The The Motor Industry Software Reliability Association (MISRA) guidelines for C published in 2023

MISRA Mission Statement:

We provide world-leading, best practice guidelines for the safe and secure application of both embedded control systems and standalone software.

MISRA is a collaboration between manufacturers, component suppliers and engineering consultancies which seeks to promote best practice in developing safety and security-related electronic systems and other software-intensive applications. 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

www.misra.org.uk

	All Rules	Advisory Rules	Mandatory Rules	Required Rules
Understand % Coverage	78%	75%	62%	80%
Understand Coverage	140	30	8	102
Total Rules	180	40	13	127

Checks

Check ID	Check Name	Supported	Category
MISRA23_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
MISRA23_1.2	1.2 Language extensions should not be used	No	Advisory
MISRA23_1.3	1.3 There shall be no occurrence of undefined or critical unspecified behaviour	No	Required
MISRA23_2.1	2.1 A project shall not contain unreachable code	Yes	Required
MISRA23_2.2	2.2 A project shall not contain dead code	No	Required

MISRA23_2.3	2.3 A project should not contain unused type declarations	Yes	Advisory
MISRA23_2.4	2.4 A project should not contain unused tag declarations	Yes	Advisory
MISRA23_2.5	2.5 A project should not contain unused macro declarations	Yes	Advisory
MISRA23_2.6	2.6 A function should not contain unused label declarations	Yes	Required
MISRA23_2.7	2.7 A function should not contain unused parameters	Yes	Advisory
MISRA23_3.1	3.1 The character sequences /* and // shall not be used within a comment	Yes	Required
MISRA23_3.2	3.2 Line-splicing shall not be used in // comments	Yes	Required
MISRA23_4.1	4.1 Octal and Hexadecimal Sequences	Yes	Required
MISRA23_4.2	4.2 Trigraphs should not be used	Yes	Advisory
MISRA23_5.1	5.1 External identifiers shall be distinct	Yes	Required
MISRA23_5.2	5.2 Identifiers declared in the same scope and name space shall be distinct	Yes	Required
MISRA23_5.3	5.3 An identifier declared in an inner scope shall not hide an identifier declared in an outer scope	Yes	Required
MISRA23_5.4	5.4 Macro identifiers shall be distinct	Yes	Required
MISRA23_5.5	5.5 Identifiers shall be distinct from macro names	Yes	Required
MISRA23_5.6	5.6 A typedef name shall be a unique identifier	Yes	Required
MISRA23_5.7	5.7 A tag name shall be a unique identifier	Yes	Required
MISRA23_5.8	5.8 Identifiers that define objects or functions with external linkage shall be unique	Yes	Required
MISRA23_5.9	5.9 Identifiers that define objects or functions with internal linkage should be unique	Yes	Advisory

MISRA23_6.1	6.1 Bit-fields shall only be declared with an appropriate type	Yes	Required
MISRA23_6.2	6.2 Single-bit named bit fields shall not be of a signed type	Yes	Required
MISRA23_6.3	6.3 A bit field shall not be declared as a member of a union	Yes	Required
MISRA23_7.1	7.1 Octal constants shall not be used	Yes	Required
MISRA23_7.2	7.2 A "u" or "U" suffix shall be applied to all integer constants that are represented in an unsigned type	Yes	Required
MISRA23_7.3	7.3 The lowercase character "l" shall not be used in a literal suffix	Yes	Required
MISRA23_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"	No	Required
MISRA23_8.1	8.1 Types shall be explicitly specified	Yes	Required
MISRA23_8.2	8.2 Use Named Parameters and Prototype Form	Yes	Required
MISRA23_8.3	8.3 All declarations of an object or function shall use the same names and type qualifiers	Yes	Required
MISRA23_8.4	8.4 A compatible declaration shall be visible when an object or function with external linkage is defined	Yes	Required
MISRA23_8.5	8.5 An external object or function shall be declared once in one and only one file	Yes	Required
MISRA23_8.6	8.6 An identifier with external linkage shall have exactly one external definition	Yes	Required
MISRA23_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
MISRA23_8.8	8.8 The static storage class specifier shall be used in all	Yes	Required

