MISRA C:2012 Amendment 2

Updates for ISO/IEC 9899:2011
Core functionality

February 2020

First published February 2020 by HORIBA MIRA Limited
Watling Street
Nuneaton
Warwickshire
CV10 0TU
UK

www.misra.org.uk

ISBN 978-1-906400-25-5 PDF

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

MISRA C:2012 Amendment 2

Updates for ISO/IEC 9899:2011
Core functionality

February 2020

MISRA Mission Statement

We provide world-leading, best practice guidelines for the safe and secure application of both embedded control systems and standalone software.

MISRA is a collaboration between manufacturers, component suppliers and engineering consultancies which seeks to promote best practice in developing safety- and security-related electronic systems and other software-intensive applications. To this end, MISRA publishes documents that provide accessible information for engineers and management, and holds events to permit the exchange of experiences between practitioners.

Foreword

An updated edition of the C Standard, commonly referred to as "C11", was released just as MISRA C:2012 was being prepared for publication, meaning it arrived too late for the MISRA C Working Group to take it into consideration.

As the adoption of C11 has become more widespread, the MISRA C Working Group have decided that it is now time to address this new edition of the C Standard, support for which will be implemented by means of a series of amendments to MISRA C:2012.

This document amends MISRA C:2012 as required to introduce support for ISO/IEC 9899:2011. Subsequent amendments will be used to introduce specific guidance for the features introduced by ISO/IEC 9899:2011.

In the three years since its publication, the additional guidance provided by MISRA Compliance has been increasingly adopted. The MISRA C Working Group have decided that now is the time to upgrade this guidance from optional to be an integral part of the MISRA C lifecycle. Therefore, this document amends MISRA C:2012 to do that.

We trust that this amendment will be welcomed by the community at large, and will offer confidence to projects and organizations who have held off migrating to C11.

Andrew Banks FBCS CITP
Chairman, MISRA C Working Group

Acknowledgements

The MISRA consortium would like to thank the following individuals for their significant contribution to the writing of this document:

Fulvio Baccaglini	Perforce
Andrew Banks	LDRA Ltd.
Jill Britton	Perforce
Chris Tapp	LDRA Ltd. and Keylevel Consultants Ltd.
Liz Whiting	LDRA Ltd.

The MISRA consortium also wishes to acknowledge contributions from the following members of the MISRA C Working Group during the development and review process:

Roberto Bagnara	BUGSENG and University of Parma
Dave Banham	Rolls-Royce plc
Gerlinde Kettl	Vitesco Technologies GmbH
Gavin McCall	Visteon Engineering Services Ltd.

The MISRA consortium also wishes to acknowledge contributions from the following individuals during the development and review process:

David Ward	HORIBA MIRA Ltd.

DokuWiki was used extensively during the drafting of this document. Our thanks go to all those involved in its development.

This document was typeset using Open Sans. Open Sans is a trademark of Google and may be registered in certain jurisdictions. Digitized data copyright © 2010–2011, Google Corporation. Licensed under the Apache License, Version 2.0.

Contents

1 Overview

1.1 Applicability

This amendment is compatible with either:

1. MISRA C:2012 (3rd Edition, 1st Revision) [4], or
2. MISRA C:2012 (3rd Edition) [1] as corrected and amended by:
   - MISRA C:2012 Technical Corrigendum 1 [2], and
   - MISRA C:2012 Amendment 1 [3]

1.2 C language updates

This document amends MISRA C:2012 [4] to reference ISO/IEC 9899:2011 [6] with the effect that:

1. ISO/IEC 9899:2011 [6] is now supported by MISRA C:2012 [4].
2. When using ISO/IEC 9899:2011 [6], the following features may be used subject to restrictions within the Guidelines:
   - Additional floating-point characteristic macros (`float.h`)
   - Unicode characters and strings (`uchar.h`)
   - Static assertions
   - Anonymous structures and unions (note that use of unions violates Rule 19.2)
   - Support for opening files for exclusive access
   - *aligned_alloc* (note that use violates Dir 4.12 and Rule 21.3)
   - *at_quick_exit*, *quick_exit* (note that use of *quick_exit* violates Rule 21.8)
   - *timespec_get* (note that use violates Rule 21.10)
3. When using ISO/IEC 9899:2011 [6], use of the following features is prohibited without the support of a deviation against Rule 1.4:
   - Type generic expressions
   - No-return functions
   - Support for multiple threads of execution (`stdatomic.h`, `threads.h`)
   - Alignment of objects (`stdalign.h`)
   - Bounds-checking interfaces

Notes:

1. ISO/IEC 9899:2018 [7] incorporates corrigenda applicable to ISO/IEC 9899:2011 [6]. As such, it is functionally equivalent to ISO/IEC 9899:2011 [6] and is therefore also supported through this amendment.

2. References to numbered sections within the various editions of the C Standard have been removed by this amendment to make the wording applicable to future editions (where the numbering may change) without the need for additional changes.

1.3 Embodiment of MISRA Compliance

MISRA Compliance [8] (first released in 2016) introduced a more precise specification of the requirements that need to be satisfied when making a claim of compliance with MISRA C. Adoption of MISRA Compliance was optional for previous editions of MISRA C as they defined their own compliance requirements. This amendment removes the integrated compliance requirements and makes the adoption of MISRA Compliance mandatory.

1.4 Future directions

It is a long-term objective for MISRA C to provide explicit guidance for the avoidance of all instances of undefined and critically unspecified behaviour. For this reason, features that are initially permitted without the support of specific guidelines may be subject to restrictions in the future.

Subsequent amendments will introduce specific guidelines to cover the features that are prohibited by Rule 1.4, supporting their use through the introduction of guidance that must be followed to avoid any undefined or unspecified behaviour.

2 Amendments

2.1 Section 1 — The vision

AMD2.1 : Replace the whole section with:

1 Introduction

1.1 Background

The MISRA C Guidelines define a subset of the C language in which the opportunity to make mistakes is either removed or reduced. Many standards for the development of safety-related software require, or recommend, the use of a language subset, and this can also be used to develop any application with security, high integrity or high reliability requirements.

As well as defining this subset, these MISRA C Guidelines will:

- Provide educational material for those developing C programs;
- Provide reference material for tool developers.

1.2 The vision

It is a long-term objective for MISRA C to define a "predictable subset" of the C language, and to provide explicit guidance for the avoidance of all instances of undefined and unspecified behaviour. For this reason, features that are initially permitted without the support of specific guidelines may be subject to restrictions in the future.

The vision for MISRA C is to:

- Enhance the existing guidance:
  - Correct any known issues with the previous editions (where still relevant);
  - Add new guidelines for which there is a strong rationale;
  - Improve the specification and the rationale for existing guidelines, where appropriate;
  - Remove any guidelines for which the rationale is insufficient;
  - Increase the number of guidelines that can be processed by static analysis tools, by improving the decidability where possible;
- Provide guidance on the applicability of the guidelines to automatically-generated code.

1.3 Scope

The versions of the C Standard supported by this document are:

- ISO/IEC 9899:1990 [2], amended and corrected by [4], [5] and [6] — referred to as "C90"
- ISO/IEC 9899:1999 [8], corrected by [9], [10] and [11] — referred to as "C99"
- ISO/IEC 9899:2011 [13], corrected by [14] — referred to as "C11"
- ISO/IEC 9899:2018 [43] — referred to as "C18"

Notes:

1. Access to a copy of the relevant C Standard is not necessary for the use of MISRA C, but it may be helpful.
2. ISO/IEC 9899:2018 [43] does not introduce any functional changes. As such, any references to C11 are equally applicable to C18.
3. The version of the C Standard chosen may be influenced by factors such as the amount of legacy code being reused within a project and/or the availability of compilers for the target processor.

1.4 Adoption

MISRA C should be adopted at the outset of a project. In addition, the guidance given in MISRA Compliance [42] shall be followed when making a claim of compliance.

Note: If a project is building on existing code that has a proven track record, then the benefits of compliance with MISRA C may be outweighed by the risks of introducing defects when making the code compliant. In such cases, a judgement should be made between the benefits of adoption and the risks of introducing defects.

2.2 Section 3 — Tool selection

AMD2.2 : Replace the whole section with:

Withdrawn – superseded by Sections 2.6.1 and 2.6.2 of MISRA Compliance:2020 [42].

