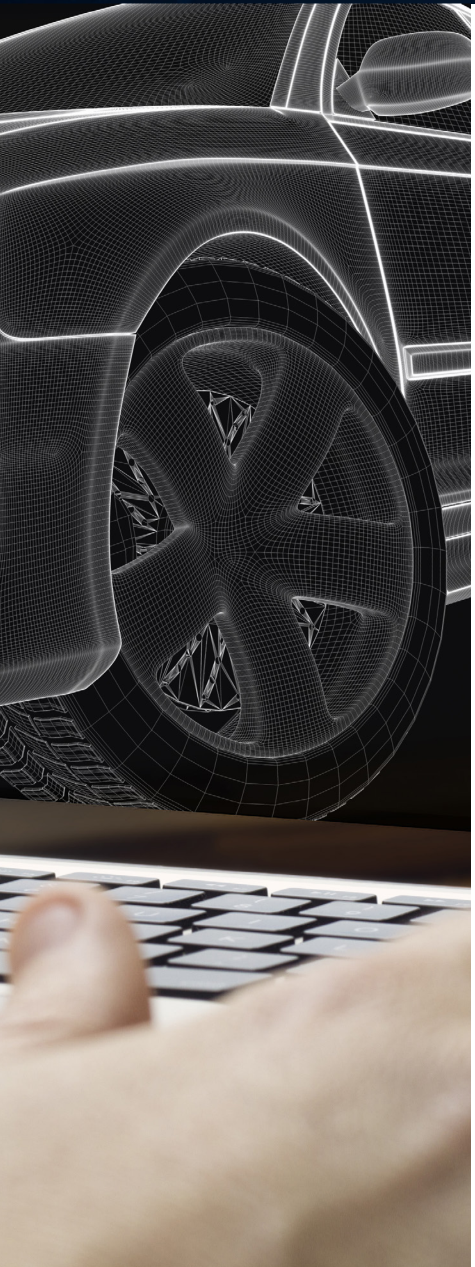
Achieving MISRA C:2012 Compliance with Parasoft C/C++test

EXECUTIVE SUMMARY

Software coding standards for automotive applications, such as MISRA, have been around for years, but with the emergence of complex technologies, such as autonomous driving and sophisticated connectivity, the need to automate the implementation of rigorous coding standards has never been greater. These innovations represent not only the next evolution of a rapidly shifting and highly-competitive market, they also present a much larger surface area for defects that impact the safety, security, and reliability of the software.

Further complicating the issue is the industry's highly distributed, multi-tiered production model. The automotive software supply chain involves many vendors and suppliers simultaneously contributing to the software that goes into the final product. Implementing quality control mechanisms in such a complex system is difficult, but failing to do so introduces additional risk into the process. It is imperative that businesses take action now to implement defect prevention strategies that reduce the risk associated with software development.



MISRA C:2012

The first step in developing safe and secure code is to establish a standard for constructing defect-free code. The MISRA standard is widely used in safety-critical industries, such as automotive, medical, military, and aerospace, and provides a set of best practices for writing C code, facilitating the authorship of safe, secure, and portable code. MISRA supports the C90 and C99 language specifications. The current version of MISRA, MISRA C:2012, has evolved over several years and includes 143 rules and 16 directives for a total of 159 guidelines. Amendment 1 to MISRA C:2012, published in 2016, expanded the standard by 14 rules.

OVERVIEW OF PARASOFT SUPPORT FOR MISRA C:2012

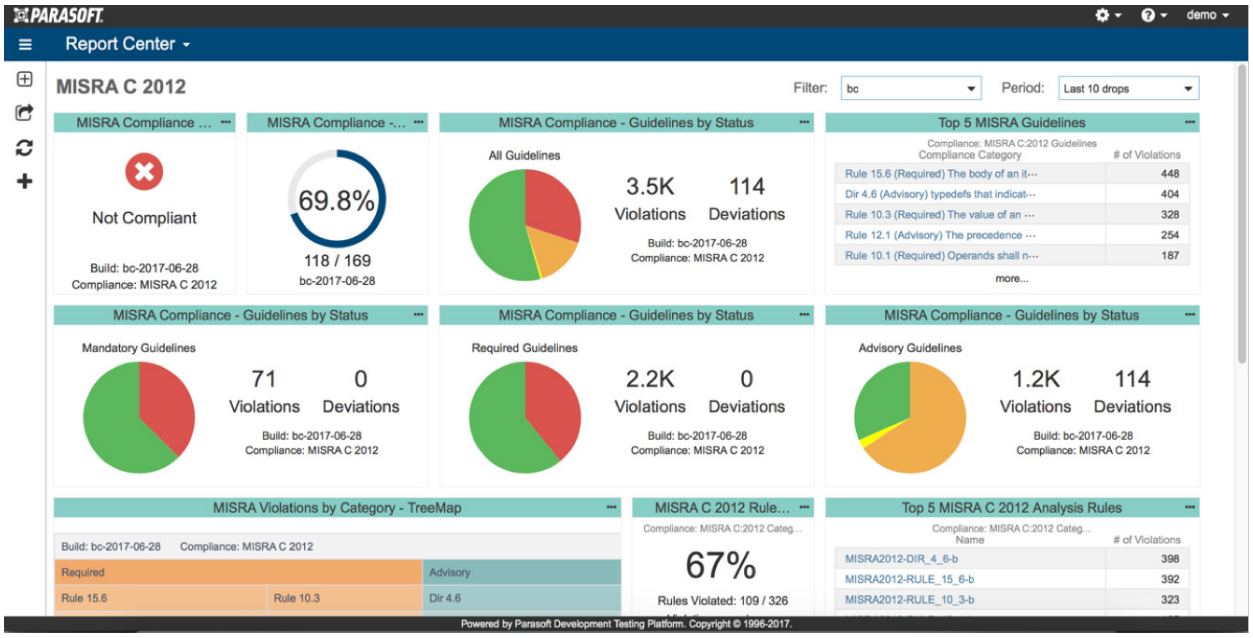
Parasoft C/C++test and Parasoft DTP provides a comprehensive solution for applying MISRA C:2012, including Amendment 1, to help organizations overcome the challenges associated with ensuring automotive software quality. Parasoft C/C++test is a TUV-certified, scalable solution that automates the application and monitoring of coding standards, such as MISRA. It also provides a unit testing platform that integrates test creation, execution, and coverage reporting. Test and analysis data from C/C++test can be sent to DTP, which aggregates, correlates, and applies additional analytics to centralize reporting for each step along the complex software supply chain.

Parasoft's MISRA Compliance Pack provides a set of reporting and configuration artifacts for DTP that automate the compliance documentation required by MISRA—greatly reducing the time and effort associated with demonstrating compliance and traceability. See examples from the MISRA Compliance Pack on the following page.

PARASOFT®
Automated Software Testing

COMPLIANCE PACK: THE MISRA COMPLIANCE DASHBOARD

An important metric of a project is its current state of compliance, including the various finer points in measuring compliance. A comprehensive MISRA compliance dashboard provides an on-the-spot evaluation of the project. This high-level view provides managers with an easily-accessible understanding of compliance at a glance, and gives developers a starting point for making progress towards achieving compliance.



Top 5 MISRA Guidelines

Compliance: MISRA C:2012 Guidelines

Compliance Category	# of Violations
Rule 15.6 (Required) The body of an it···	448
Dir 4.6 (Advisory) typedefs that indicat···	404
Rule 10.3 (Required) The value of an ···	328
Rule 12.1 (Advisory) The precedence ···	254
Rule 10.1 (Required) Operands shall n···	187

more...