	declarations of objects and functions that have internal linkage		
MISRA23_8.9	8.9 An object should be declared at block scope if its identifier only appears in a single function	Yes	Advisory
MISRA23_8.10	8.10 Non-static Inline Functions	Yes	Required
MISRA23_8.11	8.11 When an array with external linkage is declared, its size should be explicitly specified	Yes	Advisory
MISRA23_8.12	8.12 Within an enumerator list, the value of an implicitly-specified enumeration constant shall be unique	Yes	Required
MISRA23_8.13	8.13 A pointer should point to a const-qualified type whenever possible	No	Advisory
MISRA23_8.14	8.14 The restrict type qualifier shall not be used	Yes	Required
MISRA23_8.15	8.15 All declarations of an object with an explicit alignment specification shall specify the same alignment	Yes	Required
MISRA23_8.16	8.16 The alignment specification of zero should not appear in an object declaration	Yes	Advisory
MISRA23_8.17	8.17 At most one explicit alignment specifier should appear in an object declaration	Yes	Advisory
MISRA23_9.1	9.1 The value of an object with automatic storage duration shall not be read before it has been set	Yes	Mandatory
MISRA23_9.2	9.2 The initializer for an aggregate or union shall be enclosed in braces	Yes	Required
MISRA23_9.3	9.3 Arrays shall not be partially initialized	Yes	Required
MISRA23_9.4	9.4 An element of an object shall not be initialized more than once	Yes	Required
MISRA23_9.5	9.5 Where designated initializers are used to initialize an array	Yes	Required

	object the size of the array shall be specified explicitly		
MISRA23_9.7	9.7 Atomic objects shall be appropriately initialized before being accessed	Yes	Mandatory
MISRA23_10.1	10.1 Operands shall not be of an inappropriate essential type	Yes	Required
MISRA23_10.2	10.2 Expressions of essentially character type shall not be used inappropriately in addition and subtraction operations	No	Required
MISRA23_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	No	Required
MISRA23_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
MISRA23_10.5	10.5 The value of an expression should not be cast to an inappropriate essential type	Yes	Advisory
MISRA23_10.6	10.6 The value of a composite expression shall not be assigned to an object with wider essential type	Yes	Required
MISRA23_10.7	10.7 If a composite expression is used as one operand of an operator in which the usual arithmetic conversions are performed then the other operand shall not have wider essential type	No	Required
MISRA23_10.8	10.8 The value of a composite expression shall not be cast to a different essential type category or a wider essential type	Yes	Required
MISRA23_11.1	11.1 Conversions shall not be performed between a pointer to a function and any other type	Yes	Required
MISRA23_11.2	11.2 Conversions shall not be	Yes	Required

	performed between a pointer to an incomplete type and any other type		
MISRA23_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
MISRA23_11.4	11.4 A conversion should not be performed between a pointer to object and an integer type	Yes	Required
MISRA23_11.5	11.5 A conversion should not be performed from pointer to void into pointer to object	Yes	Advisory
MISRA23_11.6	11.6 A cast shall not be performed between pointer to void and an arithmetic type	Yes	Required
MISRA23_11.7	11.7 A cast shall not be performed between pointer to object and a non-integer arithmetic type	Yes	Required
MISRA23_11.8	11.8 A conversion shall not remove any const, volatile or _Atomic qualification from the type pointed to by a pointer	Yes	Required
MISRA23_11.9	11.9 The macro NULL shall be the only permitted form of integer null pointer constant	Yes	Required
MISRA23_12.1	12.1 The precedence of operators within expressions should be made explicit	No	Advisory
MISRA23_12.2	12.2 The right hand operand of a shift operator shall lie in the range zero to one less than the width in bits of the essential type of the left hand operand	Yes	Required
MISRA23_12.3	12.3 The comma operator shall not be used.	Yes	Advisory
MISRA23_12.4	12.4 Evaluation of constant expressions should not lead to unsigned integer wrap-around	No	Advisory
MISRA23_13.1	13.1 Initializer lists shall not contain persistent side effects	Yes	Required

MISRA23_13.2	13.2 The value of an expression and its persistent side effects shall be the same under all permitted evaluation orders	No	Required
MISRA23_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
MISRA23_13.4	13.4 The result of an assignment operator should not be used	Yes	Advisory
MISRA23_13.5	13.5 The right hand operand of a logical && or || operator shall not contain persistent side effects	Yes	Required
MISRA23_13.6	13.6 The operand of the sizeof operator shall not contain any expression which has potential side effects	Yes	Mandatory
MISRA23_14.1	14.1 A loop counter shall not have essentially floating type	Yes	Required
MISRA23_14.2	14.2 A for loop shall be well-formed	No	Required
MISRA23_14.3	14.3 Controlling expressions shall not be invariant	No	Required
MISRA23_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
MISRA23_15.1	15.1 The goto statement should not be used	Yes	Advisory
MISRA23_15.2	15.2 The goto statement shall jump to a label declared later in the same function	Yes	Required
MISRA23_15.3	15.3 Any label referenced by a goto statement shall be declared in the same block, or in any block enclosing the goto statement	Yes	Required
MISRA23_15.4	15.4 There should be no more than one break or goto statement used	Yes	Advisory

