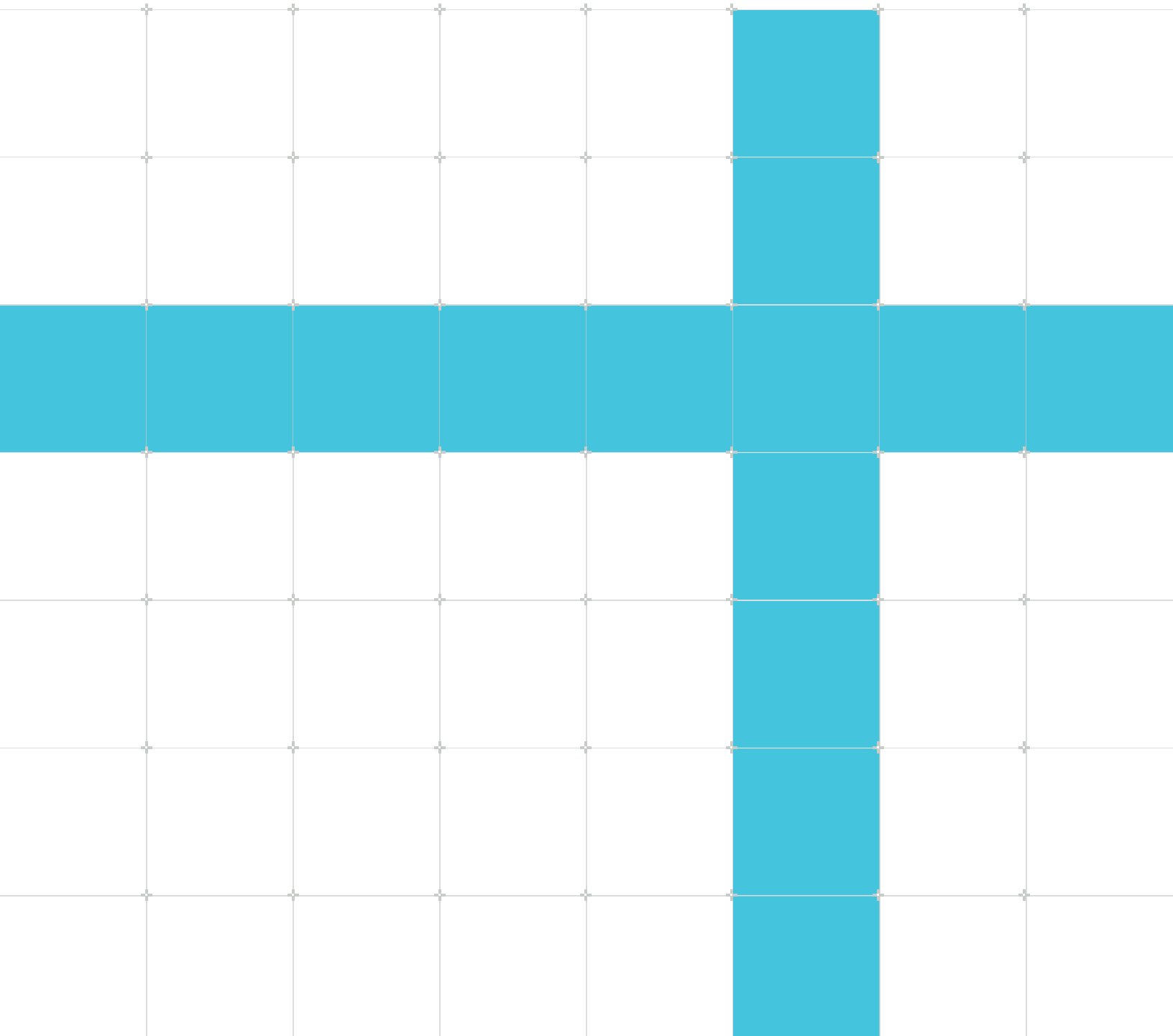
Arm Compiler for Embedded FuSa 6.16.3 Release Notes

Version 6.16.3

Issue 01
110644_061603_01_en



Arm Compiler for Embedded FuSa 6.16.3 Release Notes

This document (110644_061603_01_en) was issued on 2026-01-29. There might be a later issue at https://developer.arm.com/documentation/110644

The product version is 6.16.3.

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 FuSa to build a project with functional safety or long-term maintenance requirements. The document includes an overview of the Arm Compiler for Embedded FuSa 6.16.3 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

1. Release overview

This chapter provides an overview of the Arm Compiler for Embedded FuSa 6.16LTS product and the Arm Compiler for Embedded FuSa 6.16.3 release.

1.1 Product description

Arm Compiler for Embedded FuSa 6.16LTS is an embedded C/C++ compilation toolchain from Arm for the development of bare-metal software, firmware, and Real-Time Operating System (RTOS) applications with functional safety (FuSa) or long-term support requirements.

The toolchain is compatible with the following Arm Integrated Development Environments (IDEs):

- Arm Development Studio
- Arm Keil MDK v6
- Arm Keil MDK v5 (requires a Windows x86_32 toolchain installation on a Windows x86_64 host platform)

Through powerful optimization techniques and optimized libraries, Arm Compiler for Embedded FuSa enables functional safety embedded system developers to meet challenging performance goals and memory constraints.

