CERT-C:2016 Standards Model Summary for C

The LDRA tool suite® is developed and certified to BS EN ISO 9001:2015, TÜV SÜD and SGS-TÜV Saar.

This information is applicable to version 10.2.0 of the LDRA tool suite®.
It is correct as of 11th September 2023.

Compliance is measured against
"SEI CERT C Coding Standard, 2016 Edition"
2016

Further information is available at http://www.securecoding.cert.org

Classification	Enhanced Enforcement	Fully Implemented	Partially Implemented	Not yet Implemented	Not statically Checkable	Total
Rule	15	35	33	14	19	116
Recommendation	31	47	44	20	27	169
Total	46	82	77	34	46	285

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
ARR00-C	Recommendation	Understand how arrays work	Checks on array out of bounds are covered by the more specific ARR30-C rule.		
ARR01-C	Recommendation	Do not apply the sizeof operator to a pointer when taking the size of an array		401 S	Use of sizeof on an array parameter.
ARR02-C	Recommendation	Explicitly specify array bounds, even if implicitly defined by an initializer		127 S	Array has no bounds specified. Modifiers :285 = 0 (Default) 286 = 1 426 = 0 (Default)
				397 S	Array initialisation has insufficient items. Modifiers :415 = 1 445 = 0 (Default) 450 = 0 (Default) 455 = 0 (Default)
				404 S	Array initialisation has too many items. Modifiers :399 = 1
ARR30-C	Rule	Do not form or use out-of-bounds pointers or array subscripts		45 D	Pointer not checked for null before use.
				141 D	Heap-based array index not checked before use.
				47 S	Array bound exceeded.
				476 S	Array index not unsigned.
				489 S	Insufficient space for operation.
				692 S	Array index is negative.
				64 X	Array bound exceeded at call.
				66 X	Insufficient array space at call.
				68 X	Parameter indexing array too big at call.
				69 X	Global array bound exceeded at use.
				70 X	Array has insufficient space.
				71 X	Insufficient space for copy.
				79 X	Size mismatch in memcpy/memset.

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
ARR32-C	Rule	Ensure size arguments for variable length arrays are in a valid range	The analysis highlights all declarations of variable length arrays. The user must then manually check that the size is in a valid range.	621 S	Variable-length array declared.
ARR36-C	Rule	Do not subtract or compare two pointers that do not refer to the same array		437 S	< > <= >= used on different object pointers.
				438 S	Pointer subtraction not addressing one array.
ARR37-C	Rule	Do not add or subtract an integer to a pointer to a non-array object		567 S	Pointer arithmetic is not on array. Modifiers :436 = 1
ARR38-C	Rule	Guarantee that library functions do not form invalid pointers		64 X	Array bound exceeded at call.
				66 X	Insufficient array space at call.
				68 X	Parameter indexing array too big at call.
				69 X	Global array bound exceeded at use.
				70 X	Array has insufficient space.
				71 X	Insufficient space for copy.
				79 X	Size mismatch in memcpy/memset.
ARR39-C	Rule	Do not add or subtract a scaled integer to a pointer		47 S	Array bound exceeded.
				489 S	Insufficient space for operation.
				567 S	Pointer arithmetic is not on array.
				692 S	Array index is negative.
				64 X	Array bound exceeded at call.
				66 X	Insufficient array space at call.
				68 X	Parameter indexing array too big at call.
				69 X	Global array bound exceeded at use.
				70 X	Array has insufficient space.
				71 X	Insufficient space for copy.
CON00-C	Recommendation	Avoid race conditions with multiple threads			
CON01-C	Recommendation	Acquire and release synchronization primitives in the same module, at the same level of abstraction			
CON02-C	Recommendation	Do not use volatile as a synchronization primitive			
CON03-C	Recommendation	Ensure visibility when accessing shared variables			
CON04-C	Recommendation	Join or detach threads even if their exit status is unimportant			
CON05-C	Recommendation	Do not perform operations that can block while holding a lock			
CON06-C	Recommendation	Ensure that every mutex outlives the data it protects			
CON07-C	Recommendation	Ensure that compound operations on shared variables are atomic			
CON08-C	Recommendation	Do not assume that a group of calls to independently atomic methods is atomic			
CON09-C	Recommendation	Avoid the ABA problem when using lock-free algorithms			
CON30-C	Rule	Clean up thread-specific storage			
CON31-C	Rule	Do not destroy a mutex while it is locked			
CON32-C	Rule	Prevent data races when accessing bit-fields from multiple threads			
CON33-C	Rule	Avoid race conditions when using library functions	44 S highlights uses of getenv and strerror. The user must then manually check that the function is used a permitted by the rule.	44 S	Use of banned function, type or variable.
CON34-C	Rule	Declare objects shared between threads with appropriate storage durations			
CON35-C	Rule	Avoid deadlock by locking in a predefined order			

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
CON36-C	Rule	Wrap functions that can spuriously wake up in a loop			
CON37-C	Rule	Do not call signal() in a multithreaded program	44 S highlights all uses of signal. The user must then manually check that the function is used a permitted by the rule.	44 S	Use of banned function, type or variable.
CON38-C	Rule	Preserve thread safety and liveness when using condition variables			
CON39-C	Rule	Do not join or detach a thread that was previously joined or detached			
CON40-C	Rule	Do not refer to an atomic variable twice in an expression			
CON41-C	Rule	Wrap functions that can fail spuriously in a loop			
DCL00-C	Recommendation	const-qualify immutable objects		78 D	Global variable should be declared const.
				93 D	Local variable should be declared const.
				200 S	Define used for numeric constant.
DCL01-C	Recommendation	Do not reuse variable names in subscopes	The analysis checks for the following: presence/absence of underscore, O v 0, I v 1, l vl , l v 1, S v 5, Z v 2, n v h and B v 8. It does not check for rn v m.	131 S	Name reused in inner scope.
DCL02-C	Recommendation	Use visually distinct identifiers		67 X	Identifier is typographically ambiguous.
DCL03-C	Recommendation	Use a static assertion to test the value of a constant expression	The analysis highlights uses of the assert macro.	44 S	Use of banned function, type or variable.
DCL04-C	Recommendation	Do not declare more than one variable per declaration	The analysis highlights all cases of more than one variable in a loop declaration. This includes mulitiple declaration with no initialization which are permitted by exception DCL04-EX2. The user must then manually check whether the exception applies.	579 S	More than one variable per declaration.
DCL05-C	Recommendation	Use typedefs of non-pointer types only		299 S	Pointer to function declared without typedef.
DCL06-C	Recommendation	Use meaningful symbolic constants to represent literal values	Exception DLC06-EX1 permits numeric constants where this aids readability. This is not statically checkable, and therefore the analysis highlights all use of numeric literals in expressions. However, the documentation for 201 S shows how the analysis can be relaxed to permit a range of integer literals.	201 S	Use of numeric literal in expression. Modifiers :217 = 0 (Default) 163 = 1 (Default) 237 = 0 (Default) 207 = 0 (Default)
DCL07-C	Recommendation	Include the appropriate type information in function declarators		21 S	Number of parameters does not match.
				135 S	Parameter list is KR.
				170 S	Procedure call has no prototype and no defn.
DCL08-C	Recommendation	Properly encode relationships in constant definitions	This recommendation is not statically checkable as it requires an undertanding of the programmers intention		
DCL09-C	Recommendation	Declare functions that return errno with a return type of errno_t	The analysis checks that when errno is returned that the return type of the function is errno_t.	643 S	Function return type is not errno_t.
DCL10-C	Recommendation	Maintain the contract between the writer and caller of variadic functions	In general, this rule is not statically checkable since it depends on the intent of the algorithmm. The analysis highlights all uses of ellipsis. The user must then manually check that the number of arguments matches the algorithm.	41 S	Ellipsis used in procedure parameter list. Modifiers :522 = 0 (Default)
DCL11-C	Recommendation	Understand the type issues associated with variadic functions	In general, this rule is not statically checkable since it depends on the intent of the algorithmm. The analysis highlights all uses of ellipsis	41 S	Ellipsis used in procedure parameter list. Modifiers :522 = 0 (Default)
				589 S	Format is not appropriate type.
DCL12-C	Recommendation	Implement abstract data types using opaque types	The analysis highlights pointers that are never dereferenced in a translation unit but have their structure implementation visible.	104 D	Structure implementation not hidden.
DCL13-C	Recommendation	Declare function parameters that are pointers to values not changed by the function as const	The analysis requires the use of const for pointers that are not assigned to. Otherwise it assumes that the user intended to make the assignment	120 D	Pointer param should be declared pointer to const.
DCL15-C	Recommendation	Declare file-scope objects or functions that do not need external linkage as static		27 D	Variable should be declared static.
				61 D	Procedure should be declared static.
				553 S	Function and proto should both be static.
DCL16-C	Recommendation	Use "L," not "l," to indicate a long value		252 S	Lower case suffix to literal number. Modifiers :418 = 1
DCL17-C	Recommendation	Beware of miscompiled volatile-qualified variables	The analysis highlights where a volatile is used more than once in an expression.	134 S	Volatile variable in complex expression. Modifiers :460 = 0 (Default) 461 = 1

