Goanna 3.5.0 Standards Data Sheet for MISRA C++:2008
misrac++2008-datasheet.pdf

Motor Industry Software Reliability Association (MISRA) C++:2008 Standard

Mapping of *MISRA C++:2008* items to Goanna checks

The following table lists all the *MISRA C++:2008* items that are identified by Goanna.

MISRA C++:2008 ID	MISRA C++:2008 Description	Goanna Checks	Goanna Check Description
0-1-1	(Required) A project shall not contain unreachable code.	**MISRAC++2008-0-1-1**	In all executions, a part of the program is not executed.
0-1-2	(Required) A project shall not contain infeasible paths.	**MISRAC++2008-0-1-2_a** **MISRAC++2008-0-1-2_b** **MISRAC++2008-0-1-2_c**	The condition in if, for, while, do-while and ternary operator will always be met. The condition in if, for, while, do-while and ternary operator will never be met. A case statement within a switch statement is unreachable.
0-1-3	(Required) A project shall not contain unused variables.	**Partly implemented** **MISRAC++2008-0-1-3**	Some cases require manual checking. A variable is neither read nor written for any execution.
0-1-4	(Required) A project shall not contain non-volatile POD variables having only one use.	**Partly implemented** **MISRAC++2008-0-1-4**	Some cases require manual checking. A variable is assigned a value that is never used.
0-1-5	(Required) A project shall not contain unused type declarations.	**Not implemented**	This rule requires manual checking.
0-1-6	(Required) A project shall not contain instances of non-volatile variables being given values that are never subsequently used.	**MISRAC++2008-0-1-6**	A variable is assigned a value that is never used.
0-1-7	(Required) The value returned by a function having a non-void return type that is not an overloaded operator shall always be used.	**MISRAC++2008-0-1-7**	Unused function return values (excluding overloaded operators)
0-1-8	(Required) All functions with void return type shall have external side effect(s).	**MISRAC++2008-0-1-8**	A function with no return type and no side effects effectively does nothing.
0-1-9	(Required) There shall be no dead code.	**MISRAC++2008-0-1-9**	In all executions, a part of the program is not executed.
0-1-10	(Required) Every defined function shall be called at least once.	**Not implemented**	This rule requires manual checking.
0-1-11	(Required) There shall be no unused parameters (named or unnamed) in nonvirtual functions.	**MISRAC++2008-0-1-11**	A function parameter is declared but not used.
0-1-12	(Required) 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.	**Not implemented**	This rule requires manual checking.
0-2-1	(Required) An object shall not be assigned to an overlapping object.	**MISRAC++2008-0-2-1**	Assignments from one field of a union to another.
0-3-1	(Document) Minimization 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.	**Static analysis**	Correct use of Goanna static analysis assists in compliance with this rule

0-3-2	(Required) If a function generates error information, then that error information shall be tested.	**MISRAC++2008-0-3-2**	The return value for a library function that may return an error value is not used.
0-4-1	(Document) Use of scaled-integer or fixed-point arithmetic shall be documented.	**Document**	This rule requires manual documentation.
0-4-2	(Document) Use of floating-point arithmetic shall be documented.	**Document**	This rule requires manual documentation.
0-4-3	(Document) Floating-point implementations shall comply with a defined floating-point standard.	**Document**	This rule requires manual documentation.
1-0-1	(Required) All code shall conform to ISO/IEC 14882:2003 "The C++ Standard Incorporating Technical Corrigendum 1".	**Not implemented**	This rule requires manual checking.
1-0-2	(Document) Multiple compilers shall only be used if they have a common, defined interface.	**Document**	This rule requires manual documentation.
1-0-3	(Document) The implementation of integer division in the chosen compiler shall be determined and documented.	**Document**	This rule requires manual documentation.
2-2-1	(Document) The character set and the corresponding encoding shall be documented.	**Document**	This rule requires manual documentation.
2-3-1	(Required) Trigraphs shall not be used.	**MISRAC++2008-2-3-1**	Uses of trigraphs (in string literals only)
2-5-1	(Advisory) Digraphs should not be used.	**Not implemented**	This rule requires manual checking.
2-7-1	(Required) The character sequence /* shall not be used within a C-style comment.	**MISRAC++2008-2-7-1**	Appearances of /* inside comments
2-7-2	(Required) Sections of code shall not be "commented out" using C-style comments.	**MISRAC++2008-2-7-2**	To allow comments to contain pseudo-code or code samples, only comments that end in ';', '', or '' characters are considered to be commented-out code.
2-7-3	(Advisory) Sections of code should not be "commented out" using C++ comments.	**MISRAC++2008-2-7-3**	To allow comments to contain pseudo-code or code samples, only comments that end in ';', '', or '' characters are considered to be commented-out code.
2-10-1	(Required) Different identifiers shall be typographically unambiguous.	**Not implemented**	This rule requires manual checking.
2-10-2	(Required) Identifiers declared in an inner scope shall not hide an identifier declared in an outer scope.	**MISRAC++2008-2-10-2_a** **MISRAC++2008-2-10-2_b** **MISRAC++2008-2-10-2_c** **MISRAC++2008-2-10-2_d**	The definition of a local variable hides a global definition. The definition of a local variable hides a previous local definition. A variable declaration hides a parameter of the function The definition of a local variable hides a member of the class.
2-10-3	(Required) A typedef name (including qualification, if any) shall be a unique identifier.	**Partly implemented** **MISRAC++2008-2-10-3**	Some cases require manual checking. Typedef with this name already declared.
2-10-4	(Required) A class, union or enum name (including qualification, if any) shall be a unique identifier.	**Partly implemented** **MISRAC++2008-2-10-4**	Some cases require manual checking. A class, struct, union or enum declaration that clashes with a previous declaration.
2-10-5	(Advisory) The identifier name of a non-member object or function with static storage duration should not be reused.	**Partly implemented** **MISRAC++2008-2-10-5**	Some cases require manual checking. A identifier is used that can clash with another static identifier.
2-10-6	(Required) If an identifier refers to a type, it shall not also refer to an object or a function in the same scope.	**Not implemented**	This rule requires manual checking.
2-13-1	(Required) Only those escape sequences that are defined in ISO/IEC 14882:2003 shall be used.	**Not implemented**	This rule requires manual checking.

