

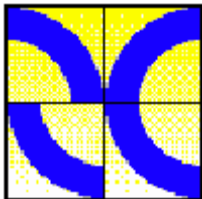
MISRA-C Compliance Matrix Using DAC

by

Chris Hills

Revision 0.1
15 April 2002

Part of the **QuEST** series:- **QA4**



RistanCASE GMBH
Zielackerstrasse 19
8394 Wallisellen-ZH
Switzerland
sales@RistanCASE.com
<http://www.ristancase.com/>



chris@phaedrus.org
quest.phaedrus.org

MISRA-C Compliance Matrix Using DAC

by

Eur Ing **Chris Hills** BSc(Hons), C. Eng., MIEE, MIEEE FRGS

Revision 0.1
15 April 2002

The copies of this paper (and subsequent versions) and any associated power point slides will be available of <http://quest.phaedsys.org/> the authors personal web site or email: Quest@phaedsys.org

This paper will be developed further.

The ART in Embedded Engineering
comes through
Engineering Discipline.

Quality Embedded Software Techniques

QuEST is a series of papers based around the theme of *Quality Embedded Software*. Not for a specific industry or type of work but for all embedded C. It is usually faster, more efficient and surprisingly a lot more fun when things work well.

QuEST Series

- QuEST 0** Introduction & SCIL-Level
- QuEST 1** Embedded C Traps and Pitfalls
- QuEST 2** Embedded Debuggers
- QuEST 3** Advanced Embedded Testing For Fun

Additional Information

- QA1** SCIL-Level
- QA2** Tile Hill Style Guide
- QA3** QuEST-C
- QA4** **MISRA-C Compliance Matrix**

Contents

0	MISRA-C Compliance Matrix.....	7
1	DAC & MISRA-C Checking.....	9
2	A Compliance Matrix Using DAC	11
3	DAC MISRA-C Rule Enforcement	17
4	References	23

0 MISRA-C Compliance Matrix

Version 0.1 15 April 2002

Since its publication in 1998 MISRA-C has gained an unprecedented level of acceptance and use. Not only in the automotive business but in all manner of embedded systems across the world. This is partly because working Engineers, who wanted something they could read easily, wrote MISRA-C with that in mind.

However, the ease of use of the rules in chapter 7 of the guide has somewhat over shadowed the information in the first six chapters. There is a great deal of useful information in these chapters, not least on the use and implementation of the rules

Chapter 5 is entitled "Using MISRA-C" It lists sections entitled:

The Software Engineering Context	(See QuEST Vol. 1)
Style Guide	(See QuEST QA2)
Tool Selection & Validation	(See QuEST Vol. 3 & 4)
Source complexity Metrics	(See QuEST Vol. 1)
Test Coverage	(See QuEST Vol. 4)

It also has a section entitled "Adopting the subset"(5.3). The first sub-section (5.3.1) concerns the Matrix. This is quite simply a chart containing a line for each rule and several columns one for each tool used in the process. Where a rule is enforced or checked this is entered in the box. The Example gives the "message number"

The idea is not to have a tick in every box in the column but at least one tick in each row. That is to say each rule is checked at least once somewhere in the process.

During the work by the MISRA-C Working Group (of which the author is a member) It has been determined that not all the rules are mechanically enforceable. Some are only possible to check by code review. Some rules are mutually exclusive. Therefore, at the current time it is not possible to claim 100% MISRA-C compliance. However the Working Group is working on it.

If you have any comments, spot any errors etc please email the author at quest@phaedsys.org

The charts on the following pages list, for each MISRA-C Rule whether it is **Advisory** or **Required** followed by where the rule is checked. In this Matrix here are four options,:

- The Static Analyser
- The Compiler
- Documentation. (The documented tool chain implementation limits)
- Manual Checking (code review)

The exact level of compliance depends on the tools used. In this case the static analyser is DAC (or Development Assistant for C from RistanCASE (see <http://www.ristancase.com/> and www.hitex.co.uk/ in the UK)

This chart will be updated as DAC and MISRA-C are updated.

Important Note

You should state that you have "Checked the source code for MISRA-C compliance using DAC V4". Not that your code is "MISRA-C Compliant"

Not all the rules are statically checkable and some are a little ambiguous you should be very wary of any tool that claims over 85% testing.

