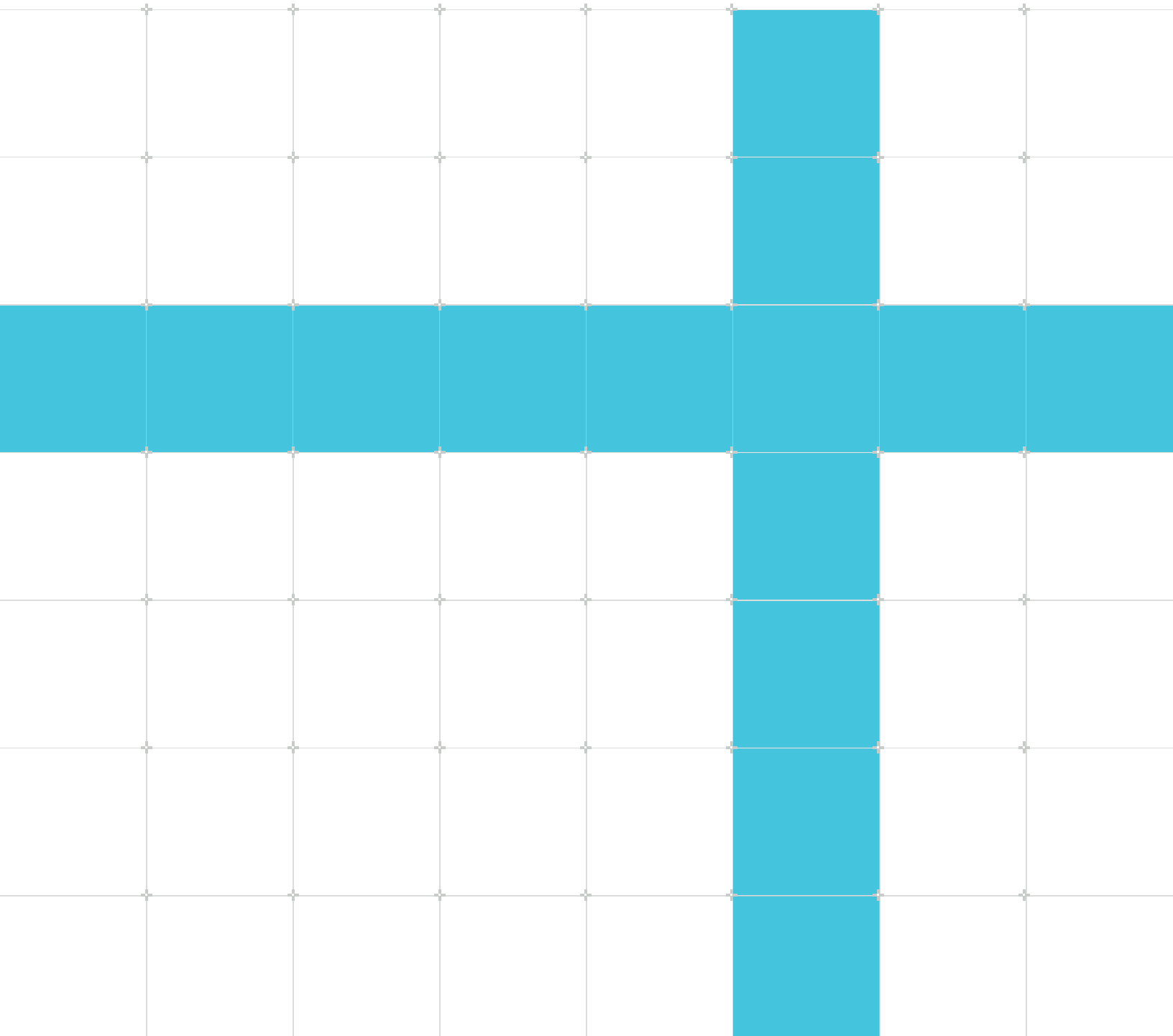
Arm Certified C Library Defect Notification Report

Version December 2025

Non-Confidential

Issue 01
111135_2025-12_01_en



Arm Certified C Library Defect Notification Report

This document (111135_2025-12-01_en) was issued on 2025-12-17. There might be a later issue at https://developer.arm.com/documentation/111135

The product version is December 2025.

