THE
ADA CONFORMITY ASSESSMENT TEST SUITE
(ACATS)
VERSION 3.0
USER'S GUIDE

January 3, 2008

Prepared by:
Ada Conformity Assessment Authority
Randall L. Brukardt, Technical Agent
621 N. Sherman Ave., Suite B6
Madison, WI 53704
# Table of Contents

Table of Contents............................................................................................................ i  

## Section 1: Introduction

1.1 ACATS Purpose ........................................................................................................... 5  
1.2 ACATS Coverage of Ada ............................................................................................. 6  

## Section 2: Changes for ACATS 3.0

........................................................................................................................................... 9  

## Section 3: Test Objectives and Coverage

3.1 Test Objectives and Rules ............................................................................................. 11  
3.2 Coverage of the Ada Standard
   3.2.1 Coverage Documents .......................................................................................... 11  
   3.2.2 General Coverage Guidelines ......................................................................... 12  
   3.2.3 Coverage Guidelines for Specific Rule Categories ........................................... 13  

## Section 4: Configuration Information

4.1 Structure
   4.1.1 Physical Organization ....................................................................................... 15  
   4.1.2 Logical Organization ....................................................................................... 16  
   4.1.3 Legacy Tests ..................................................................................................... 16  
   4.1.4 Test Foundation Code ..................................................................................... 17  
   4.1.5 Special Core Tests .......................................................................................... 17  
   4.1.6 Foreign Language Code .................................................................................. 17  
4.2 Test Classes
   4.2.1 Class A ............................................................................................................. 18  
   4.2.2 Class B ............................................................................................................. 18  
   4.2.3 Class C ............................................................................................................. 18  
   4.2.4 Class D ............................................................................................................. 19  
   4.2.5 Class E ............................................................................................................. 19  
   4.2.6 Class L ............................................................................................................. 19  
   4.2.7 Foundation Code ............................................................................................ 20  
   4.2.8 Specialized Needs Annex Tests ...................................................................... 20  
4.3 Naming Convention
   4.3.1 Legacy Naming ................................................................................................ 20  
   4.3.2 Modern Naming .............................................................................................. 22  
   4.3.3 Multiple File Tests .......................................................................................... 24  
4.4 Test Program Format................................................................................................. 24  
4.5 General Standards ................................................................................................... 25  
4.6 Test Structure .......................................................................................................... 26  
4.7 Delivery Directory Structure .................................................................................... 27  
4.8 File Format ............................................................................................................. 28  

## Section 5: Using the ACATS

5.1 Installation of the ACATS Test Suite
   5.1.1 Contents of the ACATS Delivery ...................................................................... 30  
   5.1.2 Guide to Decompressing Files ......................................................................... 31  
   5.1.3 Files With Non-Graphic Characters ................................................................. 32  
5.2 Tailoring the ACATS Test Suite
   5.2.1 ImpDef Customization ..................................................................................... 34  

---

3 January 2008
ACATS 3.0 User's Guide

5.2.2 Macro Defs Customization ................................................................. 35
5.2.3 Packages SPRT13 and FCNDECL .................................................. 35
5.2.4 Modification of Package REPORT ................................................ 36
5.2.5 Allowed Test Modifications ........................................................... 36
5.3 Processing the Support Files .............................................................. 37
5.3.1 Support Files .................................................................................. 37
5.3.2 "CZ" Acceptance Tests ................................................................. 37
5.4 Establishing Command Scripts ......................................................... 38
5.4.1 Command Scripts .......................................................................... 38
5.4.2 Dependencies ................................................................................ 38
5.5 Processing ACATS Tests ................................................................. 38
5.5.1 Required Tests ................................................................................ 39
5.5.2 Test Partitions ................................................................................ 39
5.5.3 Bundling Test Programs ................................................................. 39
5.5.4 Processing that may be Omitted ..................................................... 39
5.5.5 Tests with Special Processing Requirements ................................. 40
5.5.6 Focus on Specific Areas ................................................................. 44
5.6 Grading Test Results ........................................................................ 45
5.6.1 Expected Results for Executable Tests .......................................... 45
5.6.2 Expected Results for Class B Tests ................................................ 46
5.6.3 Expected Results for Class L Tests ................................................ 46
5.6.4 Inapplicable Tests .......................................................................... 47
5.6.5 Withdrawn Tests ............................................................................ 47
5.7 Addressing Problems or Issues ......................................................... 47
5.7.1 Typical Issues ................................................................................ 47
5.7.2 Deviation from Expected Results - Petition & Review .................... 48
5.8 Reprocessing and Regrading ............................................................. 48

Annex A: Version Description ................................................................. 49
A.1 Core Test Files .................................................................................. 49
A.2 Specialized Needs Annex Test Files ................................................ 70
A.3 Foundation Code Files ..................................................................... 72
A.4 Documentation Files ........................................................................ 72
A.5 Other Files ....................................................................................... 73
A.5.1 List of ACATS 3.0 Files ............................................................... 73
A.5.2 Support Units Referenced by Many Tests ..................................... 73
A.5.3 Preprocessing Tools and Data ...................................................... 73
A.5.4 Tests for Reporting Code ............................................................ 74
A.6 Tests With Special Requirements ..................................................... 74
A.7 Test Files Added Since ACATS 2.6 .................................................... 74
A.8 Documentation Files Added Since ACATS 2.6 ............................... 75
A.9 Test Files Modified Since ACATS 2.6 ............................................. 76
A.10 Support Files Modified Since ACATS 2.6 ....................................... 77
A.11 Test Files Deleted Since ACATS 2.6 ............................................... 77
A.12 Other Files Deleted Since ACATS 2.6 ............................................. 77

