



# Release notes for the A64 Instruction Set Architecture for Arm A-profile Architecture

2025-06

**Non-Confidential**

Copyright © 2023–2025 Arm Limited (or its affiliates).  
All rights reserved.

**Issue 01**

109389\_2025-06\_01\_en



# Release notes for the A64 Instruction Set Architecture for Arm A-profile Architecture

Copyright © 2023–2025 Arm Limited (or its affiliates). All rights reserved.

## Release information

### Document history

Issue	Date	Confidentiality	Change
2025_06-01	30 June 2025	Non-Confidential	2025-06 release
2025_03-01	26 March 2025	Non-Confidential	2025-03 release
2024_12-01	17 December 2024	Non-Confidential	2024-12 release
2024_09-01	30 September 2024	Non-Confidential	2024-09 release
2024_06-01	5 July 2024	Non-Confidential	2024-06 release
2024_04-01	27 March 2024	Non-Confidential	2024-03 release
2023_12-01	19 December 2023	Non-Confidential	2023-12 release
2023_09-01	29 September 2023	Non-Confidential	2023-09 release

## Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not

represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. Please follow Arm's trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

## 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.

Unrestricted Access is an Arm internal classification.

## 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](mailto:terms@arm.com).

# Contents

1. A64 ISA Data release for A-profile Architecture (2025-06).....6

# 1. A64 ISA Data release for A-profile Architecture (2025-06)

30 June 2025

## Product Status

The information relating to the 2024 Extensions and the rest of the A-profile Architecture is at Beta quality. Beta quality means that all major features of the specification are described, but some details might be missing.

## Change History

The following changes are made to instruction descriptions:

- The FEXPA execution in Streaming SVE mode is corrected to require FEAT\_SSVE\_FEXPA instead of FEAT\_SME2p2.
- SRSR, SSSL, SRSRA, URSR, URSRA, and URSR are added to the list of instructions that honor PSTATE.DIT.
- In the CPYP, CPYM, CPYE\* instructions, the CPYE\* option B behavior when the copy is in the backward direction is corrected.
- The CPYF\* instructions are changed to permit their use when the source and destination addresses are equal.
- The PRFUM instruction is updated to include support for the SLC target when FEAT\_PRFM\_SLC is implemented.
- The CLRBHB instruction is clarified to state that it can be used to clear the branch history and is available when FEAT\_CLRBHB is implemented.
- The instruction SVE CNTP (predicate-as-counter) is clarified to state that there is a limit to the counted number of elements that corresponds to the total number of elements in either two or four vectors.
- The following instructions are corrected to describe when they lose their acquire semantics according to if the destination registers is one of WZR or XZR:
  - CAS, CASA, CASAL, CASL.
  - CASB, CASAB, CASALB, CASLB.
  - CASH, CASAH, CASALH, CASLH.
  - LDAPR.
  - LDAPRB.
  - LDAPRH.
  - LDAPUR.
  - LDAPURB.
  - LDAPURH.
  - LDAPURSB.

- LDAPURSH.
  - LDAPURSW.
  - LDAR.
  - LDARB.
  - LDARH.
  - LDAXP.
  - LDIAPP.
  - LDAXR.
  - LDAXRB.
  - LDAXRH.
  - LDLAR.
  - LDLARB.
  - LDLARH.
  - RCWCAS, RCWCASA, RCWCASL, RCWCASAL.
  - RCWCLR, RCWCLRA, RCWCLRL, RCWCLRAL.
  - RCWSCAS, RCWSCASA, RCWSCASL, RCWSCASAL.
  - RCWSCLR, RCWSCLRA, RCWSCLRL, RCWSCLRAL.
  - RCWSET, RCWSETA, RCWSETL, RCWSETAL.
  - RCWSSET, RCWSSETA, RCWSSETL, RCWSSETAL.
  - RCWSSWP, RCWSSWPA, RCWSSWPL, RCWSSWPAL.
  - RCWSWP, RCWSWPA, RCWSWPL, RCWSWPAL.
- The operational information for an alias or pseudo-instruction is specified only in its parent.

