slot start events and therefore prevent the disabling of Message Buffer.

Note:

1.All Message Buffers can be disabled immediately if FlexRay protocol state (vPOC!State) is in following States: “POC:default config”, “POC:config”, “POC:wakeup”, “POC:ready”, “POC:halt”, “POC:startup” and (vPOC!StartupState = "POC:integration listen” or "POC: ColdStart-Listen”).

2.All Message Buffers can be disabled within three slots, if FlexRay protocol state (vPOC! State) is in following states: ”POC: Normal-Active” or ”POC: Normal-Passive”.

Workaround: Do not disable Message Buffer, while FlexRay is in STARTUP protocol State

ERR008770: FlexRAY: Missing TX frames on Channel B when in dual channel mode and Channel A is disabled

Description: If the FlexRay module is configured in Dual Channel mode, by clearing the Single Channel Device Mode bit (SCM) of the Module Control register (FR_MCR[SCM]=0), and Channel A is disabled, by clearing the Channel A Enable bit (FR_MCR[CHA]=0) and Channel B is enabled, by setting the Channel B enable bit (FR_MCR[CHB]=1), there will be a missing transmit (TX) frame in adjacent minislots (even/odd combinations in Dynamic Segment) on Channel B for certain communication cycles. Which channel handles the Dynamic Segment or Static Segment TX message buffers (MBs) is controlled by the Channel Assignment bits (CHA, CHB) of the Message Buffer Cycle Counter Filter Register (FR_MBCCFRn). The internal Static Segment boundary indicator actually only uses the Channel A slot counter to identify the Static Segment boundary even if the module configures the Static Segment to Channel B

(FR_MBCCFRn[CHA]=0 and FR_MBCCFRn[CHB]=1). This results in the Buffer Control Unit waiting for a corresponding data acknowledge signal for minislot:N in the Dynamic Segment and misses the required TX frame transmission within the immediate next minislot:N+1.

Workaround: 1. Configure the FlexRay module in Single Channel mode (FR_MCR[SCM]=1) and enable Channel B (FR_MCR[CHB]=1) and disable Channel A (FR_MCR[CHA]=0). In this mode the internal Channel A behaves as FlexRay Channel B. Note that in this mode only the internal channel A and the FlexRay Port A is used. So externally you must connect to FlexRay Port A.

2. Enable both Channel A and Channel B when in Dual Channel mode (FR_MCR[CHA=1] and FR_MCR[CHB]=1). This will allow all configured TX frames to be transmitted correctly on Channel B.

ERR008951: I2C: Attempting a start cycle while the bus is busy may generate a short clock pulse

Description: When the I2C (Inter-Integrated Circuit) is operating in a multi-master network and a start cycle is attempted by the I2C device when the bus is busy, the attempting master will lose arbitration as expected but a short extra clock cycle is generated in the bus. After losing arbitration, the master switches to slave mode but it does not detect the short clock pulse. The acknowledge signal is expected at the ninth clock by the current bus master but it is not sent as expected due to the undetected short clock pulse.

Workaround: Software must ensure that the I2C BUS is idle by checking the bus busy bit in the I2C Bus Status Register (I2C_IBSR.IBB) before switching to master mode and attempting a Start cycle.

ERR010330: JTAGM: An unexpected interrupt will occur if the Idle Interrupt is enabled after the end of frame transfer

Description: After a JTAGM frame has completed when the JTAGM Idle Interrupt was not enabled (JTAGM_MCR[IIE] = 0b0), if the JTAGM module is then configured to give an interrupt(JTAGM_MCR[IIE] = 0b1), an unexpected interrupt will be seen.

Workaround: Before enabling a JTAGM idle interrupt by setting JTAGM_MCR[IIE] = 0b1, set the Ide bit (JTAGM_SR[Idle]) to 0b1 to clear any previous interrupts.

ERR010329: JTAGM: JTAGM Data out registers are writable regardless of state of the Data Transfer Mode bit

Description: The Data Transfer Mode bit (JTAGM_MCR[DTM]) is intended to control the ability of application software to write JTAG data to the JTAGM Data Output Registers (JTAGM_DORn). However JTAGM_DOR0,JTAGM_DOR1,JTAGM_DOR2 and JTAGM_DOR3 are writable regardless of the state of the JTAGM_MCR[DTM] bit.

Workaround: If application software generated data is required to be written to the JTAGM Data Output Registers then the permissions to write to these registers should be handled within the software to ensure no genuine JTAG Data required is overwritten.

ERR007433: JTAGM: Nexus error bit is cleared by successful RWA

Description: The JTAG Master module status register includes a Nexus error status bit (JTAGM_SR[Nexus_err]) that indicates the status of the last Nexus Read/Write Access (RWA) command. Once this information is latched, it can only be cleared by performing a successful RWA transaction via the same core that caused the error. In addition, if a RWA transaction is performed by a different core, the error bit will not be cleared and it is not possible to determine if the access by the second core RWA was successful or generated another error.

In general, this bit should only be set when the Nexus RWA accesses non-existent or protected memory spaces.

Workaround: If the status information is required from a specific core, the user software or tool should read the error bit (ERR) of the e200zx core's Nexus Read/Write Access Control/Status register. To avoid setting the error bit, do not perform illegal memory accesses.

ERR010328: JTAGM: Software Reset bit does not auto clear when written with 1 when an external tool is not connected

Description: The JTAGM module Software Reset bit (JTAGM_MCR[SWRESET]) when set to 0b1 resets the state machine and counters inside the JTAGM module and should auto clear to 0b0. However, the auto clear of JTAGM_MCR[SWRESET] only occurs with an external tool connected and will not auto clear when no external tool is connected, unless the Data Transfer Mode bit is set (JTAGM_MCR[DTM]= 0b1) and that a valid TCK selection is made via JTAGM_MCR[TCKSEL] bits.

Workaround: To use the Software Reset bit with no tool connected, user must ensure that the Data Transfer Mode bit is set (JTAGM_MCR[DTM]= 0b1) and that a valid TCK selection is made via JTAGM_MCR[TCKSEL] bits. This will ensure that the JTAGM module is correctly clocked with no external tool and therefore the Software Reset bit will clear.

ERR008935: JTAGM: write accesses to registers must be 32-bit wide

Description: The JTAG Master module (JTAGM) supports only 32-bit write accesses to its registers. A byte write access will be converted into a 32-bit write with the other bytes values at 0x0.

Workaround: Perform only 32-bit write accesses on JTAGM registers. Do not use byte writes.

ERR009957: LFAST: Auxiliary clock divider for FlexCAN must be enabled in order to to use LFAST module

Description: There is a clock gate between the clock divider for LVDS Fast Asynchronous Serial Transmission (LFAST) and the module clock input. This gate has an unexpected extra dependency on the enable control of the clock divider for the FlexCAN module. If this divider is disabled, no clock will be provided to the LFAST module.

Workaround: A clock signal is only provided to the LFAST module if the Auxiliary Clock Dividers for the both the LFAST and FlexCAN modules are enabled. This requires that the Divider Enable bit is set in two Clock Generation Module Auxiliary Clock Divider Configuration Registers (CGM_AC8_DC0[DE] = 0b1 and CGM_AC2_DC0[DE] = 0b1).

ERR010391: LFAST: Receiver pads do not meet input hysteresis specification

Description: The LVDS Fast Asynchronous Transmission (LFAST) receiver pads do not meet the data sheet specification for input hysteresis (Vhys_drf). The LFAST electrical characteristics state a minimum of 25mV for this parameter but some devices do not meet this specification.

Workaround: Do not expect each device to meet the data sheet specification for LFAST input hysteresis.

ERR007274: LINFlexD: Consecutive headers received by LIN Slave triggers the LIN FSM to an unexpected state

Description: As per the Local Interconnect Network (LIN) specification, the processing of one frame should be aborted by the detection of a new header sequence and the LIN Finite State Machine (FSM) should move to the protected identifier (PID) state. In the PID state, the LIN FSM waits for the detection of an eight bit frame identifier value.

In LINFlexD, if the LIN Slave receives a new header instead of data response corresponding to a previous header received, it triggers a framing error during the new header’s reception and returns to IDLE state.

Workaround: The following three steps should be followed -

1) Configure slave to Set the MODE bit in the LIN Time-Out Control Status Register (LINTCSR[MODE]) to ‘0’.

2) Configure slave to Set Idle on Timeout in the LINTCSR[IOT] register to ‘1’. This causes the LIN Slave to go to an IDLE state before the next header arrives, which will be accepted without any framing error.

3) Configure master to wait for Frame maximum time (T Frame_Maximum as per LIN specifications) before sending the next header.

Note:

THeader_Nominal = 34 * TBit

TResponse_Nominal = 10 * (NData + 1) * TBit

THeader_Maximum = 1.4 * THeader_Nominal

TResponse_Maximum = 1.4 * TResponse_Nominal

TFrame_Maximum = THeader_Maximum + TResponse_Maximum

where TBit is the nominal time required to transmit a bit and NData is number of bits sent.

ERR006428: LINFlexD: Data reception could terminate abruptly in LIN Slave mode when Time-out counter mode is enabled

Description: When the Local Interconnect Network module (LINFlexD) is used in LIN Slave mode, and in the LIN Time-Out Control Status Register (LINTCSR) the time-out counter mode is configured to LIN mode (LINTCSR[MODE]=0), the Output Compare value 2 in LIN Output Compare Register (LINOCR[OC2]) is loaded by hardware to monitor the response duration.

This loaded value may be incorrect and may lead to setting the Output Compare Flag in the LIN Error Status register (LINESR[OCF]) after which data reception is terminated by the LIN slave.

Workaround: Use the LINFlexD in LIN Slave mode with time-out control disabled in the LIN Time-out Control Status register (LINTCSR[TOCE]=0).

ERR008933: LINFlexD: Inconsistent sync field may cause an incorrect baud rate and the Sync Field Error Flag may not be set

Description: When the LINFlexD module is configured as follows:

1. LIN (Local Interconnect Network) slave mode is enabled by clearing the Master Mode Enable bit in the LIN Control Register 1 (LINCR1[MME] = 0b0)

2. Auto synchronization is enabled by setting LIN Auto Synchronization Enable (LINCR1[LASE] = 0b1)

The LINFlexD module may automatically synchronize to an incorrect baud rate without setting the Sync Field Error Flag in the LIN Error Status register (LINESR[SFEF]) in case Sync Field value is not equal to 0x55, as per the Local Interconnect Network (LIN) specification.

