

The The Motor Industry Software Reliability Association (MISRA) guidelines for C published in 2012

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, and the third was released in 2012 with many more changes.

www.misra.org.uk

© MIRA Limited, 2004, 2008, 2012.

	All Rules	Advisory Rules	Mandatory Rules	Required Rules
Understand % Coverage	97%	100%	100%	96%
Understand Coverage	155	37	8	108
Total Rules	159	37	8	112

Checks

Check ID	Check Name	Supported	Category
MISRA12_1.1	1.1 The program shall contain no violations of the standard C syntax and constraints, and shall not exceed the implementation's translation limits	Yes	Required
MISRA12_1.2	1.2 Language extensions should not be used (Partial)	Yes	Advisory
MISRA12_1.3	1.3 There shall be no occurrence of undefined or critical unspecified behaviour (Partial)	Yes	Required
MISRA12_2.1	2.1 A project shall not contain	Yes	Required

	unreachable code		
MISRA12_2.2	2.2 There shall be no dead code (Partial)	Yes	Required
MISRA12_2.3	2.3 A project should not contain unused type declarations	Yes	Advisory
MISRA12_2.4	2.4 Unused Tag Declarations	Yes	Advisory
MISRA12_2.5	2.5 Unused Macros	Yes	Advisory
MISRA12_2.6	2.6 Unused Labels	Yes	Required
MISRA12_2.7	2.7 There should be no unused parameters in functions	Yes	Advisory
MISRA12_3.1	3.1 The character sequences /* and // shall not be used within a comment	Yes	Required
MISRA12_3.2	3.2 Line-Splicing in // Comments	Yes	Required
MISRA12_4.1	4.1 Octal and Hexadecimal Sequences	Yes	Required
MISRA12_4.2	4.2 Trigraphs should not be used	Yes	Advisory
MISRA12_5.1	5.1 External identifiers shall be distinct	Yes	Required
MISRA12_5.2	5.2 Identifiers declared in the same scope and name space shall be distinct	Yes	Required
MISRA12_5.3	5.3 Shadowed Identifiers	Yes	Required
MISRA12_5.4	5.4 Indistinct Macro Identifiers	Yes	Required
MISRA12_5.5	5.5 Same Identifier and Macro Name	Yes	Required
MISRA12_5.6	5.6 A typedef name shall be a unique identifier	Yes	Required
MISRA12_5.7	5.7 A tag name shall be a unique identifier	Yes	Required
MISRA12_5.8	5.8 Identifiers that define objects or functions with external linkage shall be unique	Yes	Required
MISRA12_5.9	5.9 Identifiers that define objects or functions with internal linkage should be unique	Yes	Advisory
MISRA12_6.1	6.1 Bit-fields shall only be declared with an appropriate type	Yes	Required
MISRA12_6.2	6.2 Signed single-bit named bit-fields	Yes	Required
MISRA12_7.1	7.1 Octal constants shall not be	Yes	Required

	used		
MISRA12_7.2	7.2 Unsigned Integer Literal Without Suffix	Yes	Required
MISRA12_7.3	7.3 Lowercase L Suffix	Yes	Required
MISRA12_7.4	7.4 A string literal shall not be assigned to an object unless the object's type is "pointer to const-qualified char"	Yes	Required
MISRA12_8.1	8.1 Types shall be explicitly specified	Yes	Required
MISRA12_8.2	8.2 Use Named Parameters and Prototype Form	Yes	
MISRA12_8.3	8.3 All declarations of an object or function shall use the same names and type qualifiers	Yes	Required
MISRA12_8.4	8.4 A compatible declaration shall be visible when an object or function with external linkage is defined	Yes	Required
MISRA12_8.5	8.5 An external object or function shall be declared once in one and only one file	Yes	Required
MISRA12_8.6	8.6 An identifier with external linkage shall have exactly one external definition	Yes	Required
MISRA12_8.7	8.7 Functions and objects should not be defined with external linkage if they are referenced in only one translation unit	Yes	Advisory
MISRA12_8.8	8.8 Use the static keyword for internal linkage	Yes	Required
MISRA12_8.9	8.9 Objects shall be local if only accessed from one function	Yes	Advisory
MISRA12_8.10	8.10 Non-static Inline Functions	Yes	Required
MISRA12_8.11	8.11 When an array with external linkage is declared, its size should be explicitly specified	Yes	Advisory
MISRA12_8.12	8.12 Within an enumerator list, the value of an implicitly-specified enumeration constant shall be unique	Yes	Required

