

# **MISRA-C Compliance Matrix** **Using PC Lint**

by

**Chris Hills**

**Revision 0.2**

**15 April 2002**

Part of the [QuEST](#) series:- [QA4](#)



[chris@phaedsys.org](mailto:chris@phaedsys.org)

[quest.phaedsys.org](http://quest.phaedsys.org)



# MISRA-C Compliance Matrix Using PC Lint

by

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

**Revision 0.2**  
**15 April 2002**

The copies of this paper (and subsequent versions) and any power point slides will be available or <http://quest.phaedsys.org> the authors personal web site. [Quest@phaedsys.org](mailto: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** PC-Lint MISRA-C Compliance Matrix



# Contents

0	MISRA-C Compliance Matrix.....	7
1	PC-Lint & MISRA-C Checking.....	9
2	A Compliance Matrix Using PC-Lint.....	11
3	PC-Lint MISRA-C Rule checking statistics.....	17
4	PC-LINT MISRA-C Rule Enforcement.....	19
5	References.....	25



## 0 MISRA-C Compliance Matrix

### **Version 0.2 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 due to the fact that working Engineers who wanted something they could read easily wrote it 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 <mailto: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. PC-Lint from Gimpel ([www.gimpel.com](http://www.gimpel.com)/in the USA and [www.hitex.co.uk](http://www.hitex.co.uk)/in Europe)

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

### **Important Note**

**You should state that you have "Checked the source code for MISRA-C compliance using PC-Lint V8". 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 PC-Lint & 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. Rule 1 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 PC-Lint suggest that they can hit 85% of the [required] rules where PC-Lint hits 82%. Any tools or vendors claiming much more than 85% of all 127 rules should be regarded with suspicion. Like PC-Lint, any tool you use should be able to show you **how** it is testing each rule. Do not settle for anything less.

At the time of writing (April 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 (April 2002 ) no definitive test suite available or compliance certification for MISRA-C. It is highly unlikely that there will be either before early 2003.

In recent tests, PC-Lint came out the fastest tool for statically analysing a large real world program



## 2 A Compliance Matrix Using PC-Lint

The following matrix shows the rules that are checked by PC-Lint. 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 PC-Lint, 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 PC-Lint 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
		PC-Lint	Compiler	Docs	Manual		
1	Required	YES				YES	
2	Advisory	NO			YES	NO	
3	Advisory	YESX				YES	X+
4	Advisory	NO			YES	NO	
5	Required	YES				YES	
6	Required	XXXXX	YES			YES	
7	Required	YES				YES	
8	Required	XXXXX	YES			YES	
9	Required	YES				YES	
10	Advisory	NO			YES	NO	
11	Required	YES				YES	
12	Advisory	YES				YES	
13	Advisory	YES				YES	
14	Required	YES				YES	
15	Advisory	NO			YES	NO	
16	Required	NO			YES	NO	XXXX
17	Required	YES				YES	
18	Advisory	YES				YES	
19	Required	NO			YES	NO	XXXX
20	Required	YES				YES	
21	Required	YESX				YES	X+
22	Advisory	NO				NO	
23	Advisory	YES				YES	
24	Required	YES				YES	
25	Required	YES				YES	

Yes+ == no Specific MISRA-Rule Message

\*\*\*\*\* == REQUIRED RULE Not checked by PC-Lint Required manual checking

MISRA Rule	Required Advisory	PC-Lint	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	
30	Required	YES				YES	
31	Required	YES				YES	
32	Required	YES+				YES	X+
33	Required	YES+				YES	X+
34	Required	NO			YES	NO	XXXXX
35	Required	YES				YES	
36	Advisory	NO				NO	
37	Required	YES			YES	YES	
38	Required	YES				YES	
39	Required	YES				YES	
40	Advisory	YES+				YES	
41	Advisory	NO			YES	NO	
42	Required	YES				YES	
43	Required	YES				YES	
44	Advisory	YES+				YES	
45	Required	YES				YES	
46	Required	YES				YES	
47	Advisory	YES+				YES	
48	Advisory	YES+				YES	
49	Advisory	YES+				YES	
50	Required	YES				YES	

Yes+ == no Specific MISRA-Rule Message

\*\*\*\*\* == REQUIRED RULE Not checked by PC-Lint Required manual checking

MISRA Rule	Required Advisory	PC-Lint	Tools			Tool checked	Required NOT CHECKED
			Compiler	Docs	Manual		
51	Advisory	YES				YES	
52	Required	YES				YES	
53	Required	YES				YES	
54	Required	YES				YES	
55	Advisory	YES+				YES	
56	Required	YES				YES	
57	Required	YES+				YES	X+
58	Required	YES+				YES	X+
59	Required	YES+				YES	X+
60	Advisory	YES				YES	
61	Required	YES				YES	
62	Required	YES				YES	
63	Advisory	YES+				YES	
64	Required	YES				YES	
65	Required	YES+				YES	X+
66	Advisory	NO			YES	NO	
67	Advisory	NO			YES	NO	
68	Required	YES+				YES	X+
69	Required	YES				YES	
70	Required	NO	NOTE1			NO	XXXXX
71	Required	YES				YES	
72	Required	YES				YES	
73	Required	YES+				YES	
74	Required	NO				NO	XXXXX
75	Required	YES				YES	

Yes+ == no Specific MISRA-Rule Message

\*\*\*\*\* == REQUIRED RULE Not checked by PC-Lint Required manual checking

Note 1

Some embedded compilers do not permit recursion.

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

Yes+ == no Specific MISRA-Rule Message

\*\*\*\*\* == REQUIRED RULE Not checked by PC-Lint Required manual checking

Rule	Advisory	PC-Lint	Compiler	Docs	Manual	checked	NOT CHECKED
101	Advisory	NO			YES	NO	
102	Advisory	NO			YES	NO	
103	Required	YES				YES	
104	Required	NO			YES	NO	XXXXX
105	Required	NO			YES	NO	XXXXX
106	Required	YES				YES	
107	Required	YES				YES	
108	Required	YES				YES	
109	Required	NO	YES		YES	YES	XXXXX
110	Required	YES+				YES	X+
111	Required	YES				YES	
112	Required	YES				YES	
113	Required	NO			YES	NO	XXXXX
114	Required	YES				YES	
115	Required	NO			YES	NO	XXXXX
116	Required	NO			YES	NO	XXXXX
117	Required	YESX+				YESX++	X++
118	Required	YES				YES	
119	Required	YES				YES	X+
120	Required	YES				YES	X+
121	Required	YES				YES	
122	Required	YES				YES	
123	Required	YES				YES	
124	Required	YES				YES	
125	Required	YES				YES	
126	Required	YES				YES	
127	Required	YES				YES	

Yes++ can be checked but requires specific setting up with Pc-Lint

Yes+ == no Specific MISRA-Rule Message

\*\*\*\*\* == REQUIRED RULE Not checked by PC-Lint Required manual checking



### 3 PC-Lint MISRA-C Rule checking statistics

As of April 2002 PC-Lint tests 79% overall and 88% of the "Required" rules, which is about average among the current crop of MISRA-C checking, tools.

	<b>#</b>	<b>of</b>	<b>total</b>	<b>percentage</b>
<b>Required</b>	81	of	93	87%
<b>Advisory</b>	19	of	34	56%
<b>Total</b>	<b>100</b>	<b>of</b>	<b>127</b>	<b>79%</b>

There are several "Required" rules, 12 by my count, that are not detected by PC-Lint or likely 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.

Note not matter what automated tools are used a manual code review is still required.



## 4 PC-LINT MISRA-C Rule Enforcement

The text below is the standard list of those checked, partially checked, not checked and uncheck able..