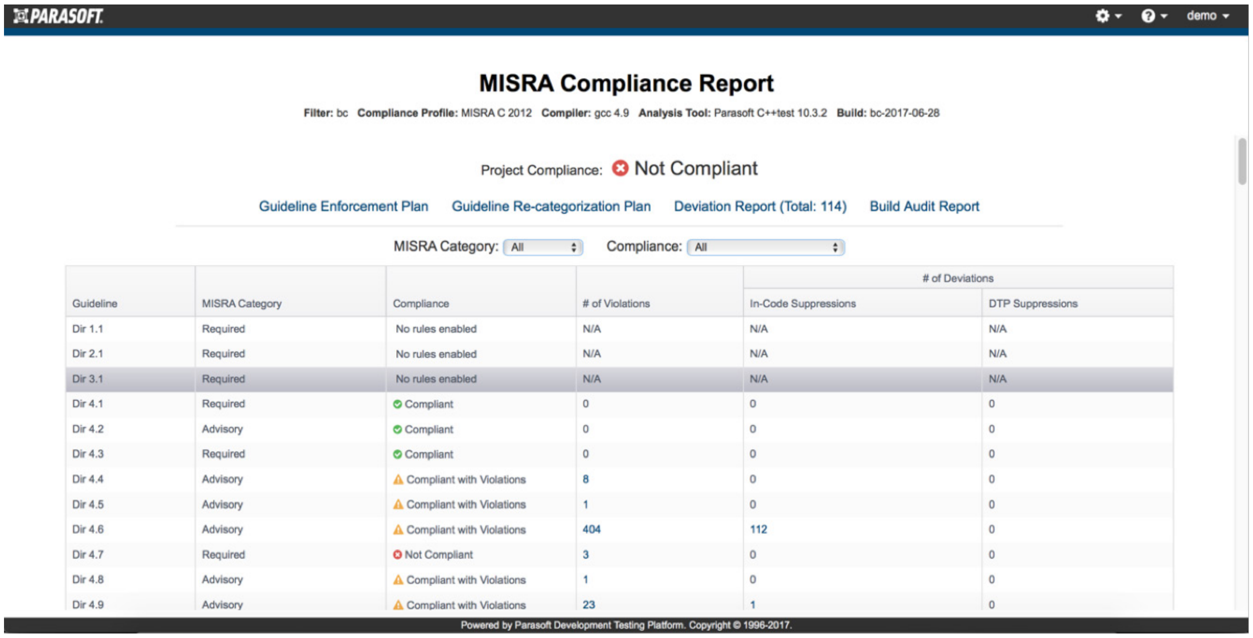
Top 5 MISRA C 2012 Analysis Rules

Compliance: MISRA C:2012 Categ...

Name	# of Violations
MISRA2012-DIR_4_6-b	398
MISRA2012-RULE_15_6-b	392
MISRA2012-RULE_10_3-b	323

COMPLIANCE PACK: THE GUIDELINES COMPLIANCE SUMMARY

The Guidelines Compliance Summary is the primary record of overall project compliance. This report documents the state of compliance for each guideline, as well as any associated deviations or re-categorizations.


PARASOFT
demo
Guideline Enforcement Plan
Guideline Re-categorization Plan
Deviation Report (Total: 114)
Build Audit Report
Powered by Parasoft Development Testing Platform. Copyright © 1996-2017.

MISRA Compliance Report

Filter: bc **Compliance Profile:** MISRA C 2012 **Compiler:** gcc 4.9 **Analysis Tool:** Parasoft C++test 10.3.2 **Build:** bc-2017-06-28

Project Compliance: Not Compliant

MISRA Category: All Compliance: All

Guideline	MISRA Category	Compliance	# of Violations	# of Deviations	
				In-Code Suppressions	DTP Suppressions
Dir 1.1	Required	No rules enabled	N/A	N/A	N/A
Dir 2.1	Required	No rules enabled	N/A	N/A	N/A
Dir 3.1	Required	No rules enabled	N/A	N/A	N/A
Dir 4.1	Required	✓ Compliant	0	0	0
Dir 4.2	Advisory	✓ Compliant	0	0	0
Dir 4.3	Required	✓ Compliant	0	0	0
Dir 4.4	Advisory	⚠ Compliant with Violations	8	0	0
Dir 4.5	Advisory	⚠ Compliant with Violations	1	0	0
Dir 4.6	Advisory	⚠ Compliant with Violations	404	112	0
Dir 4.7	Required	✖ Not Compliant	3	0	0
Dir 4.8	Advisory	⚠ Compliant with Violations	1	0	0
Dir 4.9	Advisory	⚠ Compliant with Violations	23	1	0

Detailed rule mappings are specified on the following pages.

MISRA C:2012 Summary

	Decidable	Undecidable
	Supported/Total (Coverage)	**Supported/Total (Coverage)**
All	116/116 (100%)	43/43 (100%)
Mandatory	5/5 (100%)	11/11 (100%)
Required	84 /84 (100%)	23/26 (88.5%)
Advisory	23/23 (100%)	11/11 (100%)

MISRA C:2012 Amendment 1 Summary

	Decidable	Undecidable
	Supported/Total (Coverage)	**Supported/Total (Coverage)**
All	4/4 (100%)	10/10 (100%)
Mandatory	1/1 (100%)	5/5 (100%)
Required	3 /3 (100%)	5/5 (100%)
Advisory	n/a	n/a

Parasoft MISRA C:2012 Rule Mapping

The following table provides a line-by-line correlation of Parasoft support for MISRA.

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Dir-1.1: Any implementation-defined behavior on which the output of the program depends shall be documented and understood	Required	Undecidable	Cannot be statically verified
Dir-2.1: All source files shall compile without any compilation errors	Required	Undecidable	Non-compliance code will generate a parse error
Dir-3.1: All code shall be traceable to documented requirements	Required	Undecidable	Parasoft DTP provides traceability between tests and requirements and tests and code

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Dir-4.1: Run-time failures shall be minimized	Required	Undecidable	**MISRA2012-DIR-4_1_a*:** Avoid accessing arrays out of bounds **MISRA2012-DIR-4_1_b*:** Avoid null pointer dereferencing **MISRA2012-DIR-4_1_c*:** Avoid division by zero **MISRA2012-DIR-4_1_d*:** Avoid buffer overflow due to defining incorrect format limits **MISRA2012-DIR-4_1_e*:** Avoid overflow due to reading a not zero terminated string **MISRA2012-DIR-4_1_f*:** Do not check for null after dereferencing **MISRA2012-DIR-4_1_g*:** Avoid overflow when reading from a buffer **MISRA2012-DIR-4_1_h*:** Avoid overflow when writing to a buffer **MISRA2012-DIR-4_1_i:** Pointer arithmetic shall only be applied to pointers that address an array or array element **MISRA2012-DIR-4_1_j:** >, >=, <, <= shall not be applied to objects of pointer type, except where they point to the same array
Dir-4.2: All usage of assembly language should be documented	Advisory	Undecidable	**MISRA2012-DIR-4_2:** All usage of assembly language should be documented
Dir-4.3: Assembly language shall be encapsulated and isolated	Required	Undecidable	**MISRA2012-DIR-4_3:** Assembly language shall be encapsulated and isolated
Dir-4.4: Sections of code should not be commented out	Advisory	Undecidable	**MISRA2012-DIR-4_4:** Sections of code should not be “commented out”
Dir-4.5: Identifiers in the same namespace with overlapping visibility should be typographically unambiguous	Advisory	Undecidable	**MISRA2012-DIR-4_5:** Identifiers in the same name space with overlapping visibility should be typographically unambiguous
Dir-4.6: Typedefs that indicate size and signedness should be used in place of the basic numerical types	Advisory	Undecidable	**MISRA2012-DIR-4_6_a:** typedefs to basic types should contain some digits in their name **MISRA2012-DIR-4_6_b:** typedefs should be used in place of the basic types **MISRA2012-DIR-4_6_c:** Use typedefs from stdint.h instead of declaring your own in C99 code

