Mask Set Errata for Mask 0N15P

This report applies to mask 0N15P for these products:

- MPC574xP

Mask Specific Information

Major mask revision number	1
Minor mask revision number	2
JTAG identifier	0x29B4501D

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
e6407	e200zx: Circular Addressing issue on LSP Load/Store instructions, and zcircinc instruction
e7259	e200zx: ICNT and branch history information may be incorrect following a nexus overflow
e7305	e200zx: JTAG reads of the Performance Monitor Counter registers are not reliable
e6358	ENET: Write to Transmit Descriptor Active Register (ENET_TDAR) is ignored
e7099	FCCU: Error pin signal length is not extended when the next enabled fault, with its alarm timeout disabled, occurs
e7227	FCCU: FCCU Output Supervision Unit (FOSU) will not monitor faults enabled while already pending
e7869	FCCU: FOSU monitoring of a fault is blocked for second or later occurrence of the same fault
e8770	FlexRAY: Missing TX frames on Channel B when in dual channel mode and Channel A is disabled
e7274	LINFlexD: Consecutive headers received by LIN Slave triggers the LIN FSM to an unexpected state
e8933	LINFlexD: Inconsistent sync field may cause an incorrect baud rate and Sync Field Error Flag may not be set
e8970	LINFlexD: Spurious bit error in extended frame mode may cause an incorrect Idle State
e8228	MC_ME: Wakeup from STOP mode may lead to a system hang scenario
e8049	MPC574xP: Current injection causes leakage path across the LFAST LVDS pins
e7204	SENT: Number of Expected Edges Error status flag spuriously set when operating with Option 1 of the Successive Calibration Check method
e7425	SENT: Unexpected NUM_EDGES_ERR error in certain conditions when message has a pause pulse
e8967	TSENS: (MPC5744P) Temperature sensor flag glitch during power up
e8683	TSENS: Temperature sensor status output bits in PMC_ESR_TD register shows indeterminate behavior

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
e7236	XBIC: XBIC may trigger false FCCU alarm
e8730	XBIC: XBIC may store incorrect fault information when a fault occurs

Table 2. Revision History

Revision	Changes
APR 2015	Initial revision
Apr 23 2015	The following errata were added. • e7274
May 2015	The following errata were revised. • e8049

e6407: e200zx: Circular Addressing issue on LSP Load/Store instructions, and zcircinc instruction

Description: The circular addressing mode of the e200zx Lightweight Signal Processing (LSP) Auxiliary Processing Unit for the circular increment (zcircinc) and Load/Store (zl*, zs*) instructions do not wrap properly in some cases when using positive offset.

Workaround: Use one of the following options to workaround the issue.

1. Always use negative offset with these instructionss;

or

2. For a small buffer size of 1, 2, 3, or 4 double-words (8,16,24, or 32 bytes), a positive offset can be emulated by using a negative offset value equal to the “buffer length in bytes - desired_positive_offset”. An example for a buffer length of 2 double-words with a desired offset of 2 bytes, an offset of 2-16=-14 can be used;

or

3. Use ODD index and EVEN positive offset greater than 1.

e7259: e200zx: ICNT and branch history information may be incorrect following a nexus overflow

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

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

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

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

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

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

e7305: e200zx: JTAG reads of the Performance Monitor Counter registers are not reliable

Description: Reads of the Performance Monitor Counter (PMC0, PMC1, PMC2, and PMC4) registers through the IEEE 1149.1 or IEEE 1149.7 (JTAG) interfaces may return occasional corrupted values.

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

e6358: ENET: Write to Transmit Descriptor Active Register (ENET_TDAR) is ignored

Description: If the ready bit in the transmit buffer descriptor (TxBD[R]) is previously detected as not set during a prior frame transmission, then the ENET_TDAR[TDAR] bit is cleared at a later time, even if additional TxBDs were added to the ring and the ENET_TDAR[TDAR] bit is set. This results in frames not being transmitted until there is a 0-to-1 transition on ENET_TDAR[TDAR].

Workaround: Code can use the transmit frame interrupt flag (ENET_EIR[TXF]) as a method to detect whether the ENET has completed transmission and the ENET_TDAR[TDAR] has been cleared. If ENET_TDAR[TDAR] is detected as cleared when packets are queued and waiting for transmit, then a write to the TDAR bit will restart TxBD processing.

e7099: FCCU: Error pin signal length is not extended when the next enabled fault, with its alarm timeout disabled, occurs

Description: In the Fault Collection and Control Unit (FCCU), when the following conditions are met:

- two faults occur
- the second fault arrises with a delay (T_delay) from the first error

- the second fault has its alarm timeout disabled
- T_delay is lower than the FCCU error pin minimum active time (T_min, defined in the Delta T register (FCCU_DELTA_T))

Then the error output signal is not extended and its duration is only T_min, if the faults are cleared before the timer expires.

