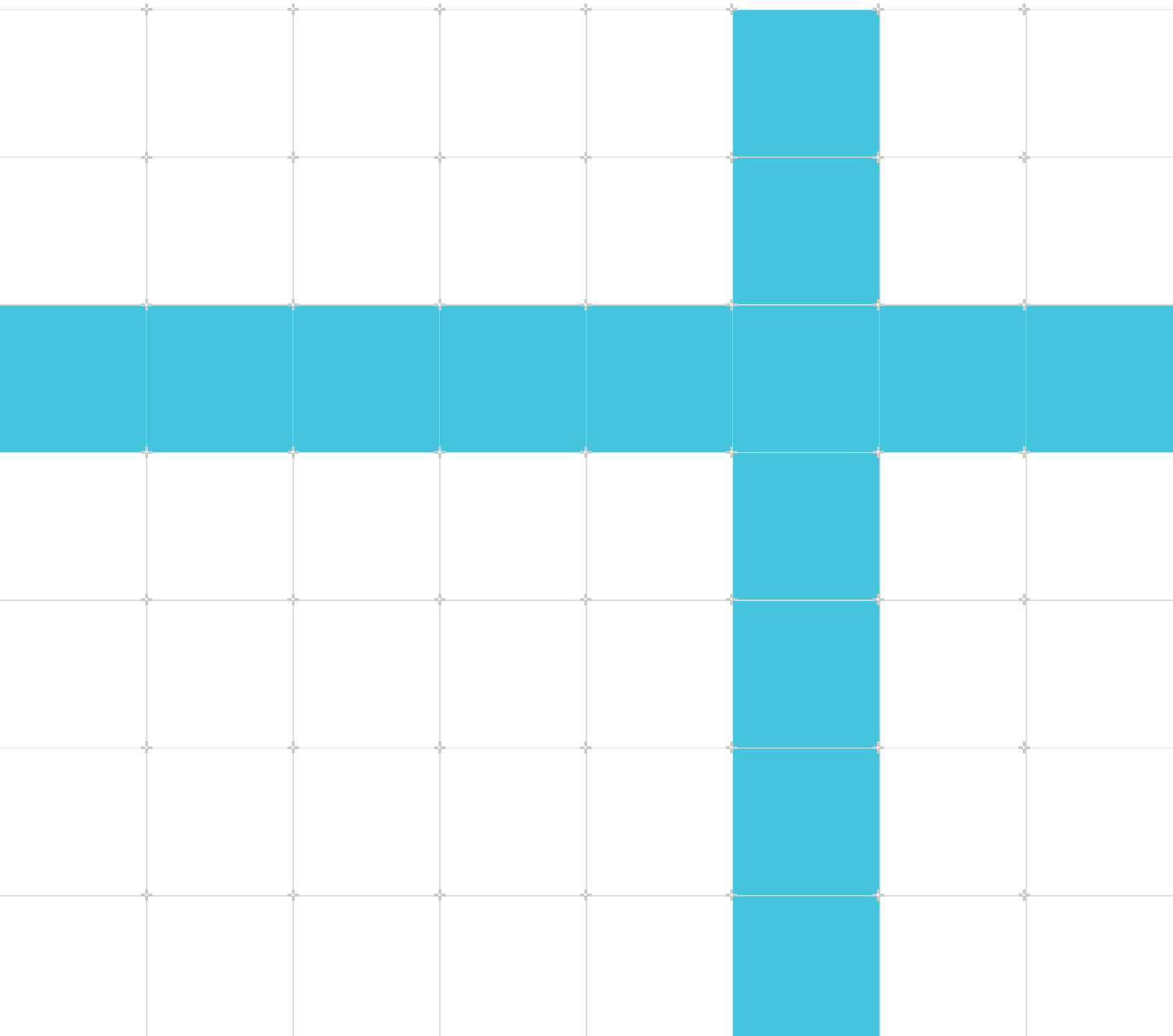
Arm® Architecture Reference Manual Supplement, Armv9-A