MISRA12_8.13	8.13 A pointer should point to a const-qualified type whenever possible	Yes	Advisory
MISRA12_8.14	8.14 Restrict Qualifier	Yes	Required
MISRA12_9.1	9.1 The value of an object with automatic storage duration shall not be read before it has been set	Yes	Mandatory
MISRA12_9.2	9.2 The initializer for an aggregate or union shall be enclosed in braces	Yes	Required
MISRA12_9.3	9.3 Arrays shall not be partially initialized	Yes	Required
MISRA12_9.4	9.4 An element of an object shall not be initialized more than once	Yes	Required
MISRA12_9.5	9.5 Where designated initializers are used to initialize an array object the size of the array shall be specified explicitly	Yes	Required
MISRA12_10.1	10.1 Operands shall not be of an inappropriate essential type	Yes	Required
MISRA12_10.2	10.2 Expressions of essentially character type shall not be used inappropriately in addition and subtraction operations	Yes	Required
MISRA12_10.3	10.3 The value of an expression shall not be assigned to an object with a narrower essential type or of a different essential type category	Yes	Required
MISRA12_10.4	10.4 Both operands of an operator in which the usual arithmetic conversions are performed shall have the same essential type category	Yes	Required
MISRA12_10.5	10.5 The value of an expression should not be cast to an inappropriate essential type	Yes	Advisory
MISRA12_10.6	10.6 The value of a composite expression shall not be assigned to an object with wider essential type	Yes	Required
MISRA12_10.7	10.7 Implicit Casts of Operations	Yes	Required
MISRA12_10.8	10.8 The value of a composite expression shall not be cast to a	Yes	Required

	different essential type category or a wider essential type		
MISRA12_11.1	11.1 Conversions shall not be performed between a pointer to a function and any other type	Yes	Required
MISRA12_11.2	11.2 Conversions shall not be performed between a pointer to an incomplete type and any other type	Yes	Required
MISRA12_11.3	11.3 A cast shall not be performed between a pointer to object type and a pointer to a different object type	Yes	Required
MISRA12_11.4	11.4 A conversion should not be performed between a pointer to object and an integer type	Yes	Advisory
MISRA12_11.5	11.5 A conversion should not be performed from pointer to void into pointer to object	Yes	Advisory
MISRA12_11.6	11.6 A cast shall not be performed between pointer to void and an arithmetic type	Yes	Required
MISRA12_11.7	11.7 A cast shall not be performed between pointer to object and a non-integer arithmetic type	Yes	Required
MISRA12_11.8	11.8 A cast shall not remove any const or volatile qualification from the type pointed to by a pointer	Yes	Required
MISRA12_11.9	11.9 The macro NULL shall be the only permitted form of integer null pointer constant	Yes	Required
MISRA12_12.1	12.1 The precedence of operators within expressions should be made explicit	Yes	Advisory
MISRA12_12.2	12.2 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
MISRA12_12.3	12.3 The comma operator shall not be used.	Yes	Advisory
MISRA12_12.4	12.4 Evaluation of constant	Yes	Advisory

	expressions should not lead to unsigned integer wrap-around		
MISRA12_13.1	13.1 Initializer lists shall not contain persistent side effects	Yes	Required
MISRA12_13.2	13.2 The value of an expression and its persistent side effects shall be the same under all permitted evaluation orders	Yes	Required
MISRA12_13.3	13.3 A full expression containing an increment (++) or decrement (--) operator should have no other potential side effects other than that caused by the increment or decrement operator	Yes	Advisory
MISRA12_13.4	13.4 The result of an assignment operator should not be used	Yes	Advisory
MISRA12_13.5	13.5 The right hand operand of a logical && or operator shall not contain persistent side effects	Yes	Required
MISRA12_13.6	13.6 The operand of the sizeof operator shall not contain any expression which has potential side effects	Yes	Mandatory
MISRA12_14.1	14.1 A loop counter shall not have essentially floating type	Yes	Required
MISRA12_14.2	14.2 A for loop shall be well-formed	Yes	Required
MISRA12_14.3	14.3 Controlling expressions shall not be invariant	Yes	Required
MISRA12_14.4	14.4 The controlling expression of an if statement and the controlling expression of an iteration-statement shall have essentially Boolean type	Yes	Required
MISRA12_15.1	15.1 The goto statement should not be used	Yes	Advisory
MISRA12_15.2	15.2 The goto statement shall jump to a label declared later in the same function	Yes	Required
MISRA12_15.3	15.3 Goto Into or Between Blocks	Yes	Required
MISRA12_15.4	15.4 There should be no more than one break or goto statement used	Yes	Advisory