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 has a valid license for the Arm Certified C Library, and is using a compatible Arm Compiler for Embedded FuSa release with the libraries to build a project with functional safety or long-term maintenance requirements. The document includes descriptions of known safety-related defects that affect each release of the Arm Certified C Library.

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. Introduction

This document is intended for functional safety managers and software developers using Arm Certified C Library for functional safety projects.


Note

Arm Certified C Library is also known as:

- Arm Compiler 6.6 C Library for Functional Safety
- Arm FuSa C Library

This document has been created based on information available to Arm as of 17 December 2025. It provides an updated list of known safety-related defects that affect a release of Arm Certified C Library, and has been published on a discretionary basis.

Functional safety managers can reference the known defects in Arm Certified C Library to address requirement 11.4.4 in ISO 26262-8, *Planning of usage of a software tool*, and the equivalent requirement in IEC 61508-4 section 7.4.4.5.

Software developers can study the known defect list and apply appropriate safeguards and workarounds if they think they are at risk.

For information on the referenced documents, see the *Safety Manual* for the Arm Certified C Library release you are using.

1.1 Scope of the Defect Lists

The defect lists within this document contain an entry for each known defect that is in a safety-related fault category and, at the time this document was generated, identified as affecting the following Arm Certified C Library releases: 6.6.A and 6.6.B.

See *The role of Arm Compiler for Embedded FuSa in Safety-related Development* section in the *Arm Compiler for Embedded FuSa Qualification Kit Safety Manual* for the Arm Compiler for Embedded FuSa release you are using with Arm Certified C Library for an explanation of the safety-related fault categories.

Defects are grouped according to the fault category, and are listed in descending order of the defect identifier number.

1.2 Derivation of the Defect Lists

This section describes how the information in the defect lists within this document are derived.

The information in the defect lists in this document is derived directly from the Arm defect tracking system.

The provided information might change in future versions of this document. Such changes can include the removal of a defect from the document.

1.3 Documentation releases for documentation synchronization faults

This section explains the relationship between documentation releases and toolchain releases in the context of Documentation synchronization faults.

Documentation synchronization faults apply to specific releases of the documentation. Such faults do not indicate a fault in the libraries.

For example, if a documentation synchronization fault affects release 6.6.B of Arm Certified C Library, the fault applies to the documentation associated with Arm Certified C Library version 6.6.B.

2. Defects

This chapter contains information about all known safety-related defects that affect releases of Arm Certified C Library available as of 17 December 2025.

2.1 Format of a Defect Entry

This section describes the format of a defect entry in this document.

Each defect entry contains the following information:

Item	Description
Defect identifier	A unique identifier for the defect, of the form `SDCOMP-<N>`. This identifier is used as the title of the section describing the defect. It should be used in all communication regarding the defect.
Fault category	Each defect in this documented is listed in a section based on its safety-related fault category classification: • Translation fault • Missing diagnostic fault • Determinism fault • Documentation synchronization fault For more information about fault categories, see the *Arm Compiler for Embedded FuSa Qualification Kit Safety Manual* for the release of Arm Compiler for Embedded FuSa you are using with Arm Certified C Library.
Target environment	Where feasible, describes the set of target Arm architectures or processor states that might be affected by the defect. The default value is "Any", which means the issue could affect any supported target Arm architecture or processor state. For more information, see the Target environment section.
Affected releases	A list of the releases in scope that the defect is observable in.
Unaffected releases	A list of the releases in scope that the defect is not observable in.
Description	A summary of the defect and its impact.
Conditions	A list of conditions that must hold to observe the defect.

The information describing the scope of a defect is included in a table in each defect entry in this document. You can use the information in this table to determine if a defect is relevant to your project without having to read the full details of the defect.

To avoid a known defect, manually inspect the source code and command-line options to ensure that at least one condition for the defect does not hold true. Arm Support might be able to help you identify other workarounds for known defects, if a generic workaround is not suitable.

2.1.1 Target environment

This section describes the purpose and meaning of the target environment associated with each defect entry included in this document.

Where feasible, the target environment field is used to limit the scope of a defect to a specific execution context.

The target environment field can specify one or more of the following:

- Architectures (for example, Armv8-M)
- Processor states (for example, AArch32 state)
- A combination of architecture and state
- The value "Any", which indicates no restriction on architecture or processor state

The target environment field does not specify any of the following:

- Architecture revisions. For example, Armv8.1-M.
- Architecture extensions. For example, the M-profile Vector Extension (MVE).