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Dir-4.7: If a function returns error information, then that error information shall be tested	Required	Undecidable	**MISRA2012-DIR-4_7_a*:** Consistently check the returned value of non-void functions **MISRA2012-DIR-4_7_b*:** Always check the returned value of non-void function
Dir-4.8: If a pointer to a structure or union is never dereferenced within a translation unit, then the implementation of the object should be hidden	Advisory	Undecidable	**MISRA2012-DIR-4_8:** If a pointer to a structure or union is never dereferenced within a translation unit, then the implementation of the object should be hidden
Dir-4.9: A function should be used in preference to a function-like macro where they are interchangeable	Advisory	Undecidable	**MISRA2012-DIR-4_9:** A function should be used in preference to a function-like macro where they are interchangeable
Dir-4.10: Precautions shall be taken in order to prevent the contents of a header file being included more than once	Required	Undecidable	**MISRA2012-DIR-4_10:** Precautions shall be taken in order to prevent the contents of a header file being included more than once
Dir-4.11: The validity of values passed to library functions shall be checked	Required	Undecidable	**MISRA2012-DIR-4_11*:** Validate values passed to library functions
Dir-4.12: Dynamic memory allocation shall not be used	Required	Undecidable	**MISRA2012-DIR-4_12:** Dynamic memory allocation shall not be used
Dir-4.13: Functions which are designed to provide operations on a resource should be called in an appropriate sequence	Advisory	Undecidable	**MISRA2012-DIR-4_13_a*:** All resources obtained dynamically by means of Standard Library functions shall be explicitly released **MISRA2012-DIR-4_13_b*:** Do not use resources that have been freed **MISRA2012-DIR-4_13_c*:** Do not free resources using invalid pointers **MISRA2012-DIR-4_13_d*:** Do not abandon unreleased locks **MISRA2012-DIR-4_13_e*:** Avoid double locking

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-1.1: The program shall contain no violations of the standard C syntax and constraints, and shall not exceed the implementation's translation limits	Required	Decidable	**MISRA2012-RULE-1_1_a_c90:** A program should not exceed the translation limits imposed by The Standard (c90) **MISRA2012-RULE-1_1_a_c99:** A program should not exceed the translation limits imposed by The Standard (c99) **MISRA2012-RULE-1_1_b_c90:** A program should not exceed the translation limits imposed by The Standard (c90) **MISRA2012-RULE-1_1_b_c99:** A program should not exceed the translation limits imposed by The Standard (c99)
Rule-1.2: Language extensions should not be used	Advisory	Undecidable	Cannot be statically verified
Rule-1.3: There shall be no occurrence of undefined or critical unspecified behavior	Required	Undecidable	**MISRA2012-RULE-1_3_a***: Avoid division by zero **MISRA2012-RULE-1_3_b***: Avoid use before initialization **MISRA2012-RULE-1_3_c***: Do not use resources that have been freed **MISRA2012-RULE-1_3_d***: Avoid overflow when reading from a buffer **MISRA2012-RULE-1_3_e***: Avoid overflow when writing to a buffer **MISRA2012-RULE-1_3_f:** The value of an expression shall be the same under any order of evaluation that the standard permits **MISRA2012-RULE-1_3_g:** Don't write code that depends on the order of evaluation of function arguments **MISRA2012-RULE-1_3_h:** Don't write code that depends on the order of evaluation of function designator and function arguments **MISRA2012-RULE-1_3_i:** Don't write code that depends on the order of evaluation of expression that involves a function call **MISRA2012-RULE-1_3_j:** Between sequence points an object shall have its stored value modified at most once by the evaluation of an expression **MISRA2012-RULE-1_3_k:** Do not use more than one volatile in one expression **MISRA2012-RULE-1_3_l:** Don't write code that depends on the order of evaluation of function calls **MISRA2012-RULE-1_3_m:** A function shall not return a pointer or reference to a non-static local object **MISRA2012-RULE-1_3_n:** The address of an object with automatic storage shall not be assigned to an object which persists after the object has ceased to exist **MISRA2012-RULE-1_3_o:** The left-hand operand of a right-shift operator shall not have a negative value

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-2.1: A project shall not contain unreachable code	Required	Undecidable	**MISRA2012-RULE-2_1_a:** There shall be no unreachable code in 'else' block **MISRA2012-RULE-2_1_b:** There shall be no unreachable code after 'return', 'break', 'continue', and 'goto' statements **MISRA2012-RULE-2_1_c:** There shall be no unreachable code in 'if', 'else', 'while', 'for' block **MISRA2012-RULE-2_1_d:** There shall be no unreachable code in 'switch' statement **MISRA2012-RULE-2_1_e:** There shall be no unreachable code in 'for' loop **MISRA2012-RULE-2_1_f:** There shall be no unreachable code after 'if' or 'switch' statement **MISRA2012-RULE-2_1_g:** There shall be no unreachable code after 'if' or 'switch' statement inside 'while'/'for'/'do...while' loop
Rule-2.2: There shall be no dead code	Required	Undecidable	**MISRA2012-RULE-2_2_a:** All non-null statements shall either have at least one side-effect however executed or cause control flow to change
Rule-2.3: A project should not contain unused type declarations	Advisory	Decidable	**MISRA2012-RULE-2_3_a:** A function should not contain unused type declarations **MISRA2012-RULE-2_3_b:** A source file should not contain unused type declarations
Rule-2.4: A project should not contain unused tag declarations	Advisory	Decidable	**MISRA2012-RULE-2_4_a:** A function should not contain unused local tag declarations **MISRA2012-RULE-2_4_b:** A source file should not contain unused tag declarations
Rule-2.5: A project should not contain unused macro declarations	Advisory	Decidable	**MISRA2012-RULE-2_5:** A source file should not contain unused macro declarations
Rule-2.6: A function should not contain unused label declarations	Advisory	Decidable	**MISRA2012-RULE-2_6:** A function should not contain unused label declarations
Rule-2.7: There should be no un-used parameters in functions	Advisory	Decidable	**MISRA2012-RULE-2_7:** There should be no unused parameters in functions
Rule-3.1: The character sequences /* and // shall not be used within a comment	Required	Decidable	**MISRA2012-RULE-3_1_a:** The character sequence /* shall not be used within a C-style comment **MISRA2012-RULE-3_1_b:** The character sequence // shall not be used within a C-style comment **MISRA2012-RULE-3_1_c:** The character sequence /* shall not be used within a C++-style comment

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-3.2: Line-splicing shall not be used in // comments	Required	Decidable	**MISRA2012-RULE-3_2:** Line-splicing shall not be used in // comments
Rule-4.1: Octal and hexadecimal escape sequences shall be terminated	Required	Decidable	**MISRA2012-RULE-4_1:** Octal and hexadecimal escape sequences shall be terminated
Rule-4.2: Trigraphs should not be used	Advisory	Decidable	**MISRA2012-RULE-4_2:** Trigraphs should not be used
Rule-5.1: External identifiers shall be distinct	Required	Decidable	**MISRA2012-RULE-5_1:** External identifiers shall be distinct
Rule-5.2: Identifiers declared in the same scope and name space shall be distinct	Required	Decidable	**MISRA2012-RULE-5_2_a_c90:** Identifiers declared in the file scope and in the same name space shall be distinct (c90) **MISRA2012-RULE-5_2_a_c99:** Identifiers declared in the file scope and in the same name space shall be distinct (c99) **MISRA2012-RULE-5_2_b_c90:** Identifiers declared in the same block scope and name space shall be distinct (c90) **MISRA2012-RULE-5_2_b_c99:** Identifiers declared in the same block scope and name space shall be distinct (c99)
Rule-5.3: An identifier declared in an inner scope shall not hide an identifier declared in an outer scope	Required	Decidable	**MISRA2012-RULE-5_3_a:** An identifier declared in an inner scope shall not hide an identifier declared in an outer scope **MISRA2012-RULE-5_3_b:** An identifier declared in an inner scope shall not hide an identifier declared in an outer scope
Rule-5.4: Macro identifiers shall be distinct	Required	Decidable	**MISRA2012-RULE-5_4_a_c90:** The name of a macro should be distinct from the names of its parameters (c90) **MISRA2012-RULE-5_4_a_c99:** The name of a macro should be distinct from the names of its parameters (c99) **MISRA2012-RULE-5_4_b_c90:** The name of a macro should be distinct from the names of other macros that are currently defined (c90) **MISRA2012-RULE-5_4_b_c99:** The name of a macro should be distinct from the names of other macros that are currently defined (c99)
Rule-5.5: Identifiers shall be distinct from macro names	Required	Decidable	**MISRA2012-RULE-5_5_c90:** Identifiers shall be distinct from macro names (c90) **MISRA2012-RULE-5_5_c99:** Identifiers shall be distinct from macro names (c99)

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-5.6: A typedef name shall be a unique identifier	Required	Decidable	**MISRA2012-RULE-5_6_a:** A typedef name shall be a unique identifier **MISRA2012-RULE-5_6_b:** A typedef name shall be a unique identifier
Rule-5.7: A tag name shall be a unique identifier	Required	Decidable	**MISRA2012-RULE-5_7_a:** A tag name shall not be reused for other purpose within the program **MISRA2012-RULE-5_7_b:** A tag name shall not be reused to define a different tag
Rule-5.8: Identifiers that define ob- jects or functions with external linkage shall be unique	Required	Decidable	**MISRA2012-RULE-5_8:** Identifiers that define objects or functions with external linkage shall be unique
Rule-5.9: Identifiers that define objects or functions with internal linkage should be unique	Advisory	Decidable	**MISRA2012-RULE-5_9_a:** Identifiers that define objects or functions with internal linkage should be unique **MISRA2012-RULE-5_9_b:** Identifiers that define objects or functions with internal linkage should be unique
Rule-6.1: Bit-fields shall only be declared with an appropriate type	Required	Decidable	**MISRA2012-RULE-6_1:** Bit-fields shall only be declared with an appropriate type
Rule-6.2: Single-bit named bit fields shall not be of a signed type	Required	Decidable	**MISRA2012-RULE-6_2:** Single-bit named bit fields shall not be of a signed type
Rule-7.1: Octal constants shall not be used	Required	Decidable	**MISRA2012-RULE-7_1:** Octal constants shall not be used
Rule-7.2: A “u” or “U” suffix shall be applied to all integer constants that are represented in an unsigned type	Required	Decidable	**MISRA2012-RULE-7_2:** A ‘u’ or ‘U’ suffix shall be applied to all integer constants that are represented in an unsigned type
Rule-7.3: The lowercase character “l” shall not be used in a literal suffix	Required	Decidable	**MISRA2012-RULE-7_3:** The lowercase character ‘l’ shall not be used in a literal suffix
Rule-7.4: A string literal shall not be assigned to an object unless the object’s type is “pointer to const-qualified char”	Required	Decidable	**MISRA2012-RULE-7_4:** A string literal shall not be assigned to an object unless the object’s type is pointer to const-qualified char

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-8.1: Types shall be explicitly specified	Required	Decidable	**MISRA2012-RULE-8_1_a:** Whenever a function is declared or defined, its type shall be explicitly stated **MISRA2012-RULE-8_1_b:** Whenever an object is declared or defined, its type shall be explicitly stated
Rule-8.2: Function types shall be in prototype form with named parameters	Required	Decidable	**MISRA2012-RULE-8_2_a:** Identifiers shall be given for all of the parameters in a function prototype declaration **MISRA2012-RULE-8_2_b:** Function types shall have named parameters **MISRA2012-RULE-8_2_c:** Function types shall be in prototype form
Rule-8.3: All declarations of an object or function shall use the same names and type qualifiers	Required	Decidable	**MISRA2012-RULE-8_3_a:** If objects or functions are declared more than once their types shall be compatible **MISRA2012-RULE-8_3_b:** The identifiers used in the declaration and definition of a function shall be identical
Rule-8.4: A compatible declaration shall be visible when an object or function with external linkage is defined	Required	Decidable	**MISRA2012-RULE-8_4_a:** A compatible declaration shall be visible when an object or function with external linkage is defined **MISRA2012-RULE-8_4_b:** A compatible declaration shall be visible when an object or function with external linkage is defined
Rule-8.5: An external object or function shall be declared once in one and only one file	Required	Decidable	**MISRA2012-RULE-8_5:** An external object or function shall not have more than one non-defining declaration in translation unit
Rule-8.6: An identifier with external linkage shall have exactly one external definition	Required	Decidable	**MISRA2012-RULE-8_6:** An identifier with external linkage shall have exactly one external definition
Rule-8.7: Functions and objects should not be defined with external linkage if they are referenced in only one translation unit	Advisory	Decidable	**MISRA2012-RULE-8_7:** Functions and objects should not be defined with external linkage if they are referenced in only one translation unit
Rule-8.8: The static storage class specifier shall be used in all declarations of objects and functions that have internal linkage	Required	Decidable	**MISRA2012-RULE-8_8:** The static storage class specifier shall be used in all declarations of objects and functions that have internal linkage
Rule-8.9: An object should be defined at block scope if its identifier only appears in a single function	Advisory	Decidable	**MISRA2012-RULE-8_9:** An object should be defined at block scope if its identifier only appears in a single function

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-8.10: An inline function shall be declared with the static storage class	Required	Decidable	**MISRA2012-RULE-8_10:** An inline function shall be declared with the static storage class
Rule-8.11: When an array with external linkage is declared, its size should be explicitly specified	Advisory	Decidable	**MISRA2012-RULE-8_11:** When an array with external linkage is declared, its size should be explicitly specified
Rule-8.12: Within an enumerator list, the value of an implicitly-specified enumeration constant shall be unique	Required	Decidable	**MISRA2012-RULE-8_12:** Within an enumerator list, the value of an implicitly-specified enumeration constant shall be unique
Rule-8.13: A pointer should point to a const qualified type whenever possible	Advisory	Undecidable	**MISRA2012-RULE-8_13_a:** 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 **MISRA2012-RULE-8_13_b:** Declare a type of parameter as typedef to pointer to const if the pointer is not used to modify the addressed object
Rule-8.14: The restrict type qualifier shall not be used	Required	Decidable	**MISRA2012-RULE-8_14:** The restrict type qualifier shall not be used
Rule-9.1: The value of an object with automatic storage duration shall not be read before it has been set	Mandatory	Undecidable	**MISRA2012-RULE-9_1*:** Avoid use before initialization
Rule-9.2: The initializer for an aggregate or union shall be enclosed in braces	Required	Decidable	**MISRA2012-RULE-9_2:** The initializer for an aggregate or union shall be enclosed in braces
Rule-9.3: Arrays shall not be partially initialized	Required	Decidable	**MISRA2012-RULE-9_3:** Arrays shall not be partially initialized
Rule-9.4: An element of an object shall not be initialized more than once	Required	Decidable	**MISRA2012-RULE-9_4:** An element of an object shall not be initialized more than once
Rule-9.5: Where designated initializers are used to initialize an array object the size of the array shall be specified explicitly	Required	Decidable	**MISRA2012-RULE-9_5:** Where designated initializers are used to initialize an array object the size of the array shall be specified explicitly

