ARM720T r4pN Errata List

IP Products Division

Document number: ARM720T-PRDC-003802 4.0
Date of Issue: 20 March 2009

Abstract

This document describes the known errata in the ARM720T r4p3 design.

Keywords

ARM720T, errata

This is a working document throughout the product lifecycle and, as such, the content may be modified as new information is uncovered.

.

Contents

.

.

1 ABOUT THIS DOCUMENT

1.1 References

This document refers to the following documents.

Ref.	Document No	Author(s)	Title
1	ARM DDI 0229 A	ARM	ARM720T r4p3 Technical Reference Manual

1.2 Scope

This document describes the errata discovered in the implementation of the ARM720T r4pN, categorised by level of severity. Each description includes:

- where the implementation deviates from the specification
- the conditions under which erroneous behaviour occurs
- the implications of the erratum with respect to typical applications
- the application and limitations of a ‘work-around’ where possible
- the status of corrective action.

1.3 Terms and Abbreviations

This document uses the following terms and abbreviations.

Term	Meaning
AVS	ARM Validation Suite

.

2 CATEGORISATION OF ERRATA

Errata recorded in this document are split into three groups:

Category 1	Features which are impossible to work around and severely restrict the use of the device in all or the majority of applications rendering the device unusable.
Category 2	Features which contravene the specified behaviour and may limit or severely impair the intended use of specified features but does not render the device unusable in all or the majority of applications.
Category 3	Features that were not the originally intended behaviour but should not cause any problems in applications.
Implementation	Errata that are of particular interest to those implementing the product and that have no software implications
System	Errata or possible issues that have system implications and therefore should be considered by system designers
Testchip	Errata that are identified with components in the Testchip RTL that make up the validation testbench such as the AHB components.

2.1 Errata Summary

The errata associated with this product are categorised in the following way. Numbers in brackets after the errata description indicate the order in which the errata were found chronologically. Rev4p1 does not exist as a delivered layout.

		Rev4p0	Rev4p2	Rev4p3
Category 1	[1] Possible HOLD time violation in MMU	X	Ok	Ok
Category 2	[2] Incorrect HSIZE indication on AHB bus during a non cacheable thumb fetch	X	Ok	Ok
	[3] Incorrect HLOCK timing	X	Ok	Ok
	[4] Incorrect behaviour when an Interrupt occurs in Debug state	X	X	X
	[6] Abort link register not properly updated on data aborts in Thumb state	X	X	Ok
Category 3	[5] Incorrect value of EmbeddedICE-RT version number read through JTAG interface	X	X	X
System	[7] LDC/LDCL/STC/STCL instructions incorrectly decoded when non-indexed addressing mode is used	X	X	X
	[8] Sequenial MRC instructions may not execute correctly	X	X	X

.

3 CATEGORY 1 ERRATA

3.1 Possible HOLD time violation in MMU

3.1.1 Summary

Layout delivered to the lead partner was not validated with the appropriate spice parameters. After running all PFC, it appears there is a hold time marginality on the MMU address bus that required insertion of more margin. This hold marginality appears in best case corners.

3.1.2 Description

Layout with the added buffer has following margin:

Pfc1 tc1	0.294 ns
Pfc1 wc1	0.537 ns
Pfc1 bc1	0.089 ns
Pfc1 tc2	0.398 ns
Pfc1 wc2	0.837 ns
Pfc1 bc2	0.181 ns
Pfc1 wc3	0.556 ns
Pfc1 bc3	0.090 ns
Pfc1 wc4	0.849 ns
Pfc1 bc4	0.171 ns
Pfc2 tc1	0.191 ns
Pfc2 wc1	0.332 ns
Pfc2 bc1	0.126 ns
Pfc2 wc2	0.388 ns
Pfc2 bc2	0.081 ns
Pfc2 tc3	0.470 ns
Pfc2 wc3	0.902 ns
Pfc2 bc3	0.284 ns
Pfc2 wc4	1.014 ns
Pfc2 bc4	0.220 ns
Pfc3 tc1	0.110 ns
Pfc3 wc1	0.195 ns
Pfc3 bc1	0.068 ns
Pfc3 wc2	0.226 ns
Pfc3 bc2	0.036 ns
Pfc3 tc3	0.247 ns

.

