MISRA C 2004 Mapping to CodeSonar®

Category ID	Category Name	CodeSonar Class Mnemonic	CodeSonar Class Name	Relationship Type (category to class)
Misra2004:1.1	All code shall conform to ISO/IEC 9899:1990 "Programming languages C", amended and corrected by ISO/IEC 9899/COR1:1995, ISO/IEC 9899/AMD1:1995, and ISO/IEC 9899/COR2:1996	LANG.COMM.CPP	C++ Comment in C	closely mapped
Misra2004:1.1	All code shall conform to ISO/IEC 9899:1990 "Programming languages C", amended and corrected by ISO/IEC 9899/COR1:1995, ISO/IEC 9899/AMD1:1995, and ISO/IEC 9899/COR2:1996	LANG.EXT.GNU	GNU Extension	closely mapped
Misra2004:1.1	All code shall conform to ISO/IEC 9899:1990 "Programming languages C", amended and corrected by ISO/IEC 9899/COR1:1995, ISO/IEC 9899/AMD1:1995, and ISO/IEC 9899/COR2:1996	LANG.EXT.TYPEOF	GNU Typeof	closely mapped
Misra2004:1.1	All code shall conform to ISO/IEC 9899:1990 "Programming languages C", amended and corrected by ISO/IEC 9899/COR1:1995, ISO/IEC 9899/AMD1:1995, and ISO/IEC 9899/COR2:1996	LANG.EXT.MS	Microsoft Extension	closely mapped
Misra2004:2.1	Assembly language shall be encapsulated and isolated	LANG.ASM.MIXED	Mixed Assembly and Code	closely mapped
Misra2004:2.2	Source code shall only use /* ... */ style comments	LANG.COMM.CPP	C++ Comment in C	closely mapped
Misra2004:2.2	Source code shall only use /* ... */ style comments	LANG.EXT.GNU	GNU Extension	closely mapped
Misra2004:2.2	Source code shall only use /* ... */ style comments	LANG.EXT.TYPEOF	GNU Typeof	closely mapped
Misra2004:2.2	Source code shall only use /* ... */ style comments	LANG.EXT.MS	Microsoft Extension	closely mapped
Misra2004:2.3	The character sequence /* shall not be used within a comment	LANG.COMM.NEST.CSTYLE	/* in Comment	closely mapped
Misra2004:2.3	The character sequence /* shall not be used within a comment	LANG.COMM.NEST.CPPSTYLE	// in Comment	closely mapped
Misra2004:2.4	Sections of code should not be "commented out"	LANG.COMM.CODE	Commented-out Code	closely mapped
Misra2004:4.2	Trigraphs shall not be used	LANG.STRUCT.TRIGRAPH	Trigraph	closely mapped
Misra2004:5.1	Identifiers (internal and external) shall not rely on the significance of more than 31 characters	LANG.ID.ND.EXT	Non-distinct Identifiers: External Names	closely mapped
Misra2004:5.1	Identifiers (internal and external) shall not rely on the significance of more than 31 characters	LANG.ID.ND.MM	Non-distinct Identifiers: Macro/Macro	closely mapped
Misra2004:5.1	Identifiers (internal and external) shall not rely on the significance of more than 31 characters	LANG.ID.ND.MO	Non-distinct Identifiers: Macro/Other	closely mapped
Misra2004:5.1	Identifiers (internal and external) shall not rely on the significance of more than 31 characters	LANG.ID.ND.NEST	Non-distinct Identifiers: Nested Scope	closely mapped