The auto synchronization is only required when the baud-rate in the slave node can not be programmed directly in software and the slave node must synchronize to the master node baud rate.

Workaround: There are 2 possible workarounds.

Workaround 1:

When the LIN time-out counter is configured in LIN Mode by clearing the MODE bit of the LIN Time-Out Control Status register (LINTCSR[MODE]= 0x0]):

1. Set the LIN state Interrupt enable bit in the LIN Interrupt Enable register (LINIER[LSIE] = 0b1)

2. When the Data Reception Completed Flag is asserted in the LIN Status Register (LINSR[DRF] = 0b1) read the LIN State field (LINSR[LINS])

3. If LINSR[LINS]= 0b0101, read the Counter Value field of the LIN Time-Out Control Status register (LINTCSR[CNT]), otherwise repeat step 2

4. If LINTCSR[CNT] is greater than 0xA, discard the frame.

When the LIN Time-out counter is configured in Output Compare Mode by setting the LINTCSR[MODE] bit:

1. Set the LIN State Interrupt Enable bit in the LIN Interrupt Enable register (LINIER[LSIE])

2. When the Data Reception Completed flag bit is asserted in the LIN Status Register (LINSR[DRF] = 0b1), read the LINSR[LINS] field

3. If LINSR[LINS]= 0b0101, store LINTCSR[CNT] value in a variable (ValueA), otherwise repeat step 2

4. Clear LINSR[DRF] flag by writing LINSR[LINS] field with 0xF

5. Wait for LINSR[DRF] to become asserted again and read LINSR[LINS] field

6. If LINSR[LINS] = 0b0101, store LINTCSR[CNT] value in a variable (ValueB), else repeat step 4

7. If ValueB – ValueA is greater than 0xA, discard the frame

Workaround 2:

Do not use the auto synchronization feature (disable with LINCR1[LASE] = 0b0) in LIN slave mode.

ERR008970: LINFlexD: Spurious bit error in extended frame mode may cause an incorrect Idle State

Description: The LINFlexD module may set a spurious Bit Error Flag (BEF) in the LIN Error Status Register (LINESR), when the LINFlexD module is configured as follows:

- Data Size greater than eight data bytes (extended frames) by configuring the Data Field Length (DFL) bitfield in the Buffer Identifier Register (BIDR) with a value greater than seven (eight data bytes)
- Bit error is able to reset the LIN state machine by setting Idle on Bit Error (IOBE) bit in the LIN Control Register 2 (LINCR2)

As consequence, the state machine may go to the Idle State when the LINFlexD module tries the transmission of the next eight bytes, after the first ones have been successfully transmitted and Data Buffer Empty Flag (DBEF) was set in the LIN Status Register (LINSR).

Workaround: Do not use the extended frame mode by configuring Data Field Length (DFL) bit-field with a value less than eight in the Buffer Identifier Register (BIDR) (BIDR[DFL] < 8)

ERR008080: LINFlexD: TX pin gets set to High-Z when in IDLE state

Description: LINFlex drives the buffer enable signal for it's transmit pin output (TX) to be '0' after transmitting the LIN frame. This causes the TX line to go to High-Z which will be an issue if the associated LIN transceiver has an internal "pull down".

Issue will also occur when module is configured in UART mode with the TX output pin becoming High-Z when idle.

Workaround: When operating in LIN mode, use a LIN transceiver with internal "pull up". If the transceiver has an internal "pull down", add an external "pull up".

When operating in UART mode, the issue can be worked around by enabling the internal pull up on the TX pin using the corresponding SIU_MSCR register.

ERR006350: LINFlexD: WLS feature cannot be used in buffered mode

Description: The Flexible Local Interconnect Network (LINFlex) module may not operate correctly if the Special Word Length (WLS) for enabling 12-bit data length is selected in the UniversalAsynchronous Receiver/Transmitter (UART) Mode Control Register (UARTCR) and the module is configured in the receive buffered mode.

Workaround: When WLS mode is required, always use the First In, First Out (FIFO) mode of the UART LINFLEX module by setting the Receive FIFO/Buffer mode bit of the UARTCR (UARTCR[RFBM]=1). In addition, the UART word length bits must be set (UARTCR[WL0 = 0b1] and UARTCR[WL1] = 0b1).

ERR010365: MC_ME: An invalid mode (Clock Usage) configuration error can occur if the Sigma-Delta PLL and PLL1 are switched on or off independently

Description: Performing a system mode transition which switches the Sigma-Delta PLL (SDPLL) and PLL1 to different states (using the PLL control bits in the Mode Entry Module Mode Configuration Registers (MC_ME_[mode]_MC[SDPLLON] and MC_ME_[mode]_MC[PLL1ON])) with any enabled peripherals selecting either of these as their clock source can cause an Invalid mode configuration interrupt (Clock Usage) (Interrupt Status Register MC_ME_IS[I_ICONF_CU] = 0b1).

Workaround: User must ensure any mode transition that switches on or switches off SDPLL and PLL1 sets both to the same on or off state in the target mode. This can be done by setting MC_ME_[mode]_MC[SDPLLON] and MC_ME_[mode]_MC[PLL1ON] both to zero or both to one.

ERR008228: MC_ME: Wakeup from STOP mode may lead to a system hang scenario

Description: If a wakeup is given to the microcontroller (MCU) within 10us during transition into STOP mode the system may hang. If the transition into STOP Mode is not complete and an abort command is issued waking up the MCU within 10us the phase lock loop (PLL) specification if violated leading to an unknown output from the PLL.

Workaround: While transitioning to STOP mode, do not generate a wake-up within 10us of the execution of STOP mode transition

ERR010317: MEMU: The Memory Error Management Unit may not report a consecutive error from the same address for some memories if the valid bit has been cleared for the initial error

Description: The device Memory Error Management Unit (MEMU) has the capability to block duplicate errors coming from the same address if the valid bit (VLD) is not cleared in reporting table. However if an entry is logged in the reporting table for an address and the valid bit is cleared, then if the next error occurs from the same address then the MEMU will not log this error in the reporting table. This is only the case where consecutive errors occur from the same address. If an error from a different address occurs in between errors from the same address then these will all be correctly reported.

The impacted memories are as follows:

SPT_TRAM_Access,

SPT_MOD_ORRAM_Access,

SPT_PDMA_OPRAM_Access,

e200z7b Data AHB,

e200z7b Data Cache,

e200z7b Instruction Cache,

e200z7b Data TCM (DMEM),

e200z7a Data AHB,

e200z7a Data Cache,

e200z7a Instruction Cache,

e200z7a Data TCM (DMEM)

Workaround: Application software should take appropriate action to the initial error reported by the MEMU module and should not rely on subsequent reporting of consecutive errors from the same address.

ERR007833: M_CAN: Incorrect frame transmission after recovery from Restricted Operation Mode

Description: When the Modular CAN (M_CAN) module detects a Message RAM Access Failure (MRAF) during a frame transmission, it sets the MRAF bit of the Interrupt Register (IR) and enters the Restricted Operation Mode (ASM) bit of the CAN Core Control Register [CCCR] is set.

During the first transmission after leaving the Restricted Operation Mode by resetting the CCCR.ASM bit, a frame with an unexpected identifier and control field may be transmitted and could be accepted and acknowledged by a receiver.

Workaround: Use the following procedure to exit from the Restricted Operation Mode:

Step 1: Cancel all pending transmission requests by writing 0xFFFF_FFFF to Transmit (TX) Buffer Cancellation Request (TXBCR) register

Step 2: Issue a clock stop request by setting the Clock Stop Request (CSR) bit of the CC Control Register (CCCR)

Step 3: Wait until the M_CAN sets the Initialization (INIT) and Clock Stop Acknowledge (CSA) bits of the CC Control Register (CCCR) to one

Step 4: First clear CSR bit

Step 5: Then clear INIT bit

Step 6: Wait until INIT is read as zero

Step 7: Issue a second clock stop request by setting CSR bit

Step 8: Wait until the M_CAN sets INIT and CSA bits to one

Step 9: Set the Configuration Change Enable (CCE) bit of the CC Control Register (CCCR), clear CSR and ASM bits in a single write operation.

Step 10: Restart M_CAN by clearing INIT bit.

Step 11: Configure the CAN operation mode by writing to the CAN Mode Request (CMR) field of the CC Control register.

Step 12: Request the transmissions cancelled by step one

ERR007832: M_CAN: Change of CAN operation mode during start of transmission causes incorrect behaviour

Description: In the Modular CAN (M_CAN) module, when the transmit Event FIFO is used and a change of CAN operation mode is requested (writing to the CAN Mode Request (CMR) field of the CAN Core Control Register (CCCR)) during the start of transmission, the following incorrect behaviors will occur.

Case 1: Change from classic CAN frame to Flexible Data rate (FD) frame with bit rate switching

The Extended Data Length (EDL) and Bit Rate Switch (BRS) bits of the related Tx Event FIFO element do not match with the transmitted frame type. They signal a CAN FD frame with bit rate switching (EDL and BRS bits are set) while a classic CAN frame was transmitted.

Case 2: Change from classic CAN to CAN FD without bit rate switching

The EDL bit of the related Tx Event FIFO element does not match with the transmitted frame type. It signals a CAN FD frame without bit rate switching (EDL is set) while a classic CAN frame was transmitted.

Case 3: Change from CAN FD with bit rate switching to CAN FD without bit rate switching

The BRS bit of the related Tx Event FIFO element does not match with the transmitted frame type. It signals a CAN FD frame without bit rate switching while a CAN FD frame with bit rate switching was transmitted.

Case 4: Change from CAN FD without bit rate switching to CAN FD with bit rate switching

The BRS bit of the related Tx Event FIFO element does not match with the transmitted frame type. It signals a CAN FD frame with bit rate switching while a CAN FD frame without bit rate switching was transmitted.

Case 5: Change from CAN FD with/without bit rate switching to Classic

The Message RAM Access Failure flag (MRAF) of the M_CAN module Interrupt Register (IR) is set (IR.MRAF = 1), the M_CAN switches to Restricted Operation Mode and the transmission is aborted.

Workaround: Wait that all the Transmission Request Pending n flags (TRPn) of the Tx Buffer Request Pending register (TXBRP) are cleared (TXBRP.TRPn = ‘0’) before changing the CAN operation mode.