2.3 Section 4 — Prerequisite knowledge

AMD2.3 : Replace the whole section with:

Withdrawn – superseded by Sections 2.3, 2.6.4, 2.6.5 and 7.1 of MISRA Compliance:2020 [42].

2.4 Section 5 — Adopting and using MISRA C

AMD2.4 : Replace the whole section with:

Withdrawn – superseded by MISRA Compliance:2020 [42].

2.5 Section 6.2.2 — Required guidelines

AMD2.5 : Replace “Section 5.4” with “Section 4 of MISRA Compliance [42]”

2.6 Section 6.5 — Decidability of rules

AMD2.6 : In the final paragraph, replace “Section 5.3.3” with “Section 3.4 of MISRA Compliance:2020 [42]”

2.7 Section 6.9 — Presentation of guidelines

AMD2.7 : Replace the final bullet point with:

- "Cxx" is a comma separated list indicating which versions of the C Standard the guideline applies to.

AMD2.8 : Insert the following immediately before the sentence starting *Note:*

Note: Any guidelines applying to C11 also apply to C18.

2.8 Section 6.10.1 — ISO C portability issue references

AMD2.9 : Replace the first paragraph with:

The ISO C Portability Issues are described in subsections of Annexes in the C Standard. It is important to note that the numbering of the references is derived from the original standard and not from any later version that incorporates corrections and amendments.

AMD2.10 : Amend the table column headings as follows:

Reference	Annex	
	C90	C99, C11

AMD2.11 : In the first bullet point, replace "Annex G of [2]" with "Annex G (C90)".

AMD2.12 : In the second bullet point, replace "Annex J of [8]" with "Annex J (C99 and C11)".

AMD2.13 : In the first line of the fourth paragraph, replace "Annex J (C99)" with "Annex J (C99 and C11)".

2.9 Section 7 — Directives

AMD2.14 : Update the *Applies to* and *Source ref* entries for the Directives, as shown below:

Directive	Applies to	Source ref
Dir 1.1	Append ", C11"	Add ", C11 [Annex J.3]"
Dir 2.1	Append ", C11"	
Dir 3.1	Append ", C11"	
Dir 4.1	Append ", C11"	Add ", C11 [Undefined 17, 18, 36, 43, 46–48, 51, 52, 119]"
Dir 4.2	Append ", C11"	
Dir 4.3	Append ", C11"	
Dir 4.4	Append ", C11"	
Dir 4.5	Append ", C11"	
Dir 4.6	Append ", C11"	
Dir 4.7	Append ", C11"	
Dir 4.8	Append ", C11"	
Dir 4.9	Append ", C11"	
Dir 4.10	Append ", C11"	
Dir 4.11	Append ", C11"	Add ", C11 [Unspecified 30, 31, 44, 52–56; Undefined 108, 109, 113, 118, 191, 192, 194, 199, 201; Implementation J.3.12(8-14)]"
Dir 4.12	Append ", C11"	

Directive	Applies to	Source ref
Dir 4.13	Append ", C11"	
Dir 4.14	Append ", C11"	Add ", C11 [Undefined 17, 18, 36, 43, 46–48, 51, 52, 119]"

2.9.1 Dir 1.1

AMD2.15 : Replace the first paragraph of the Amplification with:

Appendix G of this document lists those implementation-defined behaviours that:

AMD2.16 : Replace the second paragraph within the "Extensions" subsection with:

An implementation may provide extensions which implement a subset of the features for a later version of the C Standard.

AMD2.17 : Replace the first bullet point within the "Extensions" subsection with:

- The *#pragma* preprocessing directive or the *_Pragma* operator;

AMD2.18 : Replace "C99" with "C99 and later" within the last sentence of the "Integer division" subsection.

2.9.2 Dir 2.1

AMD2.19 : Remove the last paragraph from the Rationale.

2.9.3 Dir 4.1

AMD2.20 : Remove the last sentence from the third paragraph from the Rationale.

2.9.4 Dir 4.3

AMD2.21 : Replace "C99" with "C99 and later" within the second bullet point of the Amplification.

AMD2.22 : Replace the last sentence of the first paragraph to the Rationale with:

If this is necessary, then it is recommended that it be achieved by using macros or *inline functions*.

2.9.5 Dir 4.6

AMD2.23 : Remove "(C99)" from within the first paragraph of the Amplification.

AMD2.24 : Replace "C99" with "C99 and later" within the second paragraph of the Amplification.

AMD2.25 : Replace the last paragraph of the Exception with:

Therefore `int main( int argc, char *argv[] )` is permitted.

2.9.6 Dir 4.9

AMD2.26 : Remove "in C99" from within the second bullet point of the Rationale.

2.10 Section 8 — Rules

AMD2.27 : Update the *Applies to* and *Source ref* entries for the Rules, as shown below:

Rule	Applies to	Source ref
Rule 1.1	Append ", C11"	
Rule 1.2	Append ", C11"	
Rule 1.3	Append ", C11"	
Rule 2.1	Append ", C11"	
Rule 2.2	Append ", C11"	
Rule 2.3	Append ", C11"	
Rule 2.4	Append ", C11"	
Rule 2.5	Append ", C11"	
Rule 2.6	Append ", C11"	
Rule 2.7	Append ", C11"	
Rule 3.1	Append ", C11"	
Rule 3.2	Append ", C11"	
Rule 4.1	Append ", C11"	Add ", C11 [Implementation J.3.4(7,8)]"
Rule 4.2	Append ", C11"	
Rule 5.1	Append ", C11"	Add ", C11 [Unspecified 8; Undefined 31]"
Rule 5.2	Append ", C11"	Add ", C11 [Undefined 31]"
Rule 5.3	Append ", C11"	
Rule 5.4	Append ", C11"	Add ", C11 [Unspecified 8; Undefined 31]"
Rule 5.5	Append ", C11"	Add ", C11 [Unspecified 8; Undefined 31]"
Rule 5.6	Append ", C11"	
Rule 5.7	Append ", C11"	
Rule 5.8	Append ", C11"	
Rule 5.9	Append ", C11"	
Rule 6.1	Append ", C11"	Add ", C11 [Implementation J.3.9(1,2)]"
Rule 6.2	Append ", C11"	
Rule 7.1	Append ", C11"	
Rule 7.2	Append ", C11"	
Rule 7.3	Append ", C11"	
Rule 7.4	Append ", C11"	Add ", C11 [Unspecified 15; Undefined 33]"
Rule 8.2	Append ", C11"	Add ", C11 [Undefined 38–41, 79, 85]"
Rule 8.3	Append ", C11"	Add ", C11 [Undefined 15]"
Rule 8.4	Append ", C11"	Add ", C11 [Undefined 41]"
Rule 8.5	Append ", C11"	
Rule 8.6	Append ", C11"	Add ", C11 [Undefined 84]"
Rule 8.7	Append ", C11"	
Rule 8.8	Append ", C11"	
Rule 8.9	Append ", C11"	
Rule 8.10	Append ", C11"	Add ", C11 [Unspecified 21; Undefined 70]"
Rule 8.11	Append ", C11"	
Rule 8.12	Append ", C11"	