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-10.1: Operands shall not be of an inappropriate essential type	Required	Decidable	**MISRA2012-RULE-10_1_a:** An expression of essentially Boolean type should always be used where an operand is interpreted as a Boolean value **MISRA2012-RULE-10_1_b:** An operand of essentially Boolean type should not be used where an operand is interpreted as a numeric value **MISRA2012-RULE-10_1_c:** An operand of essentially character type should not be used where an operand is interpreted as a numeric value **MISRA2012-RULE-10_1_d:** An operand of essentially enum type should not be used in an arithmetic operation **MISRA2012-RULE-10_1_e:** Shift and bitwise operations should not be performed on operands of essentially signed or enum type **MISRA2012-RULE-10_1_f:** An operand of essentially signed or enum type should not be used as right hand side operand to the bitwise shifting operator **MISRA2012-RULE-10_1_g:** An operand of essentially unsigned type should not be used as the operand to the unary minus operator
Rule-10.2: Expressions of essentially character type shall not be used inappropriately in addition and subtraction operations	Required	Decidable	**MISRA2012-RULE-10_2:** Expressions of essentially character type shall not be used inappropriately in addition and subtraction operations
Rule-10.3: The value of an expression shall not be assigned to an object with a narrower essential type or of a different essential type category	Required	Decidable	**MISRA2012-RULE-10_3_a:** The value of an expression shall not be assigned to an object with a narrower essential type **MISRA2012-RULE-10_3_b:** The value of an expression shall not be assigned to an object of a different essential type category
Rule-10.4: Both operands of an operator in which the usual arithmetic conversions are performed shall have the same essential type category	Required	Decidable	**MISRA2012-RULE-10_4_a:** Both operands of an operator in which the usual arithmetic conversions are performed shall have the same essential type category **MISRA2012-RULE-10_4_b:** The second and third operands of the ternary operator shall have the same essential type category
Rule-10.5: The value of an expression should not be cast to an inappropriate essential type	Advisory	Decidable	**MISRA2012-RULE-10_5_a:** The cast operation to essentially enumeration type is not allowed **MISRA2012-RULE-10_5_b:** Do not cast from or to essentially Boolean type **MISRA2012-RULE-10_5_c:** Do not use casts between essentially character types and essentially floating types

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-10.6: The value of a composite expression shall not be assigned to an object with wider essential type	Required	Decidable	**MISRA2012-RULE-10_6:** The value of a composite expression shall not be assigned to an object with wider essential type
Rule-10.7: If a composite expression is used as one operand of an operator in which the usual arithmetic conversions are performed then the other operand shall not have wider essential type	Required	Decidable	**MISRA2012-RULE-10_7_a:** If a composite expression is used as one operand of an operator in which the usual arithmetic conversions are performed then the other operand shall not have wider essential type **MISRA2012-RULE-10_7_b:** If a composite expression is used as one (second or third) operand of a conditional operator then the other operand shall not have wider essential type
Rule-10.8: The value of a composite expression shall not be cast to a different essential type category or a wider essential type	Required	Decidable	**MISRA2012-RULE-10_8:** The value of a composite expression shall not be cast to a different essential type category or a wider essential type
Rule-11.1: Conversions shall not be performed between a pointer to a function and any other type	Required	Decidable	**MISRA2012-RULE-11_1_a:** Conversions shall not be performed between a pointer to a function and any other type **MISRA2012-RULE-11_1_b:** Conversions shall not be performed between a pointer to a function and any other type
Rule-11.2: Conversions shall not be performed between a pointer to an incomplete type and any other type	Required	Decidable	**MISRA2012-RULE-11_2:** Conversions shall not be performed between a pointer to an incomplete type and any other type
Rule-11.3: A cast shall not be per- formed between a pointer to object type and a pointer to a different object type	Required	Decidable	**MISRA2012-RULE-11_3:** A cast shall not be performed between a pointer to object type and a pointer to a different object type
Rule-11.4: A conversion should not be performed between a pointer to object and an integer type	Advisory	Decidable	**MISRA2012-RULE-11_4:** A conversion should not be performed between a pointer to object and an integer type
Rule-11.5: A conversion should not be performed from pointer to void into pointer to object	Advisory	Decidable	**MISRA2012-RULE-11_5:** A conversion should not be performed from pointer to void into pointer to object
Rule-11.6: A cast shall not be per- formed between pointer to void and an arithmetic type	Required	Decidable	**MISRA2012-RULE-11_6:** A cast shall not be performed between pointer to void and an arithmetic type

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-11.7: A cast shall not be performed between pointer to object and a non-integer arithmetic type	Required	Decidable	**MISRA2012-RULE-11_7:** A cast shall not be performed between pointer to object and a non-integer arithmetic type
Rule-11.8: A cast shall not remove any const or volatile qualification from the type pointed to by a pointer	Required	Decidable	**MISRA2012-RULE-11_8:** A cast shall not remove any const or volatile qualification from the type pointed to by a pointer
Rule-11.9: The macro NULL shall be the only permitted form of integer null pointer constant	Required	Decidable	**MISRA2012-RULE-11_9_a:** The macro NULL shall be the only permitted form of integer null pointer constant **MISRA2012-RULE-11_9_b:** The macro NULL shall be the only permitted form of integer null pointer constant
Rule-12.1: The precedence of operators within expressions should be made explicit	Advisory	Decidable	**MISRA2012-RULE-12_1_a:** Use parentheses unless all operators in the expression are the same **MISRA2012-RULE-12_1_b:** The operands of a logical && or || shall be primary-expressions **MISRA2012-RULE-12_1_c:** Parenthesis shall be used with the 'return' and 'sizeof' statements
Rule-12.2: The right hand operand of a shift operator shall lie in the range	Required	Undecidable	**MISRA2012-RULE-12_2:** The right hand operand of a shift operator shall lie in the range zero to one less than the width in bits of the essential type of the left hand operand
Rule-12.3: The of the essential type of the left hand should not be used operand	Advisory	Decidable	**MISRA2012-RULE-12_3:** The comma operator should not be used
Rule-12.4: Evaluation of constant expressions should not lead to unsigned integer wrap-around	Advisory	Decidable	**MISRA2012-RULE-12_4_a:** Integer overflow or underflow in constant expression in '+', '-', '*' operator **MISRA2012-RULE-12_4_b:** Integer overflow or underflow in constant expression in '<<' operator
Rule-13.1: Initializer lists shall not contain persistent side effects	Required	Undecidable	**MISRA2012-RULE-13_1_a:** Initializer lists shall not contain persistent side effects

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-13.2: The value of an expression and its persistent side effects shall be the same under all permitted evaluation orders	Required	Undecidable	**MISRA2012-RULE-13_2_a:** The value of an expression shall be the same under any order of evaluation that the standard permits **MISRA2012-RULE-13_2_b:** Don't write code that depends on the order of evaluation of function arguments **MISRA2012-RULE-13_2_c:** Don't write code that depends on the order of evaluation of function designator and function arguments **MISRA2012-RULE-13_2_d:** Don't write code that depends on the order of evaluation of expression that involves a function call **MISRA2012-RULE-13_2_e:** Between sequence points an object shall have its stored value modified at most once by the evaluation of an expression **MISRA2012-RULE-13_2_f:** Do not use more than one volatile in one expression **MISRA2012-RULE-13_2_g:** Don't write code that depends on the order of evaluation of function calls
Rule-13.3: A full expression containing an increment (++) or decrement (–) operator should have no other potential side effects other than that caused by the increment or decrement operator	Advisory	Decidable	**MISRA2012-RULE-13_3:** A full expression containing an increment (++) or decrement (--) operator should have no other potential side effects other than that caused by the increment or decrement operator
Rule-13.4: The result of an assignment operator should not be used	Advisory	Decidable	**MISRA2012-RULE-13_4:** The result of an assignment operator should not be used
Rule-13.5: The right hand operand of a logical && or || operator shall not contain persistent side effects	Required	Undecidable	**MISRA2012-RULE-13_5:** The right hand operand of a logical && or || operator shall not contain persistent side effects
Rule-13.6: The operand of the sizeof operator shall not contain any expression which has potential side effects	Mandatory	Decidable	**MISRA2012-RULE-13_6_a:** The operand of the sizeof operator shall not contain any expression which has potential side effects **MISRA2012-RULE-13_6_b:** The operand of the sizeof operator shall not contain any expression which has potential side effects **MISRA2012-RULE-13_6_c:** The operand of the sizeof operator shall not contain any expression which has potential side effects

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-14.1: A loop counter shall not have essentially floating type	Required	Undecidable	**MISRA2012-RULE-14_1_a:** A loop counter in a 'for' loop shall not have essentially floating type **MISRA2012-RULE-14_1_b:** A loop counter in 'while' and 'do-while' loops shall not have essentially floating type
Rule-14.2: A for loop shall be well formed	Required	Undecidable	**MISRA2012-RULE-14_2_a:** There shall only be one loop counter in a 'for' loop, which shall not be modified in the 'for' loop body **MISRA2012-RULE-14_2_b:** The first clause of a 'for' loop shall be well-formed **MISRA2012-RULE-14_2_c:** The second clause of a 'for' loop shall be well-formed **MISRA2012-RULE-14_2_d:** The third clause of a 'for' statement shall be well-formed

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-14.3: Controlling expressions shall not be invariant	Required	Undecidable	**MISRA2012-RULE-14_3_a:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_b:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_c:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_d:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_e:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_f:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_g:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_h:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_i:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_j:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_k:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_l:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_m:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_n:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_o:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_p:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_q:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_r:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_s:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_t:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_u:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_v:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_w:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_x:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_y:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_z:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_za:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_zb:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_zc*:** Controlling expressions shall not be invariant **MISRA2012-RULE-14_3_zd*:** Avoid switch with unreachable branches

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-14.4: The controlling expression of an if statement and the controlling expression of an iteration-statement shall have essentially Boolean type	Required	Decidable	**MISRA2012-RULE-14_4:** The controlling expression of an if statement and the controlling expression of an iteration-statement shall have essentially Boolean type
Rule-15.1: The goto statement should not be used	Advisory	Decidable	**MISRA2012-RULE-15_1:** The goto statement should not be used
Rule-15.2: The goto statement shall jump to a label declared later in the same function	Required	Decidable	**MISRA2012-RULE-15_2:** The goto statement shall jump to a label declared later in the same function
Rule-15.3: Any label referenced by a goto statement shall be declared in the same block, or in any block enclosing the goto statement	Required	Decidable	**MISRA2012-RULE-15_3:** Any label referenced by a goto statement shall be declared in the same block, or in any block enclosing the goto statement
Rule-15.4: There should be no more than one break or goto statement used to terminate any iteration statement	Advisory	Decidable	**MISRA2012-RULE-15_4:** There should be no more than one break or goto statement used to terminate any iteration statement
Rule-15.5: A function should have a single point of exit at the end	Advisory	Decidable	**MISRA2012-RULE-15_5:** A function should have a single point of exit at the end
Rule-15.6: The body of an iteration- statement or a selection-statement shall be a compound statement	Required	Decidable	**MISRA2012-RULE-15_6_a:** The body of an iteration-statement or a selection-statement shall be a compound-statement **MISRA2012-RULE-15_6_b:** The body of an iteration-statement or a selection-statement shall be a compound-statement
Rule-15.7: All if … else if constructs shall be terminated with an else statement	Required	Decidable	**MISRA2012-RULE-15_7:** All 'if … else if' constructs shall be terminated with an 'else' statement

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-16.1: All switch statements shall be well-formed	Required	Decidable	**MISRA2012-RULE-16_1_a:** A switch statement shall only contain switch labels and switch clauses, and no other code **MISRA2012-RULE-16_1_b:** A switch label shall only be used when the most closely-enclosing compound statement is the body of a switch statement **MISRA2012-RULE-16_1_c:** An unconditional break statement shall terminate every non-empty case clause **MISRA2012-RULE-16_1_d:** An unconditional break statement shall terminate every non-empty default clause **MISRA2012-RULE-16_1_e:** Always provide a default branch for switch statements **MISRA2012-RULE-16_1_f:** A 'default' label shall have a statement or a comment before terminating 'break' **MISRA2012-RULE-16_1_g:** A 'default' label, if it exists, shall appear as either the first or the last switch label of a switch statement **MISRA2012-RULE-16_1_h:** Every switch statement shall have at least two switch-clauses
Rule-16.2: A switch label shall only be used when the most closely-enclosing compound statement is the body of a switch statement	Required	Decidable	**MISRA2012-RULE-16_2:** A switch label shall only be used when the most closely-enclosing compound statement is the body of a switch statement
Rule-16.3: An unconditional break statement shall terminate every switch-clause	Required	Decidable	**MISRA2012-RULE-16_3_a:** An unconditional break statement shall terminate every switch-clause **MISRA2012-RULE-16_3_b:** An unconditional break statement shall terminate every switch-clause
Rule-16.4: Every switch statement shall have a default label	Required	Decidable	**MISRA2012-RULE-16_4_a:** Every 'switch' statement shall have a 'default' label **MISRA2012-RULE-16_4_b:** A 'default' label shall have a statement or a comment before terminating 'break'
Rule-16.5: A default label shall appear as either the first or the last switch label of a switch statement	Required	Decidable	**MISRA2012-RULE-16_5:** A default label shall appear as either the first or the last switch label of a switch statement
Rule-16.6: Every switch statement shall have at least two switch-clauses	Required	Decidable	**MISRA2012-RULE-16_6:** Every switch statement shall have at least two switch-clauses

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-16.7: A switch-expression shall not have essentially Boolean type	Required	Decidable	**MISRA2012-RULE-16_7_a:** A switch-expression shall not have essentially Boolean type **MISRA2012-RULE-16_7_b:** A switch-expression shall not have essentially Boolean type
Rule-17.1: The features of <stdarg.h> shall not be used	Required	Decidable	**MISRA2012-RULE-17_1_a:** The features of <stdarg.h> shall not be used **MISRA2012-RULE-17_1_b:** The features of <stdarg.h> shall not be used
Rule-17.2: Functions shall not call themselves, either directly or indirectly	Required	Undecidable	**MISRA2012-RULE-17_2:** Functions shall not call themselves, either directly or indirectly
Rule-17.3: A function shall not be declared implicitly	Mandatory	Decidable	**MISRA2012-RULE-17_3:** A function shall not be declared implicitly
Rule-17.4: All exit paths from a function with non-void return type shall have an explicit return statement with an expression	Mandatory	Decidable	**MISRA2012-RULE-17_4:** All exit paths from a function with non-void return type shall have an explicit return statement with an expression
Rule-17.5: The function argument corresponding to a parameter declared to have an array type shall have an appropriate number of elements	Advisory	Undecidable	**MISRA2012-RULE-17_5:** The function argument corresponding to a parameter declared to have an array type shall have an appropriate number of elements
Rule-17.6: The declaration of an array parameter shall not contain the static keyword between the [ ]	Mandatory	Decidable	**MISRA2012-RULE-17_6:** The declaration of an array parameter shall not contain the ‘static’ keyword between the [ ]
Rule-17.7: The value returned by a function having non-void return type shall be used	Required	Decidable	**MISRA2012-RULE-17_7_a:** The value returned by a function having non-void return type shall be used **MISRA2012-RULE-17_7_b:** The value returned by a function having non-void return type shall be used
Rule-17.8: A function parameter should not be modified	Advisory	Undecidable	**MISRA2012-RULE-17_8:** A function parameter should not be modified

