Mask Set Errata for Mask 0N72D

Introduction

This report applies to mask 0N72D for these products:

- PXS30

Errata ID	Errata Title
3866	ADC Self Test algorithm S0 result can be incorrect at low temperature
4168	ADC: "Abort switch" aborts the ongoing injected channel as well as the upcoming normal channel
4186	ADC: triggering an ABORT or ABORTCHAIN before the conversion starts
4166	CTU: FIFO full and concurrent push and pop operations
3449	DEBUG: Device may hang due to external or 'functional' reset while using debug handshaking mechanism
2656	FlexCAN: Abort request blocks the CODE field
3407	FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1
3511	FlexPWM : Incorrect PWM operation when IPOL is set.
3335	FlexPWM: Incorrect PWM operation when mixing DMA and non-DMA controlled channels
3570	MC_ME: Possibility of Machine Check on Low-Power Mode Exit
4334	MC_RGM: Device stays in reset state on external reset assertion.
4396	e200z7: Erroneous Address Fetch
3419	e200z: Exceptions generated on speculative prefetch

e3866: ADC Self Test algorithm S0 result can be incorrect at low temperature

Errata type: Errata

Description: The ADC Self Test algorithm S step 0 (S0) measures ADC Vbandgap / VDD_HV_ADR. In the case where the S0 step occurs after the ADC has sampled and converted a value close to VDD_HV_ADR at low temperature (for example -40C) in some process corners the sampling time (specified by INPSAMP_S) of 80h is not long enough to allow the correct ADC sampling capacitor settling. This may lead to an incorrect converted value.

The Band Gap in the above specified condition is slow in discharging the sampling capacitor when the previous sampled voltage is much larger than the Band Gap output voltage (nominally 1.2V). The larger the voltage sampled before S0, the slower is the settling.

This issue can also affect S1 algorithm results since S1 = VDD_HV_ADV / Vbandgap.

Workaround: To eliminate the problem it is mandatory to:

(a) increase the sampling time for S supply self test (INSAMP_S) from 80h to FFh and

(b) insert a sacrificial ADC conversion immediately before the S supply self test.

The user software must insert a single-shot S algorithm Step 0 conversion (also called sacrificial S0 conversion) before the normal S supply self test to achieve accurate sample capacitor settling. The user software must prohibit any other conversions between the sacrificial S0 conversion and normal S0 conversion of the S supply self test.

e4168: ADC: "Abort switch" aborts the ongoing injected channel as well as the upcoming normal channel

Errata type: Errata

Description: If an Injected chain (jch1,jch2,jch3) is injected over a Normal chain (nch1,ch2,ch3,ch4) the Abort switch doesn't behave as expected.

Expected behavior:

Correct Case (without SW Abort on jch3): Nch1-> Nch2(aborted)->Jch1 -> Jch2> Jch3 ->Nch2(restored)-> Nch3->Nch4

Correct Case(with SW Abort on jch3): Nch1-> Nch2(aborted)->Jch1 -> Jch2> Jch3(aborted) ->Nch2(restored)-> Nch3->Nch4

Observed unexpected behavior:

Fault1 (without SW abort on jch3): Nch1-> Nch2(aborted)->Jch1 -> Jch2> Jch3 -> Nch3->Nch4 (Nch2 not restorted)

Fault2 (with SW abort on jch3): Nch1-> Nch2->Jch1 -> Jch2> Jch3(aborted) ->Nch4 (Nch2 not restored & Nch3 conversion skipped)

Workaround: It's possible to detect the unexpected behavior by using the CEOCFR0 register. The CEOCFR0.EOC_CHx field will not be set for a not restored or skipped channel, which indcates this issue has occurred. The CEOCFR0.EOC_CHx fields need to be checked before the next Normal chain execution (in scan mode). The CEOCFR0.EOC_CHx fields should be read by every ECH interrupt at the end of every chain execution.

e4186: ADC: triggering an ABORT or ABORTCHAIN before the conversion starts

Errata type: Errata

Description: When ABORTCHAIN is programmed and an injected chain conversion is programmed afterwards, the injected chain is aborted, but neither JECH is set, nor ABORTCHAIN is reset.

When ABORT is programmed and normal/injected chain conversion comes afterwards, the ABORT bit is reset and chain conversion runs without a channel abort.

If ABORT, or ABORTCHAIN, feature is programmed after the start of the chain conversion, it works properly.

Workaround: Do not program ABORT/ABORTCHAIN before starting the execution of the chain conversion.

e4166: CTU: FIFO full and concurrent push and pop operations

Errata type: Errata

Description: With a full FIFO, if concurrent FIFO read and write operations occur, then the order of the FIFO is not correct.