Rule	Applies to	Source ref
Rule 8.13	Append ", C11"	
Rule 8.14	Append ", C11"	Add ", C11 [Undefined 68, 69]"
Rule 9.1	Append ", C11"	Add ", C11 [Undefined 11, 19]"
Rule 9.2	Append ", C11"	Add ", C11 [Undefined 82, 83]"
Rule 9.3	Append ", C11"	
Rule 9.4	Append ", C11"	
Rule 9.5	Append ", C11"	
Rule 10.1	Append ", C11"	Add ", C11 [Undefined 14, 51, 52; Implementation J.3.4(2,5), J.3.5(5), J.3.9(7)]"
Rule 10.2	Append ", C11"	
Rule 10.3	Append ", C11"	Add ", C11 [Undefined 17, 18; Implementation 3.5(4)]"
Rule 10.4	Append ", C11"	Add ", C11 [Implementation 3.6(5)]"
Rule 10.5	Append ", C11"	
Rule 10.6	Append ", C11"	
Rule 10.7	Append ", C11"	
Rule 10.8	Append ", C11"	
Rule 11.1	Append ", C11"	Add ", C11 [Undefined 24, 26, 41, 44]"
Rule 11.2	Append ", C11"	Add ", C11 [Undefined 24, 25, 44]"
Rule 11.3	Append ", C11"	Add ", C11 [Undefined 25, 37]"
Rule 11.4	Append ", C11"	Add ", C11 [Undefined 24, 37; ImplementationJ.3.7(1)]"
Rule 11.5	Append ", C11"	Add ", C11 [Undefined 25, 37]"
Rule 11.6	Append ", C11"	Add ", C11 [Undefined 24, 44; Implementation J.3.7(1)]"
Rule 11.7	Append ", C11"	Add ", C11 [Undefined 24, 44; Implementation J.3.7(1)]"
Rule 11.8	Append ", C11"	Add ", C11 [Undefined 33, 64, 65]"
Rule 11.9	Append ", C11"	
Rule 12.1	Append ", C11"	
Rule 12.2	Append ", C11"	Add ", C11 [Undefined 51]"
Rule 12.3	Append ", C11"	
Rule 12.4	Append ", C11"	
Rule 12.5	Append ", C11"	
Rule 13.1	Append ", C11"	Add ", C11 [Unspecified 18, 23]"
Rule 13.2	Append ", C11"	Add ", C11 [Unspecified 16–19; Undefined 35]"
Rule 13.3	Append ", C11"	Add ", C11 [Unspecified 16; Undefined 35]"
Rule 13.4	Append ", C11"	Add ", C11 [Unspecified 16, 19; Undefined 35]"
Rule 13.5	Append ", C11"	
Rule 13.6	Append ", C11"	Add ", C11 [Unspecified 22]"
Rule 14.1	Append ", C11"	
Rule 14.2	Append ", C11"	
Rule 14.3	Append ", C11"	
Rule 14.4	Append ", C11"	
Rule 15.1	Append ", C11"	
Rule 15.2	Append ", C11"	
Rule 15.3	Append ", C11"	
Rule 15.4	Append ", C11"	

Rule	Applies to	Source ref
Rule 15.5	Append ", C11"	
Rule 15.6	Append ", C11"	
Rule 15.7	Append ", C11"	
Rule 16.1	Append ", C11"	
Rule 16.2	Append ", C11"	
Rule 16.3	Append ", C11"	
Rule 16.4	Append ", C11"	
Rule 16.5	Append ", C11"	
Rule 16.6	Append ", C11"	
Rule 16.7	Append ", C11"	
Rule 17.1	Append ", C11"	Add ", C11 [Undefined 87, 136–143]"
Rule 17.2	Append ", C11"	
Rule 17.4	Append ", C11"	Add ", C11 [Undefined 88]"
Rule 17.5	Append ", C11"	
Rule 17.6	Append ", C11"	Add ", C11 [Undefined 77]"
Rule 17.7	Append ", C11"	
Rule 17.8	Append ", C11"	
Rule 18.1	Append ", C11"	Add ", C11 [Undefined 46, 47, 49, 62]"
Rule 18.2	Append ", C11"	Add ", C11 [Undefined 48]"
Rule 18.3	Append ", C11"	Add ", C11 [Undefined 53]"
Rule 18.4	Append ", C11"	
Rule 18.5	Append ", C11"	
Rule 18.6	Append ", C11"	Add ", C11 [Undefined 9, 10, 43]"
Rule 18.7	Append ", C11"	Add ", C11 [Undefined 62]"
Rule 18.8	Append ", C11"	Add ", C11 [Unspecified 22; Undefined 16, 75, 76]"
Rule 19.1	Append ", C11"	Add ", C11 [Undefined 54, 100]"
Rule 19.2	Append ", C11"	Add ", C11 [Unspecified 11; Undefined 64, 65]"
Rule 20.1	Append ", C11"	Add ", C11 [Undefined 102, 103]"
Rule 20.2	Append ", C11"	Add ", C11 [Undefined 34]"
Rule 20.3	Append ", C11"	Add ", C11 [Undefined 91]"
Rule 20.4	Append ", C11"	Add ", C11 [Undefined 104]"
Rule 20.5	Append ", C11"	
Rule 20.6	Append ", C11"	Add ", C11 [Undefined 93]"
Rule 20.7	Append ", C11"	
Rule 20.8	Append ", C11"	
Rule 20.9	Append ", C11"	
Rule 20.10	Append ", C11"	Add ", C11 [Unspecified 26; Undefined 3, 94, 95]"
Rule 20.11	Append ", C11"	Add ", C11 [Unspecified 26]"
Rule 20.12	Append ", C11"	
Rule 20.13	Append ", C11"	
Rule 20.14	Append ", C11"	
Rule 21.1	Append ", C11"	Add ", C11 [Undefined 99, 106, 107, 110, 114, 122, 126, 138]"
Rule 21.2	Append ", C11"	Add ", C11 [Undefined 99, 106, 107, 110, 114, 122, 126, 138]"

Rule	Applies to	Source ref
Rule 21.3	Append ", C11"	Add ", C11 [Unspecified 42, 43; Undefined 9, 10, 177–181; Implementation J.3.12(37)]"
Rule 21.4	Append ", C11"	Add ", C11 [Unspecified 35; Undefined 124–127, 183]"
Rule 21.5	Append ", C11"	Add ", C11 [Undefined 128–135, 185; Implementation J.3.12(14)]"
Rule 21.6	Append ", C11"	Add ", C11 [Unspecified 4–7, 37–40; Undefined 146–175, 198; Implementation J.3.12(16–34)]"
Rule 21.7	Append ", C11"	Add ", C11 [Undefined 119]"
Rule 21.9	Append ", C11"	Add ", C11 [Unspecified 46, 42; Undefined 187–189]"
Rule 21.10	Append ", C11"	Add ", C11 [Unspecified 48, 49; Undefined 154, 162, 193, 197; Implementation J.3.12(41–45)]"
Rule 21.11	Append ", C11"	Add ", C11 [Undefined 195, 196]"
Rule 21.12	Append ", C11"	Add ", C11 [Unspecified 27, 28; Undefined 115–117; Implementation J.3.6(9)]"
Rule 21.13	Append ", C11"	Add ", C11 [Undefined 113]"
Rule 21.14	Append ", C11"	
Rule 21.15	Append ", C11"	
Rule 21.16	Append ", C11"	Add ", C11 [Unspecified 10]"
Rule 21.17	Append ", C11"	Add ", C11 [Undefined 109, 191]"
Rule 21.18	Append ", C11"	Add ", C11 [Undefined 109, 191, 192]"
Rule 21.19	Append ", C11"	Add ", C11 [Undefined 120, 121, 184]"
Rule 21.20	Append ", C11"	
Rule 22.1	Append ", C11"	
Rule 22.2	Append ", C11"	Add ", C11 [Undefined 179]"
Rule 22.3	Append ", C11"	Add ", C11 [Implementation J.3.12(24)]"
Rule 22.4	Append ", C11"	
Rule 22.5	Append ", C11"	
Rule 22.6	Append ", C11"	Add ", C11 [Undefined 148]"
Rule 22.7	Append ", C11"	
Rule 22.8	Append ", C11"	
Rule 22.9	Append ", C11"	
Rule 22.10	Append ", C11"	

2.10.1 Rule 1.1

AMD2.28 : Replace "Section 3.1" in the first paragraph of the Amplification with "Section 1.3".

2.10.2 Rule 1.3

AMD2.29 : Replace the second paragraph of the Rationale with:

Many of the MISRA C guidelines have been designed to avoid certain undefined and unspecified behaviours. However, other behaviours are not covered by specific guidelines, for example because:

2.10.3 Rule 1.4

AMD2.30 : Add the following as a new rule after Rule 1.3:

Rule 1.4	Emergent language features shall not be used
Category	Required
Analysis	Decidable, Single Translation Unit
Applies to	C11

Amplification

Updates to the C Standard have introduced new language features. The following shall not be used:

- The *_Generic* operator;
- The *_Noreturn* function specifier and the `<stdnoreturn.h>` header file;
- The *_Atomic* type specifier, the *_Atomic* type qualifier and the facilities that are specified as being provided by `<stdatomic.h>`;
- The *_Thread_local* storage class specifier and the facilities that are specified as being provided by `<threads.h>`;
- The *_Alignas* alignment specifier, the *_Alignof* operator and the `<stdalign.h>` header file;
- Other than defining `__STDC_WANT_LIB_EXT1__` to '0', the facilities of Annex K (Bounds-checking interfaces) shall not be used.