The expected behavior is to have the error output signal duration of T_min + T_delay, if the faults are cleared before the timer expires.

Workaround: Take into account that the error out signal duration will only be T_min, if the faults are cleared before the timer expires.

The timer count is meaningful only when the Error pin is driven low, which can be checked by reading the pin status FCCU_STAT[ESTAT].

e7227: FCCU: FCCU Output Supervision Unit (FOSU) will not monitor faults enabled while already pending

Description: The Fault Collection and Control Unit (FCCU) Output Supervision Unit (FOSU) will not monitor the FCCU reaction to fault inputs that are enabled with an already pending notification.

The FOSU monitoring is triggered by an edge from a fault input. The edge detection will be blocked in following cases:

1) When a fault input is disabled in the FCCU and a fault occurs,

2) When a fault input is enabled in the FCCU and a fault occurs in the CONFIG state. FOSU edge detection remains blocked until it gets initialized by a FCCU reaction or a destructive reset.

Workaround: Apply the following procedure when enabling fault inputs in the FCCU in order to ensure correct monitoring by the FOSU:

1) Check for FCCU pending faults and clear them.

2) Configure the FCCU as desired. In addition enable fault input for interrupt reaction (software recovery mode) to an injected error on this input.

3) Immediately on exiting the CONFIG state, check for FCCU pending faults. If there is a fault status set then initiate a destructive reset.

4) Clear the FOSU status by injecting a fault on the FCCU fault input configured for software recovery mode. This will generate a FCCU reaction that will clear the FOSU edge detection logic.

Apply the following procedure when exiting FCCU CONFIG state in order to ensure correct monitoring by the FOSU:

- Check for FCCU pending faults. If there is a fault status set then initiate a destructive reset.

e7869: FCCU: FOSU monitoring of a fault is blocked for second or later occurrence of the same fault

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

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

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

Workaround: There are two possible workarounds. Either one can be used with same effectiveness.

1. In addition to the reset reaction, enable either the interrupt (IRQ) or Non-maskable Interrupt (NMI) or error out signaling reaction for the faults that have a reset reaction enabled.

2. Apply the following procedure during the FCCU configuration after a reset and in the fault service routine while clearing the fault status inside the FCCU.

i. Check for FCCU pending faults and clear them.

ii. Configure the FCCU as desired.

iii. Enable a fault as software recoverable by setting its corresponding bit in the NCF Configuration Register (FCCU_NCF_CFGn)

iv. Inject a fake fault to the fault set up in step “iii” by writing the corresponding code into the NCF Fake Register (FCCU_NCFF)

v. Check that there are no pending faults else clear the pending faults and repeat steps “iv” and “v”

vi. Reconfigure the fault that was configured for software recovery mode.

e8770: 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.

e7274: LINFlexD: Consecutive headers received by LIN Slave triggers the LIN FSM to an unexpected state

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

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

Workaround: The following three steps should be followed -

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

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

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

Note:

THeader_Nominal = 34 * TBit

TResponse_Nominal = 10 * (NData + 1) * TBit

THeader_Maximum = 1.4 * THeader_Nominal

TResponse_Maximum = 1.4 * TResponse_Nominal

TFrame_Maximum = THeader_Maximum + TResponse_Maximum

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

e8933: LINFlexD: Inconsistent sync field may cause an incorrect baud rate and Sync Field Error Flag may not be set

Description: When the LINFlexD module is configured as follows:

- LIN (Local interconnect network) slave mode is enabled by clearing the Master Mode Enable (MME) bit in the LIN Control Register 1 (LINCR1)
- Auto synchronization is enabled by setting the LIN Auto Synchronization Enable bit (LASE) in the LINCR1 register
- Sync Field value is not equal to 0x55

the LINFlexD module may automatically synchronize to an incorrect baud rate without setting the Sync Field Error Flag (SFEF) in the LIN Error Status register (LINESR).

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

Workaround: There are 2 possible workarounds.

