MISRA – What's that all about?

Andrew Banks

The Speaker

Andrew Banks

- Software Engineer & Standards Evangelist

Focus: helping YOU to get your software right, first time!

- Biography
  - Over 35 years experience in developing real-time embedded software systems, across a number of industries
  - Chartered Fellow of the British Computer Society
  - Member of the Institution of Engineering & Technology
- Standards
  - Chairman of MISRA C Working Group since June 2013...
    ... Working Group member since 2007
  - Chairman of the BSI Software Testing Working Group
    … UK Head-of-Delegation to ISO/IEC JTC1/SC7
  - Contributor to ISO 29119 “Software Testing”
  - Contributor to ISO 26262 2nd Edition “Functional Safety”
  - etc

LDRA MISRA Resources

- MISRA C Training page
  https://ldra.com/training/misra-c/

- The MISRA Language Guidelines, MISRA C:2025, and MISRA C++:2023 – What they're for, what's current and what isn't
  https://ldra.com/misra/

Agenda – Session 1

1. The C Language...
   ... and what is wrong with it
2. An introduction to The MISRA Consortium Limited (TMCL)
3. An introduction to MISRA C

The C Language...

... and what is wrong with it

“In the beginning, the Universe was created.

This has made a lot of people very angry
… and been widely regarded as a bad move.

The Restaurant at the End of the Universe
Book 2 of the Douglas Adams’ 5-part Trilogy
The Hitch Hiker’s Guide To The Galaxy

Far back in the mists of ancient time…

- 1963 CPL — *Cambridge Programming Language*
  Christopher Strachey et al of University of Cambridge

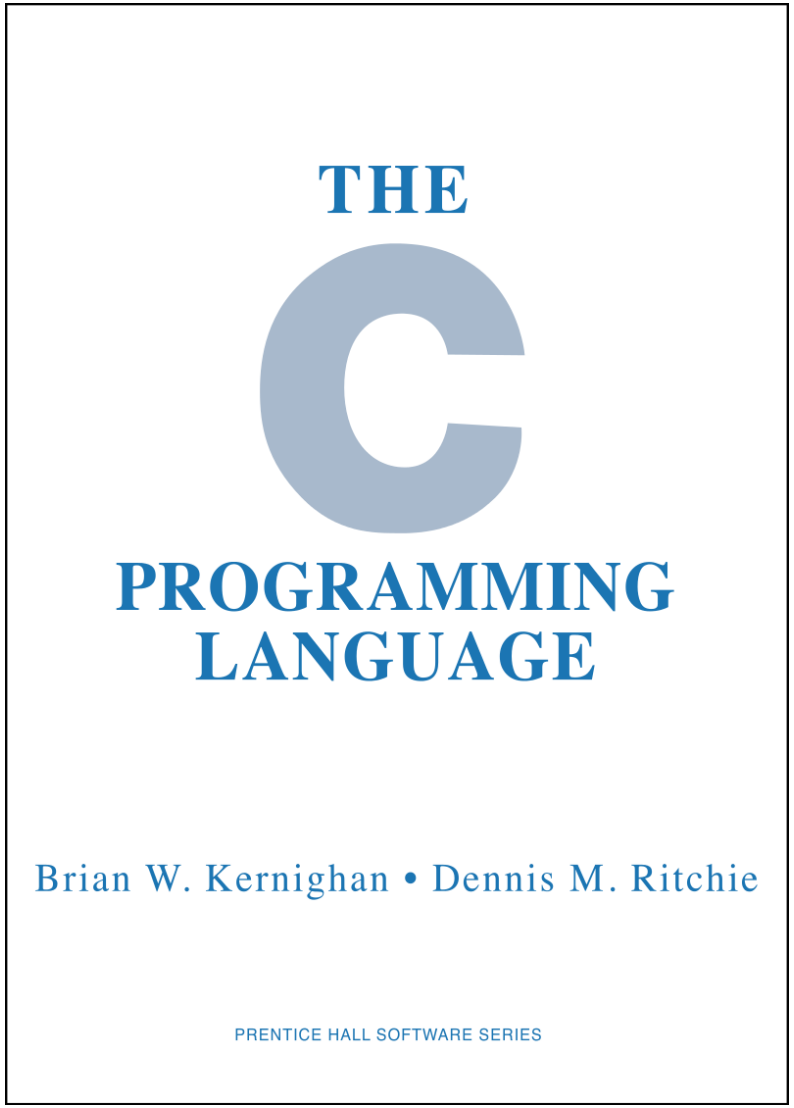
  Later known as the *Combined Programming Language*
  with the involvement of the University of London