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
DCL18-C	Recommendation	Do not begin integer constants with 0 when specifying a decimal value		83 S	Octal number found Modifiers :469 = 0 (Default)
DCL19-C	Recommendation	Minimize the scope of variables and functions		25 D	Scope of variable could be reduced.
				61 D	Procedure should be declared static.
				40 S	Loop index is not declared locally.
DCL20-C	Recommendation	Explicitly specify void when a function accepts no arguments		63 S	Empty parameter list to procedure/function.
DCL21-C	Recommendation	Understand the storage of compound literals			
DCL22-C	Recommendation	Use volatile for data that cannot be cached	The presence of a DD anomaly may indicate that a variable should be declared as a volatile. A DD anomaly occurs when a variable is assigned a value and has been reassigned without being referenced after the first assignment. Information about the variables affected are listed in the Data Flow Analysis Report.	151 D	DD data flow anomaly found. Modifiers :391 = 1
DCL23-C	Recommendation	Guarantee that mutually visible identifiers are unique		17 D	Identifier not unique within *** characters.
				355 S	Variables not unique within *** characters.
				61 X	Identifier match in *** chars.
DCL30-C	Rule	Declare objects with appropriate storage durations		42 D	Local pointer returned in function result.
				77 D	Local structure returned in function result.
				71 S	Pointer assignment to wider scope. Modifiers :222 = 1
				565 S	Assignment to wider scope.
DCL31-C	Rule	Declare identifiers before using them		24 D	Procedure definition has no associated prototype.
				41 D	Procedure call has no prototype declared.
				20 S	Parameter not declared explicitly.
				326 S	Declaration is missing type.
				496 S	Function call with no prior declaration.
DCL36-C	Rule	Do not declare an identifier with conflicting linkage classifications		461 S	Identifier with ambiguous linkage.
				575 S	Linkage differs from previous declaration.
				2 X	Ambiguous declaration of variable.
DCL37-C	Rule	Do not declare or define a reserved identifier	Note: Clashes between names in library header files, will only be idenitifed if the header files are expanded.	86 S	Attempt to define reserved word.
				218 S	Name is used in standard libraries.
				219 S	User name starts with underscore. Modifiers :412 = 0 (Default)
				580 S	Macro redefinition without using #undef.
				626 S	#define of keyword. Modifiers :391 = 1
DCL38-C	Rule	Use the correct syntax when declaring a flexible array member		648 S	Deprecated form of flexible array. Modifiers :464 = 1
DCL39-C	Rule	Avoid information leakage when passing a structure across a trust boundary			
DCL40-C	Rule	Do not create incompatible declarations of the same function or object	The analysis does not currently check the declaration types of the functions across a system.	17 D	Identifier not unique within *** characters.
				1 X	Declaration types do not match across a system.
DCL41-C	Rule	Do not declare variables inside a switch statement before the first case label		385 S	MISRA switch statement syntax violation.
ENV01-C	Recommendation	Do not make assumptions about the size of an environment variable			
ENV02-C	Recommendation	Beware of multiple environment variables with the same effective name			

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
ENV03-C	Recommendation	Sanitize the environment when invoking external programs		588 S	Use of system function.
ENV30-C	Rule	Do not modify the object referenced by the return value of certain functions		107 D	Attempt to change system call capture string.
ENV31-C	Rule	Do not rely on an environment pointer following an operation that may invalidate it	The analysis highlights declarations of main which do not match the two standard specifcations given by the language. Adding a third parameter char * envp[] will violate this standard.	118 S	main must be int (void) or int (int,char*[]).
ENV32-C	Rule	All exit handlers must return normally	The analysis highlights all uses of exit and abort, and all jumps out of a procedure. The user must then manually check whether they occur within a call of atexit.	7 S	Jump out of procedure.
				122 S	Use of abort, exit, etc.
ENV33-C	Rule	Do not call system()		588 S	Use of system function.
ENV34-C	Rule	Do not store pointers returned by certain functions		133 D	Pointer from system function used after subsequent call.
ERR00-C	Recommendation	Adopt and implement a consistent and comprehensive error-handling policy			
ERR01-C	Recommendation	Use ferror() rather than errno to check for FILE stream errors	44 S highlights all uses of errno. The user must then manually check whether the use is related to the File stream.	44 S	Use of banned function, type or variable.
ERR02-C	Recommendation	Avoid in-band error indicators			
ERR03-C	Recommendation	Use runtime-constraint handlers when calling the bounds-checking interfaces			
ERR04-C	Recommendation	Choose an appropriate termination strategy			
ERR05-C	Recommendation	Application-independent code should provide error detection without dictating error handling			
ERR06-C	Recommendation	Understand the termination behavior of assert() and abort()	44 S highlights all uses of assert. The user must then manually check whether the use is compliant with this rule.	44 S	Use of banned function, type or variable.
ERR07-C	Recommendation	Prefer functions that support error checking over equivalent functions that don't	44 S highlights all uses of atoi, atof, atol and atoll. ctime can be added as described in the standard's documentation.	44 S	Use of banned function, type or variable.
				593 S	Use fseek() rather than rewind().
				594 S	Use setvbuf() rather than setbuf().
ERR30-C	Rule	Set errno to zero before calling a library function known to set errno, and check errno only after the function returns a value indicating failure	111 D, D121 and D122 check that errno has been set before an errno setting library functions is called, and that errno is then checked after the call. D134 checks that no other function call occurs between the call of an errno setting function and the check on errno.	111 D	errno checked without having been set for errno setting fn.
				121 D	errno neither set nor checked for errno setting function.
				122 D	errno not checked after being set for errno setting fn.
				132 D	errno checked after call to non-errno setting function.
				134 D	errno not checked before subsequent function call.
ERR32-C	Rule	Do not rely on indeterminate values of errno	44 S highlights all uses of errno. The user must then manually check whether the use is compliant with this rule.	44 S	Use of banned function, type or variable.
ERR33-C	Rule	Detect and handle standard library errors		80 D	Potentially unused function-modified value.
				124 D	Var set by std lib func return not checked before use.
				130 D	Global set by std lib func return not checked before use.
				382 S	(void) missing for discarded return value. Modifiers :383 = 1
EXP00-C	Recommendation	Use parentheses for precedence of operation		49 S	Logical conjunctions need brackets.
				361 S	Expression needs brackets. Modifiers :119 = 0 (Default) 264 = 1 414 = 0 (Default) 420 = 0 (Default) 421 = 1
EXP02-C	Recommendation	Be aware of the short-circuit behavior of the logical AND and OR operators		35 D	Expression has side effects.
				1 Q	Call has execution order dependant side effects.
				133 S	Assignment operator in RHS of && or ||.
				406 S	Use of ++ or -- on RHS of && or || operator.
				408 S	Volatile variable accessed on RHS of && or ||.
EXP03-C	Recommendation	Do not assume the size of a structure is the sum of the sizes of its members	578 S highlights all uses of sizeof on an arithmetic expression. The user must then manually check whether the expression violates this rule.	578 S	Sizeof used in arithmetic expression.
EXP05-C	Recommendation	Do not cast away a const qualification		203 S	Cast on a constant value. Modifiers :390 = 0 (Default)

