Polyspace Code Verification

Developer Report for Project: radar\_acc

Report Author: SJ

Polyspace Code Verification: Developer Report for Project: radar\_acc

by Report Author: SJ

Published 20-Jul-2020 15:24:17

Verification Author(s): SJ  
Polyspace Version(s): 9.9 (R2018a)  
Project Version(s): 1.0

Table of Contents

[Chapter 1. Polyspace Code Verification Summary 1](#_Toc46151102)

[Chapter 2. Polyspace Run-Time Checks Statistics 3](#_Toc46151103)

[**Run-Time Checks Summary for radar\_acc - radar\_acc** 3](#_Toc46151104)

[**Percentage of code checked for run-time errors** 3](#_Toc46151105)

[Chapter 3. MISRA C:2012 Guidelines 4](#_Toc46151106)

[**MISRA C:2012 Summary - Violations by File** 4](#_Toc46151107)

[**MISRA C:2012 Violations** 4](#_Toc46151108)

[Chapter 4. Polyspace Run-Time Checks Results 5](#_Toc46151109)

[**Proven Run-Time Violations** 5](#_Toc46151110)

[**Proven Unreachable Code Branches** 5](#_Toc46151111)

[**Unreachable Functions** 6](#_Toc46151112)

[**Unproven Run-Time Checks** 6](#_Toc46151113)

[Chapter 5. Global Variables 7](#_Toc46151114)

[**Variable Checks for: radar\_acc - radar\_acc** 7](#_Toc46151115)

[**Used non-shared variable** 7](#_Toc46151116)

[Chapter 6. Appendix 1 - Configuration Settings 8](#_Toc46151117)

[**Polyspace Settings** 8](#_Toc46151118)

[**Analysis Assumptions** 11](#_Toc46151119)

[**Specified Constraints** 11](#_Toc46151120)

[**Coding Rules Configuration** 11](#_Toc46151121)

[Chapter 7. Appendix 2 - Definitions 20](#_Toc46151122)

Chapter 1. Polyspace Code Verification Summary

Table 1.1. Code Metrics Summary

|  |  |
| --- | --- |
| **Polyspace Code Metrics** | **Disabled** |
| Pass/Fail |  |

Table 1.2. Coding Rules Summary - MISRA:2012 Checker

|  |  |
| --- | --- |
| **MISRA:2012 Checker** | **Enabled** |
| Violations | 0 |
| Pass/Fail |  |

Table 1.3. Run-Time Checks Summary

|  |  |
| --- | --- |
| **Run-Time Checks** | **Enabled** |
| Number of Red Checks | 0 |
| Number of Gray Checks | 4 |
| Number of Orange Checks | 0 |
| Number of Green Checks | 34 |
| Proven | 100.0% |
| Pass/Fail |  |

Table 1.4. Global Variable Summary

|  |  |
| --- | --- |
| **Category** | **Total** |
| Used non-shared variable | 2 |

**Developer Name:  
  
Date Radar\_acciewed:  
  
  
Comments  
  
  
  
  
  
  
  
  
  
Approved By:  
  
Approved Date:**

Chapter 2. Polyspace Run-Time Checks Statistics

Run-Time Checks Summary for radar\_acc - radar\_acc

Globally Proven: 100.0%

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **File** | **Proven** | **Green** | **Red** | **Gray** | **Orange** |
| ACCClassical.c | 100.0% | 32 | 0 | 4 | 0 |
| \_\_polyspace\_main.c | 100.0% | 2 | 0 | 0 | 0 |
| **Total** | **100.0%** | **34** | **0** | **4** | **0** |

Percentage of code checked for run-time errors

|  |  |
| --- | --- |
| **Result Set** | **% Checked** |
| radar\_acc - radar\_acc | 63% |

Chapter 3. MISRA C:2012 Guidelines

MISRA C:2012 Summary - Violations by File

No violations were found.

MISRA C:2012 Violations

No violations were found.

Chapter 4. Polyspace Run-Time Checks Results

Proven Run-Time Violations

No Proven Run-Time Violations checks were found.