ERR007834: M_CAN: Erroneous Interrupt flag after setting / resetting INIT during frame reception

Description: In the Modular Controller Area Network (M_CAN) module, when the Initialization bit (INIT) is set in the CAN Core Control Register (CCCR) during the reception of a frame, the first reception of a frame after clearing the INIT bit will be correctly received but the Message RAM Access Failure flag (MRAF) of the Interrupt Register (IR) will be set.

Workaround: If the Initialization (INIT) bit of the CC Control Register (CCCR) needs to be set during operation, proceed as follows:

Step 1: Issue a clock stop request by setting the Clock Stop Request (CSR) bit of CCCR register

Step 2: Wait until the M_CAN sets INIT and the Clock Stop Acknowledge (CSA) bits of the CCCR register to one

Before clearing INIT, first clear CSR.

ERR009070: M_CAN: FD frame abort may cause Protocol exception event and extended Bus Integration state

Description: In the Modular Controler Area Network module (M_CAN), when a transmission is aborted shortly before the transmission of the Flexible Data Frame (FDF) bit, a receiver will detect a recessive FDF bit followed by a recessive reserved (res) bit. In this case, the receiving M_CAN modules detect a protocol exception event and will enter Bus Integration state. The receivers should leave the Bus Integration state after 11 consecutive recessive bits.

Instead of starting to count the 11 recessive bits immediately after entering the Bus Integration state, the M_CAN needs to see at least one dominant bit before it starts to count the sequence of 11 recessive bits.

Workaround: Take into account that, in the described condition, the M_CAN module will take more time to recover as it will wait for 11 recessive bits after the first dominant bit on the network and not 11 recessive bits after entering the Bus Integration state.

ERR008045: M_CAN: Frame transmission in DAR mode

Description: In the Modular CAN (M_CAN) module, when the Disable Automatic Re-transmission bit is set in the CAN Core Control Register CCCR (CCCR[DAR]), the two following incorrect behaviors occur.

1- Transmission of a frame will cause the Event Type (ET) field of the Transmit Event FIFO Element to be incorrectly set to ‘0b01’ (transmit event) instead of ‘0b10’ (transmission in spite of cancellation).

2- When multiple messages are transmitted sequentially using the same Transmit buffer, after a successful transmission, the next transmission will not start if it is requested before the CAN bus becomes idle. This message is then treated as if it had lost arbitration.

Workaround: Do not use the same transmit buffer for consecutive transmissions in DAR mode.

Or wait at least for 4 CAN bit times after successful transmission before requesting the next transmission from the same transmit buffer.

ERR009069: M_CAN: Incorrect activation of MRAF interrupt

Description: In the Modular CAN (M_CAN) module, the Message RAM Access Failure flag (MRAF) of the Interrupt register (M_CAN_IR) may be set although there was no Message RAM access failure.

This behavior occurs when the module is receiving a frame and:

- the module is in the Error Passive sate, and
- the Receive Error counter (REC), field of the Error Counter register (M_CAN_ECR), has the value 127.

Workaround: In processing the interrupt from the M_CAN module, if the Error Passive flag of the Protocol Status register (M_CAN_ECR[RP]) is set (M_CAN_ECR[RP] == 1) and the Receiver Error counter equals 127 (M_CAN_ECR[REC] == 127), clear the M_CAN_IR.MRAF flag and exit the interrupt handler.

ERR009226: M_CAN: Message loss if message RAM access is not granted prior to the next received message

Description: If the Modular Controller Area Network (M_CAN) module needs to store a received frame, and the Message RAM / RAM Arbiter does not respond in time, this message cannot be stored completely and it is discarded with the reception of the next message. The Message RAM Access Failure flag (MRAF) of the M_CAN Interrupt Register (IR) gets correctly set (IR[MRAF]=1) but the next received message may be incompletely stored. In this case, the respective receive buffer or receive FIFO element contains inconsistent data.

Workaround: Configure the RAM Watchdog register (RWD) to the maximum expected Message RAM access delay. In case the Message RAM / RAM Arbiter does not respond within this time, the Watchdog Interrupt flag (WDI) of the Interrupt Register (IR) will be set. The frame received after IR[MRAF] has been set and with IR[WDI] set must be discarded.

ERR007828: M_CAN: Message reception and transmission directly after detection of Protocol Exception Event

Description: In the Modular CAN (M_CAN) module, if a Flexible Data rate (FD) frame is received and the reserved bit (res) following the FD frame format bit (FDF) is recessive, the protocol controller correctly detects a Protocol Exception Event. The reception of the message is not finished and the message is discarded but the first message after this Protocol Exception Event will generate the following wrong behaviors.

Case 1: Message reception directly after Protocol Exception Event

The Message RAM Access Failure flag (MRAF) of the M_CAN Interrupt Register (IR) is set even if there was no incorrect access to the Message RAM and the frame has been correctly received.

Case 2: Message transmission directly after Protocol Exception Event

The first frame transmitted after a Protocol Exception Event is transmitted with a faulty frame format. In this case, the MRAF bit is not set.

The other nodes on the CAN network will react to this faulty format and generate error frames causing the M_CAN cell to resend the frame, with a correct format.

Workaround: For reception of messages: ignore the MRAF error. The MRAF signal is primarily intended to validate the correct integration of the M_CAN within a device.

For transmission of messages: the other node(s) will detect the error and trigger the resend of the frame.

ERR011469: M_CAN: Message transmitted with wrong arbitration and control fields

Description: Under the following conditions, the Modular Controller Area Network (M_CAN) module may transmit a message with wrong ID, format fields, and Data Length Code (DLC):

- M_CAN is in state “Receiver” (M_CAN_PSR[ACT] = 10), with no pending transmission
- A new transmission is requested before the 3rd bit of Intermission is reached
- The CAN bus is sampled dominant at the third bit of Intermission, which is treated as Start of Frame (SoF) (see: ISO11898-1:2015 Section 10.4.2.2). This dominant level at the 3rd bit of Intermission may result from an external disturbance or may be transmitted by another node with a significantly faster clock.

Under the conditions listed above, the following can occur:

- The shift register is not loaded with ID, format fields, and DLC of the requested message
- The M_CAN will start arbitration with wrong ID, format fields, and DLC on the next bit
- In case the ID wins arbitration, a CAN message with wrong ID, format fields, and DLC is transmitted, however this message will have a valid Cyclic Redundancy Check (CRC) and therefore will appear to the receiver as a valid frame with no errors. The incorrect format fields and/or DLC may cause the data field to be truncated or padded, but the CRC will be computed after these changes and will be valid for the transmitted frame.
- In case this message is acknowledged, the ID stored in the Tx Event FIFO is the ID of the requested Tx message and not the ID of the message transmitted on the CAN bus. No error is detected by the transmitting M_CAN
- If the message loses arbitration or is disturbed by an error, it is re-transmitted with correct arbitration and control fields.

Workaround: If another transmission is already pending or the M_CAN is not in state “Receiver” (i.e. When M_CAN_PSR[ACT] != 10), a new transmission may be requested without causing this issue.

If no transmission is pending and the M_CAN is in state “Receiver” (M_CAN_PSR[ACT] = 10) then the application must avoid requesting the new transmission during the critical time window between the sample points of the 2nd and 3rd bit of Intermission.

To accomplish this, the application software can evaluate the Rx Interrupt flags M_CAN_IR[DRX], M_CAN_IR[RF0N], and M_CAN_IR[RF1N] which are set at the last bit of End of Frame (EoF) when a received and accepted message is validated. The last bit of EoF is followed by three bits of Intermission. Therefore the critical time window has safely terminated three bit times after the Rx interrupt. Now a transmission may be requested by writing to M_CAN_TXBAR.

After the interrupt, the application has to take care that the transmission request for the CAN Protocol Controller is activated before the critical window of the following reception is reached.

If the application cannot reliably avoid the critical time interval described above, another option is to implement additional application-defined protocol checks within the data payload of each message. This protocol can be used by the receiving node to detect messages that have been corrupted by this issue, discard them, and request retransmission. Such a protocol is already recommended for safety-relevant communication as described in the Safety Manual for this device (See sub-section “Fault-tolerant communication protocol” in the Software Requirements chapter).

ERR050016: M_CAN: Retransmission in DAR mode due to lost arbitration at the first two identifier bits

Description: When the Modular Controller Area Network module (M_CAN) is configured in DAR mode (CCCR.DAR = ’1’) the Automatic Retransmission for transmitted messages that have been disturbed by an error or have lost arbitration is disabled. When the transmission attempt is not successful, the Tx Buffer’s transmission request bit (TXBRP.TRPxx) shall be cleared and its cancellation finished bit (TXBCF.CFxx) shall be set. When the transmitted message loses arbitration at one of the first two identifier bits, it may happen, that instead of the bits of the actually transmitted Tx Buffer, the TXBRP.TRPxx and TXBCF.CFxx bits of the previously started Tx Buffer (or Tx Buffer 0 if there is no previous transmission attempt) are written (TXBRP.TRPxx = ’0’, TXBCF.CFxx = ’1’). If in this case the TXBRP.TRPxx bit of the Tx Buffer that lost arbitration at the first two identifier bits has not been cleared, retransmission is attempted. When the M_CAN loses arbitration again at the immediately following retransmission, then actually and previously transmitted Tx Buffer are the same and this Tx Buffer’s TXBRP.TRPxx bit is cleared and its TXBCF.CFxx bit is set.

This erratum is limited to the case when the M_CAN loses arbitration at one of the first two transmitted identifier bits while in DAR mode. The problem does not occur when the transmitted message has been disturbed by an error.

Workaround: No workaround is necessary as there is no lasting impact to this erratum. When this issue occurs, only one retransmission attempt will be made.

ERR008299: M_CAN: Setting the Configuration Change Enable (CCE) bit during a transmission scan can halt CAN tranmissions

Description: The Modular CAN (M_CAN) Transmission Handler normally scans for Transmission Buffers with pending transmission requests. Pending transmissions are indicated by Transmit (Tx) Buffer Request Pending (TXBRP) register bits being set. If the user decides to reconfigure the M_CAN module by setting the Configuration Change Enable (CCE) bit of the CAN Core Control Register (CCCR) while the Transmission Handler is scanning for pending transmission requests, the TXBRP register can be cleared and the Transmission Handler FSM is halted. This has the effect of halting any M_CAN message transmission.

After the Initialization (INIT) and CCE bits have been cleared by the Host, the M_CAN is unable to transmit messages. When the Host requests a transmission by writing to the Tx Buffer Add Request (TXBAR) register, the respective Tx Buffer Request Pending bit in register TXBRP is set, but the Transmission Handler will not start the requested transmission.