Rationale

Use of the language features restricted by this rule may have instances of undefined, unspecified or implementation-defined behaviour associated with them. In addition, features may also exhibit well defined behaviour that does not meet developer expectations.

Any instances of undefined, unspecified or implementation-defined behaviour are diagnosed by:

- Dir 1.1, which requires that the use of implementation-defined behaviour be documented and taken into consideration; and
- Rule 1.3, which prohibits the presence of undefined and critical unspecified behaviours.

However, detection of these behaviours alone does not mitigate against well defined behaviour that does not meet developer expectations. Additional static analysis checks are needed to check for these behaviours, but there is no requirement for a static analysis tool to implement these checks as they are not specified within The Guidelines.

This Rule requires that any use of an emergent language feature be supported by a deviation to ensure that all undesirable behaviours are identified and measures put in place to ensure that they do not compromise safety or security.

See also

Dir 1.1, Rule 1.3

2.10.4 Rule 3.2

AMD2.31 : Remove the *Note:* paragraph from the Rationale.

2.10.5 Rule 4.1

AMD2.32 : Remove the "See also" section.

2.10.6 Rule 5.1

AMD2.33 : Replace "C99" with "C99 and later" within the second bullet of the Amplification.

AMD2.34 : Replace "C99" with "C99 and later" within the *Note:* of the Rationale.

2.10.7 Rule 5.2

AMD2.35 : Replace "C99" with "C99 and later" within the second bullet of the Amplification.

2.10.8 Rule 5.3

AMD2.36 : Replace "C99" with "C99 and later" within the second bullet of the Amplification.

2.10.9 Rule 5.4

AMD2.37 : Replace "C99" with "C99 and later" within the second bullet of the Amplification.

AMD2.38 : Replace "C99" with "C99 and later" within the *Note:* of the Rationale.

2.10.10 Rule 5.5

AMD2.39 : Replace "C99" with "C99 and later" within the second bullet of the Amplification.

2.10.11 Rule 5.7

AMD2.40 : Replace the last sentence of the Rationale with:

This is a *constraint* violation in C99 and later.

2.10.12 Rule 6.1

AMD2.41 : Replace "C99" with "C99 and later" within the second bullet of the Amplification.

AMD2.42 : Replace "C99" with "C99 and later" within the last paragraph of the Rationale.

AMD2.43 : Remove "C90 and to C99" from within the first paragraph of the Example.

2.10.13 Rule 6.2

AMD2.44 : Replace the first sentence of the Rationale with:

For C99 and later, the C Standard states that signed integers have exactly one sign bit, meaning that a single-bit signed bit-field will have no value bits.

AMD2.45 : Replace "the C90 Standard" with "C90" in the third paragraph of the Rationale.

2.10.14 Rule 7.2

AMD2.46 : Replace "C99" with "C99 and later" within the last two bullet points of the Amplification.

2.10.15 Rule 7.4

AMD2.47 : Replace the Amplification with:

The type used to represent characters within a string literal depends on the encoding prefix:

- *char* when there is no prefix — *character string literal*;
- *char* for a `u8` prefix — UTF-8 *string literal*;
- *wchar_t* for an `L` prefix — *wide string literal*;
- *char16_t* for a `u` prefix — *wide string literal*;
- *char32_t* for a `U` prefix — *wide string literal*.

No attempt shall be made to directly modify a character string literal, a UTF-8 string literal or a wide string literal.

The result of the address-of operator, `&`, applied to a character string literal or a UTF-8 string literal shall not be *assigned* to an object unless that object's type is "pointer to array of *const*-qualified *char*".

The result of the address-of operator, `&`, applied to a wide string literal shall not be *assigned* to an object unless that object's type is "pointer to array of *const*-qualified *wchar_t*", "pointer to array of *const*-qualified *char16_t*" or "pointer to array of *const*-qualified *char32_t*", as appropriate for the encoding prefix.

AMD2.48 : Replace "C99" with "C99 and later" within the last paragraph of the Rationale.

2.10.16 Rule 9.4

AMD2.49 : Replace "The C99 Standard" with "The C Standard" within the first paragraph of the Rationale.

2.10.17 Rule 9.5

AMD2.50 : Remove "(C99 Section 6.7.8)" from within the last sentence of the Rationale.

2.10.18 Section 8.10.2 — Essential type

AMD2.51 : Replace the *Note:* below the table with:

Note: Implementations of C99 and later may provide *extended integer types*, each of which would be allocated a location appropriate to its rank and signedness.

2.10.19 Rule 10.4

AMD2.52 : Replace the first sentence of the Amplification with:

This rule applies to operators that are described in the *usual arithmetic conversions* section of the C Standard.

2.10.20 Rule 10.5

AMD2.53 : Replace "C99" with "C99 and later" at the start of the fourth bullet point of the Rationale.

AMD2.54 : Replace the last sentence of the Exception with:

This allows the implementation of Boolean models in C90.

AMD2.55 : Replace the first line of the Example with:

```
( bool_t ) false    /* Compliant - 'false' from stdbool.h is essentially Boolean */
```

2.10.21 Section 8.11 — Pointer type conversions

AMD2.56 : Replace "(C99 only)" with "(C99 and later)" at the end of the first bullet point in the last bullet list.

AMD2.57 : Replace the two paragraphs starting "In C99, any..." and "In C90, any..." with:

Any implicit conversion that does not fall into this subset of pointer conversions either leads to undefined behaviour (C90) or violates a *constraint* (C99 and later).

2.10.22 Rule 11.1

AMD2.58 : Replace the *Note:* in the Exception with:

Note: exception 3 covers the implicit conversions that commonly occur when:

2.10.23 Rule 11.3

AMD2.59 : Remove the last sentence of the Rationale.

2.10.24 Rule 11.4

AMD2.60 : Replace the first part of the *Note:* in the Rationale with:

Note: the types *intptr_t* and *uintptr_t*, declared in `<stdint.h>` (C99 and later),

2.10.25 Rule 11.8

AMD2.61 : Replace the second *Note:* in the Rationale with:

Note: removal of the *restrict* type qualifier (C99 and later) is benign.

2.10.26 Rule 12.1

AMD2.62 : Replace the first three rows of the table in the Amplification with:

Primary	identifier, constant, string literal, `(` expression `)`, generic selection	16 (high)
Postfix	`[]` `()` (function call) `.` –> ++ (post-increment) –– (post-decrement) `(){}` (compound literal)	15
Unary	++ (pre-increment) –– (pre-decrement) `&` `*` + – ~ `!` *sizeof _Alignof defined* (preprocessor)	14

2.10.27 Rule 13.1

AMD2.63 : Replace "C99 permits" with "later versions of the C Standard permit" in the second sentence of the Rationale.

2.10.28 Rule 13.2

AMD2.64 : Replace the last two paragraphs of the Amplification with:

Sequence points are summarized in Annex C of the C Standard. The sequence points in C90 are a subset of those in later versions.

Full expressions are defined in the *statements and blocks* section of the C Standard.

2.10.29 Rule 13.6

AMD2.65 : Replace "C99" with "C99 and later" at the start of the third paragraph of the Rationale.

2.10.30 Rule 14.2

AMD2.66 : Replace "C99" with "C99 and later" at the end of the third bullet point of the Amplification.

AMD2.67 : Remove "C99" from the first paragraph of the Example.

2.10.31 Rule 15.3

AMD2.68 : Replace "C99 is" with "C99 and later are" at the start of the last paragraph of the Rationale.

2.10.32 Rule 16.1

AMD2.69 : Replace "C99" with "C99 and later" within the *switch-clause:* block of the Amplification.

2.10.33 Rule 17.1

AMD2.70 : Replace the Amplification paragraph with:

None of *va_list*, *va_arg*, *va_start*, *va_end* and *va_copy* shall be used.

2.10.34 Rule 17.4

AMD2.71 : Replace "C99" with "C99 and later" within the *Note:* of the Rationale.

