The The Motor Industry Software Reliability Association (MISRA) guidelines for C++ published in 2008

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.

www.misra-cpp.com

	All Rules	Advisory Rules	Document Rules	Required Rules
Understand % Coverage	80%	83%	8%	84%
Understand Coverage	183	15	1	167
Total Rules	228	18	12	198

Checks

Check ID	Check Name	Supported	Category
MISRA08_0-1-1	0-1-1 A project shall not contain unreachable code	Yes	Required
MISRA08_0-1-2	0-1-2 Infeasible Paths	Yes	Required
MISRA08_0-1-3	0-1-3 A project shall not contain unused variables	Yes	Required
MISRA08_0-1-4	0-1-4 A project shall not contain non-volatile POD variables having only one use.	Yes	Required
MISRA08_0-1-5	0-1-5 Unused Type Declarations	Yes	Required
MISRA08_0-1-6	0-1-6 A project shall not contain instances of non-volatile variables being given values that are never subsequently used	No	Required
MISRA08_0-1-7	0-1-7 The value returned by a	Yes	Required

	function having a non-void return type that is not an overloaded operator shall always be used		
MISRA08_0-1-8	0-1-8 All functions with void return type shall have external side effect(s)	Yes	Required
MISRA08_0-1-9	0-1-9 There shall be no dead code	No	Required
MISRA08_0-1-10	0-1-10 All defined functions called	Yes	Required
MISRA08_0-1-11	0-1-11 Unused Parameters in Non-virtual Functions	Yes	Required
MISRA08_0-1-12	0-1-12 There shall be no unused parameters (named or unnamed) in the set of parameters for a virtual function and all the functions that override it	Yes	Required
MISRA08_0-2-1	0-2-1 An object shall not be assigned to an overlapping object	No	Required
MISRA08_0-3-1	0-3-1 Minimization of run-time failures shall be ensured by the use of static analysis tools	Yes	Document
MISRA08_0-3-2	0-3-2 If a function generates error information, then that error information shall be tested	No	Required
MISRA08_0-4-1	0-4-1 Use of scaled-integer or fixed-point arithmetic shall be documented	No	Document
MISRA08_0-4-2	0-4-2 Use of floating-point arithmetic shall be documented	No	Document
MISRA08_0-4-3	0-4-3 Floating-point implementations shall comply with a defined floating-point standard	No	Document
MISRA08_1-0-1	1-0-1 All code shall conform to ISO/IEC 14882:2003 "The C++ Standard Incorporating Technical Corrigendum 1"	No	Required
MISRA08_1-0-2	1-0-2 Multiple compilers shall only be used if they have a common, defined interface	No	Document
MISRA08_1-0-3	1-0-3 The implementation of integer division in the chosen compiler shall be determined and documented	No	Document
MISRA08_2-2-1	2-2-1 The character set and the corresponding encoding shall be	No	Document

	documented		
MISRA08_2-3-1	2-3-1 Trigraphs shall not be used	Yes	Required
MISRA08_2-5-1	2-5-1 Digraphs shall not be used	Yes	Advisory
MISRA08_2-7-1	2-7-1 The character sequence /* shall not be used within a C-style comment.	Yes	Required
MISRA08_2-7-2	2-7-2 Sections of code shall not be "commented out" using C-style comments	Yes	Required
MISRA08_2-7-3	2-7-3 Sections of code should not be "commented out" using C++ comments	Yes	Advisory
MISRA08_2-10-1	2-10-1 Different identifiers shall be typographically unambiguous	Yes	Required
MISRA08_2-10-2	2-10-2 Shadowed Identifiers	Yes	Required
MISRA08_2-10-3	2-10-3 A typedef name shall be a unique identifier	Yes	Required
MISRA08_2-10-4	2-10-4 A class, union or enum name (including qualification, if any) shall be a unique identifier	Yes	Required
MISRA08_2-10-5	2-10-5 The identifier name of a non-member object or function with static storage duration should not be reused	Yes	Advisory
MISRA08_2-10-6	2-10-6 If an identifier refers to a type, it shall not also refer to an object or a function in the same scope	No	Required
MISRA08_2-13-1	2-13-1 Nonstandard Escape Sequences	Yes	Required
MISRA08_2-13-2	2-13-2 Octal constants (other than zero) and octal escape sequences (other than "\0") shall not be used.	Yes	Required
MISRA08_2-13-3	2-13-3 A "U" suffix shall be applied to all octal or hexadecimal integer literals of unsigned type.	Yes	Required
MISRA08_2-13-4	2-13-4 Literal suffixes shall be upper case	Yes	Required
MISRA08_2-13-5	2-13-5 Narrow and wide string literals shall not be concatenated	Yes	Required
MISRA08_3-1-1	3-1-1 It shall be possible to include any header file in multiple translation	Yes	Required

