CHAPTER

FIVE

REQUIREMENTS AND DESIGN

Further details for each specification may be found on the RTEMS Classic API Guide [con21c].

5.1 Interface requirements

5.1.1 spec:/req/api

spec:/req/api

The software product shall provide an API.

rationale: N/A

This requirement is refined by the following requirements:

- *spec:/c/if/group*
- *spec:/if/group*
- *spec:/newlib/if/group*
- *spec:/req/applconfig*

5.1.2 spec:/req/applconfig

spec:/req/applconfig

The system shall provide configuration options to the application to set configurable system parameters at link time.

rationale: N/A

This requirement refines *spec:/req/api*.

This requirement is refined by the following requirements:

- *spec:/acfg/if/group-bdbuf*

- *spec:/acfg/if/group-classic*
- *spec:/acfg/if/group-classicinit*
- *spec:/acfg/if/group-devdrv*
- *spec:/acfg/if/group-eventrecord*
- *spec:/acfg/if/group-filesystem*
- *spec:/acfg/if/group-general*
- *spec:/acfg/if/group-idle*
- *spec:/acfg/if/group-mpci*
- *spec:/acfg/if/group-posix*
- *spec:/acfg/if/group-posixinit*
- *spec:/acfg/if/group-schedgeneral*
- *spec:/acfg/if/group-stackalloc*

5.1.3 spec:/rtems/attr/req/bit-set

spec:/rtems/attr/req/bit-set

Each non-default directive attribute constant shall be a power of two representable as an integer of type rtems_attribute.

rationale: N/A

This requirement refines *spec:/rtems/attr/if/group*.

5.1.4 spec:/rtems/attr/req/default

spec:/rtems/attr/req/default

Each default directive attribute constant shall have a value of zero.

rationale: N/A

This requirement refines *spec:/rtems/attr/if/group*.

5.1.5 spec:/rtems/attr/req/default-equals

spec:/rtems/attr/req/default-equals

The value of macro RTEMS_DEFAULT_ATTRIBUTES shall be equal to the value of expression `RTEMS_FIFO | RTEMS_LOCAL`.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/attr/if/default*.

Traced design component: RTEMSAPIClassicAttr - RTEMS_DEFAULT_ATTRIBUTES

5.1.6 spec:/rtems/attr/req/semaphore-class

spec:/rtems/attr/req/semaphore-class

The RTEMS_SEMAPHORE_CLASS constant shall be equal to the bitwise or of RTEMS_BINARY_SEMAPHORE, RTEMS_COUNTING_SEMAPHORE, and RTEMS_SIMPLE_BINARY_SEMAPHORE.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/attr/if/semaphore-class*.

Traced design component: RTEMSAPIClassicAttr - RTEMS_SEMAPHORE_CLASS

5.1.7 spec:/rtems/attr/req/unique

spec:/rtems/attr/req/unique

The non-default directive attribute constants shall have unique values.

rationale: N/A

This requirement refines *spec:/rtems/attr/if/group*.

5.1.8 spec:/rtems/basedefs/req/alias-0

spec:/rtems/basedefs/req/alias-0

When argument `_target` is a name of a function, and the macro RTEMS_ALIAS call is in the same compilation unit as the function, and the macro is not used in block scope, and the macro is used in this form: `<return-type> newname([argument-type-list]) RTEMS_ALIAS(oldname);`, and the `<return-type>` and `argument-type-list` match the signature of the function `oldname`, and the code is compiled with the GNU C compiler, the RTEMS_ALIAS macro shall cause the compiler to create an additional name (`newname` in the syntax) for the function given as argument `_target`.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/alias*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_ALIAS

5.1.9 spec:/rtems/basedefs/req/align-down-0

spec:/rtems/basedefs/req/align-down-0

When the argument `_alignment` is a positive power of two integer, and argument `_value` is a positive or 0 integer, the macro RTEMS_ALIGN_DOWN shall result in a side-effect free formula calculating an integer which is the *greatest* whole-number *multiple* of `_alignment` which is smaller or equal `_value`.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/align-down*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_ALIGN_DOWN

5.1.10 spec:/rtems/basedefs/req/align-up-0

spec:/rtems/basedefs/req/align-up-0

When the argument `_alignment` is a positive power of two integer, and argument `_value` is a positive or 0 integer, the macro RTEMS_ALIGN_UP shall result in a side-effect free formula calculating an integer which is the *smallest* whole-number *multiple* of `_alignment` which is greater or equal `_value`.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/align-up*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_ALIGN_UP

5.1.11 spec:/rtems/basedefs/req/aligned-0

spec:/rtems/basedefs/req/aligned-0

When the argument `_alignment` is a positive power of two integer, and the macro RTEMS_ALIGNED is used on a none-`static` variable or structure field, and the used linker supports alignments of the size given by the `_alignment` argument, and the code is compiled with the GNU C compiler, the macro shall specify a minimum alignment for the variable or structure field, measured in bytes.

rationale: Note that the RTEMS_ALIGNED macro can often only increases the alignment but under some circumstances, it can also decrease the alignment.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/aligned*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_ALIGNED

5.1.12 spec:/rtems/basedefs/req/alignof-0

spec:/rtems/basedefs/req/alignof-0

When the code is compiled with a C compiler and the `__STDC_VERSION__` symbol is defined with version `201112L` or higher or the code is compiled with a C++ compiler and the `__cplusplus` symbol is defined with version `201103L` or higher, and the argument `_type_name` is a type, and the argument `_type_name` is not a function type, and the argument `_type_name` is a complete type, the macro RTEMS_ALIGNOF shall result in the alignment requirement in bytes required for any instance of the type.

