WHITE PAPER

HOW TO COMPLY WITH THE IEC 61508 STANDARD

The international standard, IEC 61508, places requirements on the quality of software, for which tools such as QA·C and QA·C++ are ideally positioned to enforce. With the highest adoption in the industry, and a strong heritage in safety-critical applications, QA·C and QA·C++ have been certified as being “fit for purpose” to be used as tools by development teams wishing to achieve compliance with IEC 61508. This document describes the parts of the standard that are addressed by using QA·C with MISRA C and QA·C++ with MISRA C++.

PRQA Programming Research

INTRODUCTION

Electronic equipment is increasingly being used in safety critical environments, and the software used in these products is becoming more and more complex. Exhaustive testing to ensure that there is no situation in which a failure could occur is rarely possible, and therefore systems must be designed in such a way to prevent failure or ensure controlled behavior if failures arise.

The introduction of standards has been an important factor in ensuring the development of robust software in safety critical applications. Coding standards such as MISRA, which mandate the use of a specific subset of a programming language, have been a major factor in the improvement of software quality. The international standard IEC 61508 mandates the use of better development processes, including the use of coding standards to encourage further gains in software quality.

ABOUT IEC 61508

International Standard IEC 61508 provides a generic approach to functional safety. Providing a basic framework with core requirements for sector specific standards of all safety lifecycle activities, it can be applied to systems comprising elements of electrical and/or electronic and/or computer-based systems (generically referred to as programmable electronic systems).

A range of industry sectors have released sector specific standards using the IEC 61508 framework including IEC 61511 (process), IEC 61513 (nuclear), IEC 62061 (manufacturing), EN 50128 and EN 50129 (railway), and ISO 26262 (automotive). A certification according to IEC 61508 is sufficient to be used in projects requiring certification of some standards derived from IEC 61508.

As the complexity of a system increases, the risk of systematic failures and random hardware failures increases. The standard includes guidance that helps developers mitigate these risks through the provision of appropriate requirements and processes.

A system to which IEC 61508 is applicable may have different levels of user risk or safety requirements. To specify the necessary safety measures of a given system, IEC 61508 introduces Safety Integrity Levels (SIL 1 – 4), where SIL 4 represents the most stringent level. This allows different methods to be applied depending upon the SIL of the system at a functional level.

Within the standard, Part 3 specifically addresses the software requirements, placing requirements on the initiation of software development; software architectural design and software unit design and implementation. QA·C with MISRA C and QA·C++ with MISRA C++ provide an approach to achieve compliance within these sub-sections of the standard.

ABOUT PRQA, QA·C AND QA·C++

PRQA pioneered coding standard inspection and is recognised worldwide as the coding standards expert because of its industry-leading software inspection and standards enforcement technology. PRQA's QA·C and QA·C++ static analysis tools offer two of the most comprehensive parsers available today, providing detailed information and accurately enforcing coding standards.

QA·C can be configured to enforce compliance with many coding standards, including MISRA C:2004 and MISRA C:2012. Likewise, QA·C++ can be configured to enforce compliance with many coding standards, including MISRA C++: 2008. Both tools can also be used for compliance checking in safety-related systems.

IEC 61508 COMPLIANCE WITH PRQA TOOLS

QA·C 8.1.2 with MISRA C and QA·C++ 3.1 with an extended MISRA C++ have been certified by SGS – TÜV-SAAR as fit for purpose to develop safety-related software up to SIL 4 according to IEC 61508 (if used as described in the appropriate Safety Manual).

The integration of one of the PRQA Tools into a development process for a safety relevant system is described in the respective safety manual, which includes all relevant information for the proper usage of the tool in a safety-related environment.

The QA·C with MISRA C certificate pack for IEC 61508 includes:

- Safety Manual, QA·C with MISRA C
- IEC 61508 Certificate from SGS–TÜV-SAAR
- Report to the Certificate

The QA·C++ with MISRA C++ Extended certificate pack for IEC 61508 includes:

- Safety Manual, QA·C++ with MISRA C++ Extended
- MISRA C++ Extended Compliance Module
- IEC 61508 Certificate from SGS –TÜV-SAAR
- Report to the Certificate

IEC 61508 – PART 3: SOFTWARE REQUIREMENTS

Part 3 of IEC 61508 addresses the software requirements of a safety-related system, including several tables that define the methods that must be considered in order to achieve compliance with the standard. The following tables summarize where QA·C with MISRA C (referred to as "QA·C") and QA·C++ with MISRA C++ Extended (referred to as "QA·C++") can be used to ensure and demonstrate compliance. The related Safety Manual also contains all necessary requirements relating to documentation and references to results and validation.

