



MISRA-C 2004 GUIDELINES FOR THE USE OF THE C LANGUAGE IN CRITICAL SYSTEMS | CODESONAR® 7.3



TRUSTED LEADERS OF SOFTWARE ASSURANCE AND ADVANCED CYBER-SECURITY SOLUTIONS

WWW.GRAMMATECH.COM

INTRODUCTION

The MISRA C:2004 standard aims to foster safety, reliability, and portability of programs written in ISO C for embedded systems. It is used in a wide range of industries, including automotive, aero-space, medical devices, and industrial control.

CodeSonar 7.3 includes a large number of warning classes that support checking for the MISRA C:2004 guidelines. Every CodeSonar warning report includes the numbers of any MISRA C:2004 rules that are closely mapped to the warning's class. (The close mapping for a warning class is the set of categories—including MISRA C:2004 rule and directive numbers—that most closely match the class, if any).

You can configure CodeSonar to enable and disable warning classes mapped to specific MISRA C:2004 rules, or use build presets to enable all warning classes that are closely mapped to any MISRA C:2004 rules and directives. In addition, you can use the CodeSonar search function to find warnings related to specific MISRA C:2004 rules, or to any MISRA C:2004 rule.

For more information on MISRA C:

<https://www.misra.org.uk/MISRAHome/tabid/181/Default.aspx>



MISRA C:2004 CLOSE MAPPING (CODESONAR V7.3)

The following table contains CodeSonar classes that are closely mapped to specific MISRA C:2004 rules and directives.

Rule	Rule Name	Category	Supported
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	Required	Yes
Misra2004:1.2	No reliance shall be placed on undefined or unspecified behaviour	Required	No
Misra2004:1.3	Multiple compilers and/or languages shall only be used if there is a common defined interface standard for object code to which the languages/compilers/assemblers conform	Required	No
Misra2004:1.4	The compiler/linker shall be checked to ensure that 31 character significance and case sensitivity are supported for external identifiers	Required	No
Misra2004:1.5	Floating-point implementations should comply with a defined floating-point standard	Advisory	No
Misra2004:2.1	Assembly language shall be encapsulated and isolated	Required	Yes
Misra2004:2.2	Source code shall only use /* ... */ style comments	Required	Yes
Misra2004:2.3	The character sequence /* shall not be used within a comment	Required	Yes
Misra2004:2.4	Sections of code should not be "commented out"	Advisory	Yes
Misra2004:3.1	All usage of implementation-defined behaviour shall be documented	Required	No
Misra2004:3.2	The character set and the corresponding encoding shall be documented	Required	No
Misra2004:3.3	The implementation of integer division in the chosen compiler should be determined, documented and taken into account	Advisory	No
Misra2004:3.4	All uses of the #pragma directive shall be documented and explained	Required	No
Misra2004:3.5	The implementation defined behaviour and packing of bitfields shall be documented if being relied upon	Required	No
Misra2004:3.6	All libraries used in production code shall be written to comply with the provisions of this document, and shall have been subject to appropriate validation	Required	No
Misra2004:4.1	Only those escape sequences that are defined in the ISO C standard shall be used	Required	No
Misra2004:4.2	Trigraphs shall not be used	Required	Yes
Misra2004:5.1	Identifiers (internal and external) shall not rely on the significance of more than 31 characters	Required	Yes
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	Required	Yes
Misra2004:5.3	A typedef name shall be a unique identifier	Required	Yes
Misra2004:5.4	A tag name shall be a unique identifier	Required	Yes
Misra2004:5.5	No object or function identifier with static storage duration should be reused	Advisory	Yes
Misra2004:5.6	No identifier in one name space should have the same spelling as an identifier in another name space, with the exception of structure member and union member names	Advisory	No
Misra2004:5.7	No identifier name should be reused	Advisory	Yes
Misra2004:6.1	The plain char type shall be used only for storage and use of character values	Required	Yes
Misra2004:6.2	signed and unsigned char type shall be used only for the storage and use of numeric values	Required	Yes
Misra2004:6.3	typedefs that indicate size and signedness should be used in place of the basic numerical types	Advisory	Yes
Misra2004:6.4	Bit fields shall only be defined to be of type unsigned int or signed int	Required	Yes
Misra2004:6.5	Bit fields of signed type shall be at least 2 bits long	Required	Yes
Misra2004:7.1	Octal constants (other than zero) and octal escape sequences shall not be used	Required	Yes
Misra2004:8.1	Functions shall have prototype declarations and the prototype shall be visible at both the function definition and call	Required	Yes
Misra2004:8.2	Whenever an object or function is declared or defined, its type shall be explicitly stated	Required	Yes
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	Required	Yes

Misra2004:8.4	If objects or functions are declared more than once their types shall be compatible	Required	Yes
Misra2004:8.5	There shall be no definitions of objects or functions in a header file	Required	Yes
Misra2004:8.6	Functions shall be declared at file scope	Required	Yes
Misra2004:8.7	Objects shall be defined at block scope if they are only accessed from within a single function	Required	Yes
Misra2004:8.8	An external object or function shall be declared in one and only one file	Required	Yes
Misra2004:8.9	An identifier with external linkage shall have exactly one external definition	Required	Yes
Misra2004:8.10	All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required	Required	Yes
Misra2004:8.11	The static storage class specifier shall be used in definitions and declarations of objects and functions that have internal linkage	Required	Yes
Misra2004:8.12	When an array is declared with external linkage, its size shall be stated explicitly or defined implicitly by initialisation	Required	Yes
Misra2004:9.1	All automatic variables shall have been assigned a value before being used	Required	Yes
Misra2004:9.2	Braces shall be used to indicate and match the structure in the non-zero initialisation of arrays and structures	Required	Yes
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	Required	Yes
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	Required	Yes
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	Required	Yes
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	Required	Yes
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	Required	Yes
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	Required	Yes
Misra2004:10.6	A "U" suffix shall be applied to all constants of unsigned type	Required	Yes
Misra2004:11.1	Conversions shall not be performed between a pointer to a function and any type other than an integral type	Required	Yes
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	Required	Yes
Misra2004:11.3	A cast should not be performed between a pointer type and an integral type	Advisory	Yes
Misra2004:11.4	A cast should not be performed between a pointer to object type and a different pointer to object type	Advisory	Yes
Misra2004:11.5	A cast shall not be performed that removes any const or volatile qualification from the type addressed by a pointer	Required	Yes
Misra2004:12.1	Limited dependence should be placed on C's operator precedence rules in expressions	Advisory	Yes
Misra2004:12.2	The value of an expression shall be the same under any order of evaluation that the standard permits	Required	No
Misra2004:12.3	The sizeof operator shall not be used on expressions that contain side effects	Required	Yes
Misra2004:12.4	The right-hand operand of a logical && or operator shall not contain side effects	Required	No
Misra2004:12.5	The operands of a logical && or shall be primary-expressions	Required	No
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 ?:)	Advisory	Yes
Misra2004:12.7	Bitwise operators shall not be applied to operands whose underlying type is signed	Required	Yes
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	Required	Yes
Misra2004:12.9	The unary minus operator shall not be applied to an expression whose underlying type is unsigned	Required	Yes
Misra2004:12.10	The comma operator shall not be used	Required	Yes

Misra2004:12.11	Evaluation of constant unsigned integer expressions should not lead to wraparound	Advisory	Yes
Misra2004:12.12	The underlying bit representations of floating-point values shall not be used	Required	No
Misra2004:12.13	The increment (++) and decrement (--) operators should not be mixed with other operators in an expression	Advisory	Yes
Misra2004:13.1	Assignment operators shall not be used in expressions that yield a Boolean value	Required	No
Misra2004:13.2	Tests of a value against zero should be made explicit, unless the operand is effectively Boolean	Advisory	Yes
Misra2004:13.3	Floating-point expressions shall not be tested for equality or inequality	Required	Yes
Misra2004:13.4	The controlling expression of a for statement shall not contain any objects of floating type	Required	Yes
Misra2004:13.5	The three expressions of a for statement shall be concerned only with loop control	Required	Yes
Misra2004:13.6	Numeric variables being used within a for loop for iteration counting shall not be modified in the body of the loop	Required	Yes
Misra2004:13.7	Boolean operations whose results are invariant shall not be permitted	Required	Yes
Misra2004:14.1	There shall be no unreachable code	Required	Yes
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	Required	Yes
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	Required	Yes
Misra2004:14.4	The goto statement shall not be used	Required	Yes
Misra2004:14.5	The continue statement shall not be used	Required	Yes
Misra2004:14.6	For any iteration statement there shall be at most one break statement used for loop termination	Required	Yes
Misra2004:14.7	A function shall have a single point of exit at the end of the function	Required	Yes
Misra2004:14.8	The statement forming the body of a switch, while, do ... while or for statement shall be a compound statement	Required	Yes
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	Required	Yes
Misra2004:14.10	All if . else if constructs shall be terminated with an else clause	Required	Yes
Misra2004:15.0	The MISRA C switch syntax shall be used	Required	Yes
Misra2004:15.1	A switch label shall only be used when the most closely-enclosing compound statement is the body of a switch statement	Required	Yes
Misra2004:15.2	An unconditional break statement shall terminate every non-empty switch clause	Required	Yes
Misra2004:15.3	The final clause of a switch statement shall be the default clause	Required	Yes
Misra2004:15.4	A switch expression shall not represent a value that is effectively Boolean	Required	Yes
Misra2004:15.5	Every switch statement shall have at least one case clause	Required	Yes
Misra2004:16.1	Functions shall not be defined with variable numbers of arguments	Required	No
Misra2004:16.2	Functions shall not call themselves, either directly or indirectly	Required	Yes
Misra2004:16.3	Identifiers shall be given for all of the parameters in a function prototype declaration	Required	Yes
Misra2004:16.4	The identifiers used in the declaration and definition of a function shall be identical	Required	Yes
Misra2004:16.5	Functions with no parameters shall be declared and defined with the parameter list void	Required	Yes
Misra2004:16.6	The number of arguments passed to a function shall match the number of parameters	Required	Yes
Misra2004:16.7	A pointer parameter in a function prototype should be declared as pointer to const if the pointer is not used to modify the addressed object	Advisory	Yes
Misra2004:16.8	All exit paths from a function with non-void return type shall have an explicit return statement with an expression	Required	Yes
Misra2004:16.9	A function identifier shall only be used with either a preceding &, or with a parenthesised parameter list, which may be empty	Required	Yes
Misra2004:16.10	If a function returns error information, then that error information shall be tested	Required	Yes
Misra2004:17.1	Pointer arithmetic shall only be applied to pointers that address an array or array element	Required	Yes
Misra2004:17.2	Pointer subtraction shall only be applied to pointers that address elements of the same array	Required	Yes
Misra2004:17.3	>, >=, <, <= shall not be applied to pointer types except where they point to the same array	Required	Yes
Misra2004:17.4	Array indexing shall be the only allowed form of pointer arithmetic	Required	Yes

Misra2004:17.5	The declaration of objects should contain no more than 2 levels of pointer indirection	Advisory	No
Misra2004:17.6	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	Required	Yes
Misra2004:18.1	All structure or union types shall be complete at the end of a translation unit	Required	No
Misra2004:18.2	An object shall not be assigned to an overlapping object	Required	Yes
Misra2004:18.3	An area of memory shall not be reused for unrelated purposes	Required	No
Misra2004:18.4	Unions shall not be used	Required	Yes
Misra2004:19.1	#include statements in a file should only be preceded by other preprocessor directives or comments	Advisory	Yes
Misra2004:19.2	Non-standard characters should not occur in header file names in #include directives	Advisory	Yes
Misra2004:19.3	The #include directive shall be followed by either a or "filename" sequence	Required	Yes
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	Required	Yes
Misra2004:19.5	Macros shall not be #define'd or #undef'd within a block	Required	Yes
Misra2004:19.6	#undef shall not be used	Required	Yes
Misra2004:19.7	A function should be used in preference to a function-like macro	Advisory	Yes
Misra2004:19.8	A function-like macro shall not be invoked without all of its arguments	Required	No
Misra2004:19.9	Arguments to a function-like macro shall not contain tokens that look like preprocessing directives	Required	Yes
Misra2004:19.10	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 ##	Required	No
Misra2004:19.11	All macro identifiers in preprocessor directives shall be defined before use, except in #ifdef and #ifndef preprocessor directives and the defined() operator	Required	Yes
Misra2004:19.12	There shall be at most one occurrence of the # or ## preprocessor operators in a single macro definition	Required	No
Misra2004:19.13	The # and ## preprocessor operators should not be used	Advisory	Yes
Misra2004:19.14	The defined preprocessor operator shall only be used in one of the two standard forms	Required	No
Misra2004:19.15	Precautions shall be taken in order to prevent the contents of a header file being included twice	Required	No
Misra2004:19.16	Preprocessing directives shall be syntactically meaningful even when excluded by the preprocessor	Required	No
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	Required	Yes
Misra2004:20.1	Reserved identifiers, macros and functions in the standard library, shall not be defined, redefined or undefined	Required	Yes
Misra2004:20.2	The names of standard library macros, objects and functions shall not be reused	Required	Yes
Misra2004:20.3	The validity of values passed to library functions shall be checked	Required	Yes
Misra2004:20.4	Dynamic heap memory allocation shall not be used	Required	Yes
Misra2004:20.5	The error indicator errno shall not be used	Required	No
Misra2004:20.6	The macro offsetof, in library , shall not be used	Required	Yes
Misra2004:20.7	The setjmp macro and the longjmp function shall not be used	Required	Yes
Misra2004:20.8	The signal handling facilities of shall not be used	Required	Yes
Misra2004:20.9	The input/output library shall not be used in production code	Required	Yes
Misra2004:20.10	The library functions atof, atoi and atol from library shall not be used	Required	Yes
Misra2004:20.11	The library functions abort, exit, getenv and system from library shall not be used	Required	Yes
Misra2004:20.12	The time handling functions of library shall not be used	Required	Yes
Misra2004:21.1	Minimisation 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	Required	No

MISRA C:2004 BROAD MAPPING (CODESONAR V7.3)

The following table contains CodeSonar warning classes that are broadly mapped to MISRA C:2004 categories.

Rule	Rule Name	Category	Supported
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	Required	Yes
Misra2004:1.2	No reliance shall be placed on undefined or unspecified behaviour	Required	No
Misra2004:1.3	Multiple compilers and/or languages shall only be used if there is a common defined interface standard for object code to which the languages/compilers/assemblers conform	Required	No
Misra2004:1.4	The compiler/linker shall be checked to ensure that 31 character significance and case sensitivity are supported for external identifiers	Required	No
Misra2004:1.5	Floating-point implementations should comply with a defined floating-point standard	Advisory	No
Misra2004:2.1	Assembly language shall be encapsulated and isolated	Required	Yes
Misra2004:2.2	Source code shall only use /* ... */ style comments	Required	Yes
Misra2004:2.3	The character sequence /* shall not be used within a comment	Required	Yes
Misra2004:2.4	Sections of code should not be "commented out"	Advisory	Yes
Misra2004:3.1	All usage of implementation-defined behaviour shall be documented	Required	No
Misra2004:3.2	The character set and the corresponding encoding shall be documented	Required	No
Misra2004:3.3	The implementation of integer division in the chosen compiler should be determined, documented and taken into account	Advisory	No
Misra2004:3.4	All uses of the #pragma directive shall be documented and explained	Required	No
Misra2004:3.5	The implementation defined behaviour and packing of bitfields shall be documented if being relied upon	Required	No
Misra2004:3.6	All libraries used in production code shall be written to comply with the provisions of this document, and shall have been subject to appropriate validation	Required	No
Misra2004:4.1	Only those escape sequences that are defined in the ISO C standard shall be used	Required	No
Misra2004:4.2	Trigraphs shall not be used	Required	Yes
Misra2004:5.1	Identifiers (internal and external) shall not rely on the significance of more than 31 characters	Required	Yes
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	Required	Yes
Misra2004:5.3	A typedef name shall be a unique identifier	Required	Yes
Misra2004:5.4	A tag name shall be a unique identifier	Required	Yes
Misra2004:5.5	No object or function identifier with static storage duration should be reused	Advisory	Yes
Misra2004:5.6	No identifier in one name space should have the same spelling as an identifier in another name space, with the exception of structure member and union member names	Advisory	No
Misra2004:5.7	No identifier name should be reused	Advisory	Yes
Misra2004:6.1	The plain char type shall be used only for storage and use of character values	Required	Yes
Misra2004:6.2	signed and unsigned char type shall be used only for the storage and use of numeric values	Required	Yes
Misra2004:6.3	typedefs that indicate size and signedness should be used in place of the basic numerical types	Advisory	Yes
Misra2004:6.4	Bit fields shall only be defined to be of type unsigned int or signed int	Required	Yes
Misra2004:6.5	Bit fields of signed type shall be at least 2 bits long	Required	Yes
Misra2004:7.1	Octal constants (other than zero) and octal escape sequences shall not be used	Required	Yes
Misra2004:8.1	Functions shall have prototype declarations and the prototype shall be visible at both the function definition and call	Required	Yes
Misra2004:8.2	Whenever an object or function is declared or defined, its type shall be explicitly stated	Required	Yes
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	Required	Yes
Misra2004:8.4	If objects or functions are declared more than once their types shall be compatible	Required	Yes
Misra2004:8.5	There shall be no definitions of objects or functions in a header file	Required	Yes

Misra2004:8.6	Functions shall be declared at file scope	Required	Yes
Misra2004:8.7	Objects shall be defined at block scope if they are only accessed from within a single function	Required	Yes
Misra2004:8.8	An external object or function shall be declared in one and only one file	Required	Yes
Misra2004:8.9	An identifier with external linkage shall have exactly one external definition	Required	Yes
Misra2004:8.10	All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required	Required	Yes
Misra2004:8.11	The static storage class specifier shall be used in definitions and declarations of objects and functions that have internal linkage	Required	Yes
Misra2004:8.12	When an array is declared with external linkage, its size shall be stated explicitly or defined implicitly by initialisation	Required	Yes
Misra2004:9.1	All automatic variables shall have been assigned a value before being used	Required	Yes
Misra2004:9.2	Braces shall be used to indicate and match the structure in the non-zero initialisation of arrays and structures	Required	Yes
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	Required	Yes
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	Required	Yes
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	Required	Yes
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	Required	Yes
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	Required	Yes
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	Required	Yes
Misra2004:10.6	A "U" suffix shall be applied to all constants of unsigned type	Required	Yes
Misra2004:11.1	Conversions shall not be performed between a pointer to a function and any type other than an integral type	Required	Yes
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	Required	Yes
Misra2004:11.3	A cast should not be performed between a pointer type and an integral type	Advisory	Yes
Misra2004:11.4	A cast should not be performed between a pointer to object type and a different pointer to object type	Advisory	Yes
Misra2004:11.5	A cast shall not be performed that removes any const or volatile qualification from the type addressed by a pointer	Required	Yes
Misra2004:12.1	Limited dependence should be placed on C's operator precedence rules in expressions	Advisory	Yes
Misra2004:12.2	The value of an expression shall be the same under any order of evaluation that the standard permits	Required	No
Misra2004:12.3	The sizeof operator shall not be used on expressions that contain side effects	Required	Yes
Misra2004:12.4	The right-hand operand of a logical && or operator shall not contain side effects	Required	No
Misra2004:12.5	The operands of a logical && or shall be primary-expressions	Required	No
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 ?:)	Advisory	Yes
Misra2004:12.7	Bitwise operators shall not be applied to operands whose underlying type is signed	Required	Yes
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	Required	Yes
Misra2004:12.9	The unary minus operator shall not be applied to an expression whose underlying type is unsigned	Required	Yes
Misra2004:12.10	The comma operator shall not be used	Required	Yes
Misra2004:12.11	Evaluation of constant unsigned integer expressions should not lead to wraparound	Advisory	Yes
Misra2004:12.12	The underlying bit representations of floating-point values shall not be used	Required	No