rationale: Note that if not `__STDC_VERSION__ >= 201112L` and neither `__cplusplus >= 201103L`, the result of this call may return a value which is not appropriate for alignment.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/alignof*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_ALIGNOF

5.1.13 spec:/rtems/basedefs/req/alignof-1

spec:/rtems/basedefs/req/alignof-1

When the code is compiled with a C compiler and the `__STDC_VERSION__` symbol is defined with version `201112L` or higher or the code is compiled with a C++ compiler and the `__cplusplus` symbol is defined with version `201103L` or higher, the macro RTEMS_ALIGNOF shall not evaluate its argument `_type_name`.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/alignof*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_ALIGNOF

5.1.14 spec:/rtems/basedefs/req/alignof-2

spec:/rtems/basedefs/req/alignof-2

When the code is compiled with a C compiler and the `__STDC_VERSION__` symbol is defined with version `201112L` or higher or the code is compiled with a C++ compiler and the `__cplusplus` symbol is defined with version `201103L` or higher, when the argument `_type_name` is an array type with none constant size expression, the macro RTEMS_ALIGNOF shall not evaluate the size expression.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/alignof*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_ALIGNOF

5.1.15 spec:/rtems/basedefs/req/alignof-3

spec:/rtems/basedefs/req/alignof-3

The macro RTEMS_ALIGNOF shall result in an constant integer of type size_t.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/alignof*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_ALIGNOF

5.1.16 spec:/rtems/basedefs/req/alloc-align-0

spec:/rtems/basedefs/req/alloc-align-0

When the code is compiled with the GNU C compiler, and the RTEMS_ALLOC_ALIGN macro is used as last part of a function declaration, and `_index` is a constant number referring to an argument of that function (counting of arguments starts at 1 from the left), and the argument with that number is an integral value of a power of two, and the declared function returns a pointer to memory which starts at an integral *multiple* of the value provided by the function argument number `_index`, the macro shall cause the compiler to use the information of the alignment of the returned memory in its pointer analysis.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/alloc-align*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_ALLOC_ALIGN

5.1.17 spec:/rtems/basedefs/req/alloc-size-0

spec:/rtems/basedefs/req/alloc-size-0

When the code is compiled with the GNU C compiler, and the RTEMS_ALLOC_SIZE macro is used as last part of a function declaration, and `_index` is a constant number referring to an argument of that function (counting of arguments starts at 1 from the left), and the declared function returns a pointer to memory with the size in bytes provided by the function argument number `_index`, the macro shall cause the compiler to improve the correctness of `__builtin_object_sizepointer` analysis.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/alloc-size*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_ALLOC_SIZE

5.1.18 spec:/rtems/basedefs/req/alloc-size-2-0

spec:/rtems/basedefs/req/alloc-size-2-0

When the code is compiled with the GNU C compiler, and the RTEMS_ALLOC_SIZE_2 macro is used as last part of a function declaration, and `_count_index` as well as `_size_index` are constant numbers referring to two different arguments of that function (counting of arguments starts at 1 from the left), and the declared function returns a pointer to memory with the size in bytes provided by the multiplication of the function arguments number `_count_index` and `_size_index`, the macro shall cause the compiler to improve the correctness of `__builtin_object_sizepointer` analysis.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/alloc-size-2*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_ALLOC_SIZE_2

5.1.19 spec:/rtems/basedefs/req/array-size-0

spec:/rtems/basedefs/req/array-size-0

When the argument `_index` evaluates to an value of a C one-dimensional array type, and the evaluation of that argument has no side effects, the macro RTEMS_ARRAY_SIZE shall result in the number of elements with which that array has been defined.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/array-size*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_ARRAY_SIZE

5.1.20 spec:/rtems/basedefs/req/compiler-deprecated-attribute-0

spec:/rtems/basedefs/req/compiler-deprecated-attribute-0

The macro RTEMS_COMPILER_DEPRECATED_ATTRIBUTE shall have exactly the same effect as the macro RTEMS_DEPRECATED.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/compiler-deprecated-attribute*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_COMPILER_DEPRECATED_ATTRIBUTE

5.1.21 spec:/rtems/basedefs/req/compiler-memory-barrier-0

spec:/rtems/basedefs/req/compiler-memory-barrier-0

When the code is compiled with the GNU C compiler, the RTEMS_COMPILER_MEMORY_BARRIER macro shall realize a Full Software Memory Barrier at the place in the code where it occurs.

rationale: A Full Software Memory Barrier prevents the compiler to move loads and stores (in any direction) beyond the point where the barrier is in the code. Otherwise this may occur as part of compiler optimizations. This is a compile time only barrier. The CPU optimizations can still move instructions over the barrier at run-time.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/compiler-memory-barrier*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_COMPILER_MEMORY_BARRIER

5.1.22 spec:/rtems/basedefs/req/compiler-no-return-attribute-0

spec:/rtems/basedefs/req/compiler-no-return-attribute-0

The macro RTEMS_COMPILER_NO_RETURN_ATTRIBUTE shall have exactly the same effect as the macro RTEMS_NO_RETURN.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/compiler-no-return-attribute*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_COMPILER_NO_RETURN_ATTRIBUTE

5.1.23 spec:/rtems/basedefs/req/compiler-packed-attribute-0

spec:/rtems/basedefs/req/compiler-packed-attribute-0

The macro RTEMS_COMPILER_PACKED_ATTRIBUTE shall have exactly the same effect as the macro RTEMS_PACKED.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/compiler-packed-attribute*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_COMPILER_PACKED_ATTRIBUTE

5.1.24 spec:/rtems/basedefs/req/compiler-pure-attribute-0

spec:/rtems/basedefs/req/compiler-pure-attribute-0

The macro RTEMS_COMPILER_PURE_ATTRIBUTE shall have exactly the same effect as the macro RTEMS_PURE.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/compiler-pure-attribute*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_COMPILER_PURE_ATTRIBUTE

