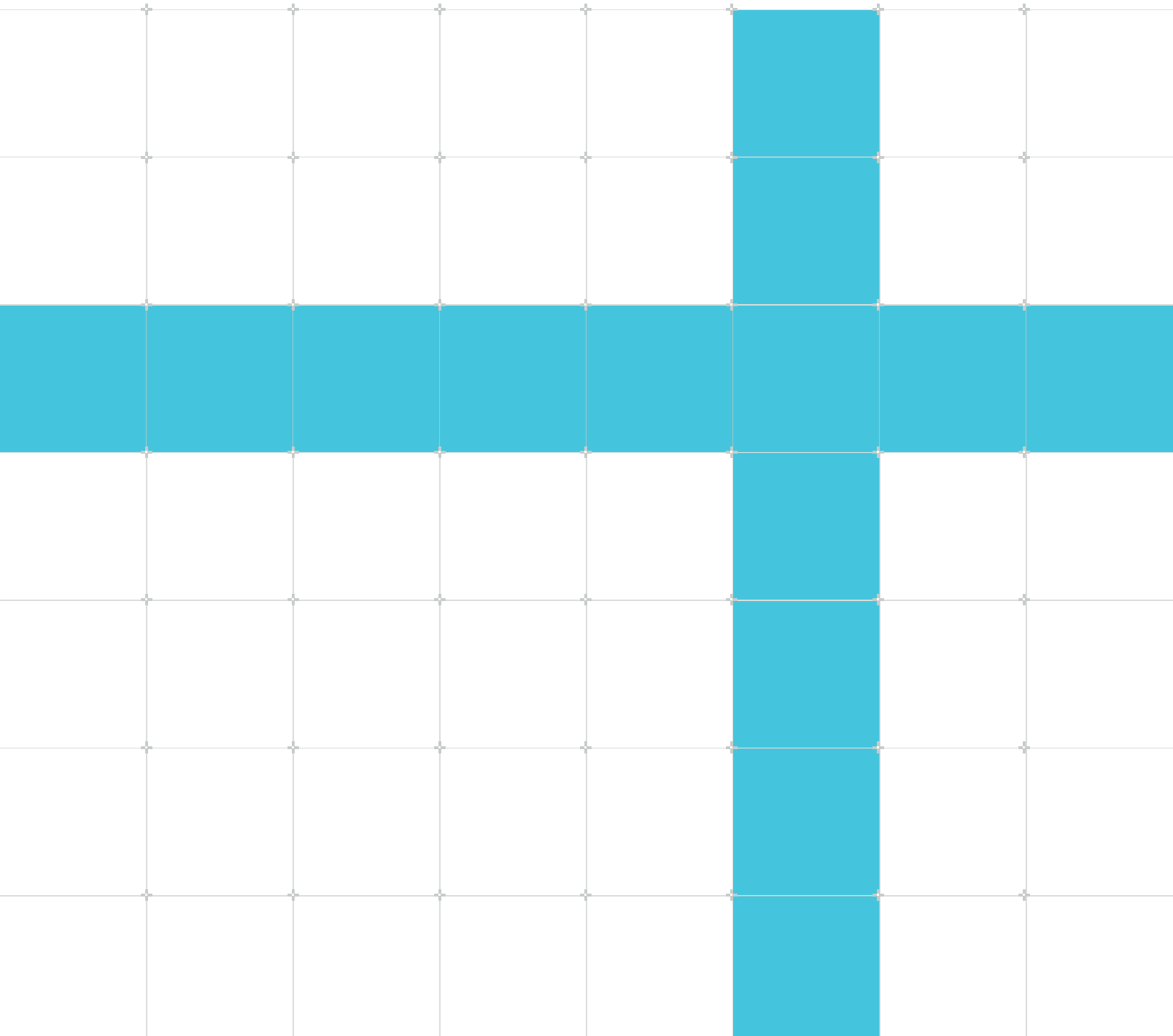
Release Notes for Arm Compiler for Embedded 6.24

Version 6.24

Non-Confidential
Copyright © 2025 Arm Limited (or its affiliates).
All rights reserved.

Issue 00
110289_062400_00_en



Release Notes for Arm Compiler for Embedded 6.24

This document is Non-Confidential.

