Mask Set Errata for Mask 1N83M

This report applies to mask 1N83M for these products:

- MPC5746R

Mask Specific Information

Major mask revision number	0x1
Minor mask revision number	0x1
JTAG identifier	0x1834_601D

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
e10327	ADC: Incorrect channel under measure is indicated after sampling phase of conversion until conversion completes
e10998	BAF: Uncorrectable ECC error at flash location 0xFAC000 prevents device from booting
e8343	DCI: EVTO[1:0] outputs remain stuck low if asserted while the system clock source is the IRC
e6788	DCI: Watchdog disable function is not available in the Emulation Device (ED) configuration
e7685	DSPI: A STOP mode transition hangs if a DSPI interrupt is issued between the STOP mode acknowledge (by DSPI module) and STOP mode entry
e9783	DSPI: Frame transfer does not restart after DSI frame matches preprogrammed value
e9664	DSPI: Frame transfer does not restart in case of DSI parity error in master mode
e9656	DSPI: Frame transfer does not restart in case of SPI parity error in master mode
e10542	DSPI: Transmit, Command, and Receive FIFO fill flags in status register is not cleared when DMA is improperly configured
e8547	e200z425: Additional cycles required for Load/Store accesses to guarded/precise space
e8036	e200z425: Back to back XBAR transfers are not pipelined
e10491	eDMA: When master ID replication is enabled, the stored ID and privilege level will change if read by another master
e11235	EMIOS: Any UC running in OPWMB or OPWMCB mode may function improperly if the source counter bus is used in another UC in MC or MCB mode
e11293	EMIOS: For any UC operating in OPWFMB mode the Channel Count register should not be written with a value greater than Channel B Data register value
e11295	EMIOS: In OPWFMB mode, A1/B1 registers do not get reloaded with A2/B2 register values if counter value returns 0x1 after counter wrap condition

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
e11294	EMIOS: OPWFMB and MCB mode counter rollover resets the counter to 0x0 instead of 0x1 as mentioned in the specification
e9978	eMIOS: Unexpected channel flag assertion during GPIO to MCB mode transition
e8252	eTPU: ETPU Angle Counter (EAC) Tooth Program Register (TPR) register write may fail
e9090	eTPU: Incorrect eTPU angle counter function under certain conditions
e9809	eTPU: MDU flags(Overflow/Carry) may be set incorrectly
e8229	FCCU: Enabling the programmable glitch filter on EIN may cause a destructive reset
e8042	FCCU: EOUT signals are active, even when error out signaling is disabled
e10900	FCCU: False indication of a fault state for a single safe clock period can be generated on the error output pin
e9581	FCCU: FOSU may assert destructive reset when a hardware recoverable fault of width less than one safe clock period occurs
e7844	FCCU: NCF status flag is not cleared after reset request from SWT_0 or SWT_3
e11256	FCCU: Redundancy control checker (RCC) reports false error in FCCU_SCFS register
e10032	FEC: MDC and MDIO signals are not on the FEC power supply segment
e2340	FEC: slot time is designed for 516 bit times; deviation from the 802.3
e8004	FLASH: Array Integrity with Breakpoints enabled may skip addresses for certain RWSC and APC combinations
e7991	FLASH: Rapid Program or Erase Suspend fail status
e8341	FlexCAN: Entering Freeze Mode or Low Power Mode from Normal Mode can cause the FlexCAN module to stop operating.
e8759	FlexCAN: FD frame format not compliant to the new ISO/CD 11898-1: 2014-12-11
e7845	FlexCAN: Limit Flexible Data rate (FD) baud rate to 6.67 Mbps
e7433	JTAGM: Nexus error bit is cleared by successful RWA
e8935	JTAGM: write accesses to registers must be 32-bit wide
e7274	LINFlexD: Consecutive headers received by LIN Slave triggers the LIN FSM to an unexpected state
e6428	LINFlexD: Data reception could terminate abruptly in LIN Slave mode when Time-out counter mode is enabled
e6425	LINFlexD: FIFO buffer can not be accessed without changing the RFC counter
e8933	LINFlexD: Inconsistent sync field may cause an incorrect baud rate and the Sync Field Error Flag may not be set
e8970	LINFlexD: Spurious bit error in extended frame mode may cause an incorrect Idle State
e7589	LINFlexD: Spurious timeout error when switching from UART to LIN mode or when resetting LINTCSR[MODE] bit in LIN mode
e6350	LINFlexD: WLS feature cannot be used in buffered mode
e10550	MC_CGM: Auxiliary clock dividers get stuck if programmed to divide by 2 and a reset occurs during operation
e10505	MC_CGM: The device can become stuck in reset after 'destructive' or external long 'functional' reset
e10506	MC_CGM: The device can become stuck in reset when a reset occurs during a system clock switch to any PLL
e7211	MC_ME: Core register IAC8 is cleared during a mode change when the core is reset
e6785	MEMU: ECC errors on FlexCAN RAM accesses indicate an incorrect error address in the peripheral RAM reporting table

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
e8539	MEMU: Software Reset bit field is not protected by the Register Protection module
e9631	MEMU: System RAM ECC errors on accesses by some masters report to MEMU Peripheral RAM table
e3872	NAR: Emulation device may transmit random trace data during initialization
e3970	NAR: Trace messages include a 6-bit Source Identification field instead of 4-bits
e11021	NAR: Trace to memory buffer pointers corrupted after break-point
e7850	NXMC: XBAR data trace may overrun internal NXMC Message Buffers
e10340	NZxC3: ICNT and HIST fields of a Nexus message are not properly reset following a device reset
e7876	PASS: Debug interface state is not preserved through reset on secured devices after a functional reset
e10552	PASS: JTAG password match bypasses flash read protection set by PASS.LOCK3[RLx] bits
e9250	PASS: JTAG password not working during reset
e7904	PASS: Programming Group Lock bit (PGL) can be de-asserted by multiple masters writing the correct password sections to the CINn registers.
e7905	PIT: Accessing the PIT by peripheral interface may fail immediately after enabling the PIT peripheral clock
e9496	PMC: FCCU NCF[4] fault may be triggered when DCF_PMC_REE_CTRL or DCF_PMC_RES_CTRL are programmed
e8555	PMC: Over and under temperature flags may be set during reset
e6775	SAR_ADC: Simultaneous calibration of SAR ADCs is not recommended
e8039	SDADC: digital filter and FIFO not disabled when MCR[EN] is cleared
e8225	SDADC: FIFO Flush Reset command requires clearing the Data FIFO Full Flag
e8631	SDADC: low threshold watchdog cannot be used with signed data
e7204	SENT: Number of Expected Edges Error status flag spuriously set when operating with Option 1 of the Successive Calibration Check method
e8082	SENT: A message overflow can lead to a loss of frames combined with NUM_EDGES_ERR being set
e7425	SENT: Unexpected NUM_EDGES_ERR error in certain conditions when message has a pause pulse
e7788	SIUL2: A transfer error is not generated for 8-bit accesses to non-existent MSCRs
e10489	SIUL2: Open-drain or open-source configured outputs may briefly drive high or low during transitions
e5570	SIUL2: Read operation of the SIUL2 MSCR bits that are permanently set (enabled) will return 1b0
e7871	SPC5746R:Current injection causes leakage path across the DSPI and LFAST LVDS pins
e9658	SPI: Inconsistent loading of shift register data into the receive FIFO following an overflow event
e7929	SSCM: FLASH ECC errors are reported after the SSCM execution
e10088	STCU2: Unexpected STCU self-test timeout can occur when a short sequence for external reset is triggered during execution of online self-test
e9493	STCU: FCCU NCF 6 does not automatically generate a destructive reset, when DCF_UTEST_Miscellaneous.STCU_CFB bit field is set (1)
e10507	STCU: The device can become stuck in reset when destructive reset occurs during offline self-test when PLL is selected as BIST clock source
e6851	SWT: Software Timer interrupt may not cause device to exit STOP mode under certain clock configurations
e7948	TDM: Erase protection not enabled by reset

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
e8310	XBIC: Crossbar Integrity Checker may miss recording information from an initial fault event in the case of back-to-back faults
e8730	XBIC: XBIC may store incorrect fault information when a fault occurs
e4136	XOSC and IRCOSC: Bus access errors are generated in only half of non-implemented address space of XOSC and IRCOSC, and the other half of address space is mirrored
e7947	XOSC: Incorrect external oscillator status flag after CMU event clear
e10436	ZipWire: SIPI can have only one initiator with one outstanding write frame at time

Table 2. Revision History

Revision	Changes
1.0	Initial revision
2.0 May 2015	The following errata were added. • e7274
2.1 July 2015	The following errata were removed. • e8145 The following errata were added. • e7947 • e9250
2.2 August 2015	The following errata were added. • e9581 • e9631 • e9496 • e9493 The following errata were revised. • e6785 • e8555
2.3 October 2015	The following errata were removed. • e8019 The following errata were added. • e8229
2.4 February 2016	The following errata were added. • e9658 • e9656 • e9664 • e9978 • e9809 • e10032 • e9783

Table continues on the next page...

Table 2. Revision History (continued)

Revision	Changes
	The following errata were revised. • e8933
2.5 September 2016	The following errata were added. • e10340 • e10489 • e10491 • e10505 • e10506 • e10327 • e10507 • e10552 • e10550 • e7948 • e10542 • e10088
2.6 August 2017	The following errata were removed. • e7356 The following errata were added. • e2340 • e10900 • e10436 • e10998 The following errata were revised. • e9581
2.7 May 2018	Revision 2.7 May 2018 The following errata were added. • e11293 • e11295 • e11294 • e11235 • e11021 • e11256 The following errata were revised. • e10436 • e10900

e10327: 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).

e10998: BAF: Uncorrectable ECC error at flash location 0xFAC000 prevents device from booting

Description: Boot Assist Flash (BAF) firmware locates a valid boot header by searching 9 flash locations in a specific, sequential order. If a valid boot header is not present in locations 1–4 and an uncorrectable ECC error is present at the 5th location (0xFAC000), the device will be reset and will not be able to boot. Valid boot headers at locations 6–9 will be ignored. If an uncorrectable ECC error is not present at 0xFAC000, any of the boot header locations can be used and the BAF will search through the locations sequentially until a valid boot header is found.

Workaround: 1. Ensure that a valid boot header is placed at one of the first 4 boot header locations: before location 5 (0xFAC000).

2. If the valid boot header must be placed after 0xFAC000, do not program the 32 byte page beginning at 0xFAC000. Leave this flash area in the factory state (blank/unprogrammed).

3. If the 32 byte page starting at 0xFAC000 must be programmed, confirm that there are no existing ECC errors after programming and mark location 0xFAC000 as OTP. This will avoid uncorrectable ECC errors at this location during future erase or programming activities.

e8343: DCI: EVTO[1:0] outputs remain stuck low if asserted while the system clock source is the IRC

Description: While the system clock source is the Internal Resistor Capacitor oscillator (IRC), if the Event Output pins (EVTO[1:0]) are asserted by the Debug and Calibration Interface (DCI)), they will never negate. This applies to all functions on EVTO[1:0] except the timer functions controlled by the EVTO Output selection (EOS0/EOS1) fields of the Generic Timer Module debug interface (GTMDI) Development Control (DC) register. This includes when the EVTO pins are being used by the Development Trigger Semaphore (DTS) module.