Misra2004:12.13	The increment (++) and decrement (--) operators should not be mixed with other operators in an expression	Advisory	Yes
Misra2004:13.1	Assignment operators shall not be used in expressions that yield a Boolean value	Required	No
Misra2004:13.2	Tests of a value against zero should be made explicit, unless the operand is effectively Boolean	Advisory	Yes
Misra2004:13.3	Floating-point expressions shall not be tested for equality or inequality	Required	Yes
Misra2004:13.4	The controlling expression of a for statement shall not contain any objects of floating type	Required	Yes
Misra2004:13.5	The three expressions of a for statement shall be concerned only with loop control	Required	Yes
Misra2004:13.6	Numeric variables being used within a for loop for iteration counting shall not be modified in the body of the loop	Required	Yes
Misra2004:13.7	Boolean operations whose results are invariant shall not be permitted	Required	Yes
Misra2004:14.1	There shall be no unreachable code	Required	Yes
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	Required	Yes
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	Required	Yes
Misra2004:14.4	The goto statement shall not be used	Required	Yes
Misra2004:14.5	The continue statement shall not be used	Required	Yes
Misra2004:14.6	For any iteration statement there shall be at most one break statement used for loop termination	Required	Yes
Misra2004:14.7	A function shall have a single point of exit at the end of the function	Required	Yes
Misra2004:14.8	The statement forming the body of a switch, while, do ... while or for statement shall be a compound statement	Required	Yes
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	Required	Yes
Misra2004:14.10	All if . else if constructs shall be terminated with an else clause	Required	Yes
Misra2004:15.0	The MISRA C switch syntax shall be used	Required	Yes
Misra2004:15.1	A switch label shall only be used when the most closely-enclosing compound statement is the body of a switch statement	Required	Yes
Misra2004:15.2	An unconditional break statement shall terminate every non-empty switch clause	Required	Yes
Misra2004:15.3	The final clause of a switch statement shall be the default clause	Required	Yes
Misra2004:15.4	A switch expression shall not represent a value that is effectively Boolean	Required	Yes
Misra2004:15.5	Every switch statement shall have at least one case clause	Required	Yes
Misra2004:16.1	Functions shall not be defined with variable numbers of arguments	Required	No
Misra2004:16.2	Functions shall not call themselves, either directly or indirectly	Required	Yes
Misra2004:16.3	Identifiers shall be given for all of the parameters in a function prototype declaration	Required	Yes
Misra2004:16.4	The identifiers used in the declaration and definition of a function shall be identical	Required	Yes
Misra2004:16.5	Functions with no parameters shall be declared and defined with the parameter list void	Required	Yes
Misra2004:16.6	The number of arguments passed to a function shall match the number of parameters	Required	Yes
Misra2004:16.7	A pointer parameter in a function prototype should be declared as pointer to const if the pointer is not used to modify the addressed object	Advisory	Yes
Misra2004:16.8	All exit paths from a function with non-void return type shall have an explicit return statement with an expression	Required	Yes
Misra2004:16.9	A function identifier shall only be used with either a preceding &, or with a parenthesised parameter list, which may be empty	Required	Yes
Misra2004:16.10	If a function returns error information, then that error information shall be tested	Required	Yes
Misra2004:17.1	Pointer arithmetic shall only be applied to pointers that address an array or array element	Required	Yes
Misra2004:17.2	Pointer subtraction shall only be applied to pointers that address elements of the same array	Required	Yes
Misra2004:17.3	>, >=, <, <= shall not be applied to pointer types except where they point to the same array	Required	Yes
Misra2004:17.4	Array indexing shall be the only allowed form of pointer arithmetic	Required	Yes
Misra2004:17.5	The declaration of objects should contain no more than 2 levels of pointer indirection	Advisory	No

