klocwork

Klocwork ISO 26262 / IEC 61508 / IEC 62304 / EN 50128 Certified Checkers

Document Number: KW2023_2_ 003

Certified version: Klocwork 2023.2

Author	Revision	Comments	Date
AWeekes	0.1	Initial draft created	May-09-13
MTooke	0.2	Removed MISRA.FOR.INIT which is no longer part of SR4.	May-16-13
AWeekes	0.3	Extended to include IEC 61508	May-28-13
MTooke	0.4	Added default checkers to MISRA lists	July-11-13
AWeekes	0.5	Updated for 2015 re-certification	June-15-15
SBommaganti	1.0	Updated for Klocwork 2016	Mar-3-16
SBommaganti	1.1	Updated list for Klocwork 2016.1	June-25-16
SBommaganti	1.2	Updated list for Klocwork 2016.3	Nov-02-16
SBommaganti	2.0	Updated list for Klocwork 2017	Mar-2-17
SBommaganti	2.1	Updated list for Klocwork 2017.1 and added EN 50128	July-5-17
SBommaganti	2.2	Updated list for Klocwork 2017.2	Oct-16-17
SBommaganti	2.3	Updated list for Klocwork 2017.3	Nov-2-17
MTofinetti	3.0	Updated for Klocwork 2018	2018-04-03
MTofinetti	3.1	Updated for Klocwork 2018.1	2018-06-27
MTofinetti	3.2	Updated for Klocwork 2018.2	2018-09-27
MTofinetti	3.3	Updated for Klocwork 2018.3	2018-11-28

MTofinetti	4.0	Updated for Klocwork 2019	2019-03-22
MTofinetti	4.1	Updated for Klocwork 2019.1	2019-05-14
MTofinetti	4.2	Updated for Klocwork 2019.2	2019-07-31
MTofinetti	4.3	Updated for Klocwork 2019.3; IEC 62304 added	2019-12-16
LRobertson	5.0	Updated for Klocwork 2020.1	2020-03-12
ABedford	5.1	Updated for Klocwork 2020.2	2020-06-29
ABedford	5.2	Updated for Klocwork 2020.3	2020-09-14
ABedford	5.3	Updated for Klocwork 2020.4	2021-02-24
ADunster	6.0	Updated for Klocwork 2021.1	2021-04-26
ADunster	6.1	Updated for Klocwork 2021.2	2021-08-17
ADunster	6.2	Updated for Klocwork 2021.3	2021-11-30
ADunster	6.3	Updated for Klocwork 2021.4	2022-01-18
JBritton	6.4	Updated for Klocwork 2022.2 and rebrand	2022-06-30
JBritton	6.5	Updated for Klocwork 2022.4	2022-12-17
JBritton	6.6	Updated for Klocwork 2023.2	2023-07-26

Contents

Referenced Standards

Standards referenced in this document refer to the following versions:

Standard	Version
ISO 26262	ISO 26262:2018
IEC 61508	IEC 61508:2010
IEC 62304	IEC 62304:2006/AMD1:2015
EN 50128	EN 50128:2011/A2:2020

Trademarks

"MISRA", "MISRA C" and "MISRA C++" are registered trademarks of The MISRA Consortium Limited.

Windows is a registered trademark of Microsoft Corporation.

Related Documents

Document ID	Title
KW2023_2_001	Functional Safety Manual for Klocwork
KW2023_2_005	Klocwork Checker Qualification Pack
KW2023_2_006	Klocwork Architecture
KW2023_2_009	Klocwork 2023.1 and 2023.2 Release Notes

Klocwork Checkers Certified for ISO 26262 / IEC 61508 / IEC 62304 / EN 50128

Purpose

This document is a complete catalog of checkers included in the ISO 26262 / IEC 61508 / EN 50128 certification.

For the purposes of this certification, Klocwork has included C/C++ checkers that are labeled with support level 1 (as described in https://help.klocwork.com/current/en-us/concepts/home.htm) that are part of the following requirements:

- MISRA C:2004 (MISRA C2),
- MISRA C++:2008,
- MISRA C:2012 and
- Severity 1 or Severity 2 checkers as designated in a default Klocwork installation.

Referenced resources

The following referenced documents are used within the project.

Standard	Location
MISRA C:2004 (MISRA C2)	http://www.misra.org.uk/Publications/
MISRA C++:2008	http://www.misra.org.uk/Publications/
MISRA C:2012 (MISRA C3)	http://www.misra.org.uk/Publications/
Klocwork documentation C and C++ checker reference	https://help.klocwork.com/current/en-us/concepts/home.htm

MISRA, The Motor Industry Software Reliability Association, is an industry consortium that establishes and promotes best practices in engineering in the automotive industry. Their standards have been adopted by many industries and associations worldwide.

C/C++ checkers, severities 1 and 2

CHECKER	DESCRIPTION
ABV.ANY_SIZE_ARRAY	Buffer Overflow - Array Index Out of Bounds
ABV.GENERAL	Buffer Overflow - Array Index Out of Bounds
ABV.GENERAL.MULTIDIMENSION	Buffer Overflow – Multi-Dimensional Array Index Out of Bounds
ABV.ITERATOR	Buffer Overflow - Array Index may be out of Bounds
ABV.MEMBER	Buffer Overflow - Array Index Out of Bounds
ABV.STACK	Buffer Overflow - Local Array Index Out of Bounds
ABV.TAINTED	Buffer Overflow from Unvalidated Input
ABV.UNICODE.BOUND_MAP	Buffer overflow in mapping character function
ABV.UNICODE.FAILED_MAP	Mapping function failed
ABV.UNICODE.NNTS_MAP	Buffer overflow from non null-terminated string in mapping function
ABV.UNICODE.SELF_MAP	Mapping function failed
ABV.UNKNOWN_SIZE	Buffer Overflow - Array Index Out of Bounds
CERT.EXIT.HANDLER_TERMINATE	Exit function called from exit handler function
CL.MLK.VIRTUAL	Memory Leak - possible in destructor
CL.SELF-ASSIGN	Use of free memory (double free) - in operator=
CL.SHALLOW.ASSIGN	Use of free memory (double free) - shallow copy in operator=
CL.SHALLOW.COPY	Use of free memory (double free) - shallow copy in copy constructor
CONC.DBL_LOCK	Object was locked twice
CONC.DBL_UNLOCK	Object was unlocked twice
CONC.DL	Deadlock
CONC.NO_LOCK	Object was not locked
CONC.NO_UNLOCK	Missing unlock for variable
CWARN.DTOR.NONVIRT.DELETE	Delete expression for an object of a class with virtual methods and no virtual destructor
CWARN.DTOR.NONVIRT.NOTEMPTY	Class has virtual functions inherited from a base class, but its destructor is not virtual and not empty
CWARN.FUNCADDR	Function address is used instead of a call to this function
CXX.SQL.INJECT	Potential for malicious SQL injection
CXX.SV.PWD.PLAIN	The application must not display passwords/PINs as clear text
DBZ.CONST	Division by a zero constant occurs
DBZ.CONST.CALL	The value '0' is passed to function that can use this value as divisor
DBZ.GENERAL	Division by zero might occur
DBZ.ITERATOR	Division by zero might occur in a loop iterator
DBZ.ITERATOR.CALL	Division by zero might occur in a function call
FMM.MIGHT	Freeing Mismatched Memory - possible
FMM.MUST	Freeing Mismatched Memory
FNH.MIGHT	Freeing Non-Heap Memory - possible
FNH.MUST	Freeing Non-Heap Memory
FUM.GEN.MIGHT	Freeing Unallocated Memory - possible
FUM.GEN.MUST	Freeing Unallocated Memory
FUNCRET.GEN	Non-void function does not return value
FUNCRET.IMPLICIT	Non-void function implicitly returning int does not return value
HCC	Implements CWE-798: Use of Hard-coded Credentials; identifies the use of both hard-coded passwords and usernames
HCC.PWD	Implements CWE-798: Use of Hard-coded Credentials; identifies the use of hard-coded passwords
HCC.USER	Implements CWE-798: Use of Hard-coded Credentials; identifies the use of hard-coded usernames
INFINITE_LOOP.GLOBAL	Infinite loop
INFINITE_LOOP.LOCAL	Infinite loop
INFINITE_LOOP.MACRO	Infinite loop
LOCRET.ARG	Function returns address of local variable
LOCRET.GLOB	Function returns address of local variable
LOCRET.RET	Function returns address of local variable
MLK.MIGHT	Memory Leak - possible
MLK.MUST	Memory Leak
MLK.RET.MIGHT	Memory Leak - possible
MLK.RET.MUST	Memory Leak
NNTS.MIGHT	Buffer Overflow - Non-null Terminated String

