MISRA C Rules

The following is a summary of the MISRA C rules. This document is not a definitive list these rules, which are only and completely defined in "**MISRA Guidelines for the use of the C language in vehicle based software**".

Environment

1 (req): All code shall conform to ISO 9899 standard, with no extensions permitted.

2 (adv): Code written in languages other than C should only be used if there is a defined interface standard for object code to which the compiler/assembler for both languages conform

3 (adv): Assembler language functions that are called from C should be written as C functions containing only in-line assembly language, and in-line assembly language should not be embedded in normal C code

4 (adv): Provision should be made for appropriate run-time checking

Character Sets

5 (req): Only those characters and escape sequences that are defined in the IOS C standard shall be used

6 (req): Values of character types shall be restricted to a defined and documented subset of ISO 10646-1

7 (req): Tri-graphs shall not be used

8 (req): Multibyte characters and wide string literals shall not be used

Comments

9 (req): Comments shall not be nested

10 (adv): Sections of code should not be commented out

Identifiers

11 (req): Identifiers shall not rely on the significance of more than 31 characters.
Compiler/linker shall check to ensure that 31 char significance and case sensitivity are supported for external identifiers

12 (adv): Identifiers in different namespace shall not have the same spelling.
Structure members are an exception

Types

13 (adv): The basic types *char*, *int*, *short*, *long*, *double* and *float* should not be used. Specific-length equivalents should be *typedef*'d for the specific compiler, and these names used in the code

14 (req): The type *char* shall always be declared as *unsigned char* or *signed char.*
See rule 13

15 (adv): Floating point implementations should comply with a defined floating point standard

16 (req): The underlying bit representation of floating point numbers shall not be used in any way by the programmer

17 (req): *typedef* names shall not be reused

Constants

18 (adv): Numeric constants should be suffixed to indicate type if possible

19 (req): Octal constants shall not be used
Zero is okay

Declarations and Definitions

20 (req): All object and function identifiers shall be declared before use

21 (req): Identifiers in an inner scope shall not use the same name as an identifier in an outer scope, and therefore hide that identifier

22 (adv): Declaration of object should be at function scope unless a wider scope is necessary

23 (adv): All declarations at file scope should be static if possible

24 (req): Identifiers shall not simultaneously have both internal and external linkage in the same translation unit

25 (req): An identifier with an external linkage shall have exactly one external definition

26 (req): If objects are declared more than once they shall have compatible declarations

27 (adv): External objects should not be declared in more than one file

28 (adv): The register storage class specifier should not be used

29 (req): The use of a tag shall agree with its declaration

Initialisation

30 (req): All automatic variables shall have a value assigned to them before use

31 (req): Braces shall be used to indicate and match the structure in the non-zero initialisation of arrays and structures

32 (req): In an enumerator list the = construct shall not be used to explicitly initialise members other than the fist unless it is used to initialise all items

Operators

33 (req): The right hand operand of a && or || shall not contain side effects

34 (req): The operands of a logical && or || shall be primary expressions
A single identifier, constant or parenthesised expression

35 (req): Assignment operators shall not be used in expressions that return Boolean values

36 (adv): Logical operators shall not be confused with bitwise operators

37 (req): Bitwise operations shall not be performed on signed integer types

38 (req): The right hand operand of a shift operator shall lie between zero and one less the width in bits of the left hand operand

39 (req): The unary minus operator shall not be applied to an unsigned expression

40 (adv): The *sizeof* operator should not be used on expressions that contain side effects

41 (adv): The implementation of integer division in the chosen compiler should be determined, documented and taken into account

42 (req): The comma operator shall not be used, except in the control expression of a for loop

Conversions

43 (req): Implicit conversions that might result in a loss of information shall not be used

44 (adv): Redundant explicit cast should not be used

45 (req): Type casting from any type to or from pointers shall not be used

Expressions

46 (req): The value of an expression shall be the same under any order of evaluation that the standard permits

47 (adv): No dependence should be placed on C's precedence rules

48 (adv): Mixed precision arithmetic should use explicit casting to generate the desired result

49 (adv): Tests of a value against zero should be explicit, unless the operant is effectively Boolean

50 (req): Floating point variables shall not be tested for exact inequality or inequality

51 (adv): Evaluation of constant unsigned integer expression should not lead to wrap-around

Control Flow

52 (req): There shall be no unreachable code

53 (req): All non-null statements shall have a side effect

54 (req): A null statement shall only appear on a line by its self, and shall not have any other text on the same line

55 (adv): Label should not be used except in *switch* statements

56 (req): The *goto* statement shall not be used

57 (req): The *continue* statement shall not be used

58 (req): The *break* statement shall not be used, except to terminate the cases of a *switch* statement

59 (req): The statements forming the body of an *if*, *else if*, *else*, *while*, *do … while* or *for* statement shall always be enclosed in braces

60 (adv): All *if*, *else if* constructs should contain a final *else* clause

61 (req): Every non-empty *case* clause in a *switch* statement shall be terminated with a *break* statement

62 (req): All *switch* statements shall contain a final *default* clause

63 (adv): A *switch* expression should not represent a Boolean value

64 (req): Every *switch* statement should have at least one *case*

65 (req): Floating point variables shall not be used as loop counters

66 (adv): Only expression concerned with loop control should appear within a *for* statement

67 (adv): Numeric variables being used within a *for* loop for iteration counting should not be modified in the body of the loop

Functions

68 (req): Functions shall always be declared at file scope

69 (req): Functions with a variable number of arguments shall not be used

70 (req): Functions shall not call themselves directly or indirectly

71 (req): Functions shall always have prototype declarations and the prototype shall be visible at both the function definition and call

72 (req): For each function parameter the type given and definition shall be identical, and the return type shall be identical

73 (req): Identifiers shall either be given for all parameters in a prototype declaration, or none

74 (req): If identifiers are given for any parameters, then the identifiers used in declaration and definition shall be identical

75 (req): Every function shall have an explicit return type

76 (req): Functions with no parameter list shall be declared with parameter type *void*

77 (req): The unqualified type of parameters passed to a function shall be compatible with the unqualified expected types defined in the function prototype

78 (req): The number of parameters passed to a function shall match the function prototype

79 (req): The value returned by *void* functions shall not be used

80 (req): Void expressions shall not be passed as function parameters

81 (adv): *const* qualification should be used on function parameters that are passed by reference, where it is intended that the function will not modify the parameter

82 (adv): A function should have a single exit point

83 (req): For functions that do not have a *void* return type:

- i) There shall be one *return* statement for every exit branch (including the end of the program)
- ii) Each *return* shall have an expression
- iii) The *return* expression shall match the return type

84 (req): For functions with *void* return type, *return* statements shall not have an expression

85 (adv): Functions called with no parameters should have empty parentheses

86 (adv): If a function returns error information then that information should be tested

Pre-Processing Directives

87 (req): *#include* statements in a file shall only be preceded by other pre-processor directives or comments

88 (req): Non-standard characters shall not occur in header file names in *#include* directives

89 (req): The *#include* directive shall be followed by either a <filename> of "filename" sequence

90 (req): C macros shall only be used for symbolic constants, function like macros, type qualifiers and storage class specifiers

91 (req): Macros shall not be *#define*'d and *#undef*'d within a block

92 (adv): *#undef* should not be used

93 (adv): A function should be used in preference to a function-like macro

94 (req): A function-like macro shall not be 'called' without all of its arguments

95 (req): Arguments to a function-like macro shall not contain tokens that look like pre-processor directives

96 (req): In the definition of a function-like macro the whole definition, and each instance of a parameter, shall be enclosed in parenthesis

97 (adv): Identifiers in pre-processor directives should be defined before use

98 (req): There shall be at most one occurrence of the # or ## pre-processor operators in a single macro definition

99 (req): All use of *#pragma* directive shall be documented and explained

100 (req): The defined pre-processor operator shall only be used in one of the two standard forms

Pointers and Arrays

101 (adv): Pointer arithmetic shall not be used

102 (adv): No more than 2 levels of pointer indirection should be used

103 (req): Relational operators shall not be applied to pointer types except where both operands are of the same type and point to the same array, structure or union

104 (req): Non-constant pointers to functions shall not be used

105 (req): All functions pointed to by a single pointer to a function shall be identical in the number and type of parameters and the return type

106 (req): The address of an object with automatic storage shall not be assigned to an object which may persist after the object has ceased to exist

107 (req): The null pointer shall not be dereferenced

Structures and Unions

108 (req): In the specification of a structure or union type, all members of the structure or union shall be fully specified

109 (req): Overlapping variable storage shall not be used

110 (req): Unions shall not be used to access the sub-parts of larger data types

111 (req): Bit fields shall only be defined to be one of type *unsigned int* or *signed int*

112 (req): Bit fields of type *signed int* shall be at least 2 bits long

113 (req): All members of a structure or union shall be named and shall only be access with their name

Standard Libraries

114 (req): Reserved words and the standard library function names shall not be redefined or undefined

115 (req): Standard library names shall not be reused

116 (req): All libraries used in production code shall be written to comply with the provisions of "MISRA Guidelines for the use of the C language in vehicle based software", and shall have been subject to appropriate validation

117 (req): The validity of values passed to library functions shall be checked

118 (req): Dynamic heap memory allocation shall not be used

119 (req): The error indicator *errno* shall not be used

120 (req): The macro *offsetof*, in library *<stddef.h>*, shall not be used

121 (req): *<local.h>* and the *setlocale* function shall not be used

122 (req): The setjmp macro and the longjmp function shall not be used

123 (req): The signal handling facilities of *<signal.h>* shall not be used

124 (req): The input/output library *<stdio.h>* shall not be used in production code

125 (req): The library functions *atof*, *atoi* and *atol* from library *<stslib.h>* shall not be used

126 (req): The library functions *abort*, *exit*, *getenv* and *system* from library *<stdlib.h>* shall not be used

127 (req): The time handling functions of library *<time.h>* shall not be used