Misra2004:17.6	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	Required	Yes
Misra2004:18.1	All structure or union types shall be complete at the end of a translation unit	Required	No
Misra2004:18.2	An object shall not be assigned to an overlapping object	Required	Yes
Misra2004:18.3	An area of memory shall not be reused for unrelated purposes	Required	No
Misra2004:18.4	Unions shall not be used	Required	Yes
Misra2004:19.1	#include statements in a file should only be preceded by other preprocessor directives or comments	Advisory	Yes
Misra2004:19.2	Non-standard characters should not occur in header file names in #include directives	Advisory	Yes
Misra2004:19.3	The #include directive shall be followed by either a or "filename" sequence	Required	Yes
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	Required	Yes
Misra2004:19.5	Macros shall not be #define'd or #undef'd within a block	Required	Yes
Misra2004:19.6	#undef shall not be used	Required	Yes
Misra2004:19.7	A function should be used in preference to a function-like macro	Advisory	Yes
Misra2004:19.8	A function-like macro shall not be invoked without all of its arguments	Required	No
Misra2004:19.9	Arguments to a function-like macro shall not contain tokens that look like preprocessing directives	Required	Yes
Misra2004:19.10	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 ##	Required	No
Misra2004:19.11	All macro identifiers in preprocessor directives shall be defined before use, except in #ifdef and #ifndef preprocessor directives and the defined() operator	Required	Yes
Misra2004:19.12	There shall be at most one occurrence of the # or ## preprocessor operators in a single macro definition	Required	No
Misra2004:19.13	The # and ## preprocessor operators should not be used	Advisory	Yes
Misra2004:19.14	The defined preprocessor operator shall only be used in one of the two standard forms	Required	No
Misra2004:19.15	Precautions shall be taken in order to prevent the contents of a header file being included twice	Required	No
Misra2004:19.16	Preprocessing directives shall be syntactically meaningful even when excluded by the preprocessor	Required	No
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	Required	Yes
Misra2004:20.1	Reserved identifiers, macros and functions in the standard library, shall not be defined, redefined or undefined	Required	Yes
Misra2004:20.2	The names of standard library macros, objects and functions shall not be reused	Required	Yes
Misra2004:20.3	The validity of values passed to library functions shall be checked	Required	Yes
Misra2004:20.4	Dynamic heap memory allocation shall not be used	Required	Yes
Misra2004:20.5	The error indicator errno shall not be used	Required	No
Misra2004:20.6	The macro offsetof, in library , shall not be used	Required	Yes
Misra2004:20.7	The setjmp macro and the longjmp function shall not be used	Required	Yes
Misra2004:20.8	The signal handling facilities of shall not be used	Required	Yes
Misra2004:20.9	The input/output library shall not be used in production code	Required	Yes
Misra2004:20.10	The library functions atof, atoi and atol from library shall not be used	Required	Yes
Misra2004:20.11	The library functions abort, exit, getenv and system from library shall not be used	Required	Yes
Misra2004:20.12	The time handling functions of library shall not be used	Required	Yes
Misra2004:21.1	Minimisation 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	Required	No

GammaTech is a leading global provider of application testing (AST) solutions used by the world's most security conscious organizations to detect, measure, analyze and resolve vulnerabilities for software they develop or use. The company is also a trusted cybersecurity and artificial intelligence research partner for the nation's civil, defense, and intelligence agencies.

CodeSonar and CodeSentry are registered trademarks of GammaTech, Inc.
© GammaTech, Inc. All rights reserved.