	to terminate any iteration statement		
MISRA12_15.5	15.5 A function should have a single point of exit at the end	Yes	Advisory
MISRA12_15.6	15.6 The body of an iteration-statement or a selection-statement shall be a compound-statement	Yes	Required
MISRA12_15.7	15.7 All if ... else if constructs shall be terminated with an else statement	Yes	Required
MISRA12_16.1	16.1 All switch statements shall be well-formed	Yes	Required
MISRA12_16.2	16.2 A switch label shall only be used when the most closely-enclosing compound statement is the body of a switch statement	Yes	Required
MISRA12_16.3	16.3 An unconditional break statement shall terminate every switch-clause	Yes	Required
MISRA12_16.4	16.4 Every switch statement shall have a default label	Yes	Required
MISRA12_16.5	16.5 A default label shall appear as either the first or the last switch label of a switch statement	Yes	Required
MISRA12_16.6	16.6 Every switch statement shall have at least two switch-clauses	Yes	Required
MISRA12_16.7	16.7 Switch Boolean	Yes	
MISRA12_17.1	17.1 The features of <stdarg.h> shall not be used	Yes	Required
MISRA12_17.2	17.2 Functions shall not call themselves, either directly or indirectly	Yes	Required
MISRA12_17.3	17.3 A function shall not be declared implicitly	Yes	Mandatory
MISRA12_17.4	17.4 Always return a value in non-void functions	Yes	Required
MISRA12_17.5	17.5 The function argument corresponding to a parameter declared to have an array type shall have an appropriate number of elements	Yes	Required
MISRA12_17.6	17.6 Static Keyword in Array	Yes	Mandatory

	Brackets		
MISRA12_17.7	17.7 The value returned by a function having non-void return type shall be used	Yes	Required
MISRA12_17.8	17.8 A function parameter should not be modified	Yes	Advisory
MISRA12_18.1	18.1 A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand (Partial)	Yes	Required
MISRA12_18.2	18.2 Subtraction between pointers shall only be applied to pointers that address elements of the same array	Yes	Required
MISRA12_18.3	18.3 The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object	Yes	Required
MISRA12_18.4	18.4 The +, -, += and -= operators should not be applied to an expression of pointer type	Yes	Advisory
MISRA12_18.5	18.5 Declarations should contain no more than two levels of pointer nesting	Yes	Advisory
MISRA12_18.6	18.6 The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist	Yes	Required
MISRA12_18.7	18.7 Flexible Array Members	Yes	Required
MISRA12_18.8	18.8 Variable-length arrays shall not be used	Yes	Required
MISRA12_19.1	19.1 An object shall not be assigned or copied to an overlapping object (Partial)	Yes	Mandatory
MISRA12_19.2	19.2 Unions	Yes	Advisory
MISRA12_20.1	20.1 #include directives should only be preceded by preprocessor directives or comments	Yes	Advisory

MISRA12_20.2	20.2 Invalid Header Name	Yes	Required
MISRA12_20.3	20.3 The #include directive shall be followed by either a <filename> or "filename" sequence	Yes	Required
MISRA12_20.4	20.4 Keyword Macros	Yes	Required
MISRA12_20.5	20.5 Preprocessor #undef	Yes	Advisory
MISRA12_20.6	20.6 Tokens that look like a preprocessing directive shall not occur within a macro argument	Yes	Required
MISRA12_20.7	20.7 Expressions resulting from the expansion of macro parameters shall be enclosed in parentheses	Yes	Required
MISRA12_20.8	20.8 The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1	Yes	Required
MISRA12_20.9	20.9 All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be #define'd before evaluation	Yes	Required
MISRA12_20.10	20.10 The # and ## operators should not be used	Yes	Advisory
MISRA12_20.11	20.11 A macro parameter immediately following a # operator shall not immediately be followed by a ## operator	Yes	Required
MISRA12_20.12	20.12 A macro parameter used as an operand to the # or ## operators, which is itself subject to further macro replacement, shall only be used as an operand to these operators	Yes	Required
MISRA12_20.13	20.13 Invalid Preprocessor Directives	Yes	Required
MISRA12_20.14	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
MISRA12_21.1	21.1 #define and #undef shall not be used on a reserved identifier or reserved macro name	Yes	Required