2-13-2	(Required) Octal constants (other than zero) and octal escape sequences (other than 0) shall not be used.	**Partly implemented** **MISRAC++2008-2-13-2**	Some cases require manual checking. Uses of octal integer constants
2-13-3	(Required) A "U" suffix shall be applied to all octal or hexadecimal integer literals of unsigned type.	**MISRAC++2008-2-13-3**	A U suffix shall be applied to all constants of unsigned type.
2-13-4	(Required) Literal suffixes shall be upper case.	**MISRAC++2008-2-13-4_a** **MISRAC++2008-2-13-4_b**	Lower case suffixes on floating constants Lower case suffixes on integer constants
2-13-5	(Required) Narrow and wide string literals shall not be concatenated.	**Not implemented**	This rule requires manual checking.
3-1-1	(Required) It shall be possible to include any header file in multiple translation units without violating the One Definition Rule.	**Partly implemented** **MISRAC++2008-3-1-1**	Some cases require manual checking. Non-inline functions defined in header files
3-1-2	(Required) Functions shall not be declared at block scope.	**Not implemented**	This rule requires manual checking.
3-1-3	(Required) When an array is declared, its size shall either be stated explicitly or defined implicitly by initialization.	**MISRAC++2008-3-1-3**	External arrays declared without size stated explicitly or defined implicitly by initialization.
3-2-1	(Required) All declarations of an object or function shall have compatible types.	**PARSE**	Parse errors are generated for this rule
3-2-2	(Required) The One Definition Rule shall not be violated.	**Not implemented**	This rule requires manual checking.
3-2-3	(Required) A type, object or function that is used in multiple translation units shall be declared in one and only one file.	**Not implemented**	This rule requires manual checking.
3-2-4	(Required) An identifier with external linkage shall have exactly one definition.	**Not implemented**	This rule requires manual checking.
3-3-1	(Required) Objects or functions with external linkage shall be declared in a header file.	**Not implemented**	This rule requires manual checking.
3-3-2	(Required) If a function has internal linkage then all re-declarations shall include the static storage class specifier.	**Not implemented**	This rule requires manual checking.
3-4-1	(Required) An identifier declared to be an object or type shall be defined in a block that minimizes its visibility.	**Not implemented**	This rule requires manual checking.
3-9-1	(Required) 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.	**Not implemented**	This rule requires manual checking.
3-9-2	(Advisory) typedefs that indicate size and signedness should be used in place of the basic numerical types.	**MISRAC++2008-3-9-2**	Uses of basic types char, int, short, long, double, and float without typedef
3-9-3	(Required) The underlying bit representations of floating-point values shall not be used.	**Partly implemented** **MISRAC++2008-3-9-3**	Some cases require manual checking. An expression provides access to the bit-representation of a floating point variable.
4-5-1	(Required) 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.	**MISRAC++2008-4-5-1**	Uses of arithmetic operators on boolean operands.

