MISRA AC INT:2025

Introduction to the MISRA guidelines for the use of automatic code generation in embedded systems

January 2025

First published January 2025 by The MISRA Consortium Limited
1 St James Court
Whitefriars
Norwich
Norfolk
NR3 1RU
UK

www.misra.org.uk

ISBN 978-1-911700-13-5 PDF

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

MISRA AC INT:2025

Introduction to the MISRA guidelines for the use of automatic code generation in embedded systems

January 2025

MISRA Mission Statement

MISRA provides 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, engineering consultancies and academics which seeks to research and promote best practice in developing safety- and security-related electronic systems and other software-intensive applications.

To this end, MISRA conducts research projects and publishes documents that provide accessible information for engineers and management.

MISRA also facilitates the dissemination and exchange of information between practitioners through supporting and holding technical events.

Disclaimer

Compliance with these guidelines does not in itself ensure error-free robust software.

Compliance with the requirements of this document, or any other standard, does not of itself confer immunity from legal obligations.

Acknowledgements

The MISRA AC Working Group

The MISRA Consortium would like to thank the following current members of the MISRA AC Working Group and other individuals for their significant contributions to the writing of this document and during its review process:

Richard Blachford	HORIBA MIRA
John Botham	Ricardo UK
Bill Burdock	Formerly Jaguar Land Rover
Ivan Ellis	Jaguar Land Rover
Alasdair Mitchell	BorgWarner
Chris Rowell	Ricardo UK
David Ward	HORIBA MIRA

The MISRA Consortium would also like to thank the following past member of the MISRA AC Working Group for his contributions to this document via material in the first edition:

Steve Montgomery

Other Acknowledgements

Contents

1 Executive summary

The MISRA Autocode Guidelines (MISRA AC) family of documents deals with the application of language subsets for automatic code generation purposes in embedded systems. This document, MISRA AC INT, explains the structure of, and relationships between, the MISRA AC documents.

2 Revisions to MISRA AC INT

This revision of the document, MISRA AC INT:2025, is based on the original version, 1.0, published in 2007. The document has undergone extensive revision to reflect the changes in the MISRA AC document family and related standards, and the advances in use and awareness of modelling and automatic code generation approaches since that time.

Whilst the MISRA Autocode Guidelines documents originated within the automotive industry, they are not automotive specific and can readily be used in other contexts. This revision of the document has been renamed to reflect this.

3 Background

Developers of embedded control systems have been accustomed to using *models* in the development process for several decades. A model is a graphical representation of the control system that is constructed in, and can be run using, a *modelling package*. *Automatic code generation* is a process in which these models are then translated into an equivalent source code representation for incorporation into the software of the embedded system.

Automatic code generation has long been popular as a means of speeding up the development of prototype control applications because of the relative ease with which changes to a control model can be reflected in the software executing in an electronic control unit (*rapid prototyping*). It is also frequently used in *hardware-in-the-loop* (HiL) systems to generate real time simulations of the plant under control, such as an engine (*plant modelling*).

Crucially, however, automatic code generation has also become increasingly common as a means of allowing source code for production embedded systems to be created from executable models without detailed programming knowledge.

The MISRA guidelines for the use of automatic code generation in embedded systems (hereafter referred to as the MISRA Autocode Guidelines) provide guidance on how to deploy these techniques robustly for production.

Although the guidelines are aimed especially at the use of automatic code generators in safety-related systems, they are applicable wherever high-quality models are needed, including plant models.

4 Language and process issues

4.1 Concepts

A model is a diagram drawn according to the rules of a *modelling language*. There are several modelling languages in which models can be developed. Models are usually viewed in a graphical representation and edited using a graphical editor, even though the underlying notation may be textual or mathematical. Models are "executable" by simulation in the modelling package, a process analogous to interpretation of some programming languages.

An *automatic code generator* is a tool that translates a model from a modelling language representation into a *target language* representation (known as *automatically generated code* or *autocode*). The process of automatic code generation is analogous to that of compilation of a programming language, in that the high-level representation of a program is transformed into a lower-level representation. For example, C compilers typically translate the C source code into assembly language[1] (from which an executable image is derived using an assembler and linker). In a similar way, an automatic code generator is designed to process programs written in a modelling language and deliver output in the form of source code in the target language.

Automatic code generators often require supplementary information to be linked to a model to be able to translate the model correctly. For a given modelling language there may be several automatic code generators that are capable of translating models into a target language. The nature and format of supplementary information is normally specific to the automatic code generator being used.

C remains a highly popular target language, especially for automotive applications. Other target languages are possible, for instance C++ or Ada.

4.2 Modelling language considerations

4.2.1 Types of modelling language

Modelling languages are divided broadly into two categories: *object-oriented languages* and *dataflow-oriented languages*.

UML and its variant (or "profile") SysML are widely used object-oriented modelling languages. UML and SysML naturally lend themselves well to object-oriented problems but their adoption for automatic code generation has largely lagged behind that of dataflow-oriented languages.

Simulink and Stateflow are popular dataflow languages. Others include SCADE and ASCET.

The MISRA Autocode Guidelines are currently concerned solely with dataflow languages since these are more widely used for automatic code generation.

4.2.2 Modelling language semantics and standardization

Programming languages provide programmers with an abstraction of the machine on which their program will execute. To be useful, a programming language should have a well-defined *syntax*

[1] This can involve a single tool or a pair of front- and back-end tools communicating via an intermediate language.

(grammar) and *semantics* (meaning) in order that a programmer can accurately predict, and reason effectively about, the behaviour of a program when it is executed.