Workaround: For proper operation of EVTO[1:0] outputs, program clocks to select a system clock source other than the IRC before enabling and using EVTO[1:0].

e6788: DCI: Watchdog disable function is not available in the Emulation Device (ED) configuration

Description: The Debug and Calibration Interface (DCI) watchdog disable function is available only on the production device configuration. The feature to disable the Software Watchdog Timer (SWT) by asserting EVTI0 is not available on the emulation device.

Workaround: Do not expect the DCI watchdog disable function to operate on the emulation device. Tools should disable the software watchdog timers by clearing the Software Watchdog Timer Control Register Watchdog Enable field (SWT_CR[WEN]) if required.

e7685: DSPI: A STOP mode transition hangs if a DSPI interrupt is issued between the STOP mode acknowledge (by DSPI module) and STOP mode entry

Description: The following sequence of events can cause the application to hang when a run mode change to STOP mode is initiated using the Mode Entry (MC_ME) module and the De-serial Serial Peripheral Interface module (DSPI) is running.

1. The application requests a transition to STOP by writing the Mode Entry Mode Control Register (MC_ME_MCTL) TARGET_MODE bit field with the value 0b1010 along with the KEY field (0xA000_A50F), and then writing MC_ME_MCTL again with the TARGET_MODE and inverted KEY field (0xA000_5AF0).

2. The MC_ME issues a STOP request to all running modules and they acknowledge this request when they are ready to go to STOP.

3. The DSPI generates an interrupt immediately before sending the STOP acknowledge to MC_ME module.

4. The e200z core is still active because the MC_ME is still waiting for all running modules to acknowledge the STOP request, and upon receiving the DSPI interrupt it initiates the service routine.

5. In the service routine, the core attempts to clear the DSPI interrupt flag by writing to the DSPI Status Register (DSPI_SR) but the write does not complete, because the DSPI module has issued the STOP acknowledge and is no longer running.

Since the interrupt from the DSPI module cannot be cleared, the mode transition to STOP mode hangs.

Workaround: Use one of the two possible workarounds:

1. The application software should ensure that the DSPI module is idle and no DSPI interrupt can be asserted before the MC_ME starts a STOP mode transition.

2. Clock gate the DSPI module before starting a STOP mode transition by clearing the Divider Enable (DE) bit in the Clock Generation Module Aux Clock x Divider Configuration y (MC_CGM_ACx_DCy) registers that configure the DSPI clocks.

e9783: DSPI: Frame transfer does not restart after DSI frame matches preprogrammed value

Description: In the Deserial Serial Peripheral Interface module, in the scenario when:

1. Master/slave mode select bit of module configuration register is set (MCR[MSTR]=0b1) to configure the module in master mode

2. Deserial Serial Interface (DSI) communication is selected via DSPI Configuration field (DCONF) in the Module Configuration Register (MCR [DCONF] = 0b01)

3. Preprogrammed value for data match with received DSI frame is configured using DSI De-serialized Data Polarity Interrupt Register (DPIR) and DSI De-serialized Data Interrupt Mask Register (DIMR)

4. Data Match Stop (DMS) bit of DSI configuration register0 is set (DSICR0 [DMS] =0b1) which stops DSI frame transfer in case of a data match with a preprogrammed value

5. DSI frame is received with bits matching preprogrammed value.

Under these conditions, the next frame transfer is stopped, DSI Data Received with Active Bits bit of status register is set (SR [DDIF] =0b1) and the corresponding DDIF interrupt is asserted. Even after the interrupt is serviced and SR [DDIF] is reset, the frame transfer does not restart.

Workaround: DSI frame transfer stop in case of DSI data match condition should be disabled. For this, keep the data match stop bit of DSI configuration register 0 de-asserted (DSICR0 [DMS]=0b0).

e9664: DSPI: Frame transfer does not restart in case of DSI parity error in master mode

Description: In the Serial Peripheral Interface module, in the scenario when:

1. Master/slave mode select bit of module configuration register is set (MCR[MSTR]=0b1) to configure the module in master mode

2. Deserial Serial Interface (DSI) communication is selected via DSPI Configuration field (DCONF) in MCR (MCR[DCONF] = 0b01)

3. Parity reception check on received DSI frame is enabled by setting Parity Enable bit (PE) of DSI configuration register 0 (DSICR0[PE]=0b1)

4. Parity Error Stop (PES) bit of DSI configuration register0 is set (DSICR0[PES]=0b1) which stops DSI frame transfer in case of parity error

5. Parity error is detected on received frame

Then the next frame transfer is stopped, DSI parity error flag bit of status register is set (SR[DPEF] =0b1) and the corresponding DSI parity error interrupt is asserted. Even after the interrupt is serviced and SR [DPEF] is reset, the frame transfer does not restart.

Workaround: DSI frame transfer stop in case of parity error detection should be disabled. For this, keep the parity error stop bit of DSI configuration register0 de-asserted (DSICR0 [PES]=0b0).

e9656: DSPI: Frame transfer does not restart in case of SPI parity error in master mode

Description: In the Deserial Serial Peripheral Interface (DSPI) module, in the scenario when:

1. Master/slave mode select bit (MTSR) of Module Configuration register (MCR) is set (MCR[MSTR]=0b1) to configure the module in master mode

2. SPI communication is selected via DSPI Configuration field (DCONF) in MCR (MCR[DCONF] = 0b00)

3. Parity reception check on received frame is enabled by setting the Parity Enable or Mask tASC delay (PE_MASC) bit of DSPI PUSH FIFO Register In Master Mode (PUSHR), i.e. PUSHR[PE]=0b1.

4. Parity Error Stop bit (PES) of MCR is set (MCR[PES]=0b1) which stops SPI frame transfer in case of parity error

5. Parity error is detected on received frame.

Then the next frame transfer is stopped, the SPI Parity Error Flag bit (SPEF) of the DSPI Status Register (DSPI_SR) is set (SR[SPEF] =0b1) and the corresponding SPI parity error interrupt is asserted. Even after the interrupt is serviced and SR[SPEF] is reset, the frame transfer does not restart.

Workaround: Do not use SPI frame transfer stop in case of parity error detection for SPI transmission in master mode. For this, keep the Parity Error Stop bit of Module Configuration Register de-asserted (MCR[PES] = 0b0).

e10542: DSPI: Transmit, Command, and Receive FIFO fill flags in status register is not cleared when DMA is improperly configured

Description: The Deserial/Serial Peripheral Interface Transmit, Receive, and Command First In/First Out (FIFO) buffers can request additional information to be transferred via the Direct Memory Access (DMA) module when either the Transmit, Receive, or Command FIFO Fill/Drain Flags are set in the DSPI Status Register (SR[TFFF/RFDF/CMDFFF]). However, the Command/ Transmit Fill Flag only indicates that at least 1 location in the FIFO is available to be written. It does not indicate that the FIFO is empty. Similarly, Receive FIFO fill flag only indicates at least 1 location of the FIFO is available to be read. It does not indicate that the FIFO is full. If the DMA is configured to transfer more than 1 FIFO location size of data, the FIFO Fill Flags may not be properly cleared indicating that the FIFO is not full even when the FIFO is actually full (for Transmit and Command FIFO) and not empty when the FIFO is actually empty (for Receive FIFO).

Workaround: Properly configure the DMA to fill the Transmit, Receive, and Command FIFOs only one FIFO location, in other words, up to 2 bytes, at a time to each of the FIFOs.

Use the DMA loop to transfer more data if needed.

e8547: e200z425: Additional cycles required for Load/Store accesses to guarded/ precise space

Description: If an access by the Load/Store unit is to a space that is guarded/precise then it will be subject to additional cycles for completion because there is no pipeline bypass for the Bus Interface Unit. These spaces and accesses are characterized by any one of the following conditions:

- The Memory Protection Unit (MPU) Guarded bit is set for the address space
- The access is a store with the Store Buffer disabled
- The access is a Decorated Load or Store
- The access is a Bypass Store
- The access is an Atomic Load or Store

On this device, the peripheral address space, by default, is set to be guarded/precise space.

Each scenario below will cause a 2 cycle increase:

- Idle to guarded/precise load
- Idle to guarded/precise store
- Store to guarded/precise load
- Store to guarded/precise store

The following scenarios may or may not add extra cycles depending on wait states and back to back requests:

- Load to guarded/precise store
- Load to guarded/precise load

Workaround: Expect additional cycles when guarded/precise accesses occur.

e8036: e200z425: Back to back XBAR transfers are not pipelined

Description: Back to back e200z425* (with no data cache) load requests are not pipelined through the Cross Bar (XBAR) bus if the first access has a wait state.

Workaround: Expect that the core performance could be slightly reduced from ideal when accessing data back to back through the Crossbar bus. Back to back loads that are performed in sequential instructions (with no instructions between the accesses). Critical, time-sensitive variables that are frequently accessed sequentially should be stored in local Data memory (DMEM).

e10491: 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 feature of a DMA channel is enabled via the Channel n Master ID Register (DMA_DCHMIDn) by setting the Enable Master ID replication (EMI) bit (DMA_DCHMIDn[EMI]=1), the DMA_DCHMIDn[PAL] and DMA_DCHMIDn[MID] fields should reflect the privileged access level (PAL) and master ID (MID) respectively of the master that wrote the Transfer Control Descriptor (TCD) Control and Status register (DMA_TCDn_WORD_7) least significant byte (DMA_TCDn_WORD_7[DONE,ACTIVE, MAJOR_E_LINK, E_SG, DREQ, INT_HALF, INT_MAJOR, START] byte). However, if a different master reads the DMA_TCDn_WORD_7 least significant byte, the MID and PAL of DMA_DCHMIDn will incorrectly change to this read access master's MID and PAL.

Workaround: Only allow the intended master ID replication core to access the DMA_TCDn_WORD_7 least significant byte (including accessing the full TCD word).

e11235: EMIOS: Any UC running in OPWMB or OPWMCB mode may function improperly if the source counter bus is used in another UC in MC or MCB mode

Description: If a user configures any Unified Channel (UC) in Modulus Counter (MC) or Modulus Counter Buffered (MCB) mode by setting the eMIOS Channel Control Register MODE bitfield to 7'h10 or 7'h11 and if pre-scalers are enabled to increment the counter (GPREN bit of the Mode Control Register = 1'b1 or UCPREN bit of the Channel Control Register = 1'b1), then the UC does not trigger the counter bus reload event. The counter bus reload event is propagated as "counter bus sync" at UC output. If this particular UC is driving local or global buses of EMIOS then the bus_sync_signal is affected. This will manifest at least on any UC channel set in Center Aligned Output Pulse Width Modulation Buffered (OPWMCB) or Output Pulse Width Modulation Buffered (OPWMB) modes, and driven using the local or global bus by faulty UC channel.