MISRA12_21.2	21.2 Reserved Identifiers or Macros	Yes	Required
MISRA12_21.3	21.3 The memory allocation and deallocation functions of <code><stdlib.h></code> shall not be used	Yes	Required
MISRA12_21.4	21.4 The standard header file <code><setjmp.h></code> shall not be used	Yes	Required
MISRA12_21.5	21.5 Standard Header <code>signal.h</code>	Yes	Required
MISRA12_21.6	21.6 C Standard Library I/O Functions	Yes	Required
MISRA12_21.7	21.7 The <code>atof</code> , <code>atoi</code> , <code>atol</code> and <code>atoll</code> functions of <code><stdlib.h></code> shall not be used	Yes	Required
MISRA12_21.8	21.8 The library functions <code>abort</code> , <code>exit</code> , <code>getenv</code> and <code>system</code> of <code><stdlib.h></code> shall not be used	Yes	Required
MISRA12_21.9	21.9 The library functions <code>bsearch</code> and <code>qsort</code> of <code><stdlib.h></code> shall not be used	Yes	Required
MISRA12_21.10	21.10 The Standard Library time and date functions shall not be used	Yes	Required
MISRA12_21.11	21.11 The standard header file <code><tgmath.h></code> shall not be used	Yes	Required
MISRA12_21.12	21.12 The exception handling features of <code><fenv.h></code> should not be used	Yes	Advisory
MISRA12_22.1	22.1 All resources obtained dynamically by means of Standard Library functions shall be explicitly released	Yes	Required
MISRA12_22.2	22.2 A block of memory shall only be freed if it was allocated by means of a Standard Library function	Yes	Required
MISRA12_22.3	22.3 The same file shall not be open for read and write access at the same time on different streams	No	Required
MISRA12_22.4	22.4 There shall be no attempt to write to a stream which has been opened as read-only	Yes	Mandatory
MISRA12_22.5	22.5 Dereference of FILE Pointer	Yes	Mandatory
MISRA12_22.6	22.6 The value of a pointer to a FILE	Yes	Mandatory

	shall not be used after the associated stream has been closed (Partial)		
MISRA12_DIR_1.1	Directive 1.1 Any implementation-defined behaviour on which the output of the program depends shall be documented and understood	No	Required
MISRA12_DIR_2.1	Directive 2.1 All source files shall compile without any compilation errors	Yes	Required
MISRA12_DIR_3.1	Directive 3.1 All code shall be traceable to documented requirements	No	Required
MISRA12_DIR_4.1	Directive 4.1 Run-time failures shall be minimized	No	Required
MISRA12_DIR_4.2	Directive 4.2 All usage of assembly language should be documented	Yes	Advisory
MISRA12_DIR_4.3	Directive 4.3 Assembly language shall be encapsulated and isolated.	Yes	Required
MISRA12_DIR_4.4	Directive 4.4 Sections of code should not be "commented out"	Yes	Advisory
MISRA12_DIR_4.5	Directive 4.5 Identifiers in the same name space with overlapping visibility should be typographically unambiguous	Yes	Advisory
MISRA12_DIR_4.6	Directive 4.6 Typedefs that indicate size and signedness should be used in place of the basic numerical types	Yes	Advisory
MISRA12_DIR_4.7	Directive 4.7 If a function returns error information, then that error information shall be tested	Yes	Required
MISRA12_DIR_4.8	Directive 4.8 If a pointer to a structure or union is never dereferenced within a translation unit, then the implementation of the object should be hidden	Yes	Advisory
MISRA12_DIR_4.9	Directive 4.9 A function should be used in preference to a function-like macro where they are	Yes	Advisory

	interchangeable		
MISRA12_DIR_4.10	Directive 4.10 Precautions shall be taken in order to prevent the contents of a header file being included more than once	Yes	Required
MISRA12_DIR_4.11	Directive 4.11 The validity of values passed to library functions shall be checked (Partial)	Yes	Required
MISRA12_DIR_4.12	Directive 4.12 Dynamic Memory Allocation	Yes	Required
MISRA12_DIR_4.13	Directive 4.13 Functions which are designed to provide operations on a resource should be called in an appropriate sequence (Partial)	Yes	Advisory