	units without violating the One Definition Rule		
MISRA08_3-1-2	3-1-2 Functions shall not be declared at block scope	Yes	Required
MISRA08_3-1-3	3-1-3 Array Size Missing	Yes	Required
MISRA08_3-2-1	3-2-1 All declarations of an object or function shall have compatible types	Yes	Required
MISRA08_3-2-2	3-2-2 The One Definition Rule	Yes	Required
MISRA08_3-2-3	3-2-3 A type, object or function that is used in multiple translation units shall be declared in one and only one file	Yes	Required
MISRA08_3-2-4	3-2-4 An identifier with external linkage shall have exactly one definition	Yes	Required
MISRA08_3-3-1	3-3-1 Objects or functions with external linkage shall be declared in a header file	Yes	Required
MISRA08_3-3-2	3-3-2 If a function has internal linkage then all redeclarations shall include the static storage class specifier	Yes	Required
MISRA08_3-4-1	3-4-1 Declarations at Lowest Scope	Yes	Required
MISRA08_3-9-1	3-9-1 The types used for an object, a function return type, or a function parameter shall be token-for-token identical in all declarations and re-declarations	Yes	Required
MISRA08_3-9-2	3-9-2 Typedefs that indicate size and signedness should be used in place of the basic numerical types	Yes	Advisory
MISRA08_3-9-3	3-9-3 The underlying bit representations of floating-point values shall not be used	Yes	Required
MISRA08_4-5-1	4-5-1 Expressions with type bool shall not be used as operands to built-in operators other than the assignment operator =, the logical operators &&, ||, !, the equality operators == and !=, the unary & operator, and the conditional operator	Yes	Required

MISRA08_4-5-2	4-5-2 Expressions with type enum shall not be used as operands to built-in operators other than the subscript operator [ ], the assignment operator =, the equality operators == and !=, the unary & operator, and the relational operators <, <=, >, >=	Yes	Required
MISRA08_4-5-3	4-5-3 Character Operators	Yes	Required
MISRA08_4-10-1	4-10-1 NULL shall not be used as an integer value	Yes	Required
MISRA08_4-10-2	4-10-2 Literal zero (0) shall not be used as the null-pointer-constant.	Yes	Required
MISRA08_5-0-1	5-0-1 The value of an expression shall be the same under any order of evaluation that the standard permits	Yes	Required
MISRA08_5-0-2	5-0-2 Limited dependence should be placed on C++ operator precedence rules in expressions	Yes	Advisory
MISRA08_5-0-3	5-0-3 A cvalue expression shall not be implicitly converted to a different underlying type	Yes	Required
MISRA08_5-0-4	5-0-4 An implicit integral conversion shall not change the signedness of the underlying type	Yes	Required
MISRA08_5-0-5	5-0-5 There shall be no implicit floating-integral conversions	Yes	Required
MISRA08_5-0-6	5-0-6 An implicit integral or floating-point conversion shall not reduce the size of the underlying type	Yes	Required
MISRA08_5-0-7	5-0-7 There shall be no explicit floating-integral conversions of a cvalue expression	Yes	Required
MISRA08_5-0-8	5-0-8 An explicit integral or floating-point conversion shall not increase the size of the underlying type of a cvalue expression	Yes	Required
MISRA08_5-0-9	5-0-9 An explicit integral conversion shall not change the signedness of the underlying type of a cvalue expression	Yes	Required
MISRA08_5-0-10	5-0-10 If the bitwise operators ~ and	Yes	Required

