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	93%	89%	100%	94%
Understand Coverage	164	41	5	118
Total Rules	176	46	5	125

Checks

Check ID	Check Name	Supported	Category
MISRA23_0.0.1	0.0.1 A function shall not contain unreachable statements	Yes	Required
MISRA23_0.0.2	0.0.2 Controlling expressions should not be invariant	No	Required
MISRA23_0.1.1	0.1.1 A value should not be unnecessarily written to a local object	No	Advisory
MISRA23_0.1.2	0.1.2 The value returned by a function shall be used	Yes	Required
MISRA23_0.2.1	0.2.1 Variables with limited visibility should be used at least once	Yes	Advisory
MISRA23_0.2.2	0.2.2 A named function parameter shall be used at least once	Yes	Required
MISRA23_0.2.3	0.2.3 Types with limited visibility	Yes	Advisory

	should be used at least once		
MISRA23_0.2.4	0.2.4 Functions with limited visibility should be used at least once	Yes	Advisory
MISRA23_0.3.1	0.3.1 Floating-point arithmetic should be used appropriately	No	Advisory
MISRA23_0.3.2	0.3.2 A function call shall not violate the function's preconditions	No	Required
MISRA23_4.1.1	4.1.1 A program shall conform to ISO/IEC 14882:2017 (C++17)	No	Required
MISRA23_4.1.2	4.1.2 Deprecated features should not be used	No	Advisory
MISRA23_4.1.3	4.1.3 There shall be no occurrence of undefined or critical unspecified behaviour	No	Required
MISRA23_4.6.1	4.6.1 Operations on a memory location shall be sequenced appropriately	No	Required
MISRA23_5.0.1	5.0.1 Trigraph-like sequences should not be used	Yes	Required
MISRA23_5.7.1	5.7.1 The character sequence /* shall not be used within a C-style comment	Yes	Required
MISRA23_5.7.2	5.7.2 Sections of code should not be "commented out"	Yes	Advisory
MISRA23_5.7.3	5.7.3 Line-Splicing in // Comments	Yes	Required
MISRA23_5.10.1	5.10.1 User-defined identifiers shall have an appropriate form	Yes	Required
MISRA23_5.13.1	5.13.1 Within character literals and non raw-string literals, \ shall only be used to form a defined escape sequence or universal character name	Yes	Required
MISRA23_5.13.2	5.13.2 Octal escape sequences, hexadecimal escape sequences and universal character names shall be terminated	Yes	Required
MISRA23_5.13.3	5.13.3 Octal constants shall not be used	Yes	Required
MISRA23_5.13.4	5.13.4 Unsigned integer literals shall be appropriately suffixed	Yes	Required
MISRA23_5.13.5	5.13.5 The lowercase form of L shall	Yes	Required

	not be used as the first character in a literal suffix		
MISRA23_5.13.6	5.13.6 An integer-literal of type long long shall not use a single L or l in any suffix	Yes	Required
MISRA23_5.13.7	5.13.7 String literals with different encoding prefixes shall not be concatenated	Yes	Required
MISRA23_6.0.1	6.0.1 Block scope declarations shall not be visually ambiguous	Yes	Required
MISRA23_6.0.2	6.0.2 When an array with external linkage is declared, its size should be explicitly specified	Yes	Advisory
MISRA23_6.0.3	6.0.3 Global Namespace Declarations	Yes	Advisory
MISRA23_6.0.4	6.0.4 The identifier main shall not be used for a function other than the global function main	Yes	Required
MISRA23_6.2.1	6.2.1 The one-definition rule shall not be violated	Yes	Required
MISRA23_6.2.2	6.2.2 All declarations of a variable or function shall have the same type	Yes	Required
MISRA23_6.2.3	6.2.3 The source code used to implement an entity shall appear only once	Yes	Required
MISRA23_6.2.4	6.2.4 A header file shall not contain definitions of functions or objects that are non-inline and have external linkage	Yes	Required
MISRA23_6.4.1	6.4.1 A variable declared in an inner scope shall not hide a variable declared in an outer scope	Yes	Required
MISRA23_6.4.2	6.4.2 Derived classes shall not conceal functions that are inherited from their bases	Yes	Required
MISRA23_6.4.3	6.4.3 Names from a dependent base shall be prefixed with this->, qualified with a using declaration, or qualified with the base name	Yes	Required
MISRA23_6.5.1	6.5.1 A function or object with external linkage should be	Yes	Advisory