Proven Unreachable Code Branches

Table 4.1. C:\Users\Public\Documents\CarSim2019.1\_Data\radar\_acc\_ert\_rtw\ACCClassical.c

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Check** | **Function** | **Line** | **Col** | **Detail** | **Jus** | **Severity** | **Status** | **Comment** |
| 14 | Unreachable code | ACCClassical() | 53 | 9 | The section of code is unreachable or the condition is redundant.  If-condition always evaluates to true at line 51 (column 6).  Block ends at line 56 (column 2)  The block is only partially unreachable. | No | Unset | Unradar\_acciewed |  |
| 39 | Unreachable code | ACCClassical() | 61 | 25 | The section of code is unreachable or the condition is redundant.  If-condition always evaluates to false at line 61 (column 6).  Block ends at line 63 (column 2) | No | Unset | Unradar\_acciewed |  |
| 38 | Unreachable code | ACCClassical() | 64 | 4 | The section of code is unreachable or the condition is redundant.  If-condition always evaluates to true at line 64 (column 8). | No | Unset | Unradar\_acciewed |  |
| 47 | Unreachable code | ACCClassical() | 79 | 25 | The section of code is unreachable or the condition is redundant.  If-condition always evaluates to false at line 79 (column 6).  Block ends at line 81 (column 2) | No | Unset | Unradar\_acciewed |  |

Unreachable Functions

No unreachable functions were found.

Unproven Run-Time Checks

No Unproven Run-Time Checks checks were found.

Chapter 5. Global Variables

Variable Checks for: radar\_acc - radar\_acc

Used non-shared variable

Table 5.1. C:\Users\Public\Documents\CarSim2019.1\_Data\radar\_acc\_ert\_rtw\radar\_acc.c

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Detail** | **Protection** | **Line** | **Col** | **Severity** | **Status** | **Comment** |
| 141 | Operations on variable 'radar\_acc.rtDW' do not interfere with each other. |  | 22 | 3 | Unset | Unradar\_acciewed |  |

Table 5.2. C:\Users\Public\Documents\CarSim2019.1\_Data\radar\_acc\_ert\_rtw\radar\_acc\_data.c

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Detail** | **Protection** | **Line** | **Col** | **Severity** | **Status** | **Comment** |
| 144 | Operations on variable 'radar\_acc\_data.rtConstB' do not interfere with each other. |  | 22 | 13 | Unset | Unradar\_acciewed |  |

Chapter 6. Appendix 1 - Configuration Settings

Polyspace Settings

|  |  |
| --- | --- |
| **Option** | **Value** |
| -align | 64 |
| -allow-negative-operand-in-shift | true |
| -author | SJ |
| -boolean-types | boolean\_T |
| -check-subnormal | allow |
| -compiler | generic |
| -D | main=main\_rtwec,\_\_restrict\_\_=,MODEL=radar\_acc,NUMST=1,NCSTATES=0,HAVESTDIO,MODEL\_HAS\_DYNAMICALLY\_LOADED\_SFCNS=0,CLASSIC\_INTERFACE=0,ALLOCATIONFCN=0,TID01EQ=0,TERMFCN=0,ONESTEPFCN=1,MAT\_FILE=0,MULTI\_INSTANCE\_CODE=0,INTEGER\_CODE=0,MT=0 |
| -data-range-specifications | C:\Users\Public\Documents\CarSim2019.1\_Data\results\_$Prover$\radar\_acc\radar\_acc\_drs.xml |
| -date | 20/07/2020 |
| -dos | true |
| -float-rounding-mode | to-nearest |
| -functions-called-before-loop | radar\_acc\_initialize |
| -functions-called-in-loop | custom=radar\_acc\_step |
| -functions-to-stub | rtIsNaN,rtIsInf,rtIsNaNF,rtIsInfF |
| -I | C:\Users\Public\Documents\CarSim2019.1\_Data\radar\_acc\_ert\_rtw |
| -ignore-constant-overflows | true |
| -import-comments | C:\Users\Public\Documents\CarSim2019.1\_Data\results\_$Prover$\radar\_acc |
| -lang | C |
| -long-is-32bits | true |
| -main-generator | true |
| -mbd | true |
| -misra3 | mandatory |
| -misra3-agc-mode | true |
| -O | -O2 |
| -prog | radar\_acc |
| -report-output-format | Word |
| -report-template | Polyspace-Doc\Developer.rpt |
| -results-dir | C:\Users\Public\Documents\CarSim2019.1\_Data\results\_$Prover$\radar\_acc |
| -scalar-overflows-behavior | wrap-around |
| -scalar-overflows-checks | signed |
| -stub-embedded-coder-lookup-table-functions | true |
| -target | x86\_64 |
| -to | Software Safety Analysis level 2 |
| -uncalled-function-checks | none |
| -variables-written-before-loop | none |
| -variables-written-in-loop | none |
| -verif-version | 1.0 |