- 1967 BCPL — *Bootstrap CPL* (or *Basic CPL*)
  Martin Richards at the University of Cambridge

- 1969 B — Ken Thompson and Dennis Ritchie at Bell Labs.

The C Language – A Quick History

- K&R C
  - 1972 First created by Dennis Ritchie
  - 1976 Lint, the first C static analyser
    ... created by Stephen Johnson
  - 1978 The C Programming Language published
- ANSI C
  - 1989 First standardized version
    ANSI X3.159-1989 (aka C89)



The standard C Language

- ANSI C
  - 1989 ANSI X3.159-1989 aka C89 First standardized version
- ISO C
  - 1990 ISO/IEC 9899:1990 aka C90 Equivalent to C89
  - 1995 Amendment 1 aka C95
  - 1999 ISO/IEC 9899:1999 aka C99
  - 2011 ISO/IEC 9899:2011 aka C11
  - 2018 ISO/IEC 9899:2018 aka C18 or C17 A “TC” in all but name
  - 2024 ISO/IEC 9899:2018 aka C24 or C23
- Very few (if any) of you will be using ANSI C any more!

The C Language – What is the problem?

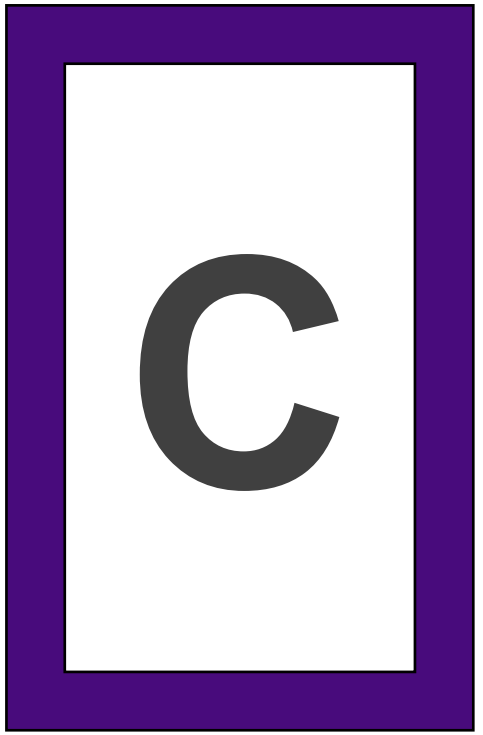
- Despite its popularity, there are several drawbacks with the C language, eg:
  - Language misuse and obfuscation
  - Language misunderstanding
  - Lack of Run-time error checking
  - The ISO Standard language definition is incomplete

“We don't demand solid facts!
What we demand is a total absence of solid facts.
We demand rigidly defined areas of doubt and uncertainty!

Vroomfondel the Philosopher, in
The Hitch Hiker’s Guide To The Galaxy
by Douglas Adams

The C Language – What is the problem?

- The ISO Standard language definition is incomplete
  - Behaviour that is Unspecified — Annex J.1 — 61 instances
  - Behaviour that is Undefined — Annex J.2 — 211 instances
  - Behaviour that is Implementation Defined — Annex J.3 — 120 instances
  - Behaviour that is Locale-dependant — Annex J.4 — 15 instances



The C Language – What is the problem?

- The ISO Standard language definition is incomplete
  - Undefined behaviour is *behaviour, upon use of a nonportable or erroneous program construct or of erroneous data, for which the International Standard imposes no requirements*
  - An example of undefined behaviour is the behaviour on integer overflow

    211 instances
  - Unspecified behaviour invokes *the use of an unspecified value, or other behaviour where the International Standard provides two or more possibilities and imposes no further requirements on which is chosen in any instance*
  - An example of unspecified behaviour is the order in which the arguments to a function are evaluated.