SECTION 6 – ADDITIONAL REQUIREMENTS FOR MANAGEMENT OF SAFETY-RELATED SOFTWARE

Reference		QA·C	QA·C++
6.2	Requirements		
6.6.2	Function safety planning	✓	✓

TABLE 1 – SOFTWARE SAFETY LIFECYCLE – OVERVIEW

Reference		QA·C	QA·C++
10.1	Software safety requirements specification	-	-
10.2	Validation plan for software aspects of system safety	-	-
10.3	Software design and development		
	Support tools and programming languages: select a suitable set of tools	✓	✓
10.4	Programmable electronics integration	-	-
10.5	Software operation and modification procedures	-	-
10.6	Software aspects of system safety validation	-	-

SECTION 7.4.4 – REQUIREMENTS FOR SUPPORT TOOLS, INCLUDING PROGRAMMING LANGUAGES

Reference			QA·C	QA·C++
7.4.4.2	Software off-line support tools shall be selected as a coherent part of the software development activities		✓	✓
7.4.4.10	The software or design representation (including a programming language) selected shall:	b) use only defined language features	✓	✓
		d) contain features that facilitate the detection of design or programming mistakes	✓	✓
7.4.4.12	Programming languages for the development of all safety-related software shall be used according to a suitable programming language coding standard		✓	✓
7.4.4.13	A programming language coding standard shall specify good programming practice, proscribe unsafe language features (e.g. undefined language features), promote code understandability.		✓	✓
7.9	Software verification			
7.9.2.12	Verification of the code		✓	✓

ANNEX A – GUIDE TO THE SELECTION OF TECHNIQUES AND MEASURES

For each technique or measure in the tables there is a recommendation for safety integrity levels (SIL) 1 to 4. These recommendations are as follows:

- “HR” indicates that the method is highly recommended for the identified SIL;
- “R” indicates that the method is recommended for the identified SIL;
- “---” indicates that the method has no recommendation for or against being used;

TABLE A.2 – SOFTWARE DESIGN AND DEVELOPMENT – SOFTWARE ARCHITECTURE DESIGN

Technique/Measure		SIL				QA·C	QA·C++
		1	2	3	4		
14.	Static resource allocation	---	R	HR	HR	✓	✓

TABLE A.3 – SOFTWARE DESIGN AND DEVELOPMENT – SUPPORT TOOLS AND PROGRAMMING LANGUAGE

Technique/Measure		SIL				QA·C	QA·C++
		1	2	3	4		
1.	Suitable programming language	HR	HR	HR	HR	✓	✓
2.	Strongly typed programming language	HR	HR	HR	HR	✓	✓
3.	Language subset	---	---	HR	HR	✓	✓
4a.	Certified tools and certified translators	R	HR	HR	HR	✓	✓
4b.	Tools and translators: increased confidence from use	HR	HR	HR	HR	✓	✓

TABLE A.4 – SOFTWARE DESIGN AND DEVELOPMENT – DETAILED DESIGN

Technique/Measure		SIL				QA·C	QA·C++
		1	2	3	4		
3.	Defensive programming	---	R	HR	HR	✓	✓
5.	Design and coding standards	R	HR	HR	HR	✓	✓
6.	Structured programming	HR	HR	HR	HR	✓	✓

TABLE A.9 – SOFTWARE VERIFICATION

Technique/Measure		SIL				QA·C	QA·C++
		1	2	3	4		
3.	Static analysis	R	HR	HR	HR	✓	✓

TABLE B.1 – DESIGN AND CODING STANDARDS

Technique/Measure		SIL				QA·C	QA·C++
		1	2	3	4		
1.	Use of coding standard to reduce likelihood of errors	HR	HR	HR	HR	✓	✓
2.	No dynamic objects	R	HR	HR	HR	✓	✓
3a.	No dynamic variables	---	R	HR	HR	✓	✓
4.	Limited use of interrupts	R	R	HR	HR	-	✓
5.	Limited use of pointers	---	R	HR	HR	✓	✓
6.	Limited use of recursion	---	R	HR	HR	✓	✓
7.	No unstructured control flow in programs in higher level languages	R	HR	HR	HR	✓	✓
8.	No automatic type conversion	R	HR	HR	HR	✓	✓

TABLE B.8 – DESIGN AND CODING STANDARDS

Technique/Measure		SIL				QA·C	QA·C++
		1	2	3	4		
3.	Control flow analysis	R	HR	HR	HR	✓	✓
4.	Data flow analysis	R	HR	HR	HR	✓	✓
7.	Symbolic execution	---	---	R	R	✓	✓

SUMMARY

QA·C with the MISRA C Compliance Module and QA·C++ with the MISRA C++ Extended Compliance Module have been certified as “fit for purpose” for achieving compliance with IEC 61508. The time and cost of meeting many of the standard’s requirements associated with development at the software level can be reduced by using these tools. The long history of widespread use of QA·C and QA·C++ in software development demonstrates its suitability for use within industries where safety is critical. QA·C and QA·C++ with MISRA are highly effective tools for any company that needs to achieve IEC 61508 compliance for its products.

ABOUT PRQA

DETECT, ENFORCE AND MEASURE

Since 1985, PRQA has pioneered software coding governance in the automotive, aerospace, transport, finance, medical device and energy industries. Supporting both small start-ups and globally recognized brands, we provide sophisticated code analysis, robust defect detection and enforcement of both bespoke and industry coding standards through functional integrity and application security/safety.

PRQA's industry-leading solutions, QA·C, QA·C++, QA·J and QA·C# offer the most meticulous static analysis of commonly used programming languages. Innovations such as multi-threading and resource analysis (MTR) complement this with refined multi-thread inspection of code streams. Used locally or centrally deployed via the Quality Management System QA·Verify, we enable early find/fix at the desktop and on the server side complete control, visibility and history to the decision maker.

ISO 9001 and TickIT certified.

www.programmingresearch.com