Workaround: If the M_CAN configuration needs to be changed while in use, place the M_CAN module in a halted state (similar to preparing a Power Down (Sleep Mode) as described in the Reference Manual). The following steps must be used:

Step 1: Cancel all pending transmission requests by writing 0xFFFF_FFFF to Tx Buffer Cancellation Request (TXBCR) register

Step 2: Issue a clock stop request by setting the Clock Stop Request (CSR) bit of the CC Control Register (CCCR)

Step 3: Wait until the M_CAN sets the Initialization (INIT) and Clock Stop Acknowledge (CSA) bits of the CC Control Register (CCCR) to one

Step 4: First clear the CSR bit of the CC Control Register (CCCR)

Step 5: Then, performing a separate write operation, clear the INIT bit of the CC Control Register (CCCR)

Step 6: Wait until INIT bit is read as zero

Step 7: Issue a second clock stop request by setting CSR bit

Step 8: Wait until the M_CAN sets INIT and CSA bits to one

Step 9: Set the Configuration Change Enable (CCE) bit of the CC Control Register (CCCR) and clear CSR bit in a single write operation

Now the M_CAN configuration can be modified.

ERR007594: M_CAN: Transmitted bit in control field is falsified when using extreme bit time configurations

Description: When the Modular CAN (M_CAN) module transmits a frame, and when the values of both the Time segment after sample point (TSEG2) and the Baud Rate Prescaler (BRP) fields of the Bit Timing and Prescaler Register (BTP) are 0 (zero), one bit in the control field will be transmitted with an erroneous value. The effect is different for frames to be transmitted in Classic CAN format or CAN Flexible Data-rate (FD) format.

Case 1: Transmission of a classic CAN frame with the 2-bit wide CAN Mode Enable field (CME) of the CC Control Register (CCCR) is 0b00.

If the frame under transmission has a 29-bit identifier where the most significant bit (bit 28 of identifier) is “1”, then the reserved bit following the RTR bit will be transmitted recessive instead of dominant. As a consequence, this frame will be incorrectly interpreted as a CAN FD frame by any node with the CAN FD mode enabled that will therefore generate an error frame. Nodes supporting only the classic CAN mode will ignore this bit (reserved) and correctly receive the frame.

Case 2: For a transmission of a CAN FD Frame with the CME of the CCCR not equal to 0b00.

If the FD frame under transmission has a 29-bit identifier where the most significant bit is “0”, or has an 11-bit identifier, then the FD Frame Format (FDF) bit of the frame is transmitted dominant instead of recessive and the rest of the frame is transmitted in Classic CAN format with an incorrect Data Length Code (DLC) field.

Workaround: Do not use bit timing configurations where BTP.TSEG2 and BTP.BRP are both zero for CAN FD communication.

ERR011456: M_CAN: Tx FIFO message sequence inversion

Description: A message sequence inversion can occur if the following condition occurs. The condition requires that there be two transmit (TX) messages in the TX FIFO and a higher priority non-Tx FIFO message is added, such as to a TX dedicated buffer. The sequence of events is as follows:

Transmission of TX FIFO message 1 is started (from position 1 of the TX FIFO).

A second message is in the second position of the TX FIFO

The message buffer then contains:

Position 1: TX FIFO message 1 (transmission ongoing)

Position 2: TX FIFO message 2

Position 3: -

A higher priority non-TX FIFO message is input into the CAN module and will be the next message to be transmitted. This message will be inserted into the message flow after the message that is being transmitted and the previous second message is pushed down in the transmission buffer.

After the following two message scans the output pipeline has the following content:

Position 1: TX FIFO message 1 (transmission ongoing)

Position 2: non TX FIFO message with higher CAN priority

Position 3: TX FIFO message 2

If the first message that is being transmitted is not successful, due to lost arbitration or CAN bus error, the higher priority (non-TX FIFO) message will be transmitted. The aborted message is put back in the output pipeline. However, instead of being put behind the non-TX FIFO message, it is placed after the previous second message in the TX FIFO.

The output pipeline will then appear as follows:

Position 1: non TX FIFO message with higher CAN priority (transmission ongoing)

Position 2: TX FIFO message 2

Position 3: TX FIFO message 1

At this point, TX FIFO message 2 is in the output pipeline in prior to TX FIFO message 1 and they are transmitted in that order, resulting in a message sequence inversion.

The scope of erratum describes the case when the M_CAN uses both dedicated TX Buffers and a TX FIFO (TXBC.TQFM = ’0’) and the messages in the TX FIFO do not have the highest internal CAN priority. The above described sequence inversion may also occur between two non TX FIFO messages (TX Queue or dedicated TX Buffers) that have the same CAN identifier and that should be transmitted in the order of their buffer numbers (not the intended use).

Workaround: When transmitting messages from a dedicated TX Buffer with higher priority than the messages in the TX FIFO, choose one of the following workarounds:

First Workaround

Use two dedicated TX Buffers, for example, use TX Buffers 4 and 5 instead of the TX FIFO. The pseudo-code below replaces the function that fills the TX FIFO.

Write message to TX Buffer 4

Transmit Loop:

• Request TX Buffer 4 by setting TXBAR.AR[27] bit to 1

• Write message to TX Buffer 5

• Wait until transmission of TX Buffer 4 completed by reading IR.TC and TXBTO.TO[27]

• Request TX Buffer 5 by setting TXBAR.AR[26] bit to 1

• Write message to TX Buffer 4

• Wait until transmission of TX Buffer 5 completed by reading IR.TC and TXBTO.TO[26]

Second Workaround

Assure that only one TX FIFO element is pending for transmission at any time. The TX FIFO elements may be filled at any time with messages to be transmitted, but their transmission requests are handled separately. Each time a TX FIFO transmission has completed and the TX FIFO gets empty (IR.TFE = ’1’) the next TX FIFO element is requested.

Third Workaround

Use only a TX FIFO. Send the message with the higher priority also from TX FIFO. However, this workaround has a drawback: the higher priority message has to wait until the preceding messages in the TX FIFO have been sent.

ERR011457: M_CAN: Unexpected High Priority Message (HPM) interrupt

Description: The High Priority Message (HPM) Interrupt flag IR.HPM is set erroneously in the following cases:

Configuration A:

- At least one Standard Message ID Filter Element is configured with priority flag set (S0.SFEC = “100”/“101”/“110”)
- No Extended Message ID Filter Element configured
- Non-matching extended frames are accepted (GFC.ANFE = “00”/“01”)

In configuration A, the HPM interrupt flag is set erroneously on reception of a non-high-priority extended message under the following conditions:

(1) A standard HPM frame is received, and accepted by a filter with priority flag set. Interrupt flag IR.HPM is set as expected for this message.

(2) Next an extended frame is received and accepted because of GFC.ANFE configuration. Interrupt flag IR.HPM is set erroneously for this message.

Configuration B:

- At least one Extended Message ID Filter Element is configured with priority flag set (F0.EFEC = “100”/“101”/“110”)
- No Standard Message ID Filter Element configured
- Non-matching standard frames are accepted (GFC.ANFS = “00”/“01”)

The HPM interrupt flag is set erroneously on reception of a non-high-priority standard message under the following conditions:

(1) An extended HPM frame is received, and accepted by a filter with priority flag set. Interrupt flag IR.HPM is set as expected for this message.

(2) Next a standard frame is received and accepted because of GFC.ANFS configuration. Interrupt flag IR.HPM is set erroneously for this message.

Workaround: For configuration A:

Set up an Extended Message ID Filter Element with the following configuration:

- F0.EFEC = “001”/“010” - select Rx FIFO for storage of extended frames
- F0.EFID1 = any value – value not relevant as all ID bits are masked out by F1.EFID2
- F1.EFT = “10” - classic filter, F0.EFID1 = filter, F1.EFID2 = mask
- F1.EFID2 = zero – all bits of the received extended ID are masked out

Now all extended frames are stored in Rx FIFO 0 or Rx FIFO 1 depending on the configuration of F0.EFEC.

Configuration B:

Set up a Standard Message ID Filter Element with the following configuration:

- S0.SFEC = “001”/“010” - select Rx FIFO for storage of standard frames
- S0.SFID1 = any value – value not relevant as all ID bits are masked out by S0.SFID2
- S0.SFT = “10” - classic filter, S0.SFID1 = filter, S0.SFID2 = mask
- S0.SFID2 = zero – all bits of the received standard ID are masked out

Now all standard frames are stored in Rx FIFO 0 or Rx FIFO 1 depending on the configuration of S0.SFEC.

ERR010331: NAL: Trace connections to the device are lost if a device reset occurs

Description: During reset, the Nexus Aurora transmit pins (TXxN/TXxP, where x is 0 through 3) are put into a high impedance state until either self-test completes or the Self-Test Control Unit (STCU) determines that self-test is disabled. In addition, the system clock frequency is reset and the system clock is set to the Internal RC oscillator. Therefore, if the device is reset, the system clock is reset to a frequency that is less than 1/10 of the Nexus trace clock, the connection to a tool will be disconnected. The tool will have to establish a new connection with the device.

Workaround: The user should expect the trace connection to be lost through a reset. Tools will have to be reconnected following the reset and the pins re-enabled.

ERR008340: NPC: EVTO_B toggles instead of remaining asserted when used by the DTS if Nexus is not enabled

Description: When the Development Trigger Semaphore (DTS) module asserts its trigger output on the Event Out (EVTO_B) pin, the EVTO_B pin will toggle instead of remaining asserted low if the Nexus Port Controller (NPC) Port Configuration Register MCKO Enable (NPC PCR[MCKO_EN]) bit is set to 0. If the NPC PCR[MCKO_EN] bit is set to 1, the EVTO_B pin behaves as expected and remains asserted low as long as the DTS asserts its trigger output.

Workaround: Always set the MCKO_EN bit to 1 when using the DTS trigger out function.

ERR010479: NPC: Nexus enable required for mode changes when a debugger is attached

Description: If the Nexus interface is enabled in the the e200zx cores, even if trace (program, data, ownership, watchpoint, data acquisition) is not enabled, the Nexus Port Controller (NPC) tracing must be enabled to allow mode changes via the Mode Entry module if debug mode is enabled (debugger connected to the MCU) since some Nexus trace messages are automatically generated regardless whether any trace mode is disabled. Nexus is enabled in the core if any Nexus feature is accessed by a tool (executing the Nexus_enable command to use the Nexus Read/Write Access feature to access memory).