Analysis Assumptions

|  |  |
| --- | --- |
| **Assumption** | **Issuer** |
| Nonfinite floats (infinities and NaNs) are not considered | Product |
| Computations involving compile-time constants do not overflow | User |
| Results of floating-point arithmetic are rounded following the IEE754 rule: round to nearest, ties to even | Product |
| Structure fields are not volatile unless the entire structure is volatile-qualified | Product |
| Stack pointers can be safely dereferenced even outside the pointed variable's scope | Product |
| External pointers cannot be null. They point to allocated data of sufficient size for safe dereference | Product |
| Absolute addresses can be safely dereferenced | Product |

Specified Constraints

No modified constraint variables to report.

Coding Rules Configuration

Guidelines prefixed with a "D" are directives.

Table 6.1. MISRA C:2012 Configuration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Guideline** | **Description** | **Mode** | **Comment** | **Enabled** |
| D1.1 | Any implementation-defined behaviour on which the output of the program depends shall be documented and understood. | required |  | no |
| D2.1 | All source files shall compile without any compilation errors. | required |  | no |
| D3.1 | All code shall be traceable to documented requirements. | required | not enforceable | no |
| D4.1 | Run-time failures shall be minimized. | required |  | no |
| D4.2 | All usage of assembly language should be documented. | advisory | not enforceable | no |
| D4.3 | Assembly language shall be encapsulated and isolated. | required |  | no |
| D4.4 | Sections of code should not be "commented out". | advisory | not enforceable | no |
| D4.5 | Identifiers in the same name space with overlapping visibility should be typographically unambiguous. | readability |  | no |
| D4.6 | typedefs that indicate size and signedness should be used in place of the basic numerical types. | advisory |  | no |
| D4.7 | If a function returns error information, then that error information shall be tested. | required | Available in Bug Finder only. | no |
| D4.8 | If a pointer to a structure or union is never dereferenced within a translation unit, then the implementation of the object should be hidden. | advisory |  | no |
| D4.9 | A function should be used in preference to a function-like macro where they are interchangeable. | advisory |  | no |
| D4.10 | Precautions shall be taken in order to pradar\_accent the contents of a header file being included more than once. | required |  | no |
| D4.11 | The validity of values passed to library functions shall be checked. | required |  | no |
| D4.12 | Dynamic memory allocation shall not be used. | required |  | no |
| D4.13 | Functions which are designed to provide operations on a resource should be called in an appropriate sequence. | advisory | Available in Bug Finder only. | no |
| D4.14 | The validity of values received from external sources shall be checked. | required | Available in Bug Finder only. MISRA C:2012 Amendment 1 | no |
| 1.1 | The program shall contain no violations of the standard C syntax and constraints, and shall not exceed the implementation's translation limits. | required |  | no |
| 1.2 | Language extensions should not be used. | advisory |  | no |
| 1.3 | There shall be no occurrence of undefined or critical unspecified behaviour. | required |  | no |
| 2.1 | A project shall not contain unreachable code. | required |  | no |
| 2.2 | There shall be no dead code. | required |  | no |
| 2.3 | A project should not contain unused type declarations. | readability |  | no |
| 2.4 | A project should not contain unused tag declarations. | readability |  | no |
| 2.5 | A project should not contain unused macro declarations. | readability |  | no |
| 2.6 | A function should not contain unused label declarations. | readability |  | no |
| 2.7 | There should be no unused parameters in functions. | readability |  | no |
| 3.1 | The character sequences /\* and // shall not be used within a comment. | required |  | no |
| 3.2 | Line-splicing shall not be used in // comments. | required |  | no |
| 4.1 | Octal and hexadecimal escape sequences shall be terminated. | required |  | no |
| 4.2 | Trigraphs should not be used. | advisory |  | no |
| 5.1 | External identifiers shall be distinct. | required |  | no |
