OVERVIEW:
FIELD-TEST RELEASE
OF ACVC VERSION 1.11

1 TEST CHANGES

The AMO has made substantive changes in a number of tests. Some of these revisions reflect interpretations of the standard that are different from those used in Version 1.10. Such changes in interpretation were implemented only after much consideration was given to AVO rulings, implementor comments, recommendations from consulting Ada experts, and the AMO's own reading of the standard. In most cases, interpretations being proposed by the Ada Rapporteur Group are in agreement with the positions taken in these tests. The most important of these interpretation changes are described below.

1.1 Required Support

Perhaps the most obvious change for Version 1.11 of the ACVC is in the extension of many of the tests for Chapter 13 objectives. Most of these tests are now ".ADA" tests, subject to the following rule:

ALL IMPLEMENTATIONS ARE EXPECTED TO PASS EVERY ".ADA" TEST.

Exceptions to the above rule are to be handled through the normal dispute resolution process. It is expected that requests for NOT_APPLICABLE grading will require justification in terms of hardware and environment restrictions (see AI-00325).

1.2 Implementation Dependencies

Tests that make use of features that are explicitly left optional in the LRM (predefined numeric types other than INTEGER, FLOAT, and DURATION; package MACHINE_CODE) are left as ".DEP" tests. The same is true of tests that assume some minimum value for SYSTEM.MAX_DIGITS or SYSTEM.MAX_MANTISSA. In addition, tests for certain kinds of representation clauses are left as ".DEP" tests because the difficulty of implementation is thought to outweigh the potential user benefit. Included in this category are tests that require floating point sizes other than those directly supported by the hardware (e.g., "for FLT'Size use Float'Size / 2;"); access type sizes other than the default (assuming an offset representation); and discrete type sizes that require biased representations. Tests raising other issues that have not yet been resolved by the language maintenance authorities have, by and large, been omitted from the test suite.

As has historically been the case, tests with the ".DEP" extension may be graded NOT_APPLICABLE provided that:

. the implementation demonstrates a NOT_APPLICABLE result according to the applicability criteria given in the test comments;

. all other tests having the same applicability criteria exhibit the same behavior; and

. the behavior is consistent with Appendix F of the implementation's documentation.

these conditions are not met, then the test must be passed, subject to the normal dispute resolution process.

1.3 Meaning Of 'Size Attribute And Length Clauses Specifying Size

After careful consideration of implementor comments, the apparent intent of the standard, and the advice of recognized Ada experts, the ACVC Maintenance Organization (AMO) has arrived at a consistent interpretation of the 'SIZE attribute for types and objects, and of representation clauses specifying 'SIZE for a type. This interpretation is given below.

NOTE THAT THIS INTERPRETATION REPRESENTS A REVERSAL OF THE POSITION TAKEN IN SOME TESTS IN VERSION 1.10.

1.3.1 The 'Size Attribute -

For the cases illustrated by the ACVC 1.11 tests, if OB is an object of type T, then the expression "OB'SIZE >= T'SIZE" must evaluate to TRUE (whether or not a size clause has been given for type T). By using pragma PACK or a record representation clause, array or record components having type T may be represented in T'SIZE bits.

1.3.2 The 'Size Representation Clause -

For the cases illustrated by the ACVC 1.11 tests, if the representation clause "FOR T'SIZE USE K;" is accepted, then the expression "T'SIZE = K" must evaluate to TRUE.

1.3.3 Required Acceptance Of 'Size Representation Clauses -

For the cases illustrated by the ACVC 1.11 tests, each implementation must accept size clauses for integer types, enumeration types, and fixed point types provided that every value of the type (or first named subtype) can be mathematically represented in the specified number of bits, using an unsigned representation, if necessary.

1.3.4 Optional Acceptance Of 'Size Representation Clauses -

It is neither expected nor particularly desirable that a representation clause for a floating point type be accepted if it specifies a size other than that of one of the implementation's predefined floating point types. A few ".DEP" tests in ACVC 1.11 give representation clauses requiring floating point sizes that are not likely to agree with the implementation's predefined types.

It is not required that a representation clause for an access type be accepted if it specifies a size other than the default size. A few ".DEP" tests in ACVC 1.11 give representation clauses requiring smaller sizes than the default. In these cases, an offset representation is expected, but it is perfectly acceptable to reject such size clauses.

3.5 Sizes Of Objects -

. Note that, if OB is an object of type T and OB is given by an object declaration, then the expression "OB'SIZE > T'SIZE" may evaluate to TRUE. Likewise, for an object of type T that is a component of an array, ARR, or a record, REC, for whose type no PACK pragma or record representation clause has been given, the expressions "ARR(INDEX)'SIZE > T'SIZE" and "REC.COMP'SIZE > T'SIZE" may evaluate to TRUE. However, if a PACK pragma has been given for the array or record type, or if a record representation clause contains a component clause specifying that COMP is represented in T'SIZE bits, then the expressions "ARR(INDEX)'SIZE = T'SIZE" and "REC.COMP'SIZE = T'SIZE" are expected to yield the value TRUE.

1.4 Effect Of Pragma Pack On Component Type Representation

The following interpretation of pragma PACK is based upon careful consideration of implementor comments, the apparent intent of the standard, and the advice of recognized Ada experts.

NOTE THAT THIS INTERPRETATION REPRESENTS A REVERSAL OF THE POSITION TAKEN BY SOME TESTS IN VERSION 1.10.

If OUTER is an array type or a record type with components of a composite type, INNER, then "PRAGMA PACK (OUTER);" can only remove gaps between the components of OUTER. This pragma is not permitted to affect the representation of the components of type INNER. (ACVC 1.10 tests that expected the pragma to affect component representation have been removed.)

1.5 Use Of Unchecked Conversion

A number of tests use one of two generic procedures to verify that representation clauses are actually obeyed. These two generic procedures (LENGTH_CHECK and ENUM_CHECK) make use of instantiations of UNCHECKED_CONVERSION. LENGTH_CHECK converts a value of the type to a Boolean array of the same size as the type; copies the resulting array to another location; converts the value at the new location to the given type; and compares it to the original value. ENUM_CHECK converts both a given enumeration value and its expected integer code to Boolean arrays of the same size and compares the Boolean arrays. In each case, the compared values are expected to be equal.

1.6 Non-Binary Values Of 'Small

Several tests involving representation clauses that specify 'Small for fixed point types now use values for 'Small that are not powers of two. (Powers of ten are most frequently used.) All implementations are expected to pass the tests requiring 'Small values that are powers of ten; tests requiring 'Small values that are neither powers of two nor powers of ten are categorized as ".DEP" tests.

DELETION OF TESTS

A number of tests that were included in Version 1.10 have been removed from the suite. In some cases, the removal of the test is based upon the opinion that the benefit to users of Ada is not sufficient to justify the extra complexity added to the test suite. In other cases, tests are being withheld until related language issues can be resolved.

The test features leading to the greatest numbers of removals are:

. Representation clauses for derived fixed point types;

. Representation clauses specifying other than the default size for access types (except for a few ".DEP" tests);

. Representation clauses specifying other than the default size for floating point types (except for a few ".DEP" tests); and

. Address clauses for constants, subprograms, packages, and task units.

3 NEW MACRO SUBSTITUTION TESTS

Tests for several areas have been rewritten as parameterized (macro substitution) tests, with extension ".TST". These tests are required of all implementations, unless otherwise indicated by the inclusion of APPLICABILITY CRITERIA in the test header. The substitutions are used to specify values for task storage size clauses, access type size clauses, and record alignment clauses.

4 SUPPORT CHANGES

The AVAT tool and the support package SPPRT13 have been modified, but the AVAT modifications are not complete.

4.1 AVAT Changes

In addition to correcting known errors in some AVAT units, the report utility in the package DATA_COLLECTION has been modified so that the lists of potentially not-applicable tests are not output. The test lists themselves appear in the AVAT units, but are now obsolete. During the field-test period, the test lists will be updated, and the report utility will be restored to its previous (ACVC 1.10) state. In addition, since many of the features tested by the 1.10 version of AVAT are now required, the corresponding units will be deleted from the AVAT system.

4.2 SPPRT13 Changes

The ACVC 1.10 version of the SPPRT13 package required implementors to provide their own versions of the package body. In addition, for those implementations requiring the expression of an 'Address clause to be static, it was necessary to modify the package specification by replacing function specifications with constant declarations.

The ACVC 1.11 version of the SPPRT13 package is somewhat more sophisticated, and many implementors will find it possible to use the package with no modification other than through the traditional macro substitution practice. Thus, the package is now contained in the file "SPPRT13SP.TST", and no package body is needed. The specification declares constants of type System.Address (using the same names as the ACVC 1.10 package, with some deletions and additions) whose values are given by macro substitutions. For example, SPPRT13 includes the declaration "VARIABLE_ADDRESS : CONSTANT

DRESS := $VARIABLE_ADDRESS;". If appropriate literals, constants, or predefined function calls can be used to initialize these constants, then they should be substituted for the macro symbols. Otherwise, the package FCNDECL must be modified (see below).

The specification of the package FCNDECL is contained in the file "FCNDECL.ADA". The package SPPRT13 includes a context clause: "WITH FCNDECL; USE FCNDECL;". The version of FCNDECL supplied with the ACVC is an empty package specification. If appropriate literals, constants, or predefined function calls cannot be used to initialize the constants declared in package SPPRT13, then the implementor must declare appropriate functions in the specification of FCNDECL and provide bodies for them in a package body or by means of a pragma INTERFACE.

5 TEST CHANGES FOR FINAL RELEASE

As previously agreed to by the AVF managers and the AVO, the AMO will make changes, as required, to the test suite during the field-test period. Tests in the field-test release, if found during the field-test period to contain errors, will be removed or corrected. Tests that have become incorrect due to new language interpretations made during the field-test period will either be removed from the suite or changed to reflect the new interpretations. As a general rule, the AMO will opt to remove incorrect tests rather than to correct them to minimize any negative impact on implementors. A test will be modified for the official release only if its need is great and the anticipated negative impact of the change is small. The number of revisions will be held to a minimum, consistent with the goal of producing correct tests.

ACVC tape READ.ME
dated 21-JUN-1989