	introduced in a header file		
MISRA23_6.5.2	6.5.2 Internal linkage should be specified appropriately	Yes	Advisory
MISRA23_6.7.1	6.7.1 Local variables shall not have static storage duration	Yes	Required
MISRA23_6.7.2	6.7.2 Global variables shall not be used	Yes	Required
MISRA23_6.8.1	6.8.1 An object shall not be accessed outside of its lifetime	Yes	Required
MISRA23_6.8.2	6.8.2 A function must not return a reference or a pointer to a local variable with automatic storage duration	Yes	Mandatory
MISRA23_6.8.3	6.8.3 An assignment operator shall not assign the address of an object with automatic storage duration to an object with a greater lifetime	Yes	Required
MISRA23_6.8.4	6.8.4 Member functions returning references to their object should be ref-qualified appropriately	Yes	Advisory
MISRA23_6.9.1	6.9.1 The same type aliases shall be used in all declarations of the same entity	Yes	Required
MISRA23_6.9.2	6.9.2 The names of the standard signed integer types and standard unsigned integer types should not be used	Yes	Advisory
MISRA23_7.0.1	7.0.1 There shall be no conversion from type bool	Yes	Required
MISRA23_7.0.2	7.0.2 There shall be no conversion to type bool	Yes	Required
MISRA23_7.0.3	7.0.3 The numerical value of a character shall not be used	Yes	Required
MISRA23_7.0.4	7.0.4 The operands of bitwise operators and shift operators shall be appropriate	Yes	Required
MISRA23_7.0.5	7.0.5 Integral promotion and the usual arithmetic conversions shall not change the signedness or the type category of an operand	Yes	Required
MISRA23_7.0.6	7.0.6 Assignment between numeric	Yes	Required

	types shall be appropriate		
MISRA23_7.11.1	7.11.1 nullptr shall be the only form of the null-pointer-constant	Yes	Required
MISRA23_7.11.2	7.11.2 Array to Pointer Decay	Yes	Required
MISRA23_7.11.3	7.11.3 A conversion from function type to pointer-to-function type shall only occur in appropriate contexts	Yes	Required
MISRA23_8.0.1	8.0.1 Parentheses should be used to make the meaning of an expression appropriately explicit	Yes	Required
MISRA23_8.1.1	8.1.1 A non-transient lambda shall not implicitly capture this	Yes	Required
MISRA23_8.1.2	8.1.2 Variables should be captured explicitly in a non-transient lambda	Yes	Advisory
MISRA23_8.2.1	8.2.1 A virtual base class shall only be cast to a derived class by means of dynamic_cast	Yes	Required
MISRA23_8.2.2	8.2.2 C-style casts and functional notation casts shall not be used	Yes	Required
MISRA23_8.2.3	8.2.3 A cast shall not remove any const or volatile qualification from the type accessed via a pointer or by reference	Yes	Required
MISRA23_8.2.4	8.2.4 Casts shall not be performed between a pointer to function and any other type	Yes	Required
MISRA23_8.2.5	8.2.5 reinterpret_cast shall not be used	Yes	Required
MISRA23_8.2.6	8.2.6 An object with integral, enumerated, or pointer to void type shall not be cast to a pointer type	Yes	Required
MISRA23_8.2.7	8.2.7 Pointer to Integer Cast	Yes	Advisory
MISRA23_8.2.8	8.2.8 An object pointer type shall not be cast to an integral type other than std::uintptr_t or std::intptr_t	Yes	Required
MISRA23_8.2.9	8.2.9 The operand to typeid shall not be an expression of polymorphic class type	Yes	Required
MISRA23_8.2.10	8.2.10 Functions shall not call themselves, either directly or indirectly	Yes	Required