| 5.2 | Identifiers declared in the same scope and name space shall be distinct. | required |  | no |
| 5.3 | An identifier declared in an inner scope shall not hide an identifier declared in an outer scope. | advisory |  | no |
| 5.4 | Macro identifiers shall be distinct. | required |  | no |
| 5.5 | Identifiers shall be distinct from macro names. | required |  | no |
| 5.6 | A typedef name shall be a unique identifier. | required |  | no |
| 5.7 | A tag name shall be a unique identifier. | required |  | no |
| 5.8 | Identifiers that define objects or functions with external linkage shall be unique. | required |  | no |
| 5.9 | Identifiers that define objects or functions with internal linkage should be unique. | readability |  | no |
| 6.1 | Bit-fields shall only be declared with an appropriate type. | required |  | no |
| 6.2 | Single-bit named bit fields shall not be of a signed type. | required |  | no |
| 7.1 | Octal constants shall not be used. | advisory |  | no |
| 7.2 | A "u" or "U" suffix shall be applied to all integer constants that are represented in an unsigned type. | readability |  | no |
| 7.3 | The lowercase character "l" shall not be used in a literal suffix. | readability |  | no |
| 7.4 | A string literal shall not be assigned to an object unless the object's type is "pointer to const-qualified char". | required |  | no |
| 8.1 | Types shall be explicitly specified. | required |  | no |
| 8.2 | Function types shall be in prototype form with named parameters. | required |  | no |
| 8.3 | All declarations of an object or function shall use the same names and type qualifiers. | required |  | no |
| 8.4 | A compatible declaration shall be visible when an object or function with external linkage is defined. | advisory |  | no |
| 8.5 | An external object or function shall be declared once in one and only one file. | advisory |  | no |
| 8.6 | An identifier with external linkage shall have exactly one external definition. | required |  | no |
| 8.7 | Functions and objects should not be defined with external linkage if they are referenced in only one translation unit. | advisory |  | no |
| 8.8 | The static storage class specifier shall be used in all declarations of objects and functions that have internal linkage. | required |  | no |
| 8.9 | An object should be defined at block scope if its identifier only appears in a single function. | advisory |  | no |
| 8.10 | An inline function shall be declared with the static storage class. | required |  | no |
| 8.11 | When an array with external linkage is declared, its size should be explicitly specified. | advisory |  | no |
| 8.12 | Within an enumerator list, the value of an implicitly-specified enumeration constant shall be unique. | required |  | no |
| 8.13 | A pointer should point to a const-qualified type whenever possible. | advisory |  | no |
| 8.14 | The restrict type qualifier shall not be used. | advisory |  | no |
| 9.1 | The value of an object with automatic storage duration shall not be read before it has been set. | mandatory |  | yes |
| 9.2 | The initializer for an aggregate or union shall be enclosed in braces. | readability |  | no |
| 9.3 | Arrays shall not be partially initialized. | readability |  | no |
| 9.4 | An element of an object shall not be initialized more than once. | required |  | no |
| 9.5 | Where designated initializers are used to initialize an array object the size of the array shall be specified explicitly. | readability |  | no |
| 10.1 | Operands shall not be of an inappropriate essential type. | advisory |  | no |
| 10.2 | Expressions of essentially character type shall not be used inappropriately in addition and subtraction operations. | advisory |  | no |
| 10.3 | The value of an expression shall not be assigned to an object with a narrower essential type or of a different essential type category. | advisory |  | no |
| 10.4 | Both operands of an operator in which the usual arithmetic conversions are performed shall have the same essential type category. | advisory |  | no |
| 10.5 | The value of an expression should not be cast to an inappropriate essential type. | advisory |  | no |
| 10.6 | The value of a composite expression shall not be assigned to an object with wider essential type. | advisory |  | no |