    61 instances

If not C, then what?

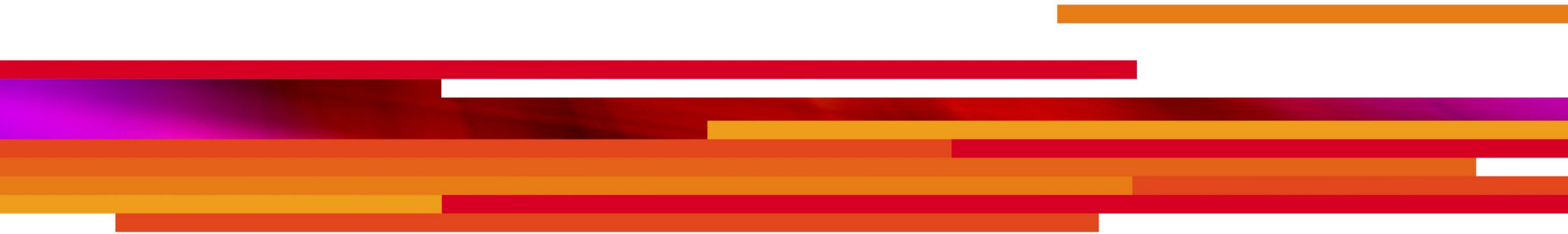
So, if not C, then what?

- 1977 etc Modula, Modula-2
- 1980(?) Perspective Pascal
- 1983 Ada (also including SPARK)
- 1993 Lua
- 1995 Java
- 2010 (yes!) Rust
- 2025 ?

There is another way…

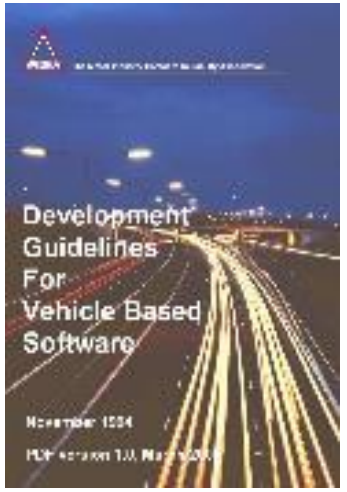
MISRA – A Quick History

From the beginning, to MISRA C:2012

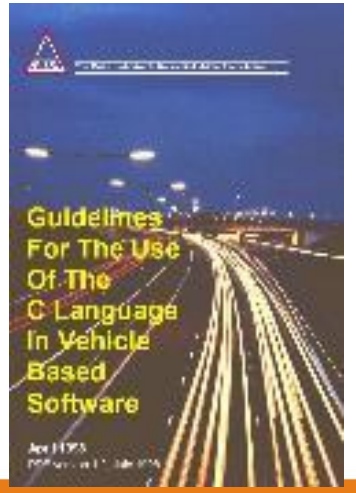


Original MISRA publications

- **November 1994**
  Development guidelines for vehicle based software
  (The MISRA Guidelines)
  - The first automotive publication concerning functional safety
  - Commenced more than 10 years before work started on ISO 26262



- **April 1998**
  Guidelines for the use of the C language in vehicle based software
  (MISRA C)



- **December 1998**
  IEC 61508 (1st Edition) published

Subsequent Work

- Working Groups
  - MISRA C
  - MISRA C++
  - MISRA Auto Code
  - MISRA Safety Case
  - MISRA Software Quality Measures
- Leads the UK's contribution to ISO 26262
- etc

MISRA in the International Standards picture

- ISO TC 22 Road vehicles
  - SC 32 Electrical and electronic components and general system aspects
    - WG 8 Functional Safety
    - WG 11 Cybersecurity
    - WG 12 Software update
    - WG 13 Safety for driving automation systems
    - WG 14 Safety and Artificial Intelligence

MISRA in the International Standards picture

- ISO/IEC JTC 1 Information Technology
  - WG 13 Trustworthiness
  - SC 7 Software and Systems Engineering
    - WG 4 Tools and environment
    - WG 6 Software product and system quality
    - WG 7 Life cycle management
    - WG 26 Software testing
  - SC 22 Programming languages, their environments and system software interfaces
    - WG 14 C
    - WG 21 C++
    - WG 23 Programming language vulnerabilities