CHECKER	DESCRIPTION
NNTS.MUST	Buffer Overflow - Non-null Terminated String
NNTS.TAINTED	Unvalidated User Input Causing Buffer Overflow - Non-Null Terminated String
NPD.CHECK.CALL.MIGHT	Pointer may be passed to function that can dereference it after it was positively checked for NULL
NPD.CHECK.CALL.MUST	Pointer will be passed to function that may dereference it after it was positively checked for NULL
NPD.CHECK.MIGHT	Pointer may be dereferenced after it was positively checked for NULL
NPD.CHECK.MUST	Pointer will be dereferenced after it was positively checked for NULL
NPD.CONST.CALL	NULL is passed to function that can dereference it
NPD.CONST.DEREF	NULL is dereferenced
NPD.FUNC.CALL.MIGHT	Result of function that may return NULL may be passed to another function that may dereference it
NPD.FUNC.CALL.MUST	Result of function that may return NULL will be passed to another function that may dereference it
NPD.FUNC.MIGHT	Result of function that can return NULL may be dereferenced
NPD.FUNC.MUST	Result of function that may return NULL will be dereferenced
NPD.GEN.CALL.MIGHT	Null pointer may be passed to function that may dereference it
NPD.GEN.CALL.MUST	Null pointer will be passed to function that may dereference it
NPD.GEN.MIGHT	Null pointer may be dereferenced
NPD.GEN.MUST	Null pointer will be dereferenced
RABV.CHECK	Suspicious use of index before boundary check
RCA	Risky cryptographic algorithm used
RCA.HASH.SALT.EMPTY	Implements CWE-759: Use of a One-Way Hash without a Salt
RETVOID.GEN	Non-void function returns void value
RETVOID.IMPLICIT	Implicitly int function returns void value
RH.LEAK	Resource leak
RN.INDEX	Suspicious use of index before negative check
RNPD.CALL	Suspicious dereference of pointer in function call before NULL check
RNPD.DEREF	Suspicious dereference of pointer before NULL check
SV.DLLPRELOAD.NONABSOLUTE.DLL	Potential DLL-preload hijack vector
SV.DLLPRELOAD.NONABSOLUTE.EXE	Potential DLL-preload process-injection vector
SV.DLLPRELOAD.SEARCHPATH	Potential DLL-preload SearchPath vector
SV.FMTSTR.GENERIC	Format String Vulnerability
SV.FMT_STR.BAD_SCAN_FORMAT	Input format specifier error
SV.FMT_STR.PRINT_FORMAT_MISMATCH.BAD	Incompatible type of a print function parameter
SV.FMT_STR.PRINT_IMPROP_LENGTH	Improper use of length modifier in a print function call
SV.FMT_STR.PRINT_PARAMS_WRONGNUM.FEW	Too few arguments in a print function call
SV.FMT_STR.PRINT_PARAMS_WRONGNUM.MANY	Too many arguments in a print function call
SV.FMT_STR.SCAN_FORMAT_MISMATCH.BAD	Incompatible type of a scan function parameter
SV.FMT_STR.SCAN_FORMAT_MISMATCH.UNDESIRED	Unexpected type of a scan function parameter
SV.FMT_STR.SCAN_IMPROP_LENGTH	Improper use of length modifier in a scan function call
SV.FMT_STR.SCAN_PARAMS_WRONGNUM.FEW	Too few arguments in a scan function call
SV.FMT_STR.SCAN_PARAMS_WRONGNUM.MANY	Too many arguments in a scan function call
SV.STRBO.BOUND_COPY.OVERFLOW	Buffer Overflow in Bound String Copy
SV.STRBO.BOUND_COPY.UNTERM	Possible Buffer Overflow in Following String Operations
SV.STRBO.BOUND_SPRINTF	Buffer Overflow in Bound sprintf
SV.STRBO.UNBOUND_COPY	Buffer Overflow in Unbound String Copy
SV.STRBO.UNBOUND_SPRINTF	Buffer Overflow in Unbound sprintf
SV.TAINTED.ALLOC_SIZE	Use of Unvalidated Integer in Memory Allocation
SV.TAINTED.CALL.INDEX_ACCESS	Use of Unvalidated Integer as Array Index by Function Call
SV.TAINTED.CALL.LOOP_BOUND	Use of Unvalidated Integer in Loop Condition through a Function Call
SV.TAINTED.FMTSTR	Use of Unvalidated Data in a Format String
SV.TAINTED.INDEX_ACCESS	Use of Unvalidated Integer as Array Index
SV.TAINTED.LOOP_BOUND	Use of Unvalidated Integer in Loop Condition
SV.TAINTED.PATH_TRAVERSAL	Use of Unvalidated Data in a Path Traversal
SV.UNBOUND_STRING_INPUT.CIN	Usage of cin for unbounded string input
SV.UNBOUND_STRING_INPUT.FUNC	Usage of unbounded string input
UFM.DEREF.MIGHT	Use of free memory (access) - possible
UFM.DEREF.MUST	Use of Freed Memory by Pointer
UFM.FFM.MIGHT	Use of free memory (double free) - possible
UFM.FFM.MUST	Freeing Freed Memory
UFM.RETURN.MIGHT	Use of freed memory (return) - possible

CHECKER	DESCRIPTION
UFM.RETURN.MUST	Use of Freed Memory on Return
UFM.USE.MIGHT	Use of free memory - possible
UFM.USE.MUST	Use of Freed Memory
UNINIT.CTOR.MIGHT	Uninitialized Variable in Constructor - possible
UNINIT.CTOR.MUST	Uninitialized Variable in Constructor
UNINIT.HEAP.MIGHT	Uninitialized Heap Use - possible
UNINIT.HEAP.MUST	Uninitialized Heap Use
UNINIT.STACK.ARRAY.MIGHT	Uninitialized Array - possible
UNINIT.STACK.ARRAY.MUST	Uninitialized Array
UNINIT.STACK.ARRAY.PARTIAL.MUST	Partially Uninitialized Array
UNINIT.STACK.MIGHT	Uninitialized Variable - possible
UNINIT.STACK.MUST	Uninitialized Variable
VOIDRET	Void function returns value

MISRA C:2004 (MISRA C2) suite of checkers

CHECKER	DESCRIPTION	MISRA-C RULE
FUNCRET.GEN	Non-void function does not return value	16.8
FUNCRET.IMPLICIT	Non-void function implicitly returning int does not return value	16.8
INVARIANT_CONDITION.GEN	Invariant expression in a condition	13.7
INVARIANT_CONDITION.UNREACH	Invariant expression in a condition	13.7
LOCRET.ARG	Function returns address of local variable	17.6
LOCRET.GLOB	Function returns address of local variable	17.6
LOCRET.RET	Function returns address of local variable	17.6
MISRA.ASM.ENCAPS	Assembly language is not isolated.	2.1
MISRA.ASSIGN.COND	Assignment operator is used in a condition	13.1
MISRA.ASSIGN.OVERLAP	Object is assigned to an overlapping object	18.2
MISRA.BITFIELD.SIGNED	Length of a named signed bit-field is less than 2	6.5
MISRA.BITFIELD.SIGNED.UNNAMED	Length of an unnamed signed bit-field is less than 2	6.5
MISRA.BITFIELD.TYPE	Type of bit-field is not signed/unsigned integer	6.4
MISRA.BITS.NOT_UNSIGNED	Operand of bitwise operation is not unsigned integer	12.7
MISRA.BITS.NOT_UNSIGNED.PREP	Operand of bitwise operation in preprocessor directive #if or #elif is not unsigned integer	12.7
MISRA.BUILTIN_NUMERIC	Builtin numeric type is used	6.3
MISRA.CAST.CONST	Cast operation removes const or volatile modifier from a pointer or reference	11.5
MISRA.CAST.FLOAT	Non-trivial float expression is cast to a wider type	10.4
MISRA.CAST.FUNC_PTR	Cast between a function pointer and a non-integral type	11.1
MISRA.CAST.INT	Non-trivial integer expression is cast to a wider type, or type with a different signedness	10.3
MISRA.CAST.PTR	Cast between a pointer to object type and a different pointer to object type	11.4
MISRA.CAST.PTR.UNRELATED	Object of pointer type cast to unrelated type	11.2
MISRA.CAST.PTR_TO_INT	Cast between a pointer and an integral type	11.3
MISRA.CAST.UNSIGNED_BITS	The result of bitwise operation on unsigned char or short is not cast back to original type	10.5
MISRA.CHAR.NOT_CHARACTER	'char' is used for non-character value	6.1
MISRA.CHAR.TRIGRAPH	Trigraph usage	4.2
MISRA.COMMA	Comma operator is used	12.10
MISRA.COMP.WRAPAROUND	Wrap-around in a condition	12.11
MISRA.CONTINUE	Continue statement is used	14.5
MISRA.CT.UNIQUE.ID	Identifier clashes with type name	5.4
MISRA.CVALUE.IMPL.CAST	The value of an expression implicitly converted to a different type	10.1, 10.2
MISRA.DECL.ARRAY_SIZE	Declaration of array with unknown size	8.12
MISRA.DECL.FUNC_LOCAL	Function is declared locally	8.6
MISRA.DECL.NO_TYPE	Declaration without a type	8.2
MISRA.DEFINE.BADEXP	Inappropriate macro expansion	19.4
MISRA.DEFINE.FUNC	Function-like macro definition	19.7
MISRA.DEFINE.LONGNAME	Macro name is too long	5.1
MISRA.DEFINE.NOPARS	Macro parameter with no parentheses	19.10
MISRA.DEFINE.NOTGLOBAL	Define not at the global level	19.5
MISRA.DEFINE.SHARP	or ## operator in a macro definition	19.13
MISRA.DEFINE.SHARP.MANY	Several # or ## operators in a macro definition	19.12
MISRA.DEFINE.WRONGNAME	Usage of a name from the standard library for naming a macro	20.1
MISRA.DEFINE.WRONGNAME.UNDERSCORE	Usage of a reserved name for naming a macro	20.1
MISRA.ELIF.DEFINED	Incorrect 'defined' usage in #elif directive	19.14
MISRA.ELIF.OTHERFILE	#elif in an improper file	19.17
MISRA.ELIF.UNDEF	Undefined macros in #elif directive	19.11
MISRA.ELIF.WRAPAROUND	Wrap-around in #elif directive	12.11
MISRA.ELSE.OTHERFILE	#else in an improper file	19.17