| 10.7 | If a composite expression is used as one operand of an operator in which the usual arithmetic conversions are performed then the other operand shall not have wider essential type. | advisory |  | no |
| 10.8 | The value of a composite expression shall not be cast to a different essential type category or a wider essential type. | advisory |  | no |
| 11.1 | Conversions shall not be performed between a pointer to a function and any other type. | required |  | no |
| 11.2 | Conversions shall not be performed between a pointer to an incomplete type and any other type. | required |  | no |
| 11.3 | A cast shall not be performed between a pointer to object type and a pointer to a different object type. | required |  | no |
| 11.4 | A conversion should not be performed between a pointer to object and an integer type. | advisory |  | no |
| 11.5 | A conversion should not be performed from pointer to void into pointer to object. | advisory |  | no |
| 11.6 | A cast shall not be performed between pointer to void and an arithmetic type. | required |  | no |
| 11.7 | A cast shall not be performed between pointer to object and a non-integer arithmetic type. | required |  | no |
| 11.8 | A cast shall not remove any const or volatile qualification from the type pointed to by a pointer. | required |  | no |
| 11.9 | The macro NULL shall be the only permitted form of integer null pointer constant. | readability |  | no |
| 12.1 | The precedence of operators within expressions should be made explicit. | advisory |  | no |
| 12.2 | The right hand operand of a shift operator shall lie in the range zero to one less than the width in bits of the essential type of the left hand operand. | required |  | no |
| 12.3 | The comma operator should not be used | advisory |  | no |
| 12.4 | Evaluation of constant expressions should not lead to unsigned integer wrap-around. | advisory |  | no |
| 12.5 | The sizeof operator shall not have an operand which is a function parameter declared as "array of type". | mandatory | MISRA C:2012 Amendment 1 | yes |
| 13.1 | Initializer lists shall not contain persistent side effects. | required |  | no |
| 13.2 | The value of an expression and its persistent side effects shall be the same under all permitted evaluation orders. | required |  | no |
| 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. | readability |  | no |
| 13.4 | The result of an assignment operator should not be used. | advisory |  | no |
| 13.5 | The right hand operand of a logical && or || operator shall not contain persistent side effects. | required |  | no |
| 13.6 | The operand of the sizeof operator shall not contain any expression which has potential side effects. | mandatory |  | yes |
| 14.1 | A loop counter shall not have essentially floating type. | advisory |  | no |
| 14.2 | A for loop shall be well-formed. | readability |  | no |
| 14.3 | Controlling expressions shall not be invariant. | required |  | no |
| 14.4 | The controlling expression of an if statement and the controlling expression of an iteration-statement shall have essentially Boolean type. | advisory |  | no |
| 15.1 | The goto statement should not be used. | advisory |  | no |
| 15.2 | The goto statement shall jump to a label declared later in the same function. | advisory |  | no |
| 15.3 | Any label referenced by a goto statement shall be declared in the same block, or in any block enclosing the goto statement. | advisory |  | no |
| 15.4 | There should be no more than one break or goto statement used to terminate any iteration statement. | advisory |  | no |
| 15.5 | A function should have a single point of exit at the end. | advisory |  | no |
| 15.6 | The body of an iteration-statement or a selection-statement shall be a compound-statement. | required |  | no |
| 15.7 | All if ... else if constructs shall be terminated with an else statement. | readability |  | no |
| 16.1 | All switch statements shall be well-formed. | advisory |  | no |
| 16.2 | A switch label shall only be used when the most closely-enclosing compound statement is the body of a switch statement. | advisory |  | no |
| 16.3 | An unconditional break statement shall terminate every switch-clause. | advisory |  | no |
| 16.4 | Every switch statement shall have a default label. | advisory |  | no |