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

This document is protected by copyright and other intellectual property rights.
Arm only permits use of this document if you have reviewed and accepted Arm's Proprietary Notice found at the end of this document.

This document (110289_062400_00_en) was issued on 2025-03-31. There might be a later issue at https://developer.arm.com/documentation/110289

The product version is 6.24.

See also: Proprietary notice | Product and document information | Useful resources

Start reading

If you prefer, you can skip to the start of the content.

Intended audience

This document is intended for use by a software developer who is using Arm Compiler for Embedded. The document includes an overview of the Arm Compiler for Embedded 6.24 release, changes and enhancements, and defects fixed.

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.

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.

Contents

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

1. Release overview

This chapter provides an overview of the Arm Compiler for Embedded product and the Arm Compiler for Embedded 6.24 release. Arm Compiler for Embedded 6.24 is the final planned feature release of Arm Compiler for Embedded.

1.1 Product description

Arm Compiler for Embedded 6.24 is the final planned feature release of Arm Compiler for Embedded. Future updates will be limited to defect fixes, with no additional features.

Arm Compiler for Embedded is a proven and mature toolchain for the Arm targets it supports, and can be used for:

- Projects for which migration to another toolchain may be difficult
- Projects that require small code size
- Projects with a big-endian target

Licenses which enable Arm Compiler for Embedded continue to be available, and the toolchain continues to be available for download via Arm Product Download Hub (PDH).

Arm Compiler for Embedded is compatible with the following Arm Integrated Development Environments (IDEs):

- Arm Development Studio (Arm DS)
- Keil MDK v6
- µVision within Keil MDK v6

For projects with long-term maintenance or functional safety requirements, consider using the latest release of Arm Compiler for Embedded FuSa 6.22LTS instead of Arm Compiler for Embedded 6.24.

Introducing the new Arm Toolchain for Embedded product family

Arm Toolchain for Embedded is Arm's 7th generation C/C++ compiler for embedded systems. It completes the transition to a fully Open-Source toolchain that Arm began in 2014 with the first release of the 6th generation C/C++ compiler for embedded systems, Arm Compiler for Embedded.

Arm Toolchain for Embedded replaces key components of Arm Compiler for Embedded with equivalent Open-Source components:

Component	Arm Compiler for Embedded	Arm Toolchain for Embedded
Compiler	`armclang`	`clang`
Linker	`armlink`	`lld`

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Component	Arm Compiler for Embedded	Arm Toolchain for Embedded
Assembler	`armclang` integrated assembler and the legacy `armasm` assembler	`clang` integrated assembler
ELF utility	`fromelf`	`llvm-objdump` and `llvm-objcopy`
Librarian	`armar`	`llvm-ar`
C Library	Arm proprietary C libraries	Open-Source C libraries

This evolution of the underlying toolchain technology may require you to make significant changes to your project when migrating to Arm Toolchain for Embedded. For more information, see the Arm Toolchain for Embedded documentation.

Consider migrating to a suitable edition of the Arm Toolchain for Embedded product family if your project requires any of the following:

- Support for an Arm architecture or processor launched after 2024
- The latest architecture features for AArch64 state
- Optimizations for the M-profile Vector Extension (MVE)
- Compatibility with the Arm GNU Toolchain
- Features not available in Arm Compiler for Embedded, but enabled through Open-Source LLVM and Clang technology

Contact your sales representative or submit an inquiry online to find out more about licensing Arm software development tools including Arm Compiler for Embedded and Arm Toolchain for Embedded.

1.2 Release highlights

Arm Compiler for Embedded 6.24 is the latest release as of March 2025, and is the final planned feature release.