An introduction to MISRA C

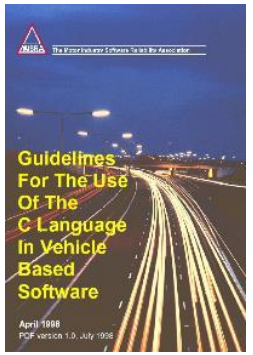
“The MISRA C Guidelines define a subset of the C language in which the opportunity to make mistakes is either removed or reduced.

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.”

MISRA C – A Quick History

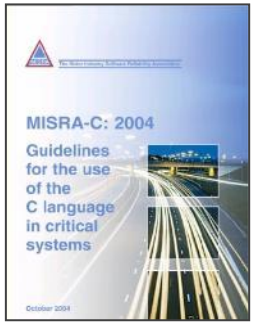
MISRA-C:1998

- "Guidelines for the use of the C language in vehicle based software"
- Compatible with ISO/IEC 9899:1990 (aka C90)



MISRA-C:2004

- "Guidelines for the use of the C language in critical systems"
- Remains compatible with ISO/IEC 9899:1990 (aka C90)



MISRA C:2012 (3rd Edition)

- Adds compatibility with ISO/IEC 9899:1999 (aka C99)
- Updated to 1st Revision in 2019 to include additional security guidelines introduced by AMD1, and TC1

MISRA C – A Quick History

MISRA C:2023 (25th Anniversary Edition)

- Adds compatibility with ISO/IEC 9899:2011 & :2018 (aka C11 and C17/C18)
- Consolidation of AMD2, AMD3, AD4 and TCs

MISRA C:2025

- Minor updates, released due to ongoing delays with ISO/IEC 9899:202x

MISRA C:20xx

- Work in progress
- Will add compatibility with ISO/IEC 9899:2024 (aka C23/C24)

Evolution of MISRA C

Edition	D Directives	R Rules	T Total	M Mandatory	R Required	A Advisory	D Disapplied
MISRA C:1998	-	127	127	-	94	33	-
MISRA C:2004	-	142	142	-	121	21	-
MISRA C:2012	16	143	159	0 + 10	9 + 101	7 + 32	-
MISRA C:2023	21	200	221	0 + 23	14 + 137	7 + 40	-
MISRA C:2025	22	201	223	0 + 22	14 + 139	8 + 39	0 + 1

For MISRA C:2012 and later, the numbers are shown as d + r:

d = the number of Directives, and

r = the number of Rules

Deleted guidelines are not counted!

MISRA C in one slide!

- 223 Guidelines: 22 Directives (5 sections) and 201 Rules (23 sections)

Directives

- The implementation
- Compilation and build
- Requirements traceability
- Code design
- Concurrency considerations

Rules

- A standard C environment
- Unused code
- Comments
- Character sets & lexical conventions
- Identifiers
- Types
- Literals & constants
- Declarations & definitions
- Initialization
- The essential type model
- Pointer type conversions
- Expressions
- Side effects
- Control statement expressions
- Control flow
- Switch statements
- Functions
- Pointers and arrays
- Overlapping storage
- Preprocessing directives
- Standard libraries
- Resources
- Generic Selections

MISRA Compliance

- MISRA Compliance

- Permits
  - Permits — Deviation permits for MISRA Compliance — Published (*)

- Guideline Reclassification Plans
  - Autocode — GRP for Automatically Generated Code … Published for 2023 and 2025 — Published
  - ISO 26262 — GRP for ISO 26262 ASILs — WIP

(*) = for 2012… will be revised for MISRA C:2023/2025 as required

The MISRA C Supporting Cast

- Not being maintained
  - Addendum 1 Rule Mapping (MISRA C:2012 v MISRA C:2004)

- Coverage of MISRA C against…
  - Addendum 2 ISO/IEC TS 17961:2013 *C Secure*
  - Addendum 3 CERT C 2016 Edition
  - Addendum 4 ISO/IEC 24772 *Language Vulnerabilities*
  - Addendum 5 MITRE Common Weakness Enumeration
  - Addendum 6 Rust

MISRA – What's that all about?