| 16.5 | A default label shall appear as either the first or the last switch label of a switch statement. | advisory |  | no |
| 16.6 | Every switch statement shall have at least two switch-clauses. | advisory |  | no |
| 16.7 | A switch-expression shall not have essentially Boolean type. | advisory |  | no |
| 17.1 | The features of <stdarg.h> shall not be used. | required |  | no |
| 17.2 | Functions shall not call themselves, either directly or indirectly. | required |  | no |
| 17.3 | A function shall not be declared implicitly. | mandatory |  | no |
| 17.4 | All exit paths from a function with non-void return type shall have an explicit return statement with an expression. | mandatory |  | yes |
| 17.5 | The function argument corresponding to a parameter declared to have an array type shall have an appropriate number of elements. | readability |  | no |
| 17.6 | The declaration of an array parameter shall not contain the static keyword between the [ ]. | mandatory |  | yes |
| 17.7 | The value returned by a function having non-void return type shall be used. | readability |  | no |
| 17.8 | A function parameter should not be modified. | readability |  | no |
| 18.1 | A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand. | required |  | no |
| 18.2 | Subtraction between pointers shall only be applied to pointers that address elements of the same array. | required |  | no |
| 18.3 | The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object. | required |  | no |
| 18.4 | The +, -, += and -= operators should not be applied to an expression of pointer type. | advisory |  | no |
| 18.5 | Declarations should contain no more than two levels of pointer nesting. | readability |  | no |
| 18.6 | The address of an object with automatic storage shall not be copied to another object that persists after the first object has ceased to exist. | required |  | no |
| 18.7 | Flexible array members shall not be declared. | required |  | no |
| 18.8 | Variable-length array types shall not be used. | required |  | no |
| 19.1 | An object shall not be assigned or copied to an overlapping object. | mandatory |  | yes |
| 19.2 | The union keyword should not be used. | advisory |  | no |
| 20.1 | #include directives should only be preceded by preprocessor directives or comments. | advisory |  | no |
| 20.2 | The ', " or \ characters and the /\* or // character sequences shall not occur in a header file name. | required |  | no |
| 20.3 | The #include directive shall be followed by either a <filename> or "filename"sequence. | required |  | no |
| 20.4 | A macro shall not be defined with the same name as a keyword. | required |  | no |
| 20.5 | #undef should not be used. | readability |  | no |
| 20.6 | Tokens that look like a preprocessing directive shall not occur within a macro argument. | required |  | no |
| 20.7 | Expressions resulting from the expansion of macro parameters shall be enclosed in parentheses. | required |  | no |
| 20.8 | The controlling expression of a #if or #elif preprocessing directive shall evaluate to 0 or 1. | advisory |  | no |
| 20.9 | All identifiers used in the controlling expression of #if or #elif preprocessing directives shall be #define'd before evaluation. | required |  | no |
| 20.10 | The # and ## preprocessor operators should not be used. | advisory |  | no |
| 20.11 | A macro parameter immediately following a # operator shall not immediately be followed by a ## operator. | required |  | no |
| 20.12 | A macro parameter used as an operand to the # or ## operators, which is itself subject to further macro replacement, shall only be used as an operand to these operators. | required |  | no |
| 20.13 | A line whose first token is # shall be a valid preprocessing directive. | required |  | no |
| 20.14 | All #else, #elif and #endif preprocessor directives shall reside in the same file as the #if, #ifdef or #ifndef directive to which they are related. | required |  | no |
| 21.1 | #define and #undef shall not be used on a reserved identifier or reserved macro name. | required |  | no |
| 21.2 | A reserved identifier or macro name shall not be declared. | required |  | no |
| 21.3 | The memory allocation and deallocation functions of <stdlib.h> shall not be used. | required |  | no |
| 21.4 | The standard header file <setjmp.h> shall not be used. | required |  | no |
