MVME7100

Product Errata

P/N: 6806800K31C

October 2019

SMART™ Embedded Computing

Trademarks

The stylized "S" and "SMART" is a registered trademark of SMART Modular Technologies, Inc. and “SMART Embedded Computing” and the SMART Embedded Computing logo are trademarks of SMART Modular Technologies, Inc. All other names and logos referred to are trade names, trademarks, or registered trademarks of their respective owners. These materials are provided by SMART Embedded Computing as a service to its customers and may be used for informational purposes only.

Disclaimer*

SMART Embedded Computing (SMART EC) assumes no responsibility for errors or omissions in these materials. **These materials are provided "AS IS" without warranty of any kind, either expressed or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.** SMART EC further does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SMART EC shall not be liable for any special, indirect, incidental, or consequential damages, including without limitation, lost revenues or lost profits, which may result from the use of these materials. SMART EC may make changes to these materials, or to the products described therein, at any time without notice. SMART EC makes no commitment to update the information contained within these materials.

Electronic versions of this material may be read online, downloaded for personal use, or referenced in another document as a URL to a SMART EC website. The text itself may not be published commercially in print or electronic form, edited, translated, or otherwise altered without the permission of SMART EC.

It is possible that this publication may contain reference to or information about SMART EC products, programming, or services that are not available in your country. Such references or information must not be construed to mean that SMART EC intends to announce such SMART EC products, programming, or services in your country.

Limited and Restricted Rights Legend

If the documentation contained herein is supplied, directly or indirectly, to the U.S. Government, the following notice shall apply unless otherwise agreed to in writing by SMART Embedded Computing.

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data clause at DFARS 252.227-7013 (Nov. 1995) and of the Rights in Noncommercial Computer Software and Documentation clause at DFARS 252.227-7014 (Jun. 1995).

SMART Embedded Computing, Inc.

2900 S. Diablo Way, Suite 190

Tempe, Arizona 85282

USA

*For full legal terms and conditions, visit *www.smartembedded.com/ec/legal*

Table of Contents

About this Manual 5

1 Known Issue(s) 7

1.1 Introduction 7
1.2 VMEbus Lock-up 7
1.2.1 Problem 7
1.2.2 Symptom 7
1.2.3 Additional Information 7
1.3 Workarounds 8

Table of Contents

About this Manual

Abbreviations

This document uses the following abbreviations.

Abbreviation	Definition
PCI	Peripheral Component Interconnect
VME	VersaModule Eurocard

Summary of Changes

This manual has been revised and replaces all prior editions.

Part Number	Publication Date	Description
6806800K31C	October 2019	Re-branded to SMART Embedded Computing template. Converted source file from Word document to FrameMaker.
6806800K31B	June 2014	Re-branded to Artesyn template
6806800K31A	December 2009	Initial version.

Known Issue(s)

1.1 Introduction

This document describes a product issue that affects all MVME7100 product variants and provides ways to work around the issue.

1.2 VMEbus Lock-up

1.2.1 Problem

A lock-up occurs when a VME read cannot complete. For example, if VME board A is trying to read memory on VME board B while VME board B is streaming writes to the VMEbus, a lockup can occur. If the writes fill the buffer in the Tsi148 and backup into the outbound queues of the PCI Express devices, then the read data will be blocked by the writes and the read cannot complete. The pending read on the VMEbus will prevent the stored writes from completing and there is a deadlock. If the writes do not backup in the PCI Express devices, the Tsi148 will allow the read to pass the writes and a deadlock would not occur.

1.2.2 Symptom

The VMEbus will lock-up and the cycle will be terminated with a VME BERR*.

1.2.3 Additional Information

This lock-up occurs because there is a fundamental incompatibility between the PCI bus and VMEbus. The PCI specifications require bridges to flush outbound data before completing an inbound read. Because the original VMEbus specification did not define a retry function, the flush before read requirement creates a potential lock-up situation. To avoid this lock-up situation, internally-designed host bridges included an option to disable the flush before read function. Some PLX PCI-to-PCI bridges also allow the flush before read function to be disabled. The Marvell host bridges allow the flush before read function to be disabled. Current PCI Express bridges and switches do not allow the flush before read function to be disabled.

The PCI Express bridges, switches and the MC8641D processor used on the MVME7100 do not allow the flush before read function to be disabled. The packet-based PCI Express protocols place the read response packet in the outbound queue requiring any queued writes to complete before the read can complete.

The PCI specifications define a relaxed ordering feature that does allow the read completion to pass the posted writes; however, it is not guaranteed that reads will always pass writes. Also, the Tsi148 does not set the relaxed ordering bit and Tsi384 will not allow the read completion to pass the writes unless the RO bit is set. The PLX switch does not support relaxed ordering.

1.3 Workarounds

A guaranteed way to avoid this problem in new designs which involve moving data between boards over VME is to design the application to move data between boards over VME using “writes” only.

Existing applications which are susceptible to this problem should use one of the following workarounds:

Table 1-1 List of Workarounds

Option	Description
1	Later versions of the VME standard support a retry function. If the boards that are locking up have the Tsi148 chip, the retry function can be enabled. The retry function allows the writes to progress but the read will be retried. However, if the writes are continuous, there will be a read completion time out. While it is possible to create a simple test where board A writes continually to another board B while board B reads continually from board A, where one would run into a read completion timeout, it is quite likely that this will not happen in a practical application. If board A moved on to other chores even at intervals exceeding 100K words, the read completion timeouts can be avoided. Deadlock detection is enabled by setting the DLT field in the VME Bus Control Register.
2	Use the "Device Wants Bus (DWB)" feature of the Tsi148. In this scheme, any board that starts a large block of writes should first claim the VME bus before starting the writes, and release the bus when done. The DWB field is in the VME Bus Master Control Register. The Writer should use this along with the "Device Has Bus (DHB)" bit in the same register to get the bus, check that it has the bus, and release the bus when done. The Tsi148 has errata related to the use of DWB. The user should be aware of these errata if this option is used.
3	The DMA controller in the Tsi148 can be used when large blocks of data are moved. This is more efficient and avoids the lockup conditions. For this to work, DMA would have to be used instead of programmed I/O for any block size above 40 words to avoid bumping into the write post buffer limit of the TSI148.
4	If multiple writes are streamed to the VMEbus, a read should be done after every 32 writes. This will prevent the outbound queue in the Tsi148 from filling up.