MISRA23_8.2.11	8.2.11 An argument passed via ellipsis shall have an appropriate type	Yes	Required
MISRA23_8.3.1	8.3.1 The built-in unary - operator should not be applied to an expression of unsigned type	Yes	Advisory
MISRA23_8.3.2	8.3.2 The built-in unary + operator should not be used	Yes	Advisory
MISRA23_8.7.1	8.7.1 Pointer arithmetic shall not form an invalid pointer	Yes	Required
MISRA23_8.7.2	8.7.2 Subtraction between pointers shall only be applied to pointers that address elements of the same array	Yes	Required
MISRA23_8.9.1	8.9.1 The built-in relational operators >, >=, < and <= shall not be applied to objects of pointer type, except where they point to the same array	Yes	Required
MISRA23_8.14.1	8.14.1 The right-hand operand of a logical && or || operator should not contain persistent side effects	Yes	Advisory
MISRA23_8.18.1	8.18.1 An object or subobject must not be copied to an overlapping object	Yes	Mandatory
MISRA23_8.18.2	8.18.2 The result of an assignment operator should not be used	No	Advisory
MISRA23_8.19.1	8.19.1 The comma operator shall not be used.	Yes	Advisory
MISRA23_8.20.1	8.20.1 An unsigned arithmetic operation with constant operands should not wrap	Yes	Advisory
MISRA23_9.2.1	9.2.1 An explicit type conversion shall not be an expression statement	Yes	Required
MISRA23_9.3.1	9.3.1 The body of an iteration-statement or a selection-statement shall be acompound-statement	Yes	Required
MISRA23_9.4.1	9.4.1 All if ... else if constructs shall be terminated with an else statement	Yes	Required
MISRA23_9.5.1	9.5.1 Legacy for statements should be simple	Yes	Advisory
MISRA23_9.5.2	9.5.2 A for-range-initializer shall	Yes	Required

	contain at most one function call		
MISRA23_9.6.1	9.6.1 The goto statement should not be used	Yes	Advisory
MISRA23_9.6.2	9.6.2 A goto statement shall reference a label in a surrounding block	Yes	Required
MISRA23_9.6.3	9.6.3 The goto statement shall jump to a label declared later in the function body	Yes	Required
MISRA23_9.6.4	9.6.4 A function declared with the [[noreturn]] attribute shall not return	Yes	Required
MISRA23_9.6.5	9.6.5 A function with non-void return type shall return a value on all paths	Yes	Required
MISRA23_10.0.1	10.0.1 A declaration should not declare more than one variable or member variable	Yes	Advisory
MISRA23_10.1.1	10.1.1 The target type of a pointer or lvalue reference parameter should be const-qualified appropriately	Yes	Advisory
MISRA23_10.1.2	10.1.2 The volatile qualifier shall be used appropriately	Yes	Required
MISRA23_10.2.1	10.2.1 An enumeration shall be defined with an explicit underlying type	Yes	Required
MISRA23_10.2.2	10.2.2 Unscoped enumerations should not be declared	Yes	Advisory
MISRA23_10.2.3	10.2.3 The numeric value of an unscoped enumeration with no fixed underlying type shall not be used	Yes	Required
MISRA23_10.3.1	10.3.1 There should be no unnamed namespaces in header files	Yes	Advisory
MISRA23_10.4.1	10.4.1 The asm declaration shall not be used	Yes	Required
MISRA23_11.3.1	11.3.1 Variables of array type should not be declared	Yes	Advisory
MISRA23_11.3.2	11.3.2 The declaration of an object should contain no more than two levels of pointer indirection	Yes	Advisory
MISRA23_11.6.1	11.6.1 All variables should be initialized	Yes	Advisory
MISRA23_11.6.2	11.6.2 The value of an object must	Yes	Mandatory