Arm Compiler for Embedded FuSa is used by leading companies in a wide variety of industries, including automotive (ISO 26262), consumer electronics, industrial (IEC 61508), medical (IEC 62304), networking, railway (EN 50716), storage, and telecommunications.

The key features of Arm Compiler for Embedded FuSa 6.16LTS include:

- Support for Cortex and Neoverse processors
- Support for dynamic linking for A-profile and R-profile targets
- Support for Thread-Local Storage (TLS)
- Support for C++14 source language modes
- Compatibility with the Arm Certified C Library and the Arm Certified C++ Library

For a summary of changes between Arm Compiler for Functional Safety 6.6 and Arm Compiler for Embedded FuSa 6.16LTS, see the *Summary of changes between Arm Compiler 6.6 LTM and Arm Compiler for Embedded FuSa 6.16LTS* section of the *Arm Compiler for Embedded FuSa Migration and Compatibility Guide version 6.16.2* document.

Contact your sales representative or submit an inquiry online to find out more about licensing Arm Compiler for Embedded FuSa or an Arm IDE.

1.2 Release highlights

Arm Compiler for Embedded FuSa 6.16.3 is the final planned maintenance release of Arm Compiler for Embedded FuSa 6.16LTS, and is the latest release as of January 2026. It has been derived from the unqualified Arm Compiler for Embedded 6.16 release, includes additional defect fixes, and adds a Qualification Kit for functional safety purposes.

Subject to your license terms, Arm Compiler for Embedded FuSa 6.16.3 can be used to build for the following Arm Architectures and Processors:

Architecture	Processor Family	Standard Processors	Automotive Enhanced Processors
Armv8-A up to Armv8.7-A	Neoverse	V1 N1 E1	-
	Cortex	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 [BETA]	Cortex	R82 [BETA]	-
Armv8-R	Cortex	R52+, R52	-
Armv7-R	Cortex	R8, R7, R5, R4F, R4	-
Armv8-M up to Armv8.1-M	Cortex	M55 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 *Arm Keil MDK v5 Release Notes*.

- The *Support level definitions* section of the *Arm Compiler for Embedded FuSa User Guide version 6.16.2* document applicable to this release.

1.3 Included components

This section lists the toolchain components and different types of documentation available for Arm Compiler for Embedded FuSa 6.16.3.

Category	Component	Description
Qualified 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 dissembler
Unqualified toolchain components	Arm C libraries	Runtime support libraries for embedded systems. Projects that require a qualified C library can use the Arm Certified C Library instead. Arm Certified C Library releases are a separate product from Arm Compiler for Embedded FuSa, and must be installed independently from Arm Compiler for Embedded FuSa 6.16.3. For more information, see Arm Certified C Library, Arm Certified C++ Library, and Arm Compiler for Embedded FuSa compatibility.
	Arm C++ libraries	Libraries based on the LLVM libc++ project. Part of the unqualified Arm C++ libraries can be used in a project with functional safety requirements, when also using the Arm Certified C Library. For more information, see Arm Certified C Library, Arm Certified C++ Library, and Arm Compiler for Embedded FuSa compatibility.
	`armasm`	Deprecated legacy assembler for armasm-syntax assembly code for older Arm architectures only. Use the `armclang` integrated assembler for all new assembly files.
Qualification Kit	Safety Manual	Describes the scope of qualification, and how to use the toolchain for safety-related development
	Defect Report	Provides information about known safety-related defects at the time of release
	Test Report	Contains results from language conformance tests
	Development Process	Contains an overview of the process used to develop the toolchain
	Release History	Identification information for all the releases to date of the Arm Compiler for Embedded FuSa 6.16LTS release series
User documentation	User Documentation and Qualification Kit Addendum	Provides information about which User Documentation and Qualification Kit documents are applicable to Arm Compiler for Embedded FuSa 6.16.3, and information about new Product Features added to Arm Compiler for Embedded FuSa 6.16.3
	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 Arm libraries and floating-point support

Category	Component	Description
	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 FuSa
	Defect Notification Report	A regularly updated version of the *Qualification Kit Defect Report*
	Release Notes	These release notes

These components can be obtained from the following sources:

Component type	Source
Qualified toolchain components	Available via Product Download Hub (PDH)
Unqualified toolchain components	Available via PDH
Qualification Kit	Available via PDH
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 the following:

- The *Changes applicable to the Arm Compiler for Embedded FuSa 6.16.2 Reference Guide for use with Arm Compiler for Embedded FuSa 6.16.3* section of the *Arm Compiler for Embedded FuSa 6.16.3 User Documentation and Qualification Kit Addendum*.
- The *Features not recommended for use in safety-related development* section of the *Arm Compiler for Embedded FuSa 6.16LTS Qualification Kit Safety Manual version 6.16.2*.
- The *Support level definitions* section of the *Arm Compiler for Embedded FuSa User Guide version 6.16.2*.

2. Download Arm Compiler for Embedded FuSa 6.16.3

This chapter provides information about how to download and install Arm Compiler for Embedded FuSa 6.16.3.

Arm Compiler for Embedded FuSa 6.16.3 might be available to download standalone or as part of an Arm Integrated Development Environment (IDE) depending on your license and entitlements.