4-5-2	(Required) Expressions with type enum shall not be used as operands to builtin operators other than the subscript operator [ ], the assignment operator =, the equality operators == and !=, the unary & operator, and the relational operators <, <=, >, >=.	**MISRAC++2008-4-5-2**	Use of unsafe operators on variable of enumeration type.
4-5-3	(Required) Expressions with type (plain) char and wchar_t shall not be used as operands to built-in operators other than the assignment operator =, the equality operators == and !=, and the unary & operator.	**MISRAC++2008-4-5-3**	Arithmetic on objects of type plain char, without an explicit signed or unsigned qualifier
4-10-1	(Required) NULL shall not be used as an integer value.	**Not implemented**	This rule requires manual checking.
4-10-2	(Required) Literal zero (0) shall not be used as the null-pointer-constant.	**Not implemented**	This rule requires manual checking.
5-0-1	(Required) The value of an expression shall be the same under any order of evaluation that the standard permits.	**MISRAC++2008-5-0-1_a** **MISRAC++2008-5-0-1_b** **MISRAC++2008-5-0-1_c**	Expressions which depend on order of evaluation There shall be no more than one read access with volatile-qualified type within one sequence point There shall be no more than one modification access with volatile-qualified type within one sequence point
5-0-2	(Advisory) Limited dependence should be placed on C++ operator precedence rules in expressions.	**MISRAC++2008-5-0-2**	Add parentheses to avoid implicit operator precedence.
5-0-3	(Required) A cvalue expression shall not be implicitly converted to a different underlying type.	**MISRAC++2008-5-0-3**	A cvalue expression shall not be implicitly converted to a different underlying type.
5-0-4	(Required) An implicit integral conversion shall not change the signedness of the underlying type.	**MISRAC++2008-5-0-4**	An implicit integral conversion shall not change the signedness of the underlying type.
5-0-5	(Required) There shall be no implicit floating-integral conversions.	**MISRAC++2008-5-0-5**	There shall be no implicit floating-integral conversions.
5-0-6	(Required) An implicit integral or floating-point conversion shall not reduce the size of the underlying type.	**MISRAC++2008-5-0-6**	An implicit integral or floating-point conversion shall not reduce the size of the underlying type.
5-0-7	(Required) There shall be no explicit floating-integral conversions of a cvalue expression.	**MISRAC++2008-5-0-7**	There shall be no explicit floating-integral conversions of a cvalue expression.
5-0-8	(Required) An explicit integral or floating-point conversion shall not increase the size of the underlying type of a cvalue expression.	**MISRAC++2008-5-0-8**	An explicit integral or floating-point conversion shall not increase the size of the underlying type of a cvalue expression.
5-0-9	(Required) An explicit integral conversion shall not change the signedness of the underlying type of a cvalue expression.	**MISRAC++2008-5-0-9**	An explicit integral conversion shall not change the signedness of the underlying type of a cvalue expression.
5-0-10	(Required) 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.	**MISRAC++2008-5-0-10**	Bitwise operation on unsigned char or unsigned short, that are not immediately cast to this type to ensure consistent truncation
5-0-11	(Required) The plain char type shall only be used for the storage and use of character values.	**Not implemented**	This rule requires manual checking.
5-0-12	(Required) signed char and unsigned char type shall only be used for the storage and use of numeric values.	**Not implemented**	This rule requires manual checking.
5-0-13	(Required) The condition of an if-statement and the condition of an iteration-statement shall have type bool.	**MISRAC++2008-5-0-13_a**	Non-boolean termination conditions in do ... while statements.

		MISRAC++2008-5-0-13_b	Non-boolean termination conditions in for loops.
		MISRAC++2008-5-0-13_c	Non-boolean conditions in if statements.
		MISRAC++2008-5-0-13_d	Non-boolean termination conditions in while statements.
5-0-14	(Required) The first operand of a conditional-operator shall have type bool.	**MISRAC++2008-5-0-14**	Non-boolean operands to the conditional (? :) operator
5-0-15	(Required) Array indexing shall be the only form of pointer arithmetic.	**MISRAC++2008-5-0-15_a**	Array indexing shall be the only allowed form of pointer arithmetic.
		MISRAC++2008-5-0-15_b	Array indexing shall only be applied to objects defined as an array type.
5-0-16	(Required) A pointer operand and any pointer resulting from pointer arithmetic using that operand shall both address elements of the same array.	**MISRAC++2008-5-0-16_a**	Pointer arithmetic applied to a pointer that references a stack address
		MISRAC++2008-5-0-16_b	Invalid pointer arithmetic with an automatic variable that is neither an array nor a pointer.
		MISRAC++2008-5-0-16_c	Array access is out of bounds.
		MISRAC++2008-5-0-16_d	Array access may be out of bounds, depending on which path is executed.
		MISRAC++2008-5-0-16_e	A pointer to an array is used outside the array bounds
		MISRAC++2008-5-0-16_f	A pointer to an array is potentially used outside the array bounds
5-0-17	(Required) Subtraction between pointers shall only be applied to pointers that address elements of the same array.	**Not implemented**	This rule requires manual checking.
5-0-18	(Required) >, >=, <, <= shall not be applied to objects of pointer type, except where they point to the same array.	**Not implemented**	This rule requires manual checking.
5-0-19	(Required) The declaration of objects shall contain no more than two levels of pointer indirection.	**MISRAC++2008-5-0-19**	The declaration of objects should contain no more than two levels of pointer indirection.
5-0-20	(Required) Non-constant operands to a binary bitwise operator shall have the same underlying type.	**Not implemented**	This rule requires manual checking.
5-0-21	(Required) Bitwise operators shall only be applied to operands of unsigned underlying type.	**MISRAC++2008-5-0-21**	Applications of bitwise operators to signed operands
5-2-1	(Required) Each operand of a logical && or || shall be a postfix-expression.	**Not implemented**	This rule requires manual checking.
5-2-2	(Required) A pointer to a virtual base class shall only be cast to a pointer to a derived class by means of dynamic_cast.	**Not implemented**	This rule requires manual checking.
5-2-3	(Advisory) Casts from a base class to a derived class should not be performed on polymorphic types.	**Not implemented**	This rule requires manual checking.
5-2-4	(Required) C-style casts (other than void casts) and functional notation casts (other than explicit constructor calls) shall not be used.	**MISRAC++2008-5-2-4**	Uses of old style casts (other than void casts)
5-2-5	(Required) A cast shall not remove any const or volatile qualification from the type of a pointer or reference.	**MISRAC++2008-5-2-5**	Casts that remove any const or volatile qualification.
5-2-6	(Required) A cast shall not convert a pointer to a function to any other pointer type, including a pointer to function type.	**MISRAC++2008-5-2-6**	A cast shall not convert a pointer to a function to any other pointer type, including a pointer to function type.
5-2-7	(Required) An object with pointer type shall not be converted to an unrelated pointer type, either directly or indirectly.	**MISRAC++2008-5-2-7**	A pointer to object type is cast to a pointer to different object type