The key highlights of this release include:

- Full support for the Armv9.6-A architecture
- Full support for 2024 extensions for A-profile architectures

Subject to your license terms, Arm Compiler for Embedded 6.24 can be used to build for the following Arm Architectures and Processors:

Architecture	Processor Family	Standard Processors	Automotive Enhanced Processors
Armv9-A up to Armv9.6-A	Neoverse	V3, V2 N3, N2	V3AE

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Architecture	Processor Family	Standard Processors	Automotive Enhanced Processors
	Cortex	X925 X4, X3, X2 A725, A720, A715, A710 A520, A510	A720AE A520AE
Armv8-A up to Armv8.9-A	Neoverse	V1 N1 E1	-
	Cortex	X1C, X1 A78C, A78, A77, A76, A75, A73, A72 A65 A57, A55, A53 A35, A34, A32	A78AE, A76AE A65AE
Armv7-A	Cortex	A17, A15, A12 A9, A8, A7, A5	-
Armv8-R AArch64	Cortex	R82	R82AE
Armv8-R	Cortex	R52+, R52	-
Armv7-R	Cortex	R8, R7, R5, R4F, R4	-
Armv8-M up to Armv8.1-M	Cortex	M85 M55, M52 M35P, M33 M23	-
	STAR	STAR-MC1	-
Armv7-M	Cortex	M7, M4, M3	-
	SecurCore	SC300	-
Armv6-M	Cortex	M1, M0, M0+	-
	SecurCore	SC000	-

For more information, see the following:

- The Arm Development Studio product page.
- The Arm Keil MDK v6 product page.
- The *Support level definitions* section of the Arm Compiler for Embedded User Guide document for this release.

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

1.3 Included components

This section lists the toolchain components and different types of documentation included with Arm Compiler for Embedded 6.24.

Category	Component	Description
Toolchain components	`armclang`	Compiler and integrated assembler based on LLVM and Clang technology
	`armar`	Archiver which enables sets of ELF object files to be collected together
	`armlink`	Linker that combines objects and libraries to produce an executable
	`fromelf`	ELF image conversion utility and disassembler
	Arm proprietary C libraries	Runtime support libraries for embedded systems
	C++ libraries	Libraries based on the LLVM libc++ project
	Legacy `armasm` assembler	Deprecated legacy assembler for armasm-syntax assembly code for older Arm architectures only. Use the `armclang` integrated assembler for all new assembly files.
User documentation	User Guide	Provides instructions and examples to help you use the toolchain
	Reference Guide	Provides information to help you configure the toolchain
	Arm C and C++ Libraries and Floating-Point Support User Guide	Provides information about the libraries and floating-point support
	Errors and Warnings Reference Guide	Provides a list of the errors and warnings that can be reported by `armar`, `armasm`, `armlink`, and `fromelf`
	Migration and Compatibility Guide	Provides information to help you migrate from Arm Compiler 5 to Arm Compiler for Embedded
	Release Notes	These release notes

These components can be obtained from the following sources:

Component type	Source
Toolchain components	Available via Arm Product Download Hub (PDH)
User documentation	Available via Arm Developer

1.4 Product quality

This product is a Final release quality product which is suitable for use in a production environment.

Certain features within this product are not Final release quality features or are unsupported. The status of features is explicitly stated within the documentation where applicable. For more information about such features, see the *Support level definitions* section of the Arm Compiler for Embedded User Guide.

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

2. Download Arm Compiler for Embedded 6.24

This chapter provides information about how to download and install Arm Compiler for Embedded 6.24.

Arm Compiler for Embedded 6.24 might be available to download standalone, as part of an Arm Integrated Development Environment (IDE), or as part of a Success Kit depending on your license and entitlements.

To download this release standalone, use the `ACOMPE` code on Arm Product Download Hub (PDH). The toolchain download packages within this product code are intended to be used in the following environments as of March 2025*:

Host architecture	Host operating system	Toolchain download package	Host environment
x86_64	• Red Hat Enterprise Linux 9 • Red Hat Enterprise Linux 8 • Red Hat Enterprise Linux 7 • Ubuntu 24.04 LTS • Ubuntu 22.04 LTS • Ubuntu 20.04 LTS	x86_64 Linux	• Standalone installation • Integrated into Arm Development Studio • Integrated into Keil MDK v6
	• Windows Server 2022 • Windows 11 • Windows 10	x86_64 Windows	• Standalone installation • Integrated into Arm Development Studio • Integrated into Keil MDK v6
		for Keil® MDK	• Integrated into µVision within Keil MDK v6
AArch64	• Ubuntu 24.04 LTS • Ubuntu 22.04 LTS • Ubuntu 20.04 LTS	AArch64 Linux	• Standalone installation • Integrated into Keil MDK v6

*Arm reserves the right to add additional environments for this release. The environments may differ between releases.