	to terminate any iteration statement		
MISRA23_15.5	15.5 A function should have a single point of exit at the end	Yes	Advisory
MISRA23_15.6	15.6 The body of an iteration-statement or a selection-statement shall be a compound-statement	Yes	Required
MISRA23_15.7	15.7 All if ... else if constructs shall be terminated with an else statement	Yes	Required
MISRA23_16.1	Switch Statement not Well-formed	Yes	Required
MISRA23_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
MISRA23_16.3	16.3 An unconditional break statement shall terminate every switch-clause	Yes	Required
MISRA23_16.4	16.4 Every switch statement shall have a default label	Yes	Required
MISRA23_16.5	16.5 A default label shall appear as either the first or the last switch label of a switch statement	Yes	Required
MISRA23_16.6	16.6 Every switch statement shall have at least two switch-clauses	Yes	Required
MISRA23_16.7	16.7 A switch-expression shall not have essentially Boolean type	No	Required
MISRA23_17.1	17.1 The standard header file <stdarg.h> shall not be used	Yes	Required
MISRA23_17.2	17.2 Functions shall not call themselves, either directly or indirectly	Yes	Required
MISRA23_17.3	17.3 A function shall not be declared implicitly	Yes	Mandatory
MISRA23_17.4	17.4 All exit paths from a function with non-void return type shall have an explicit return statement with an expression	Yes	Required
MISRA23_17.5	17.5 The function argument corresponding to a parameter	No	Advisory

	declared to have an array type shall have an appropriate number of elements		
MISRA23_17.6	17.6 The declaration of an array parameter shall not contain the static keyword between the [ ]	Yes	Mandatory
MISRA23_17.7	The value returned by a function having non-void return type shall be used	Yes	Required
MISRA23_17.8	17.8 A function parameter should not be modified	Yes	Advisory
MISRA23_17.10	17.10 A function declared with a _Noreturn function specifier shall have void return type	Yes	Required
MISRA23_17.12	17.12 A function identifier should only be used with either a preceding &, or with a parenthesized parameter list	Yes	Required
MISRA23_17.13	17.13 A function type shall not be type qualified	Yes	Required
MISRA23_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	No	Required
MISRA23_18.2	18.2 Subtraction between pointers shall only be applied to pointers that address elements of the same array	Yes	Required
MISRA23_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
MISRA23_18.4	18.4 The +, -, += and -= operators should not be applied to an expression of pointer type	No	Advisory
MISRA23_18.5	18.5 Declarations should contain no more than two levels of pointer nesting	No	Advisory
MISRA23_18.6	18.6 The address of an object with	Yes	Required

	automatic or thread-local storage shall not be copied to another object that persists after the first object has ceased to exist		
MISRA23_18.7	18.7 Flexible array members shall not be declared	Yes	Required
MISRA23_18.8	18.8 Variable-length array types shall not be used	No	Required
MISRA23_19.0.3	19.0.3 #include directives should only be preceded by preprocessor directives or comments	Yes	Advisory
MISRA23_19.1	19.1 An object shall not be assigned or copied to an overlapping object	No	Mandatory
MISRA23_19.2	19.2 The union keyword should not be used	Yes	Advisory
MISRA23_20.1	20.1 #include directives should only be preceded by preprocessor directives or comments	Yes	Advisory
MISRA23_20.2	20.2 The ', " or backslash characters and the /* or // character sequences shall not occur in a header file name	Yes	Required
MISRA23_20.3	20.3 The #include directive shall be followed by either a <filename> or "filename" sequence	Yes	Required
MISRA23_20.4	20.4 A macro shall not be defined with the same name as a keyword	Yes	Required
MISRA23_20.5	20.5 #undef should not be used	Yes	Advisory
MISRA23_20.6	20.6 Tokens that look like a preprocessing directive shall not occur within a macro argument	Yes	Required
MISRA23_20.7	20.7 Expressions resulting from the expansion of macro parameters shall be enclosed in parentheses	No	Required
MISRA23_20.8	20.8 The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1	No	Required
MISRA23_20.9	20.9 All identifiers used in the controlling expression of #if or	No	Required

	#elif preprocessing directives shall be #define'd before evaluation		
MISRA23_20.10	20.10 The # and ## operators should not be used	Yes	Advisory
MISRA23_20.11	20.11 A macro parameter immediately following a # operator shall not immediately be followed by a ## operator	Yes	Required
MISRA23_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	No	Required
MISRA23_20.13	20.13 A line whose first token is # shall be a valid preprocessing directive	Yes	Required
MISRA23_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
MISRA23_21.1	21.1 #define and #undef shall not be used on a reserved identifier or reserved macro name	Yes	Required
MISRA23_21.2	21.2 Reserved Identifiers or Macros	Yes	Required
MISRA23_21.3	21.3 The memory allocation and deallocation functions of <stdlib.h> shall not be used	Yes	Required
MISRA23_21.4	21.4 The standard header file <setjmp.h> shall not be used	Yes	Required
MISRA23_21.5	21.5 The standard header file <signal.h> shall not be used	Yes	Required
MISRA23_21.6	21.6 The Standard Library input/ output functions shall not be used	Yes	Required
MISRA23_21.7	21.7 The Standard Library functions atof, atoi, atol and atoll of <stdlib.h> shall not be used	Yes	Required
MISRA23_21.8	21.8 The Standard Library termination functions of <stdlib.h>	Yes	Required