5-2-8	(Required) An object with integer type or pointer to void type shall not be converted to an object with pointer type.	**Not implemented**	This rule requires manual checking.
5-2-9	(Advisory) A cast should not convert a pointer type to an integral type.	**MISRAC++2008-5-2-9**	A cast should not be performed between a pointer type and an integral type.
5-2-10	(Advisory) The increment (++) and decrement (–) operators should not be mixed with other operators in an expression.	**MISRAC++2008-5-2-10**	Uses of increment (++) and decrement (–) operators mixed with other operators in an expression.
5-2-11	(Required) The comma operator, && operator and the || operator shall not be overloaded.	**MISRAC++2008-5-2-11_a** **MISRAC++2008-5-2-11_b**	Overloaded && and || operators Overloaded comma operator
5-2-12	(Required) An identifier with array type passed as a function argument shall not decay to a pointer.	**Not implemented**	This rule requires manual checking.
5-3-1	(Required) Each operand of the ! operator, the logical && or the logical || operators shall have type bool.	**MISRAC++2008-5-3-1**	Operands of logical operators (&&, ||, and !) that are not of type bool.
5-3-2	(Required) The unary minus operator shall not be applied to an expression whose underlying type is unsigned.	**MISRAC++2008-5-3-2_a** **MISRAC++2008-5-3-2_b**	Uses of unary - on unsigned expressions Uses of unary - on unsigned expressions
5-3-3	(Required) The unary & operator shall not be overloaded.	**MISRAC++2008-5-3-3**	The & operator shall not be overloaded.
5-3-4	(Required) Evaluation of the operand to the sizeof operator shall not contain side effects.	**MISRAC++2008-5-3-4**	Sizeof expressions containing side effects
5-8-1	(Required) 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.	**MISRAC++2008-5-8-1**	Out of range shifts
5-14-1	(Required) The right hand operand of a logical && or || operator shall not contain side effects.	**MISRAC++2008-5-14-1**	Right hand operands of && or || that contain side effects
5-17-1	(Required) The semantic equivalence between a binary operator and its assignment operator form shall be preserved.	**Not implemented**	This rule requires manual checking.
5-18-1	(Required) The comma operator shall not be used.	**MISRAC++2008-5-18-1**	Uses of the comma operator
5-19-1	(Advisory) Evaluation of constant unsigned integer expressions should not lead to wrap-around.	**Partly implemented** **MISRAC++2008-5-19-1**	Some cases require manual checking. A constant unsigned integer expression overflows
6-2-1	(Required) Assignment operators shall not be used in sub-expressions.	**MISRAC++2008-6-2-1**	Assignment in a sub-expression.
6-2-2	(Required) Floating-point expressions shall not be directly or indirectly tested for equality or inequality.	**MISRAC++2008-6-2-2**	Floating point comparisons using == or !=
6-2-3	(Required) 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.	**MISRAC++2008-6-2-3**	Stray semicolons on the same line as other code
6-3-1	(Required) The statement forming the body of a switch, while, do ... while or for statement shall be a compound statement.	**MISRAC++2008-6-3-1_a** **MISRAC++2008-6-3-1_b** **MISRAC++2008-6-3-1_c** **MISRAC++2008-6-3-1_d**	Missing braces in do ... while statements Missing braces in for statements Missing braces in switch statements Missing braces in while statements
6-4-1	(Required) An if (condition) construct shall be followed by a compound statement. The else keyword shall be followed by either a compound statement, or another if statement.	**MISRAC++2008-6-4-1**	Missing braces in if, else, and else if statements

