ACAL Control Number: EDS19991203DDC02-2.2

Ada Conformity Assessment Test Report

Certificate Number: A000118E2.2-004

DDC-I A/S

DDC-I SCORE Ada Compiler System, Version 1.1, SPARC/FM586

UltraSPARC5 under Solaris, 2.5.1 =>

Zitech Pro, Pentium 100 (bare)

(Final)

20 January 2000

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 Ada Conformity Assessment Authority Operating Procedures, Version 2.0. The Ada Conformity Assessment Test Suite (ACATS), Version 2.2, 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 A000118E2.2-004 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 2.2 (VCS label A2_2H) (See Section 2.2 for details)
Location of Testing	DDC-I A/S Gl. Lundtoftevej 1B DK-2800 Lyngby, Denmark
Test Completion Date	18 January 2000

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 A000118E2.2-004

Specialized Needs Annexes

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

SPECIALIZED NEEDS ANNEXES	Total	With-Drawn	Passed	Inappli-cable	Unsup-ported
C Systems Programming	25	0	24	1	0
& required Section 13	12	0	12	0	0
(representation support)	---	---	---	---	---
	37	0	36	1	0
D Real-Time Systems					
(which requires Annex C)	54	0	54	0	0
E Distributed Systems	26	0	** NOT TESTED **		
F Information Systems	21	0			
& required Section 13	1	0	** NOT TESTED **		
(representation support)	---	---			
	22	0			
G Numerics	29	0	** NOT TESTED **		
H Safety and Security	31	0	31	0	0

DECLARATION OF CONFORMITY

Customer: DDC-I A/S

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: 2.2

Ada Processor

Ada Compiler Name and Version: DDC-I SCORE Ada Compiler System,
Version 1.1, SPARC/FM586

Host Computer System: UltraSPARC5
Solaris, 2.5.1

Target Computer System: Zitech Pro, Pentium 100
(bare)

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 Ada Conformity Assessment Test 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 2.0 of the Operating Procedures of the ACAA [Pro99]. Testing was accomplished using Version 2.2 of the Ada Conformity Assessment Test Suite (ACATS), also known as the Ada Compiler Validation Capability (ACVC). 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 [Pro99]. A detailed description of the ACATS may be found in the ACVC User's Guide [UG98].

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 which 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 [UG98]. 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 [UG98]).

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 Compiler Validation Capability	The means of checking conformity of Ada processors, consisting of tests, support programs, and a User's Guide. Also referred to as the Ada Conformity Assessment Test Suite.
Ada Conformity Assessment Test Suite (ACATS)	Alternate name for the ACVC (which see).
Ada Conformity Assessment Laboratory	An organization which 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.

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.

C45624A..B (2 tests) check that the proper exception is raised if MACHINE_OVERFLOWS is FALSE for floating point types and the results of various floating-point operations lie outside the range of the base type; for this processor, MACHINE_OVERFLOWS is TRUE.

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 following 262 tests check operations on sequential, text, and direct access files; this processor does not support external files:

CE2102A..C (3)	CE2102G..H (2)	CE2102K	CE2102N..Y (12)
CE2103A..D (4)	CE2104A..D (4)	CE2106A..B (2)	CE2108E..H (4)
CE2109A..C (3)	CE2110A	CE2110C	CE2111A..C (3)
CE2111E..G (3)	CE2111I	CE2120A..B (2)	CE2201A..N (14)
CE2203A	CE2204A..D (4)	CE2205A	CE2206A
CE2208B	CE2401A..C (3)	CE2401E..F (2)	CE2401H..L (5)
CE2403A	CE2404A..B (2)	CE2405B	CE2406A
CE2407A..B (2)	CE2408A..B (2)	CE2409A..B (2)	CE2410A..B (2)
CE2411A	CE3102A..B (2)	CE3102F..H (3)	CE3102J..K (2)
CE3103A	CE3104A..C (3)	CE3106A..B (2)	CE3107A..B (2)
CE3108A..B (2)	CE3110A	CE3112C..D (2)	CE3114A
CE3115A	CE3119A	EE3203A	EE3204A

CE3207A	CE3301A	CE3302A	CE3304A
CE3305A	CE3401A	CE3402A	EE3402B
CE3402C..D (2)	CE3403A..C (3)	CE3403E..F (2)	CE3404B..D (3)
CE3405A	CE3405C..D (2)	CE3406A..D (4)	CE3407A..C (3)
CE3408A..C (3)	CE3409A	CE3409C..E (3)	EE3409F
CE3410A	CE3410C..E (3)	CE3411A	CE3411C
CE3412A	EE3412C	CE3413A..C (3)	CE3414A
CE3602A..D (4)	CE3603A	CE3604A..B (2)	CE3605A..E (5)
CE3606A..B (2)	CE3704A..F (6)	CE3704M..O (3)	CE3705A..E (5)
CE3706D	CE3706F..G (2)	CE3804A..J (10)	CE3804M
CE3804O..P (2)	CE3805A..B (2)	CE3806A..B (2)	CE3806D..E (2)
CE3806G..H (2)	CE3902B	CE3904A..B (2)	CE3905A..C (3)
CE3905L	CE3906A..C (3)	CE3906E..F (2)	CXA8001..3 (3)
CXA9001..2(2)	CXAA001..18 (18)	CXAB001	CXAC001..4 (4)
CXACA01..2 (2)	CXACB01..2 (2)	CXACC01	

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.

CXC7003 checks that the Task_attributes operations Set_Value and Reinitialize perform finalization on the old value of the attribute of the specified task. This processor reports Not-Applicable because the implementation limits the allowed size of a task attribute (in instances of Ada.Task_Attributes) to 32 bits.

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:

- Test Modification: The source code of the test is changed.
  Examples of 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.

- Processing Modification: The processing of the test by the Ada processor for conformity assessment is changed.
  Examples of 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.

- Evaluation Modification: The evaluation of a test result is changed.
  An example of 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 2.2H were used along with any modifications detailed below.

Modifications were required for 38 tests.

The following 29 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.

B24007A	B29001A	B33201A	B33201C	B33204A
B36201A	B37106A	B37301I	B43201A	B48002A
B48002D	B54A01L	B64003A	B660001	B67001A
B67001B	B67001C	B67001D	B83001D	B83F02C
B95007B	B95081A	BA11005	BA12001	BA12002
BA12004	BA1109A	BC3009C	BC51016	

The following 7 allowable test modifications from the "ACATS Modifications List", Version 2.2H were used.

C840001	CD30004	CXB3004	CXB3006	CXB3009
CXB3010	CXB3012			

B95040D, as directed by the ACAA, was graded passed with the following evaluation modification:

> The test will be graded as PASSED as long as the final Report.Result call reports PASSED, even though calls to Report.Comment in the test can produce overlapped output, (because these calls come from three separate tasks).

BXH4004, as directed by the ACAA, was graded passed with the following grading modification:

> The test will be graded as PASSED as long as all expected errors are reported, along with an additional error on line 100. (A restriction may cause additional constructs to be rejected. The construct on line 100 violates one of the restrictions given in this test for this implementation.)

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 G, Numerics            (all CXG* files)
```

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

No tests for Annex D, Real-Time Systems, were graded "unsupported".

No tests for Annex H, Safety and Security, were graded "unsupported".

CHAPTER 3

PROCESSING INFORMATION

3.1 CONFORMITY ASSESSMENT PROCESS

A partial 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:

For the compiler, the used options are:

```
  -list       : make a listing file
  -nowarnings : suppress warning messages
```

PROCESSING INFORMATION

For the linker, the used options are:

```
  -nowarnings         : suppress warning messages
  -objects cd300051.o : always include object file for the C source needed by
                        impdef for the setting of the constant
                        CD30005_1_Foreign_Address
```

Apart from this, the linking of the following tests use special options that were approved by the ACAA:

```
  cxb3013 : -objects cxb30130.o,cxb30131.o      (add needed C objects)
```

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 [UG98]. 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	255
$BIG_ID1	AAA ... A1 (255 characters)
$BIG_ID2	AAA ... A2 (255 characters)
$BIG_ID3	AAA ... A3A ... A (255 characters)
$BIG_ID4	AAA ... A4A ... A (255 characters)
$BIG_STRING1	"AAA ... A" ((255+1)/2 characters)
$BIG_STRING2	"AAA ... A1" (255/2 characters)
$BLANKS	" ... " (255-20 blanks)
$MAX_STRING_LITERAL	"AAA ... A" (255 characters)
$ACC_SIZE	32

```
$ALIGNMENT                      4

$COUNT_LAST                     2_147_483_647

$ENTRY_ADDRESS                  Fcndecl.Entry_Address

$ENTRY_ADDRESS1                 Fcndecl.Entry_Address1

$ENTRY_ADDRESS2                 Fcndecl.Entry_Address2

$FIELD_LAST                     255

$FORM_STRING                    ""

$FORM_STRING2                   "CANNOT_RESTRICT_FILE_CAPACITY"

$GREATER_THAN_DURATION          75_000.0

$ILLEGAL_EXTERNAL_FILE_NAME1    /NODIRECTORY1/FILENAME1

$ILLEGAL_EXTERNAL_FILE_NAME2    /NODIRECTORY2/FILENAME2

$INAPPROPRIATE_LINE_LENGTH      -1

$INAPPROPRIATE_PAGE_LENGTH      -1

$INTEGER_FIRST                  -2147483648

$INTEGER_LAST                   2147483647

$LESS_THAN_DURATION             -75_000.0

$MACHINE_CODE_STATEMENT         ASM_Instruction'( Asm( "nop" ) );

$MAX_INT                        2147483647

$MIN_INT                        -2147483648

$NAME                           SHORT_SHORT_INTEGER

$NAME_SPECIFICATION1
         /usr/sparc12/ada9x/acvc/acvc22/work_native/work/X2120A

$NAME_SPECIFICATION2
         /usr/sparc12/ada9x/acvc/acvc22/work_native/work/X2120B

$NAME_SPECIFICATION3
         /usr/sparc12/ada9x/acvc/acvc22/work_native/work/X3119A

$OPTIONAL_DISC                  OPTIONAL_DISC

$RECORD_DEFINITION              RECORD NULL; END RECORD;
```

PROCESSING INFORMATION

$RECORD_NAME	ASM_Instruction
$TASK_SIZE	(32+2*32+4*32+32+128*32)
$TASK_STORAGE_SIZE	1024
$VARIABLE_ADDRESS	Fcndecl.Variable_Address
$VARIABLE_ADDRESS1	Fcndecl.Variable_Address1
$VARIABLE_ADDRESS2	Fcndecl.Variable_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
--
--*
--
-- 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.
```

```
PROCESSING INFORMATION

--     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 := False;
   --                                       ^^^^^ --- MODIFY HERE AS NEEDED

   Validating_Annex_H : constant Boolean := True;
   --                                       ^^^^^ --- 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

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

```
PROCESSING INFORMATION

   -- 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 := "**NONE**";
   --           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 order of the next two items were switched by DDC-I, so that
   -- CD30005_1_External_Name can be used in the definition of
   -- CD30005_1_Foreign_Address.

   -- 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 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.

   function Impdef_Cd30005_1( Value : Integer ) return Integer;
   pragma Import( C, Impdef_Cd30005_1, CD30005_1_External_Name );

   CD30005_1_Foreign_Address : constant System.Address:=
                                                  Impdef_Cd30005_1'Address;
   --                     MODIFY HERE AS REQUIRED --- ^^^^^^^^^^^^

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

```
   -- 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.
```

```
PROCESSING INFORMATION

   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 :=
     "../src_fm586/c2/";
   -- ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ --- 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.

   -- I package Address_Value_IO is
   -- I     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
--
--*
--
-- 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 :=
      20;  -- to allow trivial compilation
   -- ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  --- 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 := 1.0;
   --                               ^^^^ --- 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
   begin
      --      Report.Comment (". >>>>> GENERATE THE INTERRUPT NOW <<<<< ");

      System.Machine_Code.Asm( "int $20" );

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

   end Generate_Interrupt;

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

end ImpDef.Annex_C;
```

3.2.2.3 Package ImpDef.Annex_D

```
-- IMPDEFD.A
--
--*
--
-- 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
--!

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);

   Processor : constant Processor_Type := Uni_Processor;
   --                                     ^^^^^^^^^^ --- MODIFY HERE AS NEEDED

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

end ImpDef.Annex_D;
```

3.2.2.4 Package ImpDef.Annex_H

```
-- IMPDEFH.A
--
--*
--
-- DESCRIPTION:
--      This package is used to define those values that are implementation
--      defined for use with validating the Safety and Security special needs
--      annex, Annex-H.
--
-- APPLICABILITY CRITERIA:
--     This package is only required for implementations validating the
--     Safety and Security Annex.
--
-- CHANGE HISTORY:
--      13 FEB 96   SAIC    Initial version
--      25 NOV 96   SAIC    Revised for release 2.1
--
--!

package Impdef.Annex_H is

  type Scalar_To_Normalize is
       (   Id0,  Id1,  Id2,  Id3,  Id4,  Id5,  Id6,  Id7,  Id8,  Id9,
          Id10, Id11, Id12, Id13, Id14, Id15, Id16, Id17, Id18, Id19,
          Id20, Id21, Id22, Id23, Id24, Id25, Id26, Id27, Id28, Id29,
          Id30, Id31, Id32, Id33, Id34, Id35, Id36, Id37, Id38, Id39,
          Id40, Id41, Id42, Id43, Id44, Id45, Id46, Id47, Id48, Id49,
          Id50, Id51, Id52, Id53, Id54, Id55, Id56, Id57, Id58, Id59,
          Id60, Id61, Id62, Id63, Id64, Id65, Id66, Id67, Id68, Id69,
          Id70, Id71, Id72, Id73, Id74, Id75, Id76, Id77, Id78, Id79,
          Id80, Id81, Id82, Id83, Id84, Id85, Id86, Id87, Id88, Id89,
          Id90, Id91, Id92, Id93, Id94, Id95, Id96, Id97, Id98, Id99,
          IdA0, IdA1, IdA2, IdA3, IdA4, IdA5, IdA6, IdA7, IdA8, IdA9,
          IdB0, IdB1, IdB2, IdB3, IdB4, IdB5, IdB6 );

  -- NO MODIFICATION NEEDED TO TYPE SCALAR_TO_NORMALIZE.  DO NOT MODIFY.

  type Small_Number is range 1..100;

  -- NO MODIFICATION NEEDED TO TYPE SMALL_NUMBER.  DO NOT MODIFY.

--=====================================================================
  -- When the value documented in H.1(5) as the predictable initial value
  -- for an uninitialized object of the type Scalar_To_Normalize
  -- (an enumeration type containing 127 identifiers) is to be in the range
  -- Id0..IdB6, set the following constant to True; otherwise leave it set
  -- to False.

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

```
--=========================================================================
  -- If the above constant Default_For_Scalar_To_Normalize_Is_In_Range is
  -- set True, the following constant must be set to the value documented
  -- in H.1(5) as the predictable initial value for the type
  -- Scalar_To_Normalize.

  Default_For_Scalar_To_Normalize : constant Scalar_To_Normalize := Id0;
  --                                       MODIFY HERE AS NEEDED --- ^^^

--=========================================================================
  -- When the value documented in H.1(5) as the predictable initial value
  -- for an uninitialized object of the type Small_Number
  -- (an integer type containing 100 values) is to be in the range
  -- 1..100, set the following constant to True; otherwise leave it set
  -- to False.

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

--=========================================================================
  -- If the above constant Default_For_Small_Number_Is_In_Range is
  -- set True, the following constant must be set to the value documented
  -- in H.1(5) as the predictable initial value for the type Small_Number.

  Default_For_Small_Number : constant Small_Number := 100;
  --                          MODIFY HERE AS NEEDED --- ^^^

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

end Impdef.Annex_H;
```

3.3 WITHDRAWN TESTS

At the time of this conformity assessment testing, the following 1 test was withdrawn from the ACATS.

LA14023

APPENDIX A

COMPILATION SYSTEM OPTIONS AND LINKER OPTIONS

A.1 Compiler Options

```
Usage: ada [ option ... ] source_file_name ...

Compile one or more source texts.

Library Options:
   -path file_name{,file_name}        use given library search path.
   -library file_name                 use default path associated with the
                                      given library

Code Generation Options:
   -check check_name=on|off [,...]    suppress or unsuppress specified runtime
                                      checks. Valid check_names: All, Access,
                                      Accessibility, Discriminant, Division,
                                      Elaboration, Index, Length, Overflow,
                                      Range, Storage, Tag
   -nocheck                           suppress all runtime checks
   -core_andf                         generate ANDF 4.0
   -extended_andf                     generate ANDF 4.1 (default)
   -debug                             generate debug information
   -nodebug                           don't generate debug information (default)
   -install                           call ANDF installer (default)
   -noinstall                         don't call ANDF installer
   -keep_primary                      keep the primary ANDF capsule in the
                                      library (default)
   -nokeep_primary                    remove the primary ANDF capsule
   -optimize class_name=on|off [,...] perform additional optimizations. Valid
                                      class_names: All, Check, Speed, Size
   -nooptimize                        don't perform additional optimizations
                                      (default)

Compilation Options(1):
   -progress                          print progress reports
   -noprogress                        don't print progress reports (default)
   -filename_generation method
      append                          append to source file name
      change_suffix                   change suffix of source file name
                                      (default)
```

COMPILATION SYSTEM OPTIONS AND LINKER OPTIONS

-warnings	display warnings (default)
-nowarnings	don't display warnings

Compilation Options(2):

-ada83	enforce Ada 83 syntax and semantics
-noada83	Ada 95 syntax and semantics (default)
-verbose	as -progress but more verbose
-noverbose	don't print verbose messages (default)
-analyze	syntax & static semantic checks only
-compile	full compilation (default)
-parse	syntax checks and library storing

Compilation Options(3):

-syntax	syntax checks only
-nosyntax	full compilation (default)
-list	make listing file(s)
-nolist	don't make listing file(s) (default)
-lock	lock unit after saving
-nolock	don't lock unit after saving (default)
-recompile	mark units as recompilable (default)
-norecompile	mark units as non recompilable

Indirect Options:

-filelist	parameters contains names of files to be compiled

Support Options:

-identify	print tool identification, version and active options
-help	print this information and do nothing else

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.

```
Usage: link_partition [ option ... ] unit_or_partition_name ...

Link one or more partitions.

Library Options:
   -path file_name{,file_name}        use given library search path.
   -library file_name                 use default path associated with the
                                      given library

Linkage Options:
   -adamain                           link with an Ada main program (default)
   -noadamain                         link without an Ada main program
   -andf                              produce an ANDF capsule
   -noandf                            produce an executable (default)
   -lock                              lock partition after writing
   -nolock                            do not lock partition after writing
                                      (default)
   -log                               output the elaboration order and selected
                                      run-time system
   -nolog                             do not display the elaboration order on
                                      standard output (default)
   -npx                               the target has an NPX floating point unit
                                      (default)
   -nonpx                             the target does not have an NPX floating
                                      point unit
   -objects file_name{,file_name}     additional object files or libraries
   -output file_name                  name of output file
   -nooutput                          use the standard name for the output file
                                      (default)
   -rts file                          the run-time system to use (one is
                                      selected by default)
   -norts                             link without an implied run-time system
   -save                              save the updated partition (default)
   -nosave                            do not save the updated partition
   -stop_before_link                  stop before linking and retain files
   -nostop_before_link                do not stop (default)
   -target_options "Text"             transfer options to the target linker
   -main_priority priority            the default priority of the main task
   -task_priority priority            the default priority of tasks being
                                      created
   -main_stack_size size_in_bytes     the main task stack size
   -task_stack_size size_in_bytes     the task stack default size
   -task_storage_size size_in_bytes   same as -task_stack_size
   -null_stack_size size_in_bytes     the null (idle) task stack size
   -reserve_stack_size size_in_bytes  the safety area size in each stack
   -task_attr_max_no number           maximum number of simultaneously active
                                      task attributes
```

COMPILATION SYSTEM OPTIONS AND LINKER OPTIONS

```
   -task_attr_size number             maximum size of task attributes in units
                                      of 32 bits
   -tasks number_of_tasks             maximum number of simultaneous tasks
   -template_file template            the linker template file to expand (one
                                      is selected by default)
   -timer float_value                 the timer resolution in seconds
   -time_slice float_value            the time_slice in seconds
   -ucc ucc_directory_name            the ucc to use (only relevant for bare
                                      systems)
   -noucc                             link without an implied ucc

Linkage Options(2):
   -ignore_errors                     try to link all partitions irrespective
                                      of errors
   -noignore_errors                   stop linking if errors are found (default)
   -instance_subunits                 only build instance-subunits, don't
                                      actually link the partition(s)
   -noinstance_subunits               build instance-subunits and link
                                      partition(s) (default)

Code Generation Options:
   -check check_name=on|off [,...]    suppress or unsuppress specified runtime
                                      checks. Valid check_names: All, Access,
                                      Accessibility, Discriminant, Division,
                                      Elaboration, Index, Length, Overflow,
                                      Range, Storage, Tag
   -nocheck                           suppress all runtime checks
   -core_andf                         generate ANDF 4.0
   -extended_andf                     generate ANDF 4.1 (default)
   -debug                             generate debug information
   -nodebug                           don't generate debug information (default)
   -install                           call ANDF installer (default)
   -noinstall                         don't call ANDF installer
   -keep_primary                      keep the primary ANDF capsule in the
                                      library (default)
   -nokeep_primary                    remove the primary ANDF capsule
   -optimize class_name=on|off [,...] perform additional optimizations. Valid
                                      class_names: All, Check, Speed, Size
   -nooptimize                        don't perform additional optimizations
                                      (default)

Compilation Options(1):
   -progress                          print progress reports
   -noprogress                        don't print progress reports (default)
   -filename_generation method
      append                          append to source file name
      change_suffix                   change suffix of source file name
                                      (default)
   -warnings                          display warnings (default)
   -nowarnings                        don't display warnings

Support Options:
   -identify                          print tool identification, version and
                                      active options
   -help                              print this information and do nothing else
```

The following pages contain a sample test script.

As an example, the commands for handling of the first class C test, c23001a is shown.

For the Pentium PC target, the environment variable

```
SCORE_UCC_FM586
```

Is assigned the value "ucc_pc". The compiler system then chooses the appropriate board configuration code (UCC code) and the test driver chooses the appropriate ports for the target board.

Given that a library, support.alb, exists with the support units compiled into it, a new library, test.alb, is created with visibility to support.alb, which again gives visibility to the so-called root library that contains all predefined units (the names support.alb and test.alb are chosen here for being short and descriptive, these names are not the ones actually used when performing the test under the DDC-I test driver):

```
create_library -parent support.alb test.alb
ADA_LIBRARY_FM586=test.alb
export ADA_LIBRARY_FM586
```

Remember that $ADA_LIBRARY_FM586 is used as the default library in compilations and linkings. Therefore, the library name is not stated explicitly when invoking these tools. Next the source is compiled:

```
ada_fm586 -nowarnings -list ./src_fm586/acvc2_2/c2/c23001a.ada
```

and the partition is linked:

```
link_partition_fm586 -nowarnings -objects cd300051.o c23001a
```

the executable file is downloaded over ethernet and executed:

```
load_fm586 c23001a
```

The above commands are not executed directly as is. They are performed by a test driver that executes them by use of various scripts and applications that ensure that:

- the source file extension (.ada, .a, .am, .adt, .dep) is found by lookup in the source directory for each file compiled,
- a looping process is killed after a given time,
- limits on system resources are setup for all processes,
- the exit code of the tools are checked against expectations,
- output is put into log-files and compared against reference material,
- listing files are compared with reference material for e28002b and e28005d,
- libraries are (re)created at appropriate times, e.g. when it has to be ensured that a configuration pragma must only have effect on the test it is meant for,
- information on which tests pass and which fail is collected and displayed.

For example, with the sample commands for handling of c23001a given above:

1) the compilation of the source file is performed under the following constraints:
   - 30 seconds CPU time,
   - 300 seconds elapsed time,
   - created files smaller than 10000 512-byte blocks,

2) after compiling of the source file it is checked that:
   - the compiler exit code is 0,
   - no output is generated to standard output (listing file may be created, of course),
   - no error-message file is generated,

3) linking is performed under the following constraints:
   - 30 seconds CPU time,
   - 300 seconds elapsed time,
   - created files smaller than 10000 512-byte blocks,

4) after linking it is checked that:
   - the linker exit code is 0 (except for eb4011a, eb4012a and eb4014a),
   - no output is generated to standard output,

5) running the executable is performed under the following constraints:
   - 600 seconds CPU time,
   - 600 seconds elapsed time,
   - created files smaller than 10000 512-byte blocks,
   - maximum heap size 1000000 bytes,

6) after running the executable it is checked that
   - the exit code is 0,
   - the output is identical to a result that has previously been accepted manually. Only allowed deviation is the date displayed by Report.Test.

7) finally, the following cleanup actions are performed:
   - remove the executable file.

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 : Rbrukardt@bix.com

POINTS OF CONTACT

For technical information about this Ada processor, contact:

For the US:

Gary Palangian
400 N. 5th Street
Suite 1050
Phoenix Arizona 85004
USA
(602) 275-7172
support@ddci.com

For the rest of the world:

Thomas E. Hansen
DDC-I
Gl. Lundtoftevej 1B
DK-2800 Lyngby
Denmark
+45 45 87 11 44
teh@ddci.dk

For sales information about this Ada processor, contact:

For the US:

Jennifer Sanchez
DDC-I Inc.
400 N. 5th Street
Suite 1050
Phoenix Arizona 85004
USA
(602) 275-7172
sales@ddci.com

For the rest of the world:

Poul H. Munch
DDC-I
Gl. Lundtoftevej 1B
DK-2800 Lyngby
Denmark
+45 45 87 11 44
sales@ddci.dk

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

[Pro99] Operating Procedures for Ada Conformity Assessment, Version 2.0, Ada Resource Association, April 12,1999

[UG98] The Ada Compiler Validation Capability Version 2.2 User's Guide, EDS, September 25, 1998

REFERENCES

end of document

(REMOVE THIS PAGE)