AMD2.72 : Replace "C99" with "C99 and later" within the Example.

2.10.35 Rule 17.6

AMD2.73 : Replace "The C99 language standard" with "The C Standard" within the first sentence of the Rationale.

AMD2.74 : Remove "C99" from within the first sentence of the Example.

2.10.36 Rule 18.1

AMD2.75 : Replace the last *Note:* of the Amplification with:

Note: within the description of the semantics for the "additive operators", the C Standard states that, for the purposes of pointer arithmetic, a pointer to an object that is not an array is treated as if it were a pointer to the first element of an array with a single element.

AMD2.76 : Remove the last sentence of the third paragraph of the Rationale.

2.10.37 Rule 20.4

AMD2.77 : Replace the penultimate paragraph of the Example with:

The following example is only compliant in C90.

2.10.38 Rule 21.3

AMD2.78 : Replace the Amplification with:

The identifiers *calloc*, *malloc*, *realloc*, *aligned_alloc* and *free* shall not be used and no macro with one of these names shall be expanded.

2.10.39 Rule 21.6

AMD2.79 : Replace the first paragraph of the Amplification with:

This rule applies to the functions that are specified as being provided by `<stdio.h>` and the wide-character equivalents specified as being provided by `<wchar.h>`.

2.10.40 Rule 21.7

AMD2.80 : Replace the Amplification with:

The identifiers *atof*, *atoi*, *atol* and *atoll* shall not be used and no macro with one of these names shall be expanded.

2.10.41 Rule 21.8

AMD2.81 : Replace Rule 21.8 with the following:

Rule 21.8	The Standard Library termination functions of <stdlib.h> shall not be used

C90 [Undefined 93; Implementation 70–71]
C99 [Undefined 172; Implementation J.3.12(36–37)]
C11 [Undefined 182, 185; Implementation J.3.12(38-39)]

Category	Required
Analysis	Decidable, Single Translation Unit
Applies to	C90, C99, C11

Amplification

The termination functions are *abort*, *exit*, *_Exit* and *quick_exit*.

Identifiers with these names shall not be used and no macro with one of these names shall be expanded.

Rationale

These functions have undefined and implementation-defined behaviours associated with them.

2.10.42 Rule 21.10

AMD2.82 : Replace the first paragraph of the Amplification with:

This rule applies to the functions that are specified as being provided by <time.h> and *wcsftime*.

None of these identifiers shall be used and no macro with one of these names shall be expanded.

2.10.43 Rule 21.21

AMD2.83 : Add the following as a new rule after Rule 21.20:

Rule 21.21 The Standard Library function *system* of `<stdlib.h>` shall not be used

C90 [Implementation 73]
C99 [Undefined 175; Implementation J.3.2(11), J.3.12(38)]
C11 [Undefined 186; Implementation J.3.2(12), J.3.12(40)]

Category	Required
Analysis	Decidable, Single Translation Unit
Applies to	C90, C99, C11

Amplification

The identifier *system* shall not be used and no macro with this name shall be expanded.

Rationale

This function has undefined and implementation-defined behaviour associated with it.

Errors related to the use of *system* are a common cause of security vulnerabilities.

2.10.44 Rule 22.1

AMD2.84 : Replace the Amplification with:

This rule applies to *malloc*, *calloc*, *realloc*, *aligned_alloc* and *fopen*.

2.10.45 Rule 22.5

AMD2.85 : Replace the first paragraph of the Rationale with:

Within the section on "files", The Standard states that the address of a `FILE` object used to control a stream may be significant and a copy of the object may not give the same behaviour. This rule ensures that such a copy cannot be made.

2.11 Section 9 — References

AMD2.86 : Replace the entry for reference 42 with:

[42] MISRA Compliance:2020 *Achieving compliance with MISRA Coding Guidelines*,
ISBN 978-1-906400-26-2 (PDF),
HORIBA MIRA Limited, Nuneaton, February 2020

AMD2.87 : Add a new entry to the end of the list of references:

[43] ISO/IEC 9899:2018, *Programming languages — C*,
International Organization for Standardization, 2018

2.12 Appendix A — Summary of guidelines

AMD2.88 : Insert the following after the entry for Rule 1.3:

Rule 1.4	Required	Emergent language features shall not be used

AMD2.89 : Replace the entry for Rule 21.8 with:

Rule 21.8	Required	The Standard Library termination functions of `<stdlib.h>` shall not be used

AMD2.90 : Insert the following after the entry for Rule 21.20:

Rule 21.21	Required	The Standard Library function *system* of `<stdlib.h>` shall not be used

2.13 Appendix B — Guideline attributes

AMD2.91 : Update the *Applies to* entries for the Directives and Rules in line with the changes detailed in Section 2.9 and Section 2.10 of this document.

AMD2.92 : Insert the following row into the table after the row for Rule 1.3:

Rule 1.4	Required	C11	Decidable, Single Translation Unit

AMD2.93 : Insert the following row into the table after the row for Rule 21.20:

Rule 21.21	Required	C90, C99, C11	Decidable, Single Translation Unit

2.14 Appendix C — Type safety issues with C

2.14.1 C.1.1

AMD2.94 : Replace “C99” with “C99/C11” within the first and second list items.

2.15 Appendix D — Essential types

2.15.1 D.1

AMD2.95 : Replace the *Note:* in the paragraph immediately before the table with:

Note: for C99 and later, any *extended integer type* is allocated a location appropriate to the rank and signedness inferred by the C Standard.

AMD2.96 : Replace the last sentence in the paragraph immediately after the table with:

In C99 and later, “rank” is a term applied only to integer types.

2.15.2 D.6

AMD2.97 : Replace “The ISO C99 standard” with “The C Standard” at the start of the first sentence.

2.15.3 D.6.4

AMD2.98 : Replace "The C99 standard" with "The C Standard" at the start of the first sentence.

2.15.4 D.7.12

AMD2.99 : Insert the following as a new section after D.7.11:

D.7.12 Generic selection (_Generic)

The *essential type* is the same as the *essential type* of the result expression.

2.16 Appendix F — Process and tools checklist

AMD2.100 : Replace the whole Appendix with:

Withdrawn – superseded by Appendix A of MISRA Compliance:2020 [42].

2.17 Appendix G — Implementation-defined behaviour checklist

AMD2.101 : Move the first table into a sub-section by inserting a new heading and explanatory text before it:

G.1 C90

The numbers in the first column of the table identify the number of an item in the relevant section of Annex G of the C90 Standard [2]. The numbering starts from the beginning of that section and does not include subsequent Technical Corrigenda. So, for example, G.3.1 refers to the first item in section G.3.

AMD2.102 : Move the second table into a sub-section by inserting a new heading and explanatory text before it:

G.2 C99 and C11

The numbers in the first column of the table identify the number of an item in the relevant section of Annex J of the C99 Standard [8] or C11 Standard [13], as appropriate. The numbering starts from the beginning of that section and does not include subsequent Technical Corrigenda. So, for example, J.3.1 refers to the first item in section J.3.

AMD2.103 : Replace the second table as follows:

Annex	Annex Item		Implementation-defined behaviour
	C99	C11	
J.3.1	1	1	How a diagnostic is identified.
J.3.2	2	2	The name and type of the function called at program startup in a freestanding environment.
	3	3	The effect of program termination in a freestanding environment.
	4	4	An alternative manner in which the main function may be defined.
	5	5	The values given to the strings pointed to by the argv argument to main.
	10	11	The set of environment names and the method for altering the environment list used by the getenv function.

