Arm® Architecture Reference Manual Supplement, Armv9-A

Known issues in Issue A.a

Non-Confidential

Issue 06
102527_A.a_06

Arm® Architecture Reference Manual Supplement, Armv9-A

Known issues in Issue A.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

Proprietary Notice

Confidentiality Status

Product Status

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` <u>`underline`</u>	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.
⚠ Caution	Recommendations. Not following these recommendations might lead to system failure or damage.
⚠ Warning	Requirements for the system. Not following these requirements might result in system failure or damage.
⚠ Danger	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 A.a	DDI 0608A.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 A.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 General

Non-architectural issues:

2.1.1 D1198

In part A (Preface), in the 'Additional reading' section, the reference that reads:

> [2] Arm Architecture Reference Manual Supplement, The Scalable Vector Extension 2 (SVE2), for ARMv9-A. (ARM DDI 614).

is corrected to read:

> [2] Arm Architecture Reference Manual Supplement, The Scalable Vector Extensions. (ARM DDI 584).

2.2 The Transactional Memory Extension

No issues.

2.3 The Embedded Trace Extension (ETE)

The known issues in the ETE architecture are listed below:

2.3.1 C1094

In section D10.1.1 (Enabling the trace unit), in Listing D10.1 'Example code sequence to enable the trace unit', the example assembly code that reads:

```
    MSR TRCPRGCTLR, x0    ;; Enable the ETE.
    ISB                   ;; Synchronize the write to TRCPRGCTLR
```

is changed to:

```
    MSR TRCPRGCTLR, x0    ;; Enable the ETE.
    ;; Wait for TRCSTATR.IDLE==0
poll_idle
    ISB                   ;; Synchronize the write to TRCPRGCTLR
    MRS x1, TRCSTATR
    TBNZ x1, #1, poll_idle
```

In section D10.1.2 (Disabling the trace unit), in Listing D10.2 'Example code sequence to disable the trace unit', the code that reads:

```
    MSR TRCPRGCTLR, x1    ;; Disable the trace unit
    ;; Wait for TRCSTATR.IDLE==1 and TRCSTATR.PMSTABLE==1
poll_idle
    ISB
    MRS x1, TRCSTATR
    TST x1, #3
    B.EQ poll_idle
```

is changed to:

```
    MSR TRCPRGCTLR, x1    ;; Disable the trace unit
    ;; Wait for TRCSTATR.IDLE==1 and TRCSTATR.PMSTABLE==1
poll_idle
    ISB
    MRS x1, TRCSTATR
    AND x1, x1, #3
    CMP x1, #3
    B.NE poll_idle
```

2.3.2 D1121

In section D6.5 (Trace unit power states), the following text is added:

> A system might support a Debug power domain which contains the interface between the trace unit and the external debugger, which is powered up when the external debugger needs to connect to the system. Such a Debug power domain is described in the Arm architecture.
>
> If the trace unit core power domain can be powered down independently of the Debug power domain, Arm recommends the system implements an external debug component with a Power-up request mechanism which can request the trace unit core power domain to be powered up.

> Arm strongly recommends the Power-up request mechanism is a CoreSight Class `0x9` ROM Table containing a parent entry for the trace unit.
>
> A parent entry of a component is one of:
>
> - An entry in the ROM table that locates the component.
> - An entry in a first ROM table that locates a second ROM table that includes a parent entry for the component. The second ROM table is a descendant of the first ROM table.
>
> This definition of a parent entry is recursive, and therefore the Power-up request mechanism might be high up in a hierarchy of ROM tables.
>
> The ROM table and any descendants might describe other debug components, including debug components for other PEs.
>
> The ROM table might have parent entries in other ROM tables, and those parent entries might also have a Power-up request mechanism in those ROM tables.
>
> If the Power-up request mechanism is implemented, in the Class `0x9` ROM Table containing the Power-up request mechanism for the trace unit:
>
> - The POWERIDVALID bit in the parent entry must be 1.
> - The POWERID field in the parent entry has an **IMPLEMENTATION DEFINED** value.
>
> It is **IMPLEMENTATION DEFINED** whether the trace unit core power domain is the PE Core power domain or some other power domain.
>
> For more information on the CoreSight Class `0x9` ROM Table, see Arm CoreSight Architecture Specification.

2.3.3 R1122

In section D6.8.2 (ViewInst include/exclude function filtering), subsection ‘Instruction blocks’, the following relaxation is added:

> If a block of instructions is not entirely covered by at least one individual ARC selected by TRCVIIECTLR.EXCLUDE, it is **CONSTRAINED UNPREDICTABLE** whether the block is excluded or not. This applies even if other ARCs selected by TRCVIIECTLR.EXCLUDE cover the rest of the block.

2.3.4 D1212

In section D3.1.1 (Direct P0 instructions), the WFI, WFE, WFIT, and WFET instructions are added to Table D3.1, ‘A64 direct P0 instructions’. Equivalent changes are made in section D3.2.1 (Direct P0 instructions), in Table D3.5 ‘A32 direct P0 instructions’, and section D3.3.1 (Direct P0 instructions), in Table D3.9 ‘T32 direct P0 instructions’, where the WFI and WFE instructions are added.

2.3.5 C1213

In the following locations:

- Section D3.1.4 (Meaning of Atom elements), Table D3.4 'Meaning of Atom elements in AArch64 A64'.
- Section D3.2.4 (Meaning of Atom elements), Table D3.8 'Meaning of Atom elements in AArch32 A32'.
- Section D3.3.4 (Meaning of Atom elements), Table D3.12 'Meaning of Atom elements in AArch32 T32'.
- Section D3.4.2 (Meaning of Atom elements).

The following instructions are updated to clarify that any P0 instruction which is traced with an E Atom element is considered 'taken', and any P0 instruction which is traced with an N Atom element is considered 'not taken':

- All variants of ERET.
- RFE.
- WFI, WFE, WFIT, and WFET.
- All variants of RET.
- ISB.

2.3.6 D1215

In section D7.11 (External inputs), rule R_{VBCBZ} is updated to remove PMU_HOVFS from the list of PMU events that are always exported to the trace unit, and the following rule is added:

> R_{KRSMY}
>
> The following PMU events are always exported to the trace unit, unless SelfHostedTraceEnabled() == TRUE and TRFCR_EL2.E2TRE is `0b0`:
>
> - PMU_HOVFS.

In the same section, the rules that read:

> R_{RFWZB}
>
> When SelfHostedTraceEnabled() == TRUE and tracing is prohibited, no PMU events are exported to the trace unit except the PMU events defined by rule VBCBZ which are always exported.
>
> R_{WSXTC}
>
> When SelfHostedTraceEnabled() == FALSE and the PE is in Secure state and counting in Secure state is prohibited, no PMU events are exported to the trace unit except the PMU events defined by rule VBCBZ which are always exported.

are updated to read:

> R_{RFWZB}
>
> When SelfHostedTraceEnabled() == TRUE and tracing is prohibited, only the PMU events defined by rules VBCBZ and KRSMY are exported to the trace unit.
>
> R_{WSXTC}
>
> When SelfHostedTraceEnabled() == FALSE and the PE is in Secure state and counting in Secure state is prohibited, only the PMU events defined by rules VBCBZ and KRSMY are exported to the trace unit.

2.3.7 D1258

In section D6.8.3.1 (Instruction blocks), rule R_{RNPWD} that reads:

> When an instruction in an instruction block is included to be traced by the ViewInst include/exclude function, the trace unit traces all of the instruction block.

is corrected to read:

> When an instruction in an instruction block is included to be traced by the ViewInst include/exclude function, the ViewInst include/exclude function includes all of the instruction block.

2.3.8 D1259

In section D10.4 (Filtering used the exclude function), the entry that reads:

Register	Value	Description
TRCVIIECTLR	`0x00000100`	Use ARC0 for the exclude logic.

is corrected to read:

Register	Value	Description
TRCVIIECTLR	`0x00010000`	Use ARC0 for the exclude logic.

2.3.9 D1267

In section D6.6.6 (Prohibited Regions), rule R_{HYZLQ} that reads:

> If an optional authentication interface is implemented, while Secure non-invasive debug is disabled according to the optional authentication interface, for execution in Secure state, the PE executes in a prohibited region.

is corrected to read:

> If an optional authentication interface is implemented, while Secure non-invasive debug is disabled according to the optional authentication interface and when SelfHostedTraceEnabled() returns FALSE, then for execution in Secure state, the PE executes in a prohibited region.

2.3.10 R1272

In section D6.2.3 (Behavior on flushing), rule R_{JLRQH} that reads:

> When any of the following occur, a trace unit flush is requested:
> - The trace unit transitions from an enabled to a disabled state.
> - The trace capture infrastructure requests a trace unit flush.
> - A TSB CSYNC instruction is executed while the Trace Buffer Extension is implemented and enabled.

is relaxed to read:

> When any of the following occur, a trace unit flush is requested:
> - The trace unit transitions from an enabled to a disabled state.
> - The trace capture infrastructure requests a trace unit flush.
> - A TSB CSYNC instruction is executed in a prohibited region and while the Trace Buffer Extension is implemented and enabled.

Additionally, in section D6.9.16 (Timestamp Element), the following bullet is removed from rule R_{YYWTR}:

> - When not in a prohibited region, a TSB CSYNC instruction is executed.

2.4 The Trace Buffer Extension (TRBE)

The known issues in the TRBE architecture are listed below:

2.4.1 D1091

In section E1.3.2 (Common architectural events), the event definition '`0x400D`, PMU_OVFS, PMU overflow, counters accessible to EL1 and EL0' that reads:

> The event is generated each time an event causes a PMEVCTNR<n>_EL1 counter overflow when PMINTENSET_EL1[n] is 1, for each implemented PMU counter n in the range 0 <= n < UInt(MDCR_EL2.HPMN), and the Cycle Counter (n = 31).

is corrected to read:

> The event is generated each time one of the following occurs:

> - An event is counted by an event counter <n> and all of the following are true:
>   - PMINTENSET_EL1[n] is `0b1`.
>   - One of the following is true:
>   - Counting the event causes unsigned overflow of PMEVCNTR<n>_EL0[31:0], and either FEAT_PMUv3p5 is not implemented or PMCR_EL0.LP is `0b0`.
>   - Counting the event causes unsigned overflow of PMEVCNTR<n>_EL0[63:0], FEAT_PMUv3p5 is implemented, and PMCR_EL0.LP is `0b1`.
>   - Either EL2 is implemented and <n> in the range [0 .. (MDCR_EL2.HPMN-1)], or EL2 is not implemented and <n> is in the range [0 .. (PMCR_EL0.N-1)].
> - A cycle is counted by PMCCNTR_EL0, PMINTENSET_EL1[31] is `0b1`, and one of the following is true:
>   - Counting the cycle causes unsigned overflow of PMCCNTR_EL0[31:0] and PMCR_EL0.LC is `0b0`.
>   - Counting the cycle causes unsigned overflow of PMCCNTR_EL0[63:0] and PMCR_EL0.LC is `0b1`.

Similarly, the event definition '`0x400F`, PMU_HOVFS, PMU overflow, counters reserved for use by EL2' that reads:

> The event is generated each time an event causes a PMEVCTNR<n>_EL1 counter overflow when PMINTENSET_EL1[n] is 1, for each implemented PMU counter n in the range UInt(MDCR_EL2.HPMN) <= n < UInt(PMCR_EL0.N). The event is not transmitted to a PE Trace Unit while TRCFR_EL2.E2TRE == `0b0`.

is corrected to read:

> The event is generated each time an event is counted by an event counter <n> and all of the following are true:
>
> - EL2 is implemented.
> - PMINTENSET_EL1[n] is `0b1`.
> - One of the following is true:
>   - Counting the event causes unsigned overflow of PMEVCNTR<n>_EL0[31:0], and either FEAT_PMUv3p5 is not implemented or MDCR_EL2.HLP is `0b0`.
>   - Counting the event causes unsigned overflow of PMEVCNTR<n>_EL0[63:0], FEAT_PMUv3p5 is implemented, and MDCR_EL2.HLP is `0b0`.
> - <n> in the range [MDCR_EL2.HPMN .. (PMCR_EL0.N-1)].
>
> The event is not transmitted to a PE Trace Unit when TRFCR_EL2.E2TRE is `0b0`.

2.4.2 C1139

In section E1.2.1.2 (Address translation enabled), the following rules are added:

> R_{SJFRQ}
>
> When the Trace Buffer Unit is Enabled, the Trace Buffer Unit might prefetch and cache address translations for the translation regime of the owning Exception level, including when the owning Exception level is out-of-context.
>
> I_{QXJZX}
>
> R_{SJFRQ} means that, when the Trace Buffer Unit is enabled and the owning Exception level is a lower Exception level, then the Trace Buffer Unit might make memory accesses to translation table entries from the translation regime of the owning Exception level, using the settings of the System registers associated with that translation regime.
>
> If the PE is not executing in the owning Security state, or the PE is executing at EL3 and SCR_EL3.NS does not indicate the owning Security state then the translation regime of the owning Exception level might not be the owning translation regime.
>
> These memory accesses might be observed by other observers, to the extent that those accesses are required to be observed as determined by the shareability and cacheability of those translation table entries.
>
> This is an exception to the rules in the section Use of out-of-context translation regimes of the Arm ARM.

2.4.3 C1140

In section E1.2.1.8 (Restrictions on programming the Trace Buffer Unit), the text in rule I_{VRFQC} that reads:

> Software context switching the Trace Buffer Unit will avoid this issue because the trace buffer management event sets TRBSR_EL1.S to 1, meaning the Trace Buffer Unit will not become Enabled following the context switch.

is corrected to read:

> Software context switching the Trace Buffer Unit will avoid this issue because the trace buffer management event sets TRBSR_EL1.S to 1, meaning the Trace Buffer Unit will not become Running following the context switch.

2.4.4 R1141

In section E1.2.3 (Synchronization and the Trace Buffer Unit), the following rules are removed:

- R_{CJLBR}.
- I_{FXBSC}.
- R_{TGBVJ}.

2.4.5 D1145

In section E1.3.2 (Common architectural events), in the definition of the '`0x400F`, PMU_HOVFS, PMU overflow, counters reserved for use by EL2' event, the text that reads:

> The event is not transmitted to a PE Trace Unit when TRFCR_EL2.E2TRE is `0b0`.

is changed to read:

> The event is not exported to a PE Trace Unit if SelfHostedTraceEnabled() is TRUE and TRFCR_EL2.E2TRE is `0b0`. This is in addition to the rules for the export of all events to a PE Trace Unit described in Arm Architecture Reference Manual section D3.2.1.

2.4.6 R1150

In section E1.2.1.8 (Restrictions on programming the Trace Buffer Unit), rule R_{MSPSD} that reads:

> Considering the trace buffer pointer addresses as 64-bit unsigned integers, a current write pointer value is *out-of-range* if it is not both:
>
> - Greater-than-or-equal-to the Base pointer.
> - Less-than the Limit pointer.
>
> Note: R_{MSPSD} means the current write pointer is *out-of-range* if the Base pointer is not less-than the Limit pointer.

is changed to read:

> A current write pointer value is *out-of-range* if any of the following are true:
>
> - The current write pointer is less-than the Base pointer, treating both pointers as unsigned integers.
> - The current write pointer is greater-than-or-equal-to the Limit pointer, treating both pointers as unsigned integers.
> - Bits [63:56] of the current write pointer are not equal to bits [63:56] of the Base pointer.
> - Bits [63:56] of the current write pointer are not equal to bits [63:56] of the Limit pointer.

> Note: R_{MSPSD} means the current write pointer is *out-of-range* if the Base pointer is not less-than the Limit pointer or bits [63:56] of the Base pointer are not equal to bits [63:56] of the Limit pointer.

2.4.7 R1191

In section E1.2.3 (Synchronization and the Trace Buffer Unit), rule R_{GTCKK} is relaxed to read:

> In the absence of any explicit synchronization, the trace unit generates the trace data for an instruction and the Trace Buffer Unit Accepts, Discards, or Rejects the trace data in finite time. However:
>
> - If the Trace Buffer Unit Accepts the trace data, then the write of the trace data to memory requires explicit synchronization to Complete.
> - The indirect writes to System registers made by a trace operation require explicit synchronization to guarantee they are observable.

Relatedly, in section E1.2.1.5 (Accesses to the trace buffer), in rule I_{VKQBR}, the following statement is removed:

> The write reaches its endpoint in finite time.

2.4.8 D1257

In section E1.3.2 (Common architectural events), in the description of '`0x400F`, PMU_HOVFS, PMU overflow, counters reserved for use by EL2', the text that reads:

> - Counting the event causes unsigned overflow of PMEVCNTR<n>_EL0[63:0], FEAT_PMUv3p5 is implemented, and MDCR_EL2.HLP is `0b0`.

is corrected to read:

> - Counting the event causes unsigned overflow of PMEVCNTR<n>_EL0[63:0], FEAT_PMUv3p5 is implemented, and MDCR_EL2.HLP is `0b1`.

2.5 The Branch Record Buffer Extension (BRBE)

The known issues in the BRBE architecture are listed below:

2.5.1 R1064

In section F1.2.1 (Common architectural events), the following text is added to the description for the event '`0x811F`, BRB_FILTRATE, Branch Record captured':

> It is **CONSTRAINED UNPREDICTABLE** whether a BRB_FILTRATE event is generated after a BRB INJ causes a Branch record to be injected.

2.5.2 C1225

In section F1.1.9.1 (Filtering on type), the following text in rule R_{FJYDC}:

> Branch records for the following instructions are optionally generated 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 instructions when the instruction is executed in a non-prohibited region and if any of the following are true:

2.5.3 D1226

In section F1.1.2 (Cycle counting), rule R_{MJDLC} is updated to read:

> The cycle count value in a Branch record is Branch Cycle Count Unknown when any of the following are true:
> - BRBCR_EL2.CC == `0b0`, if EL2 is implemented.
> - BRBCR_EL1.CC == `0b0`.
> - This is the first Branch record after the PE exited a Prohibited Region.
> - This is the first Branch record after cycle counting has been enabled.
> - This is the first Branch record after BRBFCR_EL1.PAUSED is cleared from 1 to 0.
> - This is the first Branch record after execution of a BRB IALL instruction.

Additionally, a new permission is introduced to permit the the cycle count to be Branch Cycle Count Unknown if there are no Branch records in the buffer.

2.5.4 D1236

In section F1.1.13.1 (Manual injection of Branch records), rule I_{BCZRK} that reads:

> Branch record injection is only performed when executing from a Prohibited Region.

is relaxed to read:

> If BRB INJ is executed outside of a Prohibited region, it is **CONSTRAINED UNPREDICTABLE** whether a Branch record is created.

2.5.5 C1238

In section F1.1.9.1 (Filtering on type), the following Note is added to rule R_{BBNSZ}:

> Note: BC.cond and B.cond instructions with the AL or NV condition code are considered conditional.

2.5.6 D1242

In section F1.1.5 (Branch records for exceptions), the following rules are added:

> R_{BZCRW} A Branch record for an exception which contains a valid source address, has the source address set to the preferred exception return address for the exception.
>
> R_{FYLTC} A Branch record for an exception which contains a valid target address, has the target address set to the address of the exception vector.

2.5.7 C1288

In section F1.1.10 (Branch record buffer operation), in Table D1.11 'Captured timestamp', references to CNTPCT_EL0 are replaced with PhysicalCountInt(), and references to CNTPOFF_EL2 are replaced with physical offset, as follows:

BRBCR_EL2.TS	BRBCR_EL1.TS	Captured timestamp
`0b00`	`0b01` (virtual)	PhysicalCountInt() - CNTVOFF_EL2
`0b00`	`0b10` (offset physical)	PhysicalCountInt() - physical offset
`0b00`	`0b11` (physical)	PhysicalCountInt()
`0b01` (virtual)	0bxx	PhysicalCountInt() - CNTVOFF_EL2
`0b10` (offset physical)	0bxx	PhysicalCountInt() - physical offset
`0b11` (physical)	0bxx	PhysicalCountInt()