Pfc3 wc3	0.431 ns
Pfc3 bc3	0.153 ns
Pfc3 wc4	0.502 ns
Pfc3 bc4	0.112 ns

3.1.3 Implications

The MMU will not do address translation correctly.

3.1.4 Workaround

None, other than not using the MMU.

3.1.5 Impacted revisions

This erratum impacts the following revision of ARM720T: rev4p0, delivered to the lead partner only.

.

4 CATEGORY 2 ERRATA

4.1 Incorrect HSIZE indication on AHB bus during a non cacheable thumb fetch

4.1.1 Summary

Incorrect HSIZE indication on AHB bus during a non cacheable thumb fetch.

4.1.2 Description

During a non-cacheable Thumb fetch, when a 32 bit access is performed on the AHB, HSIZE = 01, indicating a 16 bit access, whilst it should be HSIZE = 10, indicating a 32 bit access, since all non cacheable thumb fetches are going through a prefetch buffer and assume an aligned 32 bit transfer.

4.1.3 Implications

Incorrectly fetched data within a memory system which returns 16 bit data, as incorrectly requested, during a non-cacheable thumb opcode fetch.

4.1.4 Workarounds

The following possible workarounds exist:

1. Either, use a 32 bit memory sub-system returning 32 bit word for a 16 bit opcode fetch . In other words, ignore the HSIZE indication when the HPROT[0] signal indicates an opcode fetch and always return a 32 bit value.
2. Or, only use thumb state in cache able memory with cache on
3. Or, do not use thumb state

4.1.5 Impacted revisions

This erratum impacts the following revision of ARM720T: rev4p0, delivered to the lead partner only.

.

4.2 Incorrect HLOCK timing

4.2.1 Summary

Incorrect HLOCK timing on AHB bus with the bus in a non 1:1 ratio. The HLOCK signal may be asserted in the same cycle as the address it refers to, when HCLKEN is being negated/asserted to lower the effective bus frequency.

4.2.2 Description

Incorrect HLOCK timing on AHB bus with the bus in a non 1:1 ratio, ie when HCLKEN is not always asserted. Correct timing is to have the HLOCK asserted one cycle before the address it refers to. The erratum is that the HLOCK signal may be asserted in the same cycle as the address it refers to.

4.2.3 Implications

In a multi-master configuration, with a clock ratio not 1:1 , and with the ARM720T not being the highest priority master on the bus, a semaphore operation will be corrupted.

4.2.4 Workarounds

-1- use 1:1 clock ratio

-2- do not use multi master configuration

-3- have the ARM720T r4p0 being the highest priority master on the bus

-4- do not do semaphore operations

4.2.5 Impacted revisions

This erratum impacts the following revision of ARM720T: rev4p0, delivered to the lead partner only.

.

4.3 Incorrect behaviour when an interrupt (FIQ/IRQ) is asserted in debug

4.3.1 Summary

Interrupts are not masked correctly when the core is in debug state, so that if the core attempts to execute 'at-speed' an instruction that lasts more then 6 cycles (LDM, STM) whilst an interrupt is asserted, this instruction cannot finish properly.

4.3.2 Description

When the following conditions are reached:

- The core is in debug state, and
- An interrupt (FIQ or IRQ) is asserted at any time when the core is in debug state and is still asserted when the core goes into 'at speed' state, and
- The core starts to execute ‘at speed’ an LDM or STM that lasts more then 6 core cycles (instruction loaded via JTAG),

The core should ignore the interrupt and go back to halt mode, once the LDM or STM is finished. However, the core starts to execute the LDM instruction, and after 6 data transfers, it comes back to halt debug mode without finishing the data transfers.