	<< 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		
MISRA08_5-0-11	5-0-11 The plain char type shall only be used for the storage and use of character values	Yes	Required
MISRA08_5-0-12	5-0-12 Signed char and unsigned char type shall only be used for the storage and use of numeric values	Yes	Required
MISRA08_5-0-13	5-0-13 The condition of an if-statement and the condition of an iteration-statement shall have type bool	No	Required
MISRA08_5-0-14	5-0-14 The first operand of a conditional-operator shall have type bool	Yes	Required
MISRA08_5-0-15	5-0-15 Array indexing shall be the only form of pointer arithmetic	No	Required
MISRA08_5-0-16	5-0-16 A pointer operand and any pointer resulting from pointer arithmetic using that operand shall both address elements of the same array	No	Required
MISRA08_5-0-17	5-0-17 Subtraction between pointers shall only be applied to pointers that address elements of the same array	Yes	Required
MISRA08_5-0-18	5-0-18 >, >=, <, <= shall not be applied to objects of pointer type, except where they point to the same array	Yes	Required
MISRA08_5-0-19	5-0-19 No more than 2 levels of pointer indirection	Yes	Required
MISRA08_5-0-20	5-0-20 Non-constant operands to a binary bitwise operator shall have the same underlying type	Yes	Required
MISRA08_5-0-21	5-0-21 Bitwise operators shall only be applied to operands of unsigned underlying type	Yes	Required
MISRA08_5-2-1	5-2-1 Each operand of a logical && or	No	Required

	|| shall be a postfix expression		
MISRA08_5-2-2	5-2-2 A pointer to a virtual base class shall only be cast to a pointer to a derived class by means of dynamic_cast	No	Required
MISRA08_5-2-3	5-2-3 Casts from a base class to a derived class should not be performed on polymorphic types	Yes	Advisory
MISRA08_5-2-4	5-2-4 C-style casts (other than void casts) and functional notation casts (other than explicit constructor calls) shall not be used	No	Required
MISRA08_5-2-5	5-2-5 A cast shall not remove any const or volatile qualification from the type of a pointer or reference	Yes	Required
MISRA08_5-2-6	5-2-6 A cast shall not convert a pointer to a function to any other pointer type, including a pointer to function type	Yes	Required
MISRA08_5-2-7	5-2-7 An object with pointer type shall not be converted to an unrelated pointer type, either directly or indirectly	No	Required
MISRA08_5-2-8	5-2-8 An object with integer type or pointer to void type shall not be converted to an object with pointer type.	Yes	Required
MISRA08_5-2-9	5-2-9 Pointer to Integer Cast	Yes	Advisory
MISRA08_5-2-10	5-2-10 The increment (++) and decrement (--) operators shall not be mixed with other operators in an expression	Yes	Advisory
MISRA08_5-2-11	5-2-11 The comma operator, && operator and the || operator shall not be overloaded	Yes	Required
MISRA08_5-2-12	5-2-12 Array to Pointer Decay	Yes	Required
MISRA08_5-3-1	5-3-1 Each operand of the ! operator, the logical && or the logical || operators shall have type bool	Yes	Required
MISRA08_5-3-2	5-3-2 The unary minus operator shall not be applied to an expression	No	Required