Workaround: If any local or global bus in EMIOS is driven by Unified channels and the mode configuration of such unified channel control register is set to MC or MCB mode (Channel Control Register MODE bitfield is 7'h10 or 7'h11) and if the global prescaler is enabled in the module configuration register (GPREN bit of the Mode Control Register = 1'b1 or UCPREN bit of the Channel Control Register = 1'b1), then that bus should not be used by another UC with mode configuration register using OPWMCB or OPWMB mode (Channel Control Register MODE bitfield set to 7'b11000X0 or 7'b10111XX (X = don't care)).

e11293: EMIOS: For any UC operating in OPWFMB mode the Channel Count register should not be written with a value greater than Channel B Data register value

Description: For any Unified Channel (UC) running in Output Pulse-Width and Frequency Modulation Buffered (OPWFMB) mode, Channel Control Register MODE bitfield = 7'h1011000 or 7'h1011010, the internal counter runs from 0x1 to Channel B Data register value.

The internal counter can be overwritten by software using the Chanel Count register during 'freeze' operation.

If a counter wrap occurs due to overwriting of the counter with a value greater than its expiry value (B Data Register value); than the output signal behavior cannot be guaranteed.

Workaround: For any UC operating in OPWFMB mode the Channel Count register should not be written with a value greater than Channel B Data register value.

e11295: EMIOS: In OPWFMB mode, A1/B1 registers do not get reloaded with A2/B2 register values if counter value returns 0x1 after counter wrap condition

Description: In Output Pulse-Width and Frequency Modulation Buffered (OPWFMB) mode, A1/B1 registers do not get reloaded with A2/B2 register values if counter value returns 0x1 after counter wrap condition.

In order to avoid the counter wrap condition make sure internal counter value is within the 0x1 to B1 register value range when the OPWFMB mode is entered. Also overwriting of Channel Count register by forcing 'freeze' in OPWFMB mode should not take internal counter outside 0x1 to B register value.

Workaround: In order to avoid the counter wrap condition:

1. Make sure internal counter value is within the 0x1 to (B1 register) value range when the OPWFMB mode is entered.

2. Overwrite of Channel Count register by forcing 'freeze' in OPWFMB mode should not be outside the range of 0x1 to (B register) value.

e11294: EMIOS: OPWFMB and MCB mode counter rollover resets the counter to 0x0 instead of 0x1 as mentioned in the specification

Description: When the enhanced Modular Input/Output System (eMIOS) is used in Output Pulse-Width and Frequency Modulation Buffered (OPWFMB) or Modulus Counter Buffered (MCB) modes, when the counter rolls over, the counter returns to 0x0 instead of 0x1 as specified in the reference manual.

Workaround: In order to avoid the counter wrap condition:

1. Make sure internal counter value is within the 0x1 to (B1 register) value range when the OPWFMB mode is entered.

2. Overwrite of Channel Count register by forcing 'freeze' in OPWFMB mode should not be outside the range of 0x1 to (B register) value.

e9978: eMIOS: Unexpected channel flag assertion during GPIO to MCB mode transition

Description: When changing an Enhanced Modular IO Subsystem (eMIOS) channel mode from General Purpose Input/Output (GPIO) to Modulus Counter Buffered (MCB) mode, the channel flag in the eMIOS Channel Status register (eMIOS_Sn[FLAG]) may incorrectly be asserted. This will cause an unexpected interrupt or DMA request if enabled for that channel.

Workaround: In order to change the channel mode from GPIO to MCB without causing an unexpected interrupt or DMA request, perform the following steps:

(1) Clear the FLAG enable bit in the eMIOS Control register (eMIOS_Cn[FEN] = 0).

(2) Change the channel mode (eMIOS_Cn[MODE]) to the desired MCB mode.

(3) Clear the channel FLAG bit by writing ‘1’ to the eMIOS Channel Status register FLAG field (eMIOS_Sn[FLAG] = 1).

(4) Set the FLAG enable bit (eMIOS_Cn[FEN] = 1) to re-enable the channel interrupt or DMA request reaction.

e8252: eTPU: ETPU Angle Counter (EAC) Tooth Program Register (TPR) register write may fail

Description: When the TPR is written with the Insert Physical Tooth (IPH) bit set to 1, and a physical tooth arrives at near the same time, the buffering of a second write to the TPR may fail, even if the required wait for one microcycle after the IPH write is observed.

Workaround: Wait at least two microcycles between consecutive writes to the TPR register, if the first write sets the IPH bit.

e9090: eTPU: Incorrect eTPU angle counter function under certain conditions

Description: The eTPU Angle Counter (EAC) can function incorrectly in some scenarios when all of the following conditions apply:

- EAC Tooth Program Register (TPR), Angle Ticks Number in the Current Tooth field (TICKS) = 0 [TPR.TICKS = 0]

and

- Tick Rate Register (TRR) and the eTPU Engine Time Base Configuration Register prescaler field [eTPU_TBR_TBCR_ENGn.TCRnP] satisfy the following condition:

(TRR – 1)*(TCRnP + 1) < 3, where TRR is the non-zero 15-bit integer part (the 15 most significant bits).

When the above conditions are met, three possible scenarios can cause the EAC to function incorrectly:

Scenario 1:

1. The EAC is in High Rate Mode, TRR = 1, and TPR Missing Tooth Counter field = 0 [TPR.MISSCNT = 0]

2. On an EAC transition from High Rate Mode to Normal mode, a positive value is written to TPR.MISSCNT

3. The first microcycle in Normal Mode coincides with a tick timing and either

a. A tooth does not arrive

or

b. A tooth arrives

Expected EAC behavior:

a. Nothing happens

or

b. The EAC transitions back to High Rate Mode

Actual (incorrect) EAC behavior:

a. The EAC transitions to Halt Mode, even though TPR.MISSCNT > 0

or

b. The EAC stays in Normal Mode, even though a tooth arrived before expected and TPR.MISSCNT > 0. The values of TPR.MISSCNT and TPR.LAST are reset, even though the EAC does not transition to High Rate Mode.

Scenario 2:

TCRnP = 0, TRR = 1 (integer part) and a new value is written to TPR.MISSCNT when the EAC transitions from High Rate Mode to Normal Mode. In this scenario, TPR.MISSCNT decrements on every microcycle, but the time the EAC takes to transition to Halt Mode is determined by the previous

TPR.MISSCNT value, so that one of the following unique situations is observed:

a. TPR.MISSCNT reaches zero, but the EAC transitions to Halt Mode only after a number of microcycles equal to the TPR.MISSCNT value before the write.

b. EAC transitions to Halt Mode with TPR.MISSCNT > 0 while, decrementing MISSCNT one more time. If TPR.MISSCNT > 1 during the mode transition, the EAC will stay in Halt mode with a non-zero value of TPR.MISSCNT.

Scenario 3:

1. The EAC transitions to Normal mode from High Rate or Halt Mode

2. The EAC enters Normal mode with TPR.LAST = 1

3. A tooth is received on the second or third microcycle after the EAC transitions to Normal mode. The tooth may be either a physical tooth or a dummy physical tooth generated by setting the Insert Physical Tooth (IPH) field of the TPR register (TPR.IPH = 1).

Observed result:

The EAC resets the values of TPR.LAST, TPR.IPH and the eTPU Engine Time Base2 (TCR2) register, but the EAC goes to Halt mode.

If a new TPR.TICKS value is written with the EAC in Normal mode, the value is effective after a new tooth is received in Halt mode, with TCR2 counting from 0.

Workaround: Limit the angle tick period to a minimum value that satisfies the condition (TRR – 1)* (TCRnP + 1) > 2, where TRR is the non-zero 15-bit integer part (the 15 most significant bits).

e9809: eTPU: MDU flags(Overflow/Carry) may be set incorrectly

Description: The MAC Carry (MC) & MAC Overflow (MV) flags can be incorrectly set on a MAC instruction if it is the first MDU operation in a thread and the last MDU operation in previous thread was aborted/terminated (thread ended before the operation finished).

Workaround: There are 2 workarounds:

(1) Do not abort/terminate a MDU operation

or

(2) Do not use a MAC instruction as the first MDU operation in a thread

e8229: FCCU: Enabling the programmable glitch filter on EIN may cause a destructive reset

Description: The Fault Collection and Control Unit (FCCU) external error input (EIN) can be filtered by the FCCU on-chip programmable glitch filter. However, the EIN is routed directly to the FCCU Output Supervision Unit (FOSU) without passing through the glitch filter. Additionally, when the glitch filter is programmed using the FCCU Control Register (FCCU_CTRL) FILTER_WIDTH field, the effective duration may vary depending on time at which the EIN signal arrives, which can cause a missed or false EIN signal. As a result, it is possible for the FOSU to recognize an event on EIN that is ignored by the FCCU resulting in a destructive reset when the FOSU timeout period expires.

Workaround: Bypass the FCCU on-chip glitch filter by writing a 1 to the FCCU Control Register FILTER_BYPASS field (FCCU_CTRL[FILTER_BYPASS]) and use an external glitch filter to ensure that only valid external error events are recognized by the FCCU. The impact on safety can be avoided by using a feedback based signaling on Error In (EIN), wherein the EIN signal is kept asserted until the MCU acknowledges the event to the source.

e8042: FCCU: EOUT signals are active, even when error out signaling is disabled

Description: Every time the Fault Collection and Control Unit (FCCU) moves into fault state caused by an input fault for which the error out reaction is disabled (FCCU_EOUT_SIG_ENn[EOUTENx]=0), the Error Out 1 and 2 (EOUT[0] and EOUT[1]) will become active for a duration of 250 us plus the value programmed into the FCCU Delta Time register (FCCU_DELTA_T[DELTA_T]). EOUT is not affected if the FCCU moves into the alarm state that generates an interrupt (IRQ), if the Fault is cleared before the alarm timeout.

This erratum does not affect the outputs of other pins (for example, for communication modules like CAN/Flexray). Only the EOUT signal is impacted.

Workaround: There are three possible workarounds:

1) Enable EOUT signaling for all enabled error sources.

2) In case external device (which evaluates EOUT) can communicate with the MCU, the following procedure could be used:

a) Program any duration of EOUT as per application needs (FCCU_DELTA_T[DELTA_T])

b) For faults requiring error out reaction, the software shall validate EOUT via separate communication channel (like I2C) while EOUT is asserted.

c) External device shall implement a timeout mechanism to monitor EOUT validation by separate channel.

d) Following scenarios shall be considered as valid EOUT reactions:

d1) Validation is performed while EOUT is asserted

d2) Timeout occurs but no validation and EOUT is still asserted.