Annex	Annex Item		Implementation-defined behaviour
	C99	C11	
	11	12	The manner of execution of the string by the system function.
J.3.3	1	1	Which additional multibyte characters may appear in identifiers and their correspondence to universal character names.
	2	2	The number of significant initial characters in an identifier.
J.3.4	1	1	The number of bits in a byte.
	2	2	The values of the members of the execution character set.
	4	4	The value of a char object into which has been stored any character other than a member of the basic execution character set.
	6	6	The mapping of members of the source character set (in character constants and string literals) to members of the execution character set.
	7	7	The value of an integer character constant containing more than one character or containing a character or escape sequence that does not map to a single-byte execution character.
	8	8	The value of a wide character constant containing more than one multibyte character, or containing a multibyte character or escape sequence not represented in the extended execution character set.
	9	9	The current locale used to convert a wide character constant consisting of a single multibyte character that maps to a member of the extended execution character set into a corresponding wide character code.
	10	11	The current locale used to convert a wide string literal into corresponding wide character codes.
	11	12	The value of a string literal containing a multibyte character or escape sequence not represented in the execution character set.
J.3.5	1	1	Any extended integer types that exist in the implementation.
	2	2	Whether signed integer types are represented using sign and magnitude, two's complement, or ones' complement, and whether the extraordinary value is a trap representation or an ordinary value.
J.3.6	1	1	The accuracy of the floating-point operations and of the library functions in <math.h> and <complex.h> that return floating-point results.
	2	3	The rounding behaviors characterized by non-standard values of FLT_ROUNDS.
	3	4	The evaluation methods characterized by non-standard negative values of FLT_EVAL_METHOD.
	4	5	The direction of rounding when an integer is converted to a floating-point number that cannot exactly represent the original value.
	5	6	The direction of rounding when a floating-point number is converted to a narrower floating-point number.
	6	7	How the nearest representable value or the larger or smaller representable value immediately adjacent to the nearest representable value is chosen for certain floating constants.
	7	8	Whether and how floating expressions are contracted when not disallowed by the FP_CONTRACT pragma.
	8	9	The default state for the FENV_ACCESS pragma.
	9	10	Additional floating-point exceptions, rounding modes, environments, and classifications, and their macro names.
	10	11	The default state for the FP_CONTRACT pragma.
	11		Whether the “inexact” floating-point exception can be raised when the rounded result actually does equal the mathematical result in an IEC 60559 conformant implementation.

Annex	Annex Item		Implementation-defined behaviour
	C99	C11	
	12		Whether the “underflow” (and “inexact”) floating-point exception can be raised when a result is tiny but not inexact in an IEC 60559 conformant implementation.
J.3.9	2	2	Allowable bit-field types other than _Bool, signed int, and unsigned int.
	3	4	Whether a bit-field can straddle a storage-unit boundary.
	4	5	The order of allocation of bit-fields within a unit.
J.3.10	1	1	What constitutes an access to an object that has volatile-qualified type.
J.3.11	*	1	The locations within #pragma directives where header name preprocessing tokens are recognized.
	1	2	How sequences in both forms of header names are mapped to headers or external source file names.
	2	3	Whether the value of a character constant in a constant expression that controls conditional inclusion matches the value of the same character constant in the execution character set.
	3	4	Whether the value of a single-character character constant in a constant expression that controls conditional inclusion may have a negative value.
	4	5	The places that are searched for an included < > delimited header, and how the places are specified or the header is identified.
	5	6	How the named source file is searched for in an included " " delimited header.
	6	7	The method by which preprocessing tokens (possibly resulting from macro expansion) in a #include directive are combined into a header name.
	8	9	Whether the # operator inserts a \ character before the \ character that begins a universal character name in a character constant or string literal.
	9	10	The behavior on each recognized non-STDC #pragma directive.
	10	11	The definitions for __DATE__ and __TIME__ when respectively, the date and time of translation are not available.
J.3.12	1	1	Any library facilities available to a freestanding program, other than the minimal set required by clause 4.
	4	4	Whether the feraiseexcept function raises the “inexact” floating-point exception in addition to the “overflow” or “underflow” floating-point exception.
	5	5	Strings other than "C" and "" that may be passed as the second argument to the setlocale function.
	6	6	The types defined for float_t and double_t when the value of the FLT_EVAL_METHOD macro is less than 0.
	9	9	The values returned by the mathematics functions on underflow range errors, whether errno is set to the value of the macro ERANGE when the integer expression math_errhandling & MATH_ERRNO is nonzero, and whether the “underflow” floating-point exception is raised when the integer expression math_errhandling & MATH_ERREXCEPT is nonzero.
	11	12	The base-2 logarithm of the modulus used by the remquo functions in reducing the quotient.
	33	35	The meaning of any n-char or n-wchar sequence in a string representing a NaN that is converted by the strtod, strtof, strtold, wcstod, wcstof, or wcstold function.
	34	36	Whether or not the strtod, strtof, strtold, wcstod, wcstof, or wcstold function sets errno to ERANGE when underflow occurs.
	36	38	Whether open streams with unwritten buffered data are flushed, open streams are closed, or temporary files are removed when the abort or _Exit function is called.
	37	39	The termination status returned to the host environment by the abort, exit, or _Exit function. Or quick_exit for C11.

Annex	Annex Item		Implementation-defined behaviour
	C99	C11	
	44	46	Whether the functions in <math.h> honor the rounding direction mode in an IEC 60559 conformant implementation, unless explicitly specified otherwise.
J.3.13	1	1	The values or expressions assigned to the macros specified in the headers <float.h>, <limits.h>, and <stdint.h>.
	2	3	The number, order, and encoding of bytes in any object (when not explicitly specified in this International Standard).

2.18 Appendix H.1 — Undefined behaviour

AMD2.104 : Add the following to the end of the "C90 Id" bullet point:

The numbering starts from the beginning of section G.2 and does not include subsequent Technical Corrigenda;

AMD2.105 : Add the following to the end of the "C99 Id" bullet point:

The numbering starts from the beginning of section J.2 and does not include subsequent Technical Corrigenda;

AMD2.106 : Insert the following bullet point after the "C99 Id" bullet point:

- "C11 Id" is the number of the undefined behaviour in Annex J of The C11 Standard. The numbering starts from the beginning of section J.2 and does not include subsequent Technical Corrigenda;

AMD2.107 : Replace the table, as follows:

Id			Decidable	Guidelines	Notes
C90	C99	C11			
	1	1		N/A	This behaviour is listed in C99 but each such instance is also given its own entry in Annex J. The entry for this behaviour is therefore redundant.
1	2	2	Yes		
2			Yes		
3			Yes	Rule 20.10	
	3	3	Yes		
	4	4	Yes		
		5	No		
	5	6	Yes		
	6	7	Yes		
5			Yes	Rule 5.2	
6			Yes	Rule 17.3	
8	7	8	Yes		
	8	9	No	Dir 4.12, Rule 18.6, Rule 21.3	
9			No	Dir 4.12, Rule 18.6, Rule 21.3	
	9	10	No	Dir 4.12, Rule 18.6, Rule 21.3	

Id			Decidable	Guidelines	Notes
C90	C99	C11			
	10	11	No		Compliance with Rule 9.1 avoids a common cause of this undefined behaviour but it is not sufficient to avoid all situations in which an indeterminate value might arise.
	11	12	No		The following rules help to avoid this behaviour: Rule 9.1, Rule 11.2, Rule 11.3, Rule 11.4, Rule 11.5 and Rule 19.1. However, if a trap representation is copied into an object that does not have character type, for example using memmove, memcpy or via a pointer to character type as permitted by the exception of Rule 11.3, it is not possible to avoid this behaviour.
	12	13	No	Rule 11.2,Rule 11.3, Rule 11.4, Rule 11.5	
	13	14	No		The following rules help to avoid this behaviour: Rule 9.1, Rule 10.1, Rule 11.2, Rule 11.3, Rule 11.4, Rule 11.5, and Rule 19.1. However, if the Exception of Rule 11.3 is used then it is not possible to prevent generation of a negative zero.
10	14	15	Yes	Rule 5.6, Rule 5.7, Rule 8.3	
15			No	Dir 4.1, Dir 4.14, Rule 10.3	
		16	No	Rule 18.8	
	15	17	No	Dir 4.1, Dir 4.14, Rule 10.3	
	16	18	No	Dir 4.1, Dir 4.14, Rule 10.3	
	17	19	No	Rule 9.1, Rule 11.2, Rule 11.3, Rule 11.4, Rule 11.5, Rule 19.1	
16	18	20	Yes		
		21	No		
	19	22	Yes		
17	20	23	Yes		
*	21	24	No	Rule 11.1, Rule 11.2, Rule 11.4, Rule 11.6	
	22	25	No	Rule 11.2, Rule 11.3, Rule 11.5	
27	23	26	No	Rule 11.1	
4	24	27	Yes		
*	25	28	Yes		
	26	29	Yes		
	27	30	Yes		
7	28	31	Yes	Rule 5.1, Rule 5.2, Rule 5.3, Rule 5.4, Rule 5.5	
	29	32	Yes	Rule 21.2	
11			Yes		