To download this release standalone, use the `ACOMP616` 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 January 2026*:

Host architecture	Host operating system	Toolchain download package	Host environment
x86_64	Red Hat Enterprise Linux 8 Red Hat Enterprise Linux 7 Ubuntu 24.04 LTS Ubuntu 22.04 LTS Ubuntu 20.04 LTS Ubuntu 18.04 LTS	x86_64 Linux	Standalone installation Integrated into Arm Development Studio Integrated into Keil MDK version 6
	Windows Server 2022 Windows Server 2019 Windows Server 2016 Windows Server 2012 Windows 11 Windows 10 Windows 8.1	x86_64 Windows	Standalone installation Integrated into Arm Development Studio Integrated into Keil MDK version 6
		for Keil MDK	Integrated into Keil MDK version 5

*Arm reserves the right to add additional environments for this release. For more information, see *Which host architectures, host operating systems, and host environments can I use with functional safety compiler toolchains?*.

The following restrictions apply:

- Windows x86_32 host platforms are not supported.
- The minimum required version of glibc for Linux x86_64 host platforms is 2.6.
- The toolchain must not be installed directly into an Arm Development Studio installation directory.

- For use with Keil MDK version 5:
  - The toolchain must be installed into the `ARM` sub-directory of the Keil MDK version 5 installation directory. For example, if your Keil MDK version 5 installation directory is `C:\Keil_v5`, the recommended installation path is `C:\Keil_v5\ARM\ARMCompiler6.16.3`.
  - The toolchain is supported on Windows host platforms only.
- If you are using a FlexNet Publisher (FNP) floating license, your license server must be running version 11.14.1.0 or later of both `armlmd` and `lmgrd`. Arm recommends that you always use the latest version of the license server software.

For more information, see the following:

- The *Changes to the Installing a standalone Arm Compiler for Embedded FuSa on Windows platforms section* section of the *Arm Compiler for Embedded FuSa 6.16.3 User Documentation and Qualification Kit Addendum*.
- The *System requirements and installation* section of the *Arm Compiler for Embedded FuSa User Guide version 6.16.2* 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 *Manage Arm Compiler Versions* section of the *Vision User's Guide* for instructions to integrate the toolchain into Keil MDK version 5.
- 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).
- *Product and toolkit configuration for FlexNet Publisher (FNP) licenses* for instructions to configure the toolchain to use a FlexNet Publisher (FNP) license.

3. Differences from previous release

This chapter describes differences from the previous release, Arm Compiler for Embedded FuSa 6.16.2.

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 Arm Compiler for Embedded FuSa 6.16.3.

SDCOMP-69300

The `__attribute__((target("branch-protection=<protection>")))` function attribute to control branch protection on a per-function basis has been changed from a Community Feature to a Product Feature when all the following are true:

- The program is compiled for AArch64 state.
- `<protection>` includes `pac-ret` and the function attribute is used in combination with `__attribute__((target("harden-pac-ret=load-return-address")))`.

For more information, see the *__attribute__((target("branch-protection=<protection>"))) function attribute* section of the *Arm Compiler for Embedded FuSa 6.16.3 User Documentation and Qualification Kit Addendum*.

SDCOMP-64414

The `-isystem <directory>` compiler option to specify a directory as a system header directory has been changed from a Community Feature to a Product Feature when it is used to add a directory containing the following:

- The Arm Certified C Library header files.
- The header files in the `include/libcxx` directory within the installation directory of Arm Compiler for Embedded FuSa 6.16.3.

For more information, see the *-isystem <directory> command-line option for the compiler and integrated assembler, armclang* section of the *Arm Compiler for Embedded FuSa 6.16.3 User Documentation and Qualification Kit Addendum*.

SDCOMP-63881

The Arm C library Default Initialization Sequence in the standardlib variants of the Arm C Library now restricts the scatter-loading mechanism to using only the contents of the Region Table.

SDCOMP-63862

The following Product Features have been added to the compiler for return address signing hardening for AArch64 state as a mitigation for the PACMAN vulnerability:

- The `-mharden-pac-ret` command-line option.
- The `__attribute__((target("harden-pac-ret=<value>")))` function attribute.

For more information, see:

- The *-mharden-pac-ret command-line option for the compiler and integrated assembler, armclang* section of the *Arm Compiler for Embedded FuSa 6.16.3 User Documentation and Qualification Kit Addendum*.
- The *__attribute__((target("harden-pac-ret=<value>"))) function attribute* section of the *Arm Compiler for Embedded FuSa 6.16.3 User Documentation and Qualification Kit Addendum*.
- The *Arm CPU Security Update: Pointer Authentication* article, available at https://developer.arm.com/documentation/110389.

SDCOMP-63605

The `-nobuiltininc` compiler option has been changed from a Community Feature to a Product Feature when it is used to prevent the compiler from searching the default system headers directory for the purpose of using both the Arm Certified C Library and the Arm Certified C++ Library.

For information, see the *-nobuiltininc command-line option for the compiler and integrated assembler, armclang* section of the *Arm Compiler for Embedded FuSa 6.16.3 User Documentation and Qualification Kit Addendum*.

SDCOMP-63444

The Linux installer now mitigates against the *Insecure verification of installation file signature Vulnerability (CVE-2022-43702)*.

For more information, see https://developer.arm.com/documentation/110371/latest.

SDCOMP-62927

The `-mharden-sls=<option>` compiler option for Straight-Line Speculation hardening for AArch64 state has been changed from a Community Feature to a Product Feature.