These details, if relevant, are included in the Conditions section of the defect entry. For example, a defect listed with the target environment "Armv8-M with the Main Extension" may include a condition such as the following:

- The program is compiled for a target with the M-profile Vector Extension (MVE).

Architectures and processor states are not mutually exclusive, and their relationships should be interpreted inclusively. The following hierarchy illustrates how processor states and architectures relate:

```
Any
├── AArch32 state
│   ├── A32 state
│   │   ├── Armv7-A
│   │   ├── Armv8-A
│   │   ├── Armv7-R
│   │   └── Armv8-R
│   └── T32 state
│       ├── Armv7-A
│       ├── Armv8-A
│       ├── Armv7-R
│       ├── Armv8-R
│       ├── Armv6-M
│       ├── Armv7-M
│       └── Armv8-M
│           ├── Armv8-M with the Main Extension
│           └── Armv8-M without the Main Extension
└── AArch64 state
    └── Armv8-A
```

A defect listed under a processor state, such as "T32 state", can apply to any processor or architecture that supports that state. Similarly, a defect listed under an architecture, such as "Armv8-A", can apply to any of the processor states that architecture supports, unless limited further by the Conditions section of the defect entry.

Therefore, when determining if a defect is relevant to your project, both the Target Environment field and the Conditions section must be reviewed together. You can use the hierarchy above and the guidance below to identify all applicable environments for your project.

For example:

- If your project is built for a Cortex-M3 target, you might need to look at defects that affect any of the following target environments:
  - Any
  - AArch32 state
  - T32 state
  - Armv7-M
- If your project is built for a Cortex-A53 target and AArch64 state, you might need to look at defects that affect any of the following target environments:
  - Any
  - AArch64 state
  - Armv8-A

Tip

This document includes an appendix that provides an index of known issues grouped by target environment, which can be used for this purpose.

The following target environments are included within the scope of this document:

Target environment	Description
Any	The scope defect is not limited to any specific target environments, and can affect any target Arm architecture or processor subject only to the conditions under which the defect can occur.
A32 state	An Arm architecture or processor in A32 state (formerly Arm state). Depending on the `-mcpu` or `-march` option used with the compiler, A32 state may be the default. For example, it is the default when compiling with `-mcpu=cortex-r52`. For more information, see the *-march* and *-mcpu* sections of the Arm Compiler for Embedded FuSa *Reference Guide*.
AArch32 state	An Arm architecture or processor in AArch32 state. This includes A32 state (formerly Arm state) and T32 state (formerly Thumb state).
AArch64 state	An Arm architecture or processor in AArch64 state.
Armv6-M	The Armv6-M architecture, or a processor based on the Armv6-M architecture. For example, Cortex-M0.
Armv7-A	The Armv7-A architecture, or a processor based on the Armv7-A architecture. For example, Cortex-A9.
Armv7-M	The Armv7-M architecture, or a processor based on the Armv7-M architecture. For example, Cortex-M3.
Armv7-R	The Armv7-R architecture, or a processor based on the Armv7-R architecture. For example, Cortex-R5.
Armv8-A	Any version of the Armv8-A architecture, or a processor based on any version of the Armv8-A architecture. Unless otherwise specified in the conditions of a defect, this includes both AArch64 state and AArch32 state. For example, Cortex-A53.

Target environment	Description
Armv8-M	Any version of the Armv8-M architecture, or a processor based on any version of the Armv8-M architecture. Unless otherwise specified in the conditions of a defect, this includes both Armv8-M with the Main Extension and Armv8-M without the Main Extension. For example, this includes projects that are compiled with `-march=armv8-m.base`, `-march=armv8.1-m.main`, or `-mcpu=cortex-m55`.
Armv8-M with the Main Extension	Any version of the Armv8-M architecture with the Main Extension, or a processor based on any version of the Armv8-M architecture with the Main Extension. For example, this includes projects that are compiled with `-march=armv8.1-m.main` or `-mcpu=cortex-m33`.
Armv8-M without the Main Extension	Any version of the Armv8-M architecture without the Main Extension, or a processor based on any version of the Armv8-M architecture without the Main Extension. For example, this includes projects that are compiled with `-march=armv8-m.base` or `-mcpu=cortex-m23`.
Armv8-R	The Armv8-R architecture, or a processor based on the Armv8-R architecture. This does not include the Armv8-R AArch64 architecture. For example, Cortex-R52.
T32 state	An Arm architecture or processor in T32 state (formerly Thumb state). For example, this always applies when compiling for an M-profile target.