| 21.5 | The standard header file <signal.h> shall not be used. | required |  | no |
| 21.6 | The Standard Library input/output functions shall not be used. | required |  | no |
| 21.7 | The atof, atoi, atol, and atoll functions of <stdlib.h> shall not be used. | required |  | no |
| 21.8 | The library functions abort, exit and system of <stdlib.h> shall not be used. | required |  | no |
| 21.9 | The library functions bsearch and qsort of <stdlib.h> shall not be used. | required |  | no |
| 21.10 | The Standard Library time and date functions shall not be used. | required |  | no |
| 21.11 | The standard header file <tgmath.h> shall not be used. | required |  | no |
| 21.12 | The exception handling features of <fenv.h> should not be used. | advisory |  | no |
| 21.13 | Any value passed to a function in <ctype.h> shall be representable as an unsigned char or be the value EOF. | mandatory | Available in Bug Finder only. MISRA C:2012 Amendment 1 | no |
| 21.14 | The Standard Library function memcmp shall not be used to compare null terminated strings. | required | Available in Bug Finder only. MISRA C:2012 Amendment 1 | no |
| 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. | required | MISRA C:2012 Amendment 1 | no |
| 21.16 | The pointer arguments to the Standard Library function memcmp shall point to either a pointer type, an essentially signed type, an essentially unsigned type, an essentially Boolean type or an essentially enum type. | required | MISRA C:2012 Amendment 1 | no |
| 21.17 | Use of the string handling functions from <string.h> shall not result in accesses beyond the bounds of the objects referenced by their pointer parameters. | mandatory | Available in Bug Finder only. MISRA C:2012 Amendment 1 | no |
| 21.18 | The size\_t argument passed to any function in <string.h> shall have an appropriate value. | mandatory | Available in Bug Finder only. MISRA C:2012 Amendment 1 | no |
| 21.19 | The pointers returned by the Standard Library functions localeconv, getenv, setlocale or, strerror shall only be used as if they have pointer to const-qualified type. | mandatory | Available in Bug Finder only. MISRA C:2012 Amendment 1 | no |
| 21.20 | The pointer returned by the Standard Library functions asctime, ctime, gmtime, localtime, localeconv, getenv, setlocale or strerror shall not be used following a subsequent call to the same function. | mandatory | Available in Bug Finder only. MISRA C:2012 Amendment 1 | no |
| 22.1 | All resources obtained dynamically by means of Standard Library functions shall be explicitly released. | required | Available in Bug Finder only. | no |
| 22.2 | A block of memory shall only be freed if it was allocated by means of a Standard Library function. | mandatory | Available in Bug Finder only. | no |
| 22.3 | The same file shall not be open for read and write access at the same time on different streams. | required | Available in Bug Finder only. | no |
| 22.4 | There shall be no attempt to write to a stream which has been opened as read-only. | mandatory | Available in Bug Finder only. | no |
| 22.5 | A pointer to a FILE object shall not be dereferenced. | mandatory |  | yes |
| 22.6 | The value of a pointer to a FILE shall not be used after the associated stream has been closed. | mandatory | Available in Bug Finder only. | no |
| 22.7 | The macro EOF shall only be compared with the unmodified return value from any Standard Library function capable of returning EOF. | required | Available in Bug Finder only. MISRA C:2012 Amendment 1 | no |
| 22.8 | The value of errno shall be set to zero prior to a call to an errno-setting-function. | required | Available in Bug Finder only. MISRA C:2012 Amendment 1 | no |
| 22.9 | The value of errno shall be tested against zero after calling an errno-setting-function. | required | Available in Bug Finder only. MISRA C:2012 Amendment 1 | no |
| 22.10 | The value of errno shall only be tested when the last function to be called was an errno-setting-function. | required | Available in Bug Finder only. MISRA C:2012 Amendment 1 | no |

Chapter 7. Appendix 2 - Definitions

Table 7.1. Abbradar\_acciations

|  |  |
| --- | --- |
| **Abbradar\_acciation** | **Definition** |
| Col | Column |
| Jus | Justified |
| SQO | Software Quality Objectives |
| NA | Not Available |