	shall not be used		
MISRA23_21.9	21.9 The library functions bsearch and qsort of <stdlib.h> shall not be used	Yes	Required
MISRA23_21.10	21.10 The Standard Library time and date functions shall not be used	Yes	Required
MISRA23_21.11	21.11 The standard header file <tgmath.h> shall not be used	Yes	Advisory
MISRA23_21.12	21.12 The standard header file <fenv.h> shall not be used	Yes	Required
MISRA23_21.17	21.17 Use of the string handling functions from <string.h> shall not result in accesses beyond the bounds of the objects referenced by their pointer parameters	Yes	Mandatory
MISRA23_21.19	21.19 The pointers returned by the Standard Library functions localeconv, getenv, setlocale or, strerror shall only be used as if they have pointer to const-qualified type	Yes	Mandatory
MISRA23_21.20	21.20 The pointer returned by the C++ Standard Library functions asctime, ctime, gmtime, localtime, localeconv, getenv, setlocale or strerror must not be used following a subsequent call to the same function	Yes	Mandatory
MISRA23_21.21	21.21 The Standard Library function system of <stdlib.h> shall not be used	Yes	Required
MISRA23_21.24	21.24 The random number generator functions of <stdlib.h> shall not be used	Yes	Required
MISRA23_21.26	21.26 The Standard Library function mtx_timedlock() shall only be invoked on mutex objects of appropriate mutex type	Yes	Required
MISRA23_22.1	22.1 All resources obtained dynamically by means of Standard Library functions shall be explicitly	No	Required

	released		
MISRA23_22.2	22.2 A block of memory shall only be freed if it was allocated by means of a Standard Library function	No	Mandatory
MISRA23_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
MISRA23_22.4	22.4 There shall be no attempt to write to a stream which has been opened as read-only	No	Mandatory
MISRA23_22.5	22.5 A pointer to a FILE object shall not be dereferenced	No	Mandatory
MISRA23_22.6	22.6 The value of a pointer to a FILE shall not be used after the associated stream has been closed	No	Mandatory
MISRA23_22.11	22.11 A thread that was previously either joined or detached shall not be subsequently joined nor detached	Yes	Required
MISRA23_22.13	22.13 Thread objects, thread synchronization objects and thread-specific storage pointers shall have appropriate storage duration	Yes	Required
MISRA23_22.17	22.17 No thread shall unlock a mutex or call cnd_wait() or cnd_timedwait() for a mutex it has not locked before	Yes	Required
MISRA23_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
MISRA23_DIR_2.1	Directive 2.1 All source files shall compile without any compilation errors	Yes	Required
MISRA23_DIR_3.1	Directive 3.1 All code shall be traceable to documented requirements	No	Required

MISRA23_DIR_4.1	Directive 4.1 Run-time failures shall be minimized	No	Required
MISRA23_DIR_4.2	Directive 4.2 All usage of assembly language should be documented	No	Advisory
MISRA23_DIR_4.3	Directive 4.3 Assembly language shall be encapsulated and isolated.	Yes	Required
MISRA23_DIR_4.4	Directive 4.4 Sections of code should not be "commented out"	Yes	Advisory
MISRA23_DIR_4.5	Directive 4.5 Identifiers in the same name space with overlapping visibility should be typographically unambiguous	Yes	Advisory
MISRA23_DIR_4.6	Directive 4.6 Typedefs that indicate size and signedness should be used in place of the basic numerical types	Yes	Advisory
MISRA23_DIR_4.7	Directive 4.7 If a function returns error information, then that error information shall be tested	No	Required
MISRA23_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
MISRA23_DIR_4.9	Directive 4.9 A function should be used in preference to a function-like macro where they are interchangeable	No	Advisory
MISRA23_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
MISRA23_DIR_4.11	Directive 4.11 The validity of values passed to library functions shall be checked	No	Required
MISRA23_DIR_4.12	Directive 4.12 Dynamic memory allocation shall not be used	Yes	Required
MISRA23_DIR_4.13	Directive 4.13 Functions which are designed to provide operations on	No	Advisory

	a resource should be called in an appropriate sequence		
MISRA23_DIR_5.1	Directive 5.1 There shall be no data races between threads	No	Required
MISRA23_DIR_5.2	Directive 5.2 There shall be no deadlocks between threads	No	Required
MISRA23_DIR_5.3	Directive 5.3 There shall be no dynamic thread creation	Yes	Required