2.2 Machine-readable defects list

This section provides information about the JSON format defect lists included as an attachment with this document.

The contents of the defects lists in this document are available in a machine-readable JSON format. The file `defects_as_JSON.zip` bundled with this document contains the following files that can be used to programmatically analyze the defects listed within this document:

`defects.json`

A JSON file containing a list of all defects from this document, and information about the scope of the list. The entry for each defect follows the same format as described in Format of a Defect Entry. The defect description and conditions are provided as HTML markup.

`schema.json`

The JSON schema for the file `defects.json`. It includes descriptions of the contents of the `defects.json` file.

Arm does not provide tools to analyze the JSON format defects list.

2.3 Defects

This section contains details about known safety-related defects that affect releases of Arm Certified C Library available as of 17 December 2025.

The following defects are included in this section:

Identifier	Fault category
SDCOMP-66904	Translation fault
SDCOMP-65007	Translation fault
SDCOMP-64835	Translation fault
SDCOMP-58778	Translation fault
SDCOMP-58695	Translation fault
SDCOMP-53652	Translation fault
SDCOMP-53576	Translation fault
SDCOMP-53422	Translation fault
SDCOMP-56303	Documentation synchronization fault

2.3.1 Translation faults

This section contains details about safety-related defects that have been classified as a translation fault.

For more information about the definition of a translation fault, see the *Arm Compiler for Embedded FuSa Qualification Kit Safety Manual* for the Arm Compiler for Embedded FuSa release you are using with Arm Certified C Library.

2.3.1.1 SDCOMP-66904

This section describes the scope of the translation fault defect with the unique identifier SDCOMP-66904.

The following table describes the scope of this defect:

Target environment	Affected releases	Unaffected releases
Any	6.6.A, 6.6.B	-

Description

The `snprintf()` function can process the `%n` conversion specifier incorrectly.

For example, the `snprintf()` function incorrectly sets `n` to `5` instead of `0` for the following:

```
char buf[16];
int n;
snprintf(buf, 0, "12345%n\n", &n);
```

Conditions

The safety-related system is at risk when all the following are true:

- The program is linked with the Certified C Library.
- The program contains a call `z` to the `snprintf()` function declared in the `<stdio.h>` header file.

- `Z` specifies `X` as the maximum number of characters to output.
- `Z` specifies a format string `F`.
- `F` contains a `%n` format specifier after `Y` characters within `F`.
- `X < Y`.

2.3.1.2 SDCOMP-65007

This section describes the scope of the translation fault defect with the unique identifier SDCOMP-65007.

The following table describes the scope of this defect:

Target environment	Affected releases	Unaffected releases
Any	6.6.A, 6.6.B	-

Description

The `strnlen()` function can perform an out-of-bounds memory access by reading 1 character past the end of the string.

Conditions

The safety-related system is at risk when all the following are true:

- The program is linked with the Certified C Library.
- The program contains a call `Z` to the `strnlen()` function declared in the `<string.h>` header file.
- The behavior of the program depends on `Z` not performing an out-of-bounds memory access during the execution of `F`.

2.3.1.3 SDCOMP-64835

This section describes the scope of the translation fault defect with the unique identifier SDCOMP-64835.

The following table describes the scope of this defect:

Target environment	Affected releases	Unaffected releases
AArch64 state	6.6.A, 6.6.B	-

Description

The Certified C Library implementation of the `qsort()` function can corrupt the stack when sorting an array larger than 4GB. Subsequently, this can result in unexpected run-time behavior.

Conditions

The safety-related system is at risk when all the following are true:

- The program is linked with the Certified C Library.
- The program contains a call `Z` to the `qsort()` function declared in the `<stdlib.h>` header file.
- `Z` specifies an array containing `M` members of size `N` each.
- `M * N` is larger than 4GB.

2.3.1.4 SDCOMP-58778

This section describes the scope of the translation fault defect with the unique identifier SDCOMP-58778.