Annex B: Parameterization Files ............................................................ 79
B.1 Macro Substitution File .................................................................... 79
B.2 Macro Substitution Tests ................................................................. 80
B.3 Package ImpDef and Its Children .................................................... 80
Annex C: Results of CZ Tests
C.1 Sample Output From CZ0004
C.2 Sample Output From CZ1101A
C.3 Sample Output From CZ1102A
C.4 Sample Output From CZ1103A
C.4.1 Output When External Files Are Supported
C.4.2 Output When External Files Are Not Supported
Annex D: Test Applicability Criteria
D.1 Compile-Time Inapplicability
D.1.1 Type Short_Integer
D.1.2 Type Long_Integer
D.1.3 Other Predefined Integer Types
D.1.4 Fixed Point Restrictions
D.1.5 Non-binary Values of 'Small
D.1.6 Compiler Rejection of Supposedly Static Expression
D.1.7 Machine Code Insertions
D.1.8 Illegal External File Names
D.1.9 Decimal Types
D.1.10 Instantiation of Sequential_IO with indefinite types
D.1.11 Special Handling Tests
D.2 Reported Inapplicability
D.2.1 Value of Machine_Overflows is False
D.2.2 System.Max_Digits
D.2.3 Floating Point Overflow
D.2.4 Type Duration
D.2.5 Text Files (Non-supported Features)
D.2.6 Text Files (Supported Features)
D.2.7 Sequential Files (Non-supported Features)
D.2.8 Sequential Files (Supported Features)
D.2.9 Direct Files (Non-supported Features)
D.2.10 Direct Files (Supported Features)
D.2.11 Stream Files (Non-supported Features)
D.2.12 Wide Text Files (Non-supported Features)
D.2.13 File I/O Tests
D.2.14 Memory for Allocated Objects
D.2.15 Task Attributes
D.2.16 Reserved Interrupts
D.2.17 Multiprocessor Systems
D.2.18 Non-binary Machine Radix
Annex E: Guidelines for Test Development
Annex F: Definitions
References
Index
Section 1: Introduction

The Ada Conformity Assessment Test Suite (ACATS) provides the official tests used to check conformity of an Ada implementation with the Ada programming language standard (ANSI/ISO/IEC 8652:1995 and later corrigenda and amendments). The ACATS User's Guide is part of the ACATS and is distributed with the test programs and testing support packages. It explains the contents and use of the test suite.

The ACATS is an important part of the conformity assessment process described in ISO/IEC-18009, Ada: Conformity of a Language Processor [ISO99]. This standard provides a framework for testing language processors, providing a stable and reproducible basis for testing. The Ada Resource Association (ARA) has sponsored an instantiation of that process since October 1998. The process is managed by the Ada Conformity Assessment Authority (ACAA).

Prior to the ISO standard, the U.S. Department of Defense sponsored a similar conformity assessment process under the Ada Joint Program Office (AJPO). The test suite for that process was known as the Ada Compiler Validation Capability (ACVC). The AJPO developed ACVC versions based on ANSI/MIL-STD-1815A-1983, ISO/8652:1987 (Ada 83), which were numbered 1.x where x ranged from 1 to 11. It later developed ACVC versions based on ANSI/ISO/IEC 8652:1995 ([Ada95]), numbered 2.0, 2.0.1, 2.1, and 2.2.

When the ACAA took over Ada conformity assessment, it adopted the ACVC as the basis for its test suite. The ACAA determined to continue to use the same version numbering for the test suite in order to avoid confusion. The version of the ACVC current at the time (2.1) was initially used as ACATS 2.1. Later, the already developed but unreleased ACVC 2.2 was released and used as ACATS 2.2. The ACAA later released ACATS 2.3, ACATS 2.4, ACATS 2.5, and then ACATS 2.6 to include maintenance changes and a few new tests.

This version of the ACATS is version 3.0. As with ACATS 2.3 and later, this version was completely developed under the auspices of the ACAA. ACATS 3.0 contains test programs to check for conformity to new language features defined in ISO/IEC 8652:1995/AMD 1:2007 ([Amend1]), as well as test programs to check for conformity to language features defined in earlier versions of Ada, including [Ada95] and [Ada83]. Subsequent maintenance or enhancement versions of the suite, if they are required, will be numbered 3.1, etc.

The ACATS User's Guide describes the set of ACATS tests and how they are to be used in preparation for conformity assessment. The formal procedures for conformity assessment are described in [Pro01], and the rules in that document govern all conformity assessments, notwithstanding anything in this document that may be interpreted differently. Moreover, this guide does not discuss specific requirements on processing of the ACATS test suite, or submission and grading of results that an Ada Conformity Assessment Laboratory (ACAL) may impose.

The User's Guide is intended to be used by compiler implementers, software developers who maintain a version of the ACATS as a quality control or software acceptance tool, and third-party testers (e.g., Ada Conformity Assessment Laboratories).

Section 2 of the User's Guide for ACATS 3.0 summarizes the changes between ACATS 2.6 and ACATS 3.0. Section 3 describes test objectives and their relationship to ACATS tests and to the rules of the Ada Standards documents. Section 4 describes the configuration of the ACATS, including a description of the ACATS software and delivery files. Section 5 provides step-by-step instructions for installing and using the test programs and test support packages, and for grading test results. The appendices include other information that characterizes the ACATS 3.0 release, along with information on test construction.
Refer to Annex F and Section 5.6 for the definition of an acceptable result and the rules for grading ACATS 3.0 test program results. Section 5.7.2 provides instructions for submitting a petition against a test program if a user believes that a deviation from the acceptable results for a given test program is in fact conforming behavior.

The ACATS test suite is available from any ACAL and from the Ada Information Clearinghouse (sponsored by the ARA). See http://www.adaic.org/compilers/testing.html.

1.1 ACATS Purpose

The purpose of the ACATS is to check whether an Ada compilation system is a conforming implementation, i.e., whether it produces an acceptable result for every applicable test.

A fundamental goal of conformity assessment (validation) is to promote Ada software portability by ensuring consistent processing of Ada language features as prescribed by the Ada Standard documents ([Ada95], [TC1], and [Amend1]). ACATS tests use language features in contexts and idioms expected in production software. While they exercise a wide range of language feature uses, they do not and cannot include examples of all possible feature uses and interactions.

It is important to recognize that the ACATS tests do not guarantee compiler correctness. A compilation system that correctly processes the ACATS tests is not thereby deemed error-free, nor is it thereby deemed capable of correctly processing all software that is submitted to it.

The ACATS tests do not test the quality of an Ada implementation. In particular, ACATS test do not check or report performance parameters (e.g., compile-time capacities or run-time speed). They do not check or report for characteristics such as the presence and effectiveness of compiler optimization. They do not investigate or report compiler or implementation choices in cases where the standard allows options.