Workaround 1:

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

1. Set the LIN state Interrupt enable bit (LSIE) in the LIN Interrupt Enable register (LINIER) [LINIER[LSIE] = 0x1]

2. When the Data Reception Completed Flag (DRF) get set in the LIN Status Register (LINSR), read the LIN State field (LINS) in LINSR

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

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

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

1. Set the LSIE bit in the LINIER register

2. When the LINSR[DRF] bit get set in the LIN Status Register (LINSR), read the LINSR[LINS] field

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

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

5. Wait for LINSR[DRF] to get set again and read LINSR[LINS] field

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

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

Workaround 2: Do not use the auto synchronization feature (by clearing LINCR1[LASE]=0) in LIN slave mode.

e8970: LINFlexD: Spurious bit error in extended frame mode may cause an incorrect Idle State

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

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

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

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

e8228: MC_ME: Wakeup from STOP mode may lead to a system hang scenario

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

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

e8049: MPC574xP: Current injection causes leakage path across the LFAST LVDS pins

Description: The General Purpose Input/Output (GPIO) digital pins (including all digital CMOS input or output functions of the pin) connected to the differential LVDS drivers of the LVDS Fast Asynchronous Serial Transmit Interface (LFAST) do not meet the current injection specification given in the operating conditions of the device electrical specification. When the LVDS transmitter or receiver is disabled and current is positively or negatively injected into one pin of the GPIO pins connected to the differential pair, a leakage path across the internal termination resistor of the receiver or through the output driver occurs, potentially corrupting data on the complementary GPIO pin of the differential pair. All LFAST LVDS receive and transmit GPIO pairs on the MPC574xP exhibit the current injection issue.

There is an additional leakage path for the LFAST receive pins through the loopback test path when current is negatively injected into a GPIO pin connected to an LFAST pair. In this case, current will be injected into the same terminal of the GPIO pin connected through the loopback path (terminal to positive terminal, negative terminal to negative terminal). The pins affected by the loopback path on the MPC574xP are C[12] to/from I[5], and G[7] to/from I[6].

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

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

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

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

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

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

Option 2 : Low Latency Option per SAE specification

Option 1 : Preferred but High Latency Option per SAE specification

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

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

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

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

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

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

Table 1. Erratum behavior of NUM_EDGES_ERR

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NIB_VAL_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the	Ignore both NUM_EDGES_ERR error

		Calibration pulse are seen as back to back calibration pulses and no edges in between.	
FMSG_CRC_ERR	NUM_EDGES_ERR asserted twice	Same as NIB_VAL_ERR.	Ignore both NUM_EDGES_ERR errors
CAL_LEN_ERR	NUM_EDGES_ERR asserted once	Since the calibration pulse is not detected as a valid calibration pulse, the internal edges counter does not detect the end of one message and start of bad message (which has CAL_LEN_ERR); hence the NUM_EDGES_ERR gets asserted.	Ignore NUM_EDGES_ERR error
FMSG_OFLW	NUM_EDGES_ERR asserted once (random occurrence)	A message buffer overflow may lead the state machine to enter a state where it waits for a calibration pulse (behavior also seen in ERR007404). When in this state, the state machine can detect both a Pause pulse and a Calibration pulse as back to back calibration pulses and no edges in between. Then, the NUM_EDGES_ERR can get asserted. Since entry into this state is random, the error can be seen occasionally.	Ignore NUM_EDGES_ERR error

Table 2. Expected behavior, clarification of NUM_EDGES_ERR cases

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NUM_EDGES_ERR (when edges are less than expected)	NIB_VAL_ERR is asserted	When the actual number of edges in the message are less than expected, then a pause pulse gets detected as a nibble since the state machine expects nibbles when actually there is a pause pulse present. This generates NIB_VAL_ERR.	Ignore the NIB_VAL_ERR
NUM_EDGES_ERR (when edges are more than expected)	NIB_VAL_ERR and PP_DIAG_ERR are asserted	When the actual number of edges in a message are more than expected, then after receiving the	Ignore NIB_VAL_ERR and PP_DIAG_ERR

		programmed number of data nibbles, the state machine expects a pause pulse. However, the pause pulse comes later and gets detected as a nibble and hence NIB_VAL_ERR is asserted. Since the message length is not correct, PP_DIAG_ERR is also asserted.	

e8967: TSENS: (MPC5744P) Temperature sensor flag glitch during power up

Description: On a destructive reset generated by a low voltage detection (LVD), high temperature detection, or software there is a point where the Temperature Sensor (TSENS) loads trim values from memory. While this is happening there is a chance of a glitch on the TSENS output status even if the current temperature is within the normal temp range of the device. As a result of this glitch the corresponding TSENS status flag (TEMPx_y) in Power Management Controller's Temperature Event Status registers (PMC_ESR_TD) will get set.