6-4-2	(Required) All if ... else if constructs shall be terminated with an else clause.	**MISRAC++2008-6-4-2**	If ... else if constructs that are not terminated with an else clause.
6-4-3	(Required) A switch statement shall be a well-formed switch statement.	**MISRAC++2008-6-4-3**	Switch statements that do not conform to the MISRA C switch syntax.
6-4-4	(Required) A switch-label shall only be used when the most closely-enclosing compound statement is the body of a switch statement.	**MISRAC++2008-6-4-4**	Switch labels in nested blocks.
6-4-5	(Required) An unconditional throw or break statement shall terminate every non-empty switch-clause.	**MISRAC++2008-6-4-5**	Non-empty switch cases not terminated by break
6-4-6	(Required) The final clause of a switch statement shall be the default-clause.	**MISRAC++2008-6-4-6**	Switch statements with no default clause, or a default clause that is not the final clause.
6-4-7	(Required) The condition of a switch statement shall not have bool type.	**MISRAC++2008-6-4-7**	A switch expression shall not represent a value that is effectively boolean.
6-4-8	(Required) Every switch statement shall have at least one case-clause.	**MISRAC++2008-6-4-8**	Switch statements with no cases.
6-5-1	(Required) A for loop shall contain a single loop-counter which shall not have floating type.	**Partly implemented** **MISRAC++2008-6-5-1_a**	Some cases require manual checking. Floating-point values in the controlling expression of a for statement.
6-5-2	(Required) If loop-counter is not modified by – or ++, then, within condition, the loop-counter shall only be used as an operand to <=, <, > or >=.	**MISRAC++2008-6-5-2**	Loop counter may not match loop condition test.
6-5-3	(Required) The loop-counter shall not be modified within condition or statement.	**MISRAC++2008-6-5-3**	A `for` loop counter variable is modified in the body of the loop.
6-5-4	(Required) The loop-counter shall be modified by one of: –, ++, -=n, or +=n; where n remains constant for the duration of the loop.	**MISRAC++2008-6-5-4**	Potential inconsistent loop counter modification.
6-5-5	(Required) A loop-control-variable other than the loop-counter shall not be modified within condition or expression.	**Not implemented**	This rule requires manual checking.
6-5-6	(Required) A loop-control-variable other than the loop-counter which is modified in statement shall have type bool.	**MISRAC++2008-6-5-6**	A non-boolean variable is modified in the loop and used as loop condition.
6-6-1	(Required) Any label referenced by a goto statement shall be declared in the same block, or in a block enclosing the goto statement.	**MISRAC++2008-6-6-1**	The target of the goto is a nested code block.
6-6-2	(Required) The goto statement shall jump to a label declared later in the same function body.	**MISRAC++2008-6-6-2**	Goto declared after target label.
6-6-3	(Required) The continue statement shall only be used within a well-formed for loop.	**Not implemented**	This rule requires manual checking.
6-6-4	(Required) For any iteration statement there shall be no more than one break or goto statement used for loop termination.	**MISRAC++2008-6-6-4**	Multiple break points from loop.
6-6-5	(Required) A function shall have a single point of exit at the end of the function.	**MISRAC++2008-6-6-5**	A function shall have a single point of exit at the end of the function.
7-1-1	(Required) A variable which is not modified shall be const qualified.	**MISRAC++2008-7-1-1**	A local variable is not modified after its initialization and so should be const qualified.

7-1-2	(Required) 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.	**MISRAC++2008-7-1-2**	A function does not modify one of its parameters.
7-2-1	(Required) An expression with enum underlying type shall only have values corresponding to the enumerators of the enumeration.	**MISRAC++2008-7-2-1**	Conversions to enum that are out of range of the enumeration.
7-3-1	(Required) The global namespace shall only contain main, namespace declarations and extern "C" declarations.	**Not implemented**	This rule requires manual checking.
7-3-2	(Required) The identifier main shall not be used for a function other than the global function main.	**Not implemented**	This rule requires manual checking.
7-3-3	(Required) There shall be no unnamed namespaces in header files.	**Not implemented**	This rule requires manual checking.
7-3-4	(Required) using-directives shall not be used.	**Not implemented**	This rule requires manual checking.
7-3-5	(Required) Multiple declarations for an identifier in the same namespace shall not straddle a using-declaration for that identifier.	**Not implemented**	This rule requires manual checking.
7-3-6	(Required) using-directives and using-declarations (excluding class scope or function scope using-declarations) shall not be used in header files.	**Not implemented**	This rule requires manual checking.
7-4-1	(Document) All usage of assembler shall be documented.	**Document**	This rule requires manual documentation.
7-4-2	(Required) Assembler instructions shall only be introduced using the asm declaration.	**Not implemented**	This rule requires manual checking.
7-4-3	(Required) Assembly language shall be encapsulated and isolated.	**Partly implemented** **MISRAC++2008-7-4-3**	Some cases require manual checking. Inline asm statements that are not encapsulated in functions
7-5-1	(Required) A function shall not return a reference or a pointer to an automatic variable (including parameters), defined within the function.	**MISRAC++2008-7-5-1_a** **MISRAC++2008-7-5-1_b**	A stack object is returned from a function as a reference. May return address on the stack.
7-5-2	(Required) 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.	**MISRAC++2008-7-5-2_a** **MISRAC++2008-7-5-2_b** **MISRAC++2008-7-5-2_c** **MISRAC++2008-7-5-2_d**	Store a stack address in a global pointer. Store a stack address in the field of a global struct. Store stack address outside function via parameter. Store stack address via reference parameter.
7-5-3	(Required) A function shall not return a reference or a pointer to a parameter that is passed by reference or const reference.	**Not implemented**	This rule requires manual checking.
7-5-4	(Advisory) Functions should not call themselves, either directly or indirectly.	**MISRAC++2008-7-5-4_a** **MISRAC++2008-7-5-4_b**	Functions that call themselves directly. Functions that call themselves indirectly.
8-0-1	(Required) An init-declarator-list or a member-declarator-list shall consist of a single init-declarator or member-declarator respectively.	**MISRAC++2008-8-0-1**	Declarations shall only contain one variable or constant each.
8-3-1	(Required) 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.	**Not implemented**	This rule requires manual checking.
8-4-1	(Required) Functions shall not be defined using the ellipsis notation.	**MISRAC++2008-8-4-1**	Functions defined using ellipsis (...) notation