1.2 ACATS Coverage of Ada

The ACATS needs to test as many rules as possible in order to meet the goal of enhancing Ada software portability. After all, a rule that is not tested is far more likely to be incorrectly implemented than one that is tested.

Therefore the ACATS strives for complete coverage of the standard. Complete coverage means that every rule in the Ada standard has one or more associated tests that ensure that the rule is implemented properly.

Complete coverage is especially important for legality rules and runtime checks. It is easy for implementers to miss these rules, as their compiler may do something useful in the absence of the checks. But allowing such incorrect code can be a major portability problem when a program is moved to a compiler (including a later version of the same compiler) that properly implements the checks.

Of course, complete coverage does not mean that every sentence in the Ada standard has an associated test. There are many lines in the standard that are not rules at all, such as notes and examples. There are also many lines in the standard that are not testable; documentation requirements are but one example. Finally, there are rules in the standard that could be tested, but such tests are unlikely to find errors; for instance, most tests for illegal syntax are in this category as the test would need to guess what error might be made.
Complete coverage is a goal for the ACATS; it is not expected to be achieved in near future. Thus the test coverage analysis of the ACATS is a set of living documents, which will be updated with new information and tests with each new ACATS version.

Coverage testing will generally not test combinations of features, so problems that only manifest themselves in such combinations will not be detected. Tests designed primarily to cover language rules are most useful to prevent gross errors in implementations (such as forgetting to implement checks or features). As such, the ACATS also supplements those tests with tests written to emulate the patterns of use of Ada features. Such tests provide tests of common combinations of features, and ensure that common idioms are implemented properly.

A detailed description of how coverage is determined for the Ada standard can be found in Section 3.2.
Section 2: Changes for ACATS 3.0

Version 3.0 of the ACATS is the first version to test the changes brought to Ada by [Amend1]. It includes nearly 80 new tests to check features including interface types, limited with clauses, private with clauses, overriding indicators, <> in aggregates, and expanded use of limited types.

In addition, many tests that were incompatible with [Amend1] were modified or removed.

Finally, a few tests that contained errors have been modified.

See Annex A, “Version Description” for lists of added, deleted and modified tests.
Section 3: Test Objectives and Coverage

Each ACATS test tests one or more test objectives. Test objectives ought to be relatable to rules given in the Ada Standard documents ([Ada95], [TC1], and [Amend1]).

The test objectives of modern ACATS tests are gathered into a Test Objectives Document. Each modern test is listed along with the objectives that it tests (legacy tests aren't included as they use a different form of objective). This document provides easier searching for particular objectives than a raw search of all of the test files.

This section provides information on how new test objectives are constructed and how the adequacy of objectives is checked.

3.1 Test Objectives and Rules

A test objective should relate as directly as possible to the rules of Ada. Ideally, a test objective will test an individual rule of Ada. Rule means whatever set of sentences makes up a testable statement. A rule may be as small as a single line, or spread across multiple paragraphs of the standards.

For instance, [Ada95] contains the rule

> The implicit declaration of the limited view of a library package forms an (implicit) compilation unit whose context_clause is empty.

in subclause 10.1.1. The corresponding test objective is

> Check that the context clause of a limited view is empty.

However, virtually all rules require the use of other rules (such as those for definitions of terms) in order to be testable. For instance, testing the above rule needs the definition of terms like "limited view" and "context_clause" (among others) in order for a test to be constructed.

As such, a test objective will generally apply to multiple Ada rules. Typically, it will be recorded for a specific rule. For the purposes of deciding if a rule is appropriately tested, it is necessary to study the ACATS coverage documents (see 3.2.1).

3.2 Coverage of the Ada Standard

The Ada Standard documents ([Ada95], [TC1], and [Amend1]) include many hundreds of rules. The ACATS Coverage Documents (see 3.2.1) record the mapping of test objectives and tests to Ada rules, making it possible to determine if individual rules are adequately tested. The following clauses discuss these documents and the rules by which they are constructed.

3.2.1 Coverage Documents

ACVC 2.0 originated coverage documentation for what is now the ACATS. That coverage document mapped tests to paragraph numbers in the [Ada95] Standard. Over time, a number of deficiencies in that approach have been identified:

- Where a paragraph contains multiple rules, it is not possible to tell if all of the rules are tested. Indeed, a number of instances where important rules were untested have been identified.
- Since there is not an indication of the type of rule represented by a paragraph, it is not obvious whether the rule is appropriately tested. For instance, a Legality Rule tested only with C-Tests is not appropriately tested (see 3.2.3 for more on this topic).
• Rules not modified in [Ada95] were not considered. However, sometimes additional test objectives are needed for existing rules because of other changes to the language. Moreover, there is no determination of whether the legacy tests adequately tested the original rule.

For these reasons, with ACATS 3.0, the development of new and much more detailed coverage documentation was started. The new documentation considers each sentence of the standard individually if necessary, listing the test objectives for each sentence, and documenting why no objectives are needed if none are provided.

It also lists tests for each objective (including Legacy tests), testing priorities for untested objectives, notes on each objective (including any untested cases), and any additional notes needed.

There is also a summary document that summarizes totals for important metrics for the coverage documents as a whole.

These documents are provided in Adobe Portable Document Format (PDF).

For ACATS 3.0, the following clauses of the Ada Standards are included in the new coverage documents:

- 3.9.3 through 3.9.4
- 4.1.3 through 4.4
- 6.5 through 6.7
- 7.4 through 7.6.1
- 8.3.1 through 8.5
- Section 10 (10 through 10.2.1)

These clauses were selected because of their importance in rule changes in [Amend1]

The original coverage document is also included with the ACATS for those portions of the Ada Standards that have not yet had new coverage documents constructed. This document is of primary use for features of [Ada95], as it does not cover Legacy tests nor any features added or modified by [Amend1]. As noted above, it is being replaced, so it will disappear completely in a future version of the ACATS.

### 3.2.2 General Coverage Guidelines

As noted in Section 1.2, the ACATS strives for complete coverage of the Ada standard. However, complete coverage does not mean that every sentence in the standard has an associated test objective and test. Not all rules are testable; one important way to determine testability of a rule is to check its category. The next clause (3.2.3) will discuss how the category of a rule affects its mapping to test objectives.