MISRA.ENDIF.OTHERFILE	#endif in an improper file	19.17
MISRA.ENUM.INIT	Non-first enumerator is explicitly initialized, but not all elements are explicitly initialized.	9.3
MISRA.EXPANSION.DIRECTIVE	Directive-like tokens within a macro argument	19.9
MISRA.EXPANSION.NARGS	Missing macro argument	19.8
MISRA.EXPANSION.UNSAFE	Unsafe macro usage	20.1, 20.5, 20.6, 20.7, 20.8
MISRA.EXPR.PARENS.INSUFFICIENT	Limited dependence required for operator precedence rules in expressions	12.1
MISRA.EXPR.PARENS.REDUNDANT	Limited dependence required for operator precedence rules in expressions	12.1
MISRA.FLOAT.BIT.REPR	Use of bit manipulations of floating-point values which rely on storage layout	12.12
MISRA.FLOAT_EQUAL	Floating point expression is tested for equality	13.3
MISRA.FOR.COND	For loop condition does not depend on loop counter	13.5
MISRA.FOR.COND.FLT	Floating point object is used in the condition section of a 'for' loop	13.4
MISRA.FOR.INCR.CHANGE	For loop increment expression does not change loop counter	13.5
MISRA.FOR.STMT.CHANGE	For loop counter is modified within the loop statement	13.6
MISRA.FUNC.ADDR	Address of a function is used without & operator	16.9
MISRA.FUNC.NOPROT.CALL	Function is called but has no prototype	8.1
MISRA.FUNC.NOPROT.DEF	Function has a definition but no prototype	8.1
MISRA.FUNC.NO_PARAMS	Function without parameters is missing void parameter type	16.5
MISRA.FUNC.PARAMS.IDENT	Identifiers used in declaration and definition of function are no identical	16.4
MISRA.FUNC.RECUR	Recursive function	16.2
MISRA.FUNC.STATIC.REDECL	Function or object redeclaration does not include 'static' modifier	8.11
MISRA.FUNC.UNMATCHED.PARAMS	Number of formal and actual parameters passed to function do not match	16.6
MISRA.FUNC.UNNAMED.PARAMS	Function declaration has unnamed parameters	16.3
MISRA.FUNC.VARARG	Function with variable number of arguments	16.1
MISRA.GOTO	Goto statement is used	14.4
MISRA.IDENT.LONG	Identifier is longer than 31 characters	5.1
MISRA.IF.DEFINED	Incorrect 'defined' usage in #if directive	19.14
MISRA.IF.NO_COMPOUND	The body of if/else statement is not a compound statement	14.9
MISRA.IF.NO_ELSE	A chain of if/else-if statements is not terminated with else or is terminated with an empty else clause	14.10
MISRA.IF.UNDEF	Undefined macros in #if directive	19.11
MISRA.IF.WRAPAROUND	Wrap-around in #if directive	12.11
MISRA.INCGUARD	Include guard is not provided	19.15
MISRA.INCL.BAD	Non-standard include directive	19.3
MISRA.INCL.INSIDE	Include directive preceded by a preprocessor output token	19.1
MISRA.INCL.SYMS	Non-standard characters in header file names	19.2
MISRA.INCL.UNSAFE	Unsafe header inclusion	20.1, 20.8, 20.9, 20.12
MISRA.INCOMPLETE.STRUCT	Incomplete struct type is used	18.1
MISRA.INCOMPLETE.STRUCT.UNNAMED	Incomplete unnamed struct type is used	18.1
MISRA.INCOMPLETE.UNION	Incomplete union type is used	18.1
MISRA.INCOMPLETE.UNION.UNNAMED	Incomplete unnamed union type is used	18.1
MISRA.INCR_DECR.OTHER	Increment or decrement operator is mixed with other operators in expression	12.13
MISRA.INIT.BRACES	Incorrect initializer braces placement.	9.2
MISRA.ITER.ONETERM	Iteration statement has more than one break or goto for loop termination.	14.6
MISRA.LITERAL.UNSIGNED.SUFFIX	Unsigned integer literal without 'U' suffix	10.6
MISRA.LOGIC.OPERAND.NOT_BOOL	Operand of logical operator is not effectively boolean	12.6

MISRA.LOGIC.OPERATOR.NOT_BOOL	Operand of non-logical operator is effectively boolean	12.6
MISRA.LOGIC.PRIMARY	Operand in a logical 'and' or 'or' expression is not a primary expression	12.5
MISRA.LOGIC.SIDEEFF	Right operand in a logical 'and' or 'or' expression contains side effects	12.4
MISRA.LOGIC.SIDEEFF.COND	Branch expression in a conditional expression contains side effects	12.4
MISRA.NULL.STMT	Null statement is not the only statement on line or comments are placed incorrectly	14.3
MISRA.OBJ.TYPE.COMPAT	Type not compatible with type of other declaration	8.4
MISRA.OBJ.TYPE.IDENT	Type not identical with type of other declaration	8.3
MISRA.ONEDEFRULE.FUNC	Global function definition in a header file	8.5
MISRA.ONEDEFRULE.VAR	Global variable definition in a header file	8.5
MISRA.PPARAM.NEEDS.CONST	Pointer parameter is not used to modify the addressed object but is not declared as a pointer to const	16.7
MISRA.PRAGMA	Non-documented pragma directive	3.4
MISRA.PTR.ARITH	Pointer is used in arithmetic or array index expression	17.1, 17.4
MISRA.PTR.CMP.2008	Pointer comparison using comparison operators shall only be applied if pointing to same array and within the range	17.3
MISRA.PTR.CMP.OBJECT.2008	Pointer comparison using comparison operators shall only be applied if pointing to same object and within the range	17.3
MISRA_PTR.SUB	Pointer subtraction shall only be applied if pointing to same array	17.2
MISRA_PTR.SUB.OBJECT	Pointer subtraction shall only be applied if pointing to same object	17.2
MISRA.PTR.TO_PTR_TO_PTR	Pointer declaration has more than two levels of indirection	17.5
MISRA.RETURN.NOT_LAST	Return is not the last statement in a function	14.7
MISRA.SHIFT.RANGE	Right operand of shift operation is out of range - greater or equal to max bit-length of left operand, or negative	12.8
MISRA.SIGNED_CHAR.NOT_NUMERIC	'signed char' or 'unsigned char' is used for non-numeric value	6.2
MISRA.SIZEOF.SIDE_EFFECT	Operand of sizeof has side effects	12.3
MISRA.STDLIB.ABORT	Use of 'abort', 'exit', 'getenv' or 'system' from library stdlib.h	20.11
MISRA.STDLIB.ATOI	Use of 'atof', 'atoi' or 'atol' from library stdlib.h	20.10
MISRA.STDLIB.ERRNO	Use of error indicator 'errno'	20.5
MISRA.STDLIB.LONGJMP	Use of setjmp macro or longjmp function	20.7
MISRA.STDLIB.MEMORY	Use of dynamic heap memory allocation	20.4
MISRA.STDLIB.SIGNAL	Use of the signal handling facilities of signal.h	20.8
MISRA.STDLIB.STDIO	Use of input/output library stdio.h in production code	20.9
MISRA.STDLIB.TIME	Use of the time handling functions of library time.h	20.12
MISRA.STDLIB.WRONGNAME	Reused name of standard library macro, object or function	20.2
MISRA.STDLIB.WRONGNAME.UNDERSCORE	Usage of a reserved name for naming a language entity	20.2
MISRA.STMT.NO_COMPOUND	The body of switch, while, do/while or for statement is not a compound statement	14.8
MISRA.STMT.NO_EFFECT	The statement has no side effects, and does not change control flow	14.2
MISRA.SWITCH.BOOL	Condition of switch statement is boolean expression	15.4
MISRA.SWITCH.LABEL	A switch label belongs to nested compound statement inside switch body	15.1
MISRA.SWITCH.NODEFAULT	No default clause at the end of a switch statement	15.3
MISRA.SWITCH.NO_BREAK	No break or throw statement at the end of switch-	15.2

	clause	
MISRA.SWITCH.NO_CASE	No case-clause in a switch statement	15.5
MISRA.TOKEN.BADCOM	Inappropriate character sequence in a comment	2.3
MISRA.TOKEN.CPCOM	C++ style comments	2.2
MISRA.TOKEN.OCTAL.ESCAPE	Usage of octal escape sequences	7.1
MISRA.TOKEN.OCTAL.INT	Usage of octal integer constants	7.1
MISRA.TOKEN.WRONGESC	Incorrect escape sequence in a literal	4.1
MISRA.TYPE.NAMECLASH.C.2004	Identifier in one name space has same spelling as identifier in other name space	5.6
MISRA.TYPEDEF.NOT_UNIQUE	Typedef name is used for another entity	5.3
MISRA.UMINUS.UNSIGNED	Operand of unary minus is unsigned	12.9
MISRA.UNDEF	Undef usage	19.6
MISRA.UNDEF.NOTGLOBAL	Undef not at the global level	19.5
MISRA.UNDEF.WRONGNAME	Undefinition of a name from the standard library	20.2
MISRA.UNDEF.WRONGNAME.UNDERSCORE	Undefinition of a reserved name	20.2
MISRA.UNION	Union is used	18.4
MISRA.VAR.HIDDEN	Identifier declared in an inner scope hides identifier in outer scope	5.2
MISRA.VAR.MIN.VIS	Name visibility is too wide	8.7
MISRA.VAR.UNIQUE	Identifier clashes with other identifier	5.7
MISRA.VAR.UNIQUE.STATIC	Identifier with static storage specifier clashes with other identifier	5.5
MISRA.ZERO_EQ.IMPLICIT	Non-boolean expression is implicitly tested against zero	13.2
NUM.OVERFLOW.DF	Possible numeric overflow or wraparound	12.11
PORTING.VAR.EFFECTS	Variable used twice in one expression where one usage is subject to side-effects	12.2
SV.RVT.RETVAL_NOTTESTED	Ignored return value	16.10
UNINIT.HEAP.MIGHT	Uninitialized Heap Use - possible	9.1
UNINIT.HEAP.MUST	Uninitialized Heap Use	9.1
UNINIT.STACK.ARRAY.MIGHT	Uninitialized Array - possible	9.1
UNINIT.STACK.ARRAY.MUST	Uninitialized Array	9.1
UNINIT.STACK.ARRAY.PARTIAL.MUST	Partially Uninitialized Array	9.1
UNINIT.STACK.MIGHT	Uninitialized Variable - possible	9.1
UNINIT.STACK.MUST	Uninitialized Variable	9.1
UNREACH.ENUM	Unreachable code caused by enumeration	14.1
UNREACH.GEN	Unreachable code	14.1
UNREACH.RETURN	Unreachable Void Return	14.1