	whose underlying type is unsigned		
MISRA08_5-3-3	5-3-3 The unary & operator shall not be overloaded	Yes	Required
MISRA08_5-3-4	5-3-4 Evaluation of the operand to the sizeof operator shall not contain side effects	Yes	Required
MISRA08_5-8-1	5-8-1 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
MISRA08_5-14-1	5-14-1 The right hand operand of a logical && or || operator shall not contain side effects	No	Required
MISRA08_5-17-1	5-17-1 The semantic equivalence between a binary operator and its assignment operator form shall be preserved	No	Required
MISRA08_5-18-1	5-18-1 The comma operator shall not be used	No	Required
MISRA08_5-19-1	5-19-1 Evaluation of constant unsigned integer expressions should not lead to wrap-around	No	Advisory
MISRA08_6-2-1	6-2-1 Assignment operators shall not be used in sub-expressions	No	Required
MISRA08_6-2-2	6-2-2 Floating-point expressions shall not be directly or indirectly tested for equality or inequality	Yes	Required
MISRA08_6-2-3	6-2-3 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
MISRA08_6-3-1	6-3-1 The statement forming the body of a switch, while, do ... while or for statement shall be a compound statement	Yes	Required
MISRA08_6-4-1	6-4-1 An if (condition) construct shall be followed by a compound statement. The else keyword shall be	Yes	Required

	followed by either a compound statement, or another if statement		
MISRA08_6-4-2	6-4-2 All if ... else if constructs shall be terminated with an else clause	Yes	Required
MISRA08_6-4-3	6-4-3 A switch statement shall be a well-formed switch statement	No	Required
MISRA08_6-4-4	6-4-4 A switch-label shall only be used when the most closely-enclosing compound statement is the body of a switch statement	Yes	Required
MISRA08_6-4-5	6-4-5 An unconditional throw or break statement shall terminate every non-empty switch-clause	Yes	Required
MISRA08_6-4-6	6-4-6 The final clause of a switch statement shall be the default-clause	Yes	Required
MISRA08_6-4-7	6-4-7 The condition of a switch statement shall not have bool type	No	Required
MISRA08_6-4-8	6-4-8 Every switch statement shall have at least one case clause	Yes	Required
MISRA08_6-5-1	6-5-1 A for loop shall contain a single loop-counter which shall not have floating-point type	Yes	Required
MISRA08_6-5-2	6-5-2 If loop-counter is not modified by -- or ++, then, within condition, the loop-counter shall only be used as an operand to <=, <, > or >=	Yes	Required
MISRA08_6-5-3	6-5-3 The loop-counter shall not be modified within condition or statement	Yes	Required
MISRA08_6-5-4	6-5-4 The loop-counter shall be modified by one of: --, ++, -= n, or += n; where n remains constant for the duration of the loop	Yes	Required
MISRA08_6-5-5	6-5-5 A loop-control-variable other than the loop-counter shall not be modified within condition or expression	Yes	Required
MISRA08_6-5-6	6-5-6 A loop-control-variable other than the loop-counter which is modified in statement shall have type bool	Yes	Required

MISRA08_6-6-1	6-6-1 Any label referenced by a goto statement shall be declared in the same block, or in a block enclosing the goto statement	Yes	Required
MISRA08_6-6-2	6-6-2 The goto statement shall jump to a label declared later in the same function body	Yes	Required
MISRA08_6-6-3	6-6-3 The continue statement shall only be used within a well-formed for loop	No	Required
MISRA08_6-6-4	6-6-4 For any iteration statement there shall be no more than one break or goto statement used for loop termination	Yes	Required
MISRA08_6-6-5	6-6-5 A function shall have a single point of exit at the end of the function	Yes	Required
MISRA08_7-1-1	7-1-1 A variable which is not modified shall be const qualified	Yes	Required
MISRA08_7-1-2	7-1-2 A pointer or reference parameter in a function shall be declared as pointer to const or reference to const if the corresponding object is not modified	Yes	Required
MISRA08_7-2-1	7-2-1 An expression with enum underlying type shall only have values corresponding to the enumerators of the enumeration	Yes	Required
MISRA08_7-3-1	7-3-1 Global Namespace Declarations	Yes	Required
MISRA08_7-3-2	7-3-2 The identifier main shall not be used for a function other than the global function main	Yes	Required
MISRA08_7-3-3	7-3-3 There shall be no unnamed namespaces in header files.	Yes	Required
MISRA08_7-3-4	7-3-4 Using-directives shall not be used.	Yes	Required
MISRA08_7-3-5	7-3-5 Declaration and Using-Declaration Clash	Yes	Required
MISRA08_7-3-6	7-3-6 using-directives and using-declarations (excluding class scope or function scope using-declarations)	Yes	Required