	not be read before it has been set		
MISRA23_11.6.3	11.6.3 Within an enumerator list, the value of an implicitly-specified enumeration constant shall be unique	Yes	Required
MISRA23_12.2.1	12.2.1 Bit-fields should not be declared	Yes	Advisory
MISRA23_12.2.2	12.2.2 A bit-field shall have an appropriate type	Yes	Required
MISRA23_12.2.3	12.2.3 A named bit-field with signed integer type shall not have a length of one bit	Yes	Required
MISRA23_12.3.1	12.3.1 Unions	Yes	Required
MISRA23_13.1.1	13.1.1 Classes should not be inherited virtually	Yes	Advisory
MISRA23_13.1.2	13.1.2 An accessible base class shall not be both virtual and non-virtual in the same hierarchy	Yes	Required
MISRA23_13.3.1	13.3.1 User-declared member functions shall use the virtual, override and final specifiers appropriately	Yes	Required
MISRA23_13.3.2	13.3.2 Parameters in an overriding virtual function shall not specify different default arguments	Yes	Required
MISRA23_13.3.3	13.3.3 The parameters in all declarations or overrides of a function shall either be unnamed or have identical names	Yes	Required
MISRA23_13.3.4	13.3.4 A comparison of a potentially virtual pointer to member function shall only be with nullptr	Yes	Required
MISRA23_14.1.1	14.1.1 Non-static data members should be either all private or all public	Yes	Advisory
MISRA23_15.0.2	15.0.2 User-provided copy and move member functions of a class should have appropriate signatures	Yes	Advisory
MISRA23_15.1.1	15.1.1 An object's dynamic type shall not be used from within its constructor or destructor	Yes	Required

MISRA23_15.1.2	15.1.2 All constructors of a class should explicitly initialize all of its virtual base classes and immediate base classes	Yes	Advisory
MISRA23_15.1.3	15.1.3 Conversion operators and constructors that are callable with a single argument shall be explicit	Yes	Required
MISRA23_15.1.4	15.1.4 All direct, non-static data members of a class should be initialized before the class object is accessible	Yes	Required
MISRA23_15.1.5	15.1.5 A class shall only define an initializer-list constructor when it is the only constructor	Yes	Required
MISRA23_15.8.1	15.8.1 User-provided copy assignment operators and move assignment operators shall handle self-assignment	Yes	Required
MISRA23_16.5.1	16.5.1 The logical AND and logical OR operators shall not be overloaded	Yes	Required
MISRA23_16.5.2	16.5.2 The address-of operator shall not be overloaded	Yes	Required
MISRA23_16.6.1	16.6.1 Symmetrical operators should only be implemented as non-member functions	Yes	Advisory
MISRA23_17.8.1	17.8.1 Function templates shall not be explicitly specialized	Yes	Required
MISRA23_18.1.1	18.1.1 An exception object shall not have pointer type	Yes	Required
MISRA23_18.1.2	18.1.2 An empty throw shall only occur within the compound-statement of a catch handler	Yes	Required
MISRA23_18.3.1	18.3.1 There should be at least one exception handler to catch all otherwise unhandled exceptions	Yes	Advisory
MISRA23_18.3.2	18.3.2 An exception of class type shall be caught by const reference or reference	Yes	Required
MISRA23_18.3.3	18.3.3 Handlers for a function-try-block of a constructor or destructor shall not refer to non-static members	Yes	Required

	from their class or its bases		
MISRA23_18.4.1	18.4.1 Exception-unfriendly functions shall be noexcept	Yes	Required
MISRA23_18.5.1	18.5.1 A noexcept function should not attempt to propagate an exception to the calling function	No	Advisory
MISRA23_18.5.2	18.5.2 Program-terminating functions should not be used	Yes	Advisory
MISRA23_19.0.1	19.0.1 A line whose first token is # shall be a valid preprocessing directive	Yes	Required
MISRA23_19.0.2	19.0.2 Function-like macros shall not be defined	Yes	Required
MISRA23_19.0.4	19.0.4 #undef should only be used for macros defined previously in the same file	Yes	Advisory
MISRA23_19.1.1	19.1.1 The defined preprocessor operator shall be used appropriately	Yes	Required
MISRA23_19.1.2	19.1.2 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_19.1.3	19.1.3 All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be defined prior to evaluation	No	Required
MISRA23_19.2.1	19.2.1 Precautions shall be taken in order to prevent the contents of a header file being included more than once	Yes	Required
MISRA23_19.2.2	19.2.2 The #include directive shall be followed by either a <filename> or "filename" sequence	Yes	Required
MISRA23_19.2.3	19.2.3 The ' or " or \ characters and the /* or // character sequences shall not occur in a header file name	Yes	Required
MISRA23_19.3.1	19.3.1 The # and ## operators should not be used	Yes	Advisory
MISRA23_19.3.2	19.3.2 A macro parameter immediately following a # operator	Yes	Required