Misra2004:5.1	Identifiers (internal and external) shall not rely on the significance of more than 31 characters	LANG.ID.ND.SS	Non-distinct Identifiers: Same Scope	closely mapped
Misra2004:5.2	Identifiers in an inner scope shall not use the same name as an identifier in an outer scope, and therefore hide that identifier	LANG.ID.ND.NEST	Non-distinct Identifiers: Nested Scope	closely mapped
Misra2004:5.3	A typedef name shall be a unique identifier	LANG.ID.NU.TYPE	Non-unique Identifiers: Typedef	closely mapped
Misra2004:5.4	A tag name shall be a unique identifier	LANG.ID.NU.TAG	Non-unique Identifiers: Tag	closely mapped
Misra2004:5.5	No object or function identifier with static storage duration should be reused	LANG.ID.NU.EXT	Non-unique Identifiers: External Name	closely mapped
Misra2004:5.5	No object or function identifier with static storage duration should be reused	LANG.ID.NU.INT	Non-unique Identifiers: Internal Name	closely mapped
Misra2004:6.1	The plain char type shall be used only for storage and use of character values	LANG.TYPE.IAT	Inappropriate Assignment Type	closely mapped
Misra2004:6.1	The plain char type shall be used only for storage and use of character values	LANG.TYPE.ICA	Inappropriate Character Arithmetic	closely mapped
Misra2004:6.1	The plain char type shall be used only for storage and use of character values	LANG.TYPE.IOT	Inappropriate Operand Type	closely mapped
Misra2004:6.1	The plain char type shall be used only for storage and use of character values	LANG.TYPE.MOT	Mismatched Operand Types	closely mapped
Misra2004:6.2	signed and unsigned char type shall be used only for the storage and use of numeric values	LANG.TYPE.IAT	Inappropriate Assignment Type	closely mapped
Misra2004:6.2	signed and unsigned char type shall be used only for the storage and use of numeric values	LANG.TYPE.IOT	Inappropriate Operand Type	closely mapped
Misra2004:6.2	signed and unsigned char type shall be used only for the storage and use of numeric values	LANG.TYPE.MOT	Mismatched Operand Types	closely mapped
Misra2004:6.3	typedefs that indicate size and signedness should be used in place of the basic numerical types	LANG.TYPE.BASIC	Basic Numerical Type Used	closely mapped
Misra2004:6.4	Bit fields shall only be defined to be of type unsigned int or signed int	LANG.TYPE.BFSIGN	Bit-field Signedness Not Explicit	closely mapped
Misra2004:6.4	Bit fields shall only be defined to be of type unsigned int or signed int	LANG.TYPE.BFINT	Inappropriate Bit-field Type	closely mapped
Misra2004:6.5	Bit fields of signed type shall be at least 2 bits long	LANG.TYPE.BFSHORT	Bit-field Too Short	closely mapped
Misra2004:7.1	Octal constants (other than zero) and octal escape sequences shall not be used	LANG.TYPE.OC	Octal Constant	closely mapped
Misra2004:8.1	Functions shall have prototype declarations and the prototype shall be visible at both the function definition and call	LANG.STRUCT.DECL.IMPFN	Implicit Function Declaration	closely mapped

MISRA C 2004 Mapping to CodeSonar®

Misra2004:8.1	Functions shall have prototype declarations and the prototype shall be visible at both the function definition and call	LANG.FUNCS.PROT	Incomplete Function Prototype	closely mapped
Misra2004:8.3	For each function parameter the type given in the declaration and definition shall be identical, and the return types shall also be identical	LANG.STRUCT.DECL.MGT	Global Variable Declared with Different Types	closely mapped
Misra2004:8.3	For each function parameter the type given in the declaration and definition shall be identical, and the return types shall also be identical	LANG.STRUCT.DECL.IF	Inconsistent Function Declarations	closely mapped
Misra2004:8.3	For each function parameter the type given in the declaration and definition shall be identical, and the return types shall also be identical	LANG.STRUCT.DECL.IO	Inconsistent Object Declarations	closely mapped
Misra2004:8.4	If objects or functions are declared more than once their types shall be compatible	LANG.STRUCT.DECL.MGT	Global Variable Declared with Different Types	closely mapped
Misra2004:8.4	If objects or functions are declared more than once their types shall be compatible	LANG.STRUCT.DECL.IF	Inconsistent Function Declarations	closely mapped
Misra2004:8.4	If objects or functions are declared more than once their types shall be compatible	LANG.STRUCT.DECL.IO	Inconsistent Object Declarations	closely mapped
Misra2004:8.5	There shall be no definitions of objects or functions in a header file	LANG.STRUCT.DEF.FDH	Function Defined in Header File	closely mapped
Misra2004:8.5	There shall be no definitions of objects or functions in a header file	LANG.STRUCT.DEF.ODH	Object Defined in Header File	closely mapped
Misra2004:8.6	Functions shall be declared at file scope	LANG.STRUCT.DECL.FNEST	Nested Function Declaration	closely mapped
Misra2004:8.7	Objects shall be defined at block scope if they are only accessed from within a single function	LANG.STRUCT.SCOPE.LOCAL	Scope Could Be Local Static	closely mapped
Misra2004:8.8	An external object or function shall be declared in one and only one file	LANG.STRUCT.DECL.NOEXT	Missing External Declaration	closely mapped
Misra2004:8.8	An external object or function shall be declared in one and only one file	LANG.STRUCT.DECL.MG	Multiple Declarations of a Global	closely mapped
Misra2004:8.8	An external object or function shall be declared in one and only one file	LANG.STRUCT.DECL.MULTIEXT	Multiple External Declarations	closely mapped
Misra2004:8.9	An identifier with external linkage shall have exactly one external definition	LANG.STRUCT.DEF.NOEXT	Missing External Definition	closely mapped
Misra2004:8.9	An identifier with external linkage shall have exactly one external definition	LANG.STRUCT.DEF.MULTIEXT	Multiple External Definitions	closely mapped
Misra2004:8.10	All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required	LANG.STRUCT.SCOPE.FILE	Scope Could Be File Static	closely mapped
Misra2004:8.10	All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required	LANG.STRUCT.SCOPE.LOCAL	Scope Could Be Local Static	closely mapped