In addition, you need to say (probably in the contracts with the client or customer) which tool you have used for testing. There is little point testing with one tool where the customer or validating body is using another tool. However as long as both parties agree on the tool this should be satisfactory.

1 DAC & MISRA-C Checking

It was never intended that the MISRA-C rules would be 100% static detectable. Rules, for example like Rule 2. In fact, some such as Rule 4 "Provision should be made for appropriate run-time checking" are only testable by manual code review and human judgment (probably a majority vote at that!). Some rules like Rule 14 would require the re-writing of standard library headers and functions. Strict Rule 1 conformance is not possible in most 8 and 16 bit embedded systems anyway. Therefore, no tool can claim 100% MISRA-C compliance.

The "top end" tools costing several orders of magnitude more than DAC suggest that they can hit 85% of the [required] rules where DAC hits 59%. Any tools or vendors claiming much more than 85% of all 127 rules should be regarded with suspicion.

At the time of writing (March 2002) Chris Hills, a Technical Specialist at Hitex UK, who is on the MISRA-C working Group has said that "***Whilst the MISRA-C Guide is a lot better than anything else out there for the average Engineer, it still has a lot of ambiguities that are a problem to tool vendors. It is for this reason that in 2001/2002 the MISRA-C working Group will be clarifying The Rules and producing example test cases for them.***"

This is why there is currently (March 2002) no definitive test suite available or compliance certification for MISRA-C. It is highly unlikely that there will be a test suite before early 2003.

As of December 2001 DAC tests 55% overall and 59% of the "Required" rules, which is a little below average among the current crop of MISRA-C checking, tools. **However, it is quite acceptable for a tool of this type, which is not specifically a MISRA-C test tool.**

	number	of	total	percentage
Required	55	of	93	59%
Advisory	15	of	34	46%
Total	70	of	127	55%

However there are another 23 (25%) required rules and 8 (24%) Advisory rules that are supported with various limitations. This could be argued to give 75% Required cover and 70% advisory cover.

You should state that you have "Tested the Code for MISRA-C Compliance using DAC V4". Not that your code is "MISRA-C Compliant"

As I have explained as not all the rules are statically detectable and some are a little ambiguous, you should be very ware of any tool that claims over 85% testing.

Also you need to say (probably in the contracts with the client or customer) which tool you have used for testing. There is little point testing with one tool where the customer or validating body is using another tool. However as long as both parties agree on the tool this should be satisfactory.

There are several "Required" rules that are not detected by DAC or any static analysis tool that are also unlikely to be seen by the compiler that will have to be manually checked in code review.

Therefore, Code reviews should be specifically tasked with looking for these problems. A Style Guide (such as the Tile Hill Embedded C Style Guide) can be used to make this job easier.

2 A Compliance Matrix Using DAC

The following matrix shows the rules that are checked by DAC. There are also columns for marking the rules that are checked by the compiler, the rules who's status can be discovered by checking the documentation for the tool chain, for example Rule 15, and the rules that can only be checked by manual inspection.

If there is a YES in the DAC, Compiler or Docs column then a "Yes" can be placed in the "Tool Checked" Column". The final column is for the rules that are REQUIRED by MISRA-C but not automatically checked by DAC or the compiler. This should be used to highlight the MISRA rules that will require a deviation or special attention during the manual code review.

MISRA Rule	Required Advisory	Tools				Tool checked	Required NOT CHECKED
		DA-C	Compiler	Docs	Manual		
1	Required	YES				YES	
2	Advisory			YES		NO	
3	Advisory				YES	NO	
4	Advisory				YES	NO	
5	Required	YES				YES	
6	Required	NO			YES	NO	XXXXX
7	Required	NO		YES		NO	XXXXX
8	Required	YES		YES		YES	
9	Required	YES				YES	
10	Advisory				YES	NO	
11	Required	YES		YES		YES	
12	Advisory				YES	NO	
13	Advisory				YES	NO	
14	Required	YES				YES	
15	Advisory				YES	NO	
16	Required				YES	NO	XXXXX
17	Required				YES	NO	XXXXX
18	Advisory	YES				YES	
19	Required	YES			YES	YES	
20	Required	YES				YES	
21	Required	NO			YES	NO	XXXXX
22	Advisory	YES				YES	
23	Advisory	YES				YES	
24	Required	YES				YES	
25	Required	YES				YES	

