

## Mask Set Errata for Mask 1M35Y

This report applies to mask 1M35Y for these products:

- MPC563xM

### Mask Specific Information

Major mask revision number	0x2
Minor mask revision number	0x1
JTAG identifier	0x2AE0_101D

**Table 1. Errata and Information Summary**

Erratum ID	Erratum Title
ERR003521	DECFIL: Soft reset failures at the end of filtering
ERR008251	DECFIL: timestamp may be lost in edge trigger mode
ERR001267	DSPI: DSPI Microsecond Bus Limitations
ERR009976	DSPI: Incorrect data received by master with Modified transfer format enabled when using Continuous serial communication clock mode
ERR006026	DSPI: Incorrect SPI Frame Generated in Combined Serial Interface Configuration
ERR007352	DSPI: reserved bits in slave CTAR are writable
ERR001082	DSPI: set up enough ASC time when MTFE=1 and CPHA=1
ERR010755	DSPI: Transmit and Receive FIFO fill flags in status register is not cleared when DMA is improperly configured
ERR050782	e200: Time Base TBU register contains wrong value during TBL rollover
ERR011235	EMIOS: Any Unified Channel running in OPWMB or OPWMCB mode may function improperly if the source counter bus is generated by Unified channel in MC mode
ERR050575	eMIOS: Any Unified Channel running in OPWMCB mode may function improperly if the lead or trail dead time insertion features is used and its timebase is generated by Unified channel in MCB mode
ERR011293	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
ERR011295	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
ERR011294	EMIOS: OPWFMB and MCB mode counter rollover resets the counter to 0x0 instead of 0x1 as mentioned in the specification
ERR009978	eMIOS: Unexpected channel flag assertion during GPIO to MCB mode transition

*Table continues on the next page...*



**Table 1. Errata and Information Summary (continued)**

Erratum ID	Erratum Title
ERR004480	eQADC: Differential conversions with 4x gain may halt command processing
ERR003378	EQADC: Pull devices on differential pins may be enabled for a short period of time during and just after POR
ERR005128	eQADC: Some channels do not meet SNR specification
ERR005086	eQADC: unexpected result may be pushed when Immediate Conversion Command is enabled
ERR001297	eSCI : reads of the SCI Data Register, which clears the RDRF flag, may cause loss of frame if read occurs during reception of the STOP bit
ERR009344	eSCI: Late assertion of Transmit Data Ready Interrupt Flag (TXRDY) for Local Interconnect Network (LIN) frame receive (RX) operation
ERR009001	eSCI: Incorrect behavior while in LIN Standard Bit error detection mode
ERR001221	eSCI: LIN bit error indicated at start of transmission after LIN reset
ERR001381	eSCI: LIN Wakeup flag set after aborted LIN frame transmission
ERR009361	eSCI: Timing of TXRDY interrupt flag assertion is incorrect for LIN TX Frame
ERR009797	eSCI: Unable to send next frame after timeout in LIN mode
ERR005642	ETPU2: Limitations of forced instructions executed via the debug interface
ERR002740	ETPU2: Watchdog Status Register (WDSR) may fail to update on channel timeout
ERR005640	ETPU2: Watchdog timeout may fail in busy length mode
ERR008194	eTPU: EAC may detect double teeth in a single input transition
ERR008252	eTPU: ETPU Angle Counter (EAC) Tooth Program Register (TPR) register write may fail
ERR009090	eTPU: Incorrect eTPU angle counter function under certain conditions
ERR009809	eTPU: MDU flags(Overflow/Carry) may be set incorrectly
ERR051054	eTPU: MRL Branch Condition at the start of the thread may differ from the actual MRL state if channel runs at full speed
ERR003114	FLASH: Erroneous update of the ADR register in case of multiple ECC errors
ERR003196	FLASH: PFCR3 is not directly writable
ERR005498	Flash: Prefetch during program/erase operation causes system bus stop
ERR007322	FlexCAN: Bus Off Interrupt bit is erroneously asserted when soft reset is performed while FlexCAN is in Bus Off state
ERR003407	FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1
ERR002379	FMPLL: Loss-of-clock detection may cause unexpected reset
ERR003159	MPC563xM/SPC563M: MIDR MASKNUM field is set to 0x21
ERR007590	MPC563xM: Incorrect JTAG ID[MIC] and MIDR[S_F] register values
ERR003205	NEXUS: EVTI not functional on QFP176 and BGA208 packages
ERR006726	NPC: MCKO clock may be gated one clock period early when MCKO frequency is programmed as SYS_CLK/8.and gating is enabled
ERR002338	Pad Ring: Leakage if VDDE is greater than VDD33
ERR003377	Pad Ring:Nexus pins may drive an unknown value immediately after power up but before the 1st clock edge
ERR009109	PAD_RING: Output pulse may occur on analog inputs during power on reset
ERR011321	PIT_RTI: Generates false RTI interrupt on re-enabling
ERR003425	PMC: 5V VDDREG POR De-assertion Max Level 4.2V

*Table continues on the next page...*

**Table 1. Errata and Information Summary (continued)**

Erratum ID	Erratum Title
ERR003221	PMC: SRAM standby power low voltage detect circuit is not accurate
ERR009682	SPI: Inconsistent loading of shift register data into the receive FIFO following an overflow event
ERR001421	SWT: switching SWT to system clock has very small chance of causing the SWT to enter an indeterminate state

**Table 2. Revision History**

Revision	Changes
August 2015	The following errata have been added 8251 8252 8194 9001 9090 9109 9361
December 2016	The following errata were added. <ul style="list-style-type: none"> <li>• ERR009344</li> <li>• ERR009682</li> <li>• ERR009797</li> <li>• ERR009809</li> <li>• ERR009976</li> <li>• ERR009978</li> </ul>
September 2018	The following errata were added. <ul style="list-style-type: none"> <li>• ERR010755</li> <li>• ERR011235</li> <li>• ERR011293</li> <li>• ERR011294</li> <li>• ERR011295</li> <li>• ERR011321</li> </ul>
AUG2021	The following errata were added. <ul style="list-style-type: none"> <li>• ERR001082</li> <li>• ERR001267</li> <li>• ERR050575</li> <li>• ERR050782</li> <li>• ERR051054</li> </ul> <p>The following erratum was revised.</p> <ul style="list-style-type: none"> <li>• ERR011235</li> </ul>
FEB2022	The following erratum was revised. <ul style="list-style-type: none"> <li>• ERR050575</li> </ul>

## ERR003521: DECFIL: Soft reset failures at the end of filtering

**Description:** If a software reset of a decimation filter is made exactly at the time it finishes filtering, several registers reset for one clock, but have their values updated by the filtering on the next clock, including (but not limited to) the integrator current value register DECFIL\_CINTVAL and the tap registers DECFILTER\_TAPn.