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
EXP07-C	Recommendation	Do not diminish the benefits of constants by assuming their values in expressions	201 S highlights all numeric literals in expressions. The user should manually confirm that they are not 'magic numbers'. The standard can be configured to permit user-specfied ranges of integers	201 S	Use of numeric literal in expression. Modifiers :217 = 0 (Default) 163 = 1 (Default) 237 = 0 (Default) 207 = 0 (Default)
EXP08-C	Recommendation	Ensure pointer arithmetic is used correctly		45 D	Pointer not checked for null before use.
				53 D	Attempt to use uninitialised pointer.
				54 D	Unsafe use of function pointer variable.
				438 S	Pointer subtraction not addressing one array.
				576 S	Function pointer is of wrong type.
EXP09-C	Recommendation	Use sizeof to determine the size of a type or variable	201 S highlights all numeric literals in expressions. The user should manually confirm that sizeof would not be more appropriate.	201 S	Use of numeric literal in expression. Modifiers :217 = 0 (Default) 163 = 1 (Default) 237 = 0 (Default) 207 = 0 (Default)
EXP10-C	Recommendation	Do not depend on the order of evaluation of subexpressions or the order in which side effects take place		35 D	Expression has side effects.
				72 D	Potential side effect problem in expression.
				1 Q	Call has execution order dependant side effects.
				134 S	Volatile variable in complex expression. Modifiers :460 = 0 (Default) 461 = 1
EXP11-C	Recommendation	Do not make assumptions regarding the layout of structures with bit-fields	Most C operations are not affected by the alignment of bit fields in a structure. It only presents a problem when such a structure is cast to another type. 554 S highlights all cast of pointer to structures to another type.	554 S	Cast to an unrelated type. Modifiers :439 = 1 440 = 0 (Default)
EXP12-C	Recommendation	Do not ignore values returned by functions	By default the analysis is configured to not output a warning on the printf family of functions. Instructions for enabling the check on printf functions can be found in the 382 S documentation.	382 S	(void) missing for discarded return value. Modifiers :383 = 1
EXP13-C	Recommendation	Treat relational and equality operators as if they were nonassociative		433 S	Type conversion without cast. Modifiers :374 = 0 (Default) 391 = 1 431 = 1 428 = c : 0 (Default) 470 = 0 (Default) 386 = 1 (Default) 549 = 0 (Default)
EXP15-C	Recommendation	Do not place a semicolon on the same line as an if, for, or while statement	The presence of a semicolon on the same line as an if for or while statement can be avoided by ensuring the presence of { . . . } as the body for the statements.	11 S	No brackets to loop body (added by Testbed).
				12 S	No brackets to then/else (added by Testbed). Modifiers :tbend F
				428 S	No {} for switch (added by Testbed).
EXP16-C	Recommendation	Do not compare function pointers to constant values		99 S	Function use is not a call. Modifiers :498 = 0 (Default)
EXP19-C	Recommendation	Use braces for the body of an if, for, or while statement		11 S	No brackets to loop body (added by Testbed).
				12 S	No brackets to then/else (added by Testbed). Modifiers :tbend F
				428 S	No {} for switch (added by Testbed).
EXP20-C	Recommendation	Perform explicit tests to determine success, true and false, and equality		114 S	Expression is not Boolean. Modifiers :386 = 1 (Default) 528 = 0 (Default)
EXP30-C	Rule	Do not depend on the order of evaluation for side effects		35 D	Expression has side effects.
				1 Q	Call has execution order dependant side effects.
				9 S	Assignment operation in expression. Modifiers :259 = 0 (Default) 266 = 0 (Default)
				30 S	Deprecated usage of ++ or -- operators found. Modifiers :122 = 1 (Default)
				134 S	Volatile variable in complex expression. Modifiers :460 = 0 (Default) 461 = 1
EXP32-C	Rule	Do not access a volatile object through a nonvolatile reference		344 S	Cast on volatile value. Modifiers :395 = 0 (Default)

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
EXP33-C	Rule	Do not read uninitialized memory		53 D	Attempt to use uninitialised pointer.
				69 D	UR anomaly, variable used before assignment.
				631 S	Declaration not reachable.
				652 S	Object created by malloc used before initialisation.
EXP34-C	Rule	Do not dereference null pointers		45 D	Pointer not checked for null before use.
				123 D	File pointer not checked for null before use.
				128 D	Global pointer not checked within this procedure.
				129 D	Global file pointer not checked within this procedure.
				135 D	Pointer assigned to NULL may be dereferenced.
				136 D	Global pointer assigned to NULL may be dereferenced.
				652 S	Object created by malloc used before initialisation.
EXP35-C	Rule	Do not modify objects with temporary lifetime		42 D	Local pointer returned in function result.
				77 D	Local structure returned in function result.
				642 S	Function return type with array field.
EXP36-C	Rule	Do not cast pointers into more strictly aligned pointer types		540 S	Cast from pointer to void to pointer. Modifiers :381 = 0 (Default)
				554 S	Cast to an unrelated type. Modifiers :439 = 1 440 = 0 (Default)
				606 S	Cast involving function pointer. Modifiers :398 = 0 (Default) 433 = 1 439 = 1
				701 S	Pointer cast to pointer to void.
EXP37-C	Rule	Call functions with the correct number and type of arguments		41 D	Procedure call has no prototype declared.
				21 S	Number of parameters does not match.
				98 S	Actual and formal parameters inconsistent (MR).
				170 S	Procedure call has no prototype and no defn.
				496 S	Function call with no prior declaration.
				576 S	Function pointer is of wrong type.
EXP39-C	Rule	Do not access a variable through a pointer of an incompatible type		554 S	Cast to an unrelated type. Modifiers :439 = 1 440 = 0 (Default)
				697 S	Pointer converted to different pointer type. Modifiers :441 = 0 (Default)
EXP40-C	Rule	Do not modify constant objects		582 S	const object reassigned.
EXP42-C	Rule	Do not compare padding data		618 S	Use of memcmp between structures.
EXP43-C	Rule	Avoid undefined behavior when using restrict-qualified pointers	613 S highlights all uses of the restrict qualifier. The user must then manually check that the pointers cannot point to the same object.	480 S	String function params access same variable.
				489 S	Insufficient space for operation.
				613 S	Use of restrict keyword. Modifiers :350 = 1
EXP44-C	Rule	Do not rely on side effects in operands to sizeof, _Alignof, or _Generic		54 S	Sizeof operator with side effects.
				653 S	Apparent side effects in _Generic or _Alignof.
EXP45-C	Rule	Do not perform assignments in selection statements	The analysis will also highlight the exceptions to this rule. The user must manually check that the exceptions do not apply.	114 S	Expression is not Boolean. Modifiers :386 = 1 (Default) 528 = 0 (Default)
				132 S	Assignment operator in boolean expression.
EXP46-C	Rule	Do not use a bitwise operator with a Boolean-like operand		136 S	Bit operator with boolean operand.
FIO01-C	Recommendation	Be careful using functions that use file names for identification		592 S	Use of filename based functions.
FIO02-C	Recommendation	Canonicalize path names originating from tainted sources	This rule is in general undecidable. The analysis only considers those cases that are deciadable.	85 D	Filename not verified before fopen.

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
FIO03-C	Recommendation	Do not make assumptions about fopen() and file creation	The analysis highlights all uses of fopen and fopen_s. The user must manually check whether they are protected as described in the rule.	44 S	Use of banned function, type or variable.
FIO05-C	Recommendation	Identify files using multiple file attributes	The analysis highlights all uses of fopen. The user must manually check whether they are protected as described in the rule.	44 S	Use of banned function, type or variable.
FIO06-C	Recommendation	Create files with appropriate access permissions	The analysis highlights all uses of fopen. The user must manually check whether they are protected as described in the rule	44 S	Use of banned function, type or variable.
FIO08-C	Recommendation	Take care when calling remove() on an open file		81 D	Attempt to remove an open file.
FIO09-C	Recommendation	Be careful with binary data when transferring data across systems	The analysis highlights all uses of fread and fwrite. The user must manually check whether they are protected as described in the rule.	44 S	Use of banned function, type or variable.
FIO10-C	Recommendation	Take care when using the rename() function	The analysis highlights all uses of rename. The user must manually check whether they are protected as described in the rule.	592 S	Use of filename based functions.
FIO11-C	Recommendation	Take care when specifying the mode parameter of fopen()		590 S	Mode fault in fopen.
FIO13-C	Recommendation	Never push back anything other than one read character		83 D	Potentially repeated call to ungetc.
FIO14-C	Recommendation	Understand the difference between text mode and binary mode with file streams			
FIO15-C	Recommendation	Ensure that file operations are performed in a secure directory			
FIO17-C	Recommendation	Do not rely on an ending null character when using fread()	The analysis highlights all uses of fread. The user must manually check whether they are protected as described in the rule.	44 S	Use of banned function, type or variable.
FIO18-C	Recommendation	Never expect fwrite() to terminate the writing process at a null character	The analysis highlights all uses of fwrite. The user must manually check whether they are protected as described in the rule.	44 S	Use of banned function, type or variable.
FIO19-C	Recommendation	Do not use fseek() and ftell() to compute the size of a regular file	The analysis highlights all uses of fseek and fteel. The user must manually check whether they are protected as described in the rule.	44 S	Use of banned function, type or variable.
FIO20-C	Recommendation	Avoid unintentional truncation when using fgets() or fgetws()	The analysis highlights all uses of fgets and fgetws. The user must manually check whether they are protected as described in the rule.	44 S	Use of banned function, type or variable.
FIO21-C	Recommendation	Do not create temporary files in shared directories	The analysis highlights all uses of tmpnam, tmpfile, mktemp and tmpnam_s. The user must manually check whether they are protected as described in the rule.	44 S	Use of banned function, type or variable.
FIO22-C	Recommendation	Close files before spawning processes		49 D	File pointer not closed on exit.
FIO23-C	Recommendation	Do not exit with unflushed data in stdout or stderr			
FIO24-C	Recommendation	Do not open a file that is already open		75 D	Attempt to open file pointer more than once.
FIO30-C	Rule	Exclude user input from format strings	The analysis highlights all uses of possibly tainted values from the I/O library functions.	86 D	User input not checked before use.
FIO32-C	Rule	Do not perform operations on devices that are only appropriate for files			
FIO34-C	Rule	Distinguish between characters read from a file and EOF or WEOF		662 S	EOF compared with char.
FIO37-C	Rule	Do not assume that fgets() or fgetws() returns a nonempty string when successful	The analysis highlights all uses of fgets and fgetws. The user must manually check whether they are protected as described in the rule.	44 S	Use of banned function, type or variable.
FIO38-C	Rule	Do not copy a FILE object		591 S	Inappropriate use of file pointer.
FIO39-C	Rule	Do not alternately input and output from a stream without an intervening flush or positioning call		84 D	No fseek or flush before I/O.
FIO40-C	Rule	Reset strings on fgets() or fgetws() failure	The analysis highlights all uses of fgets and fgetws. The user must manually check whether they are protected as described in the rule.	44 S	Use of banned function, type or variable.
FIO41-C	Rule	Do not call getc(), putc(), getwc(), or putwc() with a stream argument that has side effects		35 D	Expression has side effects.
				1 Q	Call has execution order dependant side effects.
				9 S	Assignment operation in expression. Modifiers :259 = 0 (Default) 266 = 0 (Default)
				30 S	Deprecated usage of ++ or -- operators found. Modifiers :122 = 1 (Default)
				134 S	Volatile variable in complex expression. Modifiers :460 = 0 (Default) 461 = 1
FIO42-C	Rule	Close files when they are no longer needed		49 D	File pointer not closed on exit.

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
FIO44-C	Rule	Only use values for fsetpos() that are returned from fgetpos()		82 D	fsetpos values not generated by fgetpos.
FIO45-C	Rule	Avoid TOCTOU race conditions while accessing files		75 D	Attempt to open file pointer more than once.
FIO46-C	Rule	Do not access a closed file		48 D	Attempt to write to unopened file.
FIO47-C	Rule	Use valid format strings		486 S	Incorrect number of formats in output function.
				589 S	Format is not appropriate type.
FLP00-C	Recommendation	Understand the limitations of floating-point numbers			
FLP01-C	Recommendation	Take care in rearranging floating-point expressions			
FLP02-C	Recommendation	Avoid using floating-point numbers when precise computation is needed		56 S	Equality comparison of floating point. Modifiers :223 = 1
FLP03-C	Recommendation	Detect and handle floating-point errors		43 D	Divide by zero found.
FLP04-C	Recommendation	Check floating-point inputs for exceptional values			
FLP05-C	Recommendation	Do not use denormalized numbers			
FLP06-C	Recommendation	Convert integers to floating point for floating-point operations	The analysis also highlights situations when one operand has integer type and the other has floating type.	435 S	Float/integer conversion without cast. Modifiers :374 = 0 (Default) 432 = 0 (Default) 453 = 1
FLP07-C	Recommendation	Cast the return value of a function that returns a floating-point type			
FLP30-C	Rule	Do not use floating-point variables as loop counters		39 S	Unsuitable type for loop variable.
FLP32-C	Rule	Prevent or detect domain and range errors in math functions			
FLP34-C	Rule	Ensure that floating-point conversions are within range of the new type	453 S highlights all implicit conversions from integer to floating. The user must then manually check whether the conversion is protected. See comments under FLP36-C for further configuration of 435 S.	435 S	Float/integer conversion without cast. Modifiers :374 = 0 (Default) 432 = 0 (Default) 453 = 1
				444 S	Integral type cast to non-integral. Modifiers :509 = 0 (Default) 510 = 1 511 = 0 (Default) 512 = 1 513 = 0 (Default) 529 = 0 (Default)
				441 S	Float cast to non-float. Modifiers :502 = 0 (Default) 503 = 0 (Default) 504 = 0 (Default) 505 = 0 (Default) 506 = 0 (Default) 529 = 0 (Default)
				445 S	Narrower float conversion without cast. Modifiers :374 = 0 (Default)
				693 S	Float expression cast to narrower float type.
FLP36-C	Rule	Preserve precision when converting integral values to floating-point type	453 S highlights all conversions from integer to floating, as this standard is also mapped to recommendation FLP06-C. If recommendation FLP06-C is not being followed, 453 S can be relaxed to permit the conversion of integer to floats when the integer size is less the float size by setting 453 to 0 in all uses of 453 S within the model.	435 S	Float/integer conversion without cast. Modifiers :374 = 0 (Default) 432 = 0 (Default) 453 = 1
FLP37-C	Rule	Do not use object representations to compare floating-point values		618 S	Use of memcmp between structures.
INT00-C	Recommendation	Understand the data model used by your implementation(s)			
INT01-C	Recommendation	Use rsize_t or size_t for all integer values representing the size of an object			
				52 S	Unsigned expression negated.
				101 S	Function return type inconsistent. Modifiers :374 = 0 (Default) 427 = 1 428 = c : 0 (Default) 430 = 0 (Default) 431 = 1 453 = 1 470 = 0 (Default)