Known issues in Issue B.a

Non-Confidential

Issue 02
102527_B.a_02_en



Arm® Architecture Reference Manual Supplement, Armv9-A

Known issues in Issue B.a

Release information

Document history

Issue	Date	Confidentiality	Change
A.a-00	14 May 2021	Non-Confidential	Known Issues in Arm® Architecture Reference Manual Supplement, Armv9-A, Issue A.a, as of 14 May 2021
A.a-01	30 June 2021	Non-Confidential	Known Issues in Arm® Architecture Reference Manual Supplement, Armv9-A, Issue A.a, as of 18 June 2021
A.a-02	30 July 2021	Non-Confidential	Known Issues in Arm® Architecture Reference Manual Supplement, Armv9-A, Issue A.a, as of 23 July 2021
A.a-03	30 September 2021	Non-Confidential	Known Issues in Arm® Architecture Reference Manual Supplement, Armv9-A, Issue A.a, as of 17 September 2021
A.a-04	29 October 2021	Non-Confidential	Known Issues in Arm® Architecture Reference Manual Supplement, Armv9-A, Issue A.a, as of 22 October 2021
A.a-05	30 November 2021	Non-Confidential	Known Issues in Arm® Architecture Reference Manual Supplement, Armv9-A, Issue A.a, as of 19 November 2021
A.a-06	31 January 2022	Non-Confidential	Known Issues in Arm® Architecture Reference Manual Supplement, Armv9-A, Issue A.a, as of 7 January 2022
B.a-00	28 February 2022	Non-Confidential	Known Issues in Arm® Architecture Reference Manual Supplement, Armv9-A, Issue B.a, as of 18 February 2022
B.a-01	29 April 2022	Non-Confidential	Known Issues in Arm® Architecture Reference Manual Supplement, Armv9-A, Issue B.a, as of 22 April 2022
B.a-02	31 May 2022	Non-Confidential	Known Issues in Arm® Architecture Reference Manual Supplement, Armv9-A, Issue B.a, as of 20 May 2022

Proprietary Notice

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm® welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1 Introduction

1.1 Conventions

The following subsections describe conventions used in Arm documents.

Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm® Glossary for more information: developer.arm.com/glossary.

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
italic	Citations.
bold	Interface elements, such as menu names. Signal names. Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace bold`	Language keywords when used outside example code.
`monospace underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`<and>`	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.
	Recommendations. Not following these recommendations might lead to system failure or damage.
	Requirements for the system. Not following these requirements might result in system failure or damage.
	Requirements for the system. Not following these requirements will result in system failure or damage.

Convention	Use
 Note	An important piece of information that needs your attention.
 Tip	A useful tip that might make it easier, better or faster to perform a task.
 Remember	A reminder of something important that relates to the information you are reading.

1.2 Additional reading

This document contains information that is specific to this product. See the following documents for other relevant information:

Table 1-2: Arm publications

Document Name	Document ID	Licensee only
Arm® Architecture Reference Manual Supplement, Armv9-A, Issue B.a	DDI 0608B.a	No


Note

Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its documents when used with any other PDF reader.

Adobe PDF reader products can be downloaded at http://www.adobe.com

1.3 Other information

See the Arm website for other relevant information.

- Arm® Developer.
- Arm® Documentation.
- Technical Support.
- Arm® Glossary.

2 Known issues

This document records known issues in the Arm Architecture Reference Manual Supplement, Armv9-A (DDI 0608), Issue B.a.

This document lists the known issues in the architectural rules content only. For a list of known issues in the register, instruction, and pseudocode XML, please see https://developer.arm.com/architectures/cpu-architecture/a-profile/exploration-tools.

Key

- C = Clarification.
- D = Defect.
- R = Relaxation.
- E = Enhancement.

2.1 The Transactional Memory Extension

No issues.

2.2 The Embedded Trace Extension (ETE)

The known issues in the ETE architecture are listed below:

2.2.1 D1416

In section D5.9 (Timestamp Packet), in the description of the COUNT field, the following text is added:

> When the COUNT field is not present, the cycle count value is **UNKNOWN**.

2.2.2 R1419

In section D6.9.16 (Timestamp Element), the rule that reads:

> R_{HZSYP} When a timestamp request occurs and ViewInst is inactive, the timestamp request is permitted to be delayed until the first of the following occurs:
>
> - ViewInst becomes active.
> - An Event element is generated.

is changed to read:

> R_{HZSYP} When a timestamp request occurs and ViewInst is inactive, the timestamp request is permitted to be delayed until after the first of the following occurs:
>
> - A P0 element is generated.
> - An Event element is generated.

2.2.3 D1438

In section D8.1.2 (System instructions), the text in R_{VGVTS} that reads:

> Instructions with CRn >= `0b1000` are **UNDEFINED**.

is changed to read:

> Instructions with CRn ≥ `0b1000` are not allocated for accessing trace unit registers.

2.3 The Trace Buffer Extension (TRBE)

The known issues in the TRBE architecture are listed below:

2.3.1 R1291

In section E1.2.2.4 (Faults), the following entry titled 'External abort on translation table walk or translation table update' is added to rule R_{QKLXR}:

> The translation of a virtual address to a physical address might generate an External abort on the translation table walk or translation table update, and is treated as a synchronous MMU fault. See also E1.2.2.5 External aborts.

In the same section, in rule R_{FSPBK}, the following text is added:

> If a write by the Trace Buffer Unit generates an External abort on a translation table walk or translation table update, it is **IMPLEMENTATION DEFINED** whether TRBSR_EL1.EA is set to `0b1` or unchanged.

