T4240 Chip Errata

A-008968: RMan Type11 duplicate segment

Affects: RMAN
Description: A Type11 outbound message consists of one to sixteen segments. Duplication of the last transmitted segment for a non-multicast (Message Descriptor[Multi-cast mode]=0) message may occur if the message starts outbound segmentation after a non-multicast Type11 has previously detected a segment response time-out. This only occurs if the messages are assigned to the same segmentation unit.

The most common scenario is for a source A to receive a segment response timeout when a single segment Type11 message is transmitted to an unresponsive destination B. Following this event, source A sends the last segment of the next Type11 message twice to destination C.

Impact: Duplication of a Type11 segment causes destination to receive additional payload. This could lead to one of the following:
- Partial message re-assembled at destination may time-out, causing an incomplete message with error status.
- If a single segment message is duplicated, destination does not detect an error, but receives a duplicate message.
- A segment of message A may be combined with subsequent segment(s) of message B and create a new message C which has a bad payload likely undetectable by the destination at the time of reception.

Workaround: Possible workarounds are listed below, each with their own limitations:

**Workaround #1**
Use Type9 data streaming protocol if data delivery does not require a response. Type9 is unaffected by the duplication issue and provides a much better performance than Type11 messages. Type9 may cause loss of packets without the sender knowing it if SRIO fabric fails to deliver data. Protocols such as Ethernet TCP/IP using sequence numbers can handle this potential temporary loss of data.

**Workaround #2**
Use Type11 multicast by setting message descriptor field MD/MM=1. This has a limit of max 256B of payload. MD[EMG/MG/ML] (MD[Extended Multicast group/Multicast group/Multicast list]) must be set accordingly to indicate a single destination.

**Workaround #3**
Allocate a segmentation engine, using arbitration group (AG) assignment with RMan Message Manager Segmentation Execution Privilege Register MMSEPR0, per destination. This works under the assumption that sending a duplicate segment to an unresponsive destination (link down), does not have any adverse effects. This first restriction is a limit of 4 AGs (4 potential destinations that can cause time-out). The second restriction is the desired arbitration between messages to these destinations may not be as desired. The details are discussed below:

Workaround #3 - Details

To separate outbound RMan traffic for two destinations onto separate segmentation engines, do the following:

1. Assign RMan sub-portal 0/1 for messages to destination A/B respectively. This implies using a separate frame/work queue for each destination. RMan arbitrates using round-robin between sub-portals to balance traffic between SRIO ports, if used this way.

2. Use a different AG on each sub-portal. For example, AG0 (WQ0) for destination A and AG1 (WQ1) for destination B. Unique arbitration groups are needed to designate traffic to a specific segmentation engine.

3. Program engines for arbitration groups using register MMSEPR0. For example, MMSEPR0= 01010202h. This allows destination A to use segmentation engine 0/1 (AG0 only), and destination B to use segmentation engine 2/3 (AG1 only).

Fix plan: No plans to fix
A-009006: Loss of inbound RMan hardware reassembly contexts for Type11

Affects: RMAN

Description: RMan receives inbound messages (up to 4KB in size) as 1 to 16 messages segments. When the first segment of a multi-segment message is received, a hardware context is opened to track reassembly of the entire message in memory. If all segment numbers of a Type11 inbound message are received, then the hardware context is closed and set back into invalid state ready to receive a new message. However, if the last segment of a multi-segment message has a different message length than the original starting message segment, loss of one reassembly hardware context occurs. The most likely source of this event would be a badly generated RapidIO message header or a duplication/loss of a Type11 segment by transmitter.

Loss of one inbound reassembly hardware context occurs for a flow based on the MMRCARn setup. This is detected as inbound reassembly idle (MMSR[IMUB]=0) and open hardware reassembly contexts (MMSR[OHWC]>0). The loss of contexts is compounded by additional errors until the dedicated/generic flow count has been exhausted. An exhausted dedicated/generic flow may result in dropped packets or retried messages.

Impact: One or more losses of inbound hardware reassembly context for Type11 may eventually result in zero contexts available as defined by MMRCARn. When this occurs, Type11 has retried for an exhausted flow or generic context.

Workaround: The following are workarounds available for this issue:

Workaround #1
User may switch to use Type9 data packets if data delivery notification is not required since it is unaffected by the errata.

Workaround #2
User may redirect inbound reassembly errors to a unique Type11 error frame queue by setting MMMR[EFQ]=1. If FD[MFE]=1 and FD[SRT] =0, potential loss of inbound reassembly context may have occurred. User should quiesce RMan by setting MMMR[MMQ]=1 and stop inbound traffic from sender to check if MMSR[OHWC]>0. To reclaim lost reassembly H/W contexts, software must perform a soft reset of RMan as defined by register field MMMR[SR]. A soft reset of RMan does not cause the SRIO link to go down. The Type11 messages in this scenario is retried.

Fix plan: No plans to fix
A-008968: RMan Type11 duplicate segment

**Affects:** RMAN

**Description:** A Type11 outbound message consists of one to sixteen segments. Duplication of the last transmitted segment for a non-multicast (Message Descriptor[Multi-cast mode]=0) message may occur if the message starts outbound segmentation after a non-multicast Type11 has previously detected a segment response time-out. This only occurs if the messages are assigned to the same segmentation unit.