The following changes are made to the pseudocode:

- The function `AArch64.S1ComputePermissions()` is corrected to avoid removing Stage 1 Base Execute permission when `s1perms.overlay` is `TRUE` and `s1perms.ox` is `'0'`.
- The pseudocode function `GPCRegistersConsistent()` is updated to check if the bypass window base address is greater than or equal to the stride value.
- The function `PMUCaptureEvent()` is corrected to avoid setting the sampled registers to **UNKNOWN** value in Debug state.
- The function `PCSRsuspended()` is corrected to return `TRUE` if PC Sample-based Profiling is suspended, and `FALSE` otherwise.
- The functions `AArch64.S1CheckPermissions()` and `AArch64.S2CheckPermissions()` are updated to return a fault due to lack of write permission for `DC I`, `DC CI`, and `IC I*` accesses when HW update of dirty state is disabled using the indirect permission scheme.
- The ordering of RNG arguments used in the RME GPT walk memory attributes is corrected.

- The function `ReportAsGPCException()` is corrected to state that, in Debug state if `EDSCR.SDD` is 1, a GPC Fault is not reported as a GPC exception. Similar updates are made to `AArch64.DataAbort()` and `AArch64.InstructionAbort()`.
- The operational pseudocode for `SETG*` is updated to report the lowest address where Allocation tags or data have not been written.
- The functions `RecipEstimate()` and `RecipSqrtEstimate()` are refactored to ensure the increased precision implementation matches the style and structure of the lower precision code.
- The function `AArch64.MemSingleRead()` is corrected to record the specific Virtual Address that caused the external abort, rather than the lowest Virtual Address accessed.
- The function `IsDataAccess()`, which is used solely to determine whether an access is subject to watchpoint checks, is corrected to account for `AccessType_SPE` and `AccessType_TRBE`.
- The redundant calls to the function `AArch64.AllocationTagAccessIsEnabled()` in the MTE pseudocode are removed. The `AArch64.AddressWithAllocationTag()` function is simplified. The new functions `AArch64.ChooseTagOrZero()` and `AArch64.ChooseNonExcludedTagOrZero()` are introduced. The instruction pseudocode for `ADDG`, `SUBG` is updated to use `AArch64.ChooseNonExcludedTagOrZero()`, and the `IRG` instruction is updated to use `AArch64.ChooseTagOrZero()`.
- The functions `AArch64.MemAtomicRCW()`, `Arch64.MemLoad64B()`, `AArch64.MemStore64BWithRet()`, and `AArch64.MemStore64B()` are corrected to check if a sample is in flight and samples the operation for `SPE`.
- The functions `AArch64.FaultSyndrome()` and `AArch64.PhysicalErrorSyndrome()` are updated to set `ESR_ELx.ISS.PFV`. The function `AArch64.PhysicalErrorSyndrome()` is refactored to set all syndrome bits for `AArch64.TakePhysicalSErrorException()` and `AArch64.ESBOperation()`. The functions `AArch64.PhysicalSErrorSyndrome()`, `AArch64.AbortSyndrome()`, `HandleExternalTTWAbort()`, and `HandleExternalAbort()` are updated to capture the faulting PA for reporting in `PFAR_ELx`.
- The function `AArch64.MemSingleRead()` is updated to check for Breakpoints for an `AccessType_IFETCH` after memory is accessed. Checks for Breakpoints are removed from the function `AArch64.TranslateAddress()`.
- The function `SPEConstructRecord()` is corrected to saturate 16-bit `SPE` counters at `0xFFFF`.
- The following pseudocode functions are clarified to consistently refer to the SET registers for indirect reads and writes to the PMU SET/CLR register pairs:
  - `IncrementInstructionCounter()`.
  - `CheckForPMUException()`.
  - `CheckPMUOverflowCondition()`.
  - `IncrementEventCounter()`.