The following table describes the scope of this defect:

Target environment	Affected releases	Unaffected releases
Any	6.6.A, 6.6.B	-

Description

The Certified C Library implementations of the `cos()`, `sin()`, and `tan()` functions can perform an out-of-bounds memory access.

Conditions

The safety-related system is at risk when all the following are true:

- The program is linked with the Certified C Library.
- The program contains a call `Z` to a function `F`.
- `F` is one of the following functions declared in the `<math.h>` header file:
  - `cos()`
  - `sin()`
  - `tan()`
- The argument to `Z` is a finite floating-point number with an absolute value greater than or equal to `2**1014`.
- The behavior of the program depends on `Z` not performing an out-of-bounds memory access during the execution of `F`.

2.3.1.5 SDCOMP-58695

This section describes the scope of the translation fault defect with the unique identifier SDCOMP-58695.

The following table describes the scope of this defect:

Target environment	Affected releases	Unaffected releases
Any	6.6.A, 6.6.B	-

Description

The `snprintf()` function can incorrectly write an empty string to the buffer but still return a non-negative result.

Conditions

The safety-related system is at risk when all the following are true:

- The program is linked with the Certified C Library.
- The program contains a call `Z` to the `snprintf()` function declared in the `<stdio.h>` header file.
- The first argument to `Z` specifies a pointer `P` to a buffer `B`.
- The second argument to `Z` specifies a buffer size `S`.
- Incrementing `P` by `S-1` causes an integer overflow.
- The behavior of the program depends on `Z` writing the intended output string to `B`.

2.3.1.6 SDCOMP-53652

This section describes the scope of the translation fault defect with the unique identifier SDCOMP-53652.

The following table describes the scope of this defect:

Target environment	Affected releases	Unaffected releases
AArch32 state	6.6.A, 6.6.B	-

Description

The `__scatterload_rt2()` function can initialize an execution region within a position-independent (PI) load region incorrectly.

Conditions

The safety-related system is at risk when all the following are true:

- The program is linked with a variant `V` of the Certified C Library.
- `V` has one of the following forms:
  - `*_ropi.b`
  - `*_ropi.l`
  - `*_ropi_rwpi.b`
  - `*_ropi_rwpi.l`
- The program contains a call `Z` to the `__scatterload_rt2()` function.
- The program is linked with a scatter file `F`.
- `F` contains a position-independent load region `L`.
- The behavior of the program depends on `Z` correctly initializing all execution regions with `L`.

2.3.1.7 SDCOMP-53576

This section describes the scope of the translation fault defect with the unique identifier SDCOMP-53576.

The following table describes the scope of this defect:

Target environment	Affected releases	Unaffected releases
AArch64 state	6.6.A, 6.6.B	-

Description

The `snprintf()` function can write an incorrectly formatted value to the output buffer.

Conditions

The safety-related system is at risk when all the following are true:

- The program is linked with the Certified C Library.
- The program contains a call `z` to the `snprintf()` function declared in the `<stdio.h>` header file.
- `z` specifies at least 6 variadic arguments.
- A sixth or later variadic argument to `z` is processed using one of the following format specifiers:
  - `%d`
  - `%i`
  - `%u`

2.3.1.8 SDCOMP-53422

This section describes the scope of the translation fault defect with the unique identifier SDCOMP-53422.

The following table describes the scope of this defect:

Target environment	Affected releases	Unaffected releases
Any	6.6.A, 6.6.B	-

Description

The Certified C Library implementation of the `pow()` function can incorrectly fail to set `errno` to `ERANGE` when the return value overflows to `HUGE_VAL`.

Conditions

The safety-related system is at risk when all the following are true:

- The program is linked with the Certified C Library.
- The program contains a call `z` to the `pow()` function declared in the `<math.h>` header file.

- `z` has arguments that result in an overflow to `HUGE_VAL`.
- The behavior of the program depends on `z` setting `errno` to `ERANGE`.

2.3.2 Missing diagnostic faults

There are no known missing diagnostic faults.

For more information about the definition of a missing diagnostic fault, see the *Arm Compiler for Embedded FuSa Qualification Kit Safety Manual* for the Arm Compiler for Embedded FuSa release you are using with Arm Certified C Library.

2.3.3 Determinism faults

There are no known determinism faults.

For more information about the definition of a determinism fault, see the *Arm Compiler for Embedded FuSa Qualification Kit Safety Manual* for the Arm Compiler for Embedded FuSa release you are using with Arm Certified C Library.