For more information, see:

- The *-mharden-sls command-line option for the compiler and integrated assembler, armclang* section of the *Arm Compiler for Embedded FuSa 6.16.3 User Documentation and Qualification Kit Addendum*.
- The *Straight-line speculation whitepaper*, available at https://developer.arm.com/documentation/102825.
- The *Straight-Line Speculation (SLS) hardening supplement for Arm Compiler for Embedded FuSa 6.16LTS*, available at https://developer.arm.com/documentation/ka005806.

SDCOMP-62131

The following linker warning has been downgraded to a remark when linking with `--sysv`:

- `L6314W: No section matches pattern <pattern>`

This remark can be upgraded to an error by linking with `--diag_error=L6314W`, or to a warning by linking with `--diag_warning=L6314W`.

SDCOMP-61795

The installation package structure for Windows host platforms has been changed. The `setup.exe` file has been removed. To install the toolchain, use the `.msi` file instead.

For more information, see the *The Installing a standalone Arm Compiler for Embedded FuSa on Windows platforms section of the User Guide* section of the *Arm Compiler for Embedded FuSa 6.16.3 User Documentation and Qualification Kit Addendum*.

SDCOMP-59807

The following Product Features have been added to the compiler for the mitigation for Cortex-A57 Erratum 1742098 and Cortex-A72 Erratum 1655431:

- The `-mfix-cortex-a57-aes-1742098` and `-mno-fix-cortex-a57-aes-1742098` command-line options.
- The `-mfix-cortex-a72-aes-1655431` and `-mno-fix-cortex-a72-aes-1655431` command-line options.

This change only affects compilation. It does not affect the inline assembler or the integrated assembler.

When compiling for AArch32 state and for a target that supports the Advanced SIMD AES instructions, the default behavior of the compiler is now as follows:

Target	Default behavior
Cortex-A57	Mitigation enabled
Cortex-A72	Mitigation enabled
All other targets	Mitigation disabled

To enable the mitigation when it is disabled by default, compile with `-mfix-cortex-a57-aes-1742098` or `-mfix-cortex-a72-aes-1655431`.

To disable the mitigation when it is enabled by default, compile with `-mno-fix-cortex-a57-aes-1742098` or `-mno-fix-cortex-a72-aes-1655431`.

`-mfix-cortex-a57-aes-1742098` is an alias of `-mfix-cortex-a72-aes-1655431`, and `-mno-fix-cortex-a57-aes-1742098` is an alias of `-mno-fix-cortex-a72-aes-1655431`.

For more information about Cortex-A57 Erratum 1742098, Cortex-A72 Erratum 1655431, and the mitigation, see the following documents:

- Arm Cortex-A57 MPCore Software Developers Errata Notice

- Cortex-A72 MPCore (MP054) Software Developers Errata Notice

3.2 Defect fixes

This section contains information about defect fixes made in this release of Arm Compiler for Embedded FuSa. 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-67544

When compiling for AArch32 state, the compiler could generate incorrect code for a `volatile` variable of a single-precision floating-point type. This has been fixed.

SDCOMP-66676

When compiling for a big-endian target and AArch64 state, the compiler could generate incorrect code for an access to a `volatile` 128-bit variable. This has been fixed.

SDCOMP-66132

When compiling at any optimization level except `-O0`, the compiler could generate code that incorrectly corrupts a member of a `class` or `struct`. This has been fixed.

SDCOMP-65264

When building for an Armv8.6-A or later target with the Scalable Vector Extension (SVE), and with an `-march` or `-mcpu` option that does not specify `+f32mm`, the compiler and integrated assembler incorrectly failed to report an error for a 32-bit element `FMMLA` instruction. This has been fixed. The compiler and integrated assembler now report the following error:

- `instruction requires: f32mm`

SDCOMP-65172

When compiling with `-gdwarf-2` or `-gdwarf-3`, the compiler could generate incorrect debug information for a bit-field. This has been fixed.

SDCOMP-64591

When compiling for AArch32 state, the compiler could generate incorrect code for the following Neon intrinsics defined in the `<arm_neon.h>` system header:

- `vld2q_dup_bf16()`
- `vld2q_dup_f16()`
- `vld2q_dup_f32()`
- `vld2q_dup_p16()`

- `vld2q_dup_p8()`
- `vld2q_dup_s16()`
- `vld2q_dup_s32()`
- `vld2q_dup_s8()`
- `vld2q_dup_u16()`
- `vld2q_dup_u32()`
- `vld2q_dup_u8()`

This has been fixed.

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

SDCOMP-64268

When using the toolchain with a User-based License (UBL), the tools could incorrectly report one of the following errors:

- `License capability data invalid`
- `Your cached license information has expired and could not be refreshed`

This has been fixed.

SDCOMP-64165

When assembling for A32 state, the inline assembler and integrated assembler could generate incorrect code for a PC-relative `ADR` instruction or a load literal instruction. This has been fixed.

SDCOMP-64066

The compiler could generate code that incorrectly uses the same register for both operands of the `__arm_sqrshr()` and `__arm_uqrshl()` MVE intrinsics 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-64059

The compiler could generate incorrect code for an M-profile Vector Extension (MVE) intrinsic `I` defined in the `<arm_mve.h>` system header, where `I` is called with an argument created using an MVE intrinsic of the form `*vuninitializedq*()`. This has been fixed.