Misra2004:8.11	The static storage class specifier shall be used in definitions and declarations of objects and functions that have internal linkage	LANG.STRUCT.SCOPE.FILE	Scope Could Be File Static	closely mapped
Misra2004:8.11	The static storage class specifier shall be used in definitions and declarations of objects and functions that have internal linkage	LANG.STRUCT.SCOPE.LOCAL	Scope Could Be Local Static	closely mapped
Misra2004:8.12	When an array is declared with external linkage, its size shall be stated explicitly or defined implicitly by initialisation	LANG.STRUCT.DECL.EAWS	Extern Array Without Size	closely mapped
Misra2004:9.1	All automatic variables shall have been assigned a value before being used	LANG.MEM.UVAR	Uninitialized Variable	closely mapped
Misra2004:9.2	Braces shall be used to indicate and match the structure in the non-zero initialisation of arrays and structures	LANG.STRUCT.INIT.MBI	Missing Braces in Initialization	closely mapped
Misra2004:9.2	Braces shall be used to indicate and match the structure in the non-zero initialisation of arrays and structures	LANG.STRUCT.INIT.PIARR	Partially Uninitialized Array	closely mapped
Misra2004:9.3	In an enumerator list, the "=" construct shall not be used to explicitly initialise members other than the first, unless all items are explicitly initialised	LANG.STRUCT.INIT.ENUM	Inconsistent Enumerator Initialization	closely mapped
Misra2004:10.1	The value of an expression of integer type shall not be implicitly converted to a different underlying type if: (a) it is not a conversion to a wider integer type of the same signedness, or (b) the expression is complex, or (c) the expression is not constant and is a function argument, or (d) the expression is not constant and is a return expression	LANG.TYPE.AWID	Expression Value Widened by Assignment	closely mapped
Misra2004:10.1	The value of an expression of integer type shall not be implicitly converted to a different underlying type if: (a) it is not a conversion to a wider integer type of the same signedness, or (b) the expression is complex, or (c) the expression is not constant and is a function argument, or (d) the expression is not constant and is a return expression	LANG.TYPE.OWID	Expression Value Widened by Other Operand	closely mapped
Misra2004:10.1	The value of an expression of integer type shall not be implicitly converted to a different underlying type if: (a) it is not a conversion to a wider integer type of the same signedness, or (b) the expression is complex, or (c) the expression is not constant and is a function argument, or (d) the expression is not constant and is a return expression	LANG.TYPE.IAT	Inappropriate Assignment Type	closely mapped

Misra2004:10.1	The value of an expression of integer type shall not be implicitly converted to a different underlying type if: (a) it is not a conversion to a wider integer type of the same signedness, or (b) the expression is complex, or (c) the expression is not constant and is a function argument, or (d) the expression is not constant and is a return expression	LANG.TYPE.IOT	Inappropriate Operand Type	hierarchy ancestor
Misra2004:10.1	The value of an expression of integer type shall not be implicitly converted to a different underlying type if: (a) it is not a conversion to a wider integer type of the same signedness, or (b) the expression is complex, or (c) the expression is not constant and is a function argument, or (d) the expression is not constant and is a return expression	LANG.TYPE.MOT	Mismatched Operand Types	closely mapped
Misra2004:10.2	The value of an expression of floating type shall not be implicitly converted to a different type if: (a) it is not a conversion to a wider floating type, or (b) the expression is complex, or (c) the expression is a function argument, or (d) the expression is a return expression	LANG.TYPE.AWID	Expression Value Widened by Assignment	closely mapped
Misra2004:10.2	The value of an expression of floating type shall not be implicitly converted to a different type if: (a) it is not a conversion to a wider floating type, or (b) the expression is complex, or (c) the expression is a function argument, or (d) the expression is a return expression	LANG.TYPE.OWID	Expression Value Widened by Other Operand	closely mapped
Misra2004:10.2	The value of an expression of floating type shall not be implicitly converted to a different type if: (a) it is not a conversion to a wider floating type, or (b) the expression is complex, or (c) the expression is a function argument, or (d) the expression is a return expression	LANG.TYPE.IAT	Inappropriate Assignment Type	closely mapped
Misra2004:10.2	The value of an expression of floating type shall not be implicitly converted to a different type if: (a) it is not a conversion to a wider floating type, or (b) the expression is complex, or (c) the expression is a function argument, or (d) the expression is a return expression	LANG.TYPE.IOT	Inappropriate Operand Type	hierarchy descendant