The following restrictions apply:

- The minimum required version of glibc for Linux host platforms is as follows:
  - 2.15 for x86_64 host platforms.
  - 2.17 for AArch64 host platforms.
- The toolchain must not be installed directly into an Arm Development Studio installation directory.
- µVision within Keil MDK version 6 requires that the toolchain is installed into the `ARM` sub-directory of the µVision installation directory. For example, `C:\Keil_v5\ARM\ARMCompiler6.24`.
- Use with a legacy Keil license is only permitted on x86_64 Windows host platforms.

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

For more information, see the following:

- The *System requirements and installation* section of the Arm Compiler for Embedded User Guide for toolchain installation instructions.
- The *Register a compiler toolchain* section of the Arm Development Studio Getting Started Guide for instructions to integrate the toolchain into Arm Development Studio.
- The Arm Keil Studio Visual Studio Code Extensions User Guide for instructions to integrate the toolchain into Arm Keil MDK v6.
- The User-based Licensing User Guide for instructions to configure the toolchain to use a User-based License (UBL).
- The *Manage Arm Compiler Versions* section of the uVision User's Guide.

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

3. Differences from previous release

This chapter describes differences from the previous release, Arm Compiler for Embedded 6.23.

For more information about the scope of this section, see the article Does Arm document all known issues that affect each Arm Compiler release?.

The information below may include technical inaccuracies or typographical errors. Each itemized change is accompanied by a unique SDCOMP-<n> identifier. If you need to contact Arm about a specific issue within these release notes, please quote the appropriate identifier.

3.1 General changes

This section contains a list of general changes made in this release of Arm Compiler for Embedded.

SDCOMP-68073

Support has been added for the `-fstrict-flex-arrays=<value>` option to control what the compiler considers to be a flexible array.

For more information, see the *-fstrict-flex-arrays=<value>* section of the *Reference Guide*.

SDCOMP-67982

Support for the following A-profile architectures has been changed:

Architecture	State	-march=<name> option	Previous support level	New support level
Armv9.6-A	AArch64	`armv9.6-a`	Beta	Supported
Armv9.6-A	AArch32	`armv9.6-a`	Beta	Supported

For more information, see the *-march* section of the *Reference Guide*.

SDCOMP-67905

Support for the following A-profile architecture features for AArch64 state has been changed:

Feature identifier	Feature description	-march / -mcpu +<feature> option(s)	Previous support level	New support level
FEAT_CMPBR	Compare and Branch instruction	`cmpbr`	Beta	Full
FEAT_F8F16MM	FP8 to Half-Precision Matrix Multiplication	`f8f16mm`	Beta	Full
FEAT_F8F32MM	FP8 to Single-Precision Matrix Multiplication	`f8f32mm`	Beta	Full
FEAT_FPRCVT	Floating-Point to/from Integer in Scalar FP register	`fprcvt`	Beta	Full
FEAT_LSFE	Large System Float Extension	`lsfe`	Beta	Full
FEAT_LSUI	Unprivileged Load Store	`lsui`	Beta	Full
FEAT_OCCMO	Outer Cacheable Cache Maintenance Operation	`occmo`	Beta	Full
FEAT_PCDPHINT	Producer-Consumer Data Placement Hints	`pcdphint`	Beta	Full

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Feature identifier	Feature description	-march / -mcpu +<feature> option(s)	Previous support level	New support level
FEAT_PoPS	Point of Physical Storage	`pops`	Beta	Full
FEAT_RME_GPC3	Realm Management Extension (RME) Granule Protection Check 3 Extension	`rme-gpc3`	Beta	Full
FEAT_SME2p2	Scalable Matrix Extension version 2.2	`sme2p2`	Beta	Full
FEAT_SSVE_AES	Streaming Scalable Vector Extensions (SVE) Mode Advanced Encryption Standard and 128-bit polynomial multiply long instructions	`ssve-aes`	Beta	Full
FEAT_SVE2p2	Scalable Vector Extensions version 2.2	`sve2p2`	Beta	Full
FEAT_SVE_AES2	SVE multi-vector Advanced Encryption Standard and 128-bit polynomial multiply long instructions	`sve-aes2`	Beta	Full
FEAT_SVE_BFSCALE	BFloat16 Floating-Point Adjust Exponent	`sve-bfscale`	Beta	Full
FEAT_SVE_F16F32MM	SVE Half-Precision to Single-Precision Matrix Multiplication	`sve-f16f32mm`	Beta	Full