**Workaround:** Before making the soft reset write (DECFIL\_MCR bit SRES=1), perform the procedure below:

- 1- disable filter inputs, writing DECFIL\_MCR bit IDIS = 1.
- 2- read the register DECFIL\_MSR and repeat the read until the bit BSY is 0.
- 3- repeat the loop of step 2; this is necessary to cover the case when a sample is left in the input buffer.

## ERR008251: DECFIL: timestamp may be lost in edge trigger mode

**Description:** The Enhanced Queued Analog-to-Digital Converter (eQADC) supports a conversion command that configures it to send a timestamp along with the specified ADC conversion data to the Decimation Filter (DECFIL) for digital processing. The DECFIL receives the data and the timestamp, and updates internal registers with these two values. When the DECFIL is configured for edge triggered output by setting the Triggered Output Result Enable bit in the Module Configuration Register (DECFIL\_MCR[TORE]) and setting the Trigger Mode bitfield to either 2b00 or 2b10, and a trigger edge is detected, the DECFIL loads an Internal Output Buffer register (DECFILT\_IOB) with the conversion data, and then the timestamp data. This register is intended to hold data to be returned on one of the two Parallel Side Interfaces (PSIO or PSI1). In the case where a trigger edge occurs and DECFILT\_IOB is loaded with the conversion and timestamp data, and then a second trigger edge occurs before any new conversion and timestamp data has been received by the DECFILT, the DECFILT will provide only the initial conversion data, and will not provide the initial timestamp data.

The level triggered mode is not affected by this issue.

**Workaround:** When the DECFILT has been configured for edge triggered output buffer mode, ensure that the trigger edge rate is slower than the input data rate of the decimation filter. That is, be sure that there is always a new conversion arriving at the DECFILT before any new output trigger edge is detected. If the DECFILT is not receiving timestamps from the eQADC, this limitation is not required.

## ERR001267: DSPI: DSPI Microsecond Bus Limitations

**Description:** The DSPI module implements Microsecond Bus functionality. There are limitations in this implementation that result in partial compliance to the specification.

The DSPI Microsecond Bus implementation has the following limitations:

1. HW Injection of command frames which are greater than or equal to 17-bits (inclusive of command select bit) is not supported.
2. Data frames which are greater than 32-bits (inclusive of data select bit) are not supported.

3. Not all integer values between 2 and 32 SCK cycles are supported for Passive Phase timing.
4. Data frames must equal Command Frames in size.
5. Continuous Serial Clock must be enabled for all Microsecond Bus operations.
6. DSPI baud rate is limited to  $F_{sys}/4$  for all Microsecond Bus operations.
7. Dual Receiver Mode frames must be less than or equal to 32-bits (inclusive of data select bits).
8. The chip select transition in Dual Receiver mode occurs on the falling edge of the last clock in the 1st Active Phase, rather than the rising edge of the first clock in the 2nd Active Phase as specified.

**Workaround:** Workarounds for Limitations:

1. For command frames greater than or equal to 17-bits (including selection bit), SW should be used to inject these on the bus during idle transmission.

SW Should follow this routine:

- Set DSPI\_MCR[HALT] = 1
- Poll DSPI\_SR[TXRSR] until set to 0, to ensure DSPI is in STOPPED state
- Re-Configure DSPI for MSC Command Frame
  - TXSS=1, CID=1, TSBCNT = No. Cmd Bits 1
- Write Command to ASDR, command must be different to previous contents of ASDR.
  - If command is the same as previous ASDR contents, then a “dummy” frame must be sent with chip select disabled.
- Clear Transmit Complete flag, DSPI\_SR[TCF] = 1
- Request Running state, DSPI\_MCR[HALT] = 0
- Poll DSPI\_SR[TCF] until set to 1, to ensure Command Frame has sent
- Request Halt State, DSPI\_MCR[HALT] = 1
- Poll DSPI\_SR[TXRSR] until set to 0, to ensure DSPI is in STOPPED state
- Re-Configure DSPI for MSC Data Frame
  - TXSS=0, CID=0, TSBCNT = No. Data Bits 1
- Request Running state, DSPI\_MCR[HALT] = 0

2. Do not use this Microsecond Bus feature.
3. Passive Phase time determined by:  $(1 / F_{sys}) \times DSPI\_CTARx[PDT] \times DSPI\_CTARx[DT]$ .
4. Do not use command and data frames of different sizes.
5. Do not use a non-continuous serial clock.
6. Do not use a baud rate greater than  $F_{sys}/4$
7. Do not use this Microsecond Bus feature.
8. Use a Microsecond bus slave which latches command/data frame bits on the rising clock edge.

**ERR009976: DSPI: Incorrect data received by master with Modified transfer format enabled when using Continuous serial communication clock mode**

**Description:** When the Deserial Serial Peripheral Interface (DSPI) module is configured as follows:

1. Master mode is enabled (Master/Slave Mode Select bit in Module Configuration Register is set (DSPI\_MCR [MSTR] = 0b1))

2. Modified transfer format is enabled (Modified Transfer Format Enable bit in Module Configuration Register is set (DSPI\_MCR [MTFE] = 0b1))

3. Continuous serial communication clock mode is enabled (Continuous SCK Enable bit in Module Configuration Register is set (DSPI\_MCR [CONT\_SCKE] = 0b1))

In this configuration if the frame size of the current frame is greater than the frame size of the next received frame, corrupt frames are received in two scenarios:

a) Continuous Peripheral Chip Select Enable bit in PUSH TX FIFO Register is set (DSPI\_PUSHR [CONT] = 0b1)

b) DSPI\_PUSHR [CONT] = 0b0 and lower significant bit of the frame is transferred first (LSB first bit in Clock and Transfer Attributes Register is set (DSPI\_CTAR [LSBFE] = 0b1))

**Workaround:** To receive correct frames:

a) When DSPI\_PUSHR [CONT] = 0b1, configure the frame size of the current frame less than or equal to the frame size of the next frame (for all frames).

b) When DSPI\_PUSHR [CONT] = 0b0, configure DSPI\_CTAR [LSBFE] = 0b0. Alternatively, configure the frame size of the current frame less than or equal to the frame size of the next frame (for all frames).

Make sure that for all received frames, the bits are read equal to their respective frame sizes and any extra bits during POP operation are masked.

## **ERR006026: DSPI: Incorrect SPI Frame Generated in Combined Serial Interface Configuration**

**Description:** In the Combined Serial Interface (CSI) configuration of the Deserial Serial Peripheral Interface (DSPI) where data frames are periodically being sent (Deserial Serial Interface, DSI), a Serial Peripheral Interface (SPI) frame may be transmitted with incorrect framing.