5.1.25 spec:/rtems/basedefs/req/compiler-unused-attribute-0

spec:/rtems/basedefs/req/compiler-unused-attribute-0

The macro RTEMS_COMPILER_UNUSED_ATTRIBUTE shall have exactly the same effect as the macro RTEMS_UNUSED.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/compiler-unused-attribute*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_COMPILER_UNUSED_ATTRIBUTE

5.1.26 spec:/rtems/basedefs/req/concat-0

spec:/rtems/basedefs/req/concat-0

When neither argument is a call of the macro RTEMS_CONCAT itself, the macro shall result in both argument values concatenated textually unaltered in the order they are provided.

rationale: The rules for nested use of the ## operator are arcane. The result of such nested macro calls is undefined.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/concat*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_CONCAT

5.1.27 spec:/rtems/basedefs/req/concat-1

spec:/rtems/basedefs/req/concat-1

The macro RTEMS_CONCAT shall result in only those characters which also appear in its argument values.

rationale: There should be no additional character before, between or after the arguments in the result.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/concat*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_CONCAT

5.1.28 spec:/rtems/basedefs/req/concat-2

spec:/rtems/basedefs/req/concat-2

The macro RTEMS_CONCAT shall make its result subject to C pre-processor operations.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/concat*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_CONCAT

5.1.29 spec:/rtems/basedefs/req/const-0

spec:/rtems/basedefs/req/const-0

When the code is compiled with the GNU C compiler, and the RTEMS_CONST macro is attached to a function declaration or definition, and the return value of that function is not affected by changes to the observable state of the program and that function has no observable effects on such state other than to return a value, the RTEMS_CONST macro shall permit the compiler to replace subsequent calls to the function with the same argument values by the result of the first call.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/const*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_CONST

5.1.30 spec:/rtems/basedefs/req/container-of-0

spec:/rtems/basedefs/req/container-of-0

When argument `_m` points to a member field of a structure or union or C++ class, and argument `_type` is the C type of this structure or union or C++ class, and argument `_member_name` is the name of this member field, the RTEMS_CONTAINER_OF macro shall result in a pointer to the start address of the structure or union or C++ class.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/container-of*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_CONTAINER_OF

5.1.31 spec:/rtems/basedefs/req/declare-global-symbol-0

spec:/rtems/basedefs/req/declare-global-symbol-0

When the macro RTEMS_DECLARE_GLOBAL_SYMBOL appears at file scope, and argument `_name` after undergoing C pre-processor substitutions results in a valid C identifier name, and this identifier name is not yet defined at file scope, the macro RTEMS_DECLARE_GLOBAL_SYMBOL shall apply all possible C pre-processor substitutions to its argument value before it results in code which declares a global symbol with the respective name.

rationale: See also RTEMS_DEFINE_GLOBAL_SYMBOL.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/declare-global-symbol*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_DECLARE_GLOBAL_SYMBOL

5.1.32 spec:/rtems/basedefs/req/deconst-0

spec:/rtems/basedefs/req/deconst-0

When `_type` is a non-const pointer type, and `_var` is a pointer to a value of const type, and the types of `_type` and `_var` are compatible in the sense of C, the macro RTEMS_DECONST shall result in an expression which returns a pointer of type `_type` pointing to the same address as `_var`.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/deconst*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_DECONST

5.1.33 spec:/rtems/basedefs/req/define-global-symbol-0

spec:/rtems/basedefs/req/define-global-symbol-0

When the macro RTEMS_DEFINE_GLOBAL_SYMBOL appears at file scope, and argument `_name` after undergoing C pre-processor substitutions results in a valid C identifier name, and this identifier name is not yet defined at file scope, and argument `_value` after undergoing C pre-processor substitutions results in a valid assembler integer value, the macro RTEMS_DEFINE_GLOBAL_SYMBOL shall apply all possible C pre-processor substitutions to its argument values before it results in assembler code which defines a global symbol with the respective name and value.

rationale: See also RTEMS_DECLARE_GLOBAL_SYMBOL. *File scope* excludes for example a placement in a function body.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/define-global-symbol*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_DEFINE_GLOBAL_SYMBOL

5.1.34 spec:/rtems/basedefs/req/define-global-symbol-1

spec:/rtems/basedefs/req/define-global-symbol-1

The macro RTEMS_DEFINE_GLOBAL_SYMBOL shall define a global symbol of `void` pointer type with the value being an address.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/define-global-symbol*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_DEFINE_GLOBAL_SYMBOL

5.1.35 spec:/rtems/basedefs/req/deprecated-0

spec:/rtems/basedefs/req/deprecated-0

When the code is compiled with the GNU C compiler, and the RTEMS_DEPRECATED macro is used as last part of a function declaration or type declaration or variable declaration or variable definition, the macro shall cause the compiler to issue a warning message when it encounters a use of the function, type or variable.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/deprecated*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_DEPRECATED

5.1.36 spec:/rtems/basedefs/req/dequalify-0

spec:/rtems/basedefs/req/dequalify-0

When `_type` is a non-const non-volatile pointer type, and `_var` is a pointer to a value of const volatile type, and the types of `_type` and `_var` are compatible in the sense of C, the macro RTEMS_DEQUALIFY shall result in an expression which returns a pointer of type `_type` pointing to the same address as `_var`.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/dequalify*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_DEQUALIFY

