|  |  |  |
| --- | --- | --- |
| RM | Topic | Effect on SPARK |
| 2.1(5/3) | Graphic representation of characters | N/A |
| 3.9(4/2) | Representation of tag for tagged type in generic body with multiple instantiations. | Ada.Tags not allowed in SPARK, so does it matter if such a tag can never be used or compared with that of another instantiation? |
| 3.9(12.5/3) | Ada.Tags.Interface\_Ancestor\_Tags | OK. Ada.Tags not allowed in SPARK |
| 4.5.2(13) | Equality of access to subprogram values | OK. Not allowed in SPARK. |
| 4.5.2(24.2/1) | Order of calling “=” on components of composite types. | OK. No worries since even a user-defined “=” must be free of side-effects in SPARK. |
| 4.5.5(21) | Rounding of result of “\*” and “/” for ordinary fixed point types | Asked SPARK Team.TN [U304-035]. By default, GNATprove assumes both round-up and round-down might be possible. If using GNAT, then package SPARK.Fixed\_Point\_Arithmetic\_Lemmas can be instantiated and used to provide the specific semantics. |
| 4.6(58.4/4) | Creation of new object (or not) on value conversion | Does it ever matter?  |
| 6.1.1(34/3) | Evaluation order of precondition checks | OK since all are free of side-effects. |
| 6.1.1(35/3) | Evaluation order of postcondition checks | OK since all are free of side-effects. |
| 6.2(11/3) | Parameter passing mechanism | OK, since no of aliasing of names etc. |
| 7.2(5/3) | Order of implicitly null package bodies | No semantic effect as far as I can see. Tucker says this can matter when elaboration order policy is *not* “sequential”, so N/A for SPARK. |
| 7.6(17.4/3) | Result of initializing expression built in place or not? | No semantic effect as far as I can see for SPARK. Can affect number of calls to Adjust and Finalize on Controlled Types, but these are not allowed in SPARK, so N/A. |
| 9.8(14) | Order of Abort statement | OK. Not permitted in SPARK. |
| 9.10(1/3) | Definition of “independently addressable” for shared variables. | OK. Un-protected shared variables are not allowed in SPARK. |
| 10.2(26) | Task creating during termination of env. Task | OK. Not possible in SPARK/Ravenscar. |
| 11.1(6) | Amount of storage needed for any construct | Obviously applies to SPARK, but well mitigated by No\_Allocators restriction and use of GNATStack |
| 11.4.1(10.1/4) | String returned by Ada.Exceptions.Exception\_Message | OK since only useful in the context of a handler part, which are not allowed in SPARK. |
| 11.5(27/2) | Suppression of Range\_Check when Overflow\_Check is suppressed. | OK... mandatory type-safety verification in SPARK. |
| 13.1(18) | Default value of representation aspect. | OK – semantics are supposed to be independent of representation, so we assume compiler-chosen default will be OK. |
| 13.7.2(5/2) | System.Address\_To\_Access\_Conversions | OK. This package cannot be used in SPARK because it declares a general access type. |
| 13.9.1(7) | Does an object really become abnormal after Unsafe Programming? | In normal code, SPARK prevents abnormal values ever being generated. See rules for legality of Unchecked\_Conversion in SPARK. Abnormal value returned from imported foreign code remains possible. |
| 13.11(20) | System.Storage\_Pools | OK. Not allowed in SPARK. |
| 13.11(21.6/3) | System.Storage\_Pools | OK. Not allowed in SPARK. |
| 13.13.2(36/2) | Order of checks for Stream\_IO Read and Input | OK. Streams and Stream\_IO not allowed in SPARK. |
| A.1(1/3) | Body of package Standard | No semantic effect as far as I can see. |
| A.5.1(34) | Does Constraint\_Error get raised by certain functions in Ada.Numerics.Generic\_Elementary\_Functions when Machine\_Overflows = False? | OK. This package has preconditions on these functions to prevent Constraint\_Error, regardless of the value of Machine\_Overflows. |
| A.5.2(28) | Default initial value of Discrete\_Random.Generator | RNG function has side-effect, so cannot be called from SPARK. Initial value is of no interest anyway. |
| A.5.2(34) | Action of Discrete\_Random.Reset | As above – no impact on SPARK really. |