When compiling with `-march=armv9.6-a`, the compiler enables the following subset of these features by default:

- FEAT_CMPBR
- FEAT_FPRCVT
- FEAT_LSUI
- FEAT_OCCMO
- FEAT_SVE2p2

For more information, refer to:

- The *-march* section of the *Reference Guide*.
- The *-mcpu* section of the *Reference Guide*.
- The relevant *Arm Architecture Reference Manual* or *Reference Manual Supplement* for each feature.

SDCOMP-67851

Support has been added for the `--print-supported-extensions` option to print the architecture features available in AArch64 state or AArch32 state.

For more information, see the *--print-supported-extensions* section of the *Reference Guide*.

SDCOMP-67587

The maximum acceptable length of an input filename or tool option within a via file for the linker has been increased. The linker reports the following fatal error if the maximum acceptable length is exceeded:

- `L3907U: Via file '<filename>' command too long for buffer`

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

SDCOMP-67299

Support has been added for AArch32 state for the Cortex-A510 processor. To target Cortex-A510 in AArch32 state, select from the following `armclang` options:

Cryptographic Extension	Options
Included	`--target=arm-arm-none-eabi -mcpu=cortex-a510 -mfpu=crypto-neon-fp-armv8`
Not included	`--target=arm-arm-none-eabi -mcpu=cortex-a510`

SDCOMP-65404

Support for `--cpu=<name>` options for the Cortex-M52 processor has been added to the linker and the ELF processing utility.

For more information, see the *Combinations of architecture features supported for the Cortex-M52 processor* section of the *Reference Guide*.

3.2 Defect fixes

This section contains information about defect fixes made in this release of Arm Compiler for Embedded. It contains sub-sections that each focus on the defect fixes in a specific component of the toolchain.

3.2.1 Compiler and integrated assembler, armclang

This section contains a list of defect fixes made in the compiler and integrated assembler, `armclang`.

SDCOMP-68081

The inline assembler and integrated assembler could incorrectly report one of the following errors for a valid `VSCCLRM` instruction:

- `invalid register in register list`
- `register expected`

This has been fixed.

SDCOMP-67974

When compiling with target options that enable the M-profile Vector Extension (MVE) and at any optimization level except `-O0`, the compiler could generate incorrect code for a memory access operation. This has been fixed.

SDCOMP-67945

When compiling for a Cortex-M52 target, a Cortex-M85 target, or with an `-march=<name>` or `-mcpu=<name>` option that includes the `+pacbti` feature modifier, and when configured without User-based Licensing, the compiler could incorrectly report one of the following errors:

- `Cortex-M52 is not available with the current toolkit edition and license`

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

- `Cortex-M85 is not available with the current toolkit edition and license`
- `use of -target-feature +pacbti is disallowed in this variant of Arm Compiler for Embedded`

This has been fixed.

SDCOMP-67921

When compiling for AArch64 state, and in a C++ source language mode, the compiler could generate code that incorrectly fails to ignore a function parameter of zero size. This has been fixed.

SDCOMP-67823

When compiling for AArch64 state, the compiler could generate incorrect code for a function with pointer authentication branch protection using the Program Counter as a second diversifier for return address signing. This has been fixed.

SDCOMP-67799

When compiling at any optimization level except `-O0`, the compiler could generate incorrect code for an M-profile Vector Extension (MVE) intrinsic of the form `*vsbcq*()` defined in the `<arm_mve.h>` system header. This has been fixed.

For more information about MVE intrinsics, see https://developer.arm.com/architectures/instruction-sets/intrinsics.

SDCOMP-67711

When compiling for AArch32 state, the compiler could incorrectly assume that the `+nofp.dp` feature modifier is equivalent to the `+nofp` feature modifier. This has been fixed.

SDCOMP-67678

When compiling at any optimization level except `-O0`, the compiler could generate incorrect code for a function that contains a call to the following forms of M-profile Vector Extension (MVE) intrinsics defined in the `<arm_mve.h>` system header:

- `*vadcq_*()`
- `*vsbcq_*()`

This has been fixed.