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
INT02-C	Recommendation	Understand integer conversion rules		107 S	Type mismatch in ternary expression. Modifiers :428 = c : 0 (Default) 429 = 1 470 = 0 (Default) 525 = 0 (Default)
				332 S	Widening cast on complex integer expression. Modifiers :529 = 0 (Default)
				334 S	No cast when ~ or << applied to small types (MR). Modifiers :191 = 1
				433 S	Type conversion without cast. Modifiers :374 = 0 (Default) 391 = 1 431 = 1 428 = c : 0 (Default) 470 = 0 (Default) 386 = 1 (Default) 549 = 0 (Default)
				434 S	Signed/unsigned conversion without cast. Modifiers :374 = 0 (Default) 358 = 0 (Default) 427 = 1 428 = c : 0 (Default) 429 = 1 430 = 0 (Default) 470 = 0 (Default)
				446 S	Narrower int conversion without cast. Modifiers :374 = 0 (Default)
				452 S	No cast for widening complex int expression (MR). Modifiers :191 = 1 529 = 0 (Default)
				457 S	Implicit int widening for function return (MR). Modifiers :191 = 1
				458 S	Implicit conversion: actual to formal param (MR).
INT04-C	Recommendation	Enforce limits on integer values originating from tainted sources			
INT05-C	Recommendation	Do not use input functions to convert character data if they cannot handle all possible inputs	The analysis highlights all uses of scanf and related functions. The user must manually check whether they are protected as described in the rule.	44 S	Use of banned function, type or variable.
INT07-C	Recommendation	Use only explicitly signed or unsigned char type for numeric values		101 S	Function return type inconsistent Modifiers :374 = 0 (Default) 427 = 1 428 = c : 0 (Default) 430 = 0 (Default) 431 = 1 453 = 1 470 = 0 (Default)
				329 S	Operation not appropriate to plain char. Modifiers :191 = 1 331 = 0 (Default) 391 = 1
				433 S	Type conversion without cast. Modifiers :374 = 0 (Default) 391 = 1 431 = 1 428 = c : 0 (Default) 470 = 0 (Default) 386 = 1 (Default) 549 = 0 (Default)
				458 S	Implicit conversion: actual to formal param (MR).
INT08-C	Recommendation	Verify that all integer values are in range		488 S	Value outside range of underlying type. Modifiers :535 = 0 (Default)
				493 S	Numeric overflow.
INT09-C	Recommendation	Ensure enumeration constants map to unique values		85 S	Incomplete initialisation of enumerator. Modifiers :377 = 0 (Default)
				630 S	Duplicated enumeration value.
INT10-C	Recommendation	Do not assume a positive remainder when using the % operator		584 S	Remainder of % op could be negative.
INT12-C	Recommendation	Do not make assumptions about the type of a plain int bit-field when used in an expression	S73 should be used when running with C90 and S520 when running with C99 or C++. The choice of standard is determined by the setting of modifier 350 to 1, which is set to 1 by default for this model.	73 S	Bit field not signed or unsigned int. Modifiers :234 = 0 (Default) 331 = 0 (Default) 350 = 1
				520 S	Bit field is not bool or explicit integral. Modifiers :331 = 0 (Default) 350 = 1
				50 S	Use of shift operator on signed type.
				51 S	Shifting value too far. Modifiers :520 = 0 (Default)
				120 S	Use of bit operator on signed type. Modifiers :457 = 1

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
INT13-C	Recommendation	Use bitwise operators only on unsigned operands		136 S	Bit operator with boolean operand.
				249 S	Shift operator with boolean operand.
				329 S	Operation not appropriate to plain char. Modifiers :191 = 1 331 = 0 (Default) 391 = 1
				345 S	Bit operator with floating point operand.
				403 S	Negative (or potentially negative) shift.
INT14-C	Recommendation	Avoid performing bitwise and arithmetic operations on the same data	The analysis detects operations on the data in the same statement.	585 S	Bitwise and arith operations on same data.
INT15-C	Recommendation	Use intmax_t or uintmax_t for formatted IO on programmer-defined integer types	The analysis highlights the use of all typedefs in printf. The user must then manually check that they obey this rule.	586 S	Format is not %j for user defined type. Modifiers :350 = 1
INT16-C	Recommendation	Do not make assumptions about representation of signed integers		50 S	Use of shift operator on signed type.
				120 S	Use of bit operator on signed type. Modifiers :457 = 1
INT17-C	Recommendation	Define integer constants in an implementation-independent manner			
INT18-C	Recommendation	Evaluate integer expressions in a larger size before comparing or assigning to that size		452 S	No cast for widening complex int expression (MR). Modifiers :191 = 1 529 = 0 (Default)
INT30-C	Rule	Ensure that unsigned integer operations do not wrap		493 S	Numeric overflow.
INT31-C	Rule	Ensure that integer conversions do not result in lost or misinterpreted data		433 S	Type conversion without cast. Modifiers :374 = 0 (Default) 391 = 1 431 = 1 428 = c : 0 (Default) 470 = 0 (Default) 386 = 1 (Default) 549 = 0 (Default)
				434 S	Signed/unsigned conversion without cast. Modifiers :374 = 0 (Default) 358 = 0 (Default) 427 = 1 428 = c : 0 (Default) 429 = 1 430 = 0 (Default) 470 = 0 (Default)
INT32-C	Rule	Ensure that operations on signed integers do not result in overflow		493 S	Numeric overflow.
INT33-C	Rule	Ensure that division and remainder operations do not result in divide-by-zero errors		43 D	Divide by zero found.
				127 D	Local or member denominator not checked before use.
				131 D	Global denominator not checked within this procedure.
				137 D	Parameter used as denominator not checked before use.
				248 S	Divide by zero in preprocessor directive.
				629 S	Divide by zero found.
				80 X	Divide by zero found.
INT34-C	Rule	Do not shift an expression by a negative number of bits or by greater than or equal to the number of bits that exist in the operand		51 S	Shifting value too far. Modifiers :520 = 0 (Default)
				403 S	Negative (or potentially negative) shift.
				479 S	Right shift loses all bits.
INT35-C	Rule	Use correct integer precisions			
INT36-C	Rule	Converting a pointer to integer or integer to pointer		439 S	Cast from pointer to integral type. Modifiers :548 = 0 (Default)
				440 S	Cast from integral type to pointer. Modifiers :397 = 0 (Default) 548 = 0 (Default)