2.3.4 Documentation synchronization faults

This section contains details about safety-related defects that have been classified as a documentation synchronization fault.

For more information about the definition of a documentation synchronization fault, see the *Arm Compiler for Embedded FuSa Qualification Kit Safety Manual* for the Arm Compiler for Embedded FuSa release you are using with Arm Certified C Library.

2.3.4.1 SDCOMP-56303

This section describes the scope of the documentation synchronization fault defect with the unique identifier SDCOMP-56303.

The following table describes the scope of this defect:

Target environment	Affected releases	Unaffected releases
Any	6.6.A, 6.6.B	-

Description

The *General recommendations* table in the *User Requirements* section of the Certified C Library *Safety Manual* incorrectly does not state the `strtok` function in the lists of functions for `FCL_GR_MT_1`, and `FCL_GR_MT_3`.

Conditions

The safety-related system is at risk when all the following are true:

- The program is linked with the Certified C Library.
- The program contains a call `z` to the `strtok()` function defined in the `<string.h>` header file supplied with the Certified C Library.
- `z` does not conform to one of the following user requirements:
  - `FCL_GR_MT_1`
  - `FCL_GR_MT_3`

Appendix A Changes since the Arm Certified C Library version 6.6.B Safety Manual

This appendix provides information about changes made to the defect lists compared to the *Arm Certified C Library version 6.6.B Safety Manual*.

A.1 Defects added

This section contains a list of defects that have been added to this document compared to the *Arm Certified C Library version 6.6.B Safety Manual*.

Identifier	Fault category	Target environment
SDCOMP-66904	Translation fault	Any
SDCOMP-65007	Translation fault	Any
SDCOMP-56303	Documentation synchronization fault	Any
SDCOMP-53652	Translation fault	AArch32 state

A.2 Defects updated

This section contains a list of defects that have been updated in this document compared to the *Arm Certified C Library version 6.6.B Safety Manual*.

Identifier	Fault category	Target environment
SDCOMP-64835	Translation fault	AArch64 state
SDCOMP-58778	Translation fault	Any
SDCOMP-58695	Translation fault	Any
SDCOMP-53576	Translation fault	AArch64 state
SDCOMP-53422	Translation fault	Any

Appendix B Defect index

This appendix provides multiple indexes for the known defects included in this document.

B.1 Indexed by affected version

This section provides an index of known defects grouped by the affected version of Arm Certified C Library.

B.1.1 Defects that affect Arm Certified C Library version 6.6.B

There are 9 known defects that affect Arm Certified C Library version 6.6.B.

Identifier	Fault category	Target environment
SDCOMP-66904	Translation fault	Any
SDCOMP-65007	Translation fault	Any
SDCOMP-64835	Translation fault	AArch64 state
SDCOMP-58778	Translation fault	Any
SDCOMP-58695	Translation fault	Any
SDCOMP-56303	Documentation synchronization fault	Any
SDCOMP-53652	Translation fault	AArch32 state
SDCOMP-53576	Translation fault	AArch64 state
SDCOMP-53422	Translation fault	Any

B.1.2 Defects that affect Arm Certified C Library version 6.6.A

There are 9 known defects that affect Arm Certified C Library version 6.6.A.

Identifier	Fault category	Target environment
SDCOMP-66904	Translation fault	Any
SDCOMP-65007	Translation fault	Any
SDCOMP-64835	Translation fault	AArch64 state
SDCOMP-58778	Translation fault	Any
SDCOMP-58695	Translation fault	Any
SDCOMP-56303	Documentation synchronization fault	Any
SDCOMP-53652	Translation fault	AArch32 state
SDCOMP-53576	Translation fault	AArch64 state
SDCOMP-53422	Translation fault	Any

B.2 Indexed by target environment

This section provides an index of known defects grouped by the affected target environment.

A defect listed under one target environment might also be relevant to others. For example, a defect listed under "Armv8-A" might also be relevant to "AArch64 state", "AArch32 state", "A32 state", or "T32 state".

You must use this index in conjunction with the Target environment section, which provides a hierarchy and description of how target environments relate to each other.

B.2.1 Defects that affect all target environments

There are 6 known defects that affect all target environments.