For more information about MVE intrinsics, see https://developer.arm.com/architectures/instruction-sets/intrinsics.

SDCOMP-67666

The compiler could generate incorrect code for a Scalable Vector Extension version 2 (SVE2) intrinsic of the form `svwhilele_*()` defined in the `<arm_sve.h>` system header. This has been fixed.

For more information about SVE2 intrinsics, see https://developer.arm.com/architectures/instruction-sets/intrinsics.

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

SDCOMP-67662

The compiler could generate incorrect code for an M-profile Vector Extension (MVE) intrinsic of the form `*vcmlaq_*_f32()` defined in the `<arm_mve.h>` system header. This has been fixed.

For more information about MVE intrinsics, see https://developer.arm.com/architectures/instruction-sets/intrinsics.

SDCOMP-67658

When compiling for AArch32 state, the compiler could generate incorrect code for the `__builtin_trap()` built-in function. This has been fixed.

`__builtin_trap()` is a [COMMUNITY] feature.

SDCOMP-67650

When compiling at any optimization level except `-O0`, the compiler could generate incorrect code for the following forms of M-profile Vector Extension (MVE) intrinsics defined in the `<arm_mve.h>` system header:

- `*vfmaq_m*()`
- `*vfmsq_m*()`
- `*vfmasq_m*()`

This has been fixed.

For more information about MVE intrinsics, see https://developer.arm.com/architectures/instruction-sets/intrinsics.

SDCOMP-67638

The compiler could generate incorrect code for an M-profile Vector Extension (MVE) intrinsic `z` of the form `*vcmp*_f*()` defined in the `<arm_mve.h>` system header, where a parameter to `z` includes a NaN. This has been fixed.

For more information about MVE intrinsics, see https://developer.arm.com/architectures/instruction-sets/intrinsics.

SDCOMP-67582

When compiling for AArch64 state, the compiler and integrated assembler incorrectly did not always enable the Statistical Profiling Extensions version 1.2 feature (FEAT_SPEv1p2). This has been fixed.

SDCOMP-67424

When assembling for AArch32 state, the inline assembler and integrated assembler incorrectly failed to report an error for a `MOVT` or `MOVW` instruction which specifies a source operand with an offset that is outside the range for the instruction. This has been fixed. The inline assembler and integrated assembler now report the following error:

- `Relocation Not In Range`

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

SDCOMP-66854

When compiling for AArch64 state, the compiler could generate an incorrect call frame information directive for a function with pointer authentication branch protection using the Program Counter as a second diversifier for return address signing. This has been fixed.

Call frame information directives are required for debugging and for C++ exception unwinding.

SDCOMP-66632

When assembling for AArch64 state, the integrated assembler incorrectly failed to implicitly set the minimum alignment requirement for a user-defined executable section to 4 bytes. This has been fixed.

SDCOMP-66569

When compiling for a target with the Scalable Vector Extension (SVE), at any optimization level except `-O0`, without `-fno-vectorize`, and with `-mno-unaligned-access`, the compiler could incorrectly report the following fatal error:

- `Invalid size request on a scalable vector`

This has been fixed.

SDCOMP-65590

When compiling at any optimization level except `-O0`, the compiler could generate incorrect code for a function that contains a call to an M-profile Vector Extension (MVE) intrinsic of the form `*vsetq_lane_*32()` defined in the `<arm_mve.h>` system header. This has been fixed.

For more information about MVE intrinsics, see https://developer.arm.com/architectures/instruction-sets/intrinsics.

SDCOMP-65579

The compiler could generate incorrect code for a Scalable Vector Extension (SVE) intrinsic of the form `svuzp*()` defined in the `<arm_sve.h>` system header. This has been fixed.

For more information about SVE intrinsics, see https://developer.arm.com/architectures/instruction-sets/intrinsics.

SDCOMP-65550

When compiling for AArch64 state with target options that enable the Advanced SIMD Extension (FEAT_AdvSIMD), the compiler could generate incorrect code. This has been fixed.