Workaround: NPC tracing must be enabled by enabling the Message Clock Output (MCKO) in the NPC Port Configuration Register (NPC_PCR) when a debugger is connected to allow messaged to exit the core Nexus module. In addition, the Full Port Mode bit should also be set.

ERR010340: NZxC3: ICNT and HIST fields of a Nexus message are not properly reset following a device reset

Description: Following reset, if instruction trace is enabled in the Nexus e200zx core Class 3 trace client (NZxC3), the e200zx core transmits a Program Trace – Synchronization Message (PT-SM). The PT-SM includes the full execution address and the number of instructions executed since the last Nexus message (ICNT) information. However, the ICNT and the Branch History field (HIST), if Branch History trace is enabled, are not properly cleared when this message is transmitted. This may cause unexpected trace reconstruction results until the next Nexus Program Trace Synchronization Message (Program Trace – Direct Branch Message with Sync, Program Trace – Indirect Branch Message with Sync, or Program Trace – Indirect Branch History Message with Sync).

In Branch History mode, the first indirect branch following the reset (and the initial PT-SM) will contain the branch history prior to the reset plus the branch history after reset. However, there is no way to determine which branches occurred prior to reset and which followed reset.

Workaround: If not using branch history trace mode, to recreate the proper trace, the tool should take into account that the ICNT field is not cleared by the first PT-SM. The previous ICNT will be added to new ICNT value in the subsequent Nexus message. This may require extra processing by the tool.

If using branch history mode, then an accurate reconstruction of the executed code just before and just after reset may not be possible. Trace reconstruction can be recovered after the next indirect branch message.

On devices that bypass the Boot Assist Flash (BAF) or Boot Assist Module (BAM) after reset (in other words, the System Status and Configuration Module [SSCM] boots directly to user code if a valid Reset Configuration Half-Word is found), perform an indirect branch instruction shortly after reset to reset the ICNT (and HIST if Branch History mode is enabled). A full program trace synchronization message will be generated after 256 direct branches even if there is no indirect branches. This will allow the tool to recover the trace reconstruction from that point onward.

On devices that always execute the BAF or BAM, an indirect branch will occur during the BAF/BAM execution and the tool trace will be re-synchronized prior to the execution of user code.

ERR010638: NZxC3: Nexus messaging becomes corrupted if more than one master (including a device core) is active and the core is subsequently disabled.

Description: This errata applies to the condition where there is more than one master active on the Nexus Port Controller (NPC) module, and one or more of these masters is a device core. In this situation, if a mode transition is initiated to a mode where that core is disabled, with the clock gated (as configured in the relevant core control register MC_ME_CCTLx for the requested mode) then the Nexus3 interface issues a core debug status message indicating the transition into core STOP mode (core disabled with clock gated). It is possible during this message transmission, that the core Nexus clock gets disabled. This blocks the Nexus interface, and the message data can be left pending on the interface until the core clock resumes. This causes overflow of the NAL (Nexus Aurora Link) FIFO and corrupts the Aurora protocol.

Workaround: Possible workarounds are:-

1) If Nexus traces are enabled then mode transitions that disable device cores should not be executed.

2) If Nexus traces are enabled and a core is needed to be stopped, then that core should use PowerPC ‘wait' instruction instead of stop mode.

ERR007905: PIT: Accessing the PIT by peripheral interface may fail immediately after enabling the PIT peripheral clock

Description: If a write to the Periodic Interrupt Timer (PIT) module enable bit (PIT_MCR[MDIS]) occurs within two bus clock cycles of enabling the PIT clock gate in the MC_CGM (Clock Generation Module) register, the write will be ignored and the PIT will not be enabled.

Workaround: After enabling the PIT clock in the MC_CGM, insert a read of the PIT_MCR register before writing to the PIT_MCR register. This guarantees a minimum delay of two bus clocks to guarantee the write is not ignored.

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.

ERR010486: PLLDIG: Device may not exit the reset sequence if a reset occurs whilst PLL1 is running with Modulation enabled.

Description: PLL1 has ability to run with modulation enabled, which is controlled by PLLDIG_PLL1FM[MODEN] bit. The PLL1 nominal frequency (fpll1_phi) output must be configured such that the output is within the operating frequency as outlined in the Reference manual, this includes the Modulation Depth (MD) configured . When the PLL1 has locked and is running with modulation enabled and a reset (from any source) is applied to the system then the PLL1 output can overshoot for a short duration which can cause the system to become stuck in reset.

Workaround: There are 3 potential workarounds:

1. Run PLL1 with Modulation disabled (PLLDIG_PLL1FM[MODEN] = 0b0)

2. Use an external watchdog circuit and power supply monitor to determine if the MCU has become unresponsive. The external circuitry should be configured to assert the device power-on reset signal (VREG_POR_B) if the device becomes unresponsive. This will ensure that the device can safely return to normal operation as per the recommendations of the device safety manual.

3. Configure the PLL1 output frequency such that the MD of the mode configured (PLLDIG_PLL1FM[MODSEL]) plus the PLL nominal frequency (fpll1_phi) is 10% lower than the frequency specified in the device reference manual for maximum clock frequencies for CORE1_CLK/CORE2_CLK.

ERR009827: PMC: Device may not exit the reset sequence if a low voltage detect event occurs during execution of offline or online self-test

Description: The device may not exit the reset sequence if a low voltage detect (LVD) event occurs during offline or online Self Test Control unit (STCU2) built-in self-test (BIST) execution, on any LVD monitored supply other than VDD_HV_IO.

Workaround: Use an external watchdog circuit to determine when the MCU is not responding because of brownout conditions on external power supply and adhere to the Safety Manual guidelines when the stuck in reset condition occurs. Asserting the device power-on reset signal (VREG_POR_B) when the MCU becomes non-responsive or when any power supply is out with the operating range will bring the device back into normal operation.

It is also possible to bypass LVD assertion for all supplies other than VDD_HV_IO during self test execution using DCF programming to work around the problem. This is achieved by setting dcl_ips_0 bit 28 and bits 26:21.

ERR010259: PMC: Device may not exit the startup reset sequence under certain power sequencing conditions until the STCU watchdog time-out interval elapses

Description: During startup, if the VDD_HV_PMU and VDD_HV_IO supplies trigger their respective low-voltage detects (LVDs) without the power-on reset signal (VREG_POR_B) asserted, the device may not exit the reset sequence until the Self Test Control Unit (STCU) watchdog timer (WDG) interval elapses and performs a device reset. This time is factory programmed to 1.5 seconds but can be user defined by programming the STCU_WDG DCF record in UTEST flash. The watchdog timer counter will elapse and bring the part out of reset regardless of whether the STCU is programmed to run offline self-test.

Workaround: To prevent the issue from occurring, VREG_POR_B should be asserted anytime the VDD_HV_IO and VDD_HV_PMU supplies are below the operating range stated in the device data sheet (this is already a requirement for external regulation mode).

ERR011156: PMC: During online PMC self-test if RESET_B is asserted then a spurious destructive or power on reset may occur and be reported as an LVD/HVD event in the RGM_DES register.

Description: When the device is configured to run PMC (power management controller) online self-test (via PMC_SELF_TEST_UM_VD_REG[ST_MODE] = 0b1) or single VD test (via PMC_SELF_TEST_UM_VD_REG[ST_MODE] = 0b10) there are two scenarios and configurations where if an external reset is applied via RESET_B (when configured as long functional reset via RGM_FESS[SS_EXR] = 0b0) during the PMC online self test of any supply that has the HVD/LVD unmasked in the dcl_ips_0 register will cause the CGM (clock generation module) dividers to be bypassed, causing an unexpected destructive or POR reset. These scenarios arrive since the LVD Self Test Time Window Register (PMC_STTW) value will be written with a value based on a certain clock speed, but the CGM being bypassed results in a faster clock being applied, effectively meaning the self test time window decreases. The 2 cases are described as follows:

1. PMC module clock configured to be less than 16MHz. In this case, when reset occurs, the CGM clock divider gets bypassed and the PMC module clock changes to 16MHz, and this causes PMC self-test timing window to reduce, which in turn causes failures and a resulting reset (destructive or POR) to occur.

2. PLL and PCFS is enabled and PMC module clock frequency are configured to be higher than 16MHz. In this case, when RESET occurs, the PCFS ramp down slowly reduces the frequency, but as the CGM divider again gets bypassed, the system clock increases (e.g before reset, with PMC module clock of 60MHz when reset occurs this will become higher by the factor of CGM divider and then gradually reduce due to PCFS). This increase in frequency causes the similar effect as in 1st case.

Either scenario will be reported as an LVD/HVD event in the RGM_DES register with bits for either the supply that was under test, the subsequent supply to be tested, or both.

Workaround: To avoid any unwanted reset during PMC online self test ensure that when self-test is initiated via PMC_SELF_TEST_UM_VD_REG[ST_MODE] = 0b1, or single VD test via PMC_SELF_TEST_UM_VD_REG[ST_MODE] = 0b10 the following clock configuration rules are applied:

1. PMC module clock frequency is greater than or equal to 16MHz or if the PMC module clock is configured to be less than 16MHz then programming the PMC_STTW register with a value of 0x172 (corresponding to 16MHz + 10% value) then no issue will occur.

2. PCS ramp-down is disabled on the selected clock source for the PMC module.

ERR008580: PMC: If VDD_HV_PMU is applied before VDD_HV_IO in internal regulation mode, an over-voltage condition on VDD can occur

Description: If the 3.3 V PMU Supply Voltage (VDD_HV_PMU) is applied before the Main GPIO 3.3 V Supply Voltage (VDD_HV_IO) during power up in internal regulation mode, an over-voltage spike on the Core Supply Voltage (VDD) may occur for a short duration. The over-voltage condition typically lasts for approximately 400us and can peak as high as 2.5V, dependent on external components.