Many simple clarifications and corrections are also present, but are too small to be listed here. Some minor formatting changes are suppressed and not highlighted in the diff output.

## Known Issues

All issues identified in the below list will be fixed in a future release.

- The assembler syntax for `MRS` and `MSR` instructions with unnamed registers will be clarified.



- In the following SME instructions, the feature conditions stated in the pseudocode are correct, but the feature conditions stated above the encoding diagrams are incomplete:
  - SMLALL (multiple vectors), SMLALL (multiple and single vector).
  - SMLSLL (multiple vectors), SMLSLL (multiple and single vector).
  - UMLALL (multiple vectors), UMLALL (multiple and single vector).
  - UMLSLL (multiple vectors), UMLSLL (multiple and single vector).
  - SDOT (4-way, multiple and single vector), SDOT (4-way, multiple vectors).
  - UDOT (4-way, multiple and single vector), UDOT (4-way, multiple vectors).
  - FMLA (multiple and single vector), FMLA (multiple vectors).
  - FMLS (multiple and single vector), FMLS (multiple vectors).
  - ADD (array results, multiple and single vector), ADD (array results, multiple vectors), ADD (array accumulators).
  - SUB (array results, multiple and single vector), SUB (array results, multiple vectors), SUB (array accumulators).
  - FADD.
  - FSUB.
- **CONSTRAINED UNPREDICTABLE** update of S2AP[1] for fault on atomic access with DBM[1] and S2AP[0:1] is 0b00 when read permission is not present.
- The accessibility for DC or IC does not show the access permissions checks are **IMPLEMENTATION DEFINED** when the operation is treated as a **NOP**.
- The Mem[] functions do not treat an SVE predicated load or store of a 128-bit element that is 64-bit aligned as a pair of 64-bit single-copy atomic accesses.
- The ASL model currently reports the input virtual address for a DC ZVA watchpoint match, even when the match occurs in a different naturally aligned block within the zeroing range. This can be incorrect, as the reported address should reflect the block where the watchpoint actually matched, not just the starting address, so will be corrected.
- The operational pseudocode for SETG\* will be corrected to check the tag AddressDescriptor for a fault on the tag setting, instead of data AddressDescriptor.

## Limitations of Arm pseudocode

The pseudocode statements IMPLEMENTATION\_DEFINED, SEE, **UNDEFINED**, and **UNPREDICTABLE** indicate behavior that differs from that indicated by the pseudocode being executed. If one of them is encountered:

- Earlier behavior indicated by the pseudocode is only specified as occurring to the extent required to determine that the statement is executed.
- No subsequent behavior indicated by the pseudocode occurs. For more information, see Special statements in the Arm® Architecture Reference Manual for A-profile architecture (ARM DDI 0487). The pseudocode descriptions have several limitations. These are mainly since, for clarity and brevity, the pseudocode is a sequential and mostly deterministic language. These limitations include:

- Pseudocode does not describe the ordering requirements when an instruction generates multiple memory accesses. For a description of the ordering requirements on memory accesses, see External ordering constraints in the Arm® Architecture Reference Manual for A-profile architecture (ARM DDI 0487).
- Pseudocode does not describe the exact ordering requirements when a single floating-point instruction generates more than one floating-point exception and one or more of those floating-point exceptions is trapped. Combinations of floating-point exceptions in the Arm® Architecture Reference Manual for A-profile architecture (ARM DDI 0487) describes the exact rules. Note: There is no limitation in the case where all the floating-point exceptions are untrapped, because the pseudocode specifies the same behavior as the referenced section.
- When the architectural behavior of an instruction could be performed as a concurrent set of operations that are not architecturally ordered, the pseudocode represents it as a sequential set of operations.
- An exception can be taken during execution of the pseudocode for an instruction, either explicitly as a result of the execution of a pseudocode function such as Abort(), or implicitly, for example if an interrupt is taken during execution of an LDM instruction, load-store pair instructions, SVE vector instructions, Memory copy and memory set instructions etc.. If this happens, the pseudocode does not describe the extent to which the normal behavior of the instruction occurs. To determine that, see the descriptions of the exceptions in Handling exceptions that are taken to an Exception level using AArch32 and Definition of a precise exception and imprecise exception in the Arm® Architecture Reference Manual for A-profile architecture (ARM DDI 0487).
- Pseudocode does not describe the exact rules when an AArch32 instruction that that fails its condition code check generates any of the following:
  - **UNDEFINED** instruction.
  - Hyp trap.
  - Monitor trap.
  - Trap to AArch64 exception.
  - Conditional execution of undefined instructions.
  - EL2 configurable controls.
  - EL3 configurable controls.
  - Configurable instruction controls.
- Where a significant aspect of the behavior is IMPLEMENTATION DEFINED, pseudocode may present only the declarations of the functions - the details of these functions is provided by each implementation of the architecture.
- Pseudocode does not present the possible observability due to speculative execution.
- Pseudocode presents various details of the architecture - but does not show how the details can be combined to form a complete implementation. There are various aspects of an implementation that are also not presented. Notably, pseudocode does not show the details of fetching, decoding, and linking to instruction execution.
- Pseudocode presents all the architectural state as global state. The possible implications of multiple PEs or other components is not shown.
- The following details are either not shown or have noted limitations in the pseudocode:

- Self-hosted trace and external trace.
- Modeling the System register state or side effects. The accessibility details of a direct or external access such as traps etc are shown.
- Generation of all architectural and micro-architectural Performance Monitoring Events. Note: Some architectural event generation is shown.
- Construction of Statistical Profiling Extension records.
- Statistical Profiling functionality for 2023 features.
- Behavior of instructions in Debug state when the behavior is **UNPREDICTABLE** is presented as if the instruction is executed identically to how it is when not in Debug state.
- Activity Monitor Events and Counters.
- Generic Interrupt Controller functionality.
- External memory system.
- External agents such as a debugger.
- PE behaviors that would lead to unrecoverable or uncontrollable errors.
- Where the behavior is **IMPLEMENTATION DEFINED** or **CONSTRAINED UNPREDICTABLE**, not all possibilities may be shown. Sometimes the pseudocode may present a simplified, but architecturally compatible view. In some situations, the possible behaviors may be outlined in a comment.
- The following architectural features are at Alpha quality and are above the limitations described below due to recency and completion of validation:
  - Halting Debug
  - Statistical Profiling Extension.
  - Performance Monitoring Events.

## Potential Upcoming Changes

The details of the architecture are presented in pseudocode in Architecture Specification Language (ASL). Arm is defining a new version of the Architecture Specification Language, ASL1, to improve and expand the capabilities of the language. Please see <https://developer.arm.com/Architectures/Architecture%20Specification%20Language> for details on this language. Arm will be publishing an equivalent release in ASL1 format later in 2025.

## Intention and quality statements for all ArmARM architecture releases

The intention and scope of the Architecture releases is to describe changes from the existing architecture to the next release. The quality of the architecture releases refers to the accuracy and completeness of the changes described in the specifications.

The intention and scope of the AARCHMRS and Data releases is to describe the content and behavior of the registers, system registers, instructions, pseudocode and features of the architecture in full, for human readers in a way that enables correct information for the current or any previous release can be deduced. The quality of the XML releases refers to the accuracy and completeness of the content to a human reader.

The intention and scope of the JSON releases is to describe aspects of the AARCHMRS and Data releases in a structured, machine readable format. The content of the AARCHMRS and Data architectural content will be approximately equivalent to the corresponding XML release. However there are some aspects of the architecture which cannot yet be represented in a machine readable format. The content of the AARCHMRS architectural content will be approximately equivalent to the corresponding Data release.

The intention and scope of the Schema for the JSON releases is to describe the syntax and format of the json files used in the json releases. The schema is still under development and is subject to change.