SDCOMP-64255

When assembling for AArch32 state, the inline assembler and integrated assembler incorrectly failed to report an error for a `DMB`, `DSB`, or `ISB` instruction that has an invalid operand. This has been fixed. The inline assembler and integrated assembler now report the following error:

- `expected an immediate or barrier type`

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

SDCOMP-64194

When compiling for an Armv6-M target or an Armv8-M target without the Main Extension, and with `-mframe-chain=aapcs`, the compiler could generate incorrect code for the `__builtin_return_address(<N>)` built-in function. This has been fixed.

`__builtin_return_address(<N>)` is a [COMMUNITY] feature.

SDCOMP-63984

When compiling code that contains a thread-local variable with `-mexecute-only`, the compiler incorrectly failed to report the following error:

- `thread-local storage is not supported for the current target`

This has been fixed.

SDCOMP-63205

When compiling for AArch32 state, a big-endian target that includes the Advanced SIMD Extension (FEAT_AdvSIMD), and without `-fno-vectorize`, the compiler could generate incorrect code for a loop. This has been fixed.

SDCOMP-63114

When compiling with `-fsanitize=memtag-stack`, the compiler could generate an incorrect `STG` instruction. This has been fixed.

SDCOMP-62378

When compiling for AArch32 state, the compiler could generate incorrect code for a Neon intrinsic of the form `vld*_x*()` or `vst*_x*()` defined in the `<arm_neon.h>` system header. This has been fixed.

For more information about Neon intrinsics, see https://developer.arm.com/architectures/instruction-sets/intrinsics.

SDCOMP-55200

When compiling with target options that enable the M-profile Vector Extension (MVE) for floating-point types, and with `-mfpu=fpv5-d16` or `-mfpu=fpv5-sp-d16`, the compiler incorrectly set bit 1 of the Arm C Language Extensions (ACLE) feature macro `__ARM_FEATURE_MVE`. This has been fixed.

SDCOMP-55040

When compiling at any optimization level except `-O0`, with `-fsanitize=memtag-stack`, and for a target that supports the Memory Tagging Extension (FEAT_MTE), the compiler could generate incorrect code. This has been fixed.

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

3.2.2 Linker, armlink

This section contains a list of defect fixes made in the linker, `armlink`.

SDCOMP-65720

When linking for AArch64 state, the linker could incorrectly report the following error:

- `L6286E: Relocation #RELA:<N> in <object>(<section>) with respect to <symbol>. Value(<value>) out of range(<range>) for (<relocation_type>)`

This has been fixed.

3.2.3 ELF processing utility, fromelf

This section contains a list of defect fixes made in the ELF processing utility, `fromelf`.

SDCOMP-66692

The fromelf utility could disassemble the `VSCCLRM` instruction incorrectly. This has been fixed.

3.2.4 Libraries and system headers

This section contains a list of defect fixes made in the C and C++ libraries and system headers supplied with the toolchain.

SDCOMP-68070

The Arm C library implementation of the `__heapvalid()` function could incorrectly fail to detect an invalid heap. This has been fixed.

SDCOMP-63111

The Arm C Library implementation of BFloat16 intrinsics of the form `vcvt*_f32_bf16()` defined in the `<arm_neon.h>` system header incorrectly relied on C undefined behavior. This has been fixed.

For more information about BFloat16 intrinsics, see https://developer.arm.com/architectures/instruction-sets/intrinsics.

SDCOMP-45879

The Arm C library implementations of the `bsearch()` and `qsort()` functions could incorrectly corrupt the stack when used to process an array larger than 4GB. This has been fixed.

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

4. Support

This chapter includes guidance on how to obtain support for using Arm Compiler for Embedded 6.24.

Your feedback is important to us, and you are welcome to send us defect reports and suggestions for improvement on any aspect of the product. Please contact your supplier or open a case with feedback or support issues, using your work or academic email address if possible. Where appropriate, please provide the following information:

- `--vsn` output from the tool.
- The complete content of any error message that the tool produces.
- Preprocessed source code, other files, and command-line options necessary to reproduce the issue. For information on how to preprocess source code, see the *-E* section of the Arm Compiler for Embedded Reference Guide.

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

5. Release history

This chapter contains the history of Arm Compiler for Embedded releases.