3) In case external device (which evaluates EOUT) cannot communicate with the MCU, following procedure could be used:

a) Program the error out duration to a duration x (FCCU_DELTA_T[DELTA_T]).

b) For faults requiring error out reaction, clear the fault after the pin has continued to be asserted for a longer duration (for example 2*duration x). This will artificially create a long pulse on EOUT.

c) For faults which do not require error out reaction, clear the fault within duration x. This will artificially create a short pulse on EOUT.

d) External device should ignore short pulse of duration x while recognizing longer pulses as valid reaction.

e) While clearing the fault, the associated software shall check the pending faults.

e10900: 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.

e9581: FCCU: FOSU may assert destructive reset when a hardware recoverable fault of width less than one safe clock period occurs

Description: The Fault Collection and Control Unit Output Supervision Unit (FOSU) may issue a destructive reset if all of the following conditions are present:

• An input fault is programmed as hardware recoverable in a FCCU Non-Critical Fault Configuration Register (FCCU_NCF_CFGn)

• The only reaction programmed for this fault is FCCU Error Output signaling (FCCU_EOUT_SIG_ENn)

• The source of the fault signal is asserted for less than one safe clock period. The safe clock for this device is the internal RC oscillator (IRC).

Workaround: Always configure faults as software recoverable in the FCCU_NCF_CFGn. Set the Non-critical Fault Configuration bit to a ‘1’, this is the default condition for implemented faults after reset.

e7844: FCCU: NCF status flag is not cleared after reset request from SWT_0 or SWT_3

Description: The fault status flag (NCF) in the Fault Control and Collection Unit (FCCU) will not be automatically cleared after a reset request from either of the Software Watchdog Timers (SWT_0 or SWT_3).

Workaround: After a SWT_0 or SWT_3 request to the FCCU and a subsequent reset, software should clear the corresponding NCF status flags (NCF index 21 for SWT_0 or NCF index 17 for SWT_3) in the FCCU_NCF_Sx register.

e11256: FCCU: Redundancy control checker (RCC) reports false error in FCCU_SCFS register

Description: The Fault collection and control unit (FCCU) has two self-checking sub-modules which compare FCCU internal signals. This feature can be enabled by setting the FCCU configuration register Redundancy Control Checker (RCC) bits (FCCU_CFG[RCCE0] and FCCU_CFG[RCCE1]) bits. The State Change Freeze Status register (FCCU_SCFS) contains the status of RCC checkers.

If the error output pin (EOUT) is configured in Bi-Stable mode and the user application software enables the RCC0 and RCC1 cross check signals with the default configuration, the FCCU_SCFS register reports an error.

Workaround: If the RCC self-checking feature is enabled via software, bit 20 of the FCCU_CFG register (FCCU_CFG[20]) must be set.

FCCU_CFG[20] is a configuration mode select (FCCU_CFG[CM]). When CM= 1’b1, during the CONFIG state, the EOUT pins functions the same as in the NORMAL state. Setting CM does not cause any other change in EOUT.

e10032: FEC: MDC and MDIO signals are not on the FEC power supply segment

Description: The Fast Ethernet Controller (FEC) Management Data Clock (MDC) and Management Data Input Output (MDIO) pins are not included in the isolated FEC I/O supply segment (VDD_HV_IO_FEC). They are powered by the main I/O supply segment (VDD_HV_IO_MAIN) . Therefore if VDD_HV_IO_MAIN is used for 5V I/O, then the MDC/MDIO pins will be over-driven for a 3.3V physical interface (PHY). Since the MCU pad output levels scale with supply voltage, the maximum output low voltage (VOL) for MDC/MDIO may violate the maximum input low voltage (VIL) for a 3.3V PHY. When MDIO is driven from a 3.3V PHY, the minimum output high voltage (VOH) may not meet the minimum input high voltage (VIH) of the MCU.

Workaround: In order to interface 5V MDC/MDIO signals to a 3.3V PHY, level shifters or series resistance should be used on the PCB. In the case of series resistance, Transistor-transistor Logic (TTL) input levels should be used with the MDIO input.

Depending on the application, a 3.3V PHY may be able to tolerate the over-voltage for limited time over the lifespan if connected directly to the 5V MCU pins. The PHY electrical specification should be consulted for absolute maximum ratings.

e2340: FEC: slot time is designed for 516 bit times; deviation from the 802.3

Description: The Fast Ethernet Controller (FEC) slot time is 516 bit times which is longer than the 512 bit times specified by the IEEE 802.3 standard.

If a collision occurs after the standard 512 bit times (but prior to 516 bit times), the FEC may generate a retry that a remote ethernet device may identify as late. In addition, the slot time is used as an input to the backoff timer, therefore the FEC retry timing could be longer than expected.

Workaround: No software workaround is needed or available.

e8004: FLASH: Array Integrity with Breakpoints enabled may skip addresses for certain RWSC and APC combinations

Description: For certain combinations of the Flash Read Wait State Control (RWSC) and Address Pipeline Control (APC) settings in the Platform Flash Configuration Register (PFLASH_PCFR1) the Flash's array integrity (AI) check when run with breakpoints enabled may skip addresses resulting in an incorrect Multiple Input Signature Register (MISR) value or in the case of back to back ECC event errors (EER) or Single Bit Correction (SBC) events, a skipped breakpoint. This occurs for the following combinations:

RWSC=1 and APC=1

RWSC=3 and APC=2

RWSC=5 and APC=3

If breakpoints are enabled and an EER or SBC cause a breakpoint to occur the address after the breakpoint will be skipped, and the resulting MISR will not match expectations. Likewise, if there are back to back errors (EER or SBC) during AI with the above RWSC/APC combinations the 2nd error (and breakpoint) will be missed.

Margin Read (which by specification is a self timed event and is independent of wait states selected) is not affected by this erratum. This erratum only applies to Array Integrity.

Workaround: One workaround is to follow the recommended RWSC and APC combinations for given frequencies. If this is done, Array Integrity with Breakpoints feature works as expected. Valid RWSC/APC combinations listed in the specification are:

Flash Operating Frequency	RWSC	APC
30 MHz	0	0
100 MHz	2	1
133 MHz	3	1
167 MHz	4	1
200 MHz	5	2

A second workaround is if the above RWSC and APC combinations (listed in the description) are desired to be checked, do so without enabling breakpoints. In this case, the first EER or SBC event will be logged, and the MISR will correctly reflect the result of all reads being executed.

e7991: FLASH: Rapid Program or Erase Suspend fail status

Description: If a flash suspend operation occurs during a 5us window during a verify operation being executed by the internal flash program and erase state machine, and the suspend rate continues at a consistent 20us rate after that, it is possible that the flash will not exit the program or erase operation. A single suspend during a single program or erase event will not cause this issue to occur.

Per the flash specification, a flash program or erase operation should not be suspended more than once every 20 us, therefore, if this requirement is met, no issue will be seen. IF the suspend rate is faster than 20 us continuously, a failure to program/erase could occur.

Workaround: When doing repeated suspends during program or erase ensure that suspend period is greater than 20us.

e8341: 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.

e8759: FlexCAN: FD frame format not compliant to the new ISO/CD 11898-1: 2014-12-11

Description: This version of the device implements a Flexible Controller Area Network (FlexCAN) module version that implements a Flexible Data (CAN-FD) frame format according to ISO/WD 11898-1: 2013-12-13. However, it is not compliant with the new ISO/CD 11898-1: 2014-12-11 format. The frame format was updated during the ISO standardization process.

The limitations are the following:

- the FD frame format is incompatible, the Cyclic Redundancy Check [CRC] does not include the added stuff bit count field
- the FD CRC computation is incompatible, a different seed value is used.

As a consequence this device is not suitable for use in CAN-FD networks that use the new FD frame format according to ISO/CD 11898-1: 2014-12-11.

FlexCAN3 with CAN FD feature enabled is affected by this defect.

Workaround: Use CAN-FD mode in networks that only includes devices that conform to the ISO/WD 11898-1: 2013-12-13 frame format.

The Classic CAN mode is unaffected and can be used without restrictions.

e7845: FlexCAN: Limit Flexible Data rate (FD) baud rate to 6.67 Mbps

Description: When operating the FlexCAN module in flexible data rate (FD) mode, the maximum baud rate is limited to 6.67 Mega-bits per second (Mbps).

Workaround: Do not configure the FlexCAN baud rate to a value greater than 6.67 Mbps when using the flexible data rate (FD) mode.

e7433: 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.

e8935: 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.

e7274: 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.

e6428: 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).

e6425: LINFlexD: FIFO buffer can not be accessed without changing the RFC counter

Description: When LINFlexD is enabled in First In, First Out (FIFO) mode by setting the Receive FIFO/ Buffer mode (RFBM) bit in the Universal Asynchronous Receiver/Transmitter Control Register (UARTCR register), the Reception Data Field Length/receive FIFO Counter (RDFL_RFC) will be decremented on every read access of Buffer Data Register Most Significant (BDRM) register through the debugger.

Workaround: If the user does not want the debugger to decrement the RDFL_RFC, access to the BDRM must be avoided in the debugger.

e8933: 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.

e8970: 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)

e7589: LINFlexD: Spurious timeout error when switching from UART to LIN mode or when resetting LINTCSR[MODE] bit in LIN mode

Description: If the LINFlexD module is enabled in Universal Asynchronous Receiver/Transmitter (UART) mode and the value of the MODE bit of the LIN Timeout Control Status register (LINTCSR) is 0 (default value after reset), any activity on the transmit or receive pins will cause an unwanted change in the value of the 8-bit field Output Compare Value 2 (OC2) of the LIN Output Compare register (LINOCR).

If the LINFlexD module is enabled in LIN mode and the value of the MODE bit of the LIN Timeout Control Status register (LINTCSR) is changed from ‘1’ to ‘0’, then the old value of the Output Compare Value 1 (OC1) and Output Compare Value 2 (OC2) of the LIN Output Compare register (LINOCR) is retained.

As a consequence, if the module is reconfigured from UART to Local Interconnect Network (LIN) mode, or LINTCSR MODE bit is changed from ‘1’ to ‘0’, an incorrect timeout exception is generated when the LIN communication starts.

Workaround: If the LINFlexD module needs to be switched from UART mode to LIN mode, before writing UARTCR[UART] to 1, ensure that the LINTCSR[MODE] is first set to 1.

If the LINFlexD module is in LIN mode and LINTCSR[MODE] needs to be switched from 1 to 0 in between frames, the LINOCR must be set to 0xFFFF by software.

e6350: 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).

e10550: MC_CGM: Auxiliary clock dividers get stuck if programmed to divide by 2 and a reset occurs during operation

Description: When a reset of any type occurs during operation, any auxiliary clock divider that is programmed to divide by 2 and is not sourced by the IRCOSC may get stuck and cannot subsequently be reprogrammed.

A stuck auxiliary clock divider output can be detected by the corresponding Clock Monitor Unit (CMU).

Workaround: Changing an auxiliary clock source selection value via software resets all its corresponding dividers and recovers them.

Apply the following sequence after each reset for enabled auxiliary clock dividers that are to be configured to divide by 2 for the application.

1. Disable the corresponding auxiliary clock divider by writing to the Aux Clock Divider Configuration (MC_CGM_ACn_DCx[DE] = 0).