The incorrect frame may occur in this configuration if the user application writes SPI data to the DSPI Push TX FIFO Register (DSPI\_PUSHR) during the last two peripheral clock cycles of the Delay-after-Transfer (DT) phase. In this case, the SPI frame is corrupted.

**Workaround:** Workaround 1: Perform SPI FIFO writes after halting the DSPI.

To prevent writing to the FIFO during the last two clock cycles of DT, perform the following steps every time a SPI frame is required to be transmitted:

Step 1: Halt the DSPI by setting the HALT control bit in the Module Configuration Register (DSPI\_MCR[HALT]).

Step 2: Poll the Status Register's Transmit and Receive Status bit (DSPI\_SR[TXRXS]) to ensure the DSPI has entered the HALT state and completed any in-progress transmission. Alternatively, if continuous polling is undesirable in the application, wait for a fixed time interval such as 35 baud clocks to ensure completion of any in-progress transmission and then check once for DSPI\_SR[TXRXS].

Step 3: Perform the write to DSPI\_PUSHR for the SPI frame.

Step 4: Clear bit DSPI\_MCR[HALT] to bring the DSPI out of the HALT state and return to normal operation.

Workaround 2: Do not use the CSI configuration. Use the DSPI in either DSI-only mode or SPI-only mode.

Workaround 3: Use the DSPI's Transfer Complete Flag (TCF) interrupt to reduce worst-case wait time of Workaround 1.

Step 1: When a SPI frame is required to be sent, halt the DSPI as in Step 1 of Workaround 1 above.

Step 2: Enable the TCF interrupt by setting the DSPI DMA/Interrupt Request Select and Enable Register's Transmission Complete Request Enable bit (DSPI\_RSER[TCF\_RE])

Step 3: In the TCF interrupt service routine, clear the interrupt status (DSPI\_SR[TCF]) and the interrupt request enable (DSPI\_RSER[TCF\_RE]). Confirm that DSPI is halted by checking DSPI\_SR[TXRXS] and then write data to DSPI\_PUSHR for the SPI frame. Finally, clear bit DSPI\_MCR[HALT] to bring the DSPI out of the HALT state and return to normal operation.

### **ERR007352: DSPI: reserved bits in slave CTAR are writable**

**Description:** When the Deserial/Serial Peripheral Interface (DSPI) module is operating in slave mode (the Master [MSTR] bit of the DSPI Module Configuration Register [DSPIx\_MCR] is cleared), bits 10 to 31 (31 = least significant bit) of the Clock and Transfer Attributes Registers (DSPIx\_CTARx) should be read only (and always read 0). However, these bits are writable, but setting any of these bits to a 1 does not change the operation of the module.

**Workaround:** There are two possible workarounds.

Workaround 1: Always write zeros to the reserved bits of the DSPIx\_CTARn\_SLAVE (when operating in slave mode).

Workaround 2: Mask the reserved bits of DSPIx\_CTARn\_SLAVE when reading the register in slave mode.

### **ERR001082: DSPI: set up enough ASC time when MTFE=1 and CPHA=1**

**Description:** When the DSPI is being used in the Modified Transfer Format mode (DSPI\_MCR[MTFE]=1) with the clock phase set for Data changing on the leading edge of the clock and captured on the following edge in the DSPI Clock and Transfer Attributes Register (DSPI\_CTARn[CPHA]=1), if the After SCK delay scaler (ASC) time is set to less than 1/2 SCK clock period the DSPI may not complete the transaction – the TCF flag will not be set, serial data will not received, and last transmitted bit can be truncated.

**Workaround:** If the Modified Transfer Format mode is required DSPI\_MCR[MTFE]=1 with the clock phase set for serial data changing on the leading edge of the clock and captured on the following edge in the SCK clock (Transfer Attributes Register (DSPI\_CTARn[CPHA]=1) make sure that the ASC time is set to be longer than half SCK clock period.

### **ERR010755: DSPI: Transmit and Receive FIFO fill flags in status register is not cleared when DMA is improperly configured**

**Description:** The Deserial/Serial Peripheral Interface Transmit and Receive First In/First Out (FIFO) buffers can request additional information to be transferred via the Direct Memory Access (DMA) module when either the Transmit or Receive FIFO Fill/Drain Flags are set in the DSPI Status Register (SR[TFFF/RDFD]). However, the Transmit Fill Flag indicates that at least 1 location each (2 bytes each) in the Transmit and Command FIFOs 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 (2 bytes) 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/Drain Flags may not be properly cleared indicating that the FIFO is not full even when the FIFO is actually full (for Transmit FIFO) and not empty when the FIFO is actually empty (for Receive FIFO).

**Workaround:** Properly configure the DMA to fill/drain only 2 bytes to Transmit, Command and Receive FIFOs. Use the DMA loop to transfer more data if needed.

### **ERR050782: e200: Time Base TBU register contains wrong value during TBL rollover**

**Description:** The e200 Time Base (TB) facility is a 64-bit structure provided for maintaining the time of day and operating interval timers. The TB consists of two 32-bit registers – time base upper (TBU) and time base lower (TBL). TBU and TBL are concatenated to provide a long-period 64-bit counter. TBL increments until its value becomes 0xFFFF\_FFFF. The intended behavior is that at the next increment when the TBL value becomes 0x0000\_0000 that the TBU value is incremented. But the actual behavior is that after the TBL value becomes 0x0000\_0000, the TBU value will not increment until the transition of the TBL value to 0x0000\_0001.

**Workaround:** Software will need to take care about the wrong TBU value during TBL rollover. Use the following sequence for reading TBU and TBL values:

```
loop:  
mfspr r12, TBL  
mfspr r3, TBU  
mfspr r4, TBL  
cmpl r4,r12  
se_bt loop
```

### **ERR011235: EMIOS: Any Unified Channel running in OPWMB or OPWMCB mode may function improperly if the source counter bus is generated by Unified channel in MC mode**

**Description:** The Unified channel (UC) configured in Center Aligned Output Pulse Width Modulation Buffered (OPWMCB) or Output Pulse Width Modulation Buffered (OPWMB) modes is not working properly when it is sourced from the UC configured in Modulus Counter (MC) mode by setting the eMIOS Channel Control Register (EMIOS\_CCR) MODE bitfield to 0x10 or 0x11 and any of its pre-scalers (internal or global) divider ratio is higher than 1.

**Workaround:** When a counter bus is generated by the UC set in the MC mode with any pre-scaler (internal or global) divider ration higher than 1, don't use this counter bus for the UC set in OPWMCB or OPWMB mode.



**ERR050575: eMIOS: Any Unified Channel running in OPWMCB mode may function improperly if the lead or trail dead time insertion features is used and its timebase is generated by Unified channel in MCB mode**

**Description:** The Unified channel (UC) configured in Center Aligned Output Pulse Width Modulation Buffered (OPWMCB) mode is not working properly when:

1. It's timebase is sourced from the UC configured in Modulus Counter Buffered (MCB) mode.
2. The lead or trail dead time insertion features is used.
3. Its channel prescaler is different than timebase channel prescaler.