In section E1.2.2.5 (External aborts), in rule R_{DJLWB}, the entry 'External abort on translation table walk or translation table update' is extended as follows:

> An External abort on a translation table walk or translation table update might be treated as a synchronous MMU fault. See E1.2.2.4 Faults.

2.3.2 D1393

In section E1.2.2 (Trace buffer management), rule R_{JLZDN} that reads:

> The Trigger Event trace buffer management event initiates a trace unit flush meaning other trace might be written to the trace buffer. This might cause a second trace buffer management event to be generated before Collection is stopped by the Trigger Event trace buffer management event.

is corrected to read:

> When the trigger mode is *Stop on Trigger*, a *Trigger Event* initiates a trace unit flush meaning more trace might be written to the trace buffer before any Collection is stopped by the Trigger Event trace buffer management event. This additional trace might cause a second trace buffer management event to be generated before the Trigger Event trace buffer management event.

2.3.3 D1399

In section E1.2.1.3 (Address translation disabled), the text in R_{FJKLW} that reads:

> If TRBLIMITR_EL1.nVM is `0b1`, TRBMAR_EL1 defines the memory type, and, as applicable, Cacheability, Shareability, and Device type attributes, for the stage 1 output addresses.

is corrected to read:

> If TRBLIMITR_EL1.nVM is `0b1`, then unless overridden by stage 2 controls, TRBMAR_EL1 defines the memory type, and, as applicable, Cacheability, Shareability, and Device type attributes, for the stage 1 output addresses.

In section E1.2.1.4 (Stage 2 translation), the example in I_{ZSDMR} that reads:

> - If the Effective value of HCR_EL2.DC in the owning translation regime is `0b1`, then stage 1 translation is disabled and the memory type produced by stage 1 is Normal Non-shareable, Inner Write-Back Cacheable Read-Allocate Write-Allocate, Outer Write-Back Cacheable Read-Allocate Write-Allocate, regardless of the value of SCTLR_EL1.C.

is extended to read:

> - If the Effective value of HCR_EL2.DC in the owning translation regime is `0b1`, then stage 1 translation is disabled and the memory type produced by stage 1 is Normal Non-shareable, Inner Write-Back Cacheable Read-Allocate Write-Allocate, Outer Write-Back Cacheable Read-Allocate Write-Allocate, regardless of the values of SCTLR_EL1.C and, if TRBLIMITR_EL1.nVM is `0b1`, TRBMAR_EL1.

2.4 The Branch Record Buffer Extension (BRBE)

The known issues in the BRBE architecture are listed below:

2.4.1 C1306

In section F1.1.9.1 (Filtering on type), rule R_{FJYDC} that currently reads:

> It is **IMPLEMENTATION DEFINED** whether Branch records are generated for the following instructions when the instruction is executed in a non prohibited region and if any of the following are true:

is updated to read:

> It is **IMPLEMENTATION DEFINED** whether Branch records are generated for the following unconditional direct branch instructions when the instruction is executed in a non-prohibited region and if any of the following are true:

Within the same section, the title of Table F1.10 is updated to 'Optional A64 unconditional direct branch instructions'.

2.4.2 R1391

Two new Rules items and one Information item are added to section F1.1.13.1 (Manual injection of Branch records):

> Rule: When a BRB INJ instruction is executed inside a BRBE Prohibited region and the contents of the Branch record injection data registers indicates an invalid record, it is **CONSTRAINED UNPREDICTABLE** whether a Branch record is injected to the Branch record buffer. An invalid record is one with BRBINFINJ_EL1.VALID set to `0b00`.
>
> Rule: When all of the following are true for a BRB INJ instruction, it is **CONSTRAINED UNPREDICTABLE** whether a Branch record is injected to the Branch record buffer, and if a Branch record is injected then the contents of the Branch record are **CONSTRAINED UNPREDICTABLE**:
>
> - The BRB INJ instruction is executed inside a BRBE Prohibited region.
> - The contents of the Branch record injection data registers indicates a valid record.
> - The other contents of the Branch record injection data registers indicate an incorrectly formatted record.
>
> Information: An example of an incorrectly formatted record is one where BRBINFINJ_EL1.VALID is `0b01` and BRBINFINJ_EL1.MPRED is `0b1`.

2.4.3 D1436

In section F1.1.10 (Branch record buffer operation), the text in rule R_{QKQZL} that reads:

> If any of the following are true, the physical offset is zero, otherwise the physical offset is the value of CNTVOFF_EL2:

is updated to read:

> If any of the following are true, the physical offset is zero, otherwise the physical offset is the value of CNTPOFF_EL2: