MISRA C 2004

The Motor Industry Software Reliability Association (MISRA) guidelines for C published in 2004

MISRA Mission Statement: To provide assistance to the automotive industry in the application and creation within vehicle systems of safe and reliable software.

MISRA, The Motor Industry Software Reliability Association, is a collaboration between vehicle manufacturers, component suppliers and engineering consultancies which seeks to promote best practice in developing safety-related electronic systems in road vehicles and other embedded systems. To this end MISRA publishes documents that provide accessible information for engineers and management, and holds events to permit the exchange of experiences between practitioners.

The first edition of the MISRA C Standard was published in 1998, the second edition was released in 2004 with many substantial changes.

www.misra.org.uk

© MIRA Limited, 2004, 2008.

	All Rules	Advisory	Required
Understand % Coverage	76%	80%	75%
Understand Coverage	108	16	92
Total Rules	142	20	122

Checks

Check ID	Check Name	Supported	Category
MISRA04_1.1	1.1 All code shall conform to ISO/IEC 9899:1990 "Programming languages — C", amended and corrected by ISO/IEC 9899/ COR1:1995, ISO/IEC 9899/AMD1:1995, and ISO/IEC 9899/COR2:1996	No	Required
MISRA04_1.2	1.2 No reliance shall be placed on undefined or unspecified behaviour	No	Required
MISRA04_1.3	1.3 Multiple compilers and/or languages shall only be used if there is a common defined interface standard for object code to which the languages/compilers/ assemblers conform	No	Required
MISRA04_1.4	1.4 The compiler/linker shall be checked to ensure that 31 character significance and case sensitivity are supported for external identifiers	No	Required
MISRA04_1.5	1.5 Floating-point implementations should	No	Advisory

Check ID	Check Name	Supported	Category
	comply with a defined floating-point standard		
MISRA04_2.1	2.1 Assembly language shall be encapsulated and isolated.	Yes	Required
MISRA04_2.2	2.2 C99 / Comments	Yes	Required
MISRA04_2.3	2.3 The character sequence /* shall not be used within a comment.	Yes	Required
MISRA04_2.4	2.4 Sections of code should not be "commented out"	Yes	Advisory
MISRA04_3.1	3.1 All usage of implementation-defined behaviour shall be documented	No	Required
MISRA04_3.2	3.2 The character set and the corresponding encoding shall be documented	No	Required
MISRA04_3.3	3.3 The implementation of integer division in the chosen compiler should be determined, documented and taken into account	No	Advisory
MISRA04_3.4	3.4 All uses of the #pragma directive shall be documented and explained	Yes	Required
MISRA04_3.5	3.5 The implementation defined behaviour and packing of bitfields shall be documented if being relied upon	No	Required
MISRA04_3.6	3.6 All libraries used in production code shall be written to comply with the provisions of this document, and shall have been subject to appropriate validation	No	Required
MISRA04_4.1	4.1 Only those escape sequences that are defined in the ISO C standard shall be used	Yes	Required
MISRA04_4.2	4.2 Trigraphs shall not be used	Yes	Required
MISRA04_5.1	5.1 Long Identifier Name Significance	Yes	Required
MISRA04_5.2	5.2 Shadowed Identifiers	Yes	Required
MISRA04_5.3	5.3 A typedef name shall be a unique identifier.	Yes	Required
MISRA04_5.4	5.4 A tag name shall be a unique identifier	Yes	Required
MISRA04_5.5	5.5 No object or function identifier with static storage duration should be reused	Yes	Advisory
MISRA04_5.6	5.6 Identifier Reuse in Multiple C Name Spaces	Yes	Advisory
MISRA04_5.7	5.7 Identifier Reuse	Yes	Advisory
MISRA04_6.1	6.1 The plain char type shall only be used for the storage and use of character values	Yes	Required
MISRA04_6.2	6.2 Signed char and unsigned char type shall	Yes	Required

Check ID	Check Name	Supported	Category
	only be used for the storage and use of numeric values		
MISRA04_6.3	6.3 Typedefs that indicate size and signedness should be used in place of the basic numerical types	Yes	Advisory
MISRA04_6.4	6.4 Bit fields shall only be defined to be of type unsigned int or signed int.	Yes	Required
MISRA04_6.5	6.5 Bit fields of signed type shall be at least 2 bits long.(Fuzzy parser)	Yes	Required
MISRA04_7.1	7.1 Octal constants (other than zero) and octal escape sequences shall not be used.	Yes	Required
MISRA04_8.1	8.1 Functions shall have prototype declarations and the prototype shall be visible at both the function definition and call.	No	Required
MISRA04_8.2	8.2 Whenever an object or function is declared or defined, its type shall be explicitly stated	Yes	Required
MISRA04_8.3	8.3 For each function parameter the type given in the declaration and definition shall be identical, and the return types shall also be identical	Yes	Required
MISRA04_8.4	8.4 If objects or functions are declared more than once their types shall be compatible.	No	Required
MISRA04_8.5	8.5 No definitions of objects or functions in a header file	Yes	Required
MISRA04_8.6	8.6 Functions shall be declared at file scope	Yes	Required
MISRA04_8.7	8.7 Objects shall be local if only accessed from one function	Yes	Required
MISRA04_8.8	8.8 An external object or function shall be declared in one and only one file	Yes	Required
MISRA04_8.9	8.9 An identifier with external linkage shall have exactly one external definition	Yes	Required
MISRA04_8.10	8.10 prefer internal linkage over external whenever possible	Yes	Required
MISRA04_8.11	8.11 Use the static keyword for internal linkage	Yes	Required
MISRA04_8.12	8.12 Array Size Missing	Yes	Required
MISRA04_9.1	9.1 All automatic variables shall have been assigned a value before being used	No	Required
MISRA04_9.2	9.2 Braces shall be used to indicate and match the structure in the non-zero	Yes	Required

Check ID	Check Name	Supported	Category
	initialisation of arrays and structures		
MISRA04_9.3	9.3 = construct in enumerator list shall only be used on either the first item alone, or all items explicitly.	Yes	Required
MISRA04_10.1	10.1 The value of an expression of integer type shall not be implicitly converted to a different underlying type if: it is not a conversion to a wider integer type of the same signedness, or (a) the expression is complex, or (b) the expression is not constant and is a function argument, or (c) the expression is not constant and is a return expression (d)	No	Required
MISRA04_10.2	10.2 The value of an expression of floating type shall not be implicitly converted to a different type if: it is not a conversion to a wider floating type, or (a) the expression is complex, or (b) the expression is a function argument, or (c) the expression is a return expression (d)	No	Required
MISRA04_10.3	10.3 The value of a complex expression of integer type shall only be cast to a type of the same signedness that is no wider than the underlying type of the expression.	No	Required
MISRA04_10.4	10.4 The value of a complex expression of floating type shall only be cast to a floating type that is narrower or of the same size.	No	Required
MISRA04_10.5	10.5 If the bitwise operators ~ and << are applied to an operand with an underlying type of unsigned char or unsigned short, the result shall be immediately cast to the underlying type of the operand	Yes	Required
MISRA04_10.6	10.6 A U suffix shall be applied to all constants of unsigned type	Yes	Required
MISRA04_11.1	11.1 Conversions shall not be performed between a pointer to a function and any type other than an integral type.	No	Required
MISRA04_11.2	11.2 Conversions shall not be performed between a pointer to object and any type other than an integral type, another pointer to object type or a pointer to void.	No	Required
MISRA04_11.3	11.3 A cast should not be performed between a pointer type and an integral type.	No	Advisory

Check ID	Check Name	Supported	Category
MISRA04_11.4	11.4 A cast should not be performed between a pointer to object type and a different pointer to object type	Yes	Advisory
MISRA04_11.5	11.5 A cast shall not be performed that removes any const or volatile qualification from the type addressed by a pointer	Yes	Required
MISRA04_12.1	12.1 Limited dependence should be placed on C's operator precedence rules in expressions	Yes	Advisory
MISRA04_12.2	12.2 The value of an expression shall be the same under any order of evaluation that the standard permits	Yes	Required
MISRA04_12.3	12.3 The sizeof operator shall not be used on expressions that contain side effects	Yes	Required
MISRA04_12.4	12.4 The right-hand operand of a logical && or || operator shall not contain side effects	Yes	Required
MISRA04_12.5	12.5 The operands of a logical && or || shall be primary-expressions	Yes	Required
MISRA04_12.6	12.6 The operands of logical operators (&&, || and !) should be effectively Boolean. Expressions that are effectively Boolean should not be used as operands to operators other than (&&, ||, !, =, ==, != and ?:)	Yes	Advisory
MISRA04_12.7	12.7 Bitwise operators shall not be applied to operands whose underlying type is signed	Yes	Required
MISRA04_12.8	12.8 The right-hand operand of a shift operator shall lie between zero and one less than the width in bits of the underlying type of the left-hand operand.	Yes	Required
MISRA04_12.9	12.9 The unary minus operator shall not be applied to an expression whose underlying type is unsigned	Yes	Required
MISRA04_12.10	12.10 The comma operator shall not be used	Yes	Required
MISRA04_12.11	12.11 Evaluation of constant unsigned integer expressions should not lead to wrap-around	Yes	Advisory
MISRA04_12.12	12.12 The underlying bit representations of floating-point values shall not be used	Yes	Required
MISRA04_12.13	12.13 The increment (++) and decrement (--) operators should not be mixed with other operators in an expression	Yes	Advisory
MISRA04_13.1	13.1 Assignment operators shall not be used in expressions that yield a Boolean value.	No	Required
MISRA04_13.2	13.2 Tests of a value against zero should be	No	Advisory

Check ID	Check Name	Supported	Category
	made explicit, unless the operand is effectively Boolean.		
MISRA04_13.3	13.3 Floating-point expressions shall not be tested for equality or inequality	Yes	Required
MISRA04_13.4	13.4 The controlling expression of a for statement shall not contain any objects of floating type	Yes	Required
MISRA04_13.5	13.5 The three expressions of a for statement shall be concerned only with loop control.	No	Required
MISRA04_13.6	13.6 Numeric variables being used within a for loop for iteration counting shall not be modified in the body of the loop	Yes	Required
MISRA04_13.7	13.7 Boolean operations whose results are invariant shall not be permitted	No	Required
MISRA04_14.1	14.1 There shall be no unreachable code	Yes	Required
MISRA04_14.2	14.2 All non-null statements shall either (a) have at least one side-effect however executed, or (b) cause control flow to change (Partial)	Yes	Required
MISRA04_14.3	14.3 Before preprocessing, a null statement shall only occur on a line by itself; it may be followed by a comment provided that the first character following the null statement is a white-space character	Yes	Required
MISRA04_14.4	14.4 The goto statement shall not be used	Yes	Required
MISRA04_14.5	14.5 No Continue Statements	Yes	Required
MISRA04_14.6	14.6 For any iteration statement there shall be at most one break statement used for loop termination	Yes	Required
MISRA04_14.7	14.7 A function shall have a single point of exit at the end of the function	Yes	Required
MISRA04_14.8	14.8 The statement forming the body of a switch, while, do ... while or for statement shall be a compound statement	Yes	Required
MISRA04_14.9	14.9 An if (expression) construct shall be followed by a compound statement. The else keyword shall be followed by either a compound statement, or another if statement	Yes	Required
MISRA04_14.10	14.10 All if ... else if constructs shall be terminated with an else clause	Yes	Required
MISRA04_15	15 The MISRA C switch syntax shall be used	Yes	Required
MISRA04_15.1	15.1 A switch label shall only be used when	Yes	Required

Check ID	Check Name	Supported	Category
	the most closely-enclosing compound statement is the body of a switch statement		
MISRA04_15.2	15.2 An unconditional break statement shall terminate every non-empty switch clause	Yes	Required
MISRA04_15.3	15.3 The final clause of a switch statement shall be the default clause	Yes	Required
MISRA04_15.4	15.4 A switch expression shall not represent a value that is effectively Boolean	Yes	Required
MISRA04_15.5	15.5 Every switch statement shall have at least one case clause	Yes	Required
MISRA04_16.1	16.1 Functions shall not be defined with variable numbers of arguments.	Yes	Required
MISRA04_16.2	16.2 Functions shall not call themselves, either directly or indirectly.	Yes	Required
MISRA04_16.3	16.3 All prototype parameters must have an identifier.	Yes	Required
MISRA04_16.4	16.4 Inconsistent Parameter Names	Yes	Required
MISRA04_16.5	16.5 Missing Parameters	Yes	Required
MISRA04_16.6	16.6 The number of arguments passed to a function shall match the number of parameters	Yes	Required
MISRA04_16.7	16.7 A pointer parameter in a function prototype should be declared as pointer to const if the pointer is not used to modify the addressed object	Yes	Advisory
MISRA04_16.8	16.8 All exit paths from a function with non-void return type shall have an explicit return statement with an expression	Yes	Required
MISRA04_16.9	16.9 A function identifier shall only be used with either a preceding &, or with a parenthesised parameter list, which may be empty	Yes	Required
MISRA04_16.10	16.10 Functions shall not be defined with variable numbers of arguments	No	Required
MISRA04_17.1	17.1 Pointer arithmetic shall only be applied to pointers that address an array or array element	No	Required
MISRA04_17.2	17.2 Pointer subtraction shall only be applied to pointers that address elements of the same array	No	Required
MISRA04_17.3	17.3 >, >=, <, <= shall not be applied to objects of pointer type, except where they	Yes	Required

Check ID	Check Name	Supported	Category
	point to the same array		
MISRA04_17.4	17.4 Array indexing shall be the only allowed form of pointer arithmetic	Yes	Required
MISRA04_17.5	17.5 No more than 2 levels of pointer indirection	Yes	Advisory
MISRA04_17.6	17.6 The address of an object with automatic storage shall not be assigned to another object that may persist after the first object has ceased to exist.	Yes	Required
MISRA04_18.1	18.1 All structure or union types shall be complete at the end of a translation unit.	No	Required
MISRA04_18.2	18.2 An object shall not be assigned to an overlapping object	No	Required
MISRA04_18.3	18.3 An area of memory shall not be reused for unrelated purposes	No	Required
MISRA04_18.4	18.4 Unions	Yes	Required
MISRA04_19.1	19.1 #include statements in a file should only be preceded by other preprocessor directives or comments	Yes	Advisory
MISRA04_19.2	19.2 Non-standard characters should not occur in header file names in #include directives	Yes	Advisory
MISRA04_19.3	19.3 The #include directive shall be followed by either a <filename> or "filename" sequence	Yes	Required
MISRA04_19.4	19.4 C macros shall only expand to a braced initialiser, a constant, a string literal, a parenthesised expression, a type qualifier, a storage class specifier, or a do-while-zero construct.	No	Required
MISRA04_19.5	19.5 Macros shall not be #define'd or #undef'd within a block	Yes	Required
MISRA04_19.6	19.6 Preprocessor #undef	Yes	Required
MISRA04_19.7	19.7 A function should be used in preference to a function-like macro	Yes	Advisory
MISRA04_19.8	19.8 A function-like macro shall not be invoked without all of its arguments.	No	Required
MISRA04_19.9	19.9 Arguments to a function-like macro shall not contain tokens that look like preprocessing directives	Yes	Required
MISRA04_19.10	19.10 In the definition of a function-like macro, each instance of a parameter shall be	Yes	Required

Check ID	Check Name	Supported	Category
	enclosed in parentheses, unless it is used as the operand of # or ##		
MISRA04_19.11	19.11 All macro identifiers in preprocessor directives shall be defined before use, except in #ifdef and #ifndef preprocessor directives and the defined() operator	No	Required
MISRA04_19.12	19.12 There shall be at most one occurrence of the # or ## operators in a single macro definition	Yes	Required
MISRA04_19.13	19.12 The # and ## operators should not be used	Yes	Advisory
MISRA04_19.14	19.14 The defined preprocessor operator shall only be used in one of the two standard forms	Yes	Required
MISRA04_19.15	19.15 Precautions shall be taken in order to prevent the contents of a header file being included twice	Yes	Required
MISRA04_19.16	19.16 Preprocessing directives shall be syntactically meaningful even when excluded by the preprocessor	Yes	Required
MISRA04_19.17	20.14 All #else, #elif and #endif preprocessor directives shall reside in the same file as the #if, #ifdef or #ifndef directive to which they are related	Yes	Required
MISRA04_20.1	20.1 Reserved or Standard Library Identifiers as Macros	Yes	Required
MISRA04_20.2	20.2 The names of standard library macros and objects shall not be reused	Yes	Required
MISRA04_20.3	20.3 The validity of values passed to library functions shall be checked	No	Required
MISRA04_20.4	20.4 Dynamic Memory Allocation	Yes	Required
MISRA04_20.5	20.5 The error indicator "errno" shall not be used	Yes	Required
MISRA04_20.6	20.6 The macro offsetof, in library <stddef.h>, shall not be used	Yes	Required
MISRA04_20.7	20.7 The setjmp macro and the longjmp function shall not be used	Yes	Required
MISRA04_20.8	20.8 The signal handling facilities of <signal.h> shall not be used	Yes	Required
MISRA04_20.9	20.9 Including <stdio.h>	Yes	Required
MISRA04_20.10	20.10 The library functions atof, atoi and atol from library <stdlib.h> shall not be used	Yes	Required

Check ID	Check Name	Supported	Category
MISRA04_20.11	20.11 The library functions abort, exit, getenv and system from library <stdlib.h> shall not be used	Yes	Required
MISRA04_20.12	20.12 The time handling functions of library <time.h> shall not be used	Yes	Required
MISRA04_21.1	21.1 Minimisation of run-time failures shall be ensured by the use of at least one of: (a) static analysis tools/techniques; (b) dynamic analysis tools/techniques; (c) explicit coding of checks to handle run-time faults.	Yes	Required