Session 2

Andrew Banks

Agenda – Session 2

1. MISRA C Myths
2. Adoption and Use of MISRA C
3. MISRA C in an ISO 26262 context
4. The MISRA C Training Course in Munich

Matters Arising from Day 1

- Who is part of the MISRA C Working Group?

- Tool Vendors
  - LDRA 1 UK
  - Perforce 2 UK
  - AbsInt 1 DE
  - Bugseng 1 IT
  - Parasoft 1 PL
  - Vector/PClint 2 US
  - Blackduck 1 CA
  - GitHub 2 (pending)

- End Users
  - GE Aerospace 1 UK
  - QNX 1 UK
  - CodeThink 1 UK
  - Robert Bosch 1 IN
  - Rimac Technologies 1 CR
  - Vitesco Technologies 1 DE
  - Woven by Toyota 1 US

MISRA C Myths

MISRA C “Myths and Legends”

Does *not*

- Say it must be followed all the time
- Define style or metric guidelines
- Stop developers writing code

Does

- Required controlled deviations
- Require style and metrics be applied
- Force developers to think

Myth Busting #1

- The Misunderstanding
  - MISRA C is only applicable to the automotive industry
- The History
  - MISRA C was originated by the automotive industry, for the automotive industry... and we are proud of our automotive heritage.
- The Reality
  - MISRA C is applicable to any industry that requires high-integrity software
  - MISRA C has been adopted by many industries, including medical, rail, aerospace, space and defence. eg:

JPL
Jet Propulsion Laboratory
California Institute of Technology

- http://lars-lab.jpl.nasa.gov/JPL_Coding_Standard_C.pdf
- http://www.stroustrup.com/JSF-AV-rules.pdf

Myth Busting #2

- The Misunderstanding
  - MISRA C is only a safety coding standard, not a secure/security one
- The History
  - MISRA C:2012 suggested (in its vision) its use in *safety-related software*
- The Reality
  - MISRA C also suggests (in its vision) its applicability to *any application with high integrity or high reliability requirements*
  - Unfortunately, a perception remains...

Myth busting #3

- The Misunderstanding
  - MISRA C is all that is needed
    - “If I do not have any compiler warnings, and no static analysis warnings, then I have good code”
- The Reality
  - Application of MISRA C is meant to take place within the context of a documented software development process that meets the requirements of an appropriate standard
    - ISO 12207
    - ISO 26262 (or IEC 61508)
    - DO178C
  - Good software does not simply begin and end with using MISRA guidelines (or similar)

Myth busting #4

- The Misunderstanding
  - I can simply ignore all of the *Advisory* guidelines
- The Reality
  - No you can't!
  - However, *MISRA Compliance* permits you to formally *Disapply* an *Advisory* guideline, within a *Guideline Reclassification Plan*
  - More on this, in my *MISRA Compliance* webinar.

MISRA C
Adoption and Use

What is MISRA C and why is it needed?

- It is a language subset for the C language:
  - ISO 9899:1990 (C90)
  - ISO 9899:1999 (C99)
  - ISO 9899:2011 (C11)
  - ISO 9899:2018 (C18)

- It helps to make code behave predictably by negating the influences of undefined-, unspecified- and implementation-defined behaviour contained within the C language

- It eliminates common programming errors

- It is **not** there just to stop developers from producing code ☺

Adoption

- MISRA C should be adopted at the start of a project
  - Trying to make code compliant after-the-fact is a time-consuming and frustrating task
  - Defects may be introduced, and re-certification needed
- It does not always make sense to try and make legacy or third-party code compliant when it was not written to be compliant
  - Time-consuming
  - Risk of introducing defects into well-proven systems

Real life experiences of subsets

- Many people jump directly to MISRA C, particularly the “rules”, and ignore the wider context in which it is meant to be used
  - The software development process
  - The other “methods” / “techniques and measures”
  - The wider system engineering context
- Even in MISRA C the introductory material is frequently overlooked

Real life experiences of subsets

- Static analysis tools are frequently used but only by:
  - Managers wanting to “tick boxes” and dash-boards
  - Programmers wanting to “keep MISRA C [checker] happy”
- Many “rules” considered picky
  - people do not necessarily understand or appreciate the rationale behind them … maybe because they rely on a tool, and don’t read the book!

