# Mask Set Errata for Mask 0M18Y

## Introduction

This report applies to mask 0M18Y for these products:
- MPC5602D
- MPC5601D

<table>
<thead>
<tr>
<th>Prior ID</th>
<th>Current ID</th>
<th>Erratum Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>e6149</td>
<td>e3103</td>
<td>Accesses to reserved PSMI registers do not return a transfer error</td>
</tr>
<tr>
<td>e4168</td>
<td></td>
<td>ADC: &quot;Abort switch&quot; aborts the ongoing injected channel as well as the upcoming normal channel</td>
</tr>
<tr>
<td>e5485</td>
<td>e3069</td>
<td>ADC: ADC_DMAE[DCLR] set to 1 clears the DMA request incorrectly</td>
</tr>
<tr>
<td></td>
<td>e4186</td>
<td>ADC: triggering an ABORT or ABORTCHAIN before the conversion starts</td>
</tr>
<tr>
<td>e9984</td>
<td>e3446</td>
<td>CTU: The CTU (Cross Trigger Unit) CLR_FLAG in EVTCFGGR register does not function as expected</td>
</tr>
<tr>
<td>e9212</td>
<td>e3252</td>
<td>CTU: ADC1_X[0:3] channels cannot be triggered by CTU</td>
</tr>
<tr>
<td>e3449</td>
<td></td>
<td>DEBUG: Device may hang due to external or 'functional' reset while using debug handshaking mechanism</td>
</tr>
<tr>
<td>e10637</td>
<td>e3512</td>
<td>ECSM: ECSM_PFEDR displays incorrect endianness</td>
</tr>
<tr>
<td>e2656</td>
<td></td>
<td>FlexCAN: Abort request blocks the CODE field</td>
</tr>
<tr>
<td>e3407</td>
<td></td>
<td>FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1</td>
</tr>
<tr>
<td>e8675</td>
<td>e3216</td>
<td>LINFlex TX pin tristated when no transmission on-going</td>
</tr>
<tr>
<td>e8099</td>
<td>e3195</td>
<td>LINFlex: Limitations for DMA access to LINFlex</td>
</tr>
<tr>
<td>e4340</td>
<td></td>
<td>LINFlexD: Buffer overrun can not be detected in UART Rx FIFO mode</td>
</tr>
<tr>
<td>e10107</td>
<td>e3466</td>
<td>LINFlexD: Register bus aborts are not generated on illegal accesses to reserved addresses within the register address space of LINFlexD</td>
</tr>
<tr>
<td>e4107</td>
<td>e2999</td>
<td>MC_CGM and MC_PCU: A data storage exception is not generated on an access to MC_CGM or MC_PCU when the respective peripheral is disabled at MC_ME</td>
</tr>
<tr>
<td>e8761</td>
<td>e3219</td>
<td>MC_CGM: System clock may stop for case when target clock source stops during clock switching transition</td>
</tr>
</tbody>
</table>

*Table continues on the next page...*
<table>
<thead>
<tr>
<th>Prior ID</th>
<th>Current ID</th>
<th>Erratum Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>e7776</td>
<td>e3190</td>
<td>MC_ME: Main VREG not disabled during STOP0 or HALT0 mode if RUN[0..3] mode selects FXOSC to be running and target mode selects FXOSC as system clock</td>
</tr>
<tr>
<td></td>
<td>e3570</td>
<td>MC_ME: Possibility of Machine Check on Low-Power Mode Exit</td>
</tr>
<tr>
<td>e9140</td>
<td>e3247</td>
<td>MC_ME: STANDBY0/HALT0/STOP0 modes cannot be entered if the FlexCAN peripheral is active</td>
</tr>
<tr>
<td>e10491</td>
<td>e3574</td>
<td>MC_RGM: A non-monotonic ramp on the VDD_HV/BV supply can cause the RGM module to clear all flags in the DES register</td>
</tr>
<tr>
<td>e2835</td>
<td>e2958</td>
<td>MC_RGM: Clearing a flag at RGM_DES or RGM_FES register may be prevented by a reset</td>
</tr>
<tr>
<td>e5301</td>
<td>e3060</td>
<td>MC_RGM: SAFE mode exit may be possible even though condition causing the SAFE mode request has not been cleared</td>
</tr>
<tr>
<td>e8499</td>
<td>e3209</td>
<td>NMI pin configuration limitation in standby mode</td>
</tr>
<tr>
<td>e8500</td>
<td>e3210</td>
<td>PA[1] pull-up enabled when NMI activated</td>
</tr>
<tr>
<td>e9175</td>
<td>e3249</td>
<td>PB[8:10],PD[0:1] pins configuration during standby</td>
</tr>
<tr>
<td>e8329</td>
<td>e3200</td>
<td>PIT events cannot be used to trigger ADC conversion incase BCTU runs on divided system clock</td>
</tr>
<tr>
<td>e9834</td>
<td>e3300</td>
<td>SIRC: Increased current consumption when the SIRC is switched off in STANDBY mode</td>
</tr>
<tr>
<td>e9725</td>
<td>e4405</td>
<td>SR bit of LINFlexD GCR register is not cleared automatically by hardware</td>
</tr>
<tr>
<td></td>
<td>e4146</td>
<td>When an ADC conversion is injected, the aborted channel is not restored under certain conditions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>25AUG2011</td>
<td>Initial revision</td>
</tr>
</tbody>
</table>

*Table continues on the next page...*
The following errata were removed.

- e3442: Now described in RM
- e3220: Moved to e3219 to maintain numbering across devices
- e3269: Now described in RM
- e3286: Moved to e3247 to maintain numbering across devices
- e3250: Moved to e3209 to maintain numbering across devices
- e3251: Moved to e3210 to maintain numbering across devices
- e3301: Moved to e3200 to maintain numbering across devices

The following errata were added.

- e4405: New errata
- e3219: Moved from e3220 to maintain numbering across devices
- e3210: Moved from e3251 to maintain numbering across devices
- e4340: New errata
- e3209: Moved from e3250 to maintain numbering across devices
- e3247: Moved from e3286 to maintain numbering across devices
- e3200: Moved from e3301 to maintain numbering across devices
- e4146: New errata
- e4186: New errata
- e4168: New errata

The following errata were revised.

- e3512: Description updated for clarity
- e3570: Description updated for clarity

---

**e3103: Accesses to reserved PSMI registers do not return a transfer error**

**Description:** When accessing the following reserved locations in the SIUL address space, a transfer error will not be returned:

SIUL base address 0xC3F90000 + offsets 0x500-0x504, 0x50A-0x50C, 0x512-0x514, 0x51A-0x525, 0x528-0x539, 0x53B-0x53D

**Workaround:** None

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

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

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

**e3069: ADC: ADC_DMAE[DCLR] set to 1 clears the DMA request incorrectly**

Description: When ADC_DMAE[DCLR] is set the DMA request should be cleared only after the data registers are read. However for this case the DMA request is automatically cleared and will not be recognised by the eDMA.

Workaround: None

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

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. In case a CTU conversion is demanded, the CTU conversion is aborted by the ADC digital logic but the CTU awaits ADC analogue acknowledgement that never arrives, stopping all CTU operation until next 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.

If the CTU is being used in trigger mode, there will always be a small vulnerable window preventing reliably reading the ADC status and using ADC.MCR[ABORT] or ADC.MCR[ABORTCHAIN]. If these functions are required, disable all CTU triggers to that ADC first and do not start the CTU if the ABORT or ABORTCHAIN flags have been set whilst the ADC was IDLE. If the CTU cannot be disabled, an optimised ABORT should be implemented in software that de-configures further channel conversions using the ADC.NMCRx registers before waiting for ADC.MSR[NSTART] == 0. At this point, the ADC should be IDLE and the NMCRx registers can be reinstated for next use.

**e3446: CTU : The CTU (Cross Trigger Unit) CLR_FLAG in EVTCFGR register does not function as expected**

Description: If the CTU CLR_FLG is set and the CTU is idle, a PIT triggered request to the CTU does not result in the correct ADC channel number being latched. The previous ADC channel number is latched instead of the requested channel number.

Workaround: There is no software workaround to allow the CLR_FLAG functionality to operate correctly. Do not program the CLR_FLAG bit to ‘1’.
e3252: CTU: ADC1_X[0:3] channels cannot be triggered by CTU

Description: CTU_EVTCFGRx[CHANNEL_VALUE] bitfield is 6 bit wide and can select ADC channels within the range 0-63.
ADC1_X[0:3] indexed CH[64-95], cannot be selected.

Workaround: None

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

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.

e3512: ECSM: ECSM_PFEDR displays incorrect endianness

Description: The ECSM_PFEDR register reports ECC data using incorrect endianness. For example, a flash location that contains the data 0xAABBCCDD would be reported as 0xDDCCBBAA at ECSM_PFEDR.
This 32-bit register contains the data associated with the faulting access of the last, properly-enabled flash ECC event. The register contains the data value taken directly from the data bus.

Workaround: Software must correct endianness.

e2656: FlexCAN: Abort request blocks the CODE field

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 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

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”).

---

e3216: LINFlex TX pin tristated when no transmission on-going

**Description:** When no LINFlex transmission is on-going, the LINFlex TX pad is put in tristate instead of remaining high.

**Workaround:** Enable internal weak pull-up on TX pin to ensure TX remains high. This will incur a small static current (~100uA) during transmission of ‘0’.

---

e3195: LINFlex: Limitations for DMA access to LINFlex

**Description:** The DMA handshaking to the LINFlex can fail when the LINFlex operates on a divided peripheral clock.

**Workaround:** Don’t divide the LINFlex peripheral clock if DMA access is required.

---

e4340: LINFlexD: Buffer overrun can not be detected in UART Rx FIFO mode

**Description:** When the LINFlexD is configured in UART Receive (Rx) FIFO mode, the Buffer Overrun Flag (BOF) bit of the UART Mode Status Register (UARTSR) register is cleared in the subsequent clock cycle after being asserted.

User software can not poll the BOF to detect an overflow.

The LINFlexD Error Combined Interrupt can still be triggered by the buffer overrun. This interrupt is enabled by setting the Buffer Overrun Error Interrupt Enable (BOIE) bit in the LIN Interrupt enable register (LINIER). But the BOF bit will be cleared when the interrupt routine is entered, preventing the user to identify the source of error.
Workaround: Buffer overrun errors in UART FIFO mode can be detected by enabling only this source in the LIN Error Combined interrupt.

**e3466: LINFlexD: Register bus aborts are not generated on illegal accesses to reserved addresses within the register address space of LINFlexD**

**Description:** Register bus aborts are not generated on illegal accesses to reserved addresses within the register address space of LINFlexD. This is applicable to LINFlex modules supporting master-only mode.

**Workaround:** None

**e2999: MC_CGM and MC_PCU: A data storage exception is not generated on an access to MC_CGM or MC_PCU when the respective peripheral is disabled at MC_ME**

**Description:** If a peripheral with registers mapped to MC_CGM or MC_PCU address spaces is disabled via the MC_ME any read or write accesses to this peripheral will be ignored without producing a data storage exception.

**Workaround:** For any mode other than a low-power mode do not disable any peripheral that is mapped to MC_CGM or MC_PCU.

**e3219: MC_CGM: System clock may stop for case when target clock source stops during clock switching transition**

**Description:** The clock switching is a two step process. The availability of the target clock is first verified. Then the system clock is switched to the new target clock source within two target clock cycles.

For the case when the FXOSC stops during the required two cycles, the switching process may not complete, causing the system clock to stop and prevent further clock switching. This may happen if one of the following cases occurs while the system clock source is switching to FXOSC:

- FXOSC oscillator failure
- SAFE mode request occurs, as this mode will immediately switch OFF the FXOSC (refer to ME_SAFE_MC register configuration)

**Workaround:** The device is able to recover through any reset event (‘functional’, ‘destructive’, internal or external), so typically either the SWT (internal watchdog) will generate a reset or, in case it is used in the application, the external watchdog will generate an external reset. In all cases the devices will restart properly after reset.

To reduce the probability that this issue occurs in the application, disable SAFE mode transitions when the device is executing a mode transition with the FXOSC as the system clock source in the target mode.
e3570: MC_ME: Possibility of Machine Check on Low-Power Mode Exit

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.

e3247: MC_ME: STANDBY0/HALT0/STOP0 modes cannot be entered if the FlexCAN peripheral is active

Description: If any FlexCAN module is enabled in current mode by the ME_RUN_PCx/ME_PCTLx registers of the MC_ME and also enabled at the FlexCAN module, (MCR.B.MDIS=0), STANDBY0/HALT0/STOP0 modes will not be entered if the target low power mode is disabling the module, the device mode transition hangs.

Workaround: The FlexCAN module must be frozen (FLEXCANx_MCR[FRZ]=1) in DRUN or RUNx mode before entering STANDBY0/HALT0/STOP0 modes.

e3574: MC_RGM: A non-monotonic ramp on the VDD_HV/BV supply can cause the RGM module to clear all flags in the DES register

Description: During power up, if there is non-monotonicity in power supply ramp with a voltage drop > 100mV due to external factors, such as battery cranking or weak board regulators, the SoC may show a no flag condition (F_POR==LVD12==LVD27==0).

Under these situations, it is recommended that customers use a workaround to detect a POR. In all cases, initialization of the device will complete normally.

Workaround: The software workaround need only be applied when neither the F_POR, LVD27 nor LVD12 flag is set and involves checking SRAM contents and monitoring for ECC errors during this process. In all cases, an ECC error is assumed to signify a power-on reset (POR).

Three suggestions are made for software workarounds. In each case, if POR is detected all RAM should be initialized otherwise no power-on condition is detected and it is possible to initialize only needed parts of RAM while preserving required information.

Software workaround #1:
An area of RAM can be reserved by the compiler into which a KEY, such as 0x3EC1_9678, is written. This area can be checked at boot and if the KEY is incorrect or an ECC error occurs, POR can be assumed and the KEY should be set. Use of a KEY increases detection rate to 31 bits(=<10e-9) or 23bits (= <5.10e-6) instead of 7-bit linked to ECC (=<10e-2).

Software workaround #2:

When runtime data should be retained and RAM only fully re-initialized in the case of POR, a checksum should be calculated on the runtime data area after each data write. In the event of a reset where no flags are set, the checksum should be read and compared with one calculated across the data area. If reading the checksum and the runtime data area succeeds without an ECC error, and the checksums match, it is assumed than no POR occurred. The checksum could be a CRC, a CMAC or any other suitable hash.

Software workaround #3:

Perform a read of memory space that is expected to be retained across an LVD reset. If there are no ECC errors, it can be assumed that an LVD reset occurred rather than a POR.

e2958:  MC_RGM: Clearing a flag at RGM_DES or RGM_FES register may be prevented by a reset

Description:  Clearing a flag at RGM_DES and RGM_FES registers requires two clock cycles because of a synchronization mechanism. As a consequence if a reset occurs while clearing is on-going the reset may interrupt the clearing mechanism leaving the flag set.

Note that this failed clearing has no impact on further flag clearing requests.

Workaround:  No workaround for all reset sources except SW reset.

Note that in case the application requests a SW reset immediately after clearing a flag in RGM_xES the same issue may occur. To avoid this effect the application must ensure that flag clearing has completed by reading the RGM_xES register before the SW reset is requested.

e3060:  MC_RGM: SAFE mode exit may be possible even though condition causing the SAFE mode request has not been cleared

Description:  A SAFE mode exit should not be possible as long as any condition that caused a SAFE mode entry is still active. However, if the corresponding status flag in the RGM_FES register has been cleared, the SAFE mode exit may incorrectly occur even though the actual condition is still active.

Workaround:  Software must clear the SAFE mode request condition at the source before clearing the corresponding RGM_FES flag. This will ensure that the condition is no longer active when the RGM_FES flag is cleared and thus the SAFE mode exit can occur under the correct conditions.

e3209:  NMI pin configuration limitation in standby mode

Description:  NMI pin cannot be configured to generate Non Maskable Interrupt event to the core (WKPU_NCR[NDSS] = "00") if the following standby mode is to be used:

- NMI pin enabled for wake-up event,
• standby exit sequence boot from RAM,
• code flash module power-down on standby exit sequence.

With following configuration following scenario may happen:
1. System is in standby
2. NMI event is triggered on PA[1]
3. System wakeup z0 core power domain.
4. z0 core reset is released and NMI event is sampled by core on first clock-edge.
5. z0 core attempt to fetch code at 0x10 address (IVPR is not yet initialized by application) and receive an exception since flash is not available
6. z0 core enter machine check and execution is stalled.

Workaround: If NMI is configured as wake-up source, WKPU_NCR[NDSS] must be configures as “11”. This will ensure no NMI event is trigger on the core but ensure system wakeup is triggered.

After standby exit, core will boot and configure its IVOR/IVPR, it may then re-configure WKPU_NCR:DSS to the appropriate configuration for enabling NMI/CI/MCP.

e3210:  PA[1] pull-up enabled when NMI activated

Description: When NMI is enabled (either WKPU_NCR[NREE] or WKPU_NCR[NFEE] set to ‘1’), PA[1] pull-up is automatically activated independently from SIUL_PCR1[WPS] and SIUL_PCR1[WPE].

This has no effect during STANDBY mode. PA[1] pull-up is then correctly configured through WKPU_WIPUER/IPUE[2].

Workaround: None

e3249:  PB[8:10],PD[0:1] pins configuration during standby

Description: PB[8:10], PD[0:1] are the pins having both wake-up functionality and analog functionality.

As for all wake-up pins, it must be driven either high level or low level (possibly using the internal pull-up) during standby.

In case the pin is connected to external component providing analog signal, it is important to check that this external analog signal is either lower than 0.2*VDD_HV or higher than 0.8*VDD_HV not to incur extra consumption.

Workaround: None

e3200:  PIT events cannot be used to trigger ADC conversion incase BCTU runs on divided system clock

Description: If BCTU operates on divided system clock (i.e MC_CGM.CGM_SC_DC2.DIV0 NOT EQUAL 0x0), events from PIT timer cannot be used to trigger ADC conversion. In this case BCTU fails to latch the channel number to be converted. So BCTU will send the conversion request to ADC but the channel number will be corresponding to previous non-PIT conversion request or 0th channel in case no previous non-PIT event has occured.
Workaround: Always write MC_CGM.CGM_SC_DC2.DIV0 to 0x0 to run BCTU at system frequency.

**e3300: SIRC: Increased current consumption when the SIRC is switched off in STANDBY mode**

**Description:** The MCU current consumption will be increased by 10uA for the case when the SIRC is switched off in STANDBY mode (SIRC_CTL[SIRCON_STDBY] = 0).

**Workaround:** Do not switch off SIRC in STANDBY mode, therefore, ensure SIRC_CTL[SIRCON_STDBY] = 1.

**e4405: SR bit of LINFlexD GCR register is not cleared automatically by hardware**

**Description:** After setting the SR bit of GCR (Global Control Register) to reset the LinFlexD controller, this bit is not cleared automatically by the hardware, keeping the peripheral in reset state.

**Workaround:** This bit should be cleared by software to perform further operations.

**e4146: When an ADC conversion is injected, the aborted channel is not restored under certain conditions**

**Description:** When triggered conversions interrupt the ADC, it is possible that the aborted conversion does not get restored to the ADC and is not converted during the chain. Vulnerable configurations are:

- Injected chain over a normal chain
- CTU trigger over a normal chain
- CTU trigger over an injected chain

When any of these triggers arrive whilst the ADC is in the conversion stage of the sample and conversion, the sample is discarded and is not restored. This means that the channel data register will not show the channel as being valid and the CEOCFFRx field will not indicate a pending conversion. The sample that was aborted is lost.

When the trigger arrives during the final channel in a normal or injected chain, the same failure mode can cause two ECH/JECH interrupts to be raised.

If the trigger arrives during the sampling phase of the last channel in the chain, an ECH is triggered immediately, the trigger is processed and the channel is restored and after sampling/conversion, a second ECH interrupt occurs.

In scan mode, the second ECH does not occur if the trigger arrives during the conversion phase. In one-shot mode, the trigger arriving during the conversion phase of the last channel restarts the whole conversion chain and the next ECH occurs at completion of that chain.

**Workaround:** It is suggested that the application check for valid data using the CDR status bits or the CEOCFFRx registers to ensure all expected channels have converted. This can be tested by running a bitwise AND and an XOR with either the JCMRx or NCMRx registers and the CEOCFFRx registers during the ECH of JECH handler. Any non-zero value for \(( xCMRx \& ( xCMRx \oplus \text{CEOCFFRx} ) \) ) indicates that a channel has been missed and conversion should be requested again.

Mask Set Errata for Mask 0M18Y, Rev. 17MAY2012

Freescale Semiconductor, Inc. 11
Spurious ECH/JECH interrupts can be detected by checking the NSTART/JSTART flags in the ADC Module Status Registers – if the flag remains set during an ECH/JECH interrupt then another interrupt will follow after the restored channel or chain has been sampled and converted.