### Environment

- 1 (req) Fully supported.**
- 2 (adv) Not statically checkable
- 3 (adv) Partially 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) Fully supported.**
- 8 (req) Partially supported.**

### Comments

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

### Identifiers

- 11 (req) Partially supported.**
- 12 (adv) Fully supported.**

### Types

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

## **Constants**

**18 (adv) Fully supported.**

**19 (req) Fully supported.**

## **Declarations and Definitions**

**20 (req) Fully supported.**

**21 (req) Fully supported.**

**22 (adv) Currently not 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) Fully supported.**

## **Initialisation**

**30 (req) Fully supported.**

**31 (req) Fully supported.**

**32 (req) Fully supported.**

## **Operators**

**33 (req) Fully supported.**

**34 (req) Currently not supported.**

**35 (req) Fully supported.**

**36 (adv) Fully supported.**

**37 (req) Fully supported.**

**38 (req) Fully supported.**

**39 (req) Fully supported.**

**40 (adv) Fully supported.**

**41 (adv) Not statically checkable.**

**42 (req) Fully supported.**

## **Conversions**

**43 (req) Fully supported.**

**44 (adv) Fully supported.**

**45 (req) Fully supported.**

## **Expressions**

- 46 (req) Fully supported.**
- 47 (adv) Fully supported.**
- 48 (adv) Fully supported.**
- 49 (adv) Fully supported.**
- 50 (req) Fully supported.**
- 51 (adv) Fully supported.**

## **Control Flow**

- 52 (req) Fully supported.**
- 53 (req) Fully supported.**
- 54 (req) Fully 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) Fully supported.**
- 64 (req) Fully supported.**
- 65 (req) Fully supported.**
- 66 (adv) Not statically checkable.
- 67 (adv) Currently not supported.

## **Functions**

- 68 (req) Fully supported.**
- 69 (req) Fully supported.**
- 70 (req) Currently not supported.
- 71 (req) Fully supported.**
- 72 (req) Fully supported.**
- 73 (req) Fully supported.**
- 74 (req) Currently not supported.
- 75 (req) Fully supported.**
- 76 (req) Fully supported.**
- 77 (req) Fully supported.**
- 78 (req) Fully supported.**

- 79 (req) Fully supported.**
- 80 (req) Fully supported.**
- 81 (adv) Currently not supported.
- 82 (adv) Currently not supported.
- 83 (req) Fully supported.**
- 84 (req) Fully supported.**
- 85 (adv) Currently not supported.
- 86 (adv) Currently not supported.

### **Pre-processing Directives**

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

### **Pointers and Arrays**

- 101 (adv) Currently not supported.
- 102 (adv) Currently not supported.
- 103 (req) Fully supported.**
- 104 (req) Currently not supported.
- 105 (req) Currently not supported.
- 106 (req) Fully supported.**
- 107 (req) Fully supported**

.

### **Structures and Unions**

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

**112 (req) Fully supported.**  
113 (req) Currently not supported.

### **Standard Libraries**

**114 (req) Fully supported.**  
115 (req) Currently not supported.  
**116 (req) Fully supported.**  
**117 (req) Fully supported.**  
**118 (req) Fully supported.**  
**119 (req) Fully supported.**  
**120 (req) Fully supported.**  
**121 (req) Fully supported.**  
**122 (req) Fully supported.**  
**123 (req) Fully supported.**  
**124 (req) Fully supported.**  
**125 (req) Fully supported.**  
**126 (req) Fully supported.**  
**127 (req) Fully supported.**



## 5 References

Beach, M. *Hitex C51 Primer* 3<sup>rd</sup> Ed, Hitex UK, 1995, <http://www.hitex.co.uk>

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](http://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](http://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* 2<sup>nd</sup> 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*. 3<sup>rd</sup> 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.



[quest.phaedsys.org/](http://quest.phaedsys.org/)  
[chris@phaedsys.org](mailto:chris@phaedsys.org)