5.1.37 spec:/rtems/basedefs/req/dequalify-depthx-0

spec:/rtems/basedefs/req/dequalify-depthx-0

When the argument value of `_ptr_level` consists of a sequence of *i* `*` and the types of both other arguments both have *i* nested pointers (for example `*` for a pointer to `int`, `**` for a pointer to a pointer of `int`, `***` for a pointer to a pointer to a pointer to `int`), and `_type` is a pointer type with different (compared to the type of `_var`) qualifiers (such as `const` or `volatile`) or the same qualifiers or without any qualifiers, and the types of `_type` and `_var` are compatible in the sense of C, the macro RTEMS_DEQUALIFY_DEPTHX shall result in an expression which returns a pointer of type `_type` pointing to the same address as `_var`.

rationale: RTEMS_DEQUALIFY_DEPTHX checks for incompatible pointer types.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/dequalify-depthx*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_DEQUALIFY_DEPTHX

5.1.38 spec:/rtems/basedefs/req/devolatile-0

spec:/rtems/basedefs/req/devolatile-0

When `_type` is a non-volatile pointer type, and `_var` is a pointer to a value of volatile type, and the types of `_type` and `_var` are compatible in the sense of C, the macro RTEMS_DEVOLATILE shall result in an expression which returns a pointer of type `_type` pointing to the same address as `_var`.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/devolatile*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_DEVOLATILE

5.1.39 spec:/rtems/basedefs/req/expand-0

spec:/rtems/basedefs/req/expand-0

The macro RTEMS_EXPAND shall apply all possible C pre-processor substitutions to its argument value before it results in the substituted value.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/expand*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_EXPAND

5.1.40 spec:/rtems/basedefs/req/false-0

spec:/rtems/basedefs/req/false-0

The macro FALSE shall result in the text 0.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/false*.

Traced design component: RTEMSAPIBaseDefs - FALSE

5.1.41 spec:/rtems/basedefs/req/have-member-same-type-0

spec:/rtems/basedefs/req/have-member-same-type-0

When the code is compiled with the GNU C compiler, and argument `_t_lhs` is a union or structure, and `_m_lhs` is a member of `_t_lhs`, and argument `_t_rhs` is a union or structure, and `_m_rhs` is a member of `_t_rhs`, the RTEMS_HAVE_MEMBER_SAME_TYPE macro shall evaluate to the integer values 1 or 0 depending on whether the types of the members `_m_lhs` and `_m_rhs` are compatible in the sense of C.

rationale: The RTEMS_HAVE_MEMBER_SAME_TYPE does only work in C. Type qualifiers do not matter (`const int` is compatible to `int`); arrays with undefined length are compatible with arrays of defined length (`int[]` is compatible to `int[5]`); enums are always incompatible; the number of pointer indirection matters (`**int` is not compatible with `*int`). See https://gcc.gnu.org/onlinedocs/gcc/Other-Builtins.html#index-_005f_005fbuiltin_005ftypes_005fcompatible_005fp

This requirement specifies the function of interface *spec:/rtems/basedefs/if/have-member-same-type*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_HAVE_MEMBER_SAME_TYPE

5.1.42 spec:/rtems/basedefs/req/inline-routine-0

spec:/rtems/basedefs/req/inline-routine-0

The RTEMS_INLINE_ROUTINE macro shall evaluate to the keywords `static inline` or `static __inline__` which ever variant is available to the used compiler.

rationale: `inline` and `__inline__` have the same effect at least for the GNU C compiler. `__inline__` works even if the GNU C compiler is invoked with the `-ansi`, or `-std` flags.

The compiler may still emit code for a function defined or declared with `static inline` or `static __inline__`. Therefore, if you want to put an inline function definition into a header file, consider `extern inline` instead (see the compiler documentation).

This requirement specifies the function of interface *spec:/rtems/basedefs/if/inline-routine*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_INLINE_ROUTINE

5.1.43 spec:/rtems/basedefs/req/malloclike-0

spec:/rtems/basedefs/req/malloclike-0

When the code is compiled with the GNU C compiler, and the RTEMS_MALLOCLIKE macro is used as last part of a function declaration or is attached to a function definition, and the function returns a pointer to memory, and this pointer cannot be an alias of any other pointer valid when the function returns, and no pointers to valid objects occur in any storage addressed by that pointer, and the function returns non-NULL in more than 50% of the cases, the macro shall cause the compiler to use this information for optimization.

rationale: Functions like malloc() and calloc() have this property but functions like realloc() do not have this property because the memory it returns may pointer to valid objects.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/malloclike*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_MALLOCLIKE

5.1.44 spec:/rtems/basedefs/req/no-inline-0

spec:/rtems/basedefs/req/no-inline-0

When the code is compiled with the GNU C compiler, and the RTEMS_NO_INLINE macro is used as last part of a function declaration or is attached to a function definition, and the function has side-effects, the macro shall prevent the compiler from inlining this function.

rationale: If the function has no side effects, it may still be subject to inlining. To avoid this, produce an artificial side effect with `asm ("");`.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/no-inline*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_NO_INLINE

5.1.45 spec:/rtems/basedefs/req/no-return-0

spec:/rtems/basedefs/req/no-return-0

When the code is compiled with the GNU C compiler starting at version 2.5 or the `__cplusplus` symbol is defined with version `201103L` or higher or the `__STDC_VERSION__` symbol is defined with version `201112L` or higher, and the RTEMS_NO_RETURN macro is used as

first part of a function declaration or definition, the RTEMS_NO_RETURN macro shall inform the compiler that this function does not return when called.

rationale: The GNU C compiler can optimize such a function without regard to what would happen if it ever did return. Declaring a function RTEMS_NO_RETURN also avoids spurious warnings of uninitialized variables.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/no-return*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_NO_RETURN

5.1.46 spec:/rtems/basedefs/req/noinit-0

spec:/rtems/basedefs/req/noinit-0

When the code is compiled with the GNU C compiler, and the RTEMS_SECTION macro is attached to a global variable definition, and the file format used supports arbitrary sections, the macro shall cause the compiler to store the variable in a section where its value is not initialized during initial start up.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/noinit*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_NOINIT