**Workaround:** Channel configured in OPWMCB mode with lead or trail dead time insertion features enabled must have channel prescaler equal to the timebase channel prescaler configured in MCB mode.

**ERR011293: 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 Channel 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.

**ERR011295: 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.

## **ERR011294: 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.

## **ERR009978: 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.

## **ERR004480: eQADC: Differential conversions with 4x gain may halt command processing**

**Description:** If the four times amplifier is enabled for a differential analog-to-digital conversion in the Enhanced Queued Analog to Digital Converter (eQADC) and the ADC clock prescaler is set to divide by 12 or greater, then the ADC will stop processing commands if a conversion command is executed immediately after a differential, gain 4x conversion.

**Workaround:** 1) Do not use a prescaler divide factor greater than or equal to 12 (11 can be used on devices that support odd prescalers).  
2) Insert a dummy write command to any internal ADC register after every 4x conversion command.

Note 1: If the command FIFO preemption feature is used and it is possible to preempt a FIFO which contains the 4x conversion + dummy write workaround, then the preempting command FIFO must be loaded FIRST with a dummy write command and then the desired preempting conversion command in order to avoid the possibility of following a 4x conversion command with another conversion command in the same ADC.

Note 2: The level sensitive triggers (when in Low/High Level Gated External Trigger, Single/Continuous Scan modes) can interrupt the command sequence at any point in time, potentially breaking the safe sequence 4x conversion command -> dummy write command.

Note 3: When using an odd prescaler ( $ADCx\_CLK\_ODD = 1$ ), the duty cycle setting ( $ADCxCLK\_DTY$ ) must be kept at the default setting of 0.

### **ERR003378: EQADC: Pull devices on differential pins may be enabled for a short period of time during and just after POR**

**Description:** The programmable pull devices (up and down) on the analog differential inputs of the eQADC may randomly be enabled during the internal Power On Reset (POR) and until the 1st clock edge propagates through the device. After the first clock edge, the pull resistors will be disabled until software enables them.

**Workaround:** Protect any external devices connected to the differential analog inputs. The worst case condition is with a 1.4K ohm resistor to VDDA (5K pull-up enabled) or VSSA (5K pull-down enabled). This may also cause temporary additional current requirements on the VDDA supply of each eQADC module, up to 15 mA on each eQADC if both the pull up and pull down resistors are enabled simultaneously on all of the differential analog pins.

### **ERR005128: eQADC: Some channels do not meet SNR specification**

**Description:** Some eQADC channels do not meet the SNR (Signal to Noise Ratio) specification in the device data sheet of -55 dB. The following table shows the performance of all of the device analog channels.

SNR		Channels
> 40 dB	8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 31, 32, 33, 34, 35, 38, 39	
> 50 dB	22, 23, 24, 25, 27, 28, 30, 37	
> 55 dB	0, 1, 2, 3, 4, 5, 6, 7, 21, 36 (meet specification)	

**Workaround:** Assign the application SNR-sensitive signals to the analog channels that provide a better signal-to-noise ratio.

### **ERR005086: eQADC: unexpected result may be pushed when Immediate Conversion Command is enabled**

**Description:** In the enhanced Queued Analog to Digital Converter (eQADC), when the Immediate Conversion Command is enabled ( $ICEAn=1$ ) in the eQADC\_MCR (Module Configuration Register), if a conversion from Command First-In-First Out (CFIFO0, conv0) is requested

concurrently with the end-of-conversion from another, lower priority conversion (convx), the result of the convx may be lost or duplicated causing an unexpected number of results in the FIFO (too few or too many).

**Workaround:** Workaround 1: Do not use the abort feature (ICEAn=0).

Workaround 2: Arrange the timing of the CFIFO0 trigger such that it does not assert the trigger at the end of another, lower priority conversion.

Workaround 3: Detect the extra or missing conversion result by checking the EQADC\_CFTCRx (EQADC CFIFO Transfer Counter Register x). This register records how many commands were issued, so it can be used to check that the expected number of results have been received.

**ERR001297: eSCI : reads of the SCI Data Register, which clears the RDRF flag, may cause loss of frame if read occurs during reception of the STOP bit**

**Description:** A received SCI frame is not written into the SCI Data Registers and the Overrun (OR) flag is not set in the SCI Status Register 1 (SCISR1), if:

- 1.) The eSCI has received the last data bit of an SCI frame n
- 2.) and the Receive Data Register Full (RDRF) flag is still set in the SCISR1 after the reception of SCI frame n-1
- 3.) and during the reception of the STOP bit of frame n the host reads the SCI Data Registers, and clears the RDRF flag

In this case the RDRF flag is erroneously set again by the controller instead of the OR flag. Thus, the host reads the data of frame n-1 a second time, and the data of frame n is lost.

**Workaround:** The application should ensure that the data of the foregoing frame is read out from the SCI Data Registers before the last data bit of the actual frame is received.

**ERR009344: eSCI: Late assertion of Transmit Data Ready Interrupt Flag (TXRDY) for Local Interconnect Network (LIN) frame receive (RX) operation**

**Description:** Assertion of the Transmit Data Ready Interrupt Flag (TXRDY) in the Interrupt Flag and Status Register 2 (eSCI\_IFSR2) indicates that data written to the LIN Transmit Register (eSCI\_LTR) has been processed by the eSCI module. For the first three data writes to the eSCI\_LTR during LIN frame generation, the TXRDY flag is asserted one clock cycle after the write access. During LIN RX operation, assertion of the TXRDY flag that coincides with the fourth data write to the eSCI\_LTR is delayed. The TXRDY flag is not asserted until the LIN RX frame has been completely received from the slave device. The TXRDY flag is asserted when the Frame Complete Interrupt (FRC) flag of the eSCI\_IFSR2 register is asserted.

**Workaround:** Application software should expect a delay in the assertion of the TXRDY flag after the fourth data write to the eSCI\_LTR. Instead of expecting TXRDY assertion within one clock cycle of the fourth data write to the eSCI\_LTR, application software should expect assertion of the TXRDY flag after the LIN RX frame has been completely received from the slave device.

## **ERR009001: eSCI: Incorrect behavior while in LIN Standard Bit error detection mode**

**Description:** After a Local Interconnect Network (LIN) wake-up signal frame is transmitted from a master device while in Standard Bit error detection mode (eSCI\_CR2[FBR] = 0), a bit error is detected in any subsequent LIN Transmit (TX) or Receive (RX) frames sent from the master device. After the bit error is detected, the Bit Error Interrupt Flag (eSCI\_IFSR1[BERR]) is asserted, and the LIN controller will not generate TX or RX frames.

**Workaround:** Workaround 1: Reset the LIN Protocol Engine of the eSCI controller by writing 1 and then a 0 to the LIN Protocol Engine Stop and Reset bit in LIN Control Register 1 (eSCI\_LCR1[LRES]) after a complete wake-up frame is sent.

Workaround 2: Use the LIN module in Fast Bit error detection mode, and do not use the Standard Bit error detection mode. Fast Bit Error detection mode can be enabled by writing 1 to the Fast Bit Error Detection bit in Control Register 2 (eSCI\_CR2[FBR] =1).

## **ERR001221: eSCI: LIN bit error indicated at start of transmission after LIN reset**

**Description:** If the eSCI module is in LIN mode and is transmitting a LIN frame, and the application sets and subsequently clears the LIN reset bit (LRES) in the LIN Control register 1 (ESCI\_LCR1), the next LIN frame transmission might incorrectly signal the occurrence of bit errors (ESCI\_IFSR1[BERR]) and frame error (ESCI\_IFSR1[FE]), and the transmitted frame might be incorrect.

**Workaround:** There is no generic work around. The implementation of a suitable workaround is highly dependent on the application and a workaround may not be possible for all applications.

## **ERR001381: eSCI: LIN Wakeup flag set after aborted LIN frame transmission**

**Description:** If the eSCI module is transmitting a LIN frame and the application sets and clears the LIN Finite State Machine Resync bit in the LIN Control Register 1 (eSCI\_LCR1[LRES]) to abort the transmission, the LIN Wakeup Receive Flag in the LIN Status Register may be set (LWAKE=1).

**Workaround:** If the application has triggered LIN Protocol Engine Reset via the eSCI\_LCR1[LRES], it should wait for the duration of a frame and clear the eSCI\_IFSR2[LWAKE] flag before waiting for a wakeup.

## **ERR009361: eSCI: Timing of TXRDY interrupt flag assertion is incorrect for LIN TX Frame**

**Description:** When generating a Local Interconnect Network (LIN) Transmit (TX) Frame, the Transmit Data Ready Interrupt flag (eSCI\_IFSR2[TXRDY]) should assert after the transmission of the Identifier (ID) field. In the TX frame generation, however, the eSCI\_IFSR2[TXRDY] asserts after the Sync field. All subsequent TXRDY Interrupt flags in the current frame assert after each subsequent byte field has been transmitted except for the final TXRDY Interrupt flag. The last TXRDY Interrupt flag asserts after the transmission of the checksum field.

**Workaround:** The timing of the TXRDY Interrupt flag cannot be changed from the incorrect behavior. The incorrect TXRDY Interrupt flag behavior does not affect LIN functionality. Even though the TXRDY Interrupt flag asserts earlier than expected, the TXRDY Interrupt flag still signals that the content of the LIN Transmit Register (eSCI\_LTR) was processed by the LIN Protocol Engine.

### **ERR009797: eSCI: Unable to send next frame after timeout in LIN mode**

**Description:** When generating a Local Interconnect Network (LIN) Transmit (Tx) and Receiver (Rx) frame, the Enhanced Serial Communication Interface (eSCI) module should first send the Header as per the LIN protocol. However, after the Slave Timeout Interrupt Flag (STO) in the Interrupt Flag and Status Register 2 (eSCI\_IFSR2) for the previous LIN Rx Frame is asserted (eSCI\_IFSR2[STO]=1), the eSCI module is not able to generate the next Header, it remains in a suspended state.

**Workaround:** Perform the following actions in this order:

- (1) Set the LIN Protocol Engine Stop and Reset (LRES) control bit to '1' in the LIN Control Register 1 (eSCI\_LCR1).
- (2) Wait until the status bits DACT, WACT, LACT, TACT, and RACT in the Interrupt Flag and Status Register (eSCI\_IFSR1) are cleared to '0'.
- (3) Clear LRES in eSCI\_LCR1 to '0'.
- (4) Begin transmission of the LIN Header for the next frame.

### **ERR005642: ETPU2: Limitations of forced instructions executed via the debug interface**

**Description:** The following limitations apply to forced instructions executed through the Nexus debug interface on the Enhanced Time Processing Unit (ETPU):

- 1- When a branch or dispatch call instruction with the pipeline flush enabled (field FLS=0) is forced (through the debug port), the Return Address Register (RAR) is updated with the current program counter (PC) value, instead of PC value + 1.
- 2- The Channel Interrupt and Data Transfer Requests (CIRC) instruction field is not operational.

**Workaround:** Workaround for limitation #1 (branch or dispatch call instruction):

Increment the PC value stored in the RAR by executing a forced Arithmetic Logic Unit (ALU) instruction after the execution of the branch or dispatch call instruction.

Workaround for limitation #2 (CIRC):

To force an interrupt or DMA request from the debugger:

- 1- Program a Shared Code Memory (SCM) location with an instruction that issues the interrupt and/or DMA request. Note: Save the original value at the SCM location.
- 2- Save the address of the next instruction to be executed.
- 3- Force a jump with flush to the instruction position.
- 4- Single-step the execution.

5- Restore the saved value to the SCM location (saved in step 1).

6- Force a jump with flush to the address of the next instruction to be executed (saved in step 2).

NOTE: This workaround cannot be executed when the eTPU is in HALT\_IDLE state.

#### **ERR002740: ETPU2: Watchdog Status Register (WDSR) may fail to update on channel timeout**

**Description:** The Watchdog Status Register (WDSR) contains a single watchdog status bit for each of the 32 eTPU channels per engine. When this bit is set, it indicates that the corresponding channel encountered a watchdog timeout and was aborted. Under certain conditions the corresponding bit is not set due to a watchdog timeout, and therefore no indication is available as to which channel timed out. However, the global exception is indicated correctly on a per engine basis, and the correct exception is issued to the interrupt controller and may be serviced.

**Workaround:** The application software should treat any watchdog event as a global eTPU exception and handle it in the eTPU global exception handler. Additionally, during the global exception handler the application should check the WDSR and clear any bits that may be set by writing '1' to that bit.

#### **ERR005640: ETPU2: Watchdog timeout may fail in busy length mode**

**Description:** When the Enhanced Time Processing Unit (eTPU) watchdog is programmed for busy length mode (eTPU Watchdog Timer Register (ETPU\_WDTR) Watchdog Mode field (WDM) = 3), a watchdog timeout will not be detected if all of the conditions below are met:

1- The watchdog timeout occurs at the time slot transition, at the first instruction of a thread, or at the thread gap. (a thread gap is a 1 microcycle period between threads that service the same channel).

2- The thread has only one instruction.

3- The eTPU goes idle right after the timed-out thread, or after consecutive single-instruction threads.

**Workaround:** Insert a NOP instruction in threads which have only one instruction.

#### **ERR008194: eTPU: EAC may detect double teeth in a single input transition**

**Description:** The eTPU Enhanced Angle Counter (EAC) may detect two consecutive teeth in a single tooth input transition, when the microengine Tooth Program Register (TPR) register bit HOLD=1.

As a consequence of the input transition, the EAC:

(1) resets HOLD, which is correct, then

(2) detects another tooth (incorrect), so that if it is in normal mode, it goes to high-rate mode, and if it is in halt mode, it goes to normal mode.

No problem occurs if the EAC was in high-rate mode when HOLD=1.

The problem occurs only if both of these configuration conditions are true:

- (a) EAC is configured with the eTPU Time Base Configuration Register (ETPU\_TBCR\_ENGx) Angle Mode Selection (AM) field = 2 (channel 1) or AM = 3 (channel 2).
- (b) Channel filter configuration with etpu Engine Control Register (ETPU\_ECR\_ENGx) Channel Digital Filter Control (CDFC) field = 1 (bypass) or ETPU\_ECR\_ENGx Filter Clock Source Selection (FCSS) field = 1 (eTPU clock as filter clock).

**Workaround:** Configure the channel filters to use any mode except bypass (ETPU\_ECR\_ENGx field CDFC ! = 0b01) and configure ETPU\_ECR\_ENGx field FCSS = 0. (CDFC should be set to 0b00, 0b10, or 0b11.)

### **ERR008252: 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.

### **ERR009090: 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).

## **ERR009809: 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

## **ERR051054: eTPU: MRL Branch Condition at the start of the thread may differ from the actual MRL state if channel runs at full speed**

**Description:** At the start of a thread, the Match Recognition Latch (MRL) Branch Condition may differ from the actual value of the MRL. Thus, inconsistency between the executed code and the actual MRL state may occur.

Conditions that together lead to this inconsistency are:

- 1.) eTPU channels run at full speed (refer to \* )
- 2.) a Time Slot Transition (TST) occurs
- 3.) the selected thread, that will be executed after the TST, has the Match Enabled (ME) flag disabled in the Entry Point Information: ME=0
- 4.) A match occurs during the TST phase 1 (TST1). Refer to eTPU reference manual for exact timing description.

In this case the MRL flag is set at the channel logic, although it had not been sampled into the thread's Branch Conditions.

- eTPU channel logic runs at full speed when any of the following conditions are true:

1) the Timer Count Register 1 Clock Source bit (TCR1CS) in the eTPU Time Base Configuration Register (ETPUTBCR) is set, selecting the system clock as the input to the TCR1 prescaler:  $ETPUTBCR[TCR1CS] = 1$

2) the Filter Clock Source Selection bit (FCSS) in the ETPUECR is set, selecting the system clock as the input to the Enhanced Digital Filter (EDF) clock prescaler:  $ETPUECR[FCSS] = 1$

**Workaround:** No workaround is available for the specific conditions (ME=0 and channel in full speed mode). If possible, turn ME=1 or do not run channels at full speed.

## **ERR003114: FLASH: Erroneous update of the ADR register in case of multiple ECC errors**

**Description:** An erroneous update of the Address register (ADR) occurs whenever there is a sequence of 3 or more events affecting the ADR (ECC single or double bit errors or RWW error) and both the following conditions apply:

- The priorities are ordered in such a way that only the first event should update ADR.
- The last event although it does not update ADR sets the Read While Write Event Error (RWE) or the ECC Data Correction (EDC) in the Module Configuration Register (MCR).

For this case the ADR is wrongly updated with the address related to one of the intervening events.

Example – If a sequence of two double-bit ECC errors is followed by a single-bit correction without clearing the ECC Event Error flag (EER) in the MCR, then the value found in ADR after the single-bit correction event is the one related to the second double-bit error (instead of the first one, as specified)

**Workaround:** Always process Flash ECC errors as soon as they are detected.

Clear MCR[RWE] at the end of each flash operation (Program, Erase, Array Integrity Check, etc...).

### **ERR003196: FLASH: PFCR3 is not directly writable**

**Description:** The Flash Configuration Register 3 (PFCR3) that can control the prefetching settings (Data Prefetch Enable [DPFEN], Instruction Prefetch Enable [IPFEN], Prefetch Limit [PFLIM], and Buffer Enable [BFEN]) of the Bank 1 (array 1 and array 2) flash modules is not directly writable. These settings are enabled by setting the Global Configuration Enable bit in the Flash Bus Interface Unit Control register (BIUCR).

**Workaround:** Set the GCE bit (BIUCR[GCE=1]) to allow the Bank 0, Array 0 prefetch settings to also control bank 1 (Array 1 and Array 2); or program a default value for the PFCR3 register that gets loaded into the register at reset into the Flash Shadow block at address 0x00FF\_FE08.

### **ERR005498: Flash: Prefetch during program/erase operation causes system bus stop**

**Description:** While performing a program/erase sequence on one flash bank, prefetches from the other flash bank may cause the system bus to stop.

**Workaround:** Before initiating any flash program/erase sequence, clear flash the flash prefetch buffers by clearing the PFLASH Line Read Buffer Enable bit in the Flash Bus Interface Unit Configuration Register (FLASH\_BIUCR1[BFEN]). After the program/erase sequence is complete, software can re-enable the prefetch buffers by setting FLASH\_BIUCR[BFEN].

### **ERR007322: FlexCAN: Bus Off Interrupt bit is erroneously asserted when soft reset is performed while FlexCAN is in Bus Off state**

**Description:** Under normal operation, when FlexCAN enters in Bus Off state, a Bus Off Interrupt is issued to the CPU if the Bus Off Mask bit (CTRL[BOFF\_MSK]) in the Control Register is set. In consequence, the CPU services the interrupt and clears the ESR[BOFF\_INT] flag in the Error and Status Register to turn off the Bus Off Interrupt.

In continuation, if the CPU performs a soft reset after servicing the bus off interrupt request, by either requesting a global soft reset or by asserting the MCR[SOFT\_RST] bit in the Module Configuration Register, once MCR[SOFT\_RST] bit transitions from 1 to 0 to acknowledge the soft reset completion, the ESR[BOFF\_INT] flag (and therefore the Bus Off Interrupt) is re-asserted.

The defect under consideration is the erroneous value of Bus Off flag after soft reset under the scenario described in the previous paragraph.

The Fault Confinement State (ESR[FLT\_CONF] bit field in the Error and Status Register) changes from 0b11 to 0b00 by the soft reset, but gets back to 0b11 again for a short period, resuming after certain time to the expected Error Active state (0b00). However, this late correct state does not reflect the correct ESR[BOFF\_INT] flag which stays in a wrong value and in consequence may trigger a new interrupt service.

**Workaround:** To prevent the occurrence of the erroneous Bus Off flag (and eventual Bus Off Interrupt) the following soft reset procedure must be used:

1. Clear CTRL[BOFF\_MSK] bit in the Control Register (optional step in case the Bus Off Interrupt is enabled).

2. Set MCR[SOFT\_RST] bit in the Module Configuration Register.
3. Poll MCR[SOFT\_RST] bit in the Module Configuration Register until this bit is cleared.
4. Wait for 4 peripheral clocks.
5. Poll ESR[FLTCNF] bit in the Error and Status Register until this field is equal to 0b00.
6. Write “1” to clear the ESR[BOFF\_INT] bit in the Error and Status Register.
7. Set CTRL[BOFF\_MSK] bit in the Control Register (optional step in case the Bus Off Interrupt is enabled).

**ERR003407: FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1**

**Description:** FlexCAN does not transmit an expected message when the same node detects an incoming Remote Request message asking for any remote answer.

The issue happens when two specific conditions occur:

- 1) The Message Buffer (MB) configured for remote answer (with code “a”) is the last MB. The last MB is specified by Maximum MB field in the Module Configuration Register (MCR[MAXMB] ).
- 2) The incoming Remote Request message does not match its ID against the last MB ID.

While an incoming Remote Request message is being received, the FlexCAN also scans the transmit (Tx) MBs to select the one with the higher priority for the next bus arbitration. It is expected that by the Intermission field it ends up with a selected candidate (winner). The coincidence of conditions (1) and (2) above creates an internal corner case that cancels the Tx winner and therefore no message will be selected for transmission in the next frame. This gives the appearance that the FlexCAN transmitter is stalled or “stops transmitting”.

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

There is NO ISSUE if any of the conditions below holds:

- a) The incoming message matches the remote answer MB with code “a”.
- b) The MB configured as remote answer with code “a” is not the last one.
- c) Any MB (despite of being Tx or Rx) is reconfigured (by writing its CS field) just after the Intermission field.
- d) A new incoming message sent by any external node starts just after the Intermission field.

**Workaround:** Do not configure the last MB as a Remote Answer (with code “a”).

**ERR002379: FMPLL: Loss-of-clock detection may cause unexpected reset**

**Description:** An unexpected Loss-Of-Clock (LOC) event may occur in the following scenario:

1. The FMPLL is initially powered down in bypass mode.
2. The FMPLL is then powered on (still in bypass mode).
3. The LOCK bit of the SYNSR register is polled to determine when the FMPLL is ready.
4. After the LOCK flag becomes set, the FMPLL is switched to normal mode.

5. Loss-of-clock detection is enabled by setting the LOCEN bit of the Enhanced Synthesizer Control Register 2 (ESYNCR2), either before or immediately after switching to normal mode.

The unexpected LOC event will activate the backup clock switching feature, causing the reference clock to be selected as the system clock. If LOC reset was also enabled by setting the LOCRE bit in the ESYNCR2 register, a system reset will occur.

The reason for the unexpected LOC event is that the time it takes for the Clock Quality Monitor (CQM) to detect a valid FMPLL clock is typically larger than the time it takes for the FMPLL to lock. Polling the LOC flag does not help because (the way it is defined) it does not flag LOC in bypass mode.

This issue only occurs when the FMPLL is turned off and then on again without going through a reset cycle. Immediately following reset, the issue can not occur because the CQM keeps the part in reset until it detects a valid crystal clock with plenty of time to detect a valid FMPLL clock.

**Workaround:** Any time the FMPLL is powered down, wait for 600us before activating the loss-of-clock function.

If the intent is just to re-program the FMPLL, it is not required to turn it off. FMPLL settings can be changed on the fly, and then the CQM will never indicate loss-of-clock.

#### **ERR003159: MPC563xM/SPC563M: MIDR MASKNUM field is set to 0x21**

**Description:** The mask revision field (MASKNUM[Major, Minor]) of the MCU Identification Register is 0b0010\_0001 (0x21).

**Workaround:** Expect that the MASKNUM fields of the MIDR register will change in the future.

#### **ERR007590: MPC563xM: Incorrect JTAG ID[MIC] and MIDR[S\_F] register values**

**Description:** The Manufacturer's Identification Code (MIC) in the JTAG Device Identification Register and the S\_F (manufacturer) bit of the System Integration Unit (SIU) Microcontroller Identification Register 2 (SIU\_MIDR2) register may indicate either Freescale or ST randomly on same device. Four bits of the JTAG ID (bits 2, 3, 4, and 6) which are part of the MIC field (bits 11:1 of the JTAG ID) may read as either a 1 or a 0 on each read of the register. Likewise, the S\_F bit of the (SIU) Microcontroller Identification Register 2 (SIU\_MIDR2) register may indicate either Freescale (0b0) or ST (0b1).

**Workaround:** Expect that the MSB of the MIDR2 could indicate either Freescale or ST as the manufacturer of the MCU. In addition, expect that 4 of the MIC bits in the JTAG ID will read 0b000\_00? 0\_???0 (the ? bits could be either a 0b0 or a 0b1). All other bits of the JTAG ID and the MIDR2 register will always read correct values. To differentiate between Freescale or ST manufactured material, read the marking on the top of the package.

#### **ERR003205: NEXUS: EVTI not functional on QFP176 and BGA208 packages**

**Description:** Event In (EVTI) is an input that is read on the negation of TRST (or JCOMP) to enable (if asserted) or disable (if deasserted) the Nexus Debug port.

After reset, EVTI is an input which, when asserted, will initiate one of two events based on the EIC (EVTI Control) bits in the DC1 (Development Control 1) Register (if the Nexus Class 2+ module is enabled at reset):

- 1) Program Trace and Data Trace synchronization messages (provided Program Trace and EIC = 0b00).
- 2) Debug request to e200z335 Nexus Class 1 module (provided EIC = 0b01 and this feature is implemented).

**Workaround:** 1) Do not expect Program Trace Sync messages after EVTI assertion. Other condition for the sync messaging are not impacted.

2) Do not use EVTI to request the CPU to enter the debug state. Other requests are functional.

In case EVTI functionality is needed, CSP496 package can also be used to emulate the 176QFP or the BGA208 packages.

### **ERR006726: NPC: MCKO clock may be gated one clock period early when MCKO frequency is programmed as SYS\_CLK/8 and gating is enabled**

**Description:** The Nexus auxiliary message clock (MCKO) may be gated one clock period early when the MCKO frequency is programmed as SYS\_CLK/8 in the Nexus Port Controller Port Configuration Register (NPC\_PCR[MCKO\_DIV]=111) and the MCKO gating function is enabled (NPC\_PCR[MCKO\_GT]=1). In this case, the last MCKO received by the tool prior to the gating will correspond to the END\_MESSAGE state. The tool will not receive an MCKO to indicate the transition to the IDLE state, even though the NPC will transition to the IDLE state internally. Upon re-enabling of MCKO, the first MCKO edge will drive the Message Start/End Output (MSEO=11) and move the tool's state to IDLE.

**Workaround:** Expect to receive the MCKO edge corresponding to the IDLE state upon re-enabling of MCKO after MCKO has been gated.

### **ERR002338: Pad Ring: Leakage if VDDE is greater than VDD33**

**Description:** If the VDDEx supplies (provided by an external supply) are greater than the VDD33 supplies (provided by the internal regulator), leakage current can occur through all pins powered by VDDEx from the VDDEx supply on the pad output driver through the pad towards ground. The highest leakage current occurs at high temperatures and is exponentially proportional to the VDDEx-VDD33 differential. Worst case leakage, per grounded pad at 150C is 29uA with a 200mV differential, and 590uA with a 400mV differential in the VDDEx-VDD33.

Any I/O configured as an input with the weak pull down enabled will rise towards VDDE level as the VDDE-VDD33 voltage differential increases (as the leakage current exceeds the weak pull-down capability). The reset state of most Nexus pads is pull-down, so this would not be guaranteed. EVTI is pulled up internally during and after RESET. EVTO must be pulled low externally for Auto-baud rate detection. I/O pads configured as outputs driving LOW will remain below VOL level but will consume the leakage current through the pad driver. External logic driving pads configured as inputs will have to sink this leakage current when driving LOW.

**Workaround:** Maintain a VDDE-VDD33 voltage difference below 200mV. If VDDE is greater than 3.45V, the PMC\_TRIMR[VDD33TRIM] for the internal regulator can be increased to 4 steps above typical (0b1011) to increase VDD33 default voltage by a nominal value of 120mV.

## **ERR003377: Pad Ring:Nexus pins may drive an unknown value immediately after power up but before the 1st clock edge**

**Description:** The Nexus Output pins (Message Data outputs 0:15 [MDO] and Message Start/End outputs 0:1 [MSEO]) may drive an unknown value (high or low) immediately after power up but before the 1st clock edge propagates through the device (instead of being weakly pulled low). This may cause high currents if the pins are tied directly to a supply/ground or any low resistance driver (when used as a general purpose input [GPI] in the application).

**Workaround:** 1. Do not tie the Nexus output pins directly to ground or a power supply.  
2. If these pins are used as GPI, limit the current to the ability of the regulator supply to guarantee correct start up of the power supply. Each pin may draw upwards of 150mA.  
If not used, the pins may be left unconnected.

## **ERR009109: PAD\_RING: Output pulse may occur on analog inputs during power on reset**

**Description:** If the 1.2V core supply voltage (VDD) power supply is the last power supply to ramp into operating voltage (after the Analog to Digital [ADC] converter [VDDA] and other supplies [VDDEn, VDDEHn] are powered) or is the first supply to go out of the operating specified voltage, it is possible that an output pulse will occur on an analog pin (ANx) of the device. This pulse could be a maximum of 3.8 volts (unloaded). If the ANx pin is grounded, a current of up to 2 mA could be seen on the ANx pin.

This pulse could occur on up to 2 pins per enhanced Queued Analog to Digital Converter (eQADC) or 2 pairs of differential analog inputs. The pulse occurs if one of the ADC channels is selected to be connected to one of the two ADCs (ADC\_0 or ADC\_1) in the eQADC.

The two ANx pins selected could be random over the complete specified device processing, temperature and voltage ranges, and VDD voltage ramp speed. The same channel could be selected to both ADCs for a total current of 4 mA (unloaded voltage remains a maximum of 3.8V).

On devices with ANx channels shared between multiple eQADC modules (currently, the maximum number of eQADC modules on a device is 2), it is theoretically possible for a channel to be selected to all ADC's (up to 4) on the device for a total current of 8 mA.

The pulse may start (rising edge) when VDD reaches 700 mv and go to a high impedance when VDD reaches 1.166 volts. The pulse width could be the time that VDD ramps between these voltages.

**Workaround:** Design circuitry connected to the analog pins to withstand a possibility of up to 4 mA (or 8 mA for shared analog pins) during power up and power down.

## **ERR011321: PIT\_RTI: Generates false RTI interrupt on re-enabling**

**Description:** A false Real-Time Interrupt (RTI) may be observed when the RTI module is re-enabled if, after servicing an RTI interrupt (by clearing TFLGn[TIF]), the clocks to the RTI module are disabled.

This occurs only if the RTI module clock is disabled within four RTI clock cycles of an RTI Interrupt being cleared.

**Workaround:** Option 1: The user should check the RTI interrupt flag, TFLGn[TIF] before servicing the interrupt, this flag won't be set for the false/spurious interrupts.

Option 2: Ensure that the module clock to the RTI module is not disabled within four RTI clock cycles after servicing an RTI interrupt. Consult the chip-specific documentation to determine the clock period of the RTI module and implement a time delay of at least five times this period before disabling the RTI module clock.

#### **ERR003425: PMC: 5V VDDREG POR De-assertion Max Level 4.2V**

**Description:** 5V Voltage regulator input (VDDREG) power on reset (POR) de-assertion maximum specification level is now 4.2V. Previously, the maximum specification level was 4.005V.

**Workaround:** Expect that 4.2V is required on the VDDREG input before the device will exit from a power on reset. The POR levels for VDDSYN and VDD in the data sheet are unchanged.

#### **ERR003221: PMC: SRAM standby power low voltage detect circuit is not accurate**

**Description:** The power management controller (PMC) SRAM standby voltage low power detect circuit cannot reliably detect the brown-out condition if the standby supply is below 1.0 volts. The Status Register Brown Out Flag (PMC.SR[LVFSTBY]) bit may not be set during a brownout condition of the SRAM standby voltage or may be set even though no data has been lost.

**Workaround:** The application software should not rely on the PMC.SR[LVFSTBY] bit to detect corrupted SRAM values.

#### **ERR009682: 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 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.



---

**ERR001421: SWT: switching SWT to system clock has very small chance of causing the SWT to enter an indeterminate state**

**Description:** The reset value for the clock source of the Software Watchdog Timer module (SWT) is the oscillator clock. If the clock source is switched to the system clock by clearing Clock Selection bit in the SWT Module Control Register (SWT\_MCR[CSL]=0), then the SWT has a very small chance of entering an indeterminate state.

**Workaround:** Only use the oscillator clock as the SWT clock source.

**How to Reach Us:****Home Page:**[nxp.com](http://nxp.com)**Web Support:**[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro,  $\mu$ Vision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2022 NXP B.V.