MISRA C++:2008 suite of checkers

CHECKER	DESCRIPTION	MISRA-C++ RULE
CWARN.NOEFFECT.UCMP.GE	Comparison of unsigned value against 0 is always true	0-1-2
CWARN.NOEFFECT.UCMP.GE.MACRO	Comparison of unsigned value against 0 within a macro is always true	0-1-2
CWARN.NOEFFECT.UCMP.LT	Comparison of unsigned value against 0 is always false	0-1-2
CWARN.NOEFFECT.UCMP.LT.MACRO	Comparison of unsigned value against 0 within a macro is always false	0-1-2
FUNCRET.GEN	Non-void function does not return value	8-4-3
LOCRET.ARG	Function returns address of local variable	7-5-2
LOCRET.GLOB	Function returns address of local variable	7-5-2
LOCRET.RET	Function returns address of local variable	7-5-1
LV_UNUSED.GEN	Local variable unused	0-1-3
MISRA.ADDR.REF.PARAM	Function returns reference to parameter passed by reference	7-5-3
MISRA.ADDR.REF.PARAM.PTR	Function returns address of parameter passed by reference	7-5-3
MISRA.ASM.ENCAPS	Assembly language is not isolated.	7-4-3
MISRA.ASSIGN.COND	Assignment operator is used in a condition	6-2-1
MISRA.ASSIGN.OVERLAP	Object is assigned to an overlapping object	0-2-1
MISRA.ASSIGN.SUBEXPR	Assignment operator is used in a sub-expression outside a condition	6-2-1
MISRA.BASE.IDS.UNIQUE	Member name is used twice in inheritance hierarchy	10-2-1
MISRA.BASE.MANYDEFS	Both overriding and overridden virtual functions have definitions	10-3-1
MISRA.BASE.VIRTUAL.NOTVIRTUAL	Base class is used as both virtual and not virtual in inheritance hierarchy	10-1-3
MISRA.BIN_OP.OVERLOAD	Comma, || or && operator overloaded	5-2-11
MISRA.BITFIELD.SIGNED	Length of a named signed bit-field is less than 2	9-6-4
MISRA.BITFIELD.TYPE.CPP	Type of bit-field is neither bool, nor signed/unsigned integer	9-6-2
MISRA.BITS.NOT_UNSIGNED	Operand of bitwise operation is not unsigned integer	5-0-21
MISRA.BITS.NOT_UNSIGNED.PREP	Operand of bitwise operation in preprocessor directive #if or #elif is not unsigned integer	5-0-21
MISRA.BITS.OPERAND	Operands of bitwise operation have different underlying types	5-0-20
MISRA.BUILTIN_NUMERIC	Builtin numeric type is used	3-9-2
MISRA.CAST.CONST	Cast operation removes const or volatile modifier from a pointer or reference	5-2-5
MISRA.CAST.FLOAT.WIDER	Cast of floating point expression to a wider floating point type	5-0-8
MISRA.CAST.FLOAT_INT	Cast of floating point expression to integral type	5-0-7
MISRA.CAST.FUNC_PTR.CPP	Cast converts function pointer to other pointer type	5-2-6
MISRA.CAST.INT.SIGN	Non-trivial integral expression is cast to type with different signedness	5-0-9
MISRA.CAST.INT.WIDER	Cast of integral expression to a wider integral type	5-0-8
MISRA.CAST.INT_FLOAT	Cast of integral expression to floating point type	5-0-7
MISRA.CAST.INT_TO_PTR	Object with integer type or pointer to void cast to pointer type	5-2-8
MISRA.CAST.POLY.TYPE	Cast from a polymorphic base class to a derived class	5-2-3
MISRA.CAST.PTR.UNRELATED	Object of pointer type cast to unrelated type	5-2-7
MISRA.CAST.PTR.VRCLASS	A cast form pointer to a virtual base class to pointer to a derived class does not use 'dynamic_cast'	5-2-2
MISRA.CAST.PTR_TO_INT	Cast between a pointer and an integral type	5-2-9
MISRA.CAST.UNSIGNED_BITS	The result of bitwise operation on unsigned char or short is not cast back to original type	5-0-10
MISRA.CATCH.ALL	No ellipsis exception handler in a try-catch block	15-3-2
MISRA.CATCH.BY_VALUE	Exception object of class type is caught by value	15-3-5
MISRA.CATCH.NOALL	Ellipsis exception handler is not the last one in a try-catch block	15-3-7
MISRA.CATCH.WRONGORD	Handler for a base exception class precedes to a handler for a derived exception class in a try-catch block	15-3-6
MISRA.CHAR.DIGRAPH	Digraph usage	2-5-1
MISRA.CHAR.NOT_CHARACTER	'char' is used for non-character value	5-0-11
MISRA.CHAR.OPERAND	Expression of type 'char' or 'wchar_t' is used as non-character operand	4-5-3
MISRA.CHAR.TRIGRAPH	Trigraph usage	2-3-1
MISRA.COMMA	Comma operator is used	5-18-1
MISRA.COMP.WRAPAROUND	Wrap-around in a condition	5-19-1

MISRA.CONST.RET.NON_CONST	Constant member function returns non-const pointer to member variable	9-3-1
MISRA.CONTINUE.ILL	Continue statement is used in an ill-formed for loop	6-6-3
MISRA.CONV.FLOAT	Implicit floating-point conversion	5-0-5
MISRA.CONV.INT.SIGN	Implicit integral conversion changes signedness	5-0-4
MISRA.CONV.NUM.NARROWER	Implicit numeric conversion to narrower type	5-0-6
MISRA.COPY.CSTR.TMPL	Class has a template constructor with a single generic parameter, but has no copy constructor defined	14-5-2
MISRA.COPYASSIGN.ABSTRACT	Copy assignment should be declared protected or private in an abstract class	12-8-2
MISRA.COPYASSIGN.TMPL	A copy assignment operator should be defined when class has a template copy assignment operator with a single generic parameter	14-5-3
MISRA.CT.UNIQUE.ID	Identifier clashes with type name	2-10-4
MISRA.CTOR.BASE	Constructor does not explicitly call constructor of its base class	12-1-2
MISRA.CTOR.DYNAMIC	Object's dynamic type is used from the body of its constructor	12-1-1
MISRA.CTOR.NOT_EXPLICIT	Constructor with one argument of built-in type is not declared 'explicit'	12-1-3
MISRA.CTOR.TRY.NON_STATIC	Function try/catch block of constructor or destructor references non-static members	15-3-3
MISRA.CVALUE.IMPL.CAST.CPP	The value of an expression implicitly converted to a different type	5-0-3
MISRA.C_CAST	C-style cast to non-void type	5-2-4
MISRA.DECL.ARRAY_SIZE	Declaration of array with unknown size	3-1-3
MISRA.DECL.EXCPT.SPEC	Function is declared with different exception specifications	15-4-1
MISRA.DECL.FUNC_LOCAL	Function is declared locally	3-1-2
MISRA.DECL.MANY_DCLS	More than one declarator in one declaration	8-0-1
MISRA.DEFINE.BADEXP.CPP	Inappropriate macro expansion in a C++ source	16-2-2
MISRA.DEFINE.FUNC	Function-like macro definition	16-0-4
MISRA.DEFINE.NOPARS	Macro parameter with no parentheses	16-0-6
MISRA.DEFINE.NOTGLOBAL	Define not at the global level	16-0-2
MISRA.DEFINE.SHARP	or ## operator in a macro definition	16-3-2
MISRA.DEFINE.SHARP.MANY	Several # or ## operators in a macro definition	16-3-1
MISRA.DEFINE.WRONGNAME	Usage of a name from the standard library for naming a macro	17-0-1
MISRA.DEFINE.WRONGNAME.UNDERSCORE	Usage of a reserved name for naming a macro	17-0-1
MISRA.DERIVE.VIRTUAL	Class is derived from virtual base	10-1-1
MISRA.DTOR.DYNAMIC	Object's dynamic type is used from the body of its destructor	12-1-1
MISRA.DTOR.THROW	Throw in destructor	15-5-1
MISRA.ELIF.DEFINED	Incorrect 'defined' usage in #elif directive	16-1-1
MISRA.ELIF.OTHERFILE	#elif in an improper file	16-1-2
MISRA.ELIF.UNDEF	Undefined macros in #elif directive	16-0-7
MISRA.ELIF.WRAPAROUND	Wrap-around in #elif directive	5-19-1
MISRA.ELSE.OTHERFILE	#else in an improper file	16-1-2
MISRA.ENDIF.OTHERFILE	#endif in an improper file	16-1-2
MISRA.ENUM.INIT	Non-first enumerator is explicitly initialized, but not all elements are explicitly initialized.	8-5-3
MISRA.ENUM.OPERAND	Expression of enum type is used in arithmetic context	4-5-2
MISRA.EXPANSION.DIRECTIVE	Directive-like tokens within a macro argument	16-0-5
MISRA.EXPANSION.UNSAFE	Unsafe macro usage	17-0-5, 18-0-1, 18-2-1, 18-7-1, 19-3-1
MISRA.EXPR.COND.NOT_BOOLEAN	First operand of conditional expression is not a boolean expression	5-0-14
MISRA.EXPR.PARENS.INSUFFICIENT	Limited dependence required for operator precedence rules in expressions	5-0-2
MISRA.EXPR.PARENS.REDUNDANT	Limited dependence required for operator precedence rules in expressions	5-0-2
MISRA.FIELD.BIT.ENUM	Bit-field has enum type.	9-6-3
MISRA.FLOAT.BIT.REPR	Use of bit manipulations of floating-point values which rely on storage layout	3-9-3
MISRA.FLOAT_EQUAL	Floating point expression is tested for equality	6-2-2
MISRA.FOR.COND.CHANGE	For loop counter is modified within the loop condition section	6-5-3
MISRA.FOR.COND.EQ	++ or – operations are not used to change loop counter, but condition tests loop counter for equality	6-5-2
MISRA.FOR.COUNTER.FLT	For loop counter has a floating point type	6-5-1