Misra2004:10.2	The value of an expression of floating type shall not be implicitly converted to a different type if: (a) it is not a conversion to a wider floating type, or (b) the expression is complex, or (c) the expression is a function argument, or (d) the expression is a return expression	LANG.TYPE.MOT	Mismatched Operand Types	closely mapped
Misra2004:10.3	The value of a complex expression of integer type shall only be cast to a type of the same signedness that is no wider than the underlying type of the expression	LANG.TYPE.ICTE	Inappropriate Cast Type: Expression	closely mapped
Misra2004:10.4	The value of a complex expression of floating type shall only be cast to a floating type that is narrower or of the same size	LANG.TYPE.ICTE	Inappropriate Cast Type: Expression	closely mapped
Misra2004:10.5	If the bitwise operators ~ and << are applied to an operand of underlying type unsigned char or unsigned short, the result shall be immediately cast to the underlying type of the operand	LANG.CAST.RIP	Risky Integer Promotion	closely mapped
Misra2004:10.6	A "U" suffix shall be applied to all constants of unsigned type	LANG.TYPE.MSUF	Missing Literal Suffix	closely mapped
Misra2004:11.1	Conversions shall not be performed between a pointer to a function and any type other than an integral type	LANG.CAST.FN	Dangerous Function Cast	closely mapped
Misra2004:11.1	Conversions shall not be performed between a pointer to a function and any type other than an integral type	LANG.STRUCT.FUNCPTR.CONVERT	Function Pointer Conversion	closely mapped
Misra2004:11.2	Conversions shall not be performed between a pointer to object and any type other than an integral type, another pointer to object type or a pointer to void	LANG.CAST.PC.AO	Cast: Non-integer Arithmetic Type/Object Pointer	closely mapped
Misra2004:11.2	Conversions shall not be performed between a pointer to object and any type other than an integral type, another pointer to object type or a pointer to void	LANG.CAST.PC.INC	Conversion: Pointer to Incomplete	closely mapped
Misra2004:11.2	Conversions shall not be performed between a pointer to object and any type other than an integral type, another pointer to object type or a pointer to void	LANG.CAST.PC.PV	Conversion: Void Pointer to Object Pointer	closely mapped
Misra2004:11.2	Conversions shall not be performed between a pointer to object and any type other than an integral type, another pointer to object type or a pointer to void	LANG.STRUCT.FUNCPTR.CONVERT	Function Pointer Conversion	closely mapped
Misra2004:11.3	A cast should not be performed between a pointer type and an integral type	LANG.CAST.PC.AV	Cast: Arithmetic Type/Void Pointer	closely mapped
Misra2004:11.3	A cast should not be performed between a pointer type and an integral type	LANG.CAST.PC.INC	Conversion: Pointer to Incomplete	closely mapped

Misra2004:11.3	A cast should not be performed between a pointer type and an integral type	LANG.CAST.PC.INT	Conversion: Pointer/Integer	closely mapped
Misra2004:11.3	A cast should not be performed between a pointer type and an integral type	LANG.STRUCT.FUNCPTR.CONVERT	Function Pointer Conversion	closely mapped
Misra2004:11.4	A cast should not be performed between a pointer to object type and a different pointer to object type	LANG.CAST.PC.OBJ	Cast: Object Pointers	closely mapped
Misra2004:11.5	A cast shall not be performed that removes any const or volatile qualification from the type addressed by a pointer	LANG.CAST.PC.CRCQ	Cast Removes const Qualifier	closely mapped
Misra2004:11.5	A cast shall not be performed that removes any const or volatile qualification from the type addressed by a pointer	LANG.CAST.PC.CRVQ	Cast Removes volatile Qualifier	closely mapped
Misra2004:12.3	The sizeof operator shall not be used on expressions that contain side effects	LANG.STRUCT.SE.SIZEOF	Side Effects in sizeof	closely mapped
Misra2004:12.6	The operands of logical operators (&&, || and !) should be effectively Boolean. Expressions that are effectively Boolean should not be used as operands to operators other than (&&, || , !, =, ==, != and ?:)	LANG.TYPE.AWID	Expression Value Widened by Assignment	hierarchy descendant
Misra2004:12.6	The operands of logical operators (&&, || and !) should be effectively Boolean. Expressions that are effectively Boolean should not be used as operands to operators other than (&&, || , !, =, ==, != and ?:)	LANG.TYPE.AWID	Expression Value Widened by Assignment	hierarchy ancestor
Misra2004:12.6	The operands of logical operators (&&, || and !) should be effectively Boolean. Expressions that are effectively Boolean should not be used as operands to operators other than (&&, || , !, =, ==, != and ?:)	LANG.TYPE.OWID	Expression Value Widened by Other Operand	hierarchy descendant
Misra2004:12.6	The operands of logical operators (&&, || and !) should be effectively Boolean. Expressions that are effectively Boolean should not be used as operands to operators other than (&&, || , !, =, ==, != and ?:)	LANG.TYPE.OWID	Expression Value Widened by Other Operand	hierarchy ancestor
Misra2004:12.6	The operands of logical operators (&&, || and !) should be effectively Boolean. Expressions that are effectively Boolean should not be used as operands to operators other than (&&, || , !, =, ==, != and ?:)	LANG.TYPE.IAT	Inappropriate Assignment Type	hierarchy descendant
Misra2004:12.6	The operands of logical operators (&&, || and !) should be effectively Boolean. Expressions that are effectively Boolean should not be used as operands to operators other than (&&, || , !, =, ==, != and ?:)	LANG.TYPE.IAT	Inappropriate Assignment Type	hierarchy ancestor