There are also additional considerations that don't apply to specific rule categories.

Text sometimes contains definitions (especially in Static Semantics rules). Definitions are hard to handle, because they usually are not testable by themselves. The definition has to be used in another rule to make them visible. For instance, a categorization rule such as "something is either this, that, or fuzzy" cannot be tested by itself. "Something" has to be used in some other rule in order for the categorization to matter. That being the case, it usually makes sense to test the definition as part of the other rule(s). Such rules are marked as having a kind of "Widely Used" or "Subpart" in the coverage documentation. "Subpart" rules are tested as part of specific other objectives, while "Widely Used" rules are thought to be tested by many tests indirectly (and no attempt is made to verify that). In some cases, especially those that otherwise would have a combinatorial explosion, it may make more sense to test the definition as directly as possible (in which case objectives will be assigned to the definition).

Text sometimes includes sentences marked as redundant by the AARM. Such sentences should be given normatively elsewhere, and the testing should be done in the place where they are normatively given. The AARM often will indicate this place.
Combinational explosion is always a problem for visibility rules. Rules that define the scope or visibility of something should always be tested in place (even though other rules will need to be used to accomplish that).

### 3.2.3 Coverage Guidelines for Specific Rule Categories

The Ada standard is divided into various categories of rules. These rule categories have different requirements and thus differ in what is the appropriate testing philosophy. This translates into different types of test objectives for each rule category.

Uncategorized text:
This is generally introductory text, and does not require any testing. This text sometimes includes definitions; these are handled as described in General.

Syntax:
Testing that legal syntax is implemented is accomplished by all of the other kinds of tests, especially by the user-oriented tests. Generally, tests for illegal syntax have a very low value. Most compilers use a syntax that is similar to that of the standard, and many use parsers generated by tools, and thus the likelihood of errors is low. Moreover, the number of possible bugs is essentially unlimited. There are two cases where testing is warranted: First, rules that are given in text form rather than BNF are treated as legality rules (see below). Second, the Ada syntax grammar is not directly usable by either of the common technologies for parsers (LL/recursive descent, and LR/LALR). In cases where the grammar used by a compiler must necessarily be more general than that given in the standard, the additional requirements are treated as legality rules.

Name Resolution Rules:
Name resolution rules should usually have a B-Test that checks that too much information is not used for resolution. (For instance, whether a type is limited or non-limited should generally not be used. Another example is whether an access type is pool-specific or general access usually should not be used.) It also may have a C-Test to check that legal cases are allowed; but this objective is often handled by tests for other test objectives (if that's the case, a note should point out where they are tested).

Legality Rules:
Legality rules should almost always have an associated B-Test that checks that illegal cases are detected. They may also have a C-Test to check that legal cases are allowed; again, this is often handled by tests for other test objectives. (They also can be covered in the B-Test with "OK" cases.)

Static Semantics:
Many Static Semantic rules are definitions, and as such can be handled as described in 3.2.2, “General Coverage Guidelines”.

Many other static semantic rules are miscategorized by the standard. For instance, most of the library packages are defined as static semantics, even though most of the rules are really dynamic. These should be tested as appropriate for the correct category.

Post-Compilation Rules:
A post-compilation rule should always have L-Tests to check that the check is made in illegal cases. These checks are especially likely to be omitted (because they're often complex to implement), so these tests should have a high priority. In some cases, a C-Test may be needed to ensure that legal cases work and the check does not go too far. Often that will be covered by other tests, so a separate test is not needed (a note should be provided to show where it is covered).

Dynamic Semantics:
Dynamic semantics rules can be divided into two sub-categories.
First, these rules can define run-time checks. These checks should have C-Tests that verify that the check is made and the appropriate exception is raised. Careful: tests like this have to be carefully constructed to avoid running afoul of 11.6 permissions.

Otherwise, these rules can define the normal operation of a construct. These should have user-oriented C-Tests. By making these tests user-oriented, the ACATS tests the normal usage of these features, not unusual corner-cases.

One special case: rules that say the execution of something "has no effect" can't be usefully tested. Testing that nothing happens is not interesting. And testing for the absence of an effect requires guessing incorrect effects that might have occurred (which is unlikely to actually detect anything), and thus any such test would have a very low value.

Bounded (Run-Time) Errors:
These usually require C-Tests, but not always. If the bound includes "work normally", then the bounded error is not usefully testable, since the ACATS does not test implementation choices. Whether something works or raises Program_Error is not interesting. In other cases, however, since there is a bound on the behavior, it can be useful to ensure that one of the prescribed results happens rather than havoc.

Erroneous Execution:
This is not testable. Since the language allows anything to happen, there is no useful information to be gained from including this in a test. Indeed, it's important to avoid any such case in an executable test.

Implementation Requirements:
These usually are dynamic requirements that should be tested like the appropriate dynamic semantics rules. Otherwise, they should be tested like a legality rule.

Implementation Permissions:
Usually permissions are not testable. Since the ACATS is not testing the quality of implementations, the choices made by an implementation are not appropriate things to test. So a test should be created only if there is value in testing for implementations exceeding the permission.

Implementation Advice:
Advice is usually not testable. Again, since the ACATS is not testing quality of implementations, the choices made by an implementation are not appropriate things to test. Moreover, advice need not be followed, and other implementation choices can be made. Usually, if there are bounds to what is acceptable, they are covered by other rules, and thus the tests would be there.

Documentation Requirements and Metrics:
No tests are required for documentation requirements. Documentation existence could be part of the conformity assessment process, but in any case it is outside of the scope of the ACATS (which is testing the implementation, not the documentation).

Notes and Examples:
These are non-normative text, and do not require any testing.
Section 4: Configuration Information

This section describes the physical and logical structure of the ACATS delivery, and it describes the test classes, naming conventions used, test program format, test structure, delivery structure, and file format.

ACATS 3.0 is a revision of ACATS 2.6, and has a similar delivery structure. The support tools are unchanged, except for updating header comments and version identification, and the removal of an obsolete character-set processing tool.

The test suite does not provide tools or scripts that can be used to manage complete test processing, since such tools are normally specific to particular implementations and host systems.

4.1 Structure

The ACATS 3.0 test software includes test code that exercises specific Ada features, foundation code (used by multiple tests), support code (used to generate test results), and tool code (used to build tools necessary to customize ACATS tests). The suite includes tests for the core language and tests for the Specialized Needs Annexes. The following table summarizes the number of tests and files in the ACATS suite.

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Core Tests</th>
<th>SNA Tests</th>
<th>Foundations</th>
<th>Docs</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Files</td>
<td>4660</td>
<td>4139</td>
<td>248</td>
<td>50</td>
<td>169</td>
<td>22</td>
</tr>
<tr>
<td>Number of Tests</td>
<td>3771</td>
<td>3581</td>
<td>190</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Others consists of:

1. List of all files
2. Code that is referenced by tests
3. Code and data used for preprocessing tests to insert implementation specific information
4. Test routines for reporting code ("CZ" tests)

The delivery structure of the test suite is described in Section 4.7.

4.1.1 Physical Organization

The preceeding table summarizes the number of files that compose ACATS 3.0. In addition to files containing test code proper, the ACATS 3.0 test suite includes various support files.

Note that the number of files containing test code is larger than the number of tests in the ACATS suite because some tests use code included in separate files.

A file name consists of a name plus an extension. Multiple files that contain code used by a single test have related names. File names are the same as that of the test contained in the file when possible. File names conform to MS-DOS naming conventions; therefore they may be shorter than the software name because of file name length restrictions (e.g., enumcheck rather than enumcheck). File (and test) names follow conventions that indicate their function in the test suite; naming conventions are explained in
Section 4.3. The files are organized into distinct directories and subdirectories based on their function in the test suite. The directory organization is explained in Section 4.7.

The ACATS is available to the general public from an ACAL or on the Internet. Links to the ACATS distribution can be found on the ACAA’s ACATS page:


Note that the ACATS files are available in both compressed Unix tar and DOS zipped formats. Section 5.1.2 provides a discussion of techniques to convert these files to a usable format.

### 4.1.2 Logical Organization

The table summarizes the number of tests that check the conformance of an Ada implementation to the core language and conformance to the Specialized Needs Annexes of Ada.

Core tests apply to all implementations. Specialized Needs Annex tests are not required for any implementation. Tests for a given Specialized Needs Annex may be processed by implementations that claim implementation of that annex.

In general, no test result depends on the processing or the result of any other test. Exceptions are noted in Section 5.4.2. No annex test depends on the implementation of any other annex, except possibly in cases where one annex specifically depends on another in Ada (e.g., no test for the Information Processing Annex uses features from any other annex, however Real Time Annex and Distributed Processing tests may depend on Systems Programming Annex features). (There is a single exception to this rule: see Section 5.5.5.3.) Annex tests may use any core feature.

Tests may be created from one or more compilation units. If a test consists of a single compilation unit (a main subprogram only), the test code will be contained in a single file. Tests built from more than one compilation unit may require multiple files. Moreover, some compilation units, called foundation code, may be used by more than one test. Even in these cases, the resulting tests are strictly independent: if test A and test B use the same foundation code, the results of processing (and running, if appropriate) A have no effect on the results of processing (and running, if appropriate) B. Foundation code is more fully explained in Section 4.1.4.

Tests are named using conventions that provide (limited) information about the test. The test naming conventions are explained in Section 4.3. Each test belongs to a single test class that indicates whether it is or is not an executable test. Test classes are explained in Section 4.2.

In addition to test code and foundation code, there is code on which many or all of the executable tests in the suite depend (e.g., package Report, package ImpDef, package TCTouch). Some of this code must be customized to each implementation. There is also code that must be used to build support tools used to customize the suite of tests to an implementation. The customization process is described in Section 5.2.

### 4.1.3 Legacy Tests

Legacy tests are tests that were included in ACVC 1.12 that have been incorporated into later ACVC and ACATS versions. These tests check only language features that are common to all versions of Ada. The vast majority of these tests came unmodified from the ACVC 1.12 suite. Some tests were modified to check for the correct implementation of Ada rules in cases where language rules changed from [Ada83].

Unlike more modern tests, legacy tests use an ALL CAPITALS programming style which is unusual and hard to read (4.5). They also use the naming conventions of early ACVC versions (see 4.3.1). Modern tests (those created for the ACATS more recently than ACVC 1.12) use a more typical Mixed Case programming style (see 4.5 and use a more flexible naming scheme (see 4.3.2).
4.1.4 Test Foundation Code

Some tests use foundation code. Foundation code is reusable across multiple tests that are themselves independent of each other. It is intended to be compiled and included in an environment as part of the compilation process of a test. If the test is executable, the foundation code must be bound with all other code for the test prior to execution.

Foundation code is always expected to compile successfully; it is never expected to be run by itself. Foundation code is not, in and of itself, a test, and is therefore not characterized by a test class (see 4.2). One may think of it as providing some utility definitions and routines to a number of different tests. Names of foundation units (and therefore names of files containing foundation code) are distinguished as described in Naming Convention, Section 4.3.

4.1.5 Special Core Tests

This section identifies tests that appear in the Core (since their requirements are enunciated there) but that may be graded as non-supported for implementations not claiming support of certain Specialized Needs Annexes.

Annex C Requirements

Section 13 of the Ada Standard includes implementation advice paragraphs. The ACATS does not require implementations to conform to those paragraphs unless they claim support for Annex C, Systems Programming (because of C.2(2): "The implementation shall support at least the functionality defined by the recommended levels of support in Section 13.")

Tests that check conformance to the implementation advice are listed below:

| CD10001 | CD30003 | CD30004 | CD90001 |
| CD20001 | CD30005 | CD40001 |
| CD30001 | CD33001 | CD72A01 |
| CD30002 | CD33002 | CD72A02 |

Implementations that claim support for Annex C are required to process and pass the tests listed above. Implementations that do not claim support for Annex C are still required to process these tests. Such implementations may reject the lines marked with the special comment -- ANX-C RQMT, in which case the test will be graded as "unsupported". If an implementation accepts such lines in one of these tests, then the test must be bound (linked) and executed, with a passed or not_applicable result.

4.1.6 Foreign Language Code

Several tests for Annex B features (and one Section 13 test) include files containing non-Ada code (Fortran, C, Cobol). These tests must be compiled, bound, and run by implementations that support foreign language interfaces to the respective non-Ada language. The foreign language code uses only the most basic language semantics and should be compilable by all Fortran, C, and Cobol compilers, respectively. In cases where a foreign language does not accept the code as provided, modifications are allowable. See Section 5.2.5.

Files that contain foreign code are identified by a special file extension. See Section 4.3.2.

The tests that include Fortran code are: CXB5004 and CXB5005

The tests that include C code are: CXB3004, CXB3006, CXB3013 and CD30005

The test that includes Cobol code is: CXB4009
4.2 Test Classes

There are six different classes of ACATS tests, reflecting different testing requirements of language conformity testing. Each test belongs to exactly one of the six classes, and its membership is encoded in the test name, as explained later. The purpose and nature of each test category is explained below. The test classifications provide an initial indication of the criteria that are used to determine whether a test has been passed or failed.

4.2.1 Class A

Class A tests check for acceptance (compilation) of language constructs that are expected to compile without error.

An implementation passes a class A test if the test compiles, binds, and executes reporting "PASSED". Any other behavior is a failure.

Only legacy tests are included in this class.

4.2.2 Class B

Class B tests check that illegal constructs are recognized and treated as fatal errors. They are not expected to successfully compile, bind, or execute. Lines that contain errors are marked -- ERROR: and generally include a brief description of the illegality on the same or following line. (The flag includes a final ":" so that search programs can easily distinguish it from other occurrences of the word "error" in the test code or documentation.) Some tests also mark some lines as -- OK, indicating that the line must not be flagged as an error.

An implementation passes a class B test if each indicated error in the test is detected and reported, and no other errors are reported. The test fails if one or more of the indicated errors are not reported, or if an error is reported that cannot be associated with one of the indicated errors. If the test structure is such that a compiler cannot recover sufficiently to identify all errors, it may be permissible to "split" the test program into separate units for re-processing (see Section 5.2.5 for instructions on modifying tests).

In some cases and for some constructs, compilers may adopt various error handling and reporting strategies. In cases where the test designers determined that an error might or might not be reported, but that an error report would be appropriate, the line is marked with -- OPTIONAL ERROR: or a similar phrase. In such cases, an implementation is allowed to report an error or fail to report an error without affecting the final grade of the test.

Similarly, in cases where the test designers determined that an error might be reported at one of several source locations, all such source locations are marked with -- POSSIBLE ERROR: and an indication of which error (if the test contains several) is expected. In such cases, an implementation is considered passing if it reports an error at any of the possible places for the error to be reported. It fails if no error is reported at any of the places.

4.2.3 Class C

Class C tests check that executable constructs are implemented correctly and produce expected results. These tests are expected to compile, bind, execute and report "PASSED" or "NOT-APPLICABLE". Each class C test reports "PASSED", "NOT-APPLICABLE", or "FAILED" based on the results of the conditions tested.
An implementation passes a class C test if it compiles, binds, executes, and reports "PASSED". It fails if it does not successfully compile or bind, if it fails to complete execution (hangs or crashes), if the reported result is "FAILED", or if it does not produce a complete output report.

The tests CZ1101A, CZ1102A, CZ1103A, and CZ00004 are treated separately, as described in Section 5.3.2.

4.2.4 Class D

Class D tests check that implementations perform exact arithmetic on large literal numbers. These tests are expected to compile, bind, execute and report "PASSED". Each test reports "PASSED" or "FAILED" based on the conditions tested. Some implementations may report errors at compile time for some of them, if the literal numbers exceed compiler limits.

An implementation passes a class D test if it compiles, binds, executes, and reports "PASSED". It passes if the compiler issues an appropriate error message because a capacity limit has been exceeded. It fails if it does not report "PASSED" unless a capacity limits is exceeded. It fails if it does not successfully compile (subject to the above caveat) or bind, if it fails to complete execution (hangs or crashes), if the reported result is "FAILED", or if it does not produce an output report or only partially produces one.

Only legacy tests are included in this class.

4.2.5 Class E

Class E tests check for constructs that may require inspection to verify. They have special grading criteria that are stated within the test source. They are generally expected to compile, bind and execute successfully, but some implementations may report errors at compile time for some tests. The "TENTATIVELY PASSED" message indicates special conditions that must be checked to determine whether the test is passed.

An implementation passes a class E test if it reports "TENTATIVELY PASSED", and the special conditions noted in the test are satisfied. It also passes if there is a compile time error reported that satisfies the special conditions. Class E tests fail if the grading criteria in the test source are not satisfied, or if they fail to complete execution (hang or crash), if the reported result is "FAILED", or if they do not produce a complete output report.

Only legacy tests are included in this class.

4.2.6 Class L

Class L tests check that all library unit dependences within a program are satisfied before the program can be bound and executed, that circularity among units is detected, or that pragmas that apply to an entire partition are correctly processed. These tests are normally expected to compile successfully but not to bind or execute. Some implementations may report errors at compile time; potentially illegal constructs are flagged with "-- ERROR:". Some class L tests indicate where bind errors are expected. Successful processing does not require that a binder match error messages with these indications.

An implementation passes a class L test if it does not successfully complete the bind phase. It passes a class L test if it detects an error and issues a compile time error message. It fails if the test successfully binds and/or begins execution. An L test need not report "FAILED" (although many do if they execute).

As with B-tests, the test designers determined that some constructs may or may not generate an error report, and that either behavior would be appropriate. Such lines are marked with "-- OPTIONAL
ERROR." In such cases, an implementation is allowed to report an error or fail to report an error. If an error is reported at compile time, the binder need not be invoked. If no errors are reported at compile time, the binder must be invoked and must not successfully complete the bind phase (as indicated by the inability to begin execution).

4.2.7 Foundation Code

Files containing foundation code are named using the regular test name conventions (see Section 4.3). It may appear from their names that they represent class F tests. There is no such test class. Foundation code is only used to build other tests, so foundation units are not graded. However, if a foundation unit fails to compile, then the tests that depend on it cannot be compiled, and therefore will be graded as failed.