The *core module* ignores properly the interrupt request so that it does not enter interrupt mode, but the interrupt is not masked correctly in the *debug module* and corrupts the execution of at least 7 core cycles or more. For shorter instructions, the problem is not observed.

4.3.3 Implication

When the debugger needs to download some code by using LDM/STM instructions, the code might not be properly loaded if an interrupt occurs. Note that all ARM debug tools disable interrupts during code download – the errata will never be seen if using ARM tools.

4.3.4 Corrective Action

Disable interrupts in debug mode:

Before loading LDM instructions via JTAG, the debugger must set INTDIS bit in Debug Control Register via JTAG so that all interrupts are disabled in debug mode.

4.3.5 Impacted revisions

This erratum impacts the following revision of ARM720T: rev4p0-p3

.

4.4 Abort link register not properly updated on data aborts in Thumb state

4.4.1 Summary

If the processor is in Thumb state and executing the code sequence STR, STMIA or PUSH followed by a PC-relative load, and the STR, STMIA or PUSH is aborted, the PC is saved to the abort link register in only word resolution, instead of half-word resolution.

4.4.2 Conditions

The processor must be in Thumb state, and the following sequence must occur:

```
<any instruction>

<STR, STMIA, PUSH>    <---- data abort on this instruction

LDR rn, [pc,#offset]
```

In this case the PC is saved to the link register R14_abt in only word resolution, not half-word resolution. The effect is that the link register holds an address that could be #2 less than it should be, so any abort handler could return to one instruction earlier than intended.

4.4.3 Implications

In a system that does not use Thumb state, there will be no problem.

In a system that uses Thumb state but does not use data aborts, or does not try to use data aborts in a recoverable manner, there will be no problem.

In a system that uses Thumb state, and uses data aborts in a recoverable manner, such as in a demand paging environment, where the STR, STMIA or PUSH aborts for paging reasons, the code will patch up the MMU and re-execute the STR, STMIA or PUSH. What matters in this case is the instruction preceding the STR etc, and whether this can be re-executed harmlessly. As this cannot be predicted with certainty, it is likely to be a problem.

4.4.4 Workaround

The workaround is to ensure that a STR, STMIA or PUSH cannot precede a PC-relative load. One method for this is to add a NOP before any PC-relative load instruction. However this is not something currently supported by our software tools, and would have to be done manually.

4.4.5 Impacted revisions

This erratum affects the following revisions of ARM720T: r4p0 to r4p2.

.

5 CATEGORY 3 ERRATA

5.1 Incorrect value of EmbeddedICE-RT version number read through JTAG interface

5.1.1 Summary

The EmbeddedICE-RT version is 7 for the ARM720T rev4, but the value read via JTAG is 4.

5.1.2 Description

Bits 31:28 of the Debug Communications Channel Control Register contain a fixed pattern that denotes the EmbeddedICE-RT version number implemented within the ARM720T r4p2. This should have a value of b0111, whether read through JTAG or CP14.

However, a read of this register through JTAG scan chain 2 results in a value of b0100 being read, which is incorrect.

5.1.3 Implication

This has an implication for debug tools only.

5.1.4 Workarounds

No workaround is necessary as it still works with debug tools.

5.1.5 Corrective Action

No corrective action is performed on rev4p0-p3.

Correct operation of the ARM720T r4p2 with this bug has been demonstrated with the following debug tools:

- Multi-ICE with armsd/ADW/AXD
- RealView-ICE with RealViewDebugger

5.1.6 Impacted revisions

This erratum impacts the following revision of ARM720T: rev4p0-p3

.