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
MEM00-C	Recommendation	Allocate and free memory in the same module, at the same level of abstraction		50 D	Memory not freed after last reference.
				112 D	Free called twice on same variable.
				484 S	Attempt to use already freed object. Modifiers :350 = 1
MEM01-C	Recommendation	Store a new value in pointers immediately after free()	The analysis highlights the root cause of the problem which is that the object has been freed twice.	112 D	Free called twice on same variable.
				484 S	Attempt to use already freed object. Modifiers :350 = 1
MEM02-C	Recommendation	Immediately cast the result of a memory allocation function call into a pointer to the allocated type			
MEM03-C	Recommendation	Clear sensitive information stored in reusable resources	The analysis highlights all uses of malloc and realloc functions. (Currently the analysis also highlights the use of calloc - this can be ignored as it is not a violation of the rule)	44 S	Use of banned function, type or variable.
MEM04-C	Recommendation	Beware of zero-length allocations			
MEM05-C	Recommendation	Avoid large stack allocations	The analysis highlights use of variable length arrays (VLA) and recursion and possible infinite loops.	5 C	Procedure contains infinite loop.
				6 D	Recursion in procedure calls found.
				28 D	Potentially infinite loop found.
				26 S	Loop control expression may not terminate loop. Modifiers :425 = 0 (Default)
				140 S	Infeasible loop condition found.
				621 S	Variable-length array declared.
				1 U	Inter-file recursion found.
MEM06-C	Recommendation	Ensure that sensitive data is not written out to disk			
MEM07-C	Recommendation	Ensure that the arguments to calloc(), when multiplied, do not wrap			
MEM10-C	Recommendation	Define and use a pointer validation function	The analysis highlights all pointer comparisons with 0 or NULL. The user must then manually check that the comparison occurs within a dedicated pointer validation function.	159 S	Comparing pointer with zero or NULL.
MEM11-C	Recommendation	Do not assume infinite heap space	The analysis highlights use of recursion and possible infinite loops.	5 C	Procedure contains infinite loop.
				6 D	Recursion in procedure calls found.
				28 D	Potentially infinite loop found.
				26 S	Loop control expression may not terminate loop. Modifiers :425 = 0 (Default)
				140 S	Infeasible loop condition found.
				1 U	Inter-file recursion found.
MEM12-C	Recommendation	Consider using a goto chain when leaving a function on error when using and releasing resources	The analysis highlights the root cause of the problem which is that the object might not have been freed	50 D	Memory not freed after last reference.
MEM30-C	Rule	Do not access freed memory		51 D	Attempt to read from freed memory.
				112 D	Free called twice on same variable.
				484 S	Attempt to use already freed object. Modifiers :350 = 1
MEM31-C	Rule	Free dynamically allocated memory when no longer needed		50 D	Memory not freed after last reference.
MEM33-C	Rule	Allocate and copy structures containing a flexible array member dynamically		649 S	Use of unallocated flexible array.
				650 S	Flexible array copy ignores last member.
MEM34-C	Rule	Only free memory allocated dynamically		125 D	free called on variable with no allocated space.
				407 S	free used on string.
				483 S	Freed parameter is not heap item.
				644 S	realloc ptr does not originate from allocation function.
				645 S	realloc ptr type does not match target type.

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
MEM35-C	Rule	Allocate sufficient memory for an object	The analysis highlights all uses of sizeof on a type. The user must then manually check whether the correct type has been chosen.	115 D	Copy length parameter not checked before use.
				400 S	Use of sizeof on a type.
				487 S	Insufficient space allocated.
MEM36-C	Rule	Do not modify the alignment of objects by calling realloc()	The analysis highlights all uses of realloc.	44 S	Use of banned function, type or variable.
MSC00-C	Recommendation	Compile cleanly at high warning levels			
MSC01-C	Recommendation	Strive for logical completeness	The analysis highlights missing paths in the control flow structure. The third example is caught by the rules on possible infinite loops	48 S	No default case in switch statement. Modifiers :252 = 0 (Default) 233 = 0 (Default)
				59 S	Else alternative missing in if. Modifiers :530 = 0 (Default)
MSC04-C	Recommendation	Use comments consistently and in a readable fashion		119 S	Nested comment found. Modifiers :413 = 0 (Default) 434 = 0 (Default)
				302 S	Comment possibly contains code.
				611 S	Line splice used in // comment. Modifiers :402 = 0 (Default)
MSC05-C	Recommendation	Do not manipulate time_t typed values directly	The analysis highlight arithmetic and assigning operations which involve time_t and a different numerical type.	101 S	Function return type inconsistent. Modifiers :374 = 0 (Default) 427 = 1 428 = c : 0 (Default) 430 = 0 (Default) 431 = 1 453 = 1 470 = 0 (Default)
				107 S	Type mismatch in ternary expression. Modifiers :428 = c : 0 (Default) 429 = 1 470 = 0 (Default) 525 = 0 (Default)
				433 S	Type conversion without cast. Modifiers :374 = 0 (Default) 391 = 1 431 = 1 428 = c : 0 (Default) 470 = 0 (Default) 386 = 1 (Default) 549 = 0 (Default)
				458 S	Implicit conversion: actual to formal param (MR).
MSC06-C	Recommendation	Beware of compiler optimizations		65 D	Void function has no side effects.
				76 D	Procedure is not called or referenced in code analysed.
				105 D	DU anomaly dead code, var value is unused on all paths. Modifiers :391 = 1
				151 D	DD data flow anomaly found. Modifiers :391 = 1
				1 J	Unreachable Code found.
				3 J	All internal linkage calls unreachable.
				35 S	Static procedure is not explicitly called in code analysed.
				57 S	Statement with no side effect. Modifiers :458 = 0 (Default)
				631 S	Declaration not reachable.
MSC09-C	Recommendation	Character encoding: use subset of ASCII for safety	This rule applies to naming files, variables, and other objects, which includes the naming of files via strings. Testbed includes a check for strings that contain characters outside the C basic character set. The user must then manually check whether the string is used as a filename. In order to minimise false positives, @ and $ are permitted but can be prohibited by using modifier 408 as described in documentation for 113 S.	113 S	Non standard character in source.
MSC10-C	Recommendation	Character encoding: UTF8-related issues		176 S	Non standard escape sequence in source.
				376 S	Use of octal escape sequence.
MSC11-C	Recommendation	Incorporate diagnostic tests using assertions			

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
MSC12-C	Recommendation	Detect and remove code that has no effect or is never executed		65 D	Void function has no side effects.
				105 D	DU anomaly dead code, var value is unused on all paths. Modifiers :391 = 1
				151 D	DD data flow anomaly found. Modifiers :391 = 1
				57 S	Statement with no side effect. Modifiers :458 = 0 (Default)
				139 S	Construct leads to infeasible code.
				140 S	Infeasible loop condition found.
				1 J	Unreachable Code found.
				631 S	Declaration not reachable.
MSC13-C	Recommendation	Detect and remove unused values	151 D highlights all DD anomalies, which includes variables that are re-assigned down only one path. As this can be noisy in legacy code, it has been classified as Informational and can be switched off.	1 D	Unused procedure parameter.
				15 D	Unused procedural parameter.
				94 D	Named variable declared but not used in code.
				105 D	DU anomaly dead code, var value is unused on all paths. Modifiers :391 = 1
				151 D	DD data flow anomaly found. Modifiers :391 = 1
MSC14-C	Recommendation	Do not introduce unnecessary platform dependencies	Note: Undefined behaviour is covered by recommendation MSC15-C and other more specific rules	17 D	Identifier not unique within *** characters.
				42 S	Use of bit field in structure declaration.
				69 S	#pragma used.
MSC15-C	Recommendation	Do not depend on undefined behavior	The mappings in this rule refer to the undefined features that are not covered by more specific rule.	63 D	No definition in system for prototyped procedure.
				84 D	No fseek or flush before I/O.
				113 D	File closed more than once.
				5 Q	File does not end with new line.
				64 S	Void procedure used in expression. Modifiers :218 = 1
				65 S	Void variable passed as parameter.
				100 S	#include filename is non conformant. Modifiers :120 = '\, (Default)
				109 S	Array subscript is not integral.
				156 S	Use of 'defined' keyword in macro body.
				296 S	Function declared at block scope.
				324 S	Macro call has wrong number of parameters.
				335 S	Operator defined contains illegal items.
				336 S	#if expansion contains define operator.
				339 S	#include directive with illegal items.
				412 S	Undefined behaviour, \ before E-O-F.
				427 S	Filename in #include not in < > or " ".
				465 S	Struct/union not completely specified.
				482 S	Incomplete structure referenced.
				497 S	Type is incomplete in translation unit.
				545 S	Assignment of overlapping storage.
				587 S	Const local variable not immediately initialised.
				608 S	Use of explicitly undefined language feature.
				647 S	Overlapping data items in memcpy.
				62 X	Function prototype/defn return type mismatch (MR).