5.1.47 spec:/rtems/basedefs/req/obfuscate-variable-0

spec:/rtems/basedefs/req/obfuscate-variable-0

When the code is compiled with the GNU C compiler, and argument `_var` is an automatic variable or function argument, and the value of that variable or function argument is of a type which fits into a register, the RTEMS_OBFUSCATE_VARIABLE macro shall prevent the compiler from performing optimizations based on the variable value.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/obfuscate-variable*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_OBFUSCATE_VARIABLE

5.1.48 spec:/rtems/basedefs/req/packed-0

spec:/rtems/basedefs/req/packed-0

When the code is compiled with the GNU C compiler, and the RTEMS_PACKED macro is used as last part of a structure member declaration, and the `aligned` attribute or RTEMS_ALIGNED macro is not used on this structure member, the RTEMS_PACKED macro shall cause the structure member to be aligned at one bit for a bit-field member and one byte otherwise.

rationale: Note: The 4.1, 4.2 and 4.3 series of GCC ignore the packed attribute on bit-fields of type char.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/packed*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_PACKED

5.1.49 spec:/rtems/basedefs/req/packed-1

spec:/rtems/basedefs/req/packed-1

When the code is compiled with the GNU C compiler, and the RTEMS_PACKED macro is attached to a struct, union, or C++ class type definition, and the `aligned` attribute or RTEMS_ALIGNED macro is not used on the struct, union, or C++ class type definition or any member thereof, the RTEMS_PACKED macro shall cause all structure, union, or class members to be aligned at one bit for a bit-field member and one byte otherwise.

rationale: The effect of the RTEMS_PACKED macro is not propagated into any structure, union or C++ class which is member of the structure, union or C++ class declaration to which the macro is attached.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/packed*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_PACKED

5.1.50 spec:/rtems/basedefs/req/packed-2

spec:/rtems/basedefs/req/packed-2

When the code is compiled with the GNU C compiler, and the RTEMS_PACKED macro is attached to a enum type definition, the RTEMS_PACKED macro shall cause the use of the smallest integral type to represent the values of the enum.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/packed*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_PACKED

5.1.51 spec:/rtems/basedefs/req/predict-false-0

spec:/rtems/basedefs/req/predict-false-0

When the code is compiled with the GNU C compiler, and the RTEMS_PREDICT_FALSE macro is used as a conditional in `if`-expressions and loop expressions, and `_exp` after undergoing all possible C pre-processor substitutions is an integral expression, the macro shall cause the compiler to assume that by the percentage of cases defined by `builtin-expect-probability` the expression evaluates to 0.

rationale: Example: `if ( RTEMS_PREDICT_FALSE( -1 == i ) ) { ... }`. The GNU C compiler uses this information for branch optimization. `builtin-expect-probability` defaults to 90%.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/predict-false*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_PREDICT_FALSE

5.1.52 spec:/rtems/basedefs/req/predict-true-0

spec:/rtems/basedefs/req/predict-true-0

When the code is compiled with the GNU C compiler, and the RTEMS_PREDICT_TRUE macro is used as a conditional in `if`-expressions and loop expressions, and `_exp` after undergoing all possible C pre-processor substitutions is an integral expression, the macro shall cause the compiler to assume that by the percentage of cases defined by `builtin-expect-probability` the expression evaluates to 1.

rationale: Example: `if ( RTEMS_PREDICT_TRUE( 99 > i ) ) { ... }`. The GNU C compiler uses this information for branch optimization. `builtin-expect-probability` defaults to 90%. Note the misleading name: The macro tells the compiler to assume “the result is 1” not “the result is not 0” as one would expect for `true`.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/predict-true*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_PREDICT_TRUE

5.1.53 spec:/rtems/basedefs/req/printflike-0

spec:/rtems/basedefs/req/printflike-0

When the code is compiled with the GNU C compiler, and the RTEMS_PRINTFLIKE macro is used as last part of a function declaration or prefixes a function definition, and `_format_pos` as well as _ap_pos are constant numbers referring to two different arguments of that function, and the function argument number `_format_pos` is a `printf`-format string, and the function argument number _ap_pos is the first argument to be used in the `printf`-format string, and

all other arguments used in the `printf`-format string are arguments _ap_pos + 1, _ap_pos + 2, _ap_pos + 3, and so on, the macro shall cause the compiler to use this information for type checking the format string and arguments.

rationale: Counting of arguments starts at 1 from the left with the exception of non-static C++ methods where the counting starts with 2 due to the implicit `this` argument.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/printflike*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_PRINTFLIKE

5.1.54 spec:/rtems/basedefs/req/printflike-1

spec:/rtems/basedefs/req/printflike-1

When the code is compiled with the GNU C compiler, and the RTEMS_PRINTFLIKE macro is used as last part of a function declaration or prefixes a function definition, and `_format_pos` is a constant number referring to an argument of that function, and the function argument number `_format_pos` is a `printf`-format string, and the function argument _ap_pos is 0, the macro shall cause the compiler to use this information for checking the format string.

rationale: This case is for functions where the arguments are not available to be checked (such as `vprintf`). The compiler will only check the format string for consistency.