4.2.8 Specialized Needs Annex Tests

Specialized Needs Annex tests have no separate classifications and are classified in the same way as all other tests. There are Class B, Class C, and Class L SNA tests.

4.3 Naming Convention

This section describes the naming conventions used in ACATS 3.0, specifically as they apply to files. All file names are of the form <name>.<type>, where <type> is a one, two, or three character extension. File names indicate test class, compilation order (if applicable), and whether the test is implementation dependent or requires customization. When a test is included in a single file, <name> duplicates the test name. The same is true of a foundation. In multiple file tests, the first 7 characters of the file <name> are normally the same as the name of the test, however in some cases, the structure of the test requires that the file name be different from the Ada unit. The application of the conventions to tests is straightforward.

There are two different but similar naming conventions used in ACATS 3.0. Legacy tests use the naming conventions of early ACVC versions. Tests new since ACVC 1.12 use the modern convention. The conventions are consistently distinguishable at the 7th character of the name: legacy names have a letter in the 7th position, whereas newer (modern) names have a digit.

4.3.1 Legacy Naming

The name of a legacy test is composed of seven or eight characters. Each character position serves a specific purpose as described in the table below. The first column identifies the character position(s) starting from the left, the second column gives the kind of character allowed, and the third gives the corresponding meaning:

<table>
<thead>
<tr>
<th>Position</th>
<th>Kind</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Letter</td>
<td>Test class (see Section 4.2)</td>
</tr>
<tr>
<td>2</td>
<td>Hexadecimal</td>
<td>AIG chapter containing the test objective</td>
</tr>
<tr>
<td>3</td>
<td>Hexadecimal</td>
<td>Section within the above AIG chapter</td>
</tr>
<tr>
<td>4</td>
<td>Alphanumeric</td>
<td>Sub-section of the above AIG section</td>
</tr>
<tr>
<td>5-6</td>
<td>Decimal</td>
<td>Number of the test objective within the above sub-section</td>
</tr>
<tr>
<td>7</td>
<td>Letter</td>
<td>Letter identifier of the sub-objective of the above objective.</td>
</tr>
</tbody>
</table>
optional – Compilation sequence identifier — indicates the compilation order of multiple files that make up a single test. This position is used only if the test comprises multiple files.

The convention is illustrated below.

In multiple file tests, the intended order of compilation is indicated by a numeral at position 8. The first file to be compiled has '0', the second has '1', and so forth.

The chapter and section numbers of the AIG (ACVC Implementer's Guide) correspond to those in [Ada83].

Note: The use of a ninth character ('m') to indicate the file containing the main subprogram has been discontinued. The following table lists the files containing the main subprograms of the legacy multiple file tests.

<table>
<thead>
<tr>
<th>Legacy File Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD7001C0.ADA</td>
</tr>
<tr>
<td>AD7001D0.ADA</td>
</tr>
<tr>
<td>B38103C3.ADA</td>
</tr>
<tr>
<td>B38103E0.ADA</td>
</tr>
<tr>
<td>B63009C3.ADA</td>
</tr>
<tr>
<td>B73004B0.ADA</td>
</tr>
<tr>
<td>B83003B0.ADA</td>
</tr>
<tr>
<td>B83004B0.ADA</td>
</tr>
<tr>
<td>B83004C2.ADA</td>
</tr>
<tr>
<td>B83004D0.ADA</td>
</tr>
<tr>
<td>B83024F0.ADA</td>
</tr>
<tr>
<td>B8304B0.ADA</td>
</tr>
<tr>
<td>B86001A1.ADA</td>
</tr>
<tr>
<td>B95020B2.ADA</td>
</tr>
<tr>
<td>BA1001A0.ADA</td>
</tr>
<tr>
<td>BA1010F0.ADA</td>
</tr>
<tr>
<td>BA1010G0.ADA</td>
</tr>
<tr>
<td>BA1010H0.ADA</td>
</tr>
<tr>
<td>BA1010I0.ADA</td>
</tr>
<tr>
<td>BA1010J0.ADA</td>
</tr>
<tr>
<td>BA1010K0.ADA</td>
</tr>
<tr>
<td>BA1010L0.ADA</td>
</tr>
<tr>
<td>BA1010M0.ADA</td>
</tr>
<tr>
<td>BA1010N0.ADA</td>
</tr>
<tr>
<td>BA1010P0.ADA</td>
</tr>
<tr>
<td>BA1010Q0.ADA</td>
</tr>
<tr>
<td>BA1011A6.ADA</td>
</tr>
<tr>
<td>BA1011C2.ADA</td>
</tr>
<tr>
<td>BA1012A0.ADA</td>
</tr>
<tr>
<td>BA1012B4.ADA</td>
</tr>
<tr>
<td>BA1012E3.ADA</td>
</tr>
<tr>
<td>BA1022A6.ADA</td>
</tr>
<tr>
<td>BA3001A0.ADA</td>
</tr>
<tr>
<td>BA3001B0.ADA</td>
</tr>
<tr>
<td>BA3001C0.ADA</td>
</tr>
<tr>
<td>BA3001E0.ADA</td>
</tr>
<tr>
<td>BA3001F0.ADA</td>
</tr>
<tr>
<td>BA3006A6.ADA</td>
</tr>
<tr>
<td>BA3006B4.ADA</td>
</tr>
<tr>
<td>BA38108C1.ADA</td>
</tr>
<tr>
<td>BA38108D0.ADA</td>
</tr>
<tr>
<td>BA39006C0.ADA</td>
</tr>
<tr>
<td>BA39006F3.ADA</td>
</tr>
<tr>
<td>BA3906F3.ADA</td>
</tr>
<tr>
<td>BA3906F3.ADA</td>
</tr>
<tr>
<td>BA4005D0.ADA</td>
</tr>
<tr>
<td>BA4005D0.ADA</td>
</tr>
<tr>
<td>BA83024G0.ADA</td>
</tr>
<tr>
<td>BA83024E1.ADA</td>
</tr>
<tr>
<td>BA83F01C2.ADA</td>
</tr>
<tr>
<td>BA83F01D0.ADA</td>
</tr>
<tr>
<td>BA83F03C2.ADA</td>
</tr>
<tr>
<td>BA83F03D0.ADA</td>
</tr>
<tr>
<td>BA1110A1.ADA</td>
</tr>
<tr>
<td>BA1110A2.ADA</td>
</tr>
<tr>
<td>BA1110A2.ADA</td>
</tr>
</tbody>
</table>
The file name extension is three characters long. There are four extensions:

- **.ada** A file that contains only Ada code. It does not require any pre-processing to create a compilable test. It will be submitted directly to the implementation for determination of test results. All implementations must correctly process these tests.