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-18.1: A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand	Required	Undecidable	**MISRA2012-RULE-18_1_a*:** Avoid accessing arrays out of bounds **MISRA2012-RULE-18_1_b*:** Avoid accessing arrays and pointers out of bounds **MISRA2012-RULE-18_1_c*:** A pointer operand, as well as any pointer resulting from pointer arithmetic using that operand, shall address elements of the same array
Rule-18.2: Subtraction between pointers shall only be applied to pointers that address elements of the same array	Required	Undecidable	**MISRA2012-RULE-18_2:** Subtraction between pointers shall only be applied to pointers that address elements of the same array
Rule-18.3: The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object	Required	Undecidable	**MISRA2012-RULE-18_3:** >, >=, <, <= shall not be applied to objects of pointer type, except where they point to the same array
Rule-18.4: The +, -, += and -= operators should not be applied to an expression of pointer type	Advisory	Decidable	**MISRA2012-RULE-18_4:** The +, -, += and -= operators should not be applied to an expression of pointer type
Rule-18.5: Declarations should contain no more than two levels of pointer nesting	Advisory	Decidable	**MISRA2012-RULE-18_5:** Declarations should contain no more than two levels of pointer nesting
Rule-18.6: The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist	Required	Undecidable	**MISRA2012-RULE-18_6_a:** The address of an object with automatic storage shall not be returned from a function **MISRA2012-RULE-18_6_b:** 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
Rule-18.7: Flexible array members shall not be declared	Required	Decidable	**MISRA2012-RULE-18_7:** Flexible array members shall not be declared
Rule-18.8: Variable length array types shall not be used	Required	Decidable	**MISRA2012-RULE-18_8:** Variable-length array types shall not be used
Rule-19.1: An object shall not be assigned or copied to an overlapping object	Mandatory	Undecidable	**MISRA2012-RULE-19_1_a:** An object shall not be assigned or copied to an overlapping object **MISRA2012-RULE-19_1_b:** An object shall not be assigned or copied to an overlapping object
Rule-19.2: The union keyword should not be used	Advisory	Decidable	**MISRA2012-RULE-19_2:** The union keyword should not be used

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-20.1: #include directives should only be preceded by preprocessor directives or comments	Advisory	Decidable	**MISRA2012-RULE-20_1:** #include directives should only be preceded by preprocessor directives or comments
Rule-20.2: The ',' or \ characters and the /* or // character sequences shall not occur in a header file name	Required	Decidable	**MISRA2012-RULE-20_2_a:** The ‘, & or \ characters and the /* or // character sequences shall not occur in a header file name **MISRA2012-RULE-20_2_b:** The ‘, & or \ characters and the /* or // character sequences shall not occur in a header file name
Rule-20.3: The #include directive shall be followed by either a <file- name> or “filename” sequence	Required	Decidable	**MISRA2012-RULE-20_3:** The #include directive shall be followed by either a <filename> or “filename” sequence
Rule-20.4: A macro shall not be defined with the same name as a keyword	Required	Decidable	**MISRA2012-RULE-20_4_a:** A macro shall not be defined with the same name as a keyword **MISRA2012-RULE-20_4_b:** A macro shall not be defined with the same name as a keyword
Rule-20.5: #undef should not be used	Advisory	Decidable	**MISRA2012-RULE-20_5:** #undef should not be used
Rule-20.6: Tokens that look like a preprocessing directive shall not occur within a macro argument	Required	Decidable	**MISRA2012-RULE-20_6:** Tokens that look like a preprocessing directive shall not occur within a macro argument
Rule-20.7: Expressions resulting from the expansion of macro parameters shall be enclosed in parentheses	Required	Decidable	**MISRA2012-RULE-20_7:** Expressions resulting from the expansion of macro parameters shall be enclosed in parentheses
Rule-20.8: The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1	Required	Decidable	**MISRA2012-RULE-20_8:** The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1
Rule-20.9: All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be #define’d before evaluation	Required	Decidable	**MISRA2012-RULE-20_9_a:** All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be #define’d before evaluation **MISRA2012-RULE-20_9_b:** All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be #define’d before evaluation

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-20.10: The # and ## preprocessor operators should not be used	Advisory	Decidable	**MISRA2012-RULE-20_10:** The # and ## preprocessor operators should not be used
Rule-20.11: A macro parameter immediately following a # operator shall not immediately be followed by a ## operator	Required	Decidable	**MISRA2012-RULE-20_11:** A macro parameter immediately following a # operator shall not immediately be followed by a ## operator
Rule-20.12: A macro parameter used as an operand to the # or ## opera- tors, which is itself subject to further macro replacement, shall only be used as an operand to these operators	Required	Decidable	**MISRA2012-RULE-20_12:** A macro parameter used as an operand to the # or ## operators, which is itself subject to further macro replacement, shall only be used as an operand to these operators
Rule-20.13: A line whose first token is # shall be a valid preprocessing directive	Required	Decidable	**MISRA2012-RULE-20_13:** A line whose first token is # shall be a valid preprocessing directive
Rule-20.14: All #else, #elif and #endif preprocessor directives shall reside in the same file as the #if, #ifdef or#ifndef directive to which they are related	Required	Decidable	**MISRA2012-RULE-20_14:** All #else, #elif and #endif preprocessor directives shall reside in the same file as the #if, #ifdef or #ifndef directive to which they are related
Rule-21.1: #define and #undef shall not be used on a reserved identifier or reserved macro name	Required	Decidable	**MISRA2012-RULE-21_1_a:** Do not #define or #undef identifiers with names which start with underscore **MISRA2012-RULE-21_1_b:** #define and #undef shall not be used on a reserved identifier or reserved macro name (for C90 code) **MISRA2012-RULE-21_1_c:** #define and #undef shall not be used on a reserved identifier or reserved macro name (for C99 code) **MISRA2012-RULE-21_1_d:** Do not #define nor #undef identifier 'defined'
Rule-21.2: A reserved identifier or macro name shall not be declared	Required	Decidable	**MISRA2012-RULE-21_2_a:** An identifier with name which starts with underscore shall not be declared **MISRA2012-RULE-21_2_b:** A reserved identifier or macro name shall not be declared (for C90 code) **MISRA2012-RULE-21_2_c:** A reserved identifier or macro name shall not be declared (for C99 code)

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-21.3: The memory allocation and deallocation functions of <stdlib.h> shall not be used	Required	Decidable	**MISRA2012-RULE-21_3:** The memory allocation and deallocation functions of <stdlib.h> shall not be used
Rule-21.4: The standard header file <setjmp.h> shall not be used	Required	Decidable	**MISRA2012-RULE-21_4_a:** The standard header file <setjmp.h> shall not be used **MISRA2012-RULE-21_4_b:** The standard header file <setjmp.h> shall not be used
Rule-21.5: The standard header file <signal.h> shall not be used	Required	Decidable	**MISRA2012-RULE-21_5_a:** The standard header file <signal.h> shall not be used **MISRA2012-RULE-21_5_b:** The standard header file <signal.h> shall not be used
Rule-21.6: The Standard Library input/output functions shall not be used	Required	Decidable	**MISRA2012-RULE-21_6:** The Standard Library input/output functions shall not be used
Rule-21.7: The atof, atoi, atol and atoll functions of <stdlib.h> shall not be used	Required	Decidable	**MISRA2012-RULE-21_7:** The atof, atoi, atol and atoll functions of <stdlib.h> shall not be used
Rule-21.8: The library functions abort, exit, getenv and system of <stdlib.h> shall not be used	Required	Decidable	**MISRA2012-RULE-21_8:** The library functions abort, exit, getenv and system of <stdlib.h> shall not be used
Rule-21.9: The library functions bsearch and qsort of <stdlib.h> shall not be used	Required	Decidable	**MISRA2012-RULE-21_9:** The library functions bsearch and qsort of <stdlib.h> shall not be used
Rule-21.10: The Standard Library time and date functions shall not be used	Required	Decidable	**MISRA2012-RULE-21_10:** The Standard Library time and date functions shall not be used
Rule-21.11: The standard header file <tgmath.h> shall not be used	Required	Decidable	**MISRA2012-RULE-21_11:** The standard header file <tgmath.h> shall not be used
Rule-21.12: The exception handling features of <fenv.h> should not be used	Advisory	Decidable	**MISRA2012-RULE-21_12:** The exception handling features of <fenv.h> should not be used
Rule-22.1: All resources obtained dynamically by means of Standard Library functions shall be explicitly released	Required	Undecidable	**MISRA2012-RULE-22_1*:** All resources obtained dynamically by means of Standard Library functions shall be explicitly released