MISRA Rule	Required Advisory	DA-C	Tools			Tool checked	Required NOT CHECKED
			Compiler	Docs	Manual		
26	Required	YES				YES	
27	Advisory	YES				YES	
28	Advisory	YES				YES	
29	Required	YES			YES	YES	Partial
30	Required	NO			YES	NO	XXXXX
31	Required	NO			YES	NO	XXXXX
32	Required	YES				YES	
33	Required	YES				YES	Partial
34	Required	YES				YES	
35	Required	YES				YES	Partial
36	Advisory	YES				YES	Partial
37	Required	YES			YES	YES	
38	Required	NO				NO	XXXXX
39	Required	YES				YES	
40	Advisory	YES				YES	
41	Advisory	NO		YES		NO	XXXXX
42	Required	YES				YES	
43	Required	NO		YES		NO	XXXXX
44	Advisory	YES				YES	
45	Required	YES				YES	
46	Required	NO				NO	XXXXX
47	Advisory	NO				NO	
48	Advisory	NO				NO	
49	Advisory	NO				NO	
50	Required	YES				YES	

MISRA Rule	Required Advisory	DA-C	Tools			Tool checked	Required NOT CHECKED
			Compiler	Docs	Manual		
51	Advisory	NO		YES	YES	NO	
52	Required	NO			YES	NO	XXXXX
53	Required	YES				YES	Partisl
54	Required	NO			YES	NO	XXXXX
55	Advisory	YES				YES	
56	Required	YES				YES	
57	Required	YES				YES	
58	Required	YES				YES	
59	Required	YES				YES	
60	Advisory	YES				YES	
61	Required	YES				YES	
62	Required	YES				YES	
63	Advisory	NO			YES	NO	
64	Required	YES				YES	
65	Required	NO			YES	NO	XXXXX
66	Advisory	NO			YES	NO	
67	Advisory	NO			YES	NO	
68	Required	YES				YES	
69	Required	YES				YES	
70	Required	YES				YES	Partisl
71	Required	NO			YES	NO	XXXXX
72	Required	NO			YES	NO	XXXXX
73	Required	YES				YES	
74	Required	NO			YES	NO	XXXXX
75	Required	YES				YES	

MISRA Rule	Required Advisory	DA-C	Tools			Tool checked	Required NOT CHECKED
			Compiler	Docs	Manual		
76	Required	YES				YES	
77	Required	NO			YES	NO	XXXXX
78	Required	YES				YES	
79	Required	YES				YES	
80	Required	YES				YES	Parital
81	Advisory	NO			YES	NO	
82	Advisory	YES				YES	
83	Required	NO			YES	NO	XXXXX
84	Required	YES				YES	
85	Advisory	YES				YES	
86	Advisory	NO				NO	
87	Required	NO			YES	NO	XXXXX
88	Required	YES				YES	
89	Required	YES				YES	
90	Required	NO			YES	NO	XXXXX
91	Required	YES			YES	NO	XXXXX
92	Advisory	YES				YES	
93	Advisory	YES				YES	
94	Required	YES				YES	
95	Required	YES				YES	
96	Required	YES				YES	
97	Advisory	YES				YES	
98	Required	YES				YES	
99	Required	YES				YES	
100	Required	YES				YES	

MISRA Rule	Required Advisory	DA-C	Tools			Tool checked	Required NOT CHECKED
			Compiler	Docs	Manual		
101	Advisory	YES				YES	
102	Advisory	YES				YES	
103	Required	NO			YES	NO	XXXXX
104	Required	NO			YES	NO	XXXXX
105	Required	NO			YES	NO	XXXXX
106	Required	NO			YES	NO	XXXXX
107	Required	NO			YES	NO	XXXXX
108	Required	YES				YES	
109	Required	YES				YES	
110	Required	YES				YES	
111	Required	YES				YES	
112	Required	NO			YES	NO	XXXXX
113	Required	YES				YES	
114	Required	YES			YES	NO	XXXXX
115	Required	YES			YES	NO	XXXXX
116	Required	YES				YES	Partial
117	Required	NO			YES	NO	XXXXX
118	Required	NO			YES	NO	XXXXX
119	Required	NO			YES	NO	XXXXX
120	Required	NO			YES	NO	XXXXX
121	Required	NO			YES	NO	XXXXX
122	Required	NO			YES	NO	XXXXX
123	Required	NO			YES	NO	XXXXX
124	Required	NO			YES	NO	XXXXX
125	Required	NO			YES	NO	XXXXX
126	Required	NO			YES	NO	XXXXX
127	Required	NO			YES	NO	XXXXX

3 DAC MISRA-C Rule Enforcement

Environment

- 1 (req) Fully supported.
- 2 (adv) Not statically checkable. DAC Symbol Analysis deals only with source code.
- 3 (adv) Currently not supported.
- 4 (adv) Not statically checkable.