Misra2004:12.6	The operands of logical operators (&&, || and !) should be effectively Boolean. Expressions that are effectively Boolean should not be used as operands to operators other than (&&, || , !, =, ==, != and ?:)	LANG.TYPE.IOT	Inappropriate Operand Type	closely mapped
Misra2004:12.6	The operands of logical operators (&&, || and !) should be effectively Boolean. Expressions that are effectively Boolean should not be used as operands to operators other than (&&, || , !, =, ==, != and ?:)	LANG.TYPE.MOT	Mismatched Operand Types	hierarchy descendant
Misra2004:12.6	The operands of logical operators (&&, || and !) should be effectively Boolean. Expressions that are effectively Boolean should not be used as operands to operators other than (&&, || , !, =, ==, != and ?:)	LANG.TYPE.MOT	Mismatched Operand Types	hierarchy ancestor
Misra2004:12.7	Bitwise operators shall not be applied to operands whose underlying type is signed	LANG.TYPE.IOT	Inappropriate Operand Type	closely mapped
Misra2004:12.8	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	LANG.ARITH.NEGSHIFT	Negative Shift Amount	closely mapped
Misra2004:12.8	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	LANG.ARITH.BIGSHIFT	Shift Amount Exceeds Bit Width	closely mapped
Misra2004:12.9	The unary minus operator shall not be applied to an expression whose underlying type is unsigned	LANG.TYPE.IOT	Inappropriate Operand Type	closely mapped
Misra2004:12.10	The comma operator shall not be used	LANG.STRUCT.COMMA	Use of Comma Operator	closely mapped
Misra2004:12.13	The increment (++) and decrement (--) operators should not be mixed with other operators in an expression	LANG.STRUCT.SE.DEC	Side Effects in Expression with Decrement	closely mapped
Misra2004:12.13	The increment (++) and decrement (--) operators should not be mixed with other operators in an expression	LANG.STRUCT.SE.INC	Side Effects in Expression with Increment	closely mapped
Misra2004:13.2	Tests of a value against zero should be made explicit, unless the operand is effectively Boolean	LANG.STRUCT.NBC	Condition Is Not Boolean	closely mapped
Misra2004:13.4	The controlling expression of a for statement shall not contain any objects of floating type	LANG.STRUCT.LOOP.FPC	Float-typed Loop Counter	closely mapped
Misra2004:13.5	The three expressions of a for statement shall be concerned only with loop control	LANG.STRUCT.LOOP.MFTERM	Malformed for-loop Condition	closely mapped
Misra2004:13.5	The three expressions of a for statement shall be concerned only with loop control	LANG.STRUCT.LOOP.MFINIT	Malformed for-loop Initialization	closely mapped
Misra2004:13.5	The three expressions of a for statement shall be concerned only with loop control	LANG.STRUCT.LOOP.NOSTEP	Missing for-loop Step	closely mapped

Misra2004:13.5	The three expressions of a for statement shall be concerned only with loop control	LANG.STRUCT.LOOP.NOTERM	Missing for-loop Termination	closely mapped
Misra2004:13.6	Numeric variables being used within a for loop for iteration counting shall not be modified in the body of the loop	LANG.STRUCT.LOOP.MFTERM	Malformed for-loop Condition	closely mapped
Misra2004:13.6	Numeric variables being used within a for loop for iteration counting shall not be modified in the body of the loop	LANG.STRUCT.LOOP.MFINIT	Malformed for-loop Initialization	closely mapped
Misra2004:13.6	Numeric variables being used within a for loop for iteration counting shall not be modified in the body of the loop	LANG.STRUCT.LOOP.NOSTEP	Missing for-loop Step	closely mapped
Misra2004:13.6	Numeric variables being used within a for loop for iteration counting shall not be modified in the body of the loop	LANG.STRUCT.LOOP.NOTERM	Missing for-loop Termination	closely mapped
Misra2004:13.7	Boolean operations whose results are invariant shall not be permitted	LANG.STRUCT.RC	Redundant Condition	closely mapped
Misra2004:14.1	There shall be no unreachable code	LANG.STRUCT.UC	Unreachable Call	closely mapped
Misra2004:14.1	There shall be no unreachable code	LANG.STRUCT.UC	Unreachable Computation	closely mapped
Misra2004:14.1	There shall be no unreachable code	LANG.STRUCT.UC	Unreachable Conditional	closely mapped
Misra2004:14.1	There shall be no unreachable code	LANG.STRUCT.UC	Unreachable Control Flow	closely mapped
Misra2004:14.1	There shall be no unreachable code	LANG.STRUCT.UC	Unreachable Data Flow	closely mapped
Misra2004:14.2	All non-null statements shall either (a) have at least one side-effect however executed, or (b) cause control flow to change	MISC.NOEFFECT	Function Call Has No Effect	closely mapped
Misra2004:14.2	All non-null statements shall either (a) have at least one side-effect however executed, or (b) cause control flow to change	LANG.STRUCT.UUVAL	Unused Value	closely mapped
Misra2004:14.2	All non-null statements shall either (a) have at least one side-effect however executed, or (b) cause control flow to change	LANG.STRUCT.UA	Useless Assignment	closely mapped
Misra2004:14.3	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	LANG.STRUCT.BNC	Body Is Not Compound Statement	closely mapped
Misra2004:14.4	The goto statement shall not be used	LANG.STRUCT.BGOTO	Backwards goto	closely mapped
Misra2004:14.4	The goto statement shall not be used	LANG.STRUCT.GOTO	Goto Statement	closely mapped
Misra2004:14.4	The goto statement shall not be used	LANG.STRUCT.GLABEL	Label Not In Enclosing Block	closely mapped
Misra2004:14.5	The continue statement shall not be used	LANG.STRUCT.CONTINUE	Continue Statement	closely mapped

Misra2004:14.6	For any iteration statement there shall be at most one break statement used for loop termination	LANG.STRUCT.LOOP.MAE	Multiple Abnormal Loop Exits	closely mapped
Misra2004:14.7	A function shall have a single point of exit at the end of the function	LANG.STRUCT.MISRS	Misplaced Return Statement	closely mapped
Misra2004:14.7	A function shall have a single point of exit at the end of the function	LANG.STRUCT.MULRS	Multiple Return Statements	closely mapped
Misra2004:14.8	The statement forming the body of a switch, while, do ... while or for statement shall be a compound statement	LANG.STRUCT.BNC	Body Is Not Compound Statement	closely mapped
Misra2004:14.9	An if (expression) construct shall be followed by a compound statement. The else keyword shall be followed by either a compound statement, or another if statement	LANG.STRUCT.BNC	Body Is Not Compound Statement	closely mapped
Misra2004:14.10	All if . else if constructs shall be terminated with an else clause	LANG.STRUCT.NOELSE	Missing Final else	closely mapped
Misra2004:15.0	The MISRA C switch syntax shall be used	LANG.STRUCT.SW.BAD	Malformed switch Statement	closely mapped
Misra2004:15.2	An unconditional break statement shall terminate every non-empty switch clause	LANG.STRUCT.SW.MB	Missing break	closely mapped
Misra2004:15.3	The final clause of a switch statement shall be the default clause	LANG.STRUCT.SW.MPD	Misplaced default	closely mapped
Misra2004:15.3	The final clause of a switch statement shall be the default clause	LANG.STRUCT.SW.MD	Missing default	closely mapped
Misra2004:15.4	A switch expression shall not represent a value that is effectively Boolean	LANG.STRUCT.SW.BOOL	Boolean switch Expression	closely mapped
Misra2004:15.5	Every switch statement shall have at least one case clause	LANG.STRUCT.SW.IF	Too Few Cases in switch	closely mapped
Misra2004:16.2	Functions shall not call themselves, either directly or indirectly	LANG.FUNCS.RECURSION	Recursion	closely mapped
Misra2004:16.3	Identifiers shall be given for all of the parameters in a function prototype declaration	LANG.FUNCS.PROT	Incomplete Function Prototype	closely mapped
Misra2004:16.4	The identifiers used in the declaration and definition of a function shall be identical	LANG.STRUCT.DECL.MGT	Global Variable Declared with Different Types	closely mapped
Misra2004:16.4	The identifiers used in the declaration and definition of a function shall be identical	LANG.STRUCT.DECL.IF	Inconsistent Function Declarations	closely mapped
Misra2004:16.4	The identifiers used in the declaration and definition of a function shall be identical	LANG.STRUCT.DECL.IO	Inconsistent Object Declarations	closely mapped
Misra2004:16.5	Functions with no parameters shall be declared and defined with the parameter list void	LANG.FUNCS.PROT	Incomplete Function Prototype	closely mapped
Misra2004:16.6	The number of arguments passed to a function shall match the number of parameters	LANG.STRUCT.DECL.IMPFN	Implicit Function Declaration	also related
Misra2004:16.6	The number of arguments passed to a function shall match the number of parameters	LANG.FUNCS.PROT	Incomplete Function Prototype	closely mapped
Misra2004:16.8	All exit paths from a function with non-void return type shall have an explicit return statement with an expression	LANG.STRUCT.MRS	Missing Return Statement	closely mapped

Misra2004:16.8	All exit paths from a function with non-void return type shall have an explicit return statement with an expression	LANG.FUNCS.MRV	Missing Return Value	closely mapped
Misra2004:16.9	A function identifier shall only be used with either a preceding &, or with a parenthesised parameter list, which may be empty	LANG.STRUCT.FNADDR	Implicit Address of Function	closely mapped
Misra2004:16.10	If a function returns error information, then that error information shall be tested	LANG.FUNCS.IRV	Ignored Return Value	closely mapped
Misra2004:17.1	Pointer arithmetic shall only be applied to pointers that address an array or array element	LANG.MEM.BO	Buffer Overrun	closely mapped
Misra2004:17.1	Pointer arithmetic shall only be applied to pointers that address an array or array element	LANG.MEM.BU	Buffer Underrun	closely mapped
Misra2004:17.1	Pointer arithmetic shall only be applied to pointers that address an array or array element	LANG.STRUCT.PBB	Pointer Before Beginning of Object	closely mapped
Misra2004:17.1	Pointer arithmetic shall only be applied to pointers that address an array or array element	LANG.STRUCT.PPE	Pointer Past End of Object	closely mapped
Misra2004:17.1	Pointer arithmetic shall only be applied to pointers that address an array or array element	LANG.MEM.TBA	Tainted Buffer Access	closely mapped
Misra2004:17.1	Pointer arithmetic shall only be applied to pointers that address an array or array element	LANG.MEM.TO	Type Overrun	closely mapped
Misra2004:17.1	Pointer arithmetic shall only be applied to pointers that address an array or array element	LANG.MEM.TU	Type Underrun	closely mapped
Misra2004:17.2	Pointer subtraction shall only be applied to pointers that address elements of the same array	LANG.MEM.BO	Buffer Overrun	closely mapped
Misra2004:17.2	Pointer subtraction shall only be applied to pointers that address elements of the same array	LANG.MEM.BU	Buffer Underrun	closely mapped
Misra2004:17.2	Pointer subtraction shall only be applied to pointers that address elements of the same array	LANG.STRUCT.PBB	Pointer Before Beginning of Object	closely mapped
Misra2004:17.2	Pointer subtraction shall only be applied to pointers that address elements of the same array	LANG.STRUCT.PPE	Pointer Past End of Object	closely mapped
Misra2004:17.2	Pointer subtraction shall only be applied to pointers that address elements of the same array	LANG.MEM.TBA	Tainted Buffer Access	closely mapped
Misra2004:17.2	Pointer subtraction shall only be applied to pointers that address elements of the same array	LANG.MEM.TO	Type Overrun	closely mapped
Misra2004:17.2	Pointer subtraction shall only be applied to pointers that address elements of the same array	LANG.MEM.TU	Type Underrun	closely mapped
Misra2004:18.2	An object shall not be assigned to an overlapping object	MISC.MEM.OR	Overlapping Memory Regions	closely mapped
Misra2004:18.4	Unions shall not be used	LANG.TYPE.UNION	Union Type	closely mapped

MISRA C 2004 Mapping to CodeSonar®

Misra2004:19.1	#include statements in a file should only be preceded by other preprocessor directives or comments	LANG.PREPROC.CBI	Code Before #include	closely mapped
Misra2004:19.2	Non-standard characters should not occur in header file names in #include directives	LANG.PREPROC.INCL.FNAME	Dangerous Include File Name	closely mapped
Misra2004:19.3	The #include directive shall be followed by either a <filename> or "filename" sequence	LANG.PREPROC.INCL.MF	Malformed #include	closely mapped
Misra2004:19.4	C macros shall only expand to a braced initialiser, a constant, a string literal, a parenthesised expression, a type qualifier, a storage class specifier, or a do-whilezero construct	LANG.ID.NU.MK	Macro Name is C Keyword	closely mapped
Misra2004:19.5	Macros shall not be #define'd or #undef'd within a block	LANG.PREPROC.DEFINFN	Macro Defined in Function Body	closely mapped
Misra2004:19.5	Macros shall not be #define'd or #undef'd within a block	LANG.PREPROC.UNDEFINFN	Macro Undefined in Function Body	closely mapped
Misra2004:19.6	#undef shall not be used	LANG.PREPROC.RUNDEF	Macro Undefinition of Reserved Name	closely mapped
Misra2004:19.6	#undef shall not be used	LANG.PREPROC.UNDEF	Use of #undef	closely mapped
Misra2004:19.9	Arguments to a function-like macro shall not contain tokens that look like preprocessing directives	LANG.PREPROC.MACROARG	Preprocessing Directives in Macro Argument	closely mapped
Misra2004:19.13	The # and ## preprocessor operators should not be used	LANG.PREPROC.PASTE	Macro Uses ## Operator	closely mapped
Misra2004:19.17	All #else, #elif and #endif preprocessor directives shall reside in the same file as the #if or #ifdef directive to which they are related	LANG.PREPROC.NOENDIF	No Matching #endif	closely mapped
Misra2004:19.17	All #else, #elif and #endif preprocessor directives shall reside in the same file as the #if or #ifdef directive to which they are related	LANG.PREPROC.NOIF	No Matching #if	closely mapped
Misra2004:20.1	Reserved identifiers, macros and functions in the standard library, shall not be defined, redefined or undefined	LANG.PREPROC.RDEF	Macro Definition of Reserved Name	closely mapped
Misra2004:20.1	Reserved identifiers, macros and functions in the standard library, shall not be defined, redefined or undefined	LANG.PREPROC.RUNDEF	Macro Undefinition of Reserved Name	closely mapped
Misra2004:20.4	Dynamic heap memory allocation shall not be used	LANG.CAST.COERCE	Coercion Alters Value	closely mapped
Misra2004:20.4	Dynamic heap memory allocation shall not be used	ALLOC.POSTINIT	Dynamic Allocation After Initialization	closely mapped
Misra2004:20.6	The macro offsetof, in library <stddef.h>, shall not be used	BADMACRO.OFFSETOF	Use of offsetof	closely mapped
Misra2004:20.7	The setjmp macro and the longjmp function shall not be used	LANG.PREPROC.INCL.SETJMP_H	Use of <setjmp.h>	closely mapped
Misra2004:20.7	The setjmp macro and the longjmp function shall not be used	BADFUNC.LONGJMP	Use of longjmp	closely mapped
Misra2004:20.7	The setjmp macro and the longjmp function shall not be used	BADFUNC.SETJMP	Use of setjmp	closely mapped

Misra2004:20.8	The signal handling facilities of <signal.h> shall not be used	LANG.PREPROC.INCL.SIGNAL_H	Use of <signal.h>	closely mapped
Misra2004:20.9	The input/output library <stdio.h> shall not be used in production code	BADFUNC.STDIO_H	Use of <stdio.h> Input/Output	closely mapped
Misra2004:20.9	The input/output library <stdio.h> shall not be used in production code	BADFUNC.WCHAR_H	Use of <wchar.h> Input/Output	closely mapped
Misra2004:20.10	The library functions atof, atoi and atol from library <stdlib.h> shall not be used	BADFUNC.ATOF	Use of atof	closely mapped
Misra2004:20.10	The library functions atof, atoi and atol from library <stdlib.h> shall not be used	BADFUNC.ATOI	Use of atoi	closely mapped
Misra2004:20.10	The library functions atof, atoi and atol from library <stdlib.h> shall not be used	BADFUNC.ATOL	Use of atol	closely mapped
Misra2004:20.10	The library functions atof, atoi and atol from library <stdlib.h> shall not be used	BADFUNC.ATOLL	Use of atoll	closely mapped
Misra2004:20.11	The library functions abort, exit, getenv and system from library <stdlib.h> shall not be used	BADFUNC.ABORT	Use of abort	closely mapped
Misra2004:20.11	The library functions abort, exit, getenv and system from library <stdlib.h> shall not be used	BADFUNC.EXIT	Use of exit	closely mapped
Misra2004:20.11	The library functions abort, exit, getenv and system from library <stdlib.h> shall not be used	BADFUNC.GETENV	Use of getenv	closely mapped
Misra2004:20.11	The library functions abort, exit, getenv and system from library <stdlib.h> shall not be used	BADFUNC.PATH.SYSTEM	Use of system	closely mapped
Misra2004:20.12	The time handling functions of library <time.h> shall not be used	BADFUNC.TIME_H	Use of <time.h> Time/Date Function	closely mapped