	shall not be used in header files.		
MISRA08_7-4-1	7-4-1 All usage of assembler shall be documented	No	Document
MISRA08_7-4-2	7-4-2 Assembler instructions shall only be introduced using the asm declaration.	Yes	Required
MISRA08_7-4-3	7-4-3 Assembly language shall be encapsulated and isolated.	Yes	Required
MISRA08_7-5-1	7-5-1 A function shall not return a reference or a pointer to an automatic variable (including parameters), defined within the function.	Yes	Required
MISRA08_7-5-2	7-5-2 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
MISRA08_7-5-3	7-5-3 A function shall not return a reference or a pointer to a parameter that is passed by reference or const reference	No	Required
MISRA08_7-5-4	7-5-4 Functions should not call themselves, either directly or indirectly.	Yes	Advisory
MISRA08_8-0-1	8-0-1 Single Declarations	Yes	Required
MISRA08_8-3-1	8-3-1 Parameters in an overriding virtual function shall either use the same default arguments as the function they override, or else shall not specify any default arguments.	Yes	Required
MISRA08_8-4-1	8-4-1 Functions shall not be defined using the ellipsis notation	Yes	Required
MISRA08_8-4-2	8-4-2 Use the same identifier in definition and declaration of functions.	Yes	Required
MISRA08_8-4-3	8-4-3 Always return a value in non-void functions	Yes	Required
MISRA08_8-4-4	8-4-4 A function identifier shall either be used to call the function or it shall be preceded by &	Yes	Required
MISRA08_8-5-1	8-5-1 All variables shall have a	Yes	Required

	defined value before they are used		
MISRA08_8-5-2	8-5-2 Incorrect Initializer Lists	Yes	Required
MISRA08_8-5-3	8-5-3 The = construct in enumerator list shall only be used on either the first item alone, or all items explicitly.	Yes	Required
MISRA08_9-3-1	9-3-1 Const Member Function Returning Non-Const Pointer or Reference	Yes	Required
MISRA08_9-3-2	9-3-2: Method Returning Non-const Handle to Class Data	Yes	Required
MISRA08_9-3-3	9-3-3 If a member function can be made static then it shall be made static, otherwise if it can be made const then it shall be made const.	Yes	Required
MISRA08_9-5-1	9-5-1 Unions	Yes	Required
MISRA08_9-6-1	9-6-1 When the absolute positioning of bits representing a bit-field is required, then the behavior and packing of bit-fields shall be documented	No	Document
MISRA08_9-6-2	9-6-2 Bool, Unsigned, or Signed Bit-fields	Yes	Required
MISRA08_9-6-3	9-6-3 Enum Bit-fields	Yes	Required
MISRA08_9-6-4	9-6-4 Named signed bit-field length	Yes	Required
MISRA08_10-1-1	10-1-1 Classes should not be derived from virtual bases	Yes	Advisory
MISRA08_10-1-2	10-1-2 A base class shall only be declared virtual if it is used in a diamond hierarchy	Yes	Required
MISRA08_10-1-3	10-1-3 An accessible base class shall not be both virtual and non-virtual in the same hierarchy	Yes	Required
MISRA08_10-2-1	10-2-1 All accessible entity names within a multiple inheritance hierarchy should be unique	No	Advisory
MISRA08_10-3-1	10-3-1 Virtual Function Redefinition	Yes	Required
MISRA08_10-3-2	10-3-2 Each overriding virtual function shall be declared with the virtual keyword.	Yes	Required
MISRA08_10-3-3	10-3-3 Pure Virtual Overriding Non-pure	Yes	Required