Workaround: Either ramp VDD_HV_IO and VDD_HV_PMC together or ramp VDD_HV_IO before VDD_HV_PMC when using internal regulation mode

ERR010414: PMC: Low Voltage Detect (LVD) self test may incorrectly indicate a self test fail if the supply is outwith its operating range during power up

Description: The Power Management Controller (PMC) executes a self test of the Low Voltage Detect (LVD) and High Voltage Detect (HVD) mechanisms during device start up (at Phase 3 of the reset sequence) after a power on reset or a reset due to an LVD event. If a scenario occurs during this self test where a ramping-up supply is out of the operating range when that supply's monitor is being tested, the PMC self test would indicate a failure correctly once self test execution has completed (indicated by PMC_SELF_TEST_UM_VD_REG[ST_RESULT]=0b0 when PMC_SELF_TEST_UM_VD_REG[ST_DONE]=0b1). This means that if a failure is evident after testing it is not certain that a genuine failure has occured or whether the supply may have been outwith the operating ranges whilst the test was executing

Workaround: In the event of a PMC LVD/HVD self test failure (PMC_SELF_TEST_UM_VD_REG[ST_RESULT]=0b0), the user should wait until all supplies return to the correct operating range, and then perform a software initiated PMC self test through PMC_SELF_TEST_UM_VD_REG[ST_MODE]=0b01. Once the software initiated self test has completed it will now indicate the correct status via the result register PMC_SELF_TEST_UM_VD_REG[ST_RESULT].

ERR009498: PMC: Performing the PMC self-test with PLL0/PLL1 selected as system clock source can cause an unexpected reset

Description: When the system clock is set to PLL0 or PLL1 and the Power Management Controller (PMC) software triggered self-test is initiated using the Voltage Detect User Mode Test Register ((PMC.SELF_TEST_UM_VD_REG[ST_MODE] = 0b01) or (PMC.SELF_TEST_UM_VD_REG[ST_MODE] = 0b10)) an unexpected reset could be issued to the system. This is because the value of DCF record dcl_pmc_self_test_time_gap[19:12] is only sufficient to operate the exit from last LVD/HVD being run for self-test when the system clock is set to the 16MHz Internal RC Oscillator (IRC).

Workaround: The self-test time per LVD/HVD is calculated by (value in DCF record dcl_pmc_self_test_time_gap[11:0]) * (PBRIDGE_0_CLK time period).

The total PMC self-test time is 10 * (self-test time per LVD/HVD) + (value in DCF record dcl_pmc_self_test_gap[12:19] * (PBRIDGE_0_CLK time period))

There are 3 possible workarounds detailed below. Note that when PMC self test is running (PMC_SELF_TEST_UM_VD_REG[ST_DONE] = 0) any register writes to PMC address space are prohibited.

1) Change the system clock source to the IRC before starting the software initiated self-test. A destructive reset will be occur when the test completes and this workaround has no impact to boot-up time after power on.

2) When running with PMC.SELF_TEST_UM_VD_REG[ST_MODE] = 0b01 program dcl_pmc_self_test_time_gap[11:0] with value so that 21 us is achieved for PBRIDGE_0_CLK frequency and dcl_pmc_self_test_time_gap[19:12] = 0xFF. Use the PMC Control Register (PMC.PMCCR) to disable the ADC HVD before the start of self-test and do not re-enable until 20 us after PMC self-test completes (PMC_SELF_TEST_UM_VD_REG[ST_DONE] = 1).

This will incur an additional boot time of up to 3.2 ms to the boot-up time on every destructive or power on reset. This time is calculated for achieving PMC self test with a PBRIDGE_0_CLK frequency of 66.66 MHz, programming of dcf_pmc_self_test_time_gap[11:0] = 0x579 and programming of dcl_pmc_self_test_gap[19:12] = 0xFF.

3) When running with PMC.SELF_TEST_UM_VD_REG[ST_MODE] = 0b10 (single LVD/HVD self-test as per programming of PMC_SELF_TEST_UM_VD_REG[VD_ST_CTRL]) Program dcl_pmc_self_test_time_gap[11:0] with a value so that 21 us is achieved for PBRIDGE_0_CLK frequency and dcl_pmc_self_test_time_gap[19:12] = 0xFF. Use the PMC Control Register (PMC.PMCCR) to mask the LVD or HVD event programmed in PMC_SELF_TEST_UM_VD_REG[VD_ST_CTRL] before initiating PMC self-test and do not re-enable until 20 us after PMC self-test completes (PMC_SELF_TEST_UM_VD_REG[ST_DONE]=1).

This workaround will incur an additional boot time of upto 3.2 ms to the boot-up time on every destructive or power on reset. This time is calculated for achieving self-test with a PBRIDGE_0_CLK frequency of 66.66 MHz, programming of dcf_pmc_self_test_time_gap[11:0] = 0x579 and programming of dcl_pmc_self_test_gap[19:12] = 0xFF.

ERR008745: PMC: Temperature Event Status Register flag bits may be incorrectly set after reset

Description: Temperature event status flags in the Power Management Controller (PMC) Temperature Event Status Register (ESR_TD[TEMP1_2,TEMP1_0,TEMP0_2,TEMP0_0]) may be set incorrectly after device reset. This could cause false reset events if the application code enables reset using the Temperature Reset Event Enable register. (PMC_REE_TD).

Workaround: Post device reset, check the status flags(PMC_ESR_TD[TEMP1_2,TEMP1_0,TEMP0_2,TEMP0_0]) and if found to be set use the SAR ADC to measure the on-chip Temperature Sensor and determine if it is the associated over temperature event that caused the flag to be set. Only if the ADC confirms the temperature is close to the over temperature condition then can the flag be considered valid. Incorrect flag bits can then be cleared by writing a one to the bit. This must be done prior to enabling any reset bits in the Temperature Reset Event Enable register (PMC_REE_TD) to avoid any false reset events.

ERR010345: PMC: The Power Management Controller (PMC) module Analog Front End (AFE) Low Voltage Detect (LVD) interrupt may not work as a valid source of interrupt

Description: The Power Management Controller (PMC) module interrupt (interrupt request number (IRQ) 477) is used to signal Low Voltage Detect (LVD) events on the internal AFE voltage regulators (VREGs). If these are enabled by setting the associated bits for each individual VREG in the AFE Interrupt Enable Register (PMC_AFE_INTR_ENA[AFE_INT_EN_VREGx] = 0b1 and PMC_AFE_INTR_ENA[IE_EN] = 0b1) then any interrupt request generated only occurs for two peripheral bridge clock cycles. If another interrupt is being serviced when the request is raised then the AFE LVD interrupt subroutine will sometimes not be executed, meaning it cannot be used reliably.

Workaround: Users should not consider the PMC module interrupt (IRQ 477) to be a valid source of interrupt. However, if an AFE VREG LVD event is configured as a source of destructive reset (by setting the corresponding bit in the MCB_AFE_LVD_MASK register) then the Reset Generation Module (MC_RGM) will latch the request for destructive reset in the case of the LVD event.

ERR011032: PMC: Unexpected events when an LVD occurs on a supply with its LVD masked

Description: Device low voltage detect (LVD) can be masked/disabled in the following situations:

a) Optionally during normal functional mode using the PMC_PMCCR register.

b) During online and offline STCU self-test, by default all LVDs and HVDs are masked except VDD_HV_IO LVD using the DCL_IPS_0 control in UTEST.

c) During PMC self test the LVD/HVD for the supply under PMC self test and the subsequent supply to be under self test are automatically masked

The following symptoms occur if an LVD occurs on a supply whose LVD has been masked/ disabled:

1) If LVD occurs on VDD_HV_PMC while it is masked: During online STCU, PMC self-test or normal operation, reset will not be masked and a reset event (corresponding to External Reset) would occur. During offline STCU operation where RESET_B is already asserted, LBIST 4 partition will fail self-test and the FCCU and MC_RGM status will not reflect the LVD event.

2) If LVD occurs on VDD_HV_IO while it is masked: The 16MHz IRCOSC will stop working and its output may glitch, but will restart when supply recovers above LVD level. This could lead to the device becoming stuck in reset.

3) If LVD occurs on VDD_HV_ADC while it is masked: The SAR-ADCs will stop converting, but recommence when supply recovers above LVD level. The ADC cannot be used during STCU execution but should not be relied upon during PMC self-test execution or if PMCCR[LVD_ADC_ENABLE] has been written to 0b0 and the supply can fall below operating level.

4) If LVD occurs on VDD_HV_FLA while it is masked: flash reads and write/erase operations can be impacted.

Workaround: It is recommended to keep all supplies within specification and/or assert VREG_POR_B if they fall out of spec, or expect the above symptoms to occur if a supply goes below LVD level while the LVD is masked. During functional operation the PMCCR register ideally should not be used to disable LVDs/HVDs.

ERR011306: SAR ADC: Incorrect value of ADC power down exit delay evaluated by the formula given in PDEDR [PDED] field description

Description: The formula in the register field ADC_PDEDR [PDED] provides the delay between the power down bit reset and start of conversion value in number of clock cycles of the ADC module clock, however the given formula of PDED x 1/[ADC_clock_frequency] is incorrect. This gives a calculated value that is too short by 1 cycle of ADC Bus clock and 1 cycle of ADC clock (AD_clk).

Workaround: The correct formula that should be used to calculate the value for the ADC_PDEDR[PDED] register is -

(1/ADC Bus clock) + ((PDED+1) x 1/[ADC_clock_frequency])

Where:

ADC_clock_frequency = Frequency of ADC clock (AD_clk)

ADC Bus clock= Module interface clock for register access (ADC_CLK)

ERR007788: SIUL2: A transfer error is not generated for 8-bit accesses to non-existent MSCRs

Description: An 8-bit access attempt to non-existent MSCRs (Multiplexed Signal Configuration Registers) in the SIUL2 (System Integration Unit Light 2) address space does not generate a transfer error. 16-bit or 32-bit accesses to non-existent MSCRs will generate a transfer error.

Workaround: Do not expect transfer errors on 8-bit accesses to non-existent MSCRs in the SIUL2 address space.

ERR007791: SIUL2: Transfer error not generated if reserved addresses within the range of SIUL BASE + 0x100 to 0x23F are accessed

Description: If any reserved register within the System Integration Unit Lite 2 (SIUL2) register range from SIUL2 BASE + 0x100 to 0x23F is accessed then no transfer error will occur.

Workaround: Software should not be dependent on the indication of a transfer error occurring from an access within the SIUL2 register range from SIUL2 BASE + 0x100 to 0x23F.

ERR010879: SPT: Adaptive scaling using count unoccupied bits from bit 23 down mode can introduce harmonics in the FFT output

Description: When adaptive scaling is enabled using count unoccupied bits from bit 23 downwards saturation can occur in the FFT calculation. This results in harmonics being introduced in the FFT result.

Workaround: There are two possible workarounds:

1. Use adaptive scaling in the mode where unoccupied bits are counted from bit 15 downwards by clearing the ADPTV_SHFT (bit 113) bit in the RDX4 instruction.