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-22.2: A block of memory shall only be freed if it was allocated by means of a Standard Library function	Mandatory	Undecidable	**MISRA2012-RULE-22_2_a*:** Do not use resources that have been freed **MISRA2012-RULE-22_2_b*:** Do not free resources using invalid pointers
Rule-22.3: The same file shall not be open for read and write access at the same time on different streams	Required	Undecidable	**MISRA2012-RULE-22_3*:** The same file shall not be opened for read and write access at the same time on different stream
Rule-22.4: There shall be no attempt to write to a stream which has been opened as read-only	Mandatory	Undecidable	**MISRA2012-RULE-22_4*:** Avoid writing to a stream which has been opened as read only
Rule-22.5: A pointer to a FILE object shall not be dereferenced	Mandatory	Undecidable	**MISRA2012-RULE-22_5_a:** A pointer to a FILE object shall not be dereferenced **MISRA2012-RULE-22_5_b:** A pointer to a FILE object shall not be dereferenced by a library function
Rule-22.6: The value of a pointer to a FILE shall not be used after the associated stream has been closed	Mandatory	Undecidable	**MISRA2012-RULE-22_6*:** The value of a pointer to a FILE shall not be used after the associated stream has been closed

MISRA C:2012 Amendment 1 Additional security guidelines for MISRA C:2012

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Dir-4.14: The validity of values received from external sources shall be checked	Required	—	**MISRA2012-DIR-4_14_a***: Avoid tainted data in array indexes **MISRA2012-DIR-4_14_b***: Protect against integer overflow/underflow from tainted data **MISRA2012-DIR-4_14_c***: Avoid buffer read overflow from tainted data **MISRA2012-DIR-4_14_d***: Avoid buffer write overflow from tainted data **MISRA2012-DIR-4_14_e***: Protect against command injection **MISRA2012-DIR-4_14_f***: Protect against file name injection **MISRA2012-DIR-4_14_g***: Protect against SQL injection **MISRA2012-DIR-4_14_h***: Prevent buffer overflows from tainted data **MISRA2012-DIR-4_14_i***: Avoid buffer overflow from tainted data due to defining incorrect format limits **MISRA2012-DIR-4_14_j*:** Protect against environment injection **MISRA2012-DIR-4_14_k***: Avoid printing tainted data on the output console
Rule-12.5: The sizeof operator shall not have an operand which is a function parameter declared as "array of type"	Mandatory	Decidable	**MISRA2012-RULE-12_5:** The 'sizeof' operator shall not have an operand which is a function parameter declared as "array of type"
Rule-21.13: Any value passed to a function in <ctype.h> shall be representable as an unsigned char or be the value EOF	Mandatory	Undecidable	**MISRA2012-RULE-21_13*:** Any value passed to a function in <ctype.h> shall be representable as an 'unsigned char' or be the value 'EOF'
Rule-21.14: The Standard Library function memcmp shall not be used to compare null terminated strings	Required	Undecidable	**MISRA2012-RULE-21_14*:** The Standard Library function 'memcmp' shall not be used to compare null-terminated strings
Rule-21.15: The pointer arguments to the Standard Library functions memcpy, memmove and memcmp shall be pointers to quali=ed or unquali=ed versions of compatible types	Required	Decidable	**MISRA2012-RULE-21_15:** The pointer arguments to the Standard Library functions 'memcmp', 'memmove' and 'memcmp' shall be pointers to qualified or unqualified versions of compatible types
Rule-21.16: The pointer arguments to the Standard Library function memcmp shall point to either a pointer type, an essentially signed type, an essentially unsigned type, an essentially Boolean type or an essentially enum type	Required	Decidable	**MISRA2012-RULE-21_16:** The pointer arguments to the Standard Library function 'memcmp' shall point to either a pointer type, an essentially signed type, an essentially unsigned type, an essentially Boolean type or an essentially enum type

MISRA ID and Description	Classification	Decidability	Parasoft ID and Description
Rule-21.17: Use of the string handling functions from <string.h> shall not result in accesses beyond the bounds of the objects referenced by their pointer parameters	Mandatory	Undecidable	**MISRA2012-RULE-21_17_a***: Avoid overflow due to reading a not zero terminated string **MISRA2012-RULE-21_17_b***: Avoid overflow when writing to a buffer
Rule-21.18: The size_t argument passed to any function in <string.h> shall have an appropriate value	Mandatory	Undecidable	**MISRA2012-RULE-21_18*:** The 'size_t' argument passed to any function in <string.h> shall have an appropriate value
Rule-21.19: The pointers returned by the Standard Library functions localeconv, getenv, setlocale or, strerror shall only be used as if they have pointer to const-quali=ed type	Mandatory	Undecidable	**MISRA2012-RULE-21_19_a:** The pointers returned by the Standard Library functions 'localeconv', 'getenv', 'setlocale' or, 'strerror' shall only be used as if they have pointer to const-qualified type **MISRA2012-RULE-21_19_b:** Strings pointed by members of the structure 'lconv' should not be modified
Rule-21.20: The pointer returned by the Standard Library functions asctime, ctime, gmtime, localtime, localeconv, getenv, setlocale or strerror shall not be used following a subsequent call to the same function	Mandatory	Undecidable	**MISRA2012-RULE-21_20*:** Pointers returned by certain Standard Library functions should not be used following a subsequent call to the same or related function
Rule-22.7: The macro EOF shall only be compared with the unmodified return value from any Standard Library function capable of returning EOF	Required	Undecidable	**MISRA2012-RULE-22_7*:** The value of a pointer to a FILE shall not be used after the associated stream has been closed
Rule-22.8: The value of errno shall be set to zero prior to a call to an errnosetting-function	Required	Undecidable	**MISRA2012-RULE-22_8*:** The value of 'errno' shall be set to zero prior to a call to an errno-setting-function
Rule-22.9: The value of errno shall be tested against zero after calling an errnosetting-function	Required	Undecidable	**MISRA2012-RULE-22_9*:** The value of 'errno' shall be tested against zero after calling an errno-setting-function
Rule-22.10: The value of errno shall only be tested when the last function to be called was an errno-setting-function	Required	Undecidable	**MISRA2012-RULE-22_10*:** The value of 'errno' shall only be tested when the last function to be called was an errno-setting-function

* – Denotes Flow Analysis rules. Flow Analysis rules require dedicated license feature.

ABOUT PARASOFT

Parasoft helps organizations perfect today's highly-connected applications by automating time-consuming testing tasks and providing management with intelligent analytics necessary to focus on what matters. Parasoft's technologies reduce the time, effort, and cost of delivering secure, reliable, and compliant software, by integrating static and runtime analysis; unit, functional, and API testing; and service virtualization. With developer testing tools, manager reporting/analytics, and executive dashboarding, Parasoft supports software organizations with the innovative tools they need to successfully develop and deploy applications in the embedded, enterprise, and IoT markets, all while enabling today's most strategic development initiatives — agile, continuous testing, DevOps, and security.

www.parasoft.com

Parasoft Headquarters:
+1-626-256-3680

Parasoft EMEA:
+31-70-3922000

Parasoft APAC:
+65-6338-3628

PARASOFT®
Automated Software Testing