MISRA08_11-0-1	11-0-1 Non-private Member Data	Yes	Required
MISRA08_12-1-1	12-1-1 An object's dynamic type shall not be used from the body of its constructor or destructor	Yes	Required
MISRA08_12-1-2	12-1-2 Explicitly call all immediate and virtual base classes	Yes	Advisory
MISRA08_12-1-3	12-1-3 All constructors that are callable with a single argument of fundamental type shall be declared explicit.	Yes	Required
MISRA08_12-8-1	12-8-1 Side Effects in Copy Constructors	Yes	Required
MISRA08_12-8-2	12-8-2 The copy assignment operator shall be declared protected or private in an abstract class	No	Required
MISRA08_14-5-1	14-5-1 A non-member generic function shall only be declared in a namespace that is not an associated namespace	No	Required
MISRA08_14-5-2	14-5-2 A copy constructor shall be declared when there is a template constructor with a single parameter that is a generic parameter	Yes	Required
MISRA08_14-5-3	14-5-3 A copy assignment operator shall be declared when there is a template assignment operator with a parameter that is a generic parameter	Yes	Required
MISRA08_14-6-1	14-6-1 In a class template with a dependent base, any name that may be found in that dependent base shall be referred to using a qualified-id or this->	No	Required
MISRA08_14-6-2	14-6-2 The function or operator chosen by overload resolution shall resolve to a function declared previously in the translation unit	No	Required
MISRA08_14-7-1	14-7-1 Templates Not Instantiated	Yes	Required
MISRA08_14-7-2	14-7-2 For any given template specialization, an explicit instantiation of the template with the template-arguments used in the specialization	No	Required

	shall not render the program ill-formed		
MISRA08_14-7-3	14-7-3 All partial and explicit specializations for a template shall be declared in the same file as the declaration of their primary template	No	Required
MISRA08_14-8-1	14-8-1 Overloaded function templates shall not be explicitly specialized	Yes	Required
MISRA08_14-8-2	14-8-2 The viable function set for a function call should either contain no function specializations, or only contain function specializations	No	Advisory
MISRA08_15-0-1	15-0-1 Exceptions shall only be used for error handling	No	Document
MISRA08_15-0-2	15-0-2 An exception object should not have pointer type	Yes	Advisory
MISRA08_15-0-3	15-0-3 Control shall not be transferred into a try or catch block using a goto or a switch statement	No	Required
MISRA08_15-1-1	15-1-1 The assignment-expression of a throw statement shall not itself cause an exception to be thrown	Yes	Required
MISRA08_15-1-2	15-1-2 NULL Throw	Yes	Required
MISRA08_15-1-3	15-1-3 Empty Throw	Yes	Required
MISRA08_15-3-1	15-3-1 Exceptions shall be raised only after start-up and before termination of the program	Yes	Required
MISRA08_15-3-2	15-3-2 There should be at least one exception handler to catch all otherwise unhandled exceptions	Yes	Advisory
MISRA08_15-3-3	15-3-3 Members in function-try-blocks in constructors or destructors	Yes	Required
MISRA08_15-3-4	15-3-4 Each exception explicitly thrown in the code shall have a handler of a compatible type in all call paths that could lead to that point	No	Required
MISRA08_15-3-5	15-3-5 Exception by Value	Yes	Required
MISRA08_15-3-6	15-3-6 Order of Catch Blocks with Derived Classes	Yes	Required
MISRA08_15-3-7	15-3-7 Catch-All Statement Before	Yes	Required