Counting of arguments starts at 1 from the left with the exception of non-static C++ methods where the counting starts with 2 due to the implicit `this` argument.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/printflike*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_PRINTFLIKE

5.1.55 spec:/rtems/basedefs/req/pure-0

spec:/rtems/basedefs/req/pure-0

When the code is compiled with the GNU C compiler, and the RTEMS_PURE macro is attached to a function declaration or definition, and the function has no observable effects on the state of the program other than to return a value, the RTEMS_PURE macro shall permit the compiler to replace subsequent calls to the function with the same argument values by the result of the first call provided the state of the program observable by that function does not change in between two calls.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/pure*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_PURE

5.1.56 spec:/rtems/basedefs/req/return-address-0

spec:/rtems/basedefs/req/return-address-0

When the code is compiled with the GNU C compiler, the RTEMS_RETURN_ADDRESS macro shall evaluate to the code `__builtin_return_address( 0 )`.

rationale: From the GNU C compiler documentation:

- When inlining the expected behavior is that the function returns the address of the function that is returned to.
- When the top of the stack has been reached, this function returns an unspecified value.
- Additional post-processing of the returned value may be needed, see `__builtin_extract_return_addr`.
- The stored representation of the return address in memory may be different from the address returned by `__builtin_return_address`.

Under these circumstances it is at least difficult to specify what the actual result of this macro is.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/return-address*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_RETURN_ADDRESS

5.1.57 spec:/rtems/basedefs/req/section-0

spec:/rtems/basedefs/req/section-0

When the code is compiled with the GNU C compiler, and the RTEMS_SECTION macro is attached to a function declaration or definition or a global variable definition, and the argument `_section` after applying all possible C pre-processor substitutions to its value is a C sting containing valid linker section name, and the file format used supports arbitary sections, the macro shall cause the compiler to store the function or variable in a section named like the result of the pre-processor substitutions on its argument `_section`.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/section*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_SECTION

5.1.58 spec:/rtems/basedefs/req/static-analysis-0

spec:/rtems/basedefs/req/static-analysis-0

When the macro `__COVERITY__` is defined, the macro RTEMS_STATIC_ANALYSIS shall be defined.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/static-analysis*.

Traced design component: None

5.1.59 spec:/rtems/basedefs/req/static-analysis-1

spec:/rtems/basedefs/req/static-analysis-1

When the macro `__COVERITY__` is not defined, the macro RTEMS_STATIC_ANALYSIS shall be not defined.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/static-analysis*.

Traced design component: None

5.1.60 spec:/rtems/basedefs/req/static-assert-0

spec:/rtems/basedefs/req/static-assert-0

When the argument `_cond` after applying all possible C pre-processor substitutions to its value results in a valid C expression of integral type, and this expression can be evaluated at compile time, and the argument `_msg` which may or may not undergo C pre-processor substitutions results into a valid C identifier, the RTEMS_STATIC_ASSERT macro shall cause the compiler to produce a compilation error if the expression resulting from `_cond` evaluates to 0.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/static-assert*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_STATIC_ASSERT

5.1.61 spec:/rtems/basedefs/req/string-0

spec:/rtems/basedefs/req/string-0

The RTEMS_STRING macro shall result in a string formed by the C pre-processor # operator placed before the formal parameter.

rationale: The exact rules on how this string is build are defined by the C standard and are too complex to be repeated in this requirement.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/string*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_STRING

5.1.62 spec:/rtems/basedefs/req/true-0

spec:/rtems/basedefs/req/true-0

The macro TRUE shall result in the text 1.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/true*.

Traced design component: RTEMSAPIBaseDefs - TRUE

5.1.63 spec:/rtems/basedefs/req/typeof-refx-0

spec:/rtems/basedefs/req/typeof-refx-0

When the argument value of `_level` consists of a sequence of *i* `*` and the type of the other argument has *i* or less than *i* nested pointers (for example `*` for a pointer to `int`, `**` for a pointer to a pointer of `int`, `***` for a pointer to a pointer to a pointer to `int`), and `_target` is either a pointer type (possibly with qualifiers) or an expression of such a pointer type, the macro RTEMS_TYPEOF_REFX shall result in a type expression which is the type of argument `_target` with the given number of pointers removed.

rationale: From the GNU C compiler documentation: The operand of `_target` is evaluated for its side effects if and only if it is an expression of variably modified type or the name of such a type.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/typeof-refx*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_TYPEOF_REFX

5.1.64 spec:/rtems/basedefs/req/unreachable-0

spec:/rtems/basedefs/req/unreachable-0

When the code is compiled with the GNU C compiler, and the RTEMS_UNREACHABLE macro is placed in a part of the code which control flow can under no circumstances ever reach, the macro shall inform the compiler that this place in code cannot be reached.

rationale: The use of this macro will suppress some compiler warnings and may permit some compiler optimizations.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/unreachable*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_UNREACHABLE

5.1.65 spec:/rtems/basedefs/req/unused-0

spec:/rtems/basedefs/req/unused-0

When the code is compiled with the GNU C compiler, and the RTEMS_UNUSED macro is attached to a function definition, the RTEMS_UNUSED macro shall prevent the compiler from emitting a warning if this function is not used.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/unused*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_UNUSED

5.1.66 spec:/rtems/basedefs/req/unused-1

spec:/rtems/basedefs/req/unused-1

When the code is compiled with the GNU C compiler, and the RTEMS_UNUSED macro is appended to a label in this form: `<label>: RTEMS_UNUSED;`, the RTEMS_UNUSED macro shall prevent the compiler from emitting a warning if this label is not used.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/unused*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_UNUSED

5.1.67 spec:/rtems/basedefs/req/unused-2

spec:/rtems/basedefs/req/unused-2

When the code is compiled with the GNU C compiler, and the RTEMS_UNUSED macro is attached to a type (including a union or a struct), the RTEMS_UNUSED macro shall prevent the compiler from emitting a warning if variables of this type are not used.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/unused*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_UNUSED

5.1.68 spec:/rtems/basedefs/req/unused-3

spec:/rtems/basedefs/req/unused-3

When the code is compiled with the GNU C compiler, and the RTEMS_UNUSED macro is attached to a variable definition, the RTEMS_UNUSED macro shall prevent the compiler from emitting a warning if this variable is not used.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/unused*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_UNUSED