2. Change the auxiliary clock source selection to IRCOSC (MC_CGM_ACn_SC[SELCTL] = 0b0000).

3. Select the desired clock source as the auxiliary clock source (e.g. for PLL0: MC_CGM_ACn_SC[SELCTL] = 0b0010).

4. Configure and enable the corresponding auxiliary clock divider by writing to the Aux Clock Divider Configuration (MC_CGM_ACn_DCx[DIV] = 1 and MC_CGM_ACn_DCx[DE] = 1).

e10505: MC_CGM: The device can become stuck in reset after 'destructive' or external long 'functional' reset

Description: The microcontroller may not exit reset after either a ‘destructive’ reset or an external long ‘functional’ reset if the reset occurs while configuring the system clock divider and switching to the Phase Lock Loop (PLL). Only a Power On Reset (POR) can clear the reset condition. Assertion of the external Power On Reset (PORST) pin will not clear a stuck condition.

Workaround: On reset exit and before setting up the system clocks for the application (i.e., while the system clock is sourced by the IRCOSC, and all the system clock dividers are configured to divide by 1), software must do the following:

1) Determine the reset cause by reading the Functional Event Status (MC_RGM_FES) register and clear the reset flags

2) Depending on the application requirements, if the reset cause was a short external reset due to a RESET_B assertion (MC_RGM_FES[F_EXR] = 1 and MC_RGM_FESS[SS_EXR] = 1) execute one of the following

a. Initiate a software long ‘functional’ reset – this will not execute a self-test but will trigger a device reconfiguration

b. Initiate a software ‘destructive’ reset – this will execute a self-test and a device reconfiguration

3) If the reset was a long external reset, configure the RESET_B to a short external reset (i.e., reset sequence starts at PHASE3[FUNC]) by setting the Functional Event Short Sequence (MC_RGM_FESS) register’s Short Sequence for External Reset (SS_EXR) bit. (By default, on a RESET_B assertion, the MC_RGM executes a long external reset)

a. MC_RGM_FESS[SS_EXR] = 1;

4) Clear the DCF_SELFTEST_CONFIG_IPS_1 register by writing 0x00000000 to address 0xFFF47FE0.

e10506: MC_CGM: The device can become stuck in reset when a reset occurs during a system clock switch to any PLL

Description: When reset occurs during system clock switch to any Phase Locked Loop (PLL) the device may get stuck in reset.

The device is only recoverable by a power on reset (POR).

Workaround: To properly initialize the system clocks and avoid a stuck in reset condition, perform the following steps.

1. Enable ‘functional’ reset escalation: escalate the first ‘functional’ reset to ‘destructive’ reset (MC_RGM_FRET[FRET] = 1).

2. Configure the Dual PLL Digital Interface (PLLDIG) to desired frequencies as per the application requirements.

3. Configure the Clock Generation Module (MC_CGM):

- When selecting the desired PLL as the system clock source, set the divider values as follows:

MC_CGM_SC_DC2 = 0x80040000 (divide by 5) MC_CGM_SC_DC1 = 0x80020000 (divide by 3) MC_CGM_SC_DC0 = 0x80000000 (divide by 1)

- Verify that the system clock divider update has completed (MC_CGM_DIV_UPD_STAT[SYS_UPD_STAT] == 0).

4. Change the system clock source to the desired PLL:

- Select the desired PLL in the target mode’s configuration register (MC_ME_ **mode** _MC[SYSCLK] = 2 for PLL0 or 4 for PLL1)
- Initiate a mode change via the Mode Control (MC_ME_CTL) register.
- Wait until the mode transition has completed.

(**mode** is DRUN, RUN0, RUN1, RUN2, or RUN3,)

NOTE: Do NOT accesses any peripherals running on the Slow Crossbar Clock (SXBAR_CLK) or LIN clock (LINCLK) by any master UNTIL the mode transition has completed.

5. Configure the MC_CGM a second time:

- Set the system clock dividers a second time as follows:

MC_CGM_SC_DC2 = 0x80030000 (divide by 4) MC_CGM_SC_DC1 = 0x80010000 (divide by 2) MC_CGM_SC_DC0 = 0x80000000 (divide by 1) (no change)

- Verify that the system clock divider update has completed (MC_CGM_DIV_UPD_STAT[SYS_UPD_STAT] == 0).

6. Disable ‘functional” reset escalation (MC_RGM_FRET[FRET] = 0)

7. Continue other configurations normally:

- Configure the auxiliary clocks as needed (MC_CGM_ACn_SC and MC_CGM_ACn_DCm registers)
- Enable/disable peripherals as needed (MC_ME_RUN_PCn, MC_ME_LP_PCn, and MC_ME_PCTLn registers).
- Enable/disable core(s) as needed (MC_ME_CADDRn and MC_ME_CCTLn registers).
- Initiate a mode change via the Mode Control (MC_ME_CTL) register.
- Wait until the mode transition has completed.

Ensure that no auxiliary clock is configured to divide by 2 before applying this workaround.

e7211: MC_ME: Core register IAC8 is cleared during a mode change when the core is reset

Description: If a core is reset (ME_CADDR[0,1,2].RMC =1) in the Core Address register during a Mode Entry module (MC_ME) mode change then the Instruction Address Compare 8 (IAC8) register within the core which receives the reset will be cleared. In this implementation IAC8 is used as the Security watchdog service address. If a watchdog time-out occurs after this mode change and no valid service address exists, the core will attempt to execute code from the invalid address potentially resulting in an exception.

The watchdog (SWT) associated with that core is not reset by this change and retains its configuration. If fixed address execution is configured by the Service Mode in the software watchdog control register (SWT_CR.SMD= 0b10) when IAC8 is cleared to 0, it will not be possible to update IAC8 with the correct value. For other service modes the IAC8 register will be cleared to 0, but can be updated.

Workaround: If the software watchdog mode is in fixed address execution (SWT_CR.SMD= 0b10), do not reset the corresponding core upon mode change. For all other modes, IAC8 must be updated by software immediately after the mode transition completes.

e6785: MEMU: ECC errors on FlexCAN RAM accesses indicate an incorrect error address in the peripheral RAM reporting table

Description: Single bit correction and multi-bit error events detected in the FlexCAN RAM by an Error Correction Code (ECC) check are not captured in the Memory Error Management Unit (MEMU) peripheral RAM reporting table with the correct error address.

Workaround: If the Fault Collection and Control Unit (FCCU) receives an alarm alert from the Peripheral RAM Correctable Error (NCF[25]), software can confirm that Peripheral RAM is the source of the failure by checking the Peripheral RAM Correctable Error (PR_CE) flag in the MEMU Error Flag register (MEMU_ERR_FLAG) and by searching for address 0x00000000 in the correctable error reporting table (MEMU_PERIPH_RAM_CERR_ADDR0 and MEMU_PERIPH_RAM_CERR_ADDR1 registers).

If the FCCU receives an alarm alert from Peripheral RAM Uncorrectable Error (NCF[26]), software can confirm that Peripheral RAM is the source of the failure by checking the Peripheral RAM Uncorrectable Error (PR_UCE) flag in the MEMU_ERR_FLAG register and by searching for address 0x00000000 in the uncorrectable error reporting table (MEMU_PERIPH_RAM_UNCERR_ADDR register).

After the source of the failure has been confirmed, software can check the Correctable Error Interrupt Flag (CEIF), Host Access With Non-Correctable Error Interrupt Flag (HANCEIF), and FlexCAN Access With Non-Correctable Error Interrupt Flag (FANCEIF) flags in the FlexCAN Error Status Register (CAN_ERRSR) for all four instances of the FlexCAN in order to determine which specific instance generated the error. The software can then read the FlexCAN Error Report Address Register (CAN_RERRAR) corresponding to the instance which caused the error to obtain the correct relative error address.

e8539: MEMU: Software Reset bit field is not protected by the Register Protection module

Description: The Memory Error Management Unit Control Register Software Reset bit (MEMU_CTRL[SWR]) is not protected by the Register Protection (REG_PROT) module. If the application software inadvertently sets the MEMU_CTRL[SWR] bit, all error flag bits from the MEMU Error Flag Register (MEMU_ERR_FLAG) will be cleared. This issue affects the application if the software polls the MEMU_ERR_FLAG register on every Fault Tolerant Time Interval (FTTI) cycle to monitor for potential ECC errors during memory accesses. See the Safety Overview section of the Functional Safety chapter in the Reference Manual.

Workaround: There are two workarounds possible for this errata:

1) Configure the Fault Collection and Control Unit (FCCU) to generate an interrupt when an Error Correction Coding (ECC) error event from the MEMU is detected. The MEMU Non-Critical Fault (NCF) channels for memory ECC errors in the FCCU are channels 22 and 23 (System Memory), 25 and 26 (Peripheral Memories), and 28 and 29 (Flash Memory).

2) Software must poll the VLD (valid entry) bit field in the following MEMU reporting registers, as these registers are not cleared by MEMU_CTRL[SWR]:

MEMU_SYS_RAM_CERR_STS0;

MEMU_SYS_RAM_CERR_STS1;

MEMU_SYS_RAM_CERR_STS2;

MEMU_SYS_RAM_CERR_STS3;

MEMU_SYS_RAM_CERR_STS4;

MEMU_SYS_RAM_CERR_STS5;

MEMU_SYS_RAM_CERR_STS6;

MEMU_SYS_RAM_CERR_STS7;

MEMU_SYS_RAM_CERR_STS8;

MEMU_SYS_RAM_CERR_STS9;

MEMU_SYS_RAM_UNCERR_STS;

MEMU_SYS_RAM_OFLW0;

MEMU_PERIPH_RAM_CERR_STS0;

MEMU_PERIPH_RAM_CERR_STS1;

MEMU_PERIPH_RAM_UNCERR_STS;

MEMU_PERIPH_RAM_OFLW0;

MEMU_FLASH_CERR_STS0;

MEMU_FLASH_CERR_STS1;

MEMU_FLASH_CERR_STS2;

MEMU_FLASH_CERR_STS3;

MEMU_FLASH_CERR_STS4;

MEMU_FLASH_CERR_STS5;

MEMU_FLASH_CERR_STS6;

MEMU_FLASH_CERR_STS7;

MEMU_FLASH_CERR_STS8;

MEMU_FLASH_CERR_STS10;

MEMU_FLASH_CERR_STS11;

MEMU_FLASH_CERR_STS12;

MEMU_FLASH_CERR_STS13;

MEMU_FLASH_CERR_STS14;

MEMU_FLASH_CERR_STS15;

MEMU_FLASH_CERR_STS16;

MEMU_FLASH_CERR_STS17;

MEMU_FLASH_CERR_STS18;

MEMU_FLASH_CERR_STS19;

MEMU_FLASH_UNCERR_STS;

MEMU_FLASH_OFLW0.

e9631: MEMU: System RAM ECC errors on accesses by some masters report to MEMU Peripheral RAM table

Description: If an Error Correction Code (ECC) error is detected by an access through the Crossbar (XBAR) bus Concentrator(s) from either of the eDMA controller(s) on XBAR Client 0 Concentrator or through the Ethernet or the Serial Inter-Processor Interface (SIPI) masters on XBAR Client 1 Concentrator, the ECC error is reported to the Memory Error Management Unit (MEMU) Peripheral RAM reporting table instead of the System RAM reporting table. ECC errors detected by other system bus masters are reported to the MEMU System RAM table. As a result, System RAM locations may appear in both the System RAM table and the Peripheral RAM table.