Character Sets

- 5 (req) Fully supported.**
- 6 (req) Not statically checkable. ISO 10646-1 defines an international standard for mapping character sets to numeric values.
- 7 (req) Not supported. DAC Symbol Analysis does not recognize trigraphs.
- 8 (req) Fully supported.**

Comments

- 9 (req) Fully supported.**
- 10 (adv) Currently not supported.

Identifiers

- 11 (req) Fully supported, as far as source code is concerned.**
- 12 (adv) Currently not supported.

Types

- 13 (adv) Currently not supported.
- 14 (req) Fully supported.**
- 15 (adv) Not statically checkable. (In compiler documentation)
- 16 (req) Currently not supported.
- 17 (req) Currently not supported.

Constants

- 18 (adv) Fully supported.**
- 19*(req) Fully supported.**

Declarations and Definitions

- 20 (req)** **Fully supported.**
- 21 (req) Currently not supported.
- 22 (adv)** **Fully supported.**
- 23 (adv)** **Fully supported.**
- 24 (req)** **Fully supported.**
- 25 (req)** **Fully supported.**
- 26 (req)** **Fully supported.**
- 27 (adv)** **Fully supported.**
- 28 (adv)** **Fully supported.**
- 29 (req) Supported in type and variable declarations. Initialises are not checked whether they match the structure declared for tag.

Initialisation

- 30 (req) Not supported.
- 31 (req) Currently not supported.
- 32 (req)** **Fully supported.**

Operators

- 33(req) Supported. The user defined function call is always considered to have side effects.
- 34 (req)** **Fully supported.**
- 35 (req) Supported in the control expressions of if, while, do-while and for statements.
- 36(adv) Supported. Uses of bitwise operators are reported in if, while, do-while and for statements.
- 37 (req) Supported. See Endnote 1.
- 38 (req) Currently not supported.
- 39 (req) Supported. See Endnote 1.
- 40(adv) Supported. The user defined function call is always considered to have side effects.
- 41 (adv) Not statically checkable.
- 42* (req)** **Fully supported.**

Conversions

- 43 (req) Not supported. DAC Symbol analysis does not give this kind of information.
- 44(adv) Supported. See Endnote 1.
- 45 (req) Supported. See Endnote 1.

Expressions

- 46 (req) Currently not supported.
- 47 (adv) Currently not supported.
- 48 (adv) Currently not supported.
- 49 (adv) Currently not supported.
- 50 (req) Supported. See Endnote 1.
- 51 (adv) Not supported.

Control Flow

- 52 (req) Currently not supported.
- 53 (req) Supported. The user defined function call is always considered to have side effects.
- 54 (req) Currently not supported.
- 55 (adv) **Fully supported.**
- 56 (req) **Fully supported.**
- 57 (req) **Fully supported.**
- 58 (req) **Fully supported.**
- 59 (req) **Fully supported.**
- 60 (adv) **Fully supported.**
- 61 (req) **Fully supported.**
- 62 (req) **Fully supported.**
- 63 (adv) Not supported.
- 64 (req) **Fully supported.**
- 65 (req) Currently not supported.
- 66 (adv) Not statically checkable.
- 67 (adv) Currently not supported.

Functions

- 68 (req) **Fully supported.**
- 69 (req) **Fully supported.**
- 70 (req) Supported. Only direct recursion is reported.
- 71 (req) Not supported.
- 72 (req) Not supported.
- 73 (req) **Fully supported.**
- 74 (req) Not supported.
- 75 (req) **Fully supported.**
- 76 (req) **Fully supported.**
- 77 (req) Not supported.
- 78 (req) **Fully supported.**
- 79 (req) **Fully supported.**
- 80 (req) Supported. See Endnote 1.
- 81 (adv) Not supported.
- 82 (adv) Supported.

- 83 (req) Currently not supported.
- 84 (req) **Fully supported.**
- 85 (adv) **Fully supported.**
- 86 (adv) Not supported.

Pre-processing Directives

- 87 (req) Currently not supported.
- 88 (req) **Fully supported.**
- 89 (req) **Fully supported.**
- 90 (req) Not supported.
- 91 (req) **Fully supported.**
- 92 (adv) **Fully supported.**
- 93 (adv) **Fully supported.**
- 94 (req) **Fully supported.**
- 95 (req) **Fully supported.**
- 96 (req) **Fully supported.**
- 97 (adv) **Fully supported.**
- 98 (req) **Fully supported.**
- 99 (req) **Fully supported.**
- 100 (req) **Fully supported.**

Pointers and Arrays

- 101 (adv) Supported. See Endnote 1.
- 102 (adv) Supported. See Endnote 1.
- 103 (req) Not statically checkable.
- 104 (req) Not supported.
- 105 (req) Currently not supported.
- 106 (req) Currently not supported.
- 107 (req) Not statically checkable.

Structures and Unions

- 108 (req) **Fully supported.**
- 109 (req) **Fully supported.**
- 110 (req) **Fully supported.**
- 111 (req) **Fully supported.**
- 112 (req) Currently not supported.
- 113 (req) **Fully supported.**

Standard Libraries

- 114 (req) Currently not supported.
- 115 (req) Currently not supported.
- 116 (req) DAC analyzes standard libraries headers and reports MISRA warnings for them if Options/Analysis for Symbols/Warnings/Disabled for all headers and Options/Analysis for Symbols/Warnings/Disabled for standard library headers are both unchecked.
- 117 (req) Not statically checkable.
- 118 (req) Currently not supported.
- 119 (req) Currently not supported.
- 120 (req) Currently not supported.
- 121 (req) Currently not supported.
- 122 (req) Currently not supported.
- 123 (req) Currently not supported.
- 124 (req) Currently not supported.
- 125 (req) Currently not supported.
- 126 (req) Currently not supported.
- 127 (req) Currently not supported.

Endnote 1: DAC Symbol Analysis has been designed primarily to extract symbol usage information for cross referencing purposes. Therefore, it can infer the type of an expression or some part of it only if that type is directly derived from some symbol used in it. That includes e.g. (a), a.x, *a, a+b, a[2] and excludes e.g. func(), (x)?(y):(z) etc.

4 References

COX B, *Software ICs and Objective C, Interactive Programming Environments*, McGraw Hill, 1984

Hatton L, *Safer C*, McGraw-Hill(1994)

Hills C A, *Embedded Debuggers* Chris Hills & Mike Beach, Hitex (UK) Ltd. April 1999 <http://www.hitex.co.uk> & quest.phaedsys.org

Hills CA & Beach M, Hitex, SCIL-Level A paper project managers, team leaders and Engineers on the classification of embedded projects and tools. Useful for getting accountants to spend money Download from www.scil-level.org

Home Office Reforming the Law on Involuntary Manslaughter : The governments Proposals <http://www.homeoffice.gov.uk/consult/lcbill.pdf>

[Johnson] S. C. Johnson, '*Lint, a Program Checker,*' in *Unix Programmer's Manual*, Seventh Edition, Vol. 2B, M. D. McIlroy and B. W. Kernighan, eds. AT&T Bell Laboratories: Murray Hill, NJ, 1979.

Kernighan Brian W, *The Practice of Programming*. Addison Wesley 1999

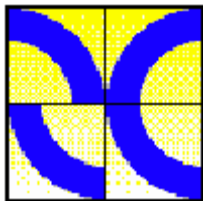
Koenig A C *Traps and Pitfalls*, Addison Wesley, 1989

K&R *The C programming Language* 2nd Ed., Prentice-Hall, 1988

MISRA Guidelines For The Use of The C Language in Vehicle Based Software. 1998 From <http://www.misra.org.uk/> and <http://www.hitex.co.uk/>

[Pressman] *Software Engineering A Practitioners Approach*. 3rd Ed McGrawHill 1992 ISBN 0-07-050814-3

Ritchie D. M. *The Development of the C Language* Bell Labs/Lucent Technologies Murray Hill, NJ 07974 USA 1993 available from his web site <http://cm.bell-labs.com/cm/cs/who/dmr/index.htm>. This is well worth reading.



RistanCASE GMBH
Zielackerstrasse 19
8394 Wallisellen-ZH
Switzerland
sales@RistanCASE.com
<http://www.ristancase.com/>



chris@phaedrus.org
quest.phaedrus.org