MISRA.FOR.COUNTER.MANY	Many counters in a for loop	6-5-1
MISRA.FOR.INCR	For loop counter is modified in an inappropriate way	6-5-4
MISRA.FOR.LOOP_CONTROL.CHANGE.COND	Loop control variable is modified in condition section of a for loop	6-5-5
MISRA.FOR.LOOP_CONTROL.CHANGE.EXPR	Loop control variable is modified in expression section of a for loop	6-5-5
MISRA.FOR.LOOP_CONTROL.NOT_BOOLEAN	Loop control variable is not boolean	6-5-6
MISRA.FOR.STMT.CHANGE	For loop counter is modified within the loop statement	6-5-3
MISRA.FUNC.ADDR	Address of a function is used without & operator	8-4-4
MISRA.FUNC.ARRAY.PARAMS	Function argument with array type decay to a pointer	5-2-12
MISRA.FUNC.DECL.AFTERUSE	Function chosen by overload resolution when instantiating a template is declared after its usage	14-6-2
MISRA.FUNC.PARAMS.IDENT	Identifiers used in declaration and definition of function are no identical	8-4-2
MISRA.FUNC.RECUR	Recursive function	7-5-4
MISRA.FUNC.SPEC.NOTSPEC	Viable function set for a function call contains both specializations and non-specializations	14-8-2
MISRA.FUNC.SPEC.OVRLD	Viable function set for a function call contains an overloaded template and its explicit specialization	14-8-1
MISRA.FUNC.STATIC.REDECL	Function or object redeclaration does not include 'static' modifier	3-3-2
MISRA.FUNC.UNUSEDPAR	Formal parameter of a non-virtual function is not used	0-1-11
MISRA.FUNC.UNUSEDPAR.UNNAMED	Unnamed formal parameter of a non-virtual function is not used	0-1-11
MISRA.FUNC.UNUSEDRET	Return value of a non-void function is not used	0-1-7
MISRA.FUNC.VARARG	Function with variable number of arguments	8-4-1
MISRA.FUNC.VIRTUAL.UNUSEDPAR	Formal parameter of a virtual function set is not used	0-1-12
MISRA.FUNC_CAST	Functional notation cast different from explicit constructor call	5-2-4
MISRA.GENFU.ASSOC	Generic function is declared in an associated namespace	14-5-1
MISRA.GOTO.AFTER.LABEL	Unconstrained use of goto	6-6-2
MISRA.GOTO.NESTED	Goto to a label declared in a nested compound statement	6-6-1
MISRA.IF.DEFINED	Incorrect 'defined' usage in #if directive	16-1-1
MISRA.IF.NO_COMPOUND	The body of if/else statement is not a compound statement	6-4-1
MISRA.IF.NO_ELSE	A chain of if/else-if statements is not terminated with else or is terminated with an empty else clause	6-4-2
MISRA.IF.UNDEF	Undefined macros in #if directive	16-0-7
MISRA.IF.WRAPAROUND	Wrap-around in #if directive	5-19-1
MISRA.INCGUARD	Include guard is not provided	16-2-3
MISRA.INCL.BAD	Non-standard include directive	16-2-6
MISRA.INCL.INSIDE	Include directive preceded by a preprocessor output token	16-0-1
MISRA.INCL.SYMS	Non-standard characters in header file names	16-2-4, 16-2-5
MISRA.INCL.UNSAFE	Unsafe header inclusion	18-0-2, 18-0-4, 18-7-1, 27-0-1
MISRA.INCR_DECR.OTHER	Increment or decrement operator is mixed with other operators in expression	5-2-10
MISRA.INIT.BRACES	Incorrect initializer braces placement.	8-5-2
MISRA.ITER.ONETERM	Iteration statement has more than one break or goto for loop termination.	6-6-4
MISRA.LINKAGE.EXTERN	Object or function declaration with external linkage not in header file	3-3-1
MISRA.LITERAL.NULL.INT	NULL used as an integer value.	4-10-1
MISRA.LITERAL.NULL.PTR	Literal zero used as the null-pointer-constant.	4-10-2
MISRA.LITERAL.SUFFIX.CASE	Literal suffix in lower case.	2-13-4
MISRA.LITERAL.UNSIGNED.SUFFIX	Unsigned integer literal without 'U' suffix	2-13-3
MISRA.LOGIC.NOT_BOOL	Operand of logical operation is not boolean	5-3-1
MISRA.LOGIC.OPERATOR.NOT_BOOL	Operand of non-logical operator is effectively boolean	4-5-1
MISRA.LOGIC.POSTFIX	Operand in a logical 'and' or 'or' expression is not a postfix expression	5-2-1
MISRA.LOGIC.SIDEEFF	Right operand in a logical 'and' or 'or' expression contains side effects	5-14-1
MISRA.MEMB.NON_CONST	Non-const member function does not change any member variables	9-3-3
MISRA.MEMB.NON_STATIC	Non-static member function does not use other non-static members of the same class	9-3-3
MISRA.MEMB.NOT_PRIVATE	Member variable in non-POD class is not private	11-0-1
MISRA.NAMESPACE.DECL	Using-declaration in header file	7-3-6

MISRA.NAMESPACE.DIR	Using-directive in header file	7-3-6
MISRA.NAMESPACE.UNMD	Unnamed namespace in header file	7-3-3
MISRA.NS.GLOBAL	Function, variable or type declaration in global namespace	7-3-1
MISRA.NS.GLOBAL.USING	Using directive or declaration in global namespace	7-3-1
MISRA.NS.MAIN	Non-global function with name 'main' is defined	7-3-2
MISRA.NS.USING.HEADER	Using directive or declaration is used in a header file	7-3-6
MISRA.NS.USING_DECL	Multiple declarations for an identifier in the same namespace should not straddle a using-declaration for that identifier	7-3-5
MISRA.NS.USING_DIR	Using directive	7-3-4
MISRA.NULL.STMT	Null statement is not the only statement on line or comments are placed incorrectly	6-2-3
MISRA.OBJ.TYPE.COMPAT	Type not compatible with type of other declaration	3-2-1
MISRA.OBJ.TYPE.IDENT	Type not identical with type of other declaration	3-9-1
MISRA.ONEDEFRULE.FUNC	Global function definition in a header file	3-1-1
MISRA.ONEDEFRULE.VAR	Global variable definition in a header file	3-1-1
MISRA.PPARAM.NEEDS.CONST	Pointer parameter is not used to modify the addressed object but is not declared as a pointer to const	7-1-2
MISRA.PRAGMA	Non-documented pragma directive	16-6-1
MISRA.PRAGMA.ASM	Incorrect assembler instruction	7-4-2
MISRA.PTR.ARITH	Pointer is used in arithmetic or array index expression	5-0-15
MISRA.PTR.ARITH.NOT_SAME.2008	A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand	5-0-16
MISRA.PTR.CMP.2008	Pointer comparison using comparison operators shall only be applied if pointing to same array and within the range	5-0-18
MISRA.PTR.CMP.OBJECT.2008	Pointer comparison using comparison operators shall only be applied if pointing to same object and within the range	5-0-18
MISRA.PTR.SUB	Pointer subtraction shall only be applied if pointing to same array	5-0-17
MISRA.PTR.SUB.OBJECT	Pointer subtraction shall only be applied if pointing to same object	5-0-17
MISRA.PTR.TO_PTR_TO_PTR	Pointer declaration has more than two levels of indirection	5-0-19
MISRA.PUREVIRT.OVRD	Pure virtual function overrides a non pure virtual function	10-3-3
MISRA.RET.REF.NON_CONST	Member function returns non-const handle to member variable	9-3-2
MISRA.RETURN.NOT_LAST	Return is not the last statement in a function	6-6-5
MISRA.SAME.DEFPARAMS	Overriding virtual function and the function it overrides have different default arguments	8-3-1
MISRA.SHIFT.RANGE	Right operand of shift operation is out of range - greater or equal to max bit-length of left operand, or negative	5-8-1
MISRA.SIGNED_CHAR.NOT_NUMERIC	'signed char' or 'unsigned char' is used for non-numeric value	5-0-12
MISRA.SIZEOF.SIDE_EFFECT	Operand of sizeof has side effects	5-3-4
MISRA.SPEC.ILL	Explicit instantiation of a template makes the code ill-formed	14-7-2
MISRA.SPEC.SAMEFILE	Template specialization and its primary template are declared in different files	14-7-3
MISRA.STDLIB.ABORT	Use of 'abort', 'exit', 'getenv' or 'system' from library stdlib.h	18-0-3
MISRA.STDLIB.ATOI	Use of 'atof', 'atoi' or 'atol' from library stdlib.h	18-0-2
MISRA.STDLIB.CSTRING	Function from 'cstring' library is used	18-0-5
MISRA.STDLIB.CSTRING.MACRO	Macro from 'cstring' library is used	18-0-5
MISRA.STDLIB.ERRNO	Use of error indicator 'errno'	19-3-1
MISRA.STDLIB.LONGJMP	Use of setjmp macro or longjmp function	17-0-5
MISRA.STDLIB.MEMORY	Use of dynamic heap memory allocation	18-4-1
MISRA.STDLIB.SIGNAL	Use of the signal handling facilities of signal.h	18-7-1
MISRA.STDLIB.STDIO	Use of input/output library stdio.h in production code	27-0-1
MISRA.STDLIB.TIME	Use of the time handling functions of library time.h	18-0-4
MISRA.STDLIB.WRONGNAME	Reused name of standard library macro, object or function	17-0-3
MISRA.STDLIB.WRONGNAME.UNDERSCORE	Usage of a reserved name for naming a language entity	17-0-3
MISRA.STMT.COND.NOT_BOOLEAN	Condition of if or loop statement is not a boolean expression	5-0-13
MISRA.STMT.NO_COMPOUND	The body of switch, while, do/while or for statement is not a compound statement	6-3-1
MISRA.STRINGS.CONCAT	Narrow and wide string literals concatenated	2-13-5
MISRA.SWITCH.BOOL	Condition of switch statement is boolean expression	6-4-7
MISRA.SWITCH.LABEL	A switch label belongs to nested compound statement inside switch body	6-4-4
MISRA.SWITCH.NODEFAULT	No default clause at the end of a switch statement	6-4-6
MISRA.SWITCH.NOT_WELL_FORMED	Switch statement is not well-formed	6-4-3