Identifier	Fault category	Affected releases
SDCOMP-66904	Translation fault	6.6.A, 6.6.B
SDCOMP-65007	Translation fault	6.6.A, 6.6.B
SDCOMP-58778	Translation fault	6.6.A, 6.6.B
SDCOMP-58695	Translation fault	6.6.A, 6.6.B
SDCOMP-56303	Documentation synchronization fault	6.6.A, 6.6.B
SDCOMP-53422	Translation fault	6.6.A, 6.6.B

B.2.2 Defects with the target environment A32 state

There are no known defects with the target environment A32 state.


Note

If you are reviewing this section, you might also need to review the defects listed in the following related target environment indexes:

- B.2.1 Defects that affect all target environments on page 20
- B.2.3 Defects with the target environment AArch32 state on page 20
- B.2.6 Defects with the target environment Armv7-A on page 22
- B.2.8 Defects with the target environment Armv7-R on page 22
- B.2.9 Defects with the target environment Armv8-A on page 23
- B.2.13 Defects with the target environment Armv8-R on page 24

B.2.3 Defects with the target environment AArch32 state

There is 1 known defect with the target environment AArch32 state.

Note

If you are reviewing the defects listed this section, you might also need to review the defects listed in the following related target environment indexes:

- B.2.1 Defects that affect all target environments on page 20
- B.2.2 Defects with the target environment A32 state on page 20
- B.2.5 Defects with the target environment Armv6-M on page 21
- B.2.6 Defects with the target environment Armv7-A on page 22
- B.2.7 Defects with the target environment Armv7-M on page 22
- B.2.8 Defects with the target environment Armv7-R on page 22
- B.2.9 Defects with the target environment Armv8-A on page 23
- B.2.10 Defects with the target environment Armv8-M on page 23
- B.2.11 Defects with the target environment Armv8-M with the Main Extension on page 23
- B.2.12 Defects with the target environment Armv8-M without the Main Extension on page 24
- B.2.13 Defects with the target environment Armv8-R on page 24
- B.2.14 Defects with the target environment T32 state on page 24

Identifier	Fault category	Affected releases
SDCOMP-53652	Translation fault	6.6.A, 6.6.B

B.2.4 Defects with the target environment AArch64 state

There are 2 known defects with the target environment AArch64 state.

Note

If you are reviewing the defects listed this section, you might also need to review the defects listed in the following related target environment indexes:

- B.2.1 Defects that affect all target environments on page 20
- B.2.9 Defects with the target environment Armv8-A on page 23

Identifier	Fault category	Affected releases
SDCOMP-64835	Translation fault	6.6.A, 6.6.B
SDCOMP-53576	Translation fault	6.6.A, 6.6.B

B.2.5 Defects with the target environment Armv6-M

There are no known defects with the target environment Armv6-M.

Note

If you are reviewing this section, you might also need to review the defects listed in the following related target environment indexes:

- B.2.1 Defects that affect all target environments on page 20
- B.2.3 Defects with the target environment AArch32 state on page 20
- B.2.14 Defects with the target environment T32 state on page 24

B.2.6 Defects with the target environment Armv7-A

There are no known defects with the target environment Armv7-A.

Note

If you are reviewing this section, you might also need to review the defects listed in the following related target environment indexes:

- B.2.1 Defects that affect all target environments on page 20
- B.2.2 Defects with the target environment A32 state on page 20
- B.2.3 Defects with the target environment AArch32 state on page 20
- B.2.14 Defects with the target environment T32 state on page 24

B.2.7 Defects with the target environment Armv7-M

There are no known defects with the target environment Armv7-M.

Note

If you are reviewing this section, you might also need to review the defects listed in the following related target environment indexes:

- B.2.1 Defects that affect all target environments on page 20
- B.2.3 Defects with the target environment AArch32 state on page 20
- B.2.14 Defects with the target environment T32 state on page 24

B.2.8 Defects with the target environment Armv7-R

There are no known defects with the target environment Armv7-R.

Note

If you are reviewing this section, you might also need to review the defects listed in the following related target environment indexes:

- B.2.1 Defects that affect all target environments on page 20
- B.2.2 Defects with the target environment A32 state on page 20
- B.2.3 Defects with the target environment AArch32 state on page 20
- B.2.14 Defects with the target environment T32 state on page 24

B.2.9 Defects with the target environment Armv8-A

There are no known defects with the target environment Armv8-A.