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
				63 X	Function prototype/defn param type mismatch (MR).
MSC17-C	Recommendation	Finish every set of statements associated with a case label with a break statement		62 S	Switch case not terminated with break. Modifiers :283 = 1 384 = 551 = 0 (Default)
MSC18-C	Recommendation	Be careful while handling sensitive data, such as passwords, in program code			
MSC19-C	Recommendation	For functions that return an array, prefer returning an empty array over a null value			
MSC20-C	Recommendation	Do not use a switch statement to transfer control into a complex block		245 S	Case statement in nested block.
MSC21-C	Recommendation	Use robust loop termination conditions		510 S	Loop counter increment and operator defect.
MSC22-C	Recommendation	Use the setjmp(), longjmp() facility securely	The analysis highlights all uses of setjmp and longjmp. The user must then manually check that the uses comply with this rule.	43 S	Use of setjmp/longjmp.
MSC23-C	Recommendation	Beware of vendor-specific library and language differences			
MSC24-C	Recommendation	Do not use deprecated or obsolescent functions		44 S	Use of banned function, type or variable.
MSC30-C	Rule	Do not use the rand() function for generating pseudorandom numbers	44 S highlights all uses of rand The user must then manually check that the function is used a permitted by the rule.	44 S	Use of banned function, type or variable.
MSC32-C	Rule	Properly seed pseudorandom number generators			
MSC33-C	Rule	Do not pass invalid data to the asctime() function	44 S highlights all uses of asctime. The user must then manually check that the function is used a permitted by the rule	44 S	Use of banned function, type or variable.
MSC37-C	Rule	Ensure that control never reaches the end of a non-void function		2 D	Function does not return a value on all paths.
				36 S	Function has no return statement. Modifiers :553 = 0 (Default)
				66 S	Function with empty return expression.
MSC38-C	Rule	Do not treat a predefined identifier as an object if it might only be implemented as a macro			
MSC39-C	Rule	Do not call va_arg() on a va_list that has an indeterminate value			
MSC40-C	Rule	Do not violate constraints	The analysis should be performed after compilation by the user's compiler. The standards mapped to this rule protect from the following constraint errors, which have been known to go undetected by some compilers. Standard C99 Section Reference 21 S 6.5.2.2 145 S 6.10 323 S 6.8.4.2 345 S 6.5.7, 6.5.10-12 347 S 6.57.2 404 S 6.7.8 481 S 6.7.2.1 580 S 6.10.3 615 S 6.5.15 646 S 6.7.8 Adhering to 612 S avoids both constraint and undefined issues with the use of inline. The user must manually check whether other uses of inline are permissible by these guidelines.	21 S	Number of parameters does not match.
				145 S	#if has invalid expression.
				323 S	Switch has more than one default case.
				345 S	Bit operator with floating point operand.
				387 S	Enum init not integer-constant-expression.
				404 S	Array initialisation has too many items. Modifiers :399 = 1
				481 S	Array with no bounds in struct.
				580 S	Macro redefinition without using #undef.
				612 S	inline function should be declared static.
				615 S	Conditional operator has incompatible types.
				646 S	Struct initialisation has too many items.
POS01-C	Recommendation	Check for the existence of links when dealing with files			
POS02-C	Recommendation	Follow the principle of least privilege			
POS04-C	Recommendation	Avoid using PTHREAD_MUTEX_NORMAL type mutex locks			
POS05-C	Recommendation	Limit access to files by creating a jail			
POS30-C	Rule	Use the readlink() function properly			
POS33-C	Rule	Do not use vfork()		44 S	Use of banned function, type or variable.
POS34-C	Rule	Do not call putenv() with a pointer to an automatic variable as the argument			
POS35-C	Rule	Avoid race conditions while checking for the existence of a symbolic link			
POS36-C	Rule	Observe correct revocation order while relinquishing privileges			
POS37-C	Rule	Ensure that privilege relinquishment is successful			

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
POS38-C	Rule	Beware of race conditions when using fork and file descriptors			
POS39-C	Rule	Use the correct byte ordering when transferring data between systems			
POS44-C	Rule	Do not use signals to terminate threads			
POS47-C	Rule	Do not use threads that can be canceled asynchronously			
POS48-C	Rule	Do not unlock or destroy another POSIX thread's mutex			
POS49-C	Rule	When data must be accessed by multiple threads, provide a mutex and guarantee no adjacent data is also accessed			
POS50-C	Rule	Declare objects shared between POSIX threads with appropriate storage durations			
POS51-C	Rule	Avoid deadlock with POSIX threads by locking in predefined order			
POS52-C	Rule	Do not perform operations that can block while holding a POSIX lock			
POS53-C	Rule	Do not use more than one mutex for concurrent waiting operations on a condition variable			
POS54-C	Rule	Detect and handle POSIX library errors		80 D	Potentially unused function-modified value.
PRE00-C	Recommendation	Prefer inline or static functions to function-like macros	The analysis highlights all uses of function-like macros. The user must then manually check whether they can be replaced by a function without causing a compiler error.	340 S	Use of function like macro.
PRE01-C	Recommendation	Use parentheses within macros around parameter names	The analysis highlights all missing brackets. The user must then manually check whether they are allowed by the exceptions to this rule.	78 S	Macro parameter not in brackets. Modifiers :454 = 0 (Default)
PRE02-C	Recommendation	Macro replacement lists should be parenthesized		77 S	Macro replacement list needs parentheses.
PRE03-C	Recommendation	Prefer typedefs to defines for encoding non-pointer types	The analysis restricts the use of macro to those places permitted by MISRA-C:2004 rule 19.4. The user must then manually filter the messages for those relating to PRE03-C and PRE09-C.	79 S	Macro contains unacceptable items. Modifiers :298 = 0 (Default) 331 = 0 (Default)
PRE04-C	Recommendation	Do not reuse a standard header file name		568 S	#include "filename" uses standard library name. Modifiers :344 = 1
PRE05-C	Recommendation	Understand macro replacement when concatenating tokens or performing stringification	The analysis highlights all uses of # and ##. The user must then manually check that the use is permitted by the rule.	76 S	More than one of # or ## in a macro.
				125 S	Use of ## or # in a macro.
				637 S	# operand followed by ##.
PRE06-C	Recommendation	Enclose header files in an inclusion guard		243 S	Included file not protected with #define.
PRE07-C	Recommendation	Avoid using repeated question marks		81 S	Use of trigraph.
PRE08-C	Recommendation	Guarantee that header file names are unique			
PRE09-C	Recommendation	Do not replace secure functions with deprecated or obsolescent functions			
PRE10-C	Recommendation	Wrap multistatement macros in a do-while loop	The analysis restricts the use of macro to those places permitted by MISRA-C:2004 rule 19.4. The user must then manually filter the messages for those relating to PRE03-C and PRE10-C	79 S	Macro contains unacceptable items. Modifiers :298 = 0 (Default) 331 = 0 (Default)
PRE11-C	Recommendation	Do not conclude macro definitions with a semicolon	Where this leads to additional semi-colons, as in the examples, there will also be violations of MSC12-C and EXP15-C.	699 S	Macro terminated with semi-colon.
PRE12-C	Recommendation	Do not define unsafe macros	The analysis permits repeated evaluation of macro parameters, if those parameters have no side effects	35 D	Expression has side effects.
				1 Q	Call has execution order dependant side effects.
				562 S	Use of ++,-- or = in macro parameters.
				572 S	Side effect in assert.
PRE13-C	Recommendation	Use the Standard predefined macros to test for versions and features.			
PRE30-C	Rule	Do not create a universal character name through concatenation		573 S	Macro concatenation of uni char names.

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
PRE31-C	Rule	Avoid side effects in arguments to unsafe macros		35 D	Expression has side effects.
				1 Q	Call has execution order dependant side effects.
				9 S	Assignment operation in expression. Modifiers :259 = 0 (Default) 266 = 0 (Default)
				562 S	Use of ++,-- or = in macro parameters.
				572 S	Side effect in assert.
PRE32-C	Rule	Do not use preprocessor directives in invocations of function-like macros	The analysis will only detect issues with library functions that are implemented using macros if the analysis scope is set to expand library header files.	341 S	Preprocessor construct as macro parameter.
SIG00-C	Recommendation	Mask signals handled by noninterruptible signal handlers	44 S highlights all uses of signal. The user must then manually check that the function is used a permitted by the rule	44 S	Use of banned function, type or variable.
SIG01-C	Recommendation	Understand implementation-specific details regarding signal handler persistence		97 D	Signal called from within signal handler.
SIG02-C	Recommendation	Avoid using signals to implement normal functionality	44 S highlights all uses of signal. The user must then manually check that the function is used a permitted by the rule.	44 S	Use of banned function, type or variable.
SIG30-C	Rule	Call only asynchronous-safe functions within signal handlers		88 D	Illegal use of longjmp in signal handler.
				89 D	Illegal use of raise in signal handler.
SIG31-C	Rule	Do not access shared objects in signal handlers		87 D	Illegal shared object in signal handler.
SIG34-C	Rule	Do not call signal() from within interruptible signal handlers		97 D	Signal called from within signal handler.
SIG35-C	Rule	Do not return from a computational exception signal handler	44 S highlights all uses of signal. The user must then manually check that the function is used a permitted by the rule	44 S	Use of banned function, type or variable.
STR00-C	Recommendation	Represent characters using an appropriate type		101 S	Function return type inconsistent. Modifiers :374 = 0 (Default) 427 = 1 428 = c : 0 (Default) 430 = 0 (Default) 431 = 1 453 = 1 470 = 0 (Default)
				329 S	Operation not appropriate to plain char. Modifiers :191 = 1 331 = 0 (Default) 391 = 1
				433 S	Type conversion without cast. Modifiers :374 = 0 (Default) 391 = 1 431 = 1 428 = c : 0 (Default) 470 = 0 (Default) 386 = 1 (Default) 549 = 0 (Default)
				458 S	Implicit conversion: actual to formal param (MR).
STR01-C	Recommendation	Adopt and implement a consistent plan for managing strings			
STR02-C	Recommendation	Sanitize data passed to complex subsystems		108 D	Tainted argument to unprototyped func ptr.
				109 D	Tainted argument to formatted i/o function.
STR03-C	Recommendation	Do not inadvertently truncate a null-terminated byte string	The analysis highlights use of strncpy, strncat,fgets and snprintf. Other functions can be added as described in the documentation for 44 S.	44 S	Use of banned function, type or variable.
				115 S	String incorrectly terminated.
STR04-C	Recommendation	Use plain char for characters in the basic character set		101 S	Function return type inconsistent. Modifiers :374 = 0 (Default) 427 = 1 428 = c : 0 (Default) 430 = 0 (Default) 431 = 1 453 = 1 470 = 0 (Default)
				329 S	Operation not appropriate to plain char. Modifiers :191 = 1 331 = 0 (Default) 391 = 1
				433 S	Type conversion without cast. Modifiers :374 = 0 (Default) 391 = 1 431 = 1 428 = c : 0 (Default) 470 = 0 (Default) 386 = 1 (Default) 549 = 0 (Default)
				458 S	Implicit conversion: actual to formal param (MR).
STR05-C	Recommendation	Use pointers to const when referring to string literals		623 S	String assigned to non const object. Modifiers :119 = 0 (Default)

CERT-C:2016 Standards Model Compliance for C

Rule	Classification	Rule Description	Additional Information	LDRA Standard	LDRA Standard Description
STR06-C	Recommendation	Do not assume that strtok() leaves the parse string unchanged	The analysis highlights all uses of strtok with a string parameter. The user must then manually check that the use is permitted by the rule.	602 S	strtok may change the parse string.
STR07-C	Recommendation	Use the bounds-checking interfaces for string manipulation	44 S highlights all uses of strcpy and the other functions listed in rule MSC24-C. The user must then manually check that the function is used a permitted by the rule.	44 S	Use of banned function, type or variable.
STR09-C	Recommendation	Don't assume numeric values for expressions with type plain character		329 S	Operation not appropriate to plain char. Modifiers :191 = 1 331 = 0 (Default) 391 = 1
STR10-C	Recommendation	Do not concatenate different type of string literals		450 S	Wide string and string concatenated.
STR11-C	Recommendation	Do not specify the bound of a character array initialized with a string literal		404 S	Array initialisation has too many items. Modifiers :399 = 1
STR30-C	Rule	Do not attempt to modify string literals		157 S	Modification of string literal. Modifiers :438 = 1
STR31-C	Rule	Guarantee that storage for strings has sufficient space for character data and the null terminator	44S which is mapped to other rules will highlight the use of the deprecated function gets.	109 D	Tainted argument to formatted i/o function.
				140 D	Copy source parameter not checked before use.
				489 S	Insufficient space for operation.
				66 X	Insufficient array space at call.
				70 X	Array has insufficient space.
				71 X	Insufficient space for copy.
STR32-C	Rule	Do not pass a non-null-terminated character sequence to a library function that expects a string	44S which is mapped to other rules will highlight the use of the deprecated function strncpy.	404 S	Array initialisation has too many items. Modifiers :399 = 1
				600 S	Argument of strlen is unterminated.
STR34-C	Rule	Cast characters to unsigned char before converting to larger integer sizes	The analysis will detect the assigning conversion of a object with plain char type to a different type. However it does not check that only an explicit cast to an unsigned type is permitted.	433 S	Type conversion without cast. Modifiers :374 = 0 (Default) 391 = 1 431 = 1 428 = c : 0 (Default) 470 = 0 (Default) 386 = 1 (Default) 549 = 0 (Default)
STR37-C	Rule	Arguments to character-handling functions must be representable as an unsigned char		663 S	Invalid value may be passed to function in <ctype.h>.
STR38-C	Rule	Do not confuse narrow and wide character strings and functions			

General Compliance Notes

Enhanced Enforcement: LDRA checks additional cases to those specified by the mapped rule for enhanced safety and security.
Fully Implemented: LDRA checks all statically checkable aspects of the mapped rule.
Partially Implemented: LDRA checks certain aspects of the rule.

The assessment of whether a rule is fully or partially implemented is based on whether the mapped LDRA standards cover all statically checkable aspects of the rule with a high level of coverage or only cover certain statically checkable aspects of the rule. If a rule is undecidable then this assessment is based on what it is deemed reasonable for a static analysis tool to check.