MISRA.SWITCH.NO_BREAK	No break or throw statement at the end of switch-clause	6-4-5
MISRA.SWITCH.NO_CASE	No case-clause in a switch statement	6-4-8
MISRA.TEMPLMEM.NOQUAL	In an instantiated template a member declared in a dependant base is used without a qualificator or 'this'	14-6-1
MISRA.TERMINATE	terminate() function is called explicitly	15-5-3
MISRA.THROW.EMPTY	Empty throw expression does not belong to a catch block	15-1-3
MISRA.THROW.NULL	NULL is thrown explicitly	15-1-2
MISRA.THROW.PTR	Exception object is a pointer	15-0-2
MISRA.TOKEN.BADCOM	Inappropriate character sequence in a comment	2-7-1
MISRA.TOKEN.OCTAL.ESCAPE	Usage of octal escape sequences	2-13-2
MISRA.TOKEN.OCTAL.INT	Usage of octal integer constants	2-13-2
MISRA.TOKEN.WRONGESC	Incorrect escape sequence in a literal	2-13-1
MISRA.TRY.JUMP	Control can be transferred into a try block with goto or switch statement	15-0-3
MISRA.TYPE.NAMECLASH.CPP.2008	Identifier in one name space has same spelling as identifier in other name space	2-10-6
MISRA.TYPEDEF.NOT_UNIQUE	Typedef name is used for another entity	2-10-3
MISRA.UMINUS.UNSIGNED	Operand of unary minus is unsigned	5-3-2
MISRA.UNDEF	Undef usage	16-0-3
MISRA.UNDEF.NOTGLOBAL	Undef not at the global level	16-0-2
MISRA.UNDEF.WRONGNAME	Undefinition of a name from the standard library	17-0-2
MISRA.UNDEF.WRONGNAME.UNDERSCORE	Undefinition of a reserved name	17-0-2
MISRA.UNION	Union is used	9-5-1
MISRA.UN_OP.OVERLOAD	Unary & operator is overloaded	5-3-3
MISRA.USE.DEFINE	Non-guarding macro definition	16-2-1
MISRA.USE.EXPANSION	Macro expansion	16-2-1
MISRA.USE.UNKNOWNDIR	Unknown preprocessor directive is used	16-0-8
MISRA.USE.WRONGDIR	Improper preprocessor directive	16-2-1
MISRA.VAR.HIDDEN	Identifier declared in an inner scope hides identifier in outer scope	2-10-2
MISRA.VAR.MIN.VIS	Name visibility is too wide	3-4-1
MISRA.VAR.NEEDS.CONST	Variable is not modified but is declared without const qualifier	7-1-1
MISRA.VAR.UNIQUE.STATIC	Identifier with static storage specifier clashes with other identifier	2-10-5
MISRA.VIRTUAL.BASE.DIAMOND	Base class is used as virtual not in diamond hierarchy	10-1-2
MISRA.VIRTUAL.NOVIRTUAL	Overriding virtual function declared with no 'virtual' keyword	10-3-2
NUM.OVERFLOW.DF	Possible numeric overflow or wraparound	5-19-1
PORTING.VAR.EFFECTS	Variable used twice in one expression where one usage is subject to side-effects	5-0-1
UNINIT.CTOR.MIGHT	Uninitialized Variable in Constructor - possible	8-5-1
UNINIT.CTOR.MUST	Uninitialized Variable in Constructor	8-5-1
UNINIT.HEAP.MIGHT	Uninitialized Heap Use - possible	8-5-1
UNINIT.HEAP.MUST	Uninitialized Heap Use	8-5-1
UNINIT.STACK.ARRAY.MIGHT	Uninitialized Array - possible	8-5-1
UNINIT.STACK.ARRAY.MUST	Uninitialized Array	8-5-1
UNINIT.STACK.ARRAY.PARTIAL.MUST	Partially Uninitialized Array	8-5-1
UNINIT.STACK.MIGHT	Uninitialized Variable - possible	8-5-1
UNINIT.STACK.MUST	Uninitialized Variable	8-5-1
UNREACH.ENUM	Unreachable code caused by enumeration	0-1-1
UNREACH.GEN	Unreachable code	0-1-1
UNREACH.RETURN	Unreachable Void Return	0-1-1
UNUSED.FUNC.GEN	Function defined but not used	0-1-10
VA_UNUSED.GEN	Value is Never Used after Assignment	0-1-9
VA_UNUSED.INIT	Value is Never Used after Initialization	0-1-9

MISRA C:2012 suite of checkers

CHECKER	DESCRIPTION	MISRA-C RULE
ABV.ANY_SIZE_ARRAY	Buffer Overflow - Array Index Out of Bounds	21.17 21.18
ABV.GENERAL	Buffer Overflow - Array Index Out of Bounds	21.17 21.18
ABV.GENERAL. MULTIDIMENSION	Buffer Overflow – Multi-Dimensional Array Index Out of Bounds	21.17 21.18
ABV.MEMBER	Buffer Overflow - Array Index Out of Bounds	21.17 21.18
ABV.STACK	Buffer Overflow - Local Array Index Out of Bounds	21.17 21.18
ABV.TAINTED	Buffer Overflow from Unvalidated Input	Dir 4.14
ABV.UNICODE.BOUND_MAP	Buffer overflow-array index out of bounds in mapping function	Dir 4.1
ABV.UNICODE.FAILED_MAP	Buffer overflow-array index out of bounds in failed mapping function	Dir 4.1
ABV.UNICODE.NNTS_MAP	Buffer overflow from non null-terminated string in mapping function	Dir 4.1
ABV.UNICODE.SELF_MAP	Buffer overflow-array index out of bounds in failed mapping function	Dir 4.1
ABV.UNKNOWN_SIZE	Buffer Overflow - Array Index Out of Bounds	21.17 21.18
CXX.ERRNO.INCORRECTLY_CHECKED	Errno was incorrectly checked	22.10
CXX.ERRNO.NOT_SET	Errno was not set	22.8
CXX.ERRNO.NOT_CHECKED	Errno was not checked	22.9
DBZ.CONST	Zero constant value is used directly as a divisor in a division or modulo operation	Dir 4.1
DBZ.CONST.CALL	Zero constant value is passed to a function and might be used in a division by zero	Dir 4.1
DBZ.GENERAL	Assigned zero constant value might be used in a division by zero	Dir 4.1
DBZ.ITERATOR	Zero constant value is used directly as a divisor in a division or modulo operation	Dir 4.1
DBZ.ITERATOR.CALL	Division by zero might occur in a function call	Dir 4.1
EFFECT	Statement has no effect	2.2
FMM.MIGHT	Freeing Mismatched Memory - possible	22.2
FMM.MUST	Freeing Mismatched Memory	22.2
FNH.MIGHT	Freeing Non-Heap Memory - possible	22.2
FNH.MUST	Freeing Non-Heap Memory	22.2
FREE.INCONSISTENT	Inconsistent Freeing of Memory	22.1
FUM.GEN.MIGHT	Freeing Unallocated Memory - possible	22.2
FUM.GEN.MUST	Freeing Unallocated Memory	22.2
FUNCRET.GEN	Non-void function does not return value	17.4
FUNCRET.IMPLICIT	Non-void function implicitly returning int does not return value	17.4
INVARIANT_CONDITION.GEN	Invariant expression in a condition	14.3
INVARIANT_CONDITION.UNREACH	Invariant expression in a condition	14.3
LA_UNUSED	Label unused	2.6
LOCRET.ARG	Function returns address of local variable	18.6
LOCRET.GLOB	Function returns address of local variable	18.6
LOCRET.RET	Function returns address of local variable	18.6
LV_UNUSED.GEN	Local variable unused	2.2
MISRA.ARRAY.VAR_LENGTH.2012	Variable-length array types shall not be used	18.8
MISRA.ASM.ENCAPS	Assembly language is not isolated.	4.3
MISRA.ASSIGN.OVERLAP	Object is assigned to an overlapping object	19.1
MISRA.ASSIGN.SUBEXPR.2012	The result of an assignment operator should not be used	13.4
MISRA.BITFIELD.SIGNED	Length of a named signed bit-field is less than 2	6.2
MISRA.BITFIELD.TYPE	Type of bit-field is not signed/unsigned integer	6.1
MISRA.BITFIELD.TYPE.2012	Type of bit-field is not signed/unsigned integer	6.1

MISRA.BREAK_OR_GOTO.MULTIPLE.2012	Iteration statement has more than one break or goto for loop termination.	15.4
MISRA.BUILTIN_NUMERIC	Builtin numeric type is used	4.6
MISRA.CAST.CONST	Cast operation removes const or volatile modifier from a pointer or reference	11.8
MISRA.CAST.FUNC_PTR.2012	Conversion performed between a pointer to a function and another incompatible type	11.1
MISRA.CAST.INCOMPLETE_PTR_TO_ANY.2012	Conversion performed between a pointer to an incomplete type and a different type	11.2
MISRA.CAST.OBJ_PTR_TO_INT.2012	Conversion performed between a pointer to an object and an integer type	11.4
MISRA.CAST.OBJ_PTR_TO_NON_INT.2012	A cast between a pointer to object and a non-integer arithmetic type	11.7
MISRA.CAST.OBJ_PTR_TO_OBJ_PTR.2012	Cast between a pointer to object type and a pointer to a different object type	11.3
MISRA.CAST.VOID_PTR_TO_INT.2012	Cast between a pointer to void and an arithmetic type	11.6
MISRA.CAST.VOID_PTR_TO_OBJ_PTR.2012	Conversion performed from a pointer to void to a pointer to an object	11.5
MISRA.CHAR.TRIGRAPH	Trigraph usage	4.2
MISRA.COMMA	Comma operator is used	12.3
MISRA.COMP.WRAPAROUND	Wrap-around in a condition	12.4
MISRA.CT.UNIQUE.ID.2012	Identifier clashes with type name	5.7
MISRA.DECL.ARRAY_SIZE	Declaration of array with unknown size	8.11
MISRA.DECL.FUNC.INLINE.STATIC.2012	Declaration of inline function without static storage class	8.10
MISRA.DECL.NO_TYPE	Declaration without a type	8.1
MISRA.DEFINE.FUNC	Function-like macro definition	4.9
MISRA.DEFINE.NOT_DISTINCT.C90.2012	Implements MISRA C 2012 Rule 5.4: Macro identifiers shall be distinct	5.4
MISRA.DEFINE.NOT_DISTINCT.C99.2012	Implements MISRA C 2012 Rule 5.4: Macro identifiers shall be distinct	5.4
MISRA.DEFINE.SHARP	or ## operator in a macro definition	20.10
MISRA.DEFINE.SHARP.ORDER.2012	A macro parameter immediately following a # operator shall not immediately be followed by a ## operator.	20.11
MISRA.DEFINE.SHARP.REPLACE.2012	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	20.12
MISRA.DEFINE.STDIO.WCHAR.2012	The Standard Library input/output functions shall not be used	21.6
MISRA.DEFINE.WCSFTIME.2012	The Standard Library time and date functions shall not be used	21.10
MISRA.DEFINE.WRONGNAME	Usage of a name from the standard library for naming a macro	21.1
MISRA.DEFINE.WRONGNAME.C90.2012	Implement MISRA C 2012 Rule 20.4: A macro shall not be defined with the same name as a keyword.	20.4
MISRA.DEFINE.WRONGNAME.C99.2012	Implement MISRA C 2012 Rule 20.4: A macro shall not be defined with the same name as a keyword.	20.4
MISRA.DEFINE.WRONGNAME.UNDERSCORE	Usage of a reserved name for naming a macro	21.1
MISRA.ELIF.COND.NOT_BOOL.2012	Implements MISRA C 2012 Rule 20.8: The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1.	20.8
MISRA.ELIF.OTHERFILE	#elif in an improper file	20.14
MISRA.ELIF.UNDEF	Undefined macros in #elif directive	20.9
MISRA.ELSE.OTHERFILE	#else in an improper file	20.14
MISRA.ENDIF.OTHERFILE	#endif in an improper file	20.14
MISRA.ENUM.IMPLICIT.VAL.NON_UNIQUE.2012	Implicit enumerator value is not unique	8.12
MISRA.ETYPE.ASSIGN.2012	Assignment to an object of a narrower essential type or a different essential type category	10.3
MISRA.ETYPE.CATEGORY.DIFFERENT.2012	The operands of an operator in which the usual arithmetic conversions are performed do not have the same essential type category	10.4
MISRA.ETYPE.COMP.ASSIGN.2012	The value of a composite expression is assigned to an object with a wider essential type	10.6