| A.5.3(41.3/2) | Result of S’Machine\_Rounding (X) when X exactly halfway between two integers. | Remains unspecified as per the RM. Onus on user to check that their SPARK Analyzer implements same rounding mode as their compiler and target. (GNATprove and GNAT both round away from zero in this case). |
| A.7(6) | Effect of I/O on access types | OK. How would you even do I/O on an access value without Unsafe Programming (e.g. Unchecked\_Conversion??) |
| A.10(8) | Effect of predefined I/O of “Control Characters” | Generally not applicable to SPARK? |
| A.10.7(8/3) | Value returned from Ada.Text\_IO.Look\_Ahead at end of line or end of file | Generally not applicable to SPARK? |
| A.10.7(12/3) | Value returned by Ada.Text\_IO.Get\_Immediate when no char available. | Generally not applicable to SPARK? |
| A.10.7(17.3/2) | File’s Column\_Number after a failed call to Ada.Text\_IO.Get\_Line function. | Generally not applicable to SPARK? SPARK RM 14.10.1 says that Get\_Line should not be called in SPARK. |
| A.10.7(19) | Value of unassigned characters in result of Ada.Text\_IO.Get\_Line procedure. | Generally not applicable to SPARK? |
| A.14(1) | Effect of shared files | Generally not applicable to SPARK? Options and effects regarding shared-files are well-documented by GNAT, though... |
| A.18.2(231/3) | Effect of Ada.Containers.Vectors.Generic\_Sorting if instantiated with a bad “<” operator | Ada RM Containers libs are not in SPARK. BUT... does generally apply to the SPARK-specific “formal” container libraries. |
| A.18.2(252/2) | Containers... | Ada RM Containers libs are not in SPARK. |
| A.18.2(83/2) | Containers... | As above |
| A.18.3(145/3) | Containers... | As above |
| A.18.3(157/2) | Containers... | As above |
| A.18.3(55/2) | Containers... | As above |
| A.18.4(3/2) | Containers... | As above |
| A.18.4(80/2) | Containers... | As above |
| A.18.5(43/2) | Containers... | As above |
| A.18.5(44/2) | Containers... | As above |
| A.18.5(45/2) | Containers... | As above |
| A.18.5(46/2) | Containers... | As above |
| A.18.6(56/3) | Containers... | As above |
| A.18.6(57/2) | Containers... | As above |
| A.18.7(3/2) | Containers... | As above |
| A.18.7(101/2) | Containers... | As above |
| A.18.7(87/2) | Containers... | As above |
| A.18.7(88/2) | Containers... | As above |
| A.18.8(65/2) | Containers... | As above |
| A.18.8(66.1/3) | Containers... | As above |
| A.18.8(66/2) | Containers... | As above |
| A.18.8(67/2) | Containers... | As above |
| A.18.8(68/2) | Containers... | As above |
| A.18.8(86/2) | Containers... | As above |
| A.18.8(87/2) | Containers... | As above |
| A.18.9(114/2) | Containers... | As above |
| A.18.9(79.1/3) | Containers... | As above |
| A.18.9(79/3) | Containers... | As above |
| A.18.9(80/2) | Containers... | As above |
| A.18.10(227/3) | Containers... | As above |
| A.18.10(72/3) | Containers... | As above |
| A.18.26(5/3) | Containers... | As above |
| A.18.26(9.4/3) | Containers... | As above |
| A.18.26(9/3) | Containers... | As above |
| D.2.2(7.1/2) | Default task dispatching policy | Is always set in SPARK to FIFO\_Within\_Priorities owing to Ravenscar (See RM D.13) |
| D.8(19) | What is the time reference and Epoch for Ada.Real\_Time | OK. No semantic effect? |
| E.3(5/1) | Meaning of “version” of a compilation unit | OK. Distributed Systems Annex not allowed in SPARK. |
| G.1.1(40) | Exceptions from Generic\_Complex\_Types when Machine\_Overflows = False | SPARK assumes Machine\_Overflows = True, but this package really needs pre-conditions on all functions. |
| G.1.2(33) | Exceptions from Generic\_Complex\_Elementary\_Functions when Machine\_Overflows = False | SPARK assumes Machine\_Overflows = True, but this package really needs pre-conditions on all functions. |
| G.1.2(48) | Conditions for overflow from Trig functions in Generic\_Complex\_Elementary\_Functions | This package needs pre-conditions on all functions. The exact condition under which Overflow occurs may be implementation-defined, though. |
| H(4.1) | Documentation of bounded errors unspecified behaviours for high-integrity systems. | N/A |
| H.2(1) | Documentation of bounded errors unspecified behaviours for high-integrity systems. | N/A. Would be nice to have if it were possible. |
| K.2(136.4/2) | Same as A.5.3(41.3/2) | See above. |