5.1.69 spec:/rtems/basedefs/req/used-0

spec:/rtems/basedefs/req/used-0

When the code is compiled with the GNU C compiler, and the RTEMS_USED is macro attached to a function or `static` variable definition, the macro shall cause the compiler to emit the function implementation or variable storage even if there is no reference from C code to the function or variable.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/used*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_USED

5.1.70 spec:/rtems/basedefs/req/warn-unused-result-0

spec:/rtems/basedefs/req/warn-unused-result-0

When the code is compiled with the GNU C compiler, and the RTEMS_WARN_UNUSED_RESULT macro is used as last part of a function declaration or attached to a function definition, and that function has a return type other than `void`, and the returned value is not used, the macro shall cause the compiler to show a compiler warning.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/warn-unused-result*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_WARN_UNUSED_RESULT

5.1.71 spec:/rtems/basedefs/req/weak-0

spec:/rtems/basedefs/req/weak-0

When the code is compiled with the GNU C compiler, and the produced target file format is ELF or a.out, and the RTEMS_WEAK macro is part of a function definition at global scope or variable definition at global scope, and there is no other symbol at global scope with the same name as the one of the above mentioned function or variable, the macro shall have no observable effect.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/weak*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_WEAK

5.1.72 spec:/rtems/basedefs/req/weak-1

spec:/rtems/basedefs/req/weak-1

When the code is compiled with the GNU C compiler, and the produced target file format is ELF or a.out, and the RTEMS_WEAK macro is part of a function definition at global scope or variable definition at global scope, and there is another symbol at global scope with the same name as the above mentioned function or variable, and this other symbol is not defined with the RTEMS_WEAK macro or otherwise defined or declared `weak`, and both functions or variables have the same type, and in case of variables both variables have the same alignment and storage size, the macro shall cause the code to behave as if the function or variable defined with the RTEMS_WEAK macro does not exist.

rationale: The other symbol with the same name can possibly be defined in another compilation unit and linked with the compilation unit containing the function or variable defined with RTEMS_WEAK.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/weak*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_WEAK

5.1.73 spec:/rtems/basedefs/req/weak-alias-0

spec:/rtems/basedefs/req/weak-alias-0

When the code is compiled with the GNU C compiler, and the produced target file format is ELF or a.out, and argument `_target` is a name of a function, and the macro RTEMS_WEAK_ALIAS call is in the same compilation unit as the function, and the macro is not used in block scope, and the macro is used in this form: `<return-type> newname([argument-type-list]) RTEMS_WEAK_ALIAS(oldname);`, and the `<return-type>` and `argument-type-list` match the signature of the function `oldname`, and there is no other function symbol at global scope with the same name as `newname`, the RTEMS_WEAK_ALIAS macro shall cause the compiler to create an additional name (`newname` in the syntax) for the function given as argument `_target`.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/weak-alias*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_WEAK_ALIAS

5.1.74 spec:/rtems/basedefs/req/weak-alias-1

spec:/rtems/basedefs/req/weak-alias-1

When the code is compiled with the GNU C compiler, and the produced target file format is ELF or a.out, and argument `_target` is a name of a function, and the macro RTEMS_WEAK_ALIAS call is in the same compilation unit as the function, and the macro is not used in block scope, and the macro is used in this form: `<return-type> newname([argument-type-list]) RTEMS_WEAK_ALIAS(oldname);`, and the `<return-type>` and `argument-type-list` match the signature of the function `oldname`, and there is another function symbol at global scope with the same name as `newname`, and this other function is not defined with the RTEMS_WEAK macro or otherwise defined or declared `weak`, and both functions have the same type, the RTEMS_WEAK_ALIAS macro shall cause the code to behave as if the function defined with the RTEMS_WEAK_ALIAS macro does not exist.

rationale: The other function at global scope with the same name as `newname` can possibly be defined in another compilation unit and linked with the compilation unit containing the function defined with RTEMS_WEAK_ALIAS.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/weak-alias*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_WEAK_ALIAS

5.1.75 spec:/rtems/basedefs/req/xconcat-0

spec:/rtems/basedefs/req/xconcat-0

The macro RTEMS_XCONCAT shall apply all possible C pre-processor substitutions to its argument values before it concatenates the resulting values.

rationale: All possible C pre-processor substitutions include here calls to the macro itself as well as none if no substitution is possible.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/xconcat*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_XCONCAT

5.1.76 spec:/rtems/basedefs/req/xconcat-1

spec:/rtems/basedefs/req/xconcat-1

The macro RTEMS_XCONCAT shall result in the substituted argument values textually concatenated in the order _x left and _y right.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/xconcat*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_XCONCAT

5.1.77 spec:/rtems/basedefs/req/xconcat-2

spec:/rtems/basedefs/req/xconcat-2

The macro RTEMS_XCONCAT shall result in only those characters which also appear in its argument values after applying all possible C pre-processor substitutions to them.

rationale: There should be no additional character before, between or after the arguments in the result.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/xconcat*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_XCONCAT

5.1.78 spec:/rtems/basedefs/req/xconcat-3

spec:/rtems/basedefs/req/xconcat-3

The macro RTEMS_XCONCAT shall make its result subject to C pre-processor substitutions.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/basedefs/if/xconcat*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_XCONCAT

5.1.79 spec:/rtems/basedefs/req/xstring-0

spec:/rtems/basedefs/req/xstring-0

The macro RTEMS_XSTRING shall apply all possible C pre-processor substitutions to its argument values before the result of this substitution is converted to a a string formed by the C pre-processor # operator and the macro results in this string.

rationale: The exact rules on how this string is build are defined by the C standard and are too complex to be repeated in this requirement.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/xstring*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_XSTRING