Version	Release Date
Arm Compiler 6.00	10 Apr 2014
Arm Compiler 6.00 update 1	29 May 2014
Arm Compiler 6.00 update 2	30 Oct 2014
Arm Compiler 6.01	18 Dec 2014
Arm Compiler 6.01 update 1	9 Feb 2015
Arm Compiler 6.01 update 2	20 Mar 2015
Arm Compiler 6.02	26 Jun 2015
Arm Compiler 6.3	17 Nov 2015
Arm Compiler 6.4	1 Mar 2016
Arm Compiler 6.5	30 Jun 2016
Arm Compiler 6.6	10 Nov 2016
Arm Compiler 6.7	31 Mar 2017
Arm Compiler 6.7.1	31 May 2017
Arm Compiler 6.8	27 Jul 2017
Arm Compiler 6.9	25 Oct 2017
Arm Compiler 6.10	15 Mar 2018
Arm Compiler 6.10.1	13 Jun 2018
Arm Compiler 6.11	25 Oct 2018
Arm Compiler 6.12	28 Feb 2019
Arm Compiler 6.13	10 Oct 2019
Arm Compiler 6.13.1	18 Nov 2019
Arm Compiler 6.14	28 Feb 2020
Arm Compiler 6.14.1	9 Jun 2020
Arm Compiler 6.15	9 Oct 2020
Arm Compiler 6.16	9 Mar 2021
Arm Compiler for Embedded 6.17	20 Oct 2021
Arm Compiler for Embedded 6.18	31 Mar 2022
Arm Compiler for Embedded 6.19	12 Oct 2022
Arm Compiler for Embedded 6.20	15 Mar 2023
Arm Compiler for Embedded 6.20.1	25 Apr 2023
Arm Compiler for Embedded 6.21	11 Oct 2023
Arm Compiler for Embedded 6.22	19 Mar 2024
Arm Compiler for Embedded 6.23	16 Oct 2024
Arm Compiler for Embedded 6.24	31 Mar 2025

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

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

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

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

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in Arm documents.

Product status

All products and services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

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

Revision history

These sections can help you understand how the document has changed over time.

Document release information

The Document history table gives the issue number and the released date for each released issue of this document.

Document history

Issue	Date	Confidentiality	Change
062400-00	31 March 2025	Non-Confidential	Initial release

Change history

Arm does not provide a detailed list of changes between different revisions of this document. For a list of changes between different releases of Arm Compiler for Embedded in the release history of Arm Compiler for Embedded, refer to the *Release Notes* for each release.

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

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. 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 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**.

Caution

We recommend the following. If you do not follow these recommendations your system might not work.

Warning

Your system requires the following. If you do not follow these requirements your system will not work.

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

You are at risk of causing permanent damage to your system or your equipment, or harming yourself.

This information is important and needs your attention.

A useful tip that might make it easier, better or faster to perform a task.

A reminder of something important that relates to the information you are reading.

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Arm documents are available on developer.arm.com/documentation.

Confidential documents are only available to licensees, when logged in. Each document link in the tables below provides direct access to the online version of the document.

Arm product resources	Document ID	Confidentiality
Arm Compiler for Embedded	–	Non-Confidential
Arm Compiler for Embedded FuSa	–	Non-Confidential
Arm Compiler for Embedded FuSa 6.22LTS documentation index	KA006002	Non-Confidential
Arm Compiler for Embedded Migration and Compatibility Guide	100068	Non-Confidential
Arm Compiler for Embedded Reference Guide	101754	Non-Confidential
Arm Compiler for Embedded User Guide	100748	Non-Confidential
Arm Compiler for Embedded documentation index	KA005061	Non-Confidential
Arm Development Studio	–	Non-Confidential
Arm Development Studio Getting Started Guide	101469	Non-Confidential
Arm Keil MDK v6	–	Non-Confidential
Arm Keil Studio Visual Studio Code Extensions User Guide	108029	Non-Confidential
Arm Toolchain for Embedded	–	Non-Confidential
Arm Toolchain for Embedded documentation index	KA006292	Non-Confidential
Does Arm document all known issues that affect each Arm Compiler release?	KA005052	Non-Confidential
User-based Licensing User Guide	102516	Non-Confidential
uVision User's Guide	101407	Non-Confidential

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential