ACAL Control Number: EDS20001201RSC07-2.4

Ada Conformity Assessment Test Report

Certificate Number: A010516E2.4-007
Rational Software Corporation
Rational Embedded Apex Ada 95/83,
Sun Sparc Solaris to PowerPC family for LynxOS
version 4.0.0b
Sun Ultra 60 under Solaris 2.6 =>
Motorola Powerstack (PowerPC 603) under LynxOS 3.1.0a

(Final)
25 May 2001

Prepared By:
Ada Conformity Assessment Laboratory
EDS Conformance Testing Center
4646 Needmore Road, Bin 46
P.O. Box 24593
Dayton, OH 45424-0593
U.S.A.

TABLE OF CONTENTS

PREFACE

This report documents the conformity assessment of an Ada processor. This assessment was conducted in accordance with the Ada Conformity Assessment Procedures of the Ada Conformity Assessment Laboratory (ACAL) named below and with the Operating Procedures for Ada Conformity Assessments, Version 3.0. The Ada Conformity Assessment Test Suite (ACATS), Version 2.4, was used for testing; the specific version identification is given below.

The successful completion of conformity assessment is the basis for the issuance of a certificate of conformity and for subsequent registration of related processors. A copy of the certificate A010516E2.4-007 which was awarded for this assessment is presented on the following page. Conformity assessment does not ensure that a processor has no nonconformities to the Ada standard other than those, if any, documented in this report. The compiler vendor declares that the tested processor contains no deliberate deviation from the Ada standard; a copy of this Declaration of Conformity is presented immediately after the certificate.

Base Test Suite Version	ACATS 2.4 (VCS label A2_4B) (See Section 2.2 for details)
Location of Testing	Rational Software Corporation Suite 200 1920 N.W Amberglen Parkway Beaverton, Oregon, 97006
Test Completion Date	16 May 2001

This report has been reviewed and approved by the signatories below. These organizations attest that, to the best of their knowledge, this report is accurate and complete; however, they make no warrant, express or implied, that omissions or errors have not occurred.

Ada Conformity Assessment Laboratory
Phil Brashear
EDS Conformance Testing Center
4646 Needmore Road, Bin 46
P.O. Box 24593
Dayton OH 45424-0593
U.S.A.

Ada Conformity Assessment Authority
Randall Brukardt
ACAA
P.O. Box 1512
Madison WI 53701
U.S.A.

(Insert copy of certificate here)

Results Summary for A010516E2.4-007

Specialized Needs Annexes

Note: Tests allocated to these annexes are processed only when the vendor claims support.

SPECIALIZED NEEDS ANNEXES	Total	With-Drawn	Passed	Inapplicable	Unsupported
C Systems Programming	26	0	24	2	0
& required Section 13	13	0	13	0	0
(representation support)	---	---	---	---	---
	39	0	37	2	0
D Real-Time Systems					
(which requires Annex C)	54	0	42	0	12
E Distributed Systems					
	28	0	** NOT TESTED **		
F Information Systems					
	21	0	** NOT TESTED **		
G Numerics	29	0	29	0	0
H Safety and Security	31	0	** NOT TESTED **		

DECLARATION OF CONFORMITY

Customer: Rational Software Corporation

Ada Conformity Assessment Laboratory: EDS Conformance Testing Center
4646 Needmore Road, Bin 46
P.O. Box 24593
Dayton OH 45424-0593
U.S.A.

ACATS Version: 2.4

Ada Processor

Ada Compiler Name and Version: Rational Apex Embedded Ada 95/83,
Sun Sparc Solaris to PowerPC family for LynxOS
version 4.0.0b

Host Computer System: Sun Ultra 60
Solaris 2.6

Target Computer System: Motorola Powerstack (PowerPC 603)
LynxOS 3.1.0a

Declaration

I, the undersigned, declare that I have no knowledge of deliberate deviations from the Ada Language Standard ANSI/ISO/IEC 8652:1995 other than the omission of features as documented in this Conformity Assessment Summary Report.

_____ _____
Customer Signature Date

CHAPTER 1

INTRODUCTION

The Ada processor described above was tested in accordance with the Ada Conformity Assessment Procedures [ACAP] (Language Processor Validation Procedures) of the ACAL and with Version 3.0 of the Operating Procedures of the ACAA [Pro01]. Testing was accomplished using Version 2.4 of the Ada Conformity Assessment Test Suite (ACATS). The ACATS checks the conformity of an Ada processor to the Ada Standard [Ada95].

This Ada Conformity Assessment Test Report (ACATR) gives an account of the testing of this Ada processor. For any technical terms used in this report, the reader is referred to [Pro01]. A detailed description of the ACATS may be found in the ACATS User's Guide [UG01].

1.1 USE OF THIS REPORT

Consistent with the national laws of the originating country, the ACAL and ACAA may make full and free public disclosure of this report. In the United States, this is provided in accordance with the "Freedom of Information Act" (5 U.S.C. #552). Certified status is awarded only to the processor identified in this report. Copies of this report are available to the public from the ACAL that performed this conformity assessment. Copies are also available online at the ACAL's Web site (www.eds-conform.com).

Questions regarding this report or the test results should be directed to the ACAL that performed this conformity assessment or to the Ada Conformity Assessment Authority. For all points of contact, see Appendix B.

1.2 TEST CLASSES

Compliance of Ada processors is tested by means of the ACATS. The ACATS contains a collection of test programs structured into six test classes: A, B, C, D, E, and L. The first letter of a test name identifies the class to which it belongs. Class A, C, D, and E tests are executable. Class B and most Class L tests are expected to produce errors at compile time and link time, respectively.

The executable tests are written in a self-checking manner and produce a PASSED, FAILED, or NOT APPLICABLE message indicating the result when they are executed. Three Ada library units, the packages REPORT and SPPRT13, and the procedure CHECK_FILE are used for this purpose. The package REPORT also provides a set of identity functions used to defeat some compiler optimizations allowed by the Ada Standard that would circumvent a test objective. The package SPPRT13 contains constants of type SYSTEM.ADDRESS. These constants are used by selected Section 13 tests and by isolated tests for other sections. The procedure CHECK_FILE is used to check the contents of text files written by some of the Class C tests for the Input-Output features of the Ada Standard, defined in Annex A of [Ada 95]. The operation of REPORT and CHECK_FILE is checked by a set of executable tests. If these units are not operating correctly, conformity testing is discontinued.

Class B tests check that a compiler detects illegal language usage. Class B tests are not executable. Each test in this class is compiled and the resulting compilation listing is examined to verify that all violations of the Ada Standard are detected. Some of the Class B tests contain legal Ada code which must not be flagged illegal by the compiler. This behavior is also verified.

Class L tests check that an Ada processor correctly detects violation of the Ada Standard involving multiple, separately compiled units. In most Class L tests, errors are expected at link time, and execution must not begin. Other L tests may execute and report the appropriate result.

For some tests of the ACATS, certain implementation-specific values must be supplied. Two insertion methods for the implementation-specific values are used: a macro substitution on the source file level of the test, and linking of a package that contains the implementation-specific values. Details are described in [UG01]. A list of the values used for this processor, along with the specification and body of the package (and children applicable to any of Specialized Needs Annexes being tested) are provided in Section 3.2 of this report.

In addition to these anticipated test modifications, changes may be required to remove unforeseen conflicts between the tests and implementation-dependent characteristics. The modifications required for this processor are described in Section 2.2.

For the conformity assessment of each Ada processor, a customized test suite is produced by the ACAL. This customization consists of making the modifications described in the preceding paragraph, removing withdrawn tests (see Section 3.3), and possibly removing some inapplicable tests (see Section 2.1 and [UG01]).

1.3 DEFINITION OF TERMS

Acceptable result	A result that is explicitly allowed by the grading criteria of the test program for a grade of passed or inapplicable.
Ada compiler	The software and any needed hardware that have to be added to a given host and target computer system to allow transformation of Ada programs into executable form and execution thereof.
Ada Conformity Assessment Test Suite (ACATS)	The means of checking conformity of Ada processors, consisting of tests, support programs, and a User's Guide.
Ada Conformity Assessment Laboratory	An organization that carries out the procedures required to assess the conformity of an Ada processor.
Ada Conformity Assessment Authority (ACAA)	The organization that provides coordination and technical guidance for the Ada Conformity Assessment Laboratories.
Ada Implementation	An Ada processor running on a particular configuration.
Ada Processor	A processor for the Ada programming language as defined in [Ada95].
Certified Status	(Also "certified as conforming") The status granted to an Ada processor by the award of an Ada Conformity Assessment Certificate.
Computer System	A functional unit, consisting of one or more computers and associated software, that uses common storage for all or part of a program and also for all or part of the data necessary for the execution of the program; executes user-written or user-designated programs; performs user-designated data manipulation, including arithmetic operations and logic operations; and that can execute programs that modify themselves during execution. A computer system may be a stand-alone unit or may consist of several inter-connected units.
Configuration	A specific host and target system. "Configuration" is usually used along with "processor" to completely specify a conformity assessment.
Conformity	Fulfillment by a product, process or service of all requirements specified.

Conformity Assessment	The process of checking the conformity of an Ada processor to the Ada programming language and of issing a certificate for that processor.
Customer	An individual or corporate entity who enters into an agreement with an ACAL which specifies the terms and conditions for ACAL services (of any kind) to be performed.
Declaration of Conformity	A formal statement from a customer assuring that conformity is realized or is attainable on the Ada processor for which certified status is realized.
Foundation Unit (Foundation Code)	An Ada package used by multiple tests. Foundation units are designed to be reusable. A valid foundation unit must be in the Ada library for those tests that are dependent on the foundation unit.
Host Computer System	A computer system where Ada source programs are transformed into executable form.
Inapplicable Test	A test that contains one or more test objectives found to be irrelevant for the given Ada processor.
ISO	International Organization for Standardization.
Operating System	Software that controls the execution of programs and that provides services such as resource allocation, scheduling, input/output control, and data management.
Processor	A compiler, translator, or interpreter. The processor includes all tools used in creating programs. For instance, many systems will include a linker in the processor. A processor works in conjunction with, but does not include, a configuration. In this document, processor typically means Ada processor.
Specialized Needs Annex	One of annexes C through H of [Ada95]. Testing of one or more specialized needs annexes is optional, and results for each tested annex are summarized in this report.
Target Computer System	A computer system where the executable form of Ada programs are executed.
Unsupported Feature Test	A test for a language feature that is not required to be supported, because it is based upon a requirement stated in an Ada 95 Specialized Needs Annex.
Withdrawn Test	A test found to be incorrect and not used in conformity testing. A test may be incorrect because it has an invalid test objective, fails to meet its test objective, or contains erroneous or illegal use of the Ada programming language.

CHAPTER 2

IMPLEMENTATION DEPENDENCIES

2.1 INAPPLICABLE TESTS

A test is inapplicable if it contains test objectives which are irrelevant for a given Ada processor. Reasons for a test's inapplicability may be supported by documents issued by the ISO known as Ada Commentaries and commonly referenced in the format AI95-ddddd. For this processor, the following tests were determined to be inapplicable for the reasons indicated; references to Ada Commentaries are included as appropriate.

C45322A, C45523A, and C45622A check that the proper exception is raised if MACHINE_OVERFLOWS is TRUE and the results of various floating-point operations lie outside the range of the base type; for this processor, MACHINE_OVERFLOWS is FALSE.

C45531M..P and C45532M..P (8 tests) check fixed-point operations for types that require a SYSTEM.MAX_MANTISSA of 47 or greater; for this processor, MAX_MANTISSA is less than 47.

C4A012B checks that the proper exception is raised when FLOAT'MACHINE_OVERFLOWS is TRUE for negative powers of 0.0; for this processor, FLOAT'MACHINE_OVERFLOWS is FALSE.

C96005B uses values of type DURATION's base type that are outside the range of type DURATION; for this processor, the ranges are the same.

The tests listed in the following table check that USE_ERROR is raised if the given file operations are not supported for the given combination of mode and access method; this processor supports these operations.

Test	File Operation	Mode	File Access Method
CE2102D	CREATE	IN_FILE	SEQUENTIAL_IO
CE2102E	CREATE	OUT_FILE	SEQUENTIAL_IO
CE2102F	CREATE	INOUT_FILE	DIRECT_IO
CE2102I	CREATE	IN_FILE	DIRECT_IO
CE2102J	CREATE	OUT_FILE	DIRECT_IO
CE2102N	OPEN	IN_FILE	SEQUENTIAL_IO

CE2102O	RESET	IN_FILE	SEQUENTIAL_IO
CE2102P	OPEN	OUT_FILE	SEQUENTIAL_IO
CE2102Q	RESET	OUT_FILE	SEQUENTIAL_IO
CE2102R	OPEN	INOUT_FILE	DIRECT_IO
CE2102S	RESET	INOUT_FILE	DIRECT_IO
CE2102T	OPEN	IN_FILE	DIRECT_IO
CE2102U	RESET	IN_FILE	DIRECT_IO
CE2102V	OPEN	OUT_FILE	DIRECT_IO
CE2102W	RESET	OUT_FILE	DIRECT_IO
CE3102E	CREATE	IN_FILE	TEXT_IO
CE3102F	RESET	Any Mode	TEXT_IO
CE3102G	DELETE	--------	TEXT_IO
CE3102I	CREATE	OUT_FILE	TEXT_IO
CE3102J	OPEN	IN_FILE	TEXT_IO
CE3102K	OPEN	OUT_FILE	TEXT_IO.

CE2203A checks that WRITE raises USE_ERROR if the capacity of an external sequential file is exceeded; this processor cannot restrict file capacity.

CE2403A checks that WRITE raises USE_ERROR if the capacity of an external direct file is exceeded; this processor cannot restrict file capacity.

CE3115A checks operations on text files when multiple internal files are associated with the same external file and one or more are open for writing; USE_ERROR is raised when this association is attempted.

CE3304A checks that SET_LINE_LENGTH and SET_PAGE_LENGTH raise USE_ERROR if they specify an inappropriate value for the external file; there are no inappropriate values for this processor.

CE3413B checks that PAGE raises LAYOUT_ERROR when the value of the page number exceeds COUNT'LAST; for this processor, the value of COUNT'LAST is greater than 150000, making the checking of this objective impractical.

CXB4001..9 (9 tests) depend on the availability of an interface to COBOL; this processor does not support COBOL interfaces.

CXB5001..5 (5 tests) depend upon the availability of an interface to Fortran; this processor does not support Fortran interfaces.

BXC6001 checks that the name referenced in pragmas Atomic and Volatile may only be an object, a non-inherited component or a full type. This test also checks that the name referenced in Atomic_Components or Volatile_Components must be an array type or an object of an anonymous array type. This implementation cannot support the indivisible reads and updates required by the Atomic pragma for type "Atomic_Array".

CXC6001 checks for incorrect usages of atomic and volatile elementary types. This processor does not support indivisible read/update for some types; the application of pragma atomic to a record type in line 65 is

rejected at compile time by this processor.

2.2 MODIFICATIONS

In order to comply with the test objective it may be required to modify the test source code, the test processing method, or the test evaluation method. Modifications are allowable because at the time of test writing not all possible execution environments of the test and the capabilities of the compiler could be foreseen. Possible kinds of modification are:

- o Test Modification: The source code of the test is changed.
  Examples for test modifications are the insertion of a pragma, the insertion of a representation clause, or the splitting of a B-test into several individual tests, if the compiler does not detect all intended errors in the original test.

- o Processing Modification: The processing of the test by the Ada processor for conformity assessment is changed.
  Examples for processing modification are the change of the compilation order for a test that consists of multiple compilations or the additional compilation of a specific support unit in the library.

- o Evaluation Modification: The evaluation of a test result is changed.
  An example for evaluation modification is the grading of a test other than the output from REPORT.RESULT indicates. This may be required if the test makes assumptions about implementation features that are not supported by the processor (e.g., the implementation of a file system on a bare target machine).

All modifications have been directed or approved by the ACAA after consulting the ACAL and the customer on the technical justification of the modification. All of the required test modifications from the "ACATS Modifications List", Version 2.4B were used along with any modifications detailed below.

Modifications were required for 93 tests.

The following 88 tests were split into two or more tests because this processor did not report the violations of the Ada Standard in the way expected by the original tests.

B23002A	B23004A	B23004B	B24001A	B24001B
B24001C	B24005A	B24005B	B24007A	B24009B
B24104A	B24204A	B24204B	B24204C	B24204D
B24204E	B24204F	B24205A	B24206A	B24206B
B25002A	B25002B	B26001A	B26002A	B26005A
B29001A	B2A003A	B2A003B	B2A003C	B2A003D
B2A003E	B2A003F	B2A005A	B2A005B	B2A007A
B2A021A	B32201A	B33101A	B33201B	B35101A
B36002A	B36201A	B37106A	B38003C	B38003D
B38009D	B392002	B393002	B41201A	B44001A
B44004A	B44004B	B44004C	B45205A	B48002A
B48002D	B51001A	B55A01A	B61005A	B67001A

B67001B	B67001C	B67001D	B67001H	B940002
B95001D	B95003A	B95004A	B95007B	B95063A
BA1001D	BC1013A	BC1109A	BC1109B	BC1109C
BC1109D	BC1201A	BC1303F	BC2001D	BC2001E
BC3005B	BC3013A	BC51016	BC51017	BC51018
BD4011A	BXC6001	BXC6003		

The following 5 allowable test modifications from the "ACATS Modifications List", Version 2.4B were used.

C761006	CDB0A02	CXA5011	CXAC002	CXB3010

2.3 UNSUPPORTED FEATURES OF THE ADA 95 SPECIALIZED NEEDS ANNEXES

As allowed by [Ada95], a processor need not support any of the capabilities specified by a Specialized Needs Annex, or it may support some or all of them. For conformity assessment testing, each set of tests for a particular Annex is processed only upon customer request, but is processed in full (even if the Ada processor provides only partial support). As required by [Ada95], the failure to support a requirement of a Specialized Needs Annex must be indicated by a compile-time rejection or by raising a run-time exception. When a test for a Specialized Needs Annex thus indicates non-support, the result is graded "unsupported" (rather than "inapplicable"). However, if such a test is accepted and reports FAILED, the result is graded "failed", and is considered evidence of non-conformity.

The set of tests for each of the following Specialized Needs Annexes was not processed during this conformity assessment testing:

Annex E, Distributed Systems (all BXE* & CXE* files)
Annex F, Information Systems (all BXF* & CXF* files)
Annex H, Safety and Security (all BXH*, CXH*, & LXH* files)

No tests for Annex C, Systems Programming, were graded "unsupported".

The following tests for Annex D, Real-Time Systems, were graded "unsupported".

CXD2001	CXD2002	CXD2003	CXD2004	CXD2006
CXD2007	CXD2008	CXDB001	CXDB002	CXDB003
CXDB004	LXD7008			

No tests for Annex G, Numerics, were graded "unsupported".

CHAPTER 3

PROCESSING INFORMATION

3.1 CONFORMITY ASSESSMENT PROCESS

A full evaluation of the customer's self-tested results was conducted at the ACAL's site.

Witness testing of this Ada processor was conducted at the customer-designated site by a representative of the ACAL.

A CD ROM containing the customized test suite (see Section 1.3) was taken on-site by the ACAL representative for processing. The contents of the CD ROM were loaded directly onto the host computer.

After the test files were loaded onto the host computer, the full set of tests was processed by the Ada processor.

The tests were compiled and linked on the host computer system, as appropriate. The executable images were transferred to the target computer system and run.

Testing was performed using command scripts provided by the customer and reviewed by the ACAL representative. See Appendix A for a complete listing of the processing options for this processor. Appendix A also indicates the default options. The following explicit option settings were used during witness testing:

COMPILER SWITCHES USED WITH NON-DEFAULT SETTINGS

The default values are used for all compiler switches except the following:

OPTIMIZATION_LEVEL
This switch is set to 1 for the embedded run.

OPTIMIZATION_OBJECTIVE
This switch is set to "space" for the embedded run.

-compile X1 X2 X3 ...
This switch is included on the command line. This enables the following actions which would not otherwise occur:

1. Compilation units in each listed file are parsed in order, and if there are duplicate units within the file Xn, the later ones overwrite the earlier ones.

2. If there are syntax errors, a listing file is produced. Syntax corrections that would normally be just applied to the file are instead converted to error messages in the listing file.

3. The compilation units of the file are installed, in an order chosen by the compiler. If there are semantic errors, a list file is produced. The order of the files in the list file will be in compilation order, not the order in which they appear in the original file.

4. If each of the comp units installed successfully, the compiler searches the view for .ada files that have been obsolesced by the newly compiled files. Any xxx.ada files that are obsolesced are renamed xxx.ada.obs.

LINKER SWITCHES USED WITH NON_DEFAULT SETTINGS

RUNTIMES
The default value is used for all Apex native compilers. For Apex cross compilers, the value would be of form,
$APEX_BASE/ada/runtimes/<target_family>.<compiler variant>.<version>/usr.

POSIX_COMPLIANT: FALSE

Test output, compiler and linker listings, and job logs were captured on floppy diskette and archived at the ACAL. The listings examined on-site by the ACAL representative were also archived.

3.2 MACRO PARAMETERS AND IMPLEMENTATION-SPECIFIC VALUES

This section contains the macro parameters used for customizing the ACATS. The meaning and purpose of these parameters are explained in [UG01]. The parameter values are presented in two tables. The first table lists the values that are defined in terms of the maximum input-line length, which is the value for $MAX_IN_LEN, also listed here. These values are expressed in a symbolic notation, using placeholders as appropriate.

3.2.1 Macro Parameters

Macro Parameter	Macro Value
$MAX_IN_LEN	254
$BIG_ID1	AAA ... A1 (254 characters)
$BIG_ID2	AAA ... A2 (254 characters)
$BIG_ID3	AAA ... A3A ... A (254 characters)
$BIG_ID4	AAA ... A4A ... A (254 characters)
$BIG_STRING1	"AAA ... A" ((254+1)/2 characters)
$BIG_STRING2	"AAA ... A1" (254/2 characters)
$BLANKS	" ... " (254-20 blanks)
$MAX_STRING_LITERAL	"AAA ... A" (254 characters)
$ACC_SIZE	32
$ALIGNMENT	1
$COUNT_LAST	1_000_000_000
$ENTRY_ADDRESS	SYSTEM.STORAGE_ELEMENTS.TO_ADDRESS(30)
$ENTRY_ADDRESS1	SYSTEM.STORAGE_ELEMENTS.TO_ADDRESS(31)
$ENTRY_ADDRESS2	SYSTEM.STORAGE_ELEMENTS.TO_ADDRESS(2)
$FIELD_LAST	2_147_483_647
$FORM_STRING	""
$FORM_STRING2	"CANNOT_RESTRICT_FILE_CAPACITY"
$GREATER_THAN_DURATION	1.0

PROCESSING INFORMATION

$ILLEGAL_EXTERNAL_FILE_NAME1	BAD/_CHARACTERS
$ILLEGAL_EXTERNAL_FILE_NAME2	CONTAINS/_WILDCARDS
$INAPPROPRIATE_LINE_LENGTH	-1
$INAPPROPRIATE_PAGE_LENGTH	-1
$INTEGER_FIRST	-2147483648
$INTEGER_LAST	2147483647
$LESS_THAN_DURATION	-1.0
$MACHINE_CODE_STATEMENT	CODE_0'(OP => NOP);
$MAX_INT	2147483647
$MIN_INT	-2147483648
$NAME	SHORT_SHORT_INTEGER
$NAME_SPECIFICATION1	
$NAME_SPECIFICATION2	
$NAME_SPECIFICATION3	
$OPTIONAL_DISC	(OP : OPCODE)
$RECORD_DEFINITION	RECORD OPRND_1 : OPERAND; END RECORD;
$RECORD_NAME	CODE_0
$TASK_SIZE	32
$TASK_STORAGE_SIZE	8192
$VARIABLE_ADDRESS	FCNDECL.ADDRESS0
$VARIABLE_ADDRESS1	FCNDECL.ADDRESS1
$VARIABLE_ADDRESS2	FCNDECL.ADDRESS2

3.2.2 Package ImpDef and Its Children

The package ImpDef is used by several tests of core language features. Before use in testing, this package is modified to specify certain implementation-defined features. In addition, package ImpDef has a child package for each Specialized Needs Annex, each of which may need similar modifications. The child packages are independent of one another, and are used only by tests for their respective annexes.

This section presents the package ImpDef and each of the relevant child packages as they were modified for this conformity assessment. In the interests of simplifying this ACATR, the header comment block was removed from each of the package files.

3.2.2.1 Package ImpDef

```
-- IMPDEF.A
--
--                              Grant of Unlimited Rights
--
--     Under contracts F33600-87-D-0337, F33600-84-D-0280, MDA903-79-C-0687,
--     F08630-91-C-0015, and DCA100-97-D-0025, the U.S. Government obtained
--     unlimited rights in the software and documentation contained herein.
--     Unlimited rights are defined in DFAR 252.227-7013(a)(19).  By making
--     this public release, the Government intends to confer upon all
--     recipients unlimited rights  equal to those held by the Government.
--     These rights include rights to use, duplicate, release or disclose the
--     released technical data and computer software in whole or in part, in
--     any manner and for any purpose whatsoever, and to have or permit others
--     to do so.
--
--                                    DISCLAIMER
--
--     ALL MATERIALS OR INFORMATION HEREIN RELEASED, MADE AVAILABLE OR
--     DISCLOSED ARE AS IS.  THE GOVERNMENT MAKES NO EXPRESS OR IMPLIED
--     WARRANTY AS TO ANY MATTER WHATSOEVER, INCLUDING THE CONDITIONS OF THE
--     SOFTWARE, DOCUMENTATION OR OTHER INFORMATION RELEASED, MADE AVAILABLE
--     OR DISCLOSED, OR THE OWNERSHIP, MERCHANTABILITY, OR FITNESS FOR A
--     PARTICULAR PURPOSE OF SAID MATERIAL.
--*
--
-- DESCRIPTION:
--     This package provides tailorable entities for a particular
--     implementation.  Each entity may be modified to suit the needs
--     of the implementation.  Default values are provided to act as
--     a guide.
--
--     The entities in this package are those which are used in at least
--     one core test. Entities which are used exclusively in tests for
--     annexes C-H are located in annex-specific child units of this package.
--
```

```
-- CHANGE HISTORY:
--     12 DEC 93   SAIC    Initial PreRelease version
--     02 DEC 94   SAIC    Second  PreRelease version
--     16 May 95   SAIC    Added constants specific to tests of the random
--                         number generator.
--     16 May 95   SAIC    Added Max_RPC_Call_Time constant.
--     17 Jul 95   SAIC    Added Non_State_String constant.
--     21 Aug 95   SAIC    Created from existing IMPSPEC.ADA and IMPBODY.ADA
--                         files.
--     30 Oct 95   SAIC    Added external name string constants.
--     24 Jan 96   SAIC    Added alignment constants.
--     29 Jan 96   SAIC    Moved entities not used in core tests into annex-
--                         specific child packages. Adjusted commentary.
--                         Renamed Validating_System_Programming_Annex to
--                         Validating_Annex_C. Added similar Validating_Annex_?
--                         constants for the other non-core annexes (D-H).
--     01 Mar 96   SAIC    Added external name string constants.
--     21 Mar 96   SAIC    Added external name string constants.
--     02 May 96   SAIC    Removed constants for draft test CXA5014, which was
--                         removed from the tentative ACVC 2.1 suite.
--                         Added constants for use with FXACA00.
--     06 Jun 96   SAIC    Added constants for wide character test files.
--     11 Dec 96   SAIC    Updated constants for wide character test files.
--     13 Dec 96   SAIC    Added Address_Value_IO
--     13 Sep 99   RLB     Added more external name string constants.
--     16 Sep 99   RLB     Corrected definition of Non_State_String constant.
--
--!

with Report;
with Ada.Text_IO;
with System.Storage_Elements;

package ImpDef is

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The following boolean constants indicate whether this validation will
   -- include any of annexes C-H. The values of these booleans affect the
   -- behavior of the test result reporting software.
   --
   --    True  means the associated annex IS included in the validation.
   --    False means the associated annex is NOT included.

   Validating_Annex_C : constant Boolean := True;
   --                                       ^^^^^ --- MODIFY HERE AS NEEDED

   Validating_Annex_D : constant Boolean := True;
   --                                       ^^^^^ --- MODIFY HERE AS NEEDED

   Validating_Annex_E : constant Boolean := False;
   --                                       ^^^^^ --- MODIFY HERE AS NEEDED

   Validating_Annex_F : constant Boolean := False;
```

```
   --                                          ^^^^^ --- MODIFY HERE AS NEEDED

   Validating_Annex_G : constant Boolean := True;
   --                                          ^^^^^ --- MODIFY HERE AS NEEDED

   Validating_Annex_H : constant Boolean := False;
   --                                          ^^^^^ --- MODIFY HERE AS NEEDED

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- This is the minimum time required to allow another task to get
   -- control.  It is expected that the task is on the Ready queue.
   -- A duration of 0.0 would normally be sufficient but some number
   -- greater than that is expected.

   Minimum_Task_Switch : constant Duration := 0.1;
   --                                            ^^^ --- MODIFY HERE AS NEEDED

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- This is the time required to activate another task and allow it
   -- to run to its first accept statement.  We are considering a simple task
   -- with very few Ada statements before the accept.  An implementation is
   -- free to specify a delay of several seconds, or even minutes if need be.
   -- The main effect of specifying a longer delay than necessary will be an
   -- extension of the time needed to run the associated tests.

   Switch_To_New_Task : constant Duration := 1.0;
   --                                            ^^^ -- MODIFY HERE AS NEEDED

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- This is the time which will clear the queues of other tasks
   -- waiting to run.  It is expected that this will be about five
   -- times greater than Switch_To_New_Task.

   Clear_Ready_Queue : constant Duration := 5.0;
   --                                          ^^^ --- MODIFY HERE AS NEEDED

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- Some implementations will boot with the time set to 1901/1/1/0.0
   -- When a delay of Delay_For_Time_Past is given, the implementation
   -- guarantees that a subsequent call to Ada.Calendar.Time_Of(1901,1,1)
   -- will yield a time that has already passed (for example, when used in
   -- a delay_until statement).

   Delay_For_Time_Past : constant Duration := 0.1;
   --                                            ^^^ --- MODIFY HERE AS NEEDED

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- Minimum time interval between calls to the time dependent Reset
   -- procedures in Float_Random and Discrete_Random packages that is
```

```
   -- guaranteed to initiate different sequences.  See RM A.5.2(45).

   Time_Dependent_Reset : constant Duration := 0.3;
   --                                          ^^^ --- MODIFY HERE AS NEEDED

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- Test CXA5013 will loop, trying to generate the required sequence
   -- of random numbers.  If the RNG is faulty, the required sequence
   -- will never be generated.  Delay_Per_Random_Test is a time-out value
   -- which allows the test to run for a period of time after which the
   -- test is failed if the required sequence has not been produced.
   -- This value should be the time allowed for the test to run before it
   -- times out.  It should be long enough to allow multiple (independent)
   -- runs of the testing code, each generating up to 1000 random
   -- numbers.

   Delay_Per_Random_Test : constant Duration := 1.0;
   --                                           ^^^ --- MODIFY HERE AS NEEDED

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The time required to execute this procedure must be greater than the
   -- time slice unit on implementations which use time slicing.  For
   -- implementations which do not use time slicing the body can be null.

   procedure Exceed_Time_Slice;

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- This constant must not depict a random number generator state value.
   -- Using this string in a call to function Value from either the
   -- Discrete_Random or Float_Random packages will result in
   -- Constraint_Error or Program_Error (expected result in test CXA5012).
   -- If there is no such string, set it to "**NONE**".

   Non_State_String : constant String := "By No Means A State";
   --           MODIFY HERE AS NEEDED --- ^^^^^^^^^^^^^^^^^^^

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- This string constant must be a legal external tag value as used by
   -- CD10001 for the type Some_Tagged_Type in the representation
   -- specification for the value of 'External_Tag.

   External_Tag_Value : constant String := "implementation_defined";
   --             MODIFY HERE AS NEEDED --- ^^^^^^^^^^^^^^^^^^^^^

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The following address constant must be a valid address to locate
   -- the C program CD30005_1.  It is shown here as a named number;
   -- the implementation may choose to type the constant as appropriate.
   procedure Cd30005_C_Support;
```

```
   pragma Import (C, Cd30005_C_Support, External_Name => "_cd30005_1");
   pragma Link_With ("cd300051.o");

   CD30005_1_Foreign_Address : constant System.Address:=
                  Cd30005_C_Support'Address;

   --             System.Storage_Elements.To_Address ( 16#0000_0000# );
   --                    MODIFY HERE AS REQUIRED --- ^^^^^^^^^^^^^

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The following string constant must be the external name resulting
   -- from the C compilation of CD30005_1.  The string will be used as an
   -- argument to pragma Import.

   CD30005_1_External_Name : constant String := "CD30005_1";
   --                   MODIFY HERE AS NEEDED --- ^^^^^^^^^

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The following constants should represent the largest default alignment
   -- value and the largest alignment value supported by the linker.
   -- See RM 13.3(35).

   Max_Default_Alignment : constant := 4;
   --                                  ^ --- MODIFY HERE AS NEEDED

   Max_Linker_Alignment  : constant := 4;
   --                                  ^ --- MODIFY HERE AS NEEDED

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The following string constants must be the external names resulting
   -- from the C compilation of CXB30040.C, CXB30060.C, CXB30130.C, and
   -- CXB30131.C.  The strings will be used as arguments to pragma Import.

   CXB30040_External_Name : constant String := "CXB30040";
   --                   MODIFY HERE AS NEEDED --- ^^^^^^^^

   CXB30060_External_Name : constant String := "CXB30060";
   --                   MODIFY HERE AS NEEDED --- ^^^^^^^^

   CXB30130_External_Name : constant String := "CXB30130";
   --                   MODIFY HERE AS NEEDED --- ^^^^^^^^

   CXB30131_External_Name : constant String := "CXB30131";
   --                   MODIFY HERE AS NEEDED --- ^^^^^^^^

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The following string constants must be the external names resulting
   -- from the COBOL compilation of CXB40090.CBL, CXB40091.CBL, and
   -- CXB40092.CBL.  The strings will be used as arguments to pragma Import.
```

```
   CXB40090_External_Name : constant String := "CXB40090";
   --                  MODIFY HERE AS NEEDED --- ^^^^^^^^

   CXB40091_External_Name : constant String := "CXB40091";
   --                  MODIFY HERE AS NEEDED --- ^^^^^^^^

   CXB40092_External_Name : constant String := "CXB40092";
   --                  MODIFY HERE AS NEEDED --- ^^^^^^^^

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The following string constants must be the external names resulting
   -- from the Fortran compilation of CXB50040.FTN, CXB50041.FTN,
   -- CXB50050.FTN, and CXB50051.FTN.
   --
   -- The strings will be used as arguments to pragma Import.
   --
   -- Note that the use of these four string constants will be split between
   -- two tests, CXB5004 and CXB5005.

   CXB50040_External_Name : constant String := "CXB50040";
   --                  MODIFY HERE AS NEEDED --- ^^^^^^^^

   CXB50041_External_Name : constant String := "CXB50041";
   --                  MODIFY HERE AS NEEDED --- ^^^^^^^^

   CXB50050_External_Name : constant String := "CXB50050";
   --                  MODIFY HERE AS NEEDED --- ^^^^^^^^

   CXB50051_External_Name : constant String := "CXB50051";
   --                  MODIFY HERE AS NEEDED --- ^^^^^^^^

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The following constants have been defined for use with the
   -- representation clause in FXACA00 of type Sales_Record_Type.
   --
   -- Char_Bits should be an integer at least as large as the number
   -- of bits needed to hold a character in an array.
   -- A value of 6 * Char_Bits will be used in a representation clause
   -- to reserve space for a six character string.
   --
   -- Next_Storage_Slot should indicate the next storage unit in the record
   -- representation clause that does not overlap the storage designated for
   -- the six character string.

   Char_Bits         : constant := 8;
   --     MODIFY HERE AS NEEDED ---^

   Next_Storage_Slot : constant := 6;
   --     MODIFY HERE AS NEEDED ---^

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--
```

```
   -- The following string constant must be the path name for the .AW
   -- files that will be processed by the Wide Character processor to
   -- create the C250001 and C250002 tests.  The Wide Character processor
   -- will expect to find the files to process at this location.

   Test_Path_Root : constant String :=
     "/data/ftp/public/AdaIC/testing/acvc/95acvc/";
   -- ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ --- MODIFY HERE AS NEEDED

   -- The following two strings must not be modified unless the .AW file
   -- names have been changed.  The Wide Character processor will use
   -- these strings to find the .AW files used in creating the C250001
   -- and C250002 tests.

  Wide_Character_Test : constant String := Test_Path_Root & "c250001";
  Upper_Latin_Test    : constant String := Test_Path_Root & "c250002";

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The following instance of Integer_IO or Modular_IO must be supplied
   -- in order for test CD72A02 to compile correctly.
   -- Depending on the choice of base type used for the type
   -- System.Storage_Elements.Integer_Address; one of the two instances will
   -- be correct.  Comment out the incorrect instance.

   --M package Address_Value_IO is
   --M     new Ada.Text_IO.Integer_IO(System.Storage_Elements.Integer_Address);

   package Address_Value_IO is
        new Ada.Text_IO.Modular_IO(System.Storage_Elements.Integer_Address);

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

end ImpDef;


     --=========================================================================--


package body ImpDef is

   -- NOTE: These are example bodies.  It is expected that implementors
   --       will write their own versions of these routines.

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The time required to execute this procedure must be greater than the
   -- time slice unit on implementations which use time slicing.  For
   -- implementations which do not use time slicing the body can be null.

   Procedure Exceed_Time_Slice is
      T : Integer := 0;
      Loop_Max : constant Integer := 4_000;
   begin
```

```
      for I in 1..Loop_Max loop
         T := Report.Ident_Int (1) * Report.Ident_Int (2);
      end loop;
   end Exceed_Time_Slice;

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

end ImpDef;
```

3.2.2.2 Package ImpDef.Annex_C

```
-- IMPDEFC.A
--
--                              Grant of Unlimited Rights
--
--     Under contracts F33600-87-D-0337, F33600-84-D-0280, MDA903-79-C-0687,
--     F08630-91-C-0015, and DCA100-97-D-0025, the U.S. Government obtained
--     unlimited rights in the software and documentation contained herein.
--     Unlimited rights are defined in DFAR 252.227-7013(a)(19).  By making
--     this public release, the Government intends to confer upon all
--     recipients unlimited rights  equal to those held by the Government.
--     These rights include rights to use, duplicate, release or disclose the
--     released technical data and computer software in whole or in part, in
--     any manner and for any purpose whatsoever, and to have or permit others
--     to do so.
--
--                                     DISCLAIMER
--
--     ALL MATERIALS OR INFORMATION HEREIN RELEASED, MADE AVAILABLE OR
--     DISCLOSED ARE AS IS.  THE GOVERNMENT MAKES NO EXPRESS OR IMPLIED
--     WARRANTY AS TO ANY MATTER WHATSOEVER, INCLUDING THE CONDITIONS OF THE
--     SOFTWARE, DOCUMENTATION OR OTHER INFORMATION RELEASED, MADE AVAILABLE
--     OR DISCLOSED, OR THE OWNERSHIP, MERCHANTABILITY, OR FITNESS FOR A
--     PARTICULAR PURPOSE OF SAID MATERIAL.
--*
--
-- DESCRIPTION:
--     This package provides tailorable entities for a particular
--     implementation.  Each entity may be modified to suit the needs
--     of the implementation.  Default values are provided to act as
--     a guide.
--
--     The entities in this package are those which are used exclusively
--     in tests for Annex C (Systems Programming).
--
-- APPLICABILITY CRITERIA:
--     This package is only required for implementations validating the
--     Systems Programming Annex.
--
-- CHANGE HISTORY:
--     29 Jan 96   SAIC    Initial version for ACVC 2.1.
--
--!

with Ada.Interrupts.Names;

package ImpDef.Annex_C is

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- Interrupt_To_Generate should identify a non-reserved interrupt
   -- that can be predictably generated within a reasonable time interval
```

```
   -- (as specified by the constant Wait_For_Interrupt) during testing.

   Interrupt_To_Generate: constant Ada.Interrupts.Interrupt_ID :=
      Ada.Interrupts.Names.Sigusr1;
   -- ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  --- MODIFY HERE AS NEEDED

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- Wait_For_Interrupt should specify the reasonable time interval during
   -- which the interrupt identified by Interrupt_To_Generate can be
   -- expected to be generated.

   Wait_For_Interrupt : constant := 0.0166;
   --                               ^^^^ --- MODIFY HERE AS NEEDED

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The procedure Enable_Interrupts should enable interrupts, if this
   -- is required by the implementation. [See additional notes on this
   -- procedure in the package body.]

   procedure Enable_Interrupts;

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The procedure Generate_Interrupt should generate the interrupt
   -- identified by Interrupt_To_Generate within the time interval
   -- specified by Wait_For_Interrupt. [See additional notes on this
   -- procedure in the package body.]

   procedure Generate_Interrupt;

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

end ImpDef.Annex_C;


     --=========================================================================--


package body ImpDef.Annex_C is

   -- NOTE: These are example bodies.  It is expected that implementors
   --       will write their own versions of these routines.

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The procedure Enable_Interrupts should enable interrupts, if this
   -- is required by the implementation.
   --
   -- The default body is null, since it is expected that most implementations
   -- will not need to perform this step.
   --
   -- Note that Enable_Interrupts will be called only once per test.
```

```
   procedure Enable_Interrupts is
   begin
      null;

   -- ^^^^^^^^^^^^^^^^^^^  MODIFY THIS BODY AS NEEDED  ^^^^^^^^^^^^^^^^^^^

   end Enable_Interrupts;

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- The procedure Generate_Interrupt should generate the interrupt
   -- identified by Interrupt_To_Generate within the time interval
   -- specified by Wait_For_Interrupt.
   --
   -- The default body assumes that an interrupt will be generated by some
   -- physical act during testing. While this approach is acceptable, the
   -- interrupt should ideally be generated by appropriate code in the
   -- procedure body.
   --
   -- Note that Generate_Interrupt may be called multiple times by a single
   -- test. The code used to implement this procedure should account for this
   -- possibility.

   procedure Generate_Interrupt is
        function Getpid return Integer;
        pragma Import (C, Getpid);

        procedure Kill (Pid : Integer; Sig : Integer);
        pragma Import (C, Kill);

        Pid : Integer := Getpid;

    begin
        Kill (Pid, Integer (Interrupt_To_Generate));
   end Generate_Interrupt;

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

end ImpDef.Annex_C;
```

3.2.2.3 Package ImpDef.Annex_D

```
-- This is an Apex-specific function to determine if one of the multiprocessor
-- (i.e. threaded) systems is being tested.  It returns an integer
-- instead of a Imdef.Annex_D.Processor_Type value to avoid
-- being dependent on the spec of Imdef.Annex_D, whose elaboration has to
-- call it.
-- ??? This currently makes no attempt to figure out if the system is
-- time sliced.  Time slicing is disabled by the FIFO_Within_Priorities
-- Task_Dispatching policy, which should be in any test that cares about this.

with System;
function Get_Processor_Type return Integer is
    type Processor_Type is (Uni_Processor, Multi_Processor);
    for Processor_Type use
       (Uni_Processor => 0, Multi_Processor => 1);

begin
    if System.Name'Image (System.System_Name) = "SPARC_SOLARIS_THREAD" or else
       System.Name'Image (System.System_Name) = "ALPHA_OSF1_THREAD" or else
       System.Name'Image (System.System_Name) = "MIPS_IRIX5_THREAD" or else
       System.Name'Image (System.System_Name) = "MIPS_IRIX6_THREAD" or else
       System.Name'Image (System.System_Name) = "HPPA_HPUX_THREAD" then
        return Processor_Type'Pos (Multi_Processor);
    else
        return Processor_Type'Pos (Uni_Processor);
    end if;
end Get_Processor_Type;

-- IMPDEFD.A
--
--                              Grant of Unlimited Rights
--
--     Under contracts F33600-87-D-0337, F33600-84-D-0280, MDA903-79-C-0687,
--     F08630-91-C-0015, and DCA100-97-D-0025, the U.S. Government obtained
--     unlimited rights in the software and documentation contained herein.
--     Unlimited rights are defined in DFAR 252.227-7013(a)(19).  By making
--     this public release, the Government intends to confer upon all
--     recipients unlimited rights  equal to those held by the Government.
--     These rights include rights to use, duplicate, release or disclose the
--     released technical data and computer software in whole or in part, in
--     any manner and for any purpose whatsoever, and to have or permit others
--     to do so.
--
--                                    DISCLAIMER
--
--     ALL MATERIALS OR INFORMATION HEREIN RELEASED, MADE AVAILABLE OR
--     DISCLOSED ARE AS IS.  THE GOVERNMENT MAKES NO EXPRESS OR IMPLIED
--     WARRANTY AS TO ANY MATTER WHATSOEVER, INCLUDING THE CONDITIONS OF THE
--     SOFTWARE, DOCUMENTATION OR OTHER INFORMATION RELEASED, MADE AVAILABLE
--     OR DISCLOSED, OR THE OWNERSHIP, MERCHANTABILITY, OR FITNESS FOR A
--     PARTICULAR PURPOSE OF SAID MATERIAL.
--*
```

```
--
-- DESCRIPTION:
--     This package provides tailorable entities for a particular
--     implementation.  Each entity may be modified to suit the needs
--     of the implementation.  Default values are provided to act as
--     a guide.
--
--     The entities in this package are those which are used exclusively
--     in tests for Annex D (Real-Time Systems).
--
-- APPLICABILITY CRITERIA:
--     This package is only required for implementations validating the
--     Real-Time Systems Annex.
--
-- CHANGE HISTORY:
--     29 Jan 96   SAIC    Initial version for ACVC 2.1.
--     27 Aug 98   EDS     Removed Processor_Type value Time_Slice
--!

with Get_Processor_Type;
package ImpDef.Annex_D is

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- This constant is the maximum storage size that can be specified
   -- for a task.  A single task that has this size must be able to
   -- run.  Ideally, this value is large enough that two tasks of this
   -- size cannot run at the same time.  If the value is too small then
   -- test CXDC001 may take longer to run.  See the test for further
   -- information.

   Maximum_Task_Storage_Size : constant := 16_000_000;
   --                                      ^^^^^^^^^^ --- MODIFY HERE AS NEEDED

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- Indicates the type of processor on which the tests are running.

   type Processor_Type is (Uni_Processor, Multi_Processor);
   for Processor_Type use
       (Uni_Processor => 0, Multi_Processor => 1);


   Processor : constant Processor_Type := Processor_Type'Val (Get_Processor_Type);
   --                                     ^^^^^^^^^^ --- MODIFY HERE AS NEEDED

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

end ImpDef.Annex_D;
```

3.2.2.4 Package ImpDef.Annex_G

```
-- IMPDEFG.A
--
--                              Grant of Unlimited Rights
--
--     Under contracts F33600-87-D-0337, F33600-84-D-0280, MDA903-79-C-0687,
--     F08630-91-C-0015, and DCA100-97-D-0025, the U.S. Government obtained
--     unlimited rights in the software and documentation contained herein.
--     Unlimited rights are defined in DFAR 252.227-7013(a)(19).  By making
--     this public release, the Government intends to confer upon all
--     recipients unlimited rights  equal to those held by the Government.
--     These rights include rights to use, duplicate, release or disclose the
--     released technical data and computer software in whole or in part, in
--     any manner and for any purpose whatsoever, and to have or permit others
--     to do so.
--
--                                    DISCLAIMER
--
--     ALL MATERIALS OR INFORMATION HEREIN RELEASED, MADE AVAILABLE OR
--     DISCLOSED ARE AS IS.  THE GOVERNMENT MAKES NO EXPRESS OR IMPLIED
--     WARRANTY AS TO ANY MATTER WHATSOEVER, INCLUDING THE CONDITIONS OF THE
--     SOFTWARE, DOCUMENTATION OR OTHER INFORMATION RELEASED, MADE AVAILABLE
--     OR DISCLOSED, OR THE OWNERSHIP, MERCHANTABILITY, OR FITNESS FOR A
--     PARTICULAR PURPOSE OF SAID MATERIAL.
--*
--
-- DESCRIPTION:
--     This package provides tailorable entities for a particular
--     implementation.  Each entity may be modified to suit the needs
--     of the implementation.  Default values are provided to act as
--     a guide.
--
--     The entities in this package are those which are used exclusively
--     in tests for Annex G (Numerics).
--
-- APPLICABILITY CRITERIA:
--     This package is only required for implementations validating the
--     Numerics Annex.
--
-- CHANGE HISTORY:
--     29 Jan 96   SAIC    Initial version for ACVC 2.1.
--
--!

package ImpDef.Annex_G is

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- This function must return a "negative zero" value for implementations
   -- for which Float'Signed_Zeros is True.

   function Negative_Zero return Float;
```

```
--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

end ImpDef.Annex_G;


     --=====================================================================--


package body ImpDef.Annex_G is

   -- NOTE: These are example bodies.  It is expected that implementors
   --       will write their own versions of these routines.

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

   -- This function must return a negative zero value for implementations
   -- for which Float'Signed_Zeros is True.
   --
   -- The default body simply returns a negated literal 0.0.  If the
   -- default body does not return the value corresponding to a negatively
   -- signed zero for the implementation under test, it must be replaced
   -- by one which does.  See RM A.5.3(13).

   function Negative_Zero return Float is
   begin
      return -0.0;      -- Note: If this value is not negative zero for the
                        --       implementation, use of this "default" value
                        --       could result in false failures in
                        --       implementations where Float'Signed_Zeros
                        --       is True.

   -- ^^^^^^^^^^^^^^^^^^^^  MODIFY THIS BODY AS NEEDED  ^^^^^^^^^^^^^^^^^^^^

   end Negative_Zero;

--=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====-=====--

end ImpDef.Annex_G;
```

3.3 WITHDRAWN TESTS

At the time of this conformity assessment testing, the following 3 tests were withdrawn from the ACATS.

CE2120A CE2120B CE3119A

APPENDIX A

COMPILATION SYSTEM OPTIONS AND LINKER OPTIONS

A.1 Compilation System Options

The compiler options of this Ada processor, as described in this Appendix, are provided by the customer. Unless specifically noted otherwise, references in this Appendix are to compiler documentation and not to this report.

-compile X1 X2, X3 ...

Processes the files X1 X2 X3... in order. For each file Xn, the following operations are performed:

The file Xn can contain several compilation units. Those units are parsed in order, and if there are duplicate units within the file Xn, the later ones overwrite the earlier ones.

If there are syntax errors, a listing file is produced. Syntax corrections that would normally be just applied to the file are instead converted to error messages in the listing file.

Next, the compilation units of the file are installed, in an order chosen by the compiler. If there are semantic errors, a list file is produced. The order of the files in the list file will be in compilation order, not the order in which they appear in Xn.

Finally, if each of the comp units installed successfully, the compiler searches the view for .ada files that have been obsolesced by the newly compiled files. Any xxx.ada files that are obsolesced are renamed xxx.ada.obs.

By default, these actions are not enabled. Placing the -compile switch on the command line enables these actions.

CLEAN_GOAL

The state to which the units will be cleaned. All compilation artifacts associated with higher states are deleted. Default: Archived

CLOSURE
Used to determine additional units to be analyzed. Default: installed

COMPILER_KEY
The compiler to use, which provides for platform-specific semantic checking and code generation.
Default: $APEX_BASE/ada/keys/$APEX_ARCH.4.0.0.rel/key

COMPILER_TOOLS_PATH
The COMPILER_TOOLS_PATH switch is used by the Ada compiler and other Apex tools to locate target-specific versions of each tools. The list of pathnames must be separated by ":"s (same conventions as $PATH.)
Default: $APEX_HOME/$APEX_ARCH/sup

FIRST_ERROR
Continue past the first unit with errors. If True, command will stop after the first unit containing an error. Default: False

FLAG_INEVITABLE_EXCEPTIONS
Control the handling of any statically determinable situation that is certain to raise an exception when executed, such as an out-of-bounds assignment. Default: False

IGNORE_INVALID_REP_SPECS
Control the handling of invalid or unsupported representation specifications. Representation specifications are considered invalid if they do not conform to the restrictions specified in "LRM Annex M: Implementation-Dependent Characteristics." Default: False

IGNORE_REP_SPECS
Ignore representation specifications during semantic analysis.
Default: False

IGNORE_UNSUPPORTED_REP_SPECS
Control the handling of unsupported representation specifications. This switch is overridden by the IGNORE_INVALID_REP_SPECS switch. When IGNORE_INVALID_REP_SPECS is True, this switch has no effect. If IGNORE_INVALID_REP_SPECS is False and this switch is True, unsupported representation specifications are reported with warning messages in the output window and are otherwise ignored. If this switch is False, unsupported representation specifications are treated as errors, causing analysis of the units that contain them to fail. Default: False

INTO
The directory into which the files will be parsed. Must name a view or a directory in a view. If blank, the current directory is used.
Default: " " (current directory)

NEW_RELEASE
Recompiles the units in each view, converting to new DIANA, CG attribute, and program library formats Default: False

OPTIMIZATION_LEVEL
The optimization level to use for compiling. 0 is fastest compilation, 2 is best code. This switch also controls how much inlining is done: at level 0, no routines are inlined; at level 1, only those routines with an applied pragma Inline are candidates for inlining; at level 2, all routines declared within the current same compilation unit as the call site are candidates for inlining, in addition to those made available at level 1 with a pragma Inline. Default: 0

OPTIMIZATION_OBJECTIVE
The optimization objective, either Time or Space, to be used for any compilation unit that does not contain a pragma Optimize. Execution speed (Time) or code size (Space) are never completely ignored, but if this switch is set to Time, the compiler places greater emphasis on optimizing for speed, while the size of the code is of secondary importance. If it is set to Space, optimizations for speed that increase size are not done.
If this switch is set to Space, the loop unrolling optimization (usually performed at level 2) is not performed, Default: Time

PROFILING
The type of profiling to use when preparing the code. Set this switch to "" to turn off profiling. For native compilers, both Gprof and Prof are valid settings. For embedded compilers, only Prof is recognized.
Default: ""

REJECT_BAD_LRM_PRAGMAS
Control the handling of illegal Ada pragmas. When True, illegal Ada pragmas are treated as errors, thus causing analysis of the units that contain them to fail. When False, illegal Ada pragmas are reported with warning messages in the output window and are otherwise ignored.
Default: False

REJECT_BAD_RATIONAL_PRAGMAS
Control the handling of illegal Rational-defined pragmas. When True, illegal Rational pragmas are treated as errors, thus causing analysis of the units that contain them to fail. When False, illegal Rational pragmas are reported with warning messages in the output window and are otherwise ignored. Default: False

REJECT_INEVITABLE_EXCEPTIONS
Control the handling of any statically determinable situation that is certain to raise an exception when executed, such as an out-of-bounds assignment. When True, this switch overrides the FLAG_INEVITABLE_EXCEPTIONS switch and inevitable exceptions are treated as errors, thus causing analysis of the units that contain them to fail. When False, the treatment of inevitable exceptions depends on the setting of the FLAG_INEVITABLE_EXCEPTIONS switch.
Default: False

REJECT_PROMPTS
: The compiler will allow you to code units that contain [statement] prompts. Default: False

REJECT_SYNTAX_ERRORS
: The editor and compiler should make syntactic corrections to the programs. When the value is False, corrections are made. When True, corrections are not made; you must make them. Default: False

REJECT_UNDEFINED_PRAGMAS
: Control the handling of any pragmas not defined in the LRM or in the Compiler Reference. When True, undefined pragmas are treated as errors, thus causing analysis of the units that contain them to fail. When False, undefined pragmas are reported with warning messages in the output window and are otherwise ignored. Default: False

TARGET_DIRECTORY
: Target directory for cross compiling. Default: " "

TARGET_MACHINE
: Host name of the target machine for cross compiling. Default: " "

TRACING
: The type of tracing to perform. The value can be a combination of types, e.g. Runtime+Call_Return. Default: RUNTIME

A.2 Linker Options

The linker options of this Ada processor, as described in this Appendix, are provided by the customer. Unless specifically noted otherwise, references in this Appendix are to linker documentation and not to this report.

ADA_LINK_MODE
: Specifies the link mode for Ada main programs in the view. Valid for targets that support shared libraries. Default: default

COMPILER_KEY
: The compiler to use, which provides for platform-specific semantic checking and code generation.
Default: $APEX_BASE/ada/keys/$APEX_ARCH.4.0.0.rel/key

COMPILER_TOOLS_PATH
: The COMPILER_TOOLS_PATH switch is used by the Ada compiler and other Apex tools to locate target-specific versions of each tools. The list of pathnames must be separated by ":"s
(same conventions as $PATH.)
Default: $APEX_HOME/$APEX_ARCH/sup

CONFIGURATION
: Configuration to use during a link to compute the closure of a main program. If blank, the imports are used. Can be used to specify units to compile beyond those found in the code closure in the Compiler Switches imported views. Default: " "

ELABORATION_ORDER_LISTING
: Create a file containing a listing of the elaboration order of the units in its closure is created when a main program is linked. Elaboration-order listings can be created only for main programs.
Default: False

INCREMENTAL_LINK
: Attempt to use incremental features of the platform linker.
Default: False

LINK_CONTRIBUTION_DEFAULT_MODE
: The LINK_CONTRIBUTION_DEFAULT_MODE switch provides view selective control over linking with/without shared libraries. This switch is only valid when the ADA_LINK_MODE switch in the Ada main program's view is "dynamic_or_static". A LINK_CONTRIBUTION_DEFAULT_MODE of "static" indicates that only object files are to be used from that view in a link process. Any shared library is to be ignored. If the LINK_CONTRIBUTION_DEFAULT_MODE is "dynamic", only the shared library in the view is to be used for a link. It is an error at link-time if the view cannot provide a shared library (that is, the view was compiled with CREATED_SHARED_LIBRARY set to FALSE). A LINK_CONTRIBUTION_DEFAULT_MODE of "dynamic_or_static" specifies that the shared library is to be used if the view is a shared library view (CREATE_SHARED_LIBRARY = TRUE) and the object files are to be used otherwise. If the LINK_CONTRIBUTION_DEFAULT_MODE switch does not have a

value in a view, the value defaults to "dynamic_or_static".
Default: dynamic_or_static

NON_ADA_LINKAGE
The arguments to pass to the target linker. This can be used to specify object files and archive libraries for non-Ada program units that will be included when an Ada main program is linked. Default: " "

NONBLOCKING_IO
When a task issues an I/O request, do not wait for the I/O to complete. When True, the task is blocked until the I/O completes, but other tasks in the program may run. When False, the entire program (all tasks) are blocked while any task is waiting for an I/O request to complete.
Default: False

POSIX_COMPLIANT
Use the POSIX I/O behavior. When False, uses traditional I/O behavior allowing users to use some of the UNIX signals that POSIX forbids.
Default: True

RUNTIMES
Identifies the directory where archive libraries and objects are found that are used during the link phase.
Default:
(native systems)
$APEX_HOME/$APEX_ARCH/lib
(cross systems)
$APEX_BASE/ada/compilers/<target_family>.<version>/<compiler_variant>/usr

TARGET_DIRECTORY
Target directory. Default: " "

TARGET_MACHINE
Host name of the target machine. Default: " "

USER_LINK_BLOCK
Sets the start address in target memory for linking the first user program. Default: target dependent address

The following contains a sample test script to compile a test with compilation errors.

```
#! /bin/sh
# -- for this script the following variables are referenced
# -- $ACVC_HOME = testing area for results collection
# -- $ACVC_VIEW = pathname to ACATS sources
# -- $WORK_VIEW = Apex view used when compiling/linking/running tests
# -- $compile="rada"
# -- $link="rada -goal linked"
#
# -- Echo the compiler key for Rev info --
# -- $WORK_VIEW is the path to the working view where test are compiled
echo ::: Compiler key '(Compiler under test)'
echo '    ''`apex report -no_register -of errors -format
echo     "<switch(COMPILER_KEY,_UNDEFINED_)>" $WORK_VIEW`
#
# -- echo to output (directed to log file) that starting tests --
echo ""
echo '||| <A' `/bin/date '+%H:%M'`'>' start .
#
# -- remember the location from where the run started --
run_home=`/bin/pwd`
#
# -- move the the working directory --
cd $WORK_VIEW
#
# -- echo to output (directed to log file) the current testnames --
echo '||| <C' `/bin/date '+%H:%M'`'>' b32101a b32103a b32104a b32106a b32201a
#
# -- compile the test source redirecting the listing lst directory
# -- $ACVC_VIEW holds the path to the ACATS sources location
$compile -into . -listing_directory $run_home/lst \
    -compile $ACVC_VIEW/b/3/b32101a.ada $ACVC_VIEW/b/3/b32103a.ada \
    $ACVC_VIEW/b/3/b32104a.ada $ACVC_VIEW/b/3/b32106a.ada \
      $ACVC_VIEW/b/3/b32201a.ada
#
# -- clean artifacts from working view to prep for next test
apex clean -batch .
#
# -- clean out working view to prep for next test
/bin/rm -f *.1.ada; /bin/rm -f *.2.ada
/bin/rm -f *.ada.obs; /bin/rm -f [RXYa-z]*
#
# -- echo to output (directed to log file) that script is complete
echo '||| <Z' `/bin/date '+%H:%M'`'>' end .
```

The following contains a sample test script to compile, link and execute a test.

```
#! /bin/sh
# -- for this script the following variables are referenced
# -- $ACVC_HOME = testing area for results collection
# -- $ACVC_VIEW = pathname to ACATS sources
# -- $WORK_VIEW = Apex view used when compiling/linking/running tests
# -- $compile="rada"
# -- $link="rada -goal linked"
# -- $execute=""
#
# -- Echo the compiler key for Rev info --
# -- $WORK_VIEW is the path to the working view where test are compiled
echo ::: Compiler key '(Compiler under test)'
echo '    ''`apex report -no_register -of errors -format
echo   "<switch(COMPILER_KEY,_UNDEFINED_)>" $WORK_VIEW`
#
# -- echo to output (directed to log file) that starting tests --
echo ""
echo '||| <A' `/bin/date '+%H:%M'`'>' start .
#
# -- remember the location from where the run started --
run_home=`/bin/pwd`
#
# -- move the the working directory --
cd $WORK_VIEW
#
# -- echo to output (directed to log file) note to indicate
# -- test is being compiled
echo '||| <C' `/bin/date '+%H:%M'`'>' c761007
#
# -- compile the test source redirecting the listing lst directory
# -- $ACVC_VIEW holds the path to the ACATS sources location
$compile -into . -listing_directory $run_home/lst -compile $ACVC_VIEW/c/7/c761007.a
#
# -- link the test program
$link c761007.2.ada
#
# -- echo to output (directed to log file) note that test
# -- is going to execute
echo '||| <X' `/bin/date '+%H:%M'`'>' c761007
#
# -- first check that an executable was generated
# -- if it does exist then run the executable
# -- execute=""
# -- (during development phase, execute is set to a script
# -- that is used to detect hung executes)
if [ $is_executable c761007 ]
then
   $execute c761007
fi
```

```
#
# -- echo to output (directed to log file) that execution complete
echo '||| <X' `/bin/date '+%H:%M'`'>' END_EXECUTE
#
# -- clean artifacts from working view to prep for next test
apex clean -batch .
#
# -- clean out working view to prep for next test
/bin/rm -f *.1.ada; /bin/rm -f *.2.ada
/bin/rm -f *.ada.obs; /bin/rm -f [RXYa-z]*
#
# -- echo to output (directed to log file) note that test
# -- is being compiled
echo '||| <C' `/bin/date '+%H:%M'`'>' c761010
#
# -- compile the test source redirecting the listing lst directory
# -- $ACVC_VIEW holds the path to the ACATS sources location
$compile -into . -listing_directory $run_home/lst -compile $ACVC_VIEW/c/7/c761010.a
#
# -- link the test program
$link c761010.2.ada
#
# -- echo to output (directed to log file) note that test
# -- is going to execute
echo '||| <X' `/bin/date '+%H:%M'`'>' c761010
#
# -- first check that an executable was generated
# -- if it does exist then run the executable
# -- execute=""
# -- (during development phase, execute is set to a script
# -- that is used to detect hung executes)
if [ $is_executable c761010 ]
then
   $execute c761010
fi
#
# -- echo to output (directed to log file) that execution complete
echo '||| <X' `/bin/date '+%H:%M'`'>' END_EXECUTE
#
# -- clean artifacts from working view to prep for next test
apex clean -batch .
#
# -- clean out working view to prep for next test
/bin/rm -f *.1.ada; /bin/rm -f *.2.ada
/bin/rm -f *.ada.obs; /bin/rm -f [RXYa-z]*
#
# -- echo to output (directed to log file) that script is complete
echo '||| <Z' `/bin/date '+%H:%M'`'>' end .
```

APPENDIX B

POINTS OF CONTACT

Ada Conformance Assessment Laboratory

Phil Brashear
EDS Conformity Testing Center
4646 Needmore Road, Bin 46
P.O. Box 24593
Dayton OH 45424-0593
U.S.A.
Phone : (937) 237-4510
Internet : Phil.Brashear@eds.com

Ada Conformity Assessment Authority

Randall Brukardt
ACAA
P.O. Box 1512
Madison WI 53701
U.S.A.
Phone : (608) 245-0375
Internet : Agent@Ada-auth.org

POINTS OF CONTACT

For technical information about this Ada processor, contact:

David J. Lofgren
Rational Software Corporation
18880 Homestead Road
Cupertino CA 95014
(408) 863-5041

For sales information about this Ada processor, contact:

Greg Bek
Apex Product Management
Rational Software Corporation
18880 Homestead Road
Cupertino CA 95014
(408) 863-4394

APPENDIX C

REFERENCES

[ACAP] Language Processor Validation Procedures, EDS Conformance Testing Center, March 12, 1999

[Ada95] Reference Manual for the Ada Programming Language, ANSI/ISO/IEC 8652:1995

[Pro01] Operating Procedures for Ada Conformity Assessments, Version 3.0, Ada Resource Association, April 23, 2001

[UG01] The Ada Conformity Assessment Test Suite (ACATS) Version 2.4 User's Guide, Ada Conformity Assessment Authority, March 21, 2001