MISRA C in an ... ISO 26262 context

MISRA C – In an ISO 26262 Context

ISO 26262-6:2018, Section 5.4.3

- Criteria for suitable modelling, design or programming languages that are not sufficiently addressed by the language itself shall be covered by the corresponding guidelines, or by the development environment, considering the topics listed in Table 1

- Example 1: MISRA C is a coding guideline for the programming language C and includes guidance on automatically generated code

> **5.4.3** Criteria for suitable modelling, design or programming languages (see 5.4.2) that are not sufficiently addressed by the language itself shall be covered by the corresponding guidelines, or by the development environment, considering the topics listed in Table 1.
>
> EXAMPLE 1 MISRA C (see Reference [3]) is a coding guideline for the programming language C and includes guidance for automatically generated code.

MISRA C – In an ISO 26262 Context

ISO 26262-6:2018, Table 1

Table 1 — Topics to be covered by modelling and coding guidelines

Topics		ASIL			
		A	B	C	D
1a	Enforcement of low complexity[a]	++	++	++	++
1b	Use of language subsets[b]	++	++	++	++
1c	Enforcement of strong typing[c]	++	++	++	++
1d	Use of defensive implementation techniques[d]	+	+	++	++
1e	Use of well-trusted design principles[e]	+	+	++	++
1f	Use of unambiguous graphical representation	+	++	++	++
1g	Use of style guides	+	++	++	++
1h	Use of naming conventions	++	++	++	++
1i	Concurrency aspects[f]	+	+	+	+

MISRA C – In an ISO 26262 Context

- ISO 26262-6:2018, Section 8.4.5
  - Design principles for software unit design and implementation at the source code level as listed in Table 6 shall be applied to achieve the following properties:
    - correct order of execution of subprograms and functions within the software units, based on the software architectural design;
    - consistency of the interfaces between the software units;
    - correctness of data flow and control flow between and within the software units;
    - simplicity;
    - readability and comprehensibility;
    - robustness;
    - suitability for software modification; and
    - verifiability

MISRA C – In an ISO 26262 Context

ISO 26262-6:2018, Table 6

Table 6 — Design principles for software unit design and implementation

Principle		ASIL			
		A	B	C	D
1a	One entry and one exit point in subprograms and functions[a]	++	++	++	++
1b	No dynamic objects or variables, or else online test during their creation[a]	+	++	++	++
1c	Initialization of variables	++	++	++	++
1d	No multiple use of variable names[a]	++	++	++	++
1e	Avoid global variables or else justify their usage[a]	+	+	++	++
1f	Restricted use of pointers[a]	+	++	++	++
1g	No implicit type conversions[a]	+	++	++	++
1h	No hidden data flow or control flow	+	++	++	++
1i	No unconditional jumps[a]	++	++	++	++
1j	No recursions	+	+	++	++

[a] Principles 1a, 1b, 1d, 1e, 1f, 1g and 1i may not be applicable for graphical modelling notations used in model-based development.

NOTE For the C language, MISRA C (see Reference [3]) covers many of the principles listed in Table 6.

MISRA C – In an ISO 26262 Context

Static Analysis, control flow analysis and data flow analysis are mentioned twice as a set:

- Table 7 ... software unit verification
- Table 10 ... verification of software integration

Control-flow and data-flow analysis are also mentioned in Table 4:

- Table 4 ... verification of software architectural design

MISRA C – In an ISO 26262 Context

ISO 26262-6:2018, Table 7 (unit)