	Last		
MISRA08_15-4-1	15-4-1 Inconsistent Exception-Specification	Yes	Required
MISRA08_15-5-1	15-5-1 A class destructor shall not exit with an exception	Yes	Required
MISRA08_15-5-2	15-5-2 Exceptions thrown shall be the type indicated by the function	Yes	Required
MISRA08_15-5-3	15-5-3 The terminate() function shall not be called implicitly	No	Required
MISRA08_16-0-1	16-0-1 #include directives in a file shall only be preceded by other preprocessor directives or comments	Yes	Required
MISRA08_16-0-2	16-0-2 Macros shall only be #define'd or #undef'd in the global namespace	Yes	Required
MISRA08_16-0-3	16-0-3 Preprocessor #undef	Yes	Required
MISRA08_16-0-4	16-0-4 Function-like macros shall not be defined	Yes	Required
MISRA08_16-0-5	16-0-5 Arguments to a function-like macro shall not contain tokens that look like preprocessing directives	Yes	Required
MISRA08_16-0-6	16-0-6 In the definition of a function-like macro, each instance of a parameter shall be enclosed in parentheses, unless it is used as the operand of # or ##	Yes	Required
MISRA08_16-0-7	16-0-7 Undefined macro identifiers shall not be used in #if or #elif preprocessor directives, except as operands to the defined operator	Yes	Required
MISRA08_16-0-8	16-0-8 Invalid Preprocessor Directives	Yes	Required
MISRA08_16-1-1	16-1-1 The defined preprocessor operator shall only be used in one of the two standard forms	Yes	Required
MISRA08_16-1-2	16-1-2 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
MISRA08_16-2-1	16-2-1 The pre-processor shall only	Yes	Required

	be used for file inclusion and include guards		
MISRA08_16-2-2	16-2-2 Invalid Macro Usage	Yes	Required
MISRA08_16-2-3	16-2-3 Include guards shall be provided	Yes	Required
MISRA08_16-2-4	16-2-4 The ', ", /* or // characters shall not occur in a header file name	Yes	Required
MISRA08_16-2-5	16-2-5 The backslash character should not occur in a header file name	Yes	Advisory
MISRA08_16-2-6	16-2-6 The #include directive shall be followed by either a <filename> or "filename" sequence	Yes	Required
MISRA08_16-3-1	16-3-1 There shall be at most one occurrence of the # or ## operators in a single macro definition	Yes	Required
MISRA08_16-3-2	16-3-2 The # and ## operators should not be used	Yes	Advisory
MISRA08_16-6-1	16-6-1 All uses of the #pragma directive shall be documented	No	Document
MISRA08_17-0-1	17-0-1 Reserved or Standard Library Identifiers as Macros	Yes	Required
MISRA08_17-0-2	17-0-2 The names of standard library macros and objects shall not be reused	Yes	Required
MISRA08_17-0-3	17-0-3 Standard Library Function Names	Yes	Required
MISRA08_17-0-4	17-0-4 All library code shall conform to MISRA C++	No	Document
MISRA08_17-0-5	17-0-5 The setjmp macro and the longjmp function shall not be used	Yes	Required
MISRA08_18-0-1	18-0-1 The C library shall not be used	Yes	Required
MISRA08_18-0-2	18-0-2 The library functions atof, atoi and atol from library <cstdlib> shall not be used	Yes	Required
MISRA08_18-0-3	18-0-3 The library functions abort, exit, getenv and system from library <cstdlib> shall not be used	Yes	Required
MISRA08_18-0-4	18-0-4 The time handling functions of library <ctime> shall not be used	Yes	Required
MISRA08_18-0-5	18-0-5 Unbounded Functions of	Yes	Required

	<cstring>		
MISRA08_18-2-1	18-2-1 The macro offsetof shall not be used.	Yes	Required
MISRA08_18-4-1	18-4-1 Dynamic Memory Allocation	Yes	Required
MISRA08_18-7-1	18-7-1 The signal handling facilities of <csignal> shall not be used	Yes	Required
MISRA08_19-3-1	19-3-1 The error indicator "errno" shall not be used.	Yes	Required
MISRA08_27-0-1	27-0-1 The stream input/output library <cstdio> shall not be used	Yes	Required