The most common scenario is for a source A to receive a segment response timeout when a single segment Type11 message is transmitted to an unresponsive destination B. Following this event, source A sends the last segment of the next Type11 message twice to destination C.

**Impact:** Duplication of a Type11 segment causes destination to receive additional payload. This could lead to one of the following:
- Partial message re-assembled at destination may time-out, causing an incomplete message with error status.
- If a single segment message is duplicated, destination does not detect an error, but receives a duplicate message.
- A segment of message A may be combined with subsequent segment(s) of message B and create a new message C which has a bad payload likely undetectable by the destination at the time of reception.

**Workaround:** Possible workarounds are listed below, each with their own limitations:

**Workaround #1**
Use Type9 data streaming protocol if data delivery does not require a response. Type9 is unaffected by the duplication issue and provides a much better performance than Type11 messages. Type9 may cause loss of packets without the sender knowing it if SRIO fabric fails to deliver data. Protocols such as Ethernet TCP/IP using sequence numbers can handle this potential temporary loss of data.

**Workaround #2**
Use Type11 multicast by setting message descriptor field MD[MM]=1. This has a limit of max 256B of payload. MD[EMG/MG/ML] (MD[Extended Multicast group/Multicast group/Multicast list]) must be set accordingly to indicate a single destination.

**Workaround #3**
Allocate a segmentation engine, using arbitration group (AG) assignment with RMan Message Manager Segmentation Execution Privilege Register MMSEPR0, per destination. This works under the assumption that sending a duplicate segment to an unresponsive destination (link down), does not have any adverse effects. This first restriction is a limit of 4 AGs (4 potential destinations that can cause time-out). The second restriction is the desired arbitration between messages to these destinations may not be as desired. The details are discussed below:

**Workaround #3 - Details**

To separate outbound RMan traffic for two destinations onto separate segmentation engines, do the following:

1. Assign RMan sub-portal 0/1 for messages to destination A/B respectively. This implies using a separate frame/work queue for each destination. RMan arbitrates using round-robin between sub-portals to balance traffic between SRIO ports, if used this way.
2. Use a different AG on each sub-portal. For example, AG0 (WQ0) for destination A and AG1 (WQ1) for destination B. Unique arbitration groups are needed to designate traffic to a specific segmentation engine.
3. Program engines for arbitration groups using register MMSEPR0. For example, MMSEPR0= 01010202h. This allows destination A to use segmentation engine 0/1 (AG0 only), and destination B to use segmentation engine 2/3 (AG1 only).

**Fix plan:** No plans to fix
A-009006: Loss of inbound RMan hardware reassembly contexts for Type11

Affects: RMAN

Description: RMan receives inbound messages (up to 4KB in size) as 1 to 16 messages segments. When the first segment of a multi-segment message is received, a hardware context is opened to track reassembly of the entire message in memory. If all segment numbers of a Type11 inbound message are received, then the hardware context is closed and set back into invalid state ready to receive a new message. However, if the last segment of a multi-segment message has a different message length than the original starting message segment, loss of one reassembly hardware context occurs. The most likely source of this event would be a badly generated RapidIO message header or a duplication/loss of a Type11 segment by transmitter.

Loss of one inbound reassembly hardware context occurs for a flow based on the MMRCARn setup. This is detected as inbound reassembly idle (MMSR[IMUB]=0) and open hardware reassembly contexts (MMSR[OHWC]>0). The loss of contexts is compounded by additional errors until the dedicated/generic flow count has been exhausted. An exhausted dedicated/generic flow may result in dropped packets or retried messages.

Impact: One or more losses of inbound hardware reassembly context for Type11 may eventually result in zero contexts available as defined by MMRCARn. When this occurs, Type11 has retried for an exhausted flow or generic context.

Workaround: The following are workarounds available for this issue:

Workaround #1
User may switch to use Type9 data packets if data delivery notification is not required since it is unaffected by the errata.

Workaround #2
User may redirect inbound reassembly errors to a unique Type11 error frame queue by setting MMR[EFQ]=1. If FD[MFE]=1 and FD[SRT] =0, potential loss of inbound reassembly context may have occurred. User should quiesce RMan by setting MMR[MMQ]=1 and stop inbound traffic from sender to check if MMSR[OHWC]>0. To reclaim lost reassembly H/W contexts, software must perform a soft reset of RMan as defined by register field MMR[SR]. A soft reset of RMan does not cause the SRIO link to go down. The Type11 messages in this scenario is retried.

Fix plan: No plans to fix
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.

Freescale, the Freescale logo, AltiVec, C-5, CodeTest, CodeWarrior, ColdFire, ColdFire+, C-Ware, Energy Efficient Solutions logo, Kinetis, mobileGT, PowerQUICC, Processor Expert, QorIQ, Qorivva, StarCore, Symphony, and VortiQa are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Airfast, BeeKit, BeeStack, CoreNet, Flexis, Layerscape, MagniV, MXC, Platform in a Package, QorIQ Qonverge, QUICC Engine, Ready Play, SafeAssure, SafeAssure logo, SMARTMOS, Tower, TurboLink, Vybrid, and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2015 Freescale Semiconductor, Inc.