Note

If you are reviewing this section, you might also need to review the defects listed in the following related target environment indexes:

- B.2.1 Defects that affect all target environments on page 20
- B.2.2 Defects with the target environment A32 state on page 20
- B.2.3 Defects with the target environment AArch32 state on page 20
- B.2.4 Defects with the target environment AArch64 state on page 21
- B.2.14 Defects with the target environment T32 state on page 24

B.2.10 Defects with the target environment Armv8-M

There are no known defects with the target environment Armv8-M.

Note

If you are reviewing this section, you might also need to review the defects listed in the following related target environment indexes:

- B.2.1 Defects that affect all target environments on page 20
- B.2.3 Defects with the target environment AArch32 state on page 20
- B.2.14 Defects with the target environment T32 state on page 24

B.2.11 Defects with the target environment Armv8-M with the Main Extension

There are no known defects with the target environment Armv8-M with the Main Extension.

Note

If you are reviewing this section, you might also need to review the defects listed in the following related target environment indexes:

- B.2.1 Defects that affect all target environments on page 20
- B.2.3 Defects with the target environment AArch32 state on page 20
- B.2.10 Defects with the target environment Armv8-M on page 23
- B.2.14 Defects with the target environment T32 state on page 24

B.2.12 Defects with the target environment Armv8-M without the Main Extension

There are no known defects with the target environment Armv8-M without the Main Extension.

Note

If you are reviewing this section, you might also need to review the defects listed in the following related target environment indexes:

- B.2.1 Defects that affect all target environments on page 20
- B.2.3 Defects with the target environment AArch32 state on page 20
- B.2.10 Defects with the target environment Armv8-M on page 23
- B.2.14 Defects with the target environment T32 state on page 24

B.2.13 Defects with the target environment Armv8-R

There are no known defects with the target environment Armv8-R.

Note

If you are reviewing this section, you might also need to review the defects listed in the following related target environment indexes:

- B.2.1 Defects that affect all target environments on page 20
- B.2.2 Defects with the target environment A32 state on page 20
- B.2.3 Defects with the target environment AArch32 state on page 20
- B.2.14 Defects with the target environment T32 state on page 24

B.2.14 Defects with the target environment T32 state

There are no known defects with the target environment T32 state.


Note

If you are reviewing this section, you might also need to review the defects listed in the following related target environment indexes:

- B.2.1 Defects that affect all target environments on page 20
- B.2.3 Defects with the target environment AArch32 state on page 20
- B.2.5 Defects with the target environment Armv6-M on page 21
- B.2.6 Defects with the target environment Armv7-A on page 22
- B.2.7 Defects with the target environment Armv7-M on page 22
- B.2.8 Defects with the target environment Armv7-R on page 22
- B.2.9 Defects with the target environment Armv8-A on page 23
- B.2.10 Defects with the target environment Armv8-M on page 23
- B.2.11 Defects with the target environment Armv8-M with the Main Extension on page 23
- B.2.12 Defects with the target environment Armv8-M without the Main Extension on page 24
- B.2.13 Defects with the target environment Armv8-R on page 24

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
202512-01	17 December 2025	Non-Confidential	Initial release

Change history

For a list of technical changes since the last release listed in the release history of this document, see Changes since the Arm Certified C Library version 6.6.B Safety Manual.

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**.

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

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.

Tip

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

Remember

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

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 following tables provides direct access to the online version of the document.

Arm product resources	Document ID	Confidentiality
Arm Certified C Library version 6.6.A Safety Manual	101498	Confidential
Arm Certified C Library version 6.6.B Safety Manual	101611	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	102288	Confidential
Arm Compiler for Embedded FuSa 6.16LTS documentation index	KA005062	Non-Confidential
Arm Compiler for Embedded FuSa 6.22LTS Defect Notification Report	110099	Non-Confidential
Arm Compiler for Embedded FuSa 6.22LTS Qualification Kit Safety Manual	109409	Confidential
Arm Compiler for Embedded FuSa 6.22LTS documentation index	KA006002	Non-Confidential
Arm Compiler for Functional Safety 6.6 Qualification Kit Safety Manual	100751	Confidential
Arm Compiler for Functional Safety 6.6 documentation index	KA005063	Non-Confidential
Does Arm document all known issues that affect each Arm Compiler release?	KA005052	Non-Confidential