6 CATEGORY SYSTEM

6.1 LDC/LDCL/STC/STCL instructions incorrectly decoded when non-indexed addressing mode is used

6.1.1 Conditions

The core is unable to decode LDC/LDCL/STC/STCL instructions without base register write back and non-indexed addressing mode. Base Register write-back is controlled by bit-21 (W-bit) of the instruction opcode.

6.1.2 Implications

These specific Coprocessor load and store instructions will fail for systems that implement external coprocessors.

6.1.3 Workaround

Use an LDC/LDCL/STC/STCL with immediate post-indexed addressing mode. This should then be followed by a ADD (U bit-23 clear) or SUB (U bit-23 set) instruction to add or subtract four times the value of the immediate offset to or from the base register. This will incur a cost of 1 additional instruction per affected LDC/LDCL/STC/STCL instruction.

6.1.4 Impacted revisions

This erratum affects the following revisions of ARM720T: r4p0 to r4p3.

.

6.2 Sequential MRC instructions may not execute correctly

6.2.1 Conditions

If two or more MRC instructions are fetched sequentially, the second and any subsequent MRC instructions may not be correctly executed by the core.

An MRC instruction takes the form of:

MRC{<cond>} <coproc>, <opcode_1>, <Rd>, <CRn>, <CRm>{, <opcode_2>}

The bug is only encountered if the value of <opcode_1> is set to 3'bx1x and only if EXTCPA/EXTCPB are being driven low by the coprocessor a cycle earlier with respect to CPnOPC/CPnMREQ going low. The bug only affects external coprocessor accesses and not internal CP15 operations.

Figure 1, shows how EXTCPA/EXTCPB are driven in the AVS testbench, causing the bug not to trigger.