Id			Decidable	Guidelines	Notes
C90	C99	C11			
12	30	33	No	Rule 7.4, Rule 11.4, Rule 11.8	
13			Yes		
14			Yes	Rule 20.2	
	31	34	Yes	Rule 20.2	
18	32	35	No	Rule 13.2, Rule 13.3, Rule 13.4	
19	33	36	No	Dir 4.1, Dir 4.14	
20			No	Rule 11.3, Rule 11.4, Rule 11.5	
	34	37	No	Rule 11.3, Rule 11.4, Rule 11.5	
	35		Yes		
21			Yes		
22	36	38	No	Rule 8.2, Rule 17.3	Rule 17.3 is only applicable to, and only required for, C90.
23			No	Rule 8.2, Rule 17.3	
24			No	Rule 5.6, Rule 5.7, Rule 8.3, Rule 8.4, Rule 8.5, Rule 11.1, Rule 21.2	
25			No	Rule 8.4, Rule 8.5, Rule 11.1, Rule 21.2, Rule 17.3	
	37	39	No	Rule 8.4, Rule 8.5, Rule 11.1, Rule 21.2	
	38	40	No	Rule 8.2	
	39	41	No	Rule 5.6, Rule 5.7, Rule 8.2, Rule 8.3, Rule 8.4, Rule 8.5, Rule 11.1, Rule 21.2	
		42	Yes		
26	40	43	No	Dir 4.1, Dir 4.14	
28			Yes	Rule 11.1	
29	41	44	Yes	Rule 11.1, Rule 11.2, Rule 11.6, Rule 11.7	
	42	45	No	Dir 4.1	
		*	No		Added by C18
30	43	46	No	Dir 4.14, Rule 18.1	
*	44	47	No	Dir 4.14, Rule 18.1	
31	45	48	No	Dir 4.14, Rule 18.2	
	46	49	No	Rule 18.1	
*	47	50	No		
32	48	51	No	Dir 4.14, Rule 10.1, Rule 12.2	

Id			Decidable	Guidelines	Notes
C90	C99	C11			
	49	52	No		Compliance with Dir 4.14 avoids some instances of this undefined behaviour that are related to data from external sources. Compliance with Rule 10.1 avoids this undefined behaviour except when the expression being left-shifted has an unsigned type that is promoted to a signed type.
33	50	53	No	Rule 18.3	
34	51	54	No	Rule 19.1	
*	52	55	Yes		
*	53	56	Yes		
*	54	57	Yes		
*	55	58	Yes		
35	56	59	Yes		
36	57	60	Yes		
37	58	61	Yes		
38			Yes	Rule 6.1	
	59	62	No	Rule 18.7	
	60	63	Yes		
39	61	64	No	Rule 11.4, Rule 11.8, Rule 19.2	
40	62	65	No	Rule 11.4, Rule 11.8, Rule 19.2	
41			No	Rule 9.1	
*	63	66	Yes		
*	64	67	Yes		
	65	68	No	Rule 8.14	
	66	69	No	Rule 8.14	
	67	70	Yes	Rule 8.10	
		71	No		
		72	Yes		
		73	Yes		
*	68	74	Yes		
	69	75	No	Rule 18.8	
	70	76	No	Rule 18.8	
	71	77	No	Rule 17.6	
	72	78	Yes		
*	73	79	Yes	Rule 8.2, Rule 11.1	
*	74	80	No		
*	75	81	Yes		
42			Yes	Rule 9.2	
	76	82	Yes	Rule 9.2	
	77	83	Yes	Rule 9.2	
44	78	84	Yes	Rule 8.6	

Id			Decidable	Guidelines	Notes
C90	C99	C11			
	79	85	Yes	Rule 8.2	
*	80	86	Yes		
45	81	87	Yes	Rule 17.1	
43	82	88	Yes	Rule 17.4	
46	83	89	Yes		
		*	Yes		Added by C18
47	84	90	Yes		
48	85	91	Yes	Rule 20.3	
	86	92	Yes		
49			Yes		
50	87	93	Yes	Rule 20.6	
51	88	94	Yes	Rule 20.10	
52	89	95	Yes	Rule 20.10	
53	90	96	Yes		
	91	97	Yes		
	92	98	Yes		
54	93	99	Yes	Rule 21.1	
55	94	100	No		Compliance with Rule 19.1 avoids a common cause of this undefined behaviour but does not prevent copying part of an object to another part of the same object, such as an array.
*	95	101	Yes		
56			Yes	Rule 17.3, Rule 20.1, Rule 20.4, Rule 21.2	
	96	102	Yes	Rule 20.1	
	97	103	Yes	Rule 20.1, Rule 21.2	
	98	104	Yes	Rule 20.4	
57			Yes	Rule 21.1, Rule 21.2	
	99	105	Yes	Rule 21.2	
	100	106	Yes	Rule 21.1, Rule 21.2	
	101	107	Yes	Rule 21.1	
60	102	108	No	Dir 4.11	
*	103	109	No	Dir 4.11, Rule 21.17 Rule 21.18	
61			Yes	Rule 17.3, Rule 21.2	
62	104	110	Yes		Compliance with Rule 21.1 prevents undefinition of the macro but no rule prevents the macro expansion from being suppressed, e.g. by means of `(assert)(E)`.
	105	111	Yes		
	106	112	Yes		
63	107	113	No	Dir 4.11, Rule 21.13	
58			Yes	Rule 21.1	

Id			Decidable	Guidelines	Notes
C90	C99	C11			
	108	114	Yes		Compliance with Rule 21.1 prevents undefinition of the macro but no rule prevents the macro expansion from being suppressed, e.g. by means of `(errno)` if it is implemented as a function-like macro. Compliance with Rule 21.2 prevents definition of the identifier errno.
	109	115	No		Compliance with Rule 21.12 avoids some instances of this undefined behaviour but does not prevent floating-point control modes from being changed.
	110	116	No	Rule 21.12	
	111	117	No	Rule 21.12	
	112	118	No	Dir 4.11	
90			No	Rule 21.7	
94			No		Compliance with Dir 4.14 avoids some instances of this undefined behaviour that are related to data from external sources.
	113	119	No		Compliance with Dir 4.14 avoids some instances of this undefined behaviour that are related to data from external sources.
*	114	120	No	Rule 21.19	
*	115	121	No	Rule 21.19	
	116	122	Yes	Rule 21.1, Rule 21.2	
	117	123	Yes		
64			Yes	Rule 21.1, Rule 21.2, Rule 21.4	
	118	124	Yes	Rule 21.1, Rule 21.2, Rule 21.4	
65	119	125	Yes	Rule 21.4	
*	120	126	No	Rule 21.4	
66	121	127	No	Rule 21.4	
67			No	Rule 21.4, Rule 21.5	Compliance with either rule is sufficient to avoid the undefined behaviour.
*	122	128	No	Rule 21.5	
*	123	129	No	Rule 21.5	
		130	No	Rule 21.5	
	124	131	No	Rule 21.5	
68			No	Rule 21.5	
	125	132	No	Rule 21.5	
69	126	133	No	Rule 21.5	
*	127	134	No	Rule 21.5	
		135	Yes		
*	128	136	No		Compliance with Rule 17.1 avoids instances of this undefined behaviour that arise through improper use of the features of <stdarg.h>.
70	129	137	No	Rule 17.1	
71			Yes	Rule 17.1, Rule 21.1, Rule 21.2	