Workaround: After coming out of reset read the PMC_ESR_TD register and check the values of TSENS output bits (TEMPx_y_OP) and status flags TEMPx_y. Refer to ERR008683. There are chances that the temp exceeds the overtemp limit (eg 150C) and TEMPx_y_OP flag is toggling at 3 MHz. Hence through software the TEMPx_y_OP status flag have to be sampled multiple times at > 6 MHz in order to prevent a wrong flag status detection. If the TEMPx_y_OP output bit is found to be zero after multiple sampling indicating that current temperature is okay then clear the corresponding TEMPx_y status flag if set.

e8683: TSENS: Temperature sensor status output bits in PMC_ESR_TD register shows indeterminate behavior

Description: There are 3 real time status bits as part of the Temperature Sensor (TSENS) that are associated with the -40C trip point, 150C trip point, and 165C trip point. These are the TEMPx_y_OP bits in the Power Management Controller's Temperature Event Status register (PMC_ESR_TD) where x = 0 or 1 (two TSENS instances) and y = 0 ,2 or 3 (-40C, 150C, and 165C flags). These bits reflect the current status of the TSENS output for their respective trip points. There are also 3 status flag bits for each trip point and TSENS instance, TEMPx_y. These bits get set when the temperature exceeds the corresponding threshold and clears when the temperature falls below its corresponding threshold and a one is written to it.

When the temperature crosses the 150C trip point setting the TEMPx_2 flag bit noting the 150C over temp condition the corresponding status bit, TEMP_x_2_OP, may be unstable and oscillate between a high and low state. This status bit should remain high as long as the over temp condition remains, but in this error condition it will toggle. The TEMPx_2_OP status bit is intended to allow the customer to monitor the over temp condition and know when to clear the TEMPx_2 flag bit. In the error condition this status bit could give a false low reading when the temperature is still above 150C.

When the unstable condition is occurring the 165C flag, TEMPx_3, may not set even if the temperature does exceed the 165C trip point. The toggling of the 150C over temp status bit will continue until the temperature is below the hysteresis window for the 150C trip point. After the temperature drops below that hysteresis window the status will correctly reflect a cleared condition.

The probability that the instability will occur will vary with the DCF trim settings (and customer trim settings) for the hot and cold flag trims. The least probable is when the cold flag trim is set to all 0's and the most probable is when the cold flag trim is set to all 1's. Also, while the status bit for the 150C over temp condition is toggling there is some loss of accuracy in the linear temperature sensor converted by the ADC.

Workaround: To get around this issue after the part heats to 150C and the user detects PMC_ESR_TD[TEMPx_2] = 1 disable the TSENS module. To disable the TSENS module it is necessary to clear both the digital output enable bit (TSx_DOUT_EN) and the analog output enable bit (TSx_AOUT_EN) in the Temperature Detector Configuration Register (PMC_CTL_TD). Then enable the temperature sensor by setting both bits back to 1. The status bits (TEMPx_y_OP) will now be operating correctly. It has been seen that upon enabling the TSENS that all flag bits (TEMPx_y) may be set and require clearing. This can be done based on of the status of the status bits (TEMPx_y_OP) to determine whether a valid over/ under temperature event is still occurring.

If the TSENS's 150C detection is used to generate system resets through setting of bits in the Temp Reset Event Enable Register (PMC_REE_TD) register either through flash programming in the DCF records or software configuration it is important to note that the TEMPx_y_OP bit is the bit that signals detection to the PMC_REE_TD. A user could decide to not implement the workaround if their desired result of 150C detection was to generate a system reset and continue to do so until temperature dropped below 150C.

Customers who are only concerned about the 165C detection also must enable the 150C flag, TEMPx_2, and implement the workaround. When the unstable condition is occurring the 165C flag, TEMPx_3, may not set even if the temperature does exceed the 165C trip point.

e7236: XBIC: XBIC may trigger false FCCU alarm

Description: The Crossbar Integrity Checker (XBIC) will incorrectly signal a fault alarm when a system bus request results in a bus error termination from a crossbar client. The Fault Correction and Collection Unit (FCCU) alarm number corresponding to the XBIC will be signaled.

Workaround: Software should handle faults on FCCU alarm corresponding to the XBIC in case of a system bus error.

e8730: XBIC: XBIC may store incorrect fault information when a fault occurs

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

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

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale 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.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale 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 Freescale 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. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.