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

SDCOMP-63917

When assembling for AArch64 state, the inline assembler and integrated assembler incorrectly failed to report an error for a `FMLAL` (by element) or `FMLAL2` (by element) instruction that specifies

an invalid second source register of the form `<Vm>.H[<index>]`. This has been fixed. The inline assembler and integrated assembler now report the following error:

- `invalid operand for instruction`

SDCOMP-63913

When compiling with a `-mharden-sls=<option>` option that enables the mitigation against Straight-Line Speculation (SLS) for `RET` and `BR` instructions, and with `-moutline` or at `-Oz` without `-mno-outline`, the compiler could generate incorrect code. This has been fixed.

SDCOMP-63894

When compiling for a big-endian target and AArch64 state, the compiler could generate incorrect code for a program that contains a `volatile` variable. This has been fixed.

SDCOMP-63761

When compiling for AArch64 state, and for the general dynamic thread local storage (TLS) model, the compiler could generate incorrect code for a function that accesses a `__thread` or `thread_local` variable. This has been fixed.

SDCOMP-63752

When compiling for AArch64 state, and for a target that supports the floating-point half-precision multiplication instructions feature (FEAT_FHM), the compiler could generate incorrect code for a `vfmlalq_laneq_high_f16()` or `vfmlalq_laneq_low_f16()` Neon intrinsic 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-63697

The compiler could incorrectly generate an output file and exit with a return code of `0` despite reporting an error. This has been fixed. The return code is also referred to as the status code.

SDCOMP-63688

When compiling with the stack protection feature enabled, and for an Armv6-M target or an Armv8-M target without the Main Extension, the compiler could generate incorrect code that corrupts register `R4`. This has been fixed.

SDCOMP-63454

When compiling for a big-endian target and T32 state, the compiler could generate incorrect code for a PC-relative `ADR` instruction or a load literal instruction that refers to a symbol in a different ELF section than the instruction. This has been fixed.

SDCOMP-62791

When compiling for AArch32 state, at any optimization level except `-O0`, with `-mfloat-abi=hard`, and for a target with half-precision floating-point support, the compiler could generate code that incorrectly does not preserve the half-precision floating-point value returned by one function when making a call to another function. This has been fixed.

SDCOMP-62725

When compiling with `-mfloat-abi=hard` for a target that includes only MVE integer support, the compiler generated code that incorrectly did not conform to the *Procedure Call Standard for the Arm Architecture*. This has been fixed.

SDCOMP-62692

When compiling for AArch32 state, the compiler could generate code that incorrectly performed an unaligned access using a `LDRD` or `STRD` instruction for an access to a `struct`. This has been fixed.

SDCOMP-62661

When compiling for AArch64 state, the compiler could generate incorrect code for a call to a variadic function. This has been fixed.

SDCOMP-62352

The compiler and integrated assembler could incorrectly generate a `VMLA.U32` or `VMLAS.U32` instruction in accordance with revision B.q of the Armv8-M Architecture. This has been fixed. The compiler and integrated assembler now generate a `VMLA.I32` or `VMLAS.I32` instruction in accordance with revision B.r of the Armv8-M Architecture.

SDCOMP-62330

When compiling with `-g` or `-gdwarf-4`, the compiler could generate incorrect debug information for a bit-field. This has been fixed.

SDCOMP-62245

Previously, the toolchain could not be used with a legacy FlexNet Publisher (FNP) license for DS-5 Ultimate without applying the workaround documented at https://developer.arm.com/documentation/ka005517/latest. This has been fixed.

SDCOMP-62234

The compiler incorrectly failed to report a warning for an invalid `#undef` preprocessor macro which specifies a name that is lexically identical to a predefined macro name as listed in the *Predefined macro names* section of the relevant C standard specification. This has been fixed. The compiler now reports the following warning:

- `undefining builtin macro`

SDCOMP-62221

When compiling for an Armv6-M target, the compiler could incorrectly generate a `B.W` instruction. This has been fixed.

SDCOMP-62123

The compiler could generate incorrect code for a `*vdup*_*f16()` Neon intrinsic 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-62028

When compiling for an Armv8-M target with the Main Extension, the compiler could generate incorrect code for an atomic compare exchange built-in or an atomic compare exchange function. This has been fixed.

SDCOMP-61298

When compiling for a target without hardware floating-point support, the compiler could incorrectly fail to define the `__SOFTFP__` predefined macro. This has been fixed.

SDCOMP-61089

When compiling with `-mfloat-abi=hard` for AArch32 state, and for a target that does not support hardware floating-point instructions, the compiler incorrectly failed to report a warning. This has been fixed. The compiler now reports the following warning:

- `'-mfloat-abi=hard': selected processor lacks floating point registers`

SDCOMP-61080

The compiler could generate incorrect code for an expression that involves both the >> and << operators. This has been fixed.

SDCOMP-60897

When compiling with `-mcmse`, the compiler could incorrectly fail to define certain predefined macros when the toolchain is used with user-based licensing. This has been fixed.

SDCOMP-60872

When using the toolchain with a User-based License (UBL), the tools incorrectly reported `Component: Arm Compiler for Embedded <version>` in the output of the `--vsn` command-line option. This has been fixed. The tools now report `Component: Arm Compiler for Embedded FuSa <version>`.

