ARM Processor Cortex™-M0+

Product Revision r0

Software Developers Errata Notice

Non-Confidential - Released

Software Developers Errata Notice

Proprietary notice

Confidentiality status

This document is **Non-Confidential** but any disclosure by you is subject to you providing notice to and the acceptance by the recipient of, the conditions set out in this notice.

Web Address

http://www.arm.com

Feedback on content

If you have any comments on content, then send an e-mail to errata@arm.com. Give:

- the document title
- the document number, ARM-EPM-018879
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Release Information

Errata are listed in this section if they are new to the document, or marked as "updated" if there has been any change to the erratum text in Chapter 2. Fixed errata are not shown as updated unless the erratum text has changed. The summary table in section 2.2 identifies errata that have been fixed in each product revision.

20 Dec 2012: Changes in Document v4

No new or updated errata in this document version.

See the Summary Table for errata fixes in the latest product revision.

18 Jul 2012: Changes in Document v3

Page	Status	ID	Cat	Rare	Summary of Erratum
9	New	789069	CatC		A debugger write to the I/O port might be corrupted during a processor write

03 May 2012: Changes in Document v2

Page	Status	ID	Cat	Rare	Summary of Erratum
8	New	777019	CatC		Processor executing at HardFault priority can go into Lockup if an NMI occurs during a waited debug transaction

(First released version of this document was v2.0)

Contents

Chapter 1.
Introduction

This chapter introduces the errata notice for the ARM Cortex-M0+ processor.

1.1. Scope of this document

This document describes errata categorized by level of severity. Each description includes:

- the current status of the defect
- where the implementation deviates from the specification and the conditions under which erroneous behavior occurs
- the implications of the erratum with respect to typical applications
- the application and limitations of a 'work-around' where possible

This document describes errata that may impact anyone who is developing software that will run on implementations of this ARM product.

1.2. Categorization of errata

Errata recorded in this document are split into the following levels of severity:

Table 1 Categorization of errata

Errata Type	Definition
Category A	A critical error. No workaround is available or workarounds are impactful. The error is likely to be common for many systems and applications.
Category A(rare)	A critical error. No workaround is available or workarounds are impactful. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category B	A significant error or a critical error with an acceptable workaround. The error is likely to be common for many systems and applications.
Category B(rare)	A significant error or a critical error with an acceptable workaround. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category C	A minor error.

Chapter 2.
Errata Descriptions

2.1. Product Revision Status

The r*n*p*n* identifier indicates the revision status of the product described in this book, where:

r*n* Identifies the major revision of the product.

p*n* Identifies the minor revision or modification status of the product.

2.2. Revisions Affected

Table 2 below lists the product revisions affected by each erratum. A cell marked with **X** indicates that the erratum affects the revision shown at the top of that column.

This document includes errata that affect revision r0 only.

Refer to the reference material supplied with your product to identify the revision of the IP.

Table 2 Revisions Affected

ID	Cat	Rare	Summary of Erratum	r0p0	r0p1
789069	CatC		A debugger write to the I/O port might be corrupted during a processor write	X	
777019	CatC		Processor executing at HardFault priority might enter Lockup state if an NMI occurs during a waited debugger transaction	X	

2.3. Category A

There are no errata in this category

2.4. Category A (Rare)

There are no errata in this category

2.5. Category B

There are no errata in this category

2.6. Category B (Rare)

There are no errata in this category

2.7. Category C

777019: Processor executing at HardFault priority might enter Lockup state if an NMI occurs during a waited debugger transaction

Category C

Products Affected: Cortex-M0+.

Present in: r0p0

Description

A debugger can perform memory accesses through the Cortex-M0+ processor bus matrix while the processor is running.

Because of this erratum, the processor might erroneously enter Lockup state if a debugger-initiated access on the AHB-Lite master port is subject to wait states while the processor is running, executing at HardFault priority and taking a Non Maskable Interrupt (NMI). Under very specific timing conditions, the processor might incorrectly stack a ReturnAddress of 0xFFFFFFFE on NMI entry. On NMI return, the processor unstacks the incorrectly stacked ReturnAddress and enters Lockup state at HardFault priority.

Configurations affected

The processor must be configured to both:

- Include debug.
- Perform most instruction fetches as word accesses.

In addition, the processor must be part of a system where both:

- The processor AHB-Lite master port can be subject to wait states.
- An NMI can be asserted to the processor.

Conditions

The following timing-specific conditions must all be met:

- The processor is running (not halted in Debug state) and is executing at HardFault priority.
- The processor executes a single-cycle instruction at a word-aligned address.
- The debugger performs an access through the AHB-Lite master port that is subject to wait states.
- An NMI becomes pending.

Implications

The processor stops executing the code in the HardFault handler and enters Lockup state at HardFault priority as if a fault had occurred.

Workaround

The debugger can work around this erratum by halting the processor in Debug state before performing accesses outside the Private Peripheral Bus (PPB) region of the memory map.

789069: A debugger write to the I/O port might be corrupted during a processor write

Category C

Products Affected: Cortex-M0+.

Present in: r0p0

Description

A debugger can perform memory accesses through the Cortex-M0+ processor bus matrix while the processor is running.

Because of this erratum, a debugger write to the I/O port might be corrupted if it occurs while the processor is executing a write. The processor write completes successfully. However, under specific timing conditions, the matrix might incorrectly replace the debugger write data with the value zero.

This erratum does not affect debugger writes outside the I/O port region of the memory map, or debugger reads.

Configurations affected

The processor must be configured to both:

- Include debug.
- Include the I/O port.

Conditions

The following timing-specific conditions must all be met:

- The processor is running (not halted in Debug state).
- The debugger performs a write within the I/O port region of the memory map.
- The processor performs a write.

Implications

The debugger might corrupt the targeted memory or configure the targeted device incorrectly.

Workaround

The debugger can work around this erratum by halting the processor in Debug state before performing writes to the I/O port region of the memory map.