The syntaxes and semantics of high-level programming languages are typically complex but relatively well defined. The C programming language, for example, has an ISO standard [1]. There are some semantic areas in C that are unspecified, intentionally undefined or left open for interpretation by the C language implementer. Most of these areas are well-documented in the ISO standard. Further guidance on avoiding these areas — and on other constructs whose semantics may differ from the programmer's expectation — is provided by documents such as MISRA C [2]. These define *language subsets*.

Modelling languages are also typically complex. However, the precision and degree of standardization with which the syntax and semantics of a modelling language are defined varies significantly according to the language. Documents defining modelling language subsets, such as MISRA AC GMG [3] and MISRA AC SLSF [4], can be essential to give guidance on avoiding ill-defined or easily-misunderstood constructs at the model level in a similar way to which MISRA C does for C.

4.3 Automatic code generator considerations

It is generally possible for automatic code generators to produce code that uses undesirable features of the target language, such as those whose semantics are undefined. This could occur due to, for instance, the modelling language features used in — or supplementary information provided with — a model, the user's configuration of the tool, or the programming of the code generator itself. For this reason, similar criteria can be applied to automatically generated code as to manually produced code.

Guidelines at the model level and, if available, on the use of the automatic code generator can be applied to aid in avoiding modelling language or tool features that lead to the generation of code with unwanted features. Such guidelines can also aid in avoiding modelling language or code generator features that may give rise to code whose behaviour deviates from that intended (including, potentially, deviations from the behaviour of the model in simulation).

4.4 Process considerations

The MISRA Autocode Guidelines can be used in isolation but are expected to form part of a coherent process covering the production and verification of models and of the code generated automatically from them.

For instance, ISO 26262 [5] Part 6 *Product development at the software level* was written with model-based development and automatic code generation as a use case, alongside more traditional manual coding processes. Where model-based development is in use this can lead to tailoring of the software lifecycle; for example, unit testing and integration testing are permitted to be performed on the model, and code generated from the integrated model. To support the robustness of such a process, ISO 26262 includes the following requirements:

- Appropriate modelling and programming languages must be selected that meet certain requirements;
- If these requirements are not directly achieved by a language, then language subsets and analysis tools are to be applied;
- Specifically in the case of modelling languages, the graphical representation must be unambiguous in conjunction with the use of modelling style guidelines;
- Static analysis is applied to the model, along with certain other analyses such as control flow analysis, data flow analysis and analysis of structural coverage during testing;

- Back-to-back testing is required, whereby a model and the generated code are stimulated with the same test cases, and it is ensured that the results are comparable.

5 Autocode guidelines document scope and structure

The MISRA Autocode Guidelines are structured hierarchically with levels corresponding to:

- Generic guidelines, applicable to all modelling languages;
- Guidelines specific to a particular modelling language;
- Guidelines specific to a particular automatic code generator;
- Guidelines specific to the target language generated by the automatic code generator.

Originally, one set of guidelines was provided at each level. MISRA's policy is now to maintain guidelines only at the top two levels for the following reasons:

- Code generator guidelines are best developed and maintained by the respective tool vendors, who will be better equipped to keep the guidance in line with their tool updates.
- For C, guidelines specific to the target language are now integrated into MISRA C.

For each level of the hierarchy, Table 1 summarizes the documents that are currently recommended at the time of issue of this document and that are previously available but now deprecated.

To be effective, the guidelines applied to a development project — whether MISRA guidelines or from other sources such as the code generator vendor — should cover each level in the hierarchy.

Table 1: Current and deprecated documents in the MISRA Autocode Guidelines series

Level	Current documents	Related notes	Deprecated documents
Generic level	MISRA AC GMG [3]	1, 3	—
Modelling language level	MISRA AC SLSF [4]	1, 3	—
Code generator level	—		MISRA AC TL [6]
Target language level	Included in MISRA C [2]	1, 2, 3	MISRA AC AGC [7]

Notes:

1. MISRA Compliance [8] should be applied at all levels. The Compliance appendices of the MISRA Autocode Guidelines documents identify adaptations for its use at the level of modelling.
2. For application to automatically generated code, the guidance formerly found in MISRA AC AGC has been incorporated into MISRA C since its 2012 edition, including a rule re-categorization scheme. A template Guideline Reclassification Plan (GRP) reflecting this scheme is forthcoming at the time of issue of this document.
3. Users should consult the MISRA website and discussion forum for further document revisions and related publications, including errata or amendments.

6 References

[1] ISO/IEC 9899:2018, *Information technology — Programming Languages — C*, International Organization for Standardization, 2018

[2] MISRA C:2023 *Guidelines for the Use of the C Language in Critical Systems*, ISBN 978-1-911700-08-1 (paperback), ISBN 978-1-911700-09-8 (PDF), The MISRA Consortium Limited, 2023

[3] MISRA AC GMG:2023 *Generic modelling design and style guidelines,* ISBN 978-1-906400-04-3 (paperback), ISBN 978-1-906400-05-0 (PDF), The MISRA Consortium Limited, 2023

[4] MISRA AC SLSF:2023 *Modelling design and style guidelines for the application of Simulink and Stateflow,* ISBN 978-1-911700-07-4 (paperback), ISBN 978-1-911700-06-7 (PDF), The MISRA Consortium Limited, 2023

[5] ISO 26262:2018 series, *Road vehicles — Functional Safety*, International Organization for Standardization, 2018

[6] MISRA AC TL *Modelling style guidelines for the application of TargetLink in the context of automatic code generation,* ISBN 978-1-906400-01-9 (PDF), Motor Industry Research Association, 2007

[7] MISRA AC AGC *Guidelines for the application of MISRA C:2004 in the context of automatic code generation,* ISBN 978-1-906400-02-6 (PDF), Motor Industry Research Association, 2007

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

7 Revision history

Revision	Description of changes	Release date
Version 1.0	First version release	November 2007
MISRA AC INT:2025	Second version release, reviewed and revised throughout	January 2025