SDCOMP-60632

When assembling for a big-endian target and AArch64 state, the inline assembler and integrated assembler could incorrectly generate a `FNMADD` instruction instead of a **`NOP`** instruction for a `.align`, `.balign`, or `.p2align` assembly directive. This has been fixed.

SDCOMP-60589

When compiling in a C source language mode, the compiler could generate code that incorrectly did not conform to the *Procedure Call Standard for the Arm Architecture* for a function with a prototype that has a homogenous floating-point aggregate (HFA) `struct` containing a zero-length bit-field. This has been fixed.

SDCOMP-60443

When compiling with `-frwpi`, and with `-g` or `-gdwarf-<version>`, the compiler could generate incorrect debug information for a global variable that is `const`. This has been fixed.

SDCOMP-60342

The compiler could generate incorrect code for a call to a formatted input/output function declared in the `<stdio.h>` header or a formatted wide character input/output function declared in the `<wchar.h>` header, when the format string is a global variable. This has been fixed.

SDCOMP-59788

When compiling for an Armv7-M or Armv8-M target and at any optimization level except `-O0`, the compiler could incorrectly generate one of the following instructions when accessing an element of a `char` or `short` array:

- `LDRBT`
- `LDRHT`
- `LDRSBT`
- `LDRSHT`
- `STRBT`
- `STRHT`

This has been fixed.

SDCOMP-59521

When compiling at any optimization level except `-O0`, the compiler could generate incorrect code for a call to an empty user-defined implementation of `operator delete`. This has been fixed.

SDCOMP-59190

When compiling in a C++ source language mode, the compiler could incorrectly fail to report an error for an ambiguous reference. This has been fixed. The compiler now reports the following error:

- `reference to '<name>' is ambiguous`

SDCOMP-57674

When compiling at any optimization level except `-O0`, the compiler could incorrectly fail to flush a denormal floating-point number to zero. This has been fixed.

SDCOMP-55580

When assembling for A32 state, the inline assembler and integrated assembler incorrectly failed to report an error for a `VLDR` instruction that loads the address of a label that is not in the same section as the instruction. This has been fixed. The inline assembler and integrated assembler now report the following error:

- `unsupported relocation type`

SDCOMP-46790

The compiler incorrectly failed to report a warning when compiling for a target without a hardware floating-point unit, and with `-mfloat-abi=hard`. This has been fixed. The compiler now reports the following warning:

- `'-mfloat-abi=hard': selected processor lacks floating point registers`

3.2.2 Librarian, armar

This section contains a list of defect fixes made in the librarian, `armar`.

SDCOMP-64268

When using the toolchain with a User-based License (UBL), the tools could incorrectly report one of the following errors:

- `License capability data invalid`
- `Your cached license information has expired and could not be refreshed`

This has been fixed.

SDCOMP-62245

Previously, the toolchain could not be used with a legacy FlexNet Publisher (FNP) license for DS-5 Ultimate without applying the workaround documented at https://developer.arm.com/documentation/ka005517/latest. This has been fixed.

SDCOMP-60872

When using the toolchain with a User-based License (UBL), the tools incorrectly reported `Component: Arm Compiler for Embedded <version>` in the output of the `--vsn` command-line option. This has been fixed. The tools now report `Component: Arm Compiler for Embedded FuSa <version>`.

3.2.3 Linker, armlink

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

SDCOMP-65517

For two execution regions `A` and `B`, where `A` is at a lower address than `B`, the linker incorrectly assumed that the name of `A` is always lexically earlier than the name of `B`. Subsequently, this could result in the linker generating incorrect callgraph or stack usage information. This has been fixed. The linker no longer assumes that the name of `A` is always lexically earlier than the name of `B`.

SDCOMP-64999

When linking an input object that has been compiled with C++ exceptions enabled and for AArch32 state, the linker could generate an incorrect C++ exception-handling table. This has been fixed.

SDCOMP-64268

When using the toolchain with a User-based License (UBL), the tools could incorrectly report one of the following errors:

- `License capability data invalid`

- `Your cached license information has expired and could not be refreshed`

This has been fixed.

SDCOMP-62251

The linker could incorrectly calculate too small a value for the `p_memsz` field of the ELF program header describing a load region. This has been fixed.

SDCOMP-62245

Previously, the toolchain could not be used with a legacy FlexNet Publisher (FNP) license for DS-5 Ultimate without applying the workaround documented at https://developer.arm.com/documentation/ka005517/latest. This has been fixed.

SDCOMP-62032

When linking an input object that has been compiled with both C++ exceptions and pointer authentication with Key B enabled, the linker could incorrectly report the following warning:

- `L6860W: Unable to optimize .eh_frame sections: <object>(.eh_frame)(CIE at offset <offset>) augmentation string contains unrecognized character: B`

This has been fixed.

SDCOMP-61150

When linking without `--datacompressor=off`, the linker could generate an image which decompresses an execution region that contains RW data incorrectly. This has been fixed.

SDCOMP-60872

When using the toolchain with a User-based License (UBL), the tools incorrectly reported `Component: Arm Compiler for Embedded <version>` in the output of the `--vsn` command-line option. This has been fixed. The tools now report `Component: Arm Compiler for Embedded FuSa <version>`.

SDCOMP-60659

The linker incorrectly placed all thread-local variables at the same offset in a dynamic symbol table. This has been fixed.

SDCOMP-60583

When linking with Link-Time Optimization (LTO) enabled, and an input object has been compiled in a C++ source language mode, the linker could incorrectly report the following error:

- `L6137E: Symbol <symbol> was not preserved by the LTO codegen but is needed by the image`

This has been fixed.

SDCOMP-59938

When linking for AArch64 state, the output for the following linker options incorrectly failed to account for calls to `static` functions:

- `--callgraph`

- `--info=stack`
- `--info=summarystack`

This has been fixed.

SDCOMP-57199

The linker could incorrectly fail to report a warning for a program that contains a T32 `BL` instruction which branches to an A32 instruction that is not aligned to a 4-byte boundary. This has been fixed. The linker now reports the following warning:

- `L6307W: Relocation #REL:0 in <object>(<section>) with respect to <function>. Branch to unaligned destination`

3.2.4 ELF processing utility, fromelf

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

SDCOMP-64268

When using the toolchain with a User-based License (UBL), the tools could incorrectly report one of the following errors:

- `License capability data invalid`
- `Your cached license information has expired and could not be refreshed`

This has been fixed.

SDCOMP-63738

The `fromelf` utility incorrectly disassembled the label operand of a `WLS` or `WLSTP` instruction as an immediate offset instead of a PC-relative offset. This has been fixed.

SDCOMP-62245

Previously, the toolchain could not be used with a legacy FlexNet Publisher (FNP) license for DS-5 Ultimate without applying the workaround documented at https://developer.arm.com/documentation/ka005517/latest. This has been fixed.

SDCOMP-62217

The `fromelf` utility incorrectly disassembled the label operand of a `LE` or `LETP` instruction as an immediate offset instead of a PC-relative offset. This has been fixed.

SDCOMP-60872

When using the toolchain with a User-based License (UBL), the tools incorrectly reported `Component: Arm Compiler for Embedded <version>` in the output of the `--vsn` command-line option. This has been fixed. The tools now report `Component: Arm Compiler for Embedded FuSa <version>`.

SDCOMP-60725

When disassembling an ELF format input file for Armv8.1-M target with the Main Extension, the `fromelf` utility disassembled `CLRM` instructions incorrectly. This has been fixed.

SDCOMP-60326

When processing an ELF format input file that contains debug information with `-g`, the `fromelf` utility reported incorrect `DW_CFA_def_cfa` and `DW_CFA_def_cfa_offset` entries for stack frame unwinding debug information. This has been fixed.

3.2.5 Unqualified libraries and system headers

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

SDCOMP-66090

The Arm C library implementation of the `calloc(num, size)` function for AArch64 state was incorrect. Subsequently, this could result in one of the following:

- `calloc()` incorrectly failing to return a null pointer when the value of `num*size` is greater than or equal to `(1<<64)`. This could result in unexpected run-time behavior.
- `calloc()` incorrectly always returning a null pointer when the value of `num*size` is greater than or equal to `(1<<32)` and less than `(1<<64)`.

This has been fixed.

SDCOMP-65388

The Arm C library was not thread-safe for a multithreaded program that contains C++ objects with static storage duration. This has been fixed.

SDCOMP-64611

The Arm C and C++ libraries defined the constant `math_errhandling` incorrectly. This has been fixed.

`math_errhandling` is now defined as follows:

-ffp-mode=<value>	math_errhandling value
`fast`	`MATH_ERRNO`
`full`	`MATH_ERRNO | MATH_ERREXCEPT`
`std` (Default)	`MATH_ERRNO`

SDCOMP-64597

The `<arm_neon.h>` system header incorrectly defined the following Neon intrinsics with a signed return type instead of an unsigned return type:

- `vqrshrund_n_s64()`
- `vqrshrunh_n_s16()`

- `vqrshruns_n_s32()`
- `vqshrund_n_s64()`
- `vqshrunh_n_s16()`
- `vqshruns_n_s32()`

This has been fixed.

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

SDCOMP-64555

The Arm C library implementations of the `__aeabi_ddiv()` function for targets without hardware floating-point support could incorrectly round up a result that is less than halfway between two adjacent representable double-precision numbers. This has been fixed.

SDCOMP-64176

The Arm C library variant of the `nearbyint()` function for AArch64 state could return an incorrect result. This has been fixed.

SDCOMP-64025

The Arm C library variant for targets that include the M-profile Vector Extension (MVE) for integer types only had the following incorrect behavior:

- `longjmp(<env>)` failed to restore callee-saved vector registers from `<env>`.
- `setjmp(<env>)` failed to save callee-saved vector registers to `<env>`.

This has been fixed.

SDCOMP-62756

The Arm C library implementation of the `memmove()` function for AArch64 state could incorrectly corrupt memory when copying at least 4GB. This has been fixed.

SDCOMP-60938

The Arm C library implementations of the `lround()`, `lroundf()`, `llround()`, and `llroundf()` functions could incorrectly fail to set `errno` to `EDOM`. This has been fixed.

SDCOMP-60700

The Arm C library implementation of the `fwrite()` function incorrectly always returned zero when an error occurs instead of returning the number of objects successfully written. This has been fixed.

SDCOMP-60430

The Arm C++ library variant built with C++ exception handling could allocate memory at a fallback address that is incorrectly not 8-byte aligned. This has been fixed.

SDCOMP-60260

The Arm C library implementations of the `fma()` and `fmaf()` functions for AArch64 state could incorrectly set `errno` to `ERANGE` when the result is exactly equal to zero. This has been fixed.

SDCOMP-59437

Microlib incorrectly did not contain the following `__fp16` conversion functions:

- `__aeabi_d2h()`
- `__aeabi_f2h()`
- `__aeabi_h2f()`

Subsequently, this could result in the linker correctly reporting the following error:

- `L6218E: Undefined symbol <symbol> (referred from <object>)`

This has been fixed. Microlib now contains these conversion functions.

SDCOMP-57673

The variant of the Arm C library for AArch64 state incorrectly failed to configure the floating-point unit as follows:

- Enable Default NaN propagation.
- Enable flushing denormalized numbers to zero.

This has been fixed.

SDCOMP-53184

The Arm C library implementation of the `fclose()` function could incorrectly fail to disassociate the stream it closes when the `_sys_close()` function returns a non-zero value. This has been fixed.

3.2.6 Legacy assembler, armasm

This section contains a list of defect fixes made in the deprecated legacy assembler, `armasm`.

SDCOMP-64268

When using the toolchain with a User-based License (UBL), the tools could incorrectly report one of the following errors:

- `License capability data invalid`
- `Your cached license information has expired and could not be refreshed`

This has been fixed.

SDCOMP-62245

Previously, the toolchain could not be used with a legacy FlexNet Publisher (FNP) license for DS-5 Ultimate without applying the workaround documented at https://developer.arm.com/documentation/ka005517/latest. This has been fixed.

SDCOMP-60872

When using the toolchain with a User-based License (UBL), the tools incorrectly reported `Component: Arm Compiler for Embedded <version>` in the output of the `--vsn` command-line option. This has been fixed. The tools now report `Component: Arm Compiler for Embedded FuSa <version>`.

4. Known issues

This chapter contains a discretionary list of known issues in Arm Compiler for Embedded FuSa 6.16.3.

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

SDCOMP-69287

The compiler and integrated assembler incorrectly classify the Neoverse N2 processor as an Armv8.5-A target instead of an Armv9-A target. Armv9-A is not included in the list of Arm architectures which can be built for using Arm Compiler for Embedded FuSa 6.16LTS.

If you need to build for an Armv9-A target such as Neoverse N2, consider upgrading to Arm Compiler for Embedded FuSa 6.22LTS.

5. Deprecated or removed features

This chapter contains a discretionary list of features that have been deprecated or removed in Arm Compiler for Embedded FuSa 6.16.3.

SDCOMP-69275

Support has been removed for the `fromelf --emit=got` option to print the contents of the Global Offset Table (GOT) section.

SDCOMP-61019

Support has been removed for the `fromelf --interleave=source` option. To view the mapping between disassembly and source code, compile with `-g` and use an external debugger tool such as Arm Development Studio.

6. Support

This chapter includes guidance on how to obtain support for using Arm Compiler for Embedded FuSa 6.16.3.

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:

- If you are using the toolchain with a User-based License (UBL), the output from running `armlm inspect`.
- `--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.

7. Release history

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

Version	Release Date
Arm Compiler for Embedded FuSa 6.16.1	15 Oct 2021
Arm Compiler for Embedded FuSa 6.16.2	3 May 2022
Arm Compiler for Embedded FuSa 6.16.3	29 Jan 2026

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.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

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
061603-01	29 January 2026	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 FuSa in the release history of Arm Compiler for Embedded FuSa, refer to the *Release Notes* for each release.

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.

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.

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 Certified C Library, Arm Certified C++ Library, and Arm Compiler for Embedded FuSa compatibility	KA006434	Non-Confidential
Arm Compiler for Embedded FuSa 6.16.3 User Documentation and Qualification Kit Addendum	111217	Non-Confidential
Arm Compiler for Embedded FuSa 6.16LTS Defect Notification Report	107987	Non-Confidential
Arm Compiler for Embedded FuSa 6.16LTS Qualification Kit Safety Manual version 6.16.2	102288	Confidential
Arm Compiler for Embedded FuSa 6.16LTS documentation index	KA005062	Non-Confidential
Arm Compiler for Embedded FuSa Migration and Compatibility Guide version 6.16.2	102285	Non-Confidential
Arm Compiler for Embedded FuSa Reference Guide version 6.16.2LTS	102284	Non-Confidential
Arm Compiler for Embedded FuSa User Guide version 6.16.2	102283	Non-Confidential
Arm Development Studio	–	Non-Confidential
Arm Development Studio Getting Started Guide	101469	Non-Confidential
Arm Keil MDK v5 Release Notes	107778	Non-Confidential
Arm Keil MDK v6	–	Non-Confidential
Arm Keil Studio Visual Studio Code Extensions User Guide	–	Non-Confidential
Does Arm document all known issues that affect each Arm Compiler release?	KA005052	Non-Confidential
Product and toolkit configuration for FlexNet Publisher (FNP) licenses	KA004977	Non-Confidential
User-based Licensing User Guide	102516	Non-Confidential
Which host architectures, host operating systems, and host environments can I use with functional safety compiler toolchains?	KA005718	Non-Confidential

Arm product resources	Document ID	Confidentiality
µVision User's Guide	101407	Non-Confidential