Mask Set Errata for Mask 2N60C

This report applies to mask 2N60C for these products:

- MPC5642A

Mask Specific Information

Major mask revision number	0x1
Minor mask revision number	0x0
JTAG identifier	0x1AE0_301D

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
ERR051064	CRC: AutoSAR 4.0 8-bit CRC8 0x2F is not supported in hardware
ERR005037	CRC: CRC-32 (Ethernet) and CRC-16 (CCITT) operation do not match industry standards.
ERR008251	DECFIL: timestamp may be lost in edge trigger mode
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
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
ERR006967	eDMA: Possible misbehavior of a preempted channel when using continuous link mode
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
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
ERR004480	eQADC: Differential conversions with 4x gain may halt command processing

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
ERR003378	EQADC: Pull devices on differential pins may be enabled for a short period of time during and just after POR
ERR005086	eQADC: unexpected result may be pushed when Immediate Conversion Command is enabled
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
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
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
ERR050807	FLASH: Flash array access may be incorrect after power up
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
ERR002424	FlexCAN: switching CAN protocol interface (CPI) to system clock has very small chance of causing the CPI to enter an indeterminate state
ERR008770	FlexRAY: Missing TX frames on Channel B when in dual channel mode and Channel A is disabled
ERR008340	NPC: EVTO_B toggles instead of remaining asserted when used by the DTS if Nexus is not enabled
ERR006726	NPC: MCKO clock may be gated one clock period early when MCKO frequency is programmed as SYS_CLK/8.and gating is enabled
ERR006481	NZ4C3/NZ7C3: Erroneous Resource Full Message under certain conditions
ERR007120	NZxC3: DQTAG implemented as variable length field in DQM message
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
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

Table 2. Revision History

Revision	Changes
Sept 2018	August 2017 The following errata were added. e5037, e5086, e5640, e5642, e6026, e6481, e6726, e6967, e7120, e7322, e7352, e8194, e8251, e8252,

Table continues on the next page...

Table 2. Revision History (continued)

Revision	Changes
	e8340, e8770, e9001, e9090, e9109, e9344, e9361, e9682, e9797, e9809, e9976, e9978, e10755 --- September 2018 The following errata were added. e11293, e11295, e11294, e11321, e11235
AUG2021	The following errata were added. • ERR002424 • ERR050782 • ERR050807 • ERR051064 The following erratum was revised. • ERR011235

ERR051064: CRC: AutoSAR 4.0 8-bit CRC8 0x2F is not supported in hardware

Description: The Cyclic Redundancy Check (CRC) module does not implement the 8-bit CRC-8-H2F required to support the Autosar 4.0 specification. The CRC-8-H2F uses a polynomial generator seed of 0x2F and an equation of $x^5 + x^3 + x^2+x+1$.

Workaround: The 8-bit CRC-8-H2F function must be written in software to support AutoSAR 4.0.

ERR005037: CRC: CRC-32 (Ethernet) and CRC-16 (CCITT) operation do not match industry standards.

Description: CRC-32 (Ethernet) and CRC-16 (CCITT) do not calculate CRCs according to industry standards. For CRC-32, the CRC engine expects the CRC Input to be byte swapped. For CRC-16, the CRC engine expects the CRC Input to be bit reversed.

Workaround: The input data has to be bit reversed and entered into the CRC Input Register (CRC_INP) for CRC-16 calculations and byte reversed for CRC-32 calculations to get the correct CRC signature.

For CRC-32, given input data of 0xABCDEF98, the user software must enter 0x98EFCDAB in CRC_INP register to get the correct result for CRC-32. Also, the Inversion (INV) and Swap selection (SWAP) bits of the CRC Configuration Register (CRC_CFG) are to be set to 1 for CRC-32 polynomial calculations along with the byte swaps.

For CRC-16, given input data of 0xABDEF98, the user software must enter 0x19F7B3D5 into the CRC_INP register. INV and SWAP bits are programmed to zero for CRC-16 operation.

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 DECFILT 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 (PSI0 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.

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.

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/RFDF]). 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_blt loop
```

ERR006967: eDMA: Possible misbehavior of a preempted channel when using continuous link mode

Description: When using Direct Memory Access (DMA) continuous link mode Control Register Continuous Link Mode (DMA_CR[CLM]) = 1) with a high priority channel linking to itself, if the high priority channel preempts a lower priority channel on the cycle before its last read/write sequence, the counters for the preempted channel (the lower priority channel) are corrupted. When the preempted channel is restored, it continues to transfer data past its “done” point (that is the byte transfer counter wraps past zero and it transfers more data than indicated by the byte transfer count (NBYTES)) instead of performing a single read/write sequence and retiring.

The preempting channel (the higher priority channel) will execute as expected.

Workaround: Disable continuous link mode (DMA_CR[CLM]=0) if a high priority channel is using minor loop channel linking to itself and preemption is enabled. The second activation of the preempting channel will experience the normal startup latency (one read/write sequence + startup) instead of the shortened latency (startup only) provided by continuous link mode.

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.

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 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.

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.

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.

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).

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.

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

ERR050807: FLASH: Flash array access may be incorrect after power up

Description: After power up, a flash read may return incorrect data or a program/erase operation may fail. The condition will persist from power up until the next reset assertion.

Workaround: Flash read: Software Watchdog Timer (SWT) needs to be enabled at the time the micro exits its reset sequence after power up which provides a recovery mechanism if code execution fails.

Flash program/erase: After power up, the device must be reset before performing flash program/erase operations. However, through proper usage of software mechanisms such as EE emulation algorithms in addition to Error Correction Code (ECC), the probability of observing a failure due to ineffective programing or erasing will be greatly reduced.

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[FLTCONF] 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”).

ERR002424: FlexCAN: switching CAN protocol interface (CPI) to system clock has very small chance of causing the CPI to enter an indeterminate state

Description: The reset value for the clock source of the CAN protocol interface (CPI) is the oscillator clock. If the CPI clock source is switched to the system clock while the FlexCAN is not in freeze mode, then the CPI has a very small chance of entering an indeterminate state.

Workaround: Switch the clock source while the FlexCAN is in a halted state by setting HALT bit in the FlexCAN Module Configuration Register (CANx_MCR[HALT]=1). If the write to the CAN Control Register to change the clock source (CANx_CR[CLK_SRC]=1) is done in the same oscillator clock period as the write to CANx_MCR[HALT], then chance of the CPI entering an indeterminate state is extremely small. If those writes are done on different oscillator clock periods, then the corruption is impossible. Even if the writes happen back-to-back, as long as the system clock to oscillator clock frequency ratio is less than three, then the writes will happen on different oscillator clock periods.

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

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

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

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

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

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

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

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.

ERR006481: NZ4C3/NZ7C3: Erroneous Resource Full Message under certain conditions

Description: The e200zx core Nexus interface may transmit an erroneous Resource Full Message (RFM) following a Program Trace history message with a full history buffer. The History information for both of the messages are the same and the RFM should have not been transmitted. This occurs when the instruction following the indirect branch instruction (which triggers the Program Trace History message) would have incremented the History field. The instructions must be executed in back to back cycles for this problem to occur. This is the only case that cases this condition.

Workaround: There are three possible workarounds for this issue.

(1) Tools can check to see if the Program Trace History message and proceeding Resource Full Message (RFM) have the same history information. If the history is the same, then the RFM is extraneous and can be ignored.

(2) Code can be rewritten to avoid the History Resource Full Messages at certain parts of the code. Insert 2 NOP instructions between problematic code. Or inserting an “isync” or a indirect branch some where in the code sequence to breakup/change the flow.

(3) If possible, use Traditional Program Trace mode can be used to avoid the issue completely. However, depending on other conditions (Nexus port width, Nexus Port speed, and other enabled trace types), overflows of the port could occur.

ERR007120: NZxC3: DQTAG implemented as variable length field in DQM message

Description: The e200zx core implements the Data Tag (DQTAG) field of the Nexus Data Acquisition Message (DQM) as a variable length packet instead of an 8-bit fixed length packet. This may result in an extra clock (“beat”) in the DQM trace message depending on the Nexus port width selected for the device.

Workaround: Tools should decode the DQTAG field as a variable length packet instead of a fixed length packet.

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.

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.

How to Reach Us:

Home Page:
nxp.com

Web Support:
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.

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.