8-4-2	(Required) The identifiers used for the parameters in a re-declaration of a function shall be identical to those in the declaration.	**Not implemented**	This rule requires manual checking.
8-4-3	(Required) All exit paths from a function with non-void return type shall have an explicit return statement with an expression.	**MISRAC++2008-8-4-3**	For some execution, no return statement is executed in a function with a non-void return type
8-4-4	(Required) A function identifier shall either be used to call the function or it shall be preceded by &.	**MISRAC++2008-8-4-4**	Function addresses taken without explicit &
8-5-1	(Required) All variables shall have a defined value before they are used.	**MISRAC++2008-8-5-1_a** **MISRAC++2008-8-5-1_b** **MISRAC++2008-8-5-1_c**	In all executions, a variable is read before it is assigned a value. In some execution, a variable is read before it is assigned a value. Dereference of an uninitialized or NULL pointer.
8-5-2	(Required) Braces shall be used to indicate and match the structure in the nonzero initialization of arrays and structures.	**Partly implemented** **MISRAC++2008-8-5-2**	Some cases require manual checking. This check points out where a non-zero array initialisation does not exactly match the structure of the array declaration.
8-5-3	(Required) In an enumerator list, the = construct shall not be used to explicitly initialize members other than the first, unless all items are explicitly initialized.	**Not implemented**	This rule requires manual checking.
9-3-1	(Required) const member functions shall not return non-const pointers or references to class-data.	**MISRAC++2008-9-3-1**	A member function qualified as `const` returns a pointer member variable.
9-3-2	(Required) Member functions shall not return non-const handles to class-data.	**MISRAC++2008-9-3-2**	Member functions that return non-const handles to members
9-3-3	(Required) 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.	**Not implemented**	This rule requires manual checking.
9-5-1	(Required) Unions shall not be used.	**MISRAC++2008-9-5-1**	All unions
9-6-1	(Document) When the absolute positioning of bits representing a bit-field is required, then the behaviour and packing of bit-fields shall be documented.	**Document**	This rule requires manual documentation.
9-6-2	(Required) Bit-fields shall be either bool type or an explicitly unsigned or signed integral type.	**MISRAC++2008-9-6-2**	Bitfields with plain int type
9-6-3	(Required) Bit-fields shall not have enum type.	**MISRAC++2008-9-6-3**	Bitfields with plain int type
9-6-4	(Required) Named bit-fields with signed integer type shall have a length of more than one bit.	**MISRAC++2008-9-6-4**	Signed single-bit fields (excluding anonymous fields)
10-1-1	(Advisory) Classes should not be derived from virtual bases.	**Not implemented**	This rule requires manual checking.
10-1-2	(Required) A base class shall only be declared virtual if it is used in a diamond hierarchy.	**Not implemented**	This rule requires manual checking.
10-1-3	(Required) An accessible base class shall not be both virtual and non-virtual in the same hierarchy.	**Not implemented**	This rule requires manual checking.
10-2-1	(Advisory) All accessible entity names within a multiple inheritance hierarchy should be unique.	**Not implemented**	This rule requires manual checking.

10-3-1	(Required) There shall be no more than one definition of each virtual function on each path through the inheritance hierarchy.	**Not implemented**	This rule requires manual checking.
10-3-2	(Required) Each overriding virtual function shall be declared with the virtual keyword.	**Not implemented**	This rule requires manual checking.
10-3-3	(Required) A virtual function shall only be overridden by a pure virtual function if it is itself declared as pure virtual.	**Not implemented**	This rule requires manual checking.
11-0-1	(Required) Member data in non-POD class types shall be private.	**Not implemented**	This rule requires manual checking.
12-1-1	(Required) An object's dynamic type shall not be used from the body of its constructor or destructor.	**Partly implemented** **MISRAC++2008-12-1-1_a** **MISRAC++2008-12-1-1_b**	Some cases require manual checking. A virtual member function is called in a class constructor. A virtual member function is called in a class destructor.
12-1-2	(Advisory) All constructors of a class should explicitly call a constructor for all of its immediate base classes and all virtual base classes.	**Not implemented**	This rule requires manual checking.
12-1-3	(Required) All constructors that are callable with a single argument of fundamental type shall be declared explicit.	**MISRAC++2008-12-1-3**	All constructors that are callable with a single argument of fundamental type shall be declared `explicit`.
12-8-1	(Required) A copy constructor shall only initialize its base classes and the nonstatic members of the class of which it is a member.	**Not implemented**	This rule requires manual checking.
12-8-2	(Required) The copy assignment operator shall be declared protected or private in an abstract class.	**Not implemented**	This rule requires manual checking.
14-5-1	(Required) A non-member generic function shall only be declared in a namespace that is not an associated namespace.	**Not implemented**	This rule requires manual checking.
14-5-2	(Required) A copy constructor shall be declared when there is a template constructor with a single parameter that is a generic parameter.	**Not implemented**	This rule requires manual checking.
14-5-3	(Required) A copy assignment operator shall be declared when there is a template assignment operator with a parameter that is a generic parameter.	**Not implemented**	This rule requires manual checking.
14-6-1	(Required) 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->	**Not implemented**	This rule requires manual checking.
14-6-2	(Required) The function chosen by overload resolution shall resolve to a function declared previously in the translation unit.	**Not implemented**	This rule requires manual checking.
14-7-1	(Required) All class templates, function templates, class template member functions and class template static members shall be instantiated at least once.	**Not implemented**	This rule requires manual checking.

14-7-2	(Required) For any given template specialization, an explicit instantiation of the template with the template-arguments used in the specialization shall not render the program ill-formed.	**Not implemented**	This rule requires manual checking.
14-7-3	(Required) All partial and explicit specializations for a template shall be declared in the same file as the declaration of their primary template.	**Not implemented**	This rule requires manual checking.
14-8-1	(Required) Overloaded function templates shall not be explicitly specialized.	**Not implemented**	This rule requires manual checking.
14-8-2	(Advisory) The viable function set for a function call should either contain no function specializations, or only contain function specializations.	**Not implemented**	This rule requires manual checking.
15-0-1	(Document) Exceptions shall only be used for error handling.	**Document**	This rule requires manual documentation.
15-0-2	(Advisory) An exception object should not have pointer type.	**MISRAC++2008-15-0-2**	Throw of exceptions by pointer
15-0-3	(Required) Control shall not be transferred into a try or catch block using a goto or a switch statement.	**PARSE**	Parse errors are generated for this rule
15-1-1	(Required) The assignment-expression of a throw statement shall not itself cause an exception to be thrown.	**Not implemented**	This rule requires manual checking.
15-1-2	(Required) NULL shall not be thrown explicitly.	**MISRAC++2008-15-1-2**	Throw of NULL integer constant
15-1-3	(Required) An empty throw (throw;) shall only be used in the compound-statement of a catch handler.	**MISRAC++2008-15-1-3**	Unsafe rethrow of exception.
15-3-1	(Required) Exceptions shall be raised only after start-up and before termination of the program.	**MISRAC++2008-15-3-1**	Exceptions thrown without a handler in some call paths leading to that point
15-3-2	(Advisory) There should be at least one exception handler to catch all otherwise unhandled exceptions	**MISRAC++2008-15-3-2**	No default exception handler for try.
15-3-3	(Required) Handlers of a function-try-block implementation of a class constructor or destructor shall not reference non-static members from this class or its bases.	**MISRAC++2008-15-3-3**	Exception handler in constructor or destructor accesses non-static member variable that may not exist.
15-3-4	(Required) 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.	**MISRAC++2008-15-3-4**	Calls to functions explicitly declared to throw an exception type that is not handled (or declared as thrown) by the caller
15-3-5	(Required) A class type exception shall always be caught by reference.	**MISRAC++2008-15-3-5**	Catch of exception objects by value
15-3-6	(Required) Where multiple handlers are provided in a single try-catch statement or function-try-block for a derived class and some or all of its bases, the handlers shall be ordered most-derived to base class.	**Not implemented**	This rule requires manual checking.
15-3-7	(Required) Where multiple handlers are provided in a single try-catch statement or function-try-block, any ellipsis (catch-all) handler shall occur last.	**PARSE**	Parse errors are generated for this rule