For example, FIFO is full and contains data A,B,C,D. Then there are POP and a PUSH requests in the same clock cycle. After the PUSH and POP operations instead of correct data B,C,D,E, the FIFO contains the data B,C,E,D. Data A is pushed out correctly, but data E and D are swapped.

Application software can detect the swap between E and D by reading the CTU.FLx.ADC and CTU.FLx.N_CH fields, unless E and D refer to the same ADC and same channel.

Workaround: To reduce the risk of this issue 2 suggestions are given:

1) lower the FIFO threshold to less than the size of the FIFO (probability is reduced, but can't be fully excluded).

2) forbid 2 consecutive conversions from the same ADC and channel source to allow swap detection by reading the CTU.FLx.ADC and CTU.FLx.N_CH fields.

e3449: DEBUG: Device may hang due to external or 'functional' reset while using debug handshaking mechanism

Errata type: Errata

Description: If the low-power mode debug handshake has been enabled and an external reset or a 'functional' reset occurs while the device is in a low-power mode, the device will not exit reset.

Workaround: The NPC_PCR[LP_DBG_EN] bit must be cleared to ensure the correct reset sequence.

e2656: FlexCAN: Abort request blocks the CODE field

Errata type: Errata

Description: An Abort request to a transmit Message Buffer (TxMB) can block any write operation into its CODE field. Therefore, the TxMB cannot be aborted or deactivated until it completes a valid transmission (by winning the CAN bus arbitration and transmitting the contents of the TxMB).

Workaround: Instead of aborting the transmission, use deactivation instead.

Note that there is a chance the the deactivated TxMB can be transmitted without setting IFLAG and updating the CODE field if it is deactivated.

e3407: FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1

Errata type: Errata

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").

e3511: FlexPWM : Incorrect PWM operation when IPOL is set.

Errata type: Errata

Description: When IPOL bit is set in complementary mode, the source of the PWMi waveform is supposed to be switched from the VAL2 and VAL3 registers to the VAL4 and VAL5 registers. This switch is not happening.

Workaround: Instead of setting IPOL bt, the application can swap the VAL2/3 values with the VAL4/5 values.

e3335: FlexPWM: Incorrect PWM operation when mixing DMA and non-DMA controlled channels

Errata type: Errata

Description: When some submodules use DMA to load their VALx registers and other submodules use non-DMA means that means direct writes from the CPU, the LDOK bits for the non-DMA submodules can be incorrectly cleared at the completion of the DMA controlled load cycle. This leads to the non-DMA channels not being properly updated.

Submodules that use DMA to read the input capture registers do not cause a problem for non-DMA submodules.

Workaround: Set the DMA enable bit to 1 also for non-DMA submodules, according to this the DMA will not incorrectly clear the LDOK bit for non-DMA submodules but they will be set to 1 at the end of each DMA cycle. When the CPU has to update the VALx registers of non-DMA submodules, first clear LDOK bit for non-DMA submodules

e3570: MC_ME: Possibility of Machine Check on Low-Power Mode Exit

Errata type: Errata

Description: When executing from the flash and entering a Low-Power Mode (LPM) where the flash is in low-power or power-down mode, 2-4 clock cycles exist at the beginning of the RUNx to LPM transition during which a wakeup or interrupt will generate a checkstop due to the flash not being available on RUNx mode re-entry. This will cause either a checkstop reset or machine check interrupt.

Workaround: If the application must avoid the reset, two workarounds are suggested:

1) Configure the application to handle the machine check interrupt in RAM dealing with the problem only if it occurs

2) Configure the MCU to avoid the machine check interrupt, executing the transition into low power modes in RAM

There is no absolute requirement to work around the possibility of a checkstop reset if the application can accept the reset, and associated delays, and continue. In this event, the WKPU.WISR will not indicate the channel that triggered the wakeup though the F_CHKSTOP flag will indicate that the reset has occurred. The F_CHKSTOP flag could still be caused by other error conditions so the startup strategy from this condition should be considered alongside any pre-existing strategy for recovering from an F_CHKSTOP condition.

e4334: MC_RGM: Device stays in reset state on external reset assertion.

Errata type: Errata

Description: On an external reset that is configured to be 'long' the device may remain in reset if the system clock is configured to be sourced by a clock source other than the 16 MHz Internal RC Oscillator (IRCOSC). Recovery from the reset in this case can only be achieved via a power-down and power-up cycle.

The failure condition can only be seen with the following Reset Generation Module (MC_RGM) settings for Functional Event Short Sequence register, External Reset field (RGM_FESS[SS_EXR]) and Functional Bidirectional Reset Enable register, External Reset field (RGM_FBRE[BE_EXR]):

- RGM_FESS[SS_EXR] = 0b0 (long external reset)

- RGM_FBRE[BE_EXR] = 0b0 (asserted on external reset event)

Note 1: This condition can only occur if the cause of the device reset was the external reset assertion. It does not occur if, for example, the device reset was due to a power-on.

Note 2: RGM_FESS[SS_EXR] = 0b0 and RGM_FBRE[BE_EXR] = 0b0 are the default settings out of power-on reset (POR).

Workaround: There are two possible workarounds. In both, the workaround takes effect only after software has reconfigured the MC_RGM. Therefore, in order to ensure that the issue cannot occur after POR exit and before the software has executed the workaround, the system clock must not be re-configured in the Mode Entry module (MC_ME) to be sourced by a clock source other than the IRCOSC until after the workaround has been executed.

Workaround #1:

Always configure the external reset event to prevent the external reset output to be driven by the MC_RGM by writing 0b1 to RGM_FBRE[BE_EXR].

If the external reset has been configured to be long (RGM_FESS[SS_EXR] = 0b0) and self testing has been enabled via the flash option, the external reset pin will still be asserted from the time of external assertion until reset sequence exit after start-up self test execution.

If the external reset has been configured to be long (RGM_FESS[SS_EXR] = 0b0) and self testing has been disabled via the flash option, the external reset pin will still be asserted from the time of external assertion until the chip configuration is loaded from the flash during reset PHASE3.

If the external reset has been configured to be short (RGM_FESS[SS_EXR] = 0b1), the external reset pin will still be released as soon as it is no longer asserted from off-chip.

Workaround #2:

Always configure the external reset as 'short' by writing 0b1 to RGM_FESS[SS_EXR]. In addition, use software to trigger a long 'functional' or 'destructive' reset via the Mode Entry module (MC_ME) if flash initialization or start-up self test is required.

The impact of this workaround is the additional time that the device is in reset (due to the short reset sequence triggered by the external reset) and the overhead required for software to check the reset status and request a software reset.

e4396: e200z7: Erroneous Address Fetch

Errata type: Errata

Description: Under certain conditions, if a static branch prediction and a dynamic return prediction (which uses the subroutine return address stack) occur simultaneously in the Branch Target Buffer (BTB), the e200z7 core can issue an errant fetch address to the memory system (instruction fetched from wrong address).

This can only happen when the static branch prediction is "taken" but the branch actually resolves to "not taken". If the branch resolves to taken, correct fetching occurs for this branch path and no issue is seen.

If Branch Unit Control and Status Register (BUCSR) Branch Prediction Control Static (BPRED) = 0b00, 0b01, or 0b10, then static branch prediction is configured as "taken". The issue can occur with these settings.

If BUSCR[BPRED] = 0b11, then static branch prediction is configured as "not taken". The issue does not occur with this setting.

The issue does not appear in applications compiled with VLE.

Workaround: To prevent the issue from occurring, confirgure static branch prediction to "not taken" by setting the Branch Unit Control and Status Register (BUCSR) Branch Prediction Control Static (BPRED) to 0b11.

For customers using VLE, there is no workaround required, the issue will not be seen.

e3419: e200z: Exceptions generated on speculative prefetch

Errata type: Errata

Description: The e200z7 core can prefetch up to 2 cache lines (64 bytes total) beyond the current instruction execution point. If a bus error occurs when reading any of these prefetch locations, the machine check exception will be taken. For example, executing code within the last 64 bytes of a memory region such as internal SRAM, may cause a bus error when the core pre-fetches past the end of memory. An ECC exception can occur if the core prefetches locations that are valid, but not yet initialized for ECC.

The Boot Assist Monitor (BAM) is located at the end of the address space and so may cause instruction pre-fetches to wrap-around to address 0 in internal flash memory. If this first block of flash memory contains ECC errors, such as from an aborted program or erase operation, a machine-check exception will be asserted. At this point in the boot procedure, exceptions are disabled, but the machine-check will remain pending and the exception vector will be taken if user application code subsequently enables the machine check interrupt.

Workaround: Do not place code to be executed within the last 64 bytes of a memory region. When executing code from internal ECC SRAM, initialize memory beyond the end of the code until the next 32-byte aligned address and then an additional 64 bytes to ensure that prefetches cannot land in uninitialized SRAM.

To guard against the possibility of the BAM causing a machine-check exception to be taken, as noted in the errata description, user application code should check and clear the Machine Check Syndrome Register (MCSR) in the core before enabling the machine check interrupt. This can be done by writing all 1s to MCSR.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Document Number: PXS30_0N72D
Rev. 12 APR 2012
Preliminary