	shall not immediately be followed by a ## operator		
MISRA23_19.3.3	19.3.3 The argument to a mixed-use macro parameter shall not be subject to further expansion	No	Required
MISRA23_19.3.4	19.3.4 Parentheses shall be used to ensure macro arguments are expanded appropriately	Yes	Required
MISRA23_19.3.5	19.3.5 Tokens that look like a preprocessing directive shall not occur within a macro argument	Yes	Advisory
MISRA23_19.6.1	19.6.1 The #pragma directive and the _Pragma operator should not be used	Yes	Advisory
MISRA23_21.2.1	21.2.1 The library functions atof, atoi, atol and atoll from library <cstdlib> shall not be used	Yes	Required
MISRA23_21.2.2	21.2.2 The string handling functions from <cstring>, <cstdlib>, <cwchar> and <cinttypes> shall not be used	Yes	Required
MISRA23_21.2.3	21.2.3 The library function system from <cstdlib> shall not be used	Yes	Required
MISRA23_21.2.4	21.2.4 The macro offsetof shall not be used	Yes	Required
MISRA23_21.6.1	21.6.1 Dynamic memory should not be used	Yes	Advisory
MISRA23_21.6.2	21.6.2 Dynamic memory shall be managed automatically	Yes	Required
MISRA23_21.6.3	21.6.3 Advanced memory management shall not be used	Yes	Required
MISRA23_21.6.4	21.6.4 If a project defines either a sized or unsized version of a global operator delete, then both shall be defined	Yes	Required
MISRA23_21.6.5	21.6.5 A pointer to an incomplete class type shall not be deleted	Yes	Required
MISRA23_21.10.1	21.10.1 The features of <cstdarg> shall not be used	Yes	Required
MISRA23_21.10.2	21.10.2 The standard header file <csetjmp> shall not be used	Yes	Required
MISRA23_21.20.3	21.20.3 The facilities provided by the	Yes	Required

	standard header file <csignal> shall not be used		
MISRA23_22.3.1	22.3.1 The assert macro shall not be used with a constant-expression	Yes	Required
MISRA23_22.4.1	22.4.1 The literal value zero shall be the only value assigned to errno	Yes	Required
MISRA23_23.11.1	23.11.1 The raw pointer constructors of std::shared_ptr and std::unique_ptr should not be used	Yes	Advisory
MISRA23_24.5.1	24.5.1 The character handling functions from <cctype> and <cwctype> shall not be used	Yes	Required
MISRA23_24.5.2	24.5.2 The C++ Standard Library functions memcpy, memmove and memcmp from <cstring> shall not be used	Yes	Required
MISRA23_25.5.1	25.5.1 The setlocale and std::locale::global functions shall not be called	Yes	Required
MISRA23_25.5.2	25.5.2 The pointers returned by the C++ Standard Library functions localeconv, getenv, setlocale or strerror must only be used as if they have pointer to const-qualified type	Yes	Mandatory
MISRA23_25.5.3	25.5.3 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_26.3.1	26.3.1 std::vector should not be specialized with bool	Yes	Advisory
MISRA23_28.3.1	28.3.1 Predicates shall not have persistent side effects	Yes	Required
MISRA23_28.6.1	28.6.1 The argument to std::move shall be a non-const lvalue	Yes	Required
MISRA23_28.6.2	28.6.2 Forwarding references and std::forward shall be used together	Yes	Required
MISRA23_28.6.3	28.6.3 An object shall not be used while in a potentially moved-from state	Yes	Required

MISRA23_28.6.4	28.6.4 The result of std::remove, std::remove_if, std::unique and empty shall be used	Yes	Required
MISRA23_30.0.1	30.0.1 The C Library input/output functions shall not be used	Yes	Required
MISRA23_30.0.2	30.0.2 Reads and writes on the same file stream shall be separated by a positioning operation	Yes	Required