Table 7 — Methods for software unit verification

Methods		ASIL			
		A	B	C	D
1a	Walk-through[a]	++	+	o	o
1b	Pair-programming[a]	+	+	+	+
1c	Inspection[a]	+	++	++	++
1d	Semi-formal verification	+	+	++	++
1e	Formal verification	o	o	+	+
1f	Control flow analysis[b, c]	+	+	++	++
1g	Data flow analysis[b, c]	+	+	++	++
1h	Static code analysis[d]	++	++	++	++
1i	Static analyses based on abstract interpretation[e]	+	+	+	+
1j	Requirements-based test[f]	++	++	++	++
1k	Interface test[g]	++	++	++	++
1l	Fault injection test[h]	+	+	+	++
1m	Resource usage evaluation[i]	+	+	+	++
1n	Back-to-back comparison test between model and code, if applicable[j]	+	+	++	++

MISRA C – In an ISO 26262 Context

ISO 26262-6:2018, Table 10

Table 10 — Methods for verification of software integration

Methods		ASIL			
		A	B	C	D
1a	Requirements-based test[a]	++	++	++	++
1b	Interface test	++	++	++	++
1c	Fault injection test[b]	+	+	++	++
1d	Resource usage evaluation[c, d]	++	++	++	++
1e	Back-to-back comparison test between model and code, if applicable[e]	+	+	++	++
1f	Verification of the control flow and data flow	+	+	++	++
1g	Static code analysis[f]	++	++	++	++
1h	Static analyses based on abstract interpretation[g]	+	+	+	+

This also maps to the MISRA C guideline scope:

- Unit Verification — Single-translation-unit guidelines
- Integration — System-wide guidelines

LDRA's MISRA C Training Course…

2-day Course

Understanding MISRA C:2025

Munich

15 - 16 Oct 25

MISRA C Training Course – Promotion

2-Day Training Course

UNDERSTANDING MISRA C:2025

Date:
15 - 16 Oct 2025

Location:
Munich, Germany

Time:
09:00 AM - 05:00 PM

Training Overview

This comprehensive two-day course on MISRA C:2025 delivers a thorough understanding of developing safe, secure embedded systems. Created with senior MISRA C Working Group representatives, this interactive training provides unique industry insights on implementation strategies, deviation handling, and compliance techniques across safety-critical sectors. The latest 2025 release is reflected throughout the curriculum.

**Please note that this is a paid training program. To know more and register, visit www.ldra.com/training/misra-c/.*

Who Should Attend?

- Software Team Leaders
- Software Quality Managers
- Software Engineers
- System Engineers
- Test Engineers

REGISTER NOW

info@ldra.com www.ldra.com/training

Agenda

DAY 1 & 2

- **Introduction to MISRA** – A background introduction to the organization, its contributors, and its products
- **Use of 'C' in embedded systems** – A discussion surrounding the C programming language, its advantages, and its drawbacks
- **Adopting and using MISRA C** – Presenting and explaining the recommended approach to the use of MISRA C
- **Security considerations** – An explanation of the underlying thinking that makes MISRA standards equally applicable to both safety- and security-critical applications, with examples
- **Deviation and MISRA Compliance:2020** – It is unlikely that any meaningful application can be developed that will comply with all MISRA rules. This discussion focuses on how those instances should be handled
- **MISRA C:2025 Guidelines** – An overview of how the guidelines are constructed and how they are applied, with reference to key examples

Trainer

Andrew Banks
Technical Specialist, LDRA | Chairman - MISRA C Working Group

Andrew has over 30 years of experience in high-integrity real-time/embedded software development. He is committed to standards development – he has been involved with MISRA since 2007 and has been Chairman of the MISRA C Working Group since early 2013; he is the Chairman of the BSI "Software Testing" Working Group; and an active participant in other BSI, ISO, IET and SCSC work, including the recent revision of ISO 26262.

Fees and Details

- The 2-Day training costs € 775.
- Fees include course materials, certificate of participation, lunch and refreshments.

REGISTER NOW

Contact details

sales@ldra.com +44 (0)151 649 9300

info@ldra.com www.ldra.com/training

What is MISRA C and Why is it Important?

What is MISRA C?

- A coding standard for the C programming language
- Developed to improve safety, security, and reliability
- Used in safety-critical industries like automotive, aerospace and defence, industrial controls and robotics, and medical…..

Why is MISRA C Important?