Workaround: Expect system RAM ECC errors encountered by masters on the Concentrator(s) to be reported to the MEMU peripheral RAM table, in addition to the MEMU System RAM table if accessed by other bus masters(cores).

e3872: NAR: Emulation device may transmit random trace data during initialization

Description: On emulation devices with multiple instantiations of the Nexus Aurora Router (NAR), if the NAR module directly connected to the Nexus clients (e200zx core Nexus clients, Nexus CrossBar clients, Generic Timer Module Debug Interface) is enabled by the second NAR (NAR on the Buddy Die in the emulation device), then the first NAR may generate 8 words (32-bit) of random data packet that could be transmitted. These would occur after the Device ID is transmitted by the emulation device only NAR. The device ID from the first NAR (in the Production die) may not be transmitted.

Workaround: Either disregard the extraneous data, or enable the NAR (NAR Enable) connected directly to the MCU Nexus clients first in the NAR Control Register (NAR_CR[NEN]=0b1), then enable the additional emulation device NAR.

e3970: NAR: Trace messages include a 6-bit Source Identification field instead of 4-bits

Description: The source field (SRC) of trace messages from the Nexus Aurora Router are 6-bits in length. All other clients implement a 4-bit SRC field. Per the IEEE-ISTO 5001 Standard (Nexus) the SRC field of all clients on a device should be the same length. The two most significant bits of the SRC are 0b00.

Workaround: Tools should treat the SRC field as a 4-bit field for all Nexus clients. In addition, tools should ignore the extra 2-bits as an extra field with no meaning. In the case of the NAR Error Message (TCODE=8), these two bits are between the 4-bit SRC field and the 4-bit Error Type (ETYPE) field. For the NAR Watchpoint Message, these bits are between the 4-bit SRC and the 6-bit Watchpoint Hit (WPHIT) field.

e11021: NAR: Trace to memory buffer pointers corrupted after break-point

Description: When using the trace to memory feature of the Nexus Aurora Router (NAR), some address pointers are not reset properly following a break-point. Therefore, following the break-point, the full trace memory cannot be used to hold trace information. The amount of memory not available depends on the pointer state after the break-point.

Workaround: To use full trace memory debugger following a break-point, perform the additional steps below before enabling trace after any break-point:

1. Perform a soft-reset of the NAR by setting the NAR Soft Reset bit in the NAR Control register (NCR[NSR]=1)

2. Re-configure all the NAR registers (that were cleared by this soft-reset) along with trace memory address register

3. Resume MCU execution from the break-point with trace enabled in the desired trace mode.

e7850: NXMC: XBAR data trace may overrun internal NXMC Message Buffers

Description: Data trace of the crossbar (XBAR or AXBS) master accesses by the Nexus XBAR Multi-Master Trace Client (NXMC) may overrun the Nexus Aurora Router (NAR) if an excessive amount of data trace data is requested by a tool.

Workaround: If an overrun of the NXMC output buffers are seen (errors are getting flagged by the NAR to the trace tool), reduce the number of successive data trace values to a maximum of 64 bytes when tracing reads or writes only (sixteen 32-bit words or 32 16-bit half-words). If both reads and writes are being traced, limit the consecutive data trace messages to successive 16 byte values at a time (four 32-bit words or eight 16-bit half-words). Sufficient time must be allowed for the NXMC buffers to be read by the NAR before resuming data trace. If data trace is limited on the NXDM to approximately 1 data trace message every 10 system clocks (5 Direct Memory Access module clocks), the buffers should not be overrun.

NXCM overflow errors will insert an error message (TCODE = 0x8), with the NXMC source id (SRC = 0xC), the Queue overrun error type (ETYPE = 0x0), and the Data Trace message(s) lost error code (ECODE = 0x02).

An alternative workaround is to allow errors to occur and expect missing data trace information.

e10340: 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.

e7876: PASS: Debug interface state is not preserved through reset on secured devices after a functional reset

Description: The Password Access and Security Module (PASS) module is a security module that uses an optional password mechanism to protect unauthorized access to the debug Interface, which allows the enabling of Nexus clients. After a successful password challenge is met, the debug tool has control of the debug interface. However the state of the interface, a successfully entered password challenge, is not preserved during a functional reset.

Workaround: After a functional reset, the password challenge must be successfully entered again to re-enable the access to the debug interface.

e10552: PASS: JTAG password match bypasses flash read protection set by PASS.LOCK3[RLx] bits

Description: On a censored device, setting the region lock bits in the Pass Lock Register 3 (PASS.LOCK3[RLx]) of the Password and Device Security Module (PASS) should protect the contents of flash memory blocks from being read when a debug tool is attached and the block is enabled for any flash region that has the region lock set. Debug enable may be achieved either through clearing the PASS.LOCK3[DBL] bit or providing a matching JTAG password.

However, providing a matching JTAG password bypasses the region lock control of the PASS.LOCK3[RLx] bits and allows all flash regions to be read, regardless of the PASS.LOCK3[RLx] bit settings.

Workaround: It is not possible to maintain the flash read protection on locked regions if the JTAG password is used to enable the debug interface.

In order to maintain the read protection on locked regions of the flash, only enable the debug interface by clearing the Debug Interface Lock bit (PASS.LOCK3[DBL]) bit. The PASS.LOCK3[DBL] bit can be cleared by providing the correct password sequence to a password challenge/response, either via JTAG Communication module (JDC), or other external interface, such as CAN using software executing on the MCU.

An invalid JTAG password can be used in UTEST flash to disable this feature. Otherwise, programming a random value for the JTAG password match, which is not communicated to users, can greatly reduce the chances of enabling the debug interface and unlocking flash read access.

e9250: PASS: JTAG password not working during reset

Description: The Debug Interface Access cannot be enabled by supplying the JTAG password during reset.

Workaround: To enable the Debug Interface Access, supply the JTAG password after reset.

e7904: PASS: Programming Group Lock bit (PGL) can be de-asserted by multiple masters writing the correct password sections to the CINn registers.

Description: The eight Challenge Input Registers (CINn) in the Password and Device Security Module (PASS) where the 256-bit unlock lock password (8 × 32-bit registers) is provided, can be written by multiple masters. If the written password is correct even though it has been provided from different masters, the password Group Lock (PASS PGL) in the Password Group n Lock 3 Status register (PASS_LOCK3_PGn) is de-asserted and UnLockMaster (MSTR) is set to 0xF.

Therefore, internal registers would not be writable by any of the master other than master whose ID is 0xF if the Master Only (MO) bit is set PASS_LOCK3_PGn.

If a Master wants to update internal registers, it needs to unlock the PASS by writing into all the 8 Password registers.

Workaround: Set the master only bit inside the PASS (LOCK3_PGn.MSTR) to block other master accesses to the unlocked registers. If the written password has been provided from different masters, a single master should perform the unlock operation again by writing into all the 8 password registers.

e7905: 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.

e9496: PMC: FCCU NCF[4] fault may be triggered when DCF_PMC_REE_CTRL or DCF_PMC_RES_CTRL are programmed

Description: The Power Management Controller (PMC) Reset Enable Control (PMC_REE) register and PMC Reset Type (Destructive/Functional) Selection (PMC_RES) register can be programmed by two methods.

(1) By programming the registers directly by software.

(2) Via the Device Client Format (DCF) clients for PMC_REE and PMC_RES (DCF_PMC_REE_CTRL and DCF_PMC_RES_CTRL) by programming the User Test (UTEST) area.

The PMC_RES register controls whether the voltage detect signal event causes a destructive or functional reset depending on whether the desired flag bit is enabled or disabled in PMC_REE register. There are both low voltage and high voltage trigger events that can be enabled.

However, due to unused DCF clients that have the same Address and Chip select as the DCF_PMC_REE_CTRL and DCF_PMC_RES_CTRL DCF clients, a fault may be flagged to the Fault Collection and Control Unit (FCCU) Non-critical Fault 4 (NCF[4]) if either of these DCF clients is used.

The unused DCF clients are the Miscellaneous User Test Spare 0 that will be selected when the DCF_PMC_REE_CTRL is accessed and the Miscellaneous User Test Spare 1 DCF (DCF_UTEST_MISC_SPARE1) that will be selected if the DCF_PMC_RES_CTRL is accessed. DCF_PMC_REE_CTRL and DCF_PMC_RES_CTRL only implement 17 valid bits, but the DCF_UTEST_MISC_SPARE0 and DCF_UTEST_MISC_SPARE1 have 32 valid bits. Since these PMC DCF clients are safe clients, they require triple voted values. Due to the length difference of valid bits, it is not possible to write values for the triple voted mirror locations that satisfy the requirements of both DCF clients.

Workaround: Program the PMC_REE or PMC_RES registers directly by software if required and do not program the DCF_PMC_REE_CTRL or DCF_PMC_RES_CTRL DCF clients in UTEST area.

e8555: PMC: Over and under temperature flags may be set during reset

Description: The Temperature Sensor over or under temperature flags in Temperature Event Status register (PMC_ESR_TD) may improperly be set during a destructive or functional reset. By default, after the initial power on reset, the over/under temperature reset feature is disabled. However, if software enables the Temperature Sensor over/under temperature detection to

cause a destructive/functional in the Temperature Reset Event Enable register (PMC_REE_TD) and Temperature Reset Event Selection register (PMC_RES_TD), these flags cannot be cleared and the device will remain in reset if a destructive or functional reset occurs. Only a power on reset can clear this condition.

Workaround: One of the following must be done:

1. Do not enable the over/under temperature reset feature.

2. Program the Temperature Reset Event Enable Device Configuration Format (DCF) client (DCF_PMC_REE_TD) in the User Test Non-Volatile Flash (UTEST) DCF area starting at 0x00400300 to clear the temperature sensor reset enable. Since this is a triple voted DCF client, two sets DCF records must be set consecutively in the UTEST NVM area. The values should be:

a) The DCF_PMC_REE_TD DCF client should be programmed with the data 0x00000040, 0x0000003F (inverted) & 0x00000020 (right rotated 1 bit).

b) The DCF_PMC_REE_TD DCF client should be programmed with the data 0x00000000, 0x0000007F (inverted) & 0x00000000 (right rotated 1 bit).

The above DCF records ensure that the Temperature Sensor Flags are always masked to the Reset Generation Module (RGM) during the boot sequence and allow the MCU to properly exit reset.