MISRA.ETYPE.COMP.CAST.EXPL.DIFFERENT.2012	The value of a composite expression is cast to a different essential type category	10.8
MISRA.ETYPE.COMP.CAST.EXPL.WIDER.2012	The value of a composite expression is cast to a wider essential type	10.8
MISRA.ETYPE.COMP.CAST.IMPL.WIDER.2012	A composite expression used as an operand of an operator in which the usual arithmetic conversions are performed has its other operand having a wider essential type	10.7
MISRA.ETYPE.INAPPR.CAST.2012	The value of an expression is cast to an inappropriate essential type	10.5
MISRA.ETYPE.INAPPR.CHAR.2012	Inappropriate usage of Essentially Character type in an addition or subtraction operation	10.2
MISRA.ETYPE.INAPPR.OPERAND.BINOP.2012	Operand(s) to a binary operator have inappropriate essential type	10.1
MISRA.ETYPE.INAPPR.OPERAND.INDEXPR.2012	Index expression has inappropriate essential type	10.1
MISRA.ETYPE.INAPPR.OPERAND.TERNOP.2012	First operand to a ternary operator has inappropriate essential type	10.1
MISRA.ETYPE.INAPPR.OPERAND.UNOP.2012	Operand to a unary operator has inappropriate essential type	10.1
MISRA.EXPANSION.DIRECTIVE	Directive-like tokens within a macro argument	20.6
MISRA.EXPR.PARENS.2012	The precedence of operators within expressions should be made explicit.	12.1
MISRA.EXPR.PARENS.SIZEOF.2012	The operand of the sizeof operator should be parenthesized.	12.1
MISRA.EXPR.SIZEOF.ARRAY_PARAM.2012_AMD1	Implements MISRA C 2012 Rule 12.5: The sizeof operator shall not have an operand which is a function parameter declared as "array of type"	12.5
MISRA.FILE_PTR.DEREF.2012	A pointer to a FILE object shall not be dereferenced	22.5
MISRA.FILE_PTR.DEREF.CAST.2012	An object cast to a FILE pointer shall not be dereferenced	22.5
MISRA.FILE_PTR.DEREF.INDIRECT.2012	A pointer to a FILE object shall not be indirectly dereferenced by a system function	22.5
MISRA.FILE_PTR.DEREF.RETURN.2012	A pointer to a FILE object (returned by a function) shall not be dereferenced	22.5
MISRA.FOR.COUNTER.FLT	For loop counter has a floating point type	14.1
MISRA.FUNC.ARRAY.PARAM.STATIC.2012	Implements MISRA C 2012 Rule 17.6: The declaration of an array parameter shall not contain the static keyword between the [ ]	17.6
MISRA.FUNC.MODIFIEDPAR.2012	Implements MISRA C 2012 Rule 17.8: A function parameter should not be modified.	17.8
MISRA.FUNC.NODECL.CALL.2012	Implements MISRA C 2012 Rule 17.3: A function shall not be declared implicitly.	17.3
MISRA.FUNC.NOPROT.DEF.2012	Implements MISRA C 2012 Rule 8.4: A compatible declaration shall be visible when an object or function with external linkage is defined	8.4
MISRA.FUNC.NO_PARAMS	Function without parameters is missing void parameter type	8.2
MISRA.FUNC.PROT_FORM.KR.2012	Function types shall be in prototype form	8.2
MISRA.FUNC.RECUR	Recursive function	17.2
MISRA.FUNC.STATIC.REDECL	Function or object redeclaration does not include 'static' modifier	8.8
MISRA.FUNC.UNMATCHED.PARAMS	Number of formal and actual parameters passed to function do not match	8.2
MISRA.FUNC.UNNAMED.PARAMS	Function declaration has unnamed parameters	8.2
MISRA.FUNC.UNUSEDPAR.2012	There should be no unused parameters in functions	2.7
MISRA.FUNC.UNUSEDRET.2012	The value returned by a function having non-void return type shall be used	17.7
MISRA.FUNC.VARARG	Function with variable number of arguments	17.1
MISRA.GOTO	Goto statement is used	15.1
MISRA.GOTO.AFTER_LABEL.2012	Goto jumps to label declared before in same function.	15.2
MISRA.GOTO.NESTED.2012	Label referenced by goto is not in this or enclosing block.	15.3
MISRA.IDENT.DISTINCT.C90.2012	Identifiers declared in the same scope and name space shall be distinct	5.2
MISRA.IDENT.DISTINCT.C99.2012	Identifiers declared in the same scope and name space shall be distinct	5.2

MISRA.IDENT.NONUNIQUE.EXTERNAL.2012	Identifiers that define objects or functions with external linkage shall be unique.	5.8
MISRA.IDENT.NONUNIQUE.INTERNAL.2012	Identifiers that define objects or functions with internal linkage should be unique.	5.9
MISRA.IF.COND.NOT_BOOL.2012	Implements MISRA C 2012 Rule 20.8: The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1.	20.8
MISRA.IF.NO_COMPOUND	The body of if/else statement is not a compound statement	15.6
MISRA.IF.NO_ELSE	A chain of if/else-if statements is not terminated with else or is terminated with an empty else clause	15.7
MISRA.IF.UNDEF	Undefined macros in #if directive	20.9
MISRA.INCGUARD	Include guard is not provided	4.10
MISRA.INCL.BAD	Non-standard include directive	20.3
MISRA.INCL.INSIDE	Include directive preceded by a preprocessor output token	20.1
MISRA.INCL.LANG.FEATURES.2012	Emergent language feature headers should not be used	1.4
MISRA.INCL.LANG.FEATURES.MT.2012	Emergent multithreading language feature headers should not be used	1.4
MISRA.INCL.SIGNAL.2012	The standard header file<signal.h>shall not be used	21.5
MISRA.INCL.STDIO.2012	The Standard Library input/output functions shall not be used	21.6
MISRA.INCL.SYMS	Non-standard characters in header file names	20.2
MISRA.INCL.TGMATH.2012	The standard header file <tgmath.h> shall not be used	21.11
MISRA.INCL.TIME.2012	The Standard Library time and date functions shall not be used	21.10
MISRA.INCR_DECR.SIDEEFF.2012	Implements MISRA C 2012 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.	13.3
MISRA.INIT.BRACES.2012	The initializer for an aggregate or union is not enclosed in braces	9.2
MISRA.INIT.MULTIPLE.2012	An element of an object is initialized more than once	9.4
MISRA.INIT.PARTIAL.2012	Array is partially initialized	9.3
MISRA.INIT.SIZE.IMPLICIT.2012	A designated initializer is used to initialize an array object when the size of the array is not specified explicitly	9.5
MISRA.LANG.EXTENSIONS	Language extensions should not be used	1.2
MISRA.LANG.FEATURES.2012	Emergent language feature should not be used	1.4
MISRA.LANG.FEATURES.MT.2012	Emergent multithreading language feature should not be used	1.4
MISRA.LITERAL.NULL.PTR.CONST.2012	The macro NULL shall be the only permitted form of integer null pointer constant	11.9
MISRA.LITERAL.UNSIGNED.SUFFIX	Unsigned integer literal without 'U' suffix	7.2
MISRA.LOGIC.SIDEEFF	Right operand in a logical 'and' or 'or' expression contains side effects	13.5
MISRA.MEMB.FLEX_ARRAY.2012	Flexible array members shall not be declared	18.7
MISRA.MEMCMP.NTS.2012_AMD1	The memcmp function shall only be used to compare non null terminated strings	21.14
MISRA.MEMCMP.NTS.GLOBAL.2012_AMD1	The memcmp function shall only be used to compare non null terminated strings	21.14
MISRA.PPARAM.NEEDS.CONST	Pointer parameter is not used to modify the addressed object but is not declared as a pointer to const	8.13
MISRA.PTR.ARITH.2012	Implements MISRA C 2012 Rule 18.4: The +, -, += and -= operators should not be applied to an expression of pointer type.	18.4
MISRA.PTR.ARITH.NOT_SAME.2012	A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand	18.1
MISRA.PTR.CMP.2008	Pointer comparison using comparison operators shall only be applied if pointing to same array and within the range	18.3
MISRA.PTR.CMP.OBJECT.2008	Pointer comparison using comparison operators shall only be applied if pointing to same object and within the range	18.3
MISRA.PTR.SUB	Pointer subtraction shall only be applied if pointing to same array	18.2
MISRA.PTR.SUB.OBJECT	Pointer subtraction shall only be applied if pointing to same	18.2