15-4-1	(Required) If a function is declared with an exception-specification, then all declarations of the same function (in other translation units) shall be declared with the same set of type-ids.	**Not implemented**	This rule requires manual checking.
15-5-1	(Required) A class destructor shall not exit with an exception.	**MISRAC++2008-15-5-1**	An exception is thrown, or may be thrown, in a class' destructor.
15-5-2	(Required) Where a function's declaration includes an exception-specification, the function shall only be capable of throwing exceptions of the indicated type(s).	**Not implemented**	This rule requires manual checking.
15-5-3	(Required) The terminate() function shall not be called implicitly.	**Not implemented**	This rule requires manual checking.
16-0-1	(Required) #include directives in a file shall only be preceded by other preprocessor directives or comments.	**Not implemented**	This rule requires manual checking.
16-0-2	(Required) Macros shall only be #define'd or #undef'd in the global namespace.	**Not implemented**	This rule requires manual checking.
16-0-3	(Required) #undef shall not be used.	**MISRAC++2008-16-0-3**	All #undef's
16-0-4	(Required) Function-like macros shall not be defined.	**MISRAC++2008-16-0-4**	Function-like macros
16-0-5	(Required) Arguments to a function-like macro shall not contain tokens that look like preprocessing directives.	**Not implemented**	This rule requires manual checking.
16-0-6	(Required) 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 ##.	**Not implemented**	This rule requires manual checking.
16-0-7	(Required) Undefined macro identifiers shall not be used in #if or #elif preprocessor directives, except as operands to the defined operator.	**Not implemented**	This rule requires manual checking.
16-0-8	(Required) If the # token appears as the first token on a line, then it shall be immediately followed by a preprocessing token.	**Not implemented**	This rule requires manual checking.
16-1-1	(Required) The defined preprocessor operator shall only be used in one of the two standard forms.	**Not implemented**	This rule requires manual checking.
16-1-2	(Required) All #else, #elif and #endif preprocessor directives shall reside in the same file as the #if or #ifdef directive to which they are related.	**Not implemented**	This rule requires manual checking.
16-2-1	(Required) The pre-processor shall only be used for file inclusion and include guards.	**Not implemented**	This rule requires manual checking.
16-2-2	(Required) C++ macros shall only be used for: include guards, type qualifiers, or storage class specifiers.	**MISRAC++2008-16-2-2**	Definition of macros (except include guards)
16-2-3	(Required) Include guards shall be provided.	**MISRAC++2008-16-2-3**	Header files without #include guards
16-2-4	(Required) The ', ", /* or // characters shall not occur in a header file name.	**MISRAC++2008-16-2-4**	Illegal characters in header file names
16-2-5	(Advisory) The backslash character should not occur in a header file name.	**PARSE** **MISRAC++2008-16-2-5**	Parse errors are generated for this rule Illegal characters in header file names
16-2-6	(Required) The #include directive shall be followed by either a <filename> or "filename" sequence.	**PARSE**	Parse errors are generated for this rule

16-3-1	(Required) There shall be at most one occurrence of the # or ## operators in a single macro definition.	**MISRAC++2008-16-3-1**	Multiple # or ## operators in a macro definition
16-3-2	(Advisory) The # and ## operators should not be used.	**MISRAC++2008-16-3-2**	The # and ## operators should not be used
16-6-1	(Document) All uses of the #pragma directive shall be documented.	**Document**	This rule requires manual documentation.
17-0-1	(Required) Reserved identifiers, macros and functions in the standard library shall not be defined, redefined or undefined.	**MISRAC++2008-17-0-1**	#define or #undef of a reserved identifier in the standard library
17-0-2	(Required) The names of standard library macros and objects shall not be reused.	**Not implemented**	This rule requires manual checking.
17-0-3	(Required) The names of standard library functions shall not be overridden.	**MISRAC++2008-17-0-3**	A library function is being overridden.
17-0-4	(Document) All library code shall conform to MISRA C++.	**Document**	This rule requires manual documentation.
17-0-5	(Required) The setjmp macro and the longjmp function shall not be used.	**MISRAC++2008-17-0-5**	All uses of <setjmp.h>
18-0-1	(Required) The C library shall not be used.	**MISRAC++2008-18-0-1**	Uses of C library includes
18-0-2	(Required) The library functions atof, atoi and atol from library <cstdlib> shall not be used.	**MISRAC++2008-18-0-2**	All uses of atof, atoi, atol and atoll
18-0-3	(Required) The library functions abort, exit, getenv and system from library <cstdlib> shall not be used.	**MISRAC++2008-18-0-3**	All uses of abort, exit, getenv, and system
18-0-4	(Required) The time handling functions of library <ctime> shall not be used.	**MISRAC++2008-18-0-4**	All uses of <time.h> functions: asctime, clock, ctime, difftime, gmtime, localtime, mktime, strftime, and time
18-0-5	(Required) The unbounded functions of library <cstring> shall not be used.	**MISRAC++2008-18-0-5**	All uses of strcpy, strcmp, strcat, strchr, strspn, strcspn, strpbrk, strrchr, strstr, strtok, and strlen
18-2-1	(Required) The macro offsetof shall not be used.	**MISRAC++2008-18-2-1**	All uses of the offsetof built-in function
18-4-1	(Required) Dynamic heap memory allocation shall not be used.	**MISRAC++2008-18-4-1**	All uses of malloc, calloc, realloc, and free
18-7-1	(Required) The signal handling facilities of <csignal> shall not be used.	**MISRAC++2008-18-7-1**	All uses of <signal.h>
19-3-1	(Required) The error indicator errno shall not be used.	**MISRAC++2008-19-3-1**	All uses of errno
27-0-1	(Required) The stream input/output library <cstdio> shall not be used.	**MISRAC++2008-27-0-1**	All uses of <stdio.h>

For more information:
http://redlizards.com
mailto:info@redlizards.com