- Reduces risk of software bugs and vulnerabilities
- Ensures code quality and consistency
- Helps organizations meet compliance with standards like ISO 26262, DO-178C, IEC 61508
- Improves maintainability and readability of code

How is MISRA used and Who should use it?

How is MISRA C used?

- It is used during design, development, and code review
- Teams apply MISRA guidelines to their C code
- Deviations are documented and justified
- Helps teams detect and avoid unsafe code patterns

Who Should use MISRA C?

- Software Team Leaders
- Software Quality Managers
- Software Engineers
- System Engineers
- Test Engineers

Which Industries should use MISRA C?

MISRA C is specifically designed for industries where safety, security, and reliability of software are critical. The following industries are the key ones (but not an exhaustive list):

- Aerospace & Defense
- Automotive
- Energy
- Industrial Controls
- Medical Devices
- Rail
- Space

Additional Information

Munich Specific

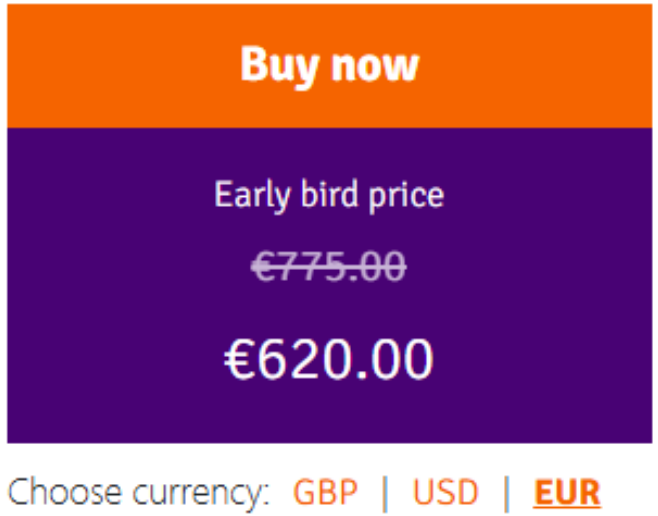
[Data Sheet: Understanding MISRA C:2025 - 2 Day training course Munich]

[Data Sheet: Understanding MISRA C:2025 - 2 Day training course overview]

General

[LDRA MISRA Training page]

[MISRA Website Page: The MISRA Language Guidelines, MISRA C:2025, and MISRA C++:2023]



MISRA C Training Course – Agenda – Day 1

Day 1			
09:00	09:15	Welcome and Administration	
09:15	10:45	Introduction and Overview	56 slides
10:45	11:00	Break	
11:00	12:30	Adopting and using MISRA C Introduction to MISRA C	29 slides 26 slides
12:30	13:30	Lunch	
13:30	15:30	MISRA Compliance	64 slides
15:30	15:45	Break	
15:45	17:30	MISRA C Guidelines – the Directives	27 slides

MISRA C Training Course – Agenda – Day 2

Day 2			
09:00	10:45	MISRA C Guidelines – the Rules	215 slides!
10:45	11:00	Break	
11:00	12:30	MISRA C Guidelines – the Rules	
12:30	13:30	Lunch	
13:30	15:15	MISRA C Guidelines – the Rules	
15:15	15:30	Break	
15:30	17:15	MISRA C Guidelines – the Rules	
17:15	17:30	Wrap-up	

MISRA C Training Course

- Location:
  - TASKING Towers, Streitfeldstrasse, München
- Dates:
  - Wednesday 15th and Thursday 16th October 2025
- Logistics:
  - I plan to arrive on the morning of Tuesday 14th at the TASKING Offices… Tom?
  - Hotel to be confirmed (thanks to Melanie for some suggestions)
  - Maybe some of the TASKING colleagues will join me/us for dinner and beer on Tuesday?
  - Any plans for dinner on Wednesday (with the delegates)?

In Summary…

In Summary

- The C Language is in widespread use, despite its limitations … many attempts have been made to supplant it, without success
- MISRA C is
  - widely respected as guiding good practice
  - appropriate for use in all high-integrity and high-reliability environments
- Writing high-integrity and high-reliability needs the right language … C, on it's own, leaves a lot to be desired
- You may not want to start from here, but at least with MISRA C, we offer you a guide.


Q
&
A