Additionally, to verify that the temperature is within the operating specifications, software can monitor the Temperature via the Analog to Digital Converter (ADC) Temperature Sensor channel to identify if the flags have been spuriously set. This allows software to clear the status flags before re-enabling the over/under temperature reset feature.

e6775: SAR_ADC: Simultaneous calibration of SAR ADCs is not recommended

Description: If the Successive Approximation Analog to Digital Converter (SAR_ADC) calibration and built in self test control and status register (SAR_CALBISTREG) test enable bit (TEST_EN) bit arre set for more than one SAR_ADC at a time, such that more than one SAR_ADC is executing calibration simultaneously, the resulting calibration calculations may be incorrect. This may cause performance degradation.

Workaround: Software should enable calibration for only one SAR_ADC at a time, by setting the CALBISTREG[TEST_EN] bit and then monitoring the calibration test busy bit (C_T_BUSY) bit in the CALBISTREG register. Do not start calibration on any other SAR_ADCs until the C_T_BUSY bit is cleared, indicating that the current calibration is complete.

e8039: SDADC: digital filter and FIFO not disabled when MCR[EN] is cleared

Description: When the Enable bit (EN) of the Sigma-Delta Analog to Digital Converter (SDADC) Module Configuration Register (MCR) is cleared (MCR[EN]=0), the digital part of the SDADC continues operating and does not go to low power mode if the module is disabled while a valid conversion is already in process and the application software continues to initiate conversions. As a consequence, the digital block of the SDADC still produces new conversion results in the Channel Data Register (CDR) and dummy data are transferred to the result First-In, First-Out (FIFO) buffers. In addition, interrupt and/or Direct Memory Access (DMA) events are still generated.

Note: the analog part does enter the power-down mode, reducing the consumption on the ADC high voltage supply domain (VDD_HV_ADV).

Workaround: Do not initiate a conversion prior to enabling the SDADC (MCR[EN]=1). In addition, once the SDADC has been enabled (MCR[EN]=1), if the SDADC needs to be disabled (MCR[EN]=0), prior to clearing the EN bit, either turn off the clock to the SDADC module in the Clock Generation Module (CGM) or Select the External Modulator Mode (EMSEL) by setting the MCR[EMSEL] bit along with the clearing the MCR[EN].

e8225: SDADC: FIFO Flush Reset command requires clearing the Data FIFO Full Flag

Description: When the Sigma-Delta Analog-to-Digital Converter (SDADC) FIFO is flushed by writing '1' to the FIFO Control Register FIFO Flush Reset bit (SDADC_FCR[FRST]), the FIFO is correctly flushed, but the Status Flag Register Data FIFO Full Flag (SDADC_SFR[DFFF]) may be incorrectly asserted, indicating the FIFO is full when it is empty..

Workaround: Clear SDADC_SFR[DFFF] by writing a '1' to this field after performing a FIFO Flush Reset command or after the FIFO is disabled.

e8631: SDADC: low threshold watchdog cannot be used with signed data

Description: Each Sigma Delta Analog to Digital Converter (SDADC) provides a watchdog (WDG) to monitor the converted data range. This watchdog should trigger when a converted value is either higher than the value configured in the WDG Threshold Register Upper Threshold Value bit-field (SDADC_WTHHLR[THRH]), or lower than the value configured in the Lower Threshold Value bitfield (SDADC_WTHHLR[THRL]). Instead, the low WDG threshold acts as a high WDG threshold, triggering when a converted value is greater than the value configured in SDADC_WTHHLR[THRL].

Workaround: There are two workarounds available:

1) Do not use the WDG function by clearing the SDADC Module Control Register Watchdog Enable Bit (SDADC_MCR[WDGEN]).

2) Configure the WDG low threshold SDADC_WTHHLR[THRL] to the value 0x7FFF. This guarantees that a low threshold trigger will not be generated. The WDG high threshold (SDADC_WTHHLR[THRH]) can be used without restriction.

e7204: SENT: Number of Expected Edges Error status flag spuriously set when operating with Option 1 of the Successive Calibration Check method

Description: When configuring the Single Edge Nibble Transmission (SENT) Receiver (SRX) to receive message with the Option 1 of the successive calibration pulse check method (CHn_CONFIG[SUCC_CAL_CHK] = 1), the number of expected edges error (CHn_STATUS[NUM[EDGES_ERR]) gets randomly asserted. Option 2 is not affected as the number of expected edges are not checked in this mode.

The error occurs randomly when the channel input (on the MCU pin) goes from idle to toggling of the calibration pulse.

Note: The Successive Calibration Pulse Check Method Option 1 and Option 2 are defined as follows:

Option 2 : Low Latency Option per SAE specification

Option 1 : Preferred but High Latency Option per SAE specification

Workaround: To avoid getting the error, the sensor should be enabled first (by the MCU software) and when it starts sending messages, the SENT module should be enabled in the SENT Global Control register (by making GBL_CTRL[SENT_EN] = 1). The delay in start of the two can be controlled by counting a fixed delay in software between enabling the sensor and enabling the SENT module. The first message will not be received but subsequent messages will get received and there will be no false assertions of the number of expected edges error status bit (CHn_STATUS[NUM[EDGES_ERR]).

Alternatively, software can count the period from SENT enable (GBL_CTRL[SENT_EN] = 1) to the first expected calibration pulse. If the number of expected edges error status bit (CHn_STATUS[NUM[EDGES_ERR]) is asserted, software can simply clear it as there have no messages which have been completely received.

Alternatively, the software can clear this bit at the start and move ahead. When pause pulse is enabled, then NUM_EDGES will not assert spuriously for subsequent messages which do not have errors in them or cause overflows.

e8082: SENT: A message overflow can lead to a loss of frames combined with NUM_EDGES_ERR being set

Description: In the case of a Single Edge Nibble Transfer (SENT) receiver (Rx) message overflow (CHn_STATUS[FMSG_OFLW] = 1) and if the following registers are continuously being read without clearing the FMSG_RDY[F_RDYn] bit, there is a possibility that one message will be lost. Additionally, if the pause pulse feature is enabled, the module assert up to two NUM_EDGES_ERR in the status register (CHn_STATUS). In this case up to two frames can be lost.

Note that some debuggers perform a continuous read of memory which can cause this issue to occur.

Register	Register Name
CHn_FMSG_DATA	Channel Fast Message Data Read Register
CHn_FMSG_CRC	Channel Cyclic Redundancy Check Register
CHn_FMSG_TS	Channel Fast Message Time-stamp Register

Workaround: 1. Software should ensure that SENT message overflow does not occur.

If interrupts are used (when the Enable FDMA (FDMA_EN) bit of Fast Message DMA Control Register (SRX_FDMA_CTRL) is set to 0) to read the SENT messages, the interrupt for data reception should be enabled by setting the Enable for Fast Message Ready Interrupt (FRDY_IE[n]) bit of Fast Message Ready Interrupt Control Register (SRX_FRDY_IE) for every channel n and the interrupt priority should be such that the software is able to read the message before the next message arrives.

When using Direct Memory Accesses (eDMA) to access the SENT (when the Enable FDMA (FDMA_EN) bit of Fast Message DMA Control Register (SRX_FDMA_CTRL) is set to 1), the DMA request from the SENT module should be serviced before the next message arrives.

The minimum duration between the reception of two consecutive messages in one channel is 92 times the utick length (time).

2. Ensure that the following registers are not read continuously either in the software code or as a result of a debugger being connected. The following registers should be read once per message and the FMSG_RDY[F_RDYn] bit should be cleared after the reads.

Register	Register Name
CHn_FMSG_DATA	Channel Fast Message Data Read Register
CHn_FMSG_CRC	Channel Cyclic Redundancy Check Register
CHn_FMSG_TS	Channel Fast Message Time-stamp Register

e7425: SENT: Unexpected NUM_EDGES_ERR error in certain conditions when message has a pause pulse

Description: When the Single Edge Nibble Transmission (SENT) Receiver (SRX) is configured to receive a pause pulse (Channel ‘n' Configuration Register – CHn_CONFIG[PAUSE_EN] = 1) the NUM_EDGES error can get asserted spuriously (Channel ‘n' Status Register – CHn_STATUS(NUM_EDGES_ERR] = 1) when there is any diagnostic error (other than number of expected edges error) or overflow in the incoming messages from the sensor.

Workaround: Software can distinguish a spurious NUM_EDGES_ERR error from a real one by monitoring other error bits. The following tables will help distinguish between a false and real assertion of NUM_EDGES_ERR error and other errors. Software should handle the first error detected as per application needs and other bits can be evaluated based on these tables. The additional error may appear in the very next SENT frame. Table 1 contains information due to erratum behavior. Table 2 contains clarification of normal NUM_EDGES_ERR behavior.

Table 1. Erratum behavior of NUM_EDGES_ERR

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NIB_VAL_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are seen as back to back calibration pulses and no edges in between.	Ignore both NUM_EDGES_ERR error
FMSG_CRC_ERR	NUM_EDGES_ERR asserted twice	Same as NIB_VAL_ERR.	Ignore both NUM_EDGES_ERR errors
CAL_LEN_ERR	NUM_EDGES_ERR asserted once	Since the calibration pulse is not detected as a valid calibration pulse, the internal edges counter does not detect the end of one message and start of bad message (which has	Ignore NUM_EDGES_ERR error

		CAL_LEN_ERR); hence the NUM_EDGES_ERR gets asserted.	
FMSG_OFLW	NUM_EDGES_ERR asserted once (random occurrence)	A message buffer overflow may lead the state machine to enter a state where it waits for a calibration pulse (behavior also seen in ERR007404). When in this state, the state machine can detect both a Pause pulse and a Calibration pulse as back to back calibration pulses and no edges in between. Then, the NUM_EDGES_ERR can get asserted. Since entry into this state is random, the error can be seen occasionally.	Ignore NUM_EDGES_ERR error

Table 2. Expected behavior, clarification of NUM_EDGES_ERR cases

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NUM_EDGES_ERR (when edges are less than expected)	NIB_VAL_ERR is asserted	When the actual number of edges in the message are less than expected, then a pause pulse gets detected as a nibble since the state machine expects nibbles when actually there is a pause pulse present. This generates NIB_VAL_ERR.	Ignore the NIB_VAL_ERR
NUM_EDGES_ERR (when edges are more than expected)	NIB_VAL_ERR and PP_DIAG_ERR are asserted	When the actual number of edges in a message are more than expected, then after receiving the programmed number of data nibbles, the state machine expects a pause pulse. However, the pause pulse comes later and gets detected as a nibble and hence NIB_VAL_ERR is asserted. Since the message length is not correct, PP_DIAG_ERR is also asserted.	Ignore NIB_VAL_ERR and PP_DIAG_ERR

e7788: 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.

e10489: SIUL2: Open-drain or open-source configured outputs may briefly drive high or low during transitions

Description: The System Integration Unit Lite 2 (SIUL2) Output Drive Control in the Multiplexed Signal Configuration registers (SIUL2_MSCR[ODC]) specifies the type of output drive control for the associated pins. When configured as open-drain, the high drive for the output buffer is disabled and when configured as open-source the low drive for the output buffer is disabled. However, during the transition from the driven to non-driven state, the output may momentarily drive the opposite value. In other words, the open-drain output may momentarily drive high and the open-source output may momentarily drive low. The duration of the contention is 0-10 nanoseconds.