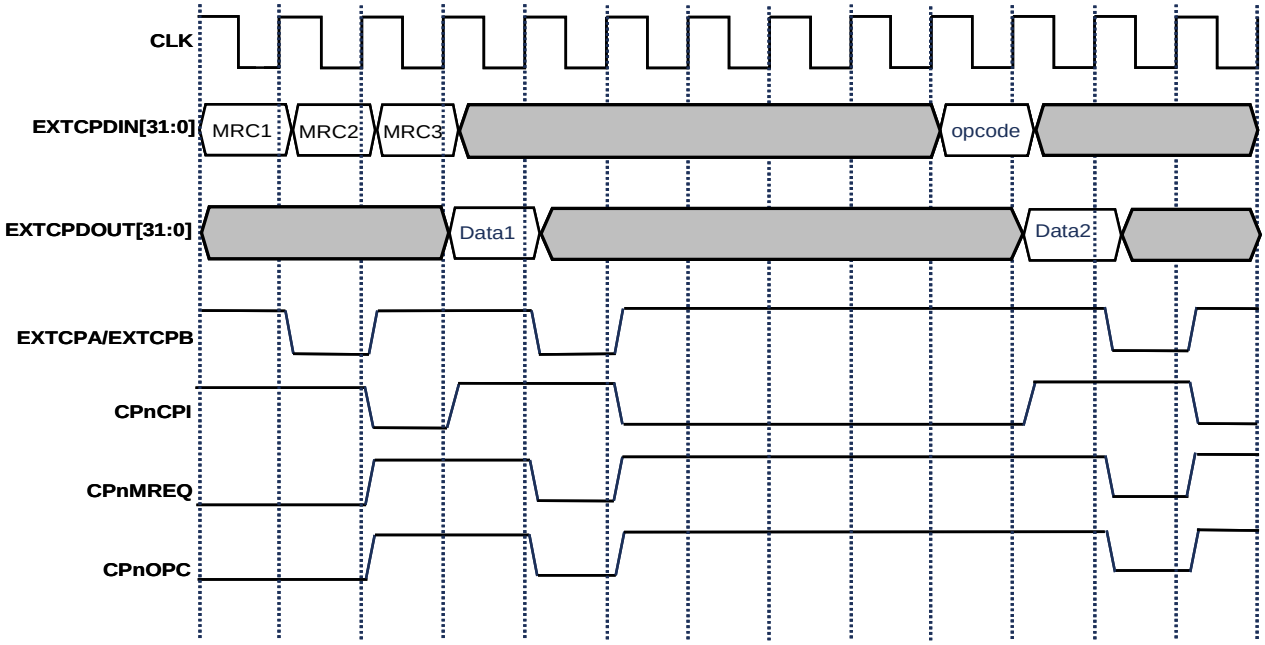

Fig 1: EXTCPA/EXTCPB as driven in the AVS testbench

Figure 2, shows how the bug is triggered, e.g. when EXTCPA/EXTCPB are driven a cycle earlier by the coprocessor.

.

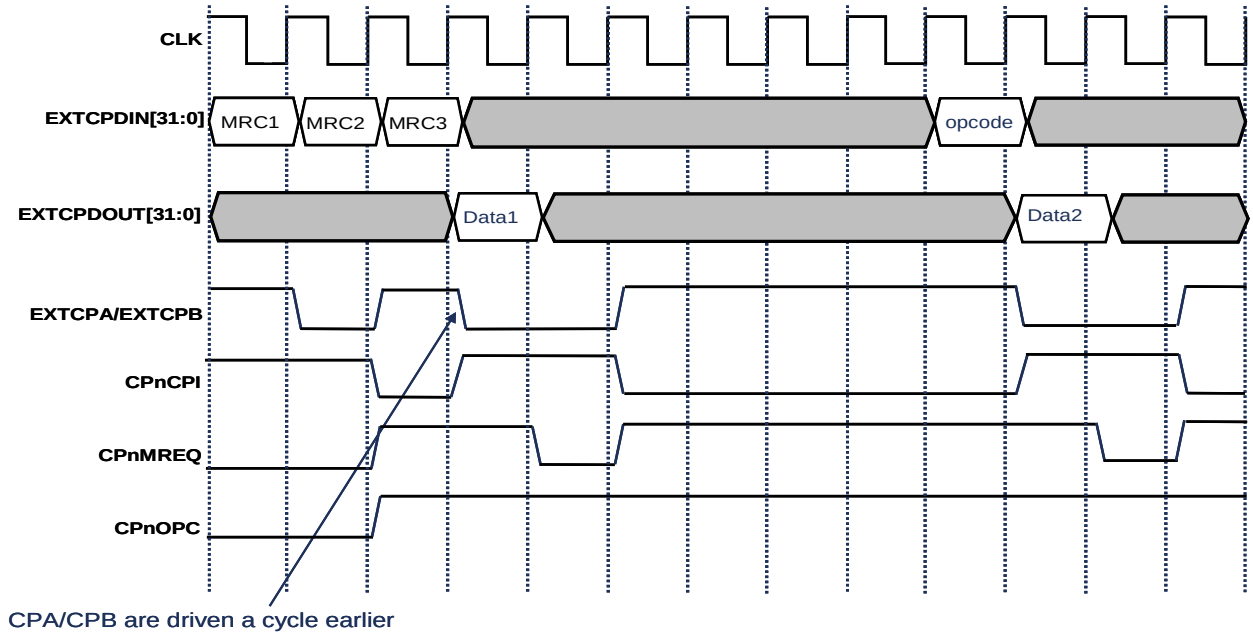

Fig 2: EXTCPA/EXTCPB driven a cycle earlier

6.2.2 Implications

This only has implications when external coprocessors are used.

6.2.3 Workaround

There are two possible workarounds:

1: Delay EXTCPA/EXTCPB inputs by a full clock cycle, e.g. so that they are taken low when CPnOPC/CPnMREQ go low.

2: Insert a NOP instruction between each MRC instruction that appear in program code, there will be an overhead of an additional instruction for each sequential MRC instruction fetched and executed by the core.

6.2.4 Impacted revisions

This erratum affects the following revisions of ARM720T: r4p0 to r4p3.

.