5.1.80 spec:/rtems/basedefs/req/zero-length-array-0

spec:/rtems/basedefs/req/zero-length-array-0

When the code is compiled with the GNU C compiler, and the RTEMS_ZERO_LENGTH_ARRAY macro is used as element count of an array declaration, and that array declaration is the last member of a `struct` that is otherwise non-empty, and that structure is never used as member of another structure or array, the macro shall cause the compiler to layout the structure as if the array had an element count of one but to reduce the total size of the structure by the size of that one array element.

rationale: From GNU C documentation:

> Although the size of a zero-length array is zero, an array member of this kind may increase the size of the enclosing type as a result of tail padding.
>
> Example:

```
struct line
{
  int length;
  char contents[RTEMS_ZERO_LENGTH_ARRAY];
};

struct line *thisline = (struct line *)
  malloc (sizeof (struct line) + this_length);
thisline->length = this_length;
```

Zero-length arrays and especially objects ending with zero-length arrays can be statically initialized so that they are larger than declared (have more that 0 elements). See the documentation of the GNU C compiler below keyword: *arrays of length zero*.

This requirement specifies the function of interface *spec:/rtems/basedefs/if/zero-length-array*.

Traced design component: RTEMSAPIBaseDefs - RTEMS_ZERO_LENGTH_ARRAY

5.1.81 spec:/rtems/clock/req/get-ticks-per-second

spec:/rtems/clock/req/get-ticks-per-second

The rtems_clock_get_ticks_per_second function shall return the number of clock ticks per second which is defined indirectly by the CONFIGURE_MICROSECONDS_PER_TICK configuration option..

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/clock/if/get-ticks-per-second*.

Traced design component: RTEMSAPIClassicClock - rtems_clock_get_ticks_per_second

5.1.82 spec:/rtems/clock/req/get-ticks-since-boot

spec:/rtems/clock/req/get-ticks-since-boot

The rtems_clock_get_ticks_since_boot function shall return the number of clock ticks since a point in time during the system initialization or the last overflow of the clock tick counter.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/clock/if/get-ticks-since-boot*.

Traced design component: RTEMSAPIClassicClock - rtems_clock_get_ticks_since_boot

5.1.83 spec:/rtems/message/req/buffer

spec:/rtems/message/req/buffer

When argument `_maximum_message_size` is the size of the largest possible message in bytes (the same value as member maximum_message_size of type rtems_message_queue_config), and `MAXIMUM_PENDING_MESSAGES` is the maximum number of messages which can be stored in the message queue (the same value as member maximum_pending_messages of type rtems_message_queue_config), and `storage_area` is a variable or structure member, the expression `RTEMS_MESSAGE_QUEUE_BUFFER(``_maximum_message_size )`` storage_area[MAXIMUM_PENDING_MESSAGES]` shall declare an object of such a size that a pointer to it is usable as value for member storage_area of type rtems_message_queue_config.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/message/if/buffer*.

Traced design component: RTEMSAPIClassicMessage - RTEMS_MESSAGE_QUEUE_BUFFER

5.1.84 spec:/rtems/mode/req/bit-set

spec:/rtems/mode/req/bit-set

Each non-default task mode constant shall be a power of two representable as an integer of type rtems_mode.

rationale: N/A

This requirement refines *spec:/rtems/mode/if/group*.

5.1.85 spec:/rtems/mode/req/default

spec:/rtems/mode/req/default

Each default task mode constant shall have a value of zero.

rationale: N/A

This requirement refines *spec:/rtems/mode/if/group*.

5.1.86 spec:/rtems/mode/req/masks

spec:/rtems/mode/req/masks

Each task mode mask constant except RTEMS_INTERRUPT_MASK shall be a power of two representable as an integer of type rtems_mode.

rationale: N/A

This requirement refines *spec:/rtems/mode/if/group*.

5.1.87 spec:/rtems/mode/req/masks-all

spec:/rtems/mode/req/masks-all

The bitwise and of a task mode mask constant and RTEMS_ALL_MODE_MASKS shall be equal to the task mode mask constant.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/mode/if/all-mode-masks*.

Traced design component: RTEMSAPIClassicModes - RTEMS_ALL_MODE_MASKS

5.1.88 spec:/rtems/mode/req/masks-unique

spec:/rtems/mode/req/masks-unique

The task mode mask constants and 0xff shall have unique values.

rationale: N/A

This requirement refines *spec:/rtems/mode/if/group*.

5.1.89 spec:/rtems/mode/req/unique

spec:/rtems/mode/req/unique

The non-default task mode constants shall have unique values.

rationale: N/A

This requirement refines *spec:/rtems/mode/if/group*.

5.1.90 spec:/rtems/option/req/bit-set

spec:/rtems/option/req/bit-set

Each non-default directive option constant shall be a power of two representable as an integer of type rtems_option.

rationale: N/A

This requirement refines *spec:/rtems/option/if/group*.

5.1.91 spec:/rtems/option/req/default

spec:/rtems/option/req/default

Each default directive option constant shall have a value of zero.

rationale: N/A

This requirement refines *spec:/rtems/option/if/group*.

5.1.92 spec:/rtems/option/req/default-equals

spec:/rtems/option/req/default-equals

The value of macro RTEMS_DEFAULT_OPTIONS shall be equal to the value of expression `RTEMS_WAIT`.

rationale: N/A

This requirement specifies the function of interface *spec:/rtems/option/if/default*.

Traced design component: RTEMSAPIClassicOptions - RTEMS_DEFAULT_OPTIONS

5.1.93 spec:/rtems/option/req/unique

spec:/rtems/option/req/unique

The non-default directive option constants shall have unique values.

rationale: N/A

This requirement refines *spec:/rtems/option/if/group*.