Workaround: Open-drain or open source pins need either internal or external weak pull devices to maintain the default pin state. Strong external drivers connected to the pin need to assure a non-overlap time with the pin driver to prevent contention.

e5570: SIUL2: Read operation of the SIUL2 MSCR bits that are permanently set (enabled) will return 1b0

Description: The System Integration Unit Lite2 (SIUL2) Multiplexed Signal Configuration Register (MSCR) bits for pad control which are permanently set (enabled) will be read as 1b0. These bits are read-only bits.

The MSCR bits which are not configurable are and permanently set (enabled) include SRC, OBE, ODE, SMC, APC, IBE, HYS, PUS, PUE, INV.

Please see the Section “Address map and configuration for MSCR/IMCR registers” in the Signal Description chapter for pad configuration options of the Reference Manual.

Workaround: Software should expect to read back 1b0 when reading the permanently set bits in the MSCR.

e7871: SPC5746R:Current injection causes leakage path across the DSPI and LFAST LVDS pins

Description: The General Purpose Input/Output (GPIO) digital pins (including all digital CMOS input or output functions of the pin) connected to the differential LVDS drivers of the Deserial/Serial Peripheral Interface (DSPI) and LVDS Fast Asynchronous Serial Transmit Interface (LFAST) do not meet the current injection specification given in the operating conditions of the device electrical specification. When the LVDS transmitter or receiver is disabled and current is

positively or negatively injected into one pin of the GPIO pins connected to the differential pair, a leakage path across the internal termination resistor of the receiver or through the output driver occurs potentially corrupting data on the complementary GPIO pin of the differential pair. All LFAST and DSPI LVDS receive and transmit GPIO pairs on the SPC5746R exhibit the current injection issue.

There is an additional leakage path for the LFAST receive pins through the loopback test path when current is negatively injected into a GPIO pin connected to an LFAST pair. In this case current will be injected into the same terminal of the GPIO pin connected through the loopback path (positive terminal to positive terminal, negative terminal to negative terminal). The pins affected by the loopback path on the SPC5746R are: PA[1] to/from PA[5], and PA[2] to/from PA[6].

There is no leakage issue when the pins are operating in normal LVDS mode (both LVDS pairs of the LFAST interface configured as LVDS).

Workaround: As long as the GPIO pad pins are operated between ground (VSS_HV_IO) and the Input/ Output supply (VDD_HV_IO) then no leakage current between the differential pins occurs. If the GPIO pad is configured as an input buffer then the input voltage cannot be above the supply, below ground, and no current injection is allowed. If the GPIO pad is configured as an output care should be taken to prevent undershoot/overshoot/ringing during transient switching of capacitive loads. This can be done by carefully configuring the output drive strength to the capacitive load and ensuring board traces match the characteristic impedance of the output buffer to critically damp the rising and falling edges of the output signal.

e9658: SPI: Inconsistent loading of shift register data into the receive FIFO following an overflow event

Description: In the Serial Peripheral Interface (SPI) module, when both the receive FIFO and shift register are full (Receive FIFO Overflow Flag bit in Status Register is set (SR [RFOF] = 0b1)) and then the Clear Rx FIFO bit in Module Configuration Register (MCR [CLR_RXF]) is asserted to clear the receive FIFO, shift register data is sometimes loaded into the receive FIFO after the clear operation completes.

Workaround: 1. Avoid a receive FIFO overflow condition (SR[RFOF] should never be 0b1). To do this, monitor the RX FIFO Counter field of the Status Register (SR[RXCTR]) which indicates the number of entries in receive FIFO and clear before the counter equals the FIFO depth.

2. Alternatively, after every receive FIFO clear operation (MCR[CLR_RXF] = 0b1) following a receive FIFO overflow (SR[RFOF] = 0b1) scenario, perform a single read from receive FIFO and discard the read data.

e7929: SSCM: FLASH ECC errors are reported after the SSCM execution

Description: During the reset phase, the System Status and Configuration Module (SSCM) reads device configuration information from FLASH memory, which generates expected Error Correction Coding (ECC) errors in the FLASH memory. These expected ECC errors are intended to be masked and not logged in the FLASH Module Configuration Register ECC Error Bit (FLASH_MCR[EER]). However, the SSCM incorrectly allows the FLASH_MCR[EER] bit to be set when the part comes out of reset, rather than cleared. This can potentially mask any legitimate FLASH ECC errors that could occur later.

Workaround: Immediately after the Boot Assist Flash (BAF) passes control to the user application code, software should clear the false ECC Log event by setting the FLASH_MCR[ERR] bit to 0b1.

e10088: 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.

e9493: STCU: FCCU NCF 6 does not automatically generate a destructive reset, when DCF_UTEST_Miscellaneous.STCU_CFB bit field is set (1)

Description: When a critical fault (CF) occurs during Self-Test Control Unit (STCU) offline self test, the Fault Collection and Control Unit (FCCU) Non Critical Fault (NCF) 6 should be set and if the DCF_UTEST_Miscellaneous.STCU_CFB bit field is set (1), an automatic destructive reset reaction is expected, however, regardless of DCF_UTEST_Miscellaneous.STCU_CFB bit field status, no destructive reset is asserted.

Workaround: The application software must read the FCCU NCF 6 status and STCU status to check whether a CF was latched and initiate a destructive reset.

e10507: STCU: The device can become stuck in reset when destructive reset occurs during offline self-test when PLL is selected as BIST clock source

Description: If a ‘destructive’ reset occurs while offline built-in self-test (BIST) is in progress and the Phase-Locked Loop (PLL) is selected as offline BIST clock source, the device may get stuck in reset.

The device is only recoverable by a power on reset (POR). Assertion of the external Power On Reset (PORST) pin will not clear a stuck condition.

Workaround: To perform an offline self test, configure the STCU to use a PLL frequency of 100MHz.

In addition, configure the system clock dividers in DCF_SELFTEST_CONFIG_IPS_2 DCF client to divide by 1, 1 and 3 (FXBAR 100MHz; SXBAR 100MHz; PBRIDGE 33.3MHz)

DCF_SELFTEST_CONFIG_IPS_2[20:22] = 0

DCF_SELFTEST_CONFIG_IPS_2[23:25] = 0

DCF_SELFTEST_CONFIG_IPS_2[26:28] = 2

e6851: SWT: Software Timer interrupt may not cause device to exit STOP mode under certain clock configurations

Description: The Software Watchdog Timer (SWT) can generate a pulse that is latched by the Mode Entry module (MC_ME), which then initiates an exit from STOP mode.

The MC_ME is clocked at the peripheral bridge clock frequency (PBRIDGEx_CLK), which is the system clock frequency divided by 4. However, during STOP mode the system clock is switched by the Clock Generation Module (MC_CGM) to the 16MHz IRC (Internal RC Oscillator), so the resulting PBRIDGEx_CLK frequency is 4 MHz.

Because the SWT wakeup pulse to the MC_ME is generated on the undivided IRC reference (16 MHz), there is a chance that the MC_ME, being clocked at 4 MHz during STOP mode, may miss the wakeup pulse assertion and fail to exit STOP mode.

Workaround: 1) Using the MC_ME, configure RUNn mode (n = 1,2, or 3) to use the IRC as the system clock source by setting the IRC On bit (IRCON) in the Mode Entry RUNn Mode Configuration register for the selected RUNn mode.

2) If the SWT 0/Core 0 pair is to be used to exit from STOP mode, configure Core 0 to be active in the selected RUNn mode, using the Mode Core 0 Control Register (For MPC5746R this is ME_MCCTL[1]). SWT 1/Core 1 are active for all run modes by default.

3) Switch to the RUNn mode chosen for the system configurations to take effect.

4) Reconfigure the CGM System Clock Divider Configuration 0, 1, and 2 registers (MC_CGM_SC_DC0, MC_CGM_SC_DC1 and MC_CGM_SC_DC2) so that the PBRIDGE_CLK is set to 16MHz.

5) Using the MC_ME, initiate the change to STOP mode.

6) When the SWT generates an interrupt, the system will exit STOP mode and resume in the last run mode selected.

7) Reconfigure the MC_CGM_SC_DC0, MC_CGM_SC_DC1 and MC_CGM_SC_DC2 divider registers to put the system back at the desired operating frequency.

8) Using the MC_ME initiate a mode change back to RUN0.

e7948: TDM: Erase protection not enabled by reset

Description: When writing a record to the Tamper Detection Module (TDM) diary to allow erase of a flash block, if functional reset occurs during the erase operation, further erase of the flash block may be allowed until the next destructive reset.

Workaround: Prior to programming a flash block protected by a TDM diary, verify that the tamper region is locked by reading the TDR Status Register (TDM_TDRSR). If the TDRSRx bit of the region indicates the TDR is unlocked, the TDR should be locked (prior to programming) by performing a destructive reset. Other alternatives are to check the TDM_TDRSR after reset, and issue destructive reset if the TDR is unlocked. Or if the TDM_TDRSR indicates the TDR is unlocked, issue another erase operation.

e8310: 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.

e8730: 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.

e4136: XOSC and IRCOSC: Bus access errors are generated in only half of non-implemented address space of XOSC and IRCOSC, and the other half of address space is mirrored

Description: Bus access errors are generated in only half of the non-implemented address space of Oscillator External Interface (40MHz XOSC) and IRCOSC Digital Interface (16MHz Internal RC oscillator [IRC]). In both cases, the other half of the address space is a mirrored version of the 1st half. Thus reads/writes to the 2nd half of address space will actually read/write the registers of corresponding offset in the 1st half of address space.

Workaround: Do not access unimplemented address space for XOSC and IRCOSC register areas OR write software that is not dependent on receiving an error when access to unimplemented XOSC and IRCOSC space occurs.

e7947: XOSC: Incorrect external oscillator status flag after CMU event clear

Description: If an external oscillator (XOSC) is enabled and it becomes unstable (or the crystal fails), the Oscillator Lost Reference status flag in the Clock Monitor Unit Interrupt Status register (CMU0.CMU_ISR[OLRI]) will be set. In addition, the Crystal Oscillator Status flag in the Mode Entry module Global Status Register (MC_ME_GS.S_XOSC) will be cleared (1 = stable clock,

0 = no valid clock). However, if the CMU_ISR[OLRI] is cleared while the oscillator is still in a failing condition, the MC_ME_GS.S_XOSC will incorrectly be set, indicating a valid crystal oscillator.

Workaround: Monitor the XOSC external oscillator status using the MC_ME_GS.S_XOSC before the CMU0.CMU_ISR.OLRI flag is set. After the CMU0.CMU_ISR.OLRI flag has been set, the MC_ME_GS.S_XOSC flag is valid only after a functional reset. Alternately, the response to the OLRI flag after loss of XOSC clock, can be set in the FCCU to cause a functional reset to clear the MC_ME_GS.S_XOSC flag.

e10436: 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.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support