2. If adaptive scaling is used counting from bit 23 downwards then the value of the twiddle factors should be divided by SQRT(2). For this workaround to be valid twiddle factor quadrature extension cannot be used and the maximum FFT size is 512 points.

ERR011329: SPT: After completion of a command sequence there is a time window in which work register access by CPU can cause the system to become non-responsive

Description: When the SPT completes execution of a command sequence there is a short time window in which access to an SPT work register by CPU can potentially cause peripheral bus 1 (PBRIDGE_1/AIPS_1) to become non-responsive and the CPU to stop executing instructions because it has stalled waiting on the peripheral bus transaction to complete. Any other core that attempts to access peripheral memory over peripheral bus 1 will also stall and the system can only be recovered by asserting power on reset (POR).

Execution of a STOP command terminates the SPT command sequence execution and the SPT finite state machine (FSM) transitions from RUN state to STOP state for one SPT clock cycle then START state. A ‘soft reset’ signal is generated for one SPT clock cycle at the STOP to START state transition but only once the Command Sequencer DMA (CS DMA) has completed all command fetches. If a work register access is requested by the CPU at the same clock cycle as the ‘soft reset’ signal generation then the SPT will never acknowledge the access request and the peripheral bus will wait forever, making it non-responsive and causing any cores that attempt access to hang. The scenario is also possible if the SPT FSM transitions to STOP state then START state by software assertion of CS_MODE_CTRL[STOP] bit or if the FSM transitions to ASYNCSTOP state then START state by software assertion of CS_MODE_CTRL[ASYNCSTOP] bit.

Practically, the only way this can occur is if the CPU polls for assertion of SPT_CS_STATUS0[PS_STOP] or SPT_CS_STATUS0[PS_ASYNCSTOP] then immediately makes repeated work register accesses so that one of these accesses coincides with SPT 'soft reset' signal assertion. Access to the SPT status registers cannot cause the failure.

The duration of the time window depends on how long the CS DMA is active after execution of the STOP command. Assuming configuration:

• SPT DMA bus master has highest priority on the XBAR.

• Burst size of INCR16 is set for CS DMA.

• There is no SPT acquisition in progress meaning the sample DMA (SDMA) cannot pre-empt the CS DMA between bursts.

• SPT instructions stored in SRAM (if instructions are fetched from flash memory the window duration will always be greater and subject to arbitration delay at the flash controller port.)

• XBAR clock (SYS_CLK) at 133 MHz (MPC577xK/N) or 120 MHz (S32R274/S32R372).

The possible worst-case duration can be calculated:

• 1024-bit transfer (64-bit * 16) takes 18 cycle for INCR16.

• 18 cycle * (1/133 MHz) = 135.3ns. 18 cycle * (1/120 MHz) = 151.2ns.

• SPT CS DMA always fetches 16 instructions, in worst case the STOP command is the first. 128-bit * 16 = 2048-bit.

• Then 2 bursts are needed. At 133 MHz XBAR this is 2 * 135.3ns = 270.6ns. At 120 MHz XBAR this is 2 * 151.2ns = 302.4ns.

Workaround: An interrupt service routine (ISR) should be used to react to SPT command sequence completion because the latency between the interrupt request being raised on assertion of CS_STATUS0[PS_STOP] and execution of the first instruction in the ISR is greater than the worst-case time window duration for assumed configuration. However if CPU polling must be used then measures must be taken before safely accessing the work registers upon completion of the command sequence:

• Check CS_STATUS0[PS_STOP] = 0b1 (This bit indicates that SPT state machine has been in STOP state and so has executed the STOP command.)

• Once CS_STATUS0[PS_STOP] assertion has been detected the user must clear it by writing 0b1.

• Check CS_STATUS3[CS_DMA_ON] = 0b0 (Ensures that all Command Sequencer fetches have been completed.)

Alternatively, enforcing a time delay greater than the worst-case window duration between execution of STOP command and accessing a work register would ensure the time window has elapsed. The delay must begin from the detection of CS_STATUS0[STOP] = 0b1.

ERR010352: SPT: AHB error status can be asserted for the incorrect SPT DMA when performing back to back AHB accesses

Description: When back to back Data Crossbar AHB accesses are performed by different Signal Processing Toolbox (SPT) DMA masters (Command Sequencer DMA (CS-DMA), Programmable DMA (PDMA) or Sample DMA (SDMA)) and there is an AHB error response in the data phase of the last access beat of the first DMA, and the second DMA access has simultaneously entered the address phase then the error response captured is recorded by the SPT for the second DMA access. This causes the incorrect error AHB Error Response status bit to be set in the DMA Error Status Register (SPT_DMA_ERR_STATUS).

Workaround: The memory address associated with a Data Crossbar AHB error response is reported for System Memory Protection Unit (SMPU) protected memory regions and ECC errors. These errors will be reported and logged in the Memory Error Management Unit (MEMU) error reporting table. This address can be compared with the particular application DMA configuration to identify which DMA truly caused the error response (by comparing the error address against the configured DMA operating address ranges) and it's memory access behaviour can be corrected. Accesses made to reserved memory areas will not be logged in the MEMU error reporting tables so the user must ensure that none of the SPT DMAs are configured to attempt an access into these illegal areas.

ERR008223: SPT: FFT instruction with quadrature extension enabled may give wrong parity error indication

Description: When the Fast Fourier Transform (FFT) instruction with quadrature extension enabled is executed, it performs extra read accesses at eight Twiddle RAM (TRAM) memory addresses before the start of the first twiddle address. If a parity error occurs at these memory locations then TRAM controller gives erroneous error indication and address

Workaround: In case of FFT instruction with quadrature extension enabled parity error indication on address immediately before first twiddle address should be ignored.

ERR010085: SPT: FIR operation writes unnecessary extra data into OPRAM when asynchronous PDMA is concurrently accessing the same OPRAM bank

Description: The FIR command will write extra values into the next OPRAM locations (the next OPRAM column address) following the last expected write location when there is an asynchronous PDMA operation concurrently accessing the same OPRAM bank.

Workaround: When using the FIR command with simultaneous asynchronous PDMA, ensure that the commands are operating on different OPRAM banks.

ERR008224: SPT: SDMA_WR_ERR is generated in certain SDMA transfer modes

Description: When using the Sample DMA (SDMA) in TILE4 or TILE8 Modes when the number of ADC channels enabled is 5, 6 or 7 and both the SDMA or the CS-DMA access the Tightly Coupled Memory (TCM) simultaneously, there is a possibility of the SDMA acquisition buffer being overwritten and SDMA_WR_ERR bit being set.

Workaround: When using the TILE4 or TILE8 mode avoid writing both the ADC data and reading code simultaneously from the TCM.

ERR010568: SPT: Spurious CS_AHB_ERR can be generated in the SPT_DMA_ERR_STATUS register

Description: The Signal Processing Toolbox (SPT) command sequence is provided by application software and written to a buffer in the system RAM or flash, where command sequencer DMA (CSDMA) fetches or alternatively transfers the list of instructions to the command queue.

It has been found that the following sequence can occur:

• Whilst the CSDMA is performing a 2-part burst access a high priority high speed crossbar (AHB) master Programmable DMA or Sample DMA (PDMA or SDMA) access occurs.

• The CSDMA will complete its first burst access, then the AHB will service the PDMA/SDMA access. The pending second burst of the CSDMA is stored at this point.

• Once the PDMA/SDMA access completes, the CSDMA will resume its second burst access, but if the SPT code being executed at this time is a NEXT instruction in the same cycle as the second access resumes, then the address fetch pointer can change.

• This means that additional instruction fetches from unwanted locations would occur.

• If the address fetched resides in protected or un-implemented memory then a spurious CS AHB Error (SPT_DMA_ERR_STATUS[CS_AHB_ERR]=0b1) would occur without any illegal opcode error (SPT_CS_STATUS1[ILL_OPCODE]=0b1). If the address fetched resides in normal memory space no error will be generated.

• In no cases (even if valid code is read from the incorrectly updated address) will these instructions be executed by the command sequencer, since the NEXT instruction will invalidate the tags for the remaining instructions. The program flow would continue on as normal.

This sequence of events is considered extremely unlikely to occur.

Workaround: CS AHB Error should only be checked in the SPT_DMA_ERR_STATUS register when an Illegal Opcode indication in SPT_CS_STATUS1 register occurs. Otherwise isolated CS AHB Errors cannot be considered as valid.

ERR008222: SPT: When a parity error occurs the Twiddle RAM (TRAM) or Operand RAM (ORAM) contoller may give erroneous address to MEMU

Description: When a Twiddle Ram (TRAM) or Operand RAM (ORAM) memory location with parity error is read the address sent by the memory controllers to the Memory Error Management Unit (MEMU) may have error in the lower three bits of the address.

Workaround: In case a parity error is indicated to MEMU the lower three bits of the address sent to MEMU should be ignored. As a result of ignoring the lower three bits of address the error may be in any of the eight memory locations indicated by the remaining address bits.

ERR010997: SPT: Work register contents not deterministic after execution of STOP command

Description: When the SPT (Signal Processing Toolbox) executes a STOP command at the end of a command sequence the work register (WR) values are not deterministic until they are re-initialized. Values read from the work registers before they have been re-initialized are not reliable.

Workaround: The CPU can re-initialize the WRs after STOP command has concluded SPT command execution by writing any value to the WRs. The CPU should not rely on read access to the work registers (WR) between a STOP command and the launching of a new command sequence, instead the WAIT and EVT commands should be used to manage the timing of accesses when both SPT and CPU must interact with work registers during command sequence execution.

ERR008910: SSCM: NXEN status in SSCM_STATUS registers is a single bit field

Description: The Nexus Enable 1 status bit (NXEN1) status in System Status and Configuration register (SSCM_STATUS) register is not implemented in this device. There should be a bit showing the status of each of the e200zx cores. Only the status of the first core (e200z4 Core_0) is shown in the Processor 0 Nexus enable status bit (NXEN). Neither of the e200z7 cores (Core_1, Core_2) Nexus enable status bits are supported.

Workaround: If it is required that software knows when a Nexus tool is connected to the device and if the Nexus interface has been enabled on each processor core individually, an alternate method of detection should be used. The Development Trigger Semaphore (DTS) module can be used to signal between the Nexus tool and software which processor cores have the Nexus interface enabled on them.

The DTS Startup register (STARTUP) can be written only by the Nexus tool and its contents can then be read by software. The Nexus tool can be configured to write to user defined bit locations in the STARTUP register to represent the Nexus interface enable status for each processor core individually. The user software can then read the bit locations specific to each core to determine if the Nexus interface has been enabled for the corresponding core. The STARTUP register defaults to all zeros after reset.