Id			Decidable	Guidelines	Notes
C90	C99	C11			
	130	138	Yes	Rule 17.1, Rule 21.1, Rule 21.2	
75			No	Rule 17.1	
76			No	Rule 17.1	
	131	139	No	Rule 17.1	
	132	140	Yes	Rule 17.1	
73			No	Rule 17.1	
74			No	Rule 17.1	
	133	141	No	Rule 17.1	
	134	142	No	Rule 17.1	
72	135	143	Yes	Rule 17.1	
		*	Yes		Added by C18
59	136	144	Yes		
	137	145	Yes		
	138	146	No	Rule 21.6	
	139	147	No	Rule 21.6	
*	140	148	No	Rule 21.6	If Rule 21.6 is deviated then Rule 22.6 provides protection against this undefined behaviour. Rule 21.6 is preferred as it is decidable.
77	141	149	No	Rule 21.6	
	142	150	No	Rule 21.6	
78	143	151	No	Rule 21.6	
*	144	152	No	Rule 21.6	
79			No	Rule 21.6	
85			No	Rule 21.6	
	145	153	No	Rule 21.6	
	146	154	No	Rule 21.6, Rule 21.10	
*	147	155	No	Rule 21.6	
*	148	156	No	Rule 21.6	
83			No	Rule 21.6	
84			No	Rule 21.6	
	149	157	No	Rule 21.6	
82			No	Rule 21.6	
87			No	Rule 21.6	
	150	158	No	Rule 21.6	
*	151	159	No	Rule 21.6	
	152	160	No	Rule 21.6	
81	153	161	No	Rule 21.6	
97			No	Rule 21.10	
80	154	162	No	Rule 21.6, Rule 21.10	
86	155	163	No	Rule 21.6	
		164	Yes	Rule 21.6	
89	156	165	No	Rule 21.6	

Id			Decidable	Guidelines	Notes
C90	C99	C11			
*	157	166	No	Rule 21.6	
	158	167	No	Rule 21.6	
88	159	168	No	Rule 21.6	
*	160	169	No	Rule 21.6	
*	161	170	No	Rule 21.6	
*	162	171	No	Rule 21.6	
*	163	172	No	Rule 21.6	
*	164	173	No	Rule 21.6	
*	165	174	No	Rule 21.6	
*	166	175	No	Rule 21.6	
*	167	176	No	Rule 21.3	
91	168	177	No	Rule 21.3	
		178	Yes	Rule 21.3	Does not apply to C18 as the behaviour is now defined
92	169	179	No	Rule 21.3, Rule 22.2	
*	170	180	No	Rule 21.3	
*	171	181	No	Rule 21.3	
93	172	182	No	Rule 21.8	
	173	183	No	Rule 21.4	
*	174	184	No	Rule 21.19	
		185	No	Rule 21.5, Rule 21.8	
	175	186	No	Rule 21.21	
	176	187	No	Rule 21.9	
	177	188	No	Rule 21.9	
*	178	189	No	Rule 21.9	
95	179	190	No		
96	180	191	No	Dir 4.11, Rule 21.17, Rule 21.18	
	181	192	No	Dir 4.11, Rule 21.18	
*	182	193	No		Compliance with Rule 21.10 avoids this undefined behaviour except in respect of wcsxfrm.
	183	194	No	Dir 4.11	
	184	195	Yes	Rule 21.11	
	185	196	Yes	Rule 21.11	
		*	No		Added by C18
		*	No		Added by C18
		*	No		Added by C18
		*	No		Added by C18
		*	Yes		Added by C18
		*	No		Added by C18
		197	No	Rule 21.10	
	186	198	No	Rule 21.6	
	187	199	No	Dir 4.11	

Id			Decidable	Guidelines	Notes
C90	C99	C11			
	188	200	No		
	189	201	No	Dir 4.11	
	190	202	No		
	191	203	No		

2.19 Appendix H.2 — Critical Unspecified behaviour

AMD2.108 : After the second bullet point ("C99 Id"), insert:

- "C11 Id" is the number of the unspecified behaviour in Annex J of The C11 Standard;

AMD2.109 : Replace the table, as follows:

Id			Critical	Guidelines	Notes
C90	C99	C11			
1	1	1	No		
	2	2	No		
		3	No		
2	3	4	No	Rule 21.6	
3	4	5	No	Rule 21.6	
4	5	6	No	Rule 21.6	
5	6	7	No	Rule 21.6	
6			Yes		
	7	8	Yes	Rule 5.1	
	8	9	Yes		
	9	10	Yes		Compliance with Rule 21.16 avoids this unspecified behaviour in respect of *memcmp* only.
	10	11	Yes	Rule 19.2	
	11	12	Yes		
	12	13	Yes		
	13	14	Yes		Compliance with Rule 10.1 avoids generation of negative zeros when operating on expressions that have a signed type before promotion.
	14	15	Yes	Rule 7.4	
7, 8	15	16	Yes	Rule 13.2	
9	16	17	Yes	Rule 13.2	
	17	18	Yes	Rule 13.1	
7	18	19	Yes	Rule 13.2	
10	19	20	No		
	20	21	Yes	Rule 8.10	
	21	22	Yes	Rule 13.6, Rule 18.8	
7	22	23	Yes	Rule 13.1	
11	23	24	No		
*	24	25	Yes		

Id			Critical	Guidelines	Notes
C90	C99	C11			
12	25	26	Yes	Rule 20.10, Rule 20.11	
13	26		No		
		*	Yes		Added by C18 – `#line __LINE__ new-line`
	27	27	Yes	Rule 21.12	
	28	28	Yes	Rule 21.12	
	29	29	No		
	30	30	Yes	Dir 4.11	
	31	31	Yes	Dir 4.11	
14	32	32	No	Rule 21.4	
15	33	33	No	Rule 17.1	
		34	No		
	34	35	Yes	Rule 21.6	
16	35	36	Yes	Rule 21.6	
17	36	37	Yes	Rule 21.6	
18	37	38	Yes	Rule 21.6	
	38	39	No		
19	39	40	No	Rule 18.1, Rule 18.2, Rule 18.3, Rule 21.3	Compliance with either Rule 21.3 or all of Rule 18.1, Rule 18.2 and Rule 18.3 will avoid this unspecified behaviour.
	40	41	Yes	Rule 21.3	
20	41	42	Yes	Rule 21.9	C11 incorrectly omitted *align_alloc*, which was corrected in C18.
21	42	43	Yes	Rule 21.9	C11 incorrectly omitted *align_alloc*, which was corrected in C18.
		44	Yes		
		45	Yes		
22	43	46	Yes	Rule 21.10	
	44	47	Yes	Rule 21.10	
		*	Yes		Added by C18 – *thrd_exit* destructor invokation ordering
		*	Yes		Added by C18 – *tss_delete* destructor invocations with multiple threads
	45	48	Yes		
	46	49	Yes		
		50	Yes		
	47	51	Yes		
	48	52	Yes	Dir 4.11	
	49	53	Yes	Dir 4.11	
	TC3	54	Yes	Dir 4.11	Added to C99 by TC3.
	TC3	55	Yes	Dir 4.11	Added to C99 by TC3.
	50	56	Yes	Dir 4.11	
	51	57	Yes		
	52	58	Yes		

2.20 Appendix I — Example Deviation Record

AMD2.110 : Replace the whole Appendix with:

Withdrawn – superseded by Appendix B of MISRA Compliance:2020 [42].

3 References

The following documents are referenced from within this amendment:

[1] MISRA C:2012 *Guidelines for the use of the C language in critical systems* (3rd Edition), ISBN 978-1-906400-10-1 (paperback), ISBN 978-1-906400-11-8 (PDF), MIRA Limited, Nuneaton, March 2013

[2] MISRA C:2012 Technical Corrigendum 1, *Technical clarification of MISRA C:2012*, ISBN 978-1-906400-17-0 (PDF), HORIBA MIRA Limited, Nuneaton, June 2017

[3] MISRA C:2012 Amendment 1, *Additional security guidelines for MISRA C:2012*, ISBN 978-1-906400-16-3 (PDF), HORIBA MIRA Limited, Nuneaton, April 2016

[4] MISRA C:2012 *Guidelines for the use of the C language in critical systems* (3rd Edition, 1st Revision), ISBN 978-1-906400-21-7 (paperback), ISBN 978-1-906400-22-4 (PDF), HORIBA MIRA Limited, Nuneaton, February 2019

[5] ISO/IEC 9899:1999, *Programming languages — C*, International Organization for Standardization, 1999

[6] ISO/IEC 9899:2011, *Programming languages — C*, International Organization for Standardization, 2011

[7] ISO/IEC 9899:2018, *Programming languages — C*, International Organization for Standardization, 2018

[8] MISRA Compliance:2020 *Achieving compliance with MISRA Coding Guidelines*, ISBN 978-1-906400-26-2 (PDF), HORIBA MIRA Limited, Nuneaton, February 2020