	object	
MISRA.PTR.TO_PTR_TO_PTR	Pointer declaration has more than two levels of indirection	18.5
MISRA.RESOURCES.FILE.OPEN_READ_WRITE.2012	Multiple streams opened for same file	22.3
MISRA.RESOURCES.FILE.READ_ONLY_WRITE.2012	Implements MISRA C 2012 Rule 22.4: There shall be no attempt to write to a stream which has been opened as read-only	22.4
MISRA.RESOURCES.FILE.USE_AFTER_CLOSE.2012	Implements MISRA C 2012 Rule 22.6: The value of a pointer to a FILE shall not be used after the associated stream has been closed.	22.6
MISRA.RETURN.NOT_LAST	Return is not the last statement in a function	15.5
MISRA.SHIFT.RANGE.2012	Right operand of shift operation is out of range - greater or equal to the essential type size of left operand, or is negative	12.2
MISRA.SIZEOF.SIDE_EFFECT	Operand of sizeof has side effects	13.6
MISRA.STDLIB.ABORT	Use of 'abort', 'exit', 'getenv' or 'system' from library stdlib.h	21.8
MISRA.STDLIB.ABORT.2012_AMD1	Implements MISRA C 2012 Rule 21.8 (Amendment 1): The library functions abort, exit, and system of 'stdlib.h' shall not be used	21.8
MISRA.STDLIB.ABORT.2012_AMD2	Use of 'abort', 'exit', 'quick_exit' or '_Exit' from library stdlib.h	21.8
MISRA.STDLIB.ATOI	Use of 'atof', 'atoi' or 'atol' from library stdlib.h	21.7
MISRA.STDLIB.BSEARCH.2012	The library functions bsearch and qsort of <stdlib.h> shall not be used	21.9
MISRA.STDLIB.CTYPE.RANGE.2012_AMD1	Any value passed to a function in ctype.h shall be representable as an unsigned char or be the value EOF	21.13
MISRA.STDLIB.EOF.BAD_CMP.2012_AMD1	The macro EOF shall only be compared with the unmodified return value from any Standard Library function capable of returning EOF	22.7
MISRA.STDLIB.FENV.2012	Implements MISRA C 2012 Rule 21.12: The exception handling features of <fenv.h> should not be used	21.12
MISRA.STDLIB.FENV.MACRO.2012	Implements MISRA C 2012 Rule 21.12: The exception handling features of <fenv.h> should not be used	21.12
MISRA.STDLIB.ILLEGAL_REUSE.2012_AMD1	The pointer returned by the Standard Library functions asctime and similar shall not be used following a subsequent call to the same function	21.20
MISRA.STDLIB.ILLEGAL_WRITE.2012_AMD1	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	21.19
MISRA.STDLIB.INCOMPAT_ARGS.2012_AMD1	Implements MISRA C 2012 Rule 21.15: The pointer arguments to the Standard Library functions *memcpy*, *memmove* and *memcmp* shall be pointers to qualified or unqualified versions of compatible types.	21.15
MISRA.STDLIB.LONGJMP	Use of setjmp macro or longjmp function	21.4
MISRA.STDLIB.MEMORY	Use of dynamic heap memory allocation	21.3
MISRA.STDLIB.SIGNAL	Use of the signal handling facilities of signal.h	21.5
MISRA.STDLIB.STDIO	Use of input/output library stdio.h in production code	21.6
MISRA.STDLIB.STDIO.WCHAR.2012	The Standard Library input/output functions shall not be used	21.6
MISRA.STDLIB.SYSTEM.2012_AMD2	Use of 'system' from library stdlib.h	21.21
MISRA.STDLIB.TIME	Use of the time handling functions of library time.h	21.10
MISRA.STDLIB.WCSFTIME.2012	The Standard Library time and date functions shall not be used	21.10
MISRA.STDLIB.WRONGNAME	Reused name of standard library macro, object or function	21.2
MISRA.STDLIB.WRONGNAME.UNDERSCORE	Usage of a reserved name for naming a language entity	21.2
MISRA.STMT.COND.NOT_BOOLEAN.2012	The controlling expression of an if statement or loop statement is not 'Essentially Boolean' type	14.4
MISRA.STMT.NO_COMPOUND	The body of switch, while, do/while or for statement is not a compound statement	15.6
MISRA.STRING_LITERAL.NON_CONST.2012	A string literal shall not be assigned to an object unless the object's type is pointer to const-qualified char	7.4
MISRA.SWITCH.COND.BOOL.2012	A switch-expression shall not have essentially Boolean type.	16.7

MISRA.SWITCH.WELL_FORMED.2012	All switch statements shall be well-formed.	16.1
MISRA.SWITCH.WELL_FORMED.BREAK.2012	An unconditional break statement shall terminate every switch-clause.	16.3
MISRA.SWITCH.WELL_FORMED.DEFAULT.2012	Every switch statement shall have a default label.	16.4
MISRA.SWITCH.WELL_FORMED.DEFAULT.FIRST_OR_LAST.2012	A default label shall appear as either the first or the last switch label of a switch statement.	16.5
MISRA.SWITCH.WELL_FORMED.NESTED_LABEL.2012	A switch label shall only be used when the most closely-enclosing compound statement is the body of the switch statement.	16.2
MISRA.SWITCH.WELL_FORMED.TWO_CLAUSES.2012	Every switch statement shall have at least two switch-clauses.	16.6
MISRA.TOKEN.BADCOM	Inappropriate character sequence in a comment	3.1
MISRA.TOKEN.CPCOM.MULTILINE.2012	Implements MISRA C 2012 Rule 3.2: Line-splicing shall not be used in // comments.	3.2
MISRA.TOKEN.L.SUFFIX.FLOAT	Usage of lowercase character "l" suffix in floating constant	7.3
MISRA.TOKEN.L.SUFFIX.INT	Usage of lowercase character "l" suffix in integer constant	7.3
MISRA.TOKEN.OCTAL.INT	Usage of octal integer constants	7.1
MISRA.TOKEN.UNTERMINATED.ESCAPE.2012	Implements MISRA C 2012 Rule 4.1: Octal and hexadecimal escape sequences shall be terminated.	4.1
MISRA.TYPE.RESTRICT.QUAL.2012	The restrict type qualifier shall not be used	8.14
MISRA.TYPEDEF.NOT_UNIQUE.2012	Typedef name is used for another entity	5.6
MISRA.UNDEF	Undef usage	20.5
MISRA.UNDEF.WRONGNAME	Undefinition of a name from the standard library	21.1
MISRA.UNDEF.WRONGNAME.UNDERSCORE	Undefinition of a reserved name	21.1
MISRA.UNION	Union is used	19.2
MISRA.USE.UNKNOWNDIR	Unknown preprocessor directive is used	20.13
MISRA.VAR.HIDDEN	Identifier declared in an inner scope hides identifier in outer scope	5.3
MLK.MIGHT	Memory Leak - possible	22.1
MLK.MUST	Memory Leak	22.1
MLK.RET.MIGHT	Memory Leak - possible	22.1
MLK.RET.MUST	Memory Leak	22.1
NNTS.MIGHT	Buffer Overflow - Non-null Terminated String	21.17
NNTS.MUST	Buffer Overflow - Non-null Terminated String	21.17
NNTS.TAINTED	Unvalidated User Input Causing Buffer Overflow - Non-Null Terminated String	Dir 4.14
NUM.OVERFLOW.DF	Possible numeric overflow or wraparound	Dir 4.1
PORTING.VAR.EFFECTS	Variable used twice in one expression where one usage is subject to side-effects	13.2
RH.LEAK	Resource leak	22.1
SV.RVT.RETVAL_NOTTESTED	Unvalidated value returned from function	4.1
SV.TAINTED.ALLOC_SIZE	Use of Unvalidated Integer in Memory Allocation	Dir 4.14
SV.TAINTED.BINOP	Use of Unvalidated Integer in Binary Operation	Dir 4.14
SV.TAINTED.CALL.BINOP	Use of Unvalidated Integer in Binary Operation	Dir 4.14
SV.TAINTED.CALL.DEREF	Dereference Of An Unvalidated Pointer	Dir 4.14
SV.TAINTED.CALL.INDEX_ACCESS	Use of Unvalidated Integer as Array Index by Function Call	Dir 4.14
SV.TAINTED.CALL.LOOP_BOUND	Use of Unvalidated Integer in Loop Condition through a Function Call	Dir 4.14
SV.TAINTED.DEREF	Dereference Of An Unvalidated Pointer	Dir 4.14
SV.TAINTED.FMTSTR	Use of Unvalidated Data in a Format String	Dir 4.14
SV.TAINTED.INDEX_ACCESS	Use of Unvalidated Integer as Array Index	Dir 4.14
SV.TAINTED.INJECTION	Command Injection	Dir 4.14
SV.TAINTED.LOOP_BOUND	Use of Unvalidated Integer in Loop Condition	Dir 4.14
SV.TAINTED.PATH_TRAVERSAL	Use of Unvalidated Data in a Path Traversal	Dir 4.14
SV.TAINTED.SECURITY_DECISION	Security Decision	Dir 4.14
UNINIT.HEAP.MIGHT	Uninitialized Heap Use - possible	9.1
UNINIT.HEAP.MUST	Uninitialized Heap Use	9.1
UNINIT.STACK.ARRAY.MIGHT	Uninitialized Array - possible	9.1
UNINIT.STACK.ARRAY.MUST	Uninitialized Array	9.1
UNINIT.STACK.ARRAY.PARTIAL.MUST	Partially Uninitialized Array	9.1
UNINIT.STACK.MIGHT	Uninitialized Variable - possible	9.1
UNINIT.STACK.MUST	Uninitialized Variable	9.1
UNREACH.ENUM	Unreachable code caused by enumeration	2.1

UNREACH.GEN	Unreachable code	2.1
UNREACH.RETURN	Unreachable Void Return	2.1
VA_UNUSED.GEN	Value is Never Used after Assignment	2.2
VA_UNUSED.INIT	Value is Never Used after Initialization	2.2

This document, as well as the software described in it, is furnished under license and may only be used or copied in accordance with the terms of such license. The information contained herein is the exclusive property of RogueWave Software, Inc. a Perforce company. No part of this documentation may be copied, translated, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of Perforce Software, Inc. If you find any problems in the documentation, please report them to us in writing.

Klocwork is a registered trademark of RogueWave Software, Inc., a Perforce company.

All other trademarks are the property of their respective owners. All help content for Klocwork's MISRA checkers is copyright by the MISRA Consortium Limited