ERR007615: SSCM: Accesses to reserved register addresses in modules present in MC_CGM memory space will not generate bus aborts

Description: When register bus aborts are enabled in the register SSCM.SSCM_ERROR[RAE], illegal accesses to reserved register addresses in modules present in MC_CGM memory space (e.g. CMU) will not generate a bus abort. An abort should be generated whenever RAE bit is set.

Workaround: Do not access reserved space or expect a bus abort to occur when doing so while accessing modules in MC_CGM memory space (e.g. CMU).

ERR008749: STCU2: Device may not come out of reset when offline STCU is enabled

Description: The device may not exit reset when Self Test Control Unit (STCU2) is run in offline mode from User Test Flash (UTEST) with the Clock Generation Module System Clock Divider Configuration register (CGM_SC_DCn) settings configured for anything other than a 1:1 divide ratio.

Workaround: If using offline STCU, set all the system clock dividers to divide by 1 by programming the stcu_offline_slftst_sys_divn_div_val (n = 0..5) functions within the dcl_ips_0 and dcl_ips_1 DCF clients to 0b000.

Note: The user must ensure the clock settings meet the requirements of the Reference Manual maximum clock speed settings. For example, the Peripheral Bridge clock (PBRIDGE_n_CLK, also called AIPSX_CLK) can run at a maximum of 66.5 MHz. Due to the errata requiring a 1:1 clock divider setting for system clock dividers, this restricts the maximum speed of STCU BIST to 66.5 MHz for all the System Clock Divider output clocks (CORE0_CLK, SYS_CLK, CORE1_CLK, CORE2_CLK, PBRIDGE_n_CLK, DMA_CLK, and MCAN_CLK).

ERR010683: STCU2: Online LBIST of partition 6 incorrectly generates reset when run with AFE LVD unmasked

Description: During Self-Test Control Unit (STCU2) Online Logic Built In Self-Test (LBIST), if the Analog Front End (AFE) Low Voltage Detects (LVDs) are configured as a source of reset (any of Multipurpose Control Block AFE LVD Mask register bits MCB_AFE_LVD_MASK[AFE_LVD4_MASK], MCB_AFE_LVD_MASK[AFE_LVD3_MASK] or MCB_AFE_LVD_MASK[AFE_LVD2_MASK] set to 1b1), then execution of LBIST on partition 6 (LB6) results in a destructive reset. If MCB_AFE_LVD_MASK[AFE_LVD9_MASK] is set to 1b1 before running online Logic Built In Self-Test (LBIST), then after online self-test the Reset Generation Module Destructive Event Status Register RGM_DES[F_AFE_LVD9] bit is set but no reset occurs.

Workaround: 1) Run Online LBIST with AFE LVDs masked as a source of reset (All of MCB_AFE_LVD_MASK[AFE_LVD4_MASK], MCB_AFE_LVD_MASK[AFE_LVD3_MASK], MCB_AFE_LVD_MASK[AFE_LVD2_MASK] and MCB_AFE_LVD_MASK[AFE_LVD9_MASK] set to 1b0).

OR

2) Do not run Logic Built In Self-Test (LBIST) on partition 6 during online self-test.

ERR010088: STCU2: Unexpected STCU self-test timeout can occur when a short sequence for external reset is triggered during execution of online self-test

Description: While an online self-test is in progress there is a finite window during the self-test execution during which if an external reset is asserted (RESET_B pulled low) and this reset is configured as short sequence for external reset by setting the Short Sequence for External Reset bit in the Reset Generation Module Functional Event Short Sequence Register (RGM_FESS[SS_EXR] = 1b1), or if another functional reset source is triggered during this window, the time after which the part waits for self-test to complete is longer than expected. This time-out value is governed by the watchdog time-out value set in the Watchdog End of Count Timer field in the STCU2 Watchdog Register Granularity register (STCU2_WDG[WDGEOC]). Further, the self-test does not issue a hardware abort (STCU2 Error Register On-line Hardware Abort Flag (STCU2_ERR_STAT [ABORTHW]) will not be set to 1b1) but signals a time-out (STCU2_ERR_STAT [WDTOSW] = 1b1). If the online self-test is being run with PLL enabled then an unexpected PLL unlock event is also observed (STCU2_ERR_STAT[LOCKESW] = 1b1)

Workaround: To avoid the longer than expected duration for self-test completion, allow the online self-test to complete without applying external reset when the external reset is configured as a short sequence for external reset. The other functional resets must also not be triggered during the online self-test execution.

ERR010326: SWT: Out of lockstep fault for instruction and data cache memories is incorrectly indicated when watchdog is configured in fixed address execution service mode

Description: An out of lockstep fault indication for instruction and data cache memories(NCF[10]) will be incorrectly raised if all of the following conditions are true:

• The safety core (e200z420) is configured in lockstep execution mode.

• The safety core Software Watchdog Timer(SWT_0) Service Mode field in SWT Control Register is set to ‘Fixed Address Execution’(SWT_CR[SMD] = 0b10).

• SWT_0 is enabled.

If the above conditions are true, then if an attempt is made to change the value of the Instruction Address Compare 8 (IAC8) register in the safety core and the value of the IAC8 register is utilized by either storing the value of the IAC8 register to memory or by servicing a watchdog time-out event, then NCF[10] fault will incorrectly assert.

Workaround: ‘Fixed Address Execution’ service mode should not be used for SWT_0 when the safety core is configured in lockstep execution mode.

ERR010880: SWT: SWT may not get serviced in Fixed and Incremental address mode

Description: The “Fixed Address Execution” and “Incremental Address Execution" modes may not work properly for servicing the Software Watchdog Timer (SWT). When a core tries to service its corresponding SWT by generating a pulse for the SWT upon execution of the code at the address loaded into the IAC8 register, then the SWT may miss the pulse and therefore not get serviced.

Workaround: Use the “Fixed Service Sequence" or “Keyed Service Sequence" modes to service the SWT.

ERR010619: WGM: Access to look-up table address register or data register when WGM clock is inactive can cause system to become non-responsive

Description: The WGM module clock (WGM_CLK) is derived from the AFE Sigma-Delta PLL (SDPLL) through a divide by 4 divider. Accesses to the WGM LUT Address Register (WGM_LUT_ADDR) and WGM LUT Data Register (WGM_LUT_DATA) that are made when the WGM_CLK is inactive will result in peripheral bridge 1 (PBRIDGE_1) becoming non-responsive and transactions will never complete, causing the system to hang. The only method of recovery is external long functional reset or power-on reset (asserting RESET_B or VREG_POR_B.)

Workaround: Do not attempt to access WGM_LUT_ADDR and WGM_LUT_DATA when the AFE SDPLL is disabled and is not providing the WGM_CLK. If the situation does occur then use an external watchdog circuit to determine when the MCU is not responding and adhere to the Safety Manual guidelines. Asserting the device functional reset signal (RESET_B) or power-on reset signal (VREG_POR_B) when the MCU becomes non-responsive will bring the device back into normal operation.

ERR008238: WGM: WGM peripheral configuration is not controllable by MC_ME_PCTL[238]

Description: Wave Generator Module (WGM) peripheral cannot be enabled in STOP or HALT mode using MC_ME_PCTL[238].

Workaround: There is no workaround to keep the WGM functioning in HALT or STOP mode.

ERR008310: XBIC: Crossbar Integrity Checker may miss recording information from an initial fault event in the case of back-to-back faults

Description: When the Crossbar Integrity Checker (XBIC) detects back-to-back faults on a system bus path through the crossbar switch (AXBS), the fault information captured in the XBIC Error Status Register (XBIC_ESR) and the XBIC Error Address Register (XBIC_EAR) does not correspond to the initial fault event, but rather the subsequent fault event. While the fault event is properly detected, diagnostic status information in the XBIC_ESR and XBIC_EAR registers describing the initial fault event is lost. This defect can only occur in the event of a series of bus transactions targeting the same crossbar slave target, where the series of bus transactions are not separated by idle or stall cycles.

Workaround: Expect that the XBIC_EAR and XBIC_ESR registers may not contain the initial fault information, but will contain the latest fault information.

ERR008730: XBIC: XBIC may store incorrect fault information when a fault occurs

Description: The Crossbar Integrity Checker (XBIC) may incorrectly identify a fault's diagnostic information in the case when the slave response signals encounter an unexpected fault when crossing the crossbar switch (XBAR) during the data phase. While the fault event is detected, the diagnostic status information stored in the XBIC's Error Status Register (XBIC_ESR) and Error Address Register (XBIC_EAR) does not reflect the proper master and slave involved in the fault. Instead, the preceding master or slave ID may be recorded.

Workaround: Expect that when a fault is reported in the XBIC_EAR and XBIC_ESR registers the actual fault information may be from the preceding transition.

ERR010436: ZipWire: SIPI can have only one initiator with one outstanding write frame at time

Description: The Serial Inter-processor Interface (SIPI) module of the Zipwire interface only supports one initiator and one outstanding write frame at a time.

If a new write is initiated (by setting SIPI_CCRn[WRT] = 0b1, where n is the respective channel number for the transmission), or a new streaming write is initiated (by setting SIPI_CCRn[ST] =0b1) with acknowledgement of a previous frame pending, then the initiator node may get a timeout error (indicated by SIPI_ERR[TOEn]=0b1). The previous write frame last byte may also be corrupted at the target node.

This also means that the target node cannot initiate a write transfer while the initiator node is in the process of a write transfer.

Workaround: The initiator should maintain only one outstanding write/streaming write frame to the target node at any one time.

The user must ensure that before initiating a new write request or initiating a new streaming write that it has received an acknowledgement for the previous write transaction (indicated by SIPI_CSRn[ACKR] =0b1). The write acknowledgement interrupt can be enabled by setting SIPI_CIRn[WAIE]=0b1.

Implement a protocol that ensures both sides of the link cannot initiate a transfer at the same time. For example, a token-passing protocol could be implemented using the SIPI trigger command feature. Send a trigger command to pass the token to the other end of the link. Upon receipt of the trigger command, either initiate a write transfer if one is pending, or pass the token back by sending a trigger command. If a write transfer is initiated, wait until ACK is received and then send a trigger command to pass the token back. In this manner, if each side agrees only to initiate a transfer when it obtains the token, there will be no simultaneous transfers that can cause the problem described.