- **.dep** A file that has a test involving implementation-dependent features of the language. These tests may not apply to all implementations.

- **.tst** A file that has "code" that is not quite Ada; it contains "macro" symbols to be replaced by implementation-dependent values, and it must be customized (macro expanded) to prepare it for compilation (see Section 5.2.2). Once customized, the resulting test must be processed as indicated by its class.

- **.adt** A file that has been modified by the macro processor. It contains only Ada code and may be submitted to the implementation for results. All implementations must correctly process these tests. There are no files in the ACATS distribution with this extension; they are only produced as the output of the macro processor.

Modern tests use different file name extensions (see 4.3.2).

*Note that legacy tests have not been renamed for ACATS 3.0. Since [Ada95] includes some organizational differences from [Ada83], this means that the name of a legacy test sometimes will not correspond to the clause of [Ada95] in which the tested feature is described.*

### 4.3.2 Modern Naming

The name of a modern ACATS test is composed of seven or eight characters. Foundation code has a name composed of seven characters. The use of each character position is described below. The first column indicates the character position(s) starting from the left, and the second column indicates the kind of character allowed, and the third column gives the corresponding meaning:

<table>
<thead>
<tr>
<th>Position</th>
<th>Kind</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Letter</td>
<td>Test class; foundations are marked 'F'</td>
</tr>
<tr>
<td></td>
<td>Alphanumeric</td>
<td>If other than an 'x', the section of the Ada Standard describing the feature under test. An 'x' indicates that the test includes one or more features from an annex of the Ada Standard</td>
</tr>
<tr>
<td>2</td>
<td>Alphanumeric</td>
<td>Core clause or annex letter identifier (either core or Specialized Needs Annex); clauses are a hexadecimal value</td>
</tr>
<tr>
<td>3</td>
<td>Alphanumeric</td>
<td>Sub-clause (if a core test), or clause (if an annex test); a number if less than 10, otherwise a letter with 10='A', 11='B', and so on</td>
</tr>
<tr>
<td>4</td>
<td>Alphanumeric</td>
<td>Foundation identifier (alphabetical, unless no foundation is required, in which case a '0')</td>
</tr>
</tbody>
</table>

### 4.3.1 Legacy Naming
Sequence number of this test in a series of tests for the same clause; foundation code will have "00".

optional – Compilation sequence identifier — indicates the suggested or required compilation order of multiple files that make up a single test (0 is compiled first). This position is used only if the test comprises multiple files.

The convention is illustrated below.

![Modern File Name Convention](image)

The file name extension is a one or two character file name extension. There are six extensions:

- **.a** A file that contains only Ada code (except for configuration pragmas in the case of some Specialized Needs Annex tests). It does not require any processing to prepare it for compilation (unless configuration pragmas must be handled separately). It is normally submitted directly to the implementation for determination of test results.

- **.am** A file that contains the main subprogram for a multi-file test. Generally, this extension is used for only one file of a test. In rare cases (some Annex E tests), a multi-file test may have more than one file containing a "main" subprogram; in such cases, the correct testing procedure is described in the Special Requirements section of the test prologue.

- **.au** A file that contains only Ada code that contains characters outside of the 7-bit ASCII character set. These files are provided in UTF-8 format with a starting byte-order mark. These tests can be converted to another format acceptable to the compiler if necessary; then the test must be compiled and run as all other tests of its test class.

- **.f7n** A file that contains Fortran language code and must be compiled by a Fortran compiler. These files are used by tests that check a foreign language interface to Fortran.

- **.c** A file that contains C language code and must be compiled by a C compiler. These files are used by tests that check a foreign language interface to C.

- **.cbl** A file that contains Cobol language code and must be compiled by a Cobol compiler. These files are used by tests that check a foreign language interface to Cobol.

A test that depends on foundation code has an alphabetic character in the fifth position of its name. The required foundation will have the same characters in the second through fifth positions of its name. For example, C123Axx depends on F123A00.
4.3.3 Multiple File Tests

When tests are contained in multiple files (i.e., compilation units are contained in different files), the file names are related. The first seven positions of the names of all the files (other than foundation files) comprised by a single test will be identical. The eighth position will provide a distinguishing alphanumeric which indicates the required compilation order. In legacy tests, the main subprogram is not indicated (see the table in section 4.3.1 for files containing main subprograms). For newer (modern) tests, the extension ".am" indicates the file with the main subprogram.

All tests apply the convention of naming the main subprogram the same as the file (excluding the file extension) plus, for legacy tests only, the letter 'm'. For example, the legacy test, C39006F, is contained in four files, named c39006f0.ada, c39006f1.ada, c39006f2.ada, and c39006f3.ada. The main subprogram of the test is contained in c39006f3.ada and is named C39006F3M. The test C390006 is also contained in four files, named c3900060.a, c3900061.a, c3900062.a, and c3900063.am. The main subprogram of the test is contained in c3900063.am and is named C3900063.

Unless otherwise required by a test objective, other library units in a test are named with the test name and a suffix. Typically, the suffix will be a number or an underscore followed by a few letters. Similarly, library units making up a foundation are usually named with the foundation name (or the foundation name and a suffix if there are multiple units in the foundation). This convention reduces name collisions with other tests and with implementation-defined units.

There are a small number of Specialized Needs Annex tests for the Distributed Processing Annex that require two active partitions and have two main subprograms. These tests have two files with the .am extension to signify the location of the (multiple) main subprograms.

4.4 Test Program Format

Each test file is composed of a test prologue, documenting the test, and the test code proper. All prologue lines are marked as comments. (The prologue in files containing non-Ada code is marked according to the comment conventions of the foreign language.)

The prologue for all tests is based on that of legacy tests. Legacy tests are generally, but not entirely, consistent in their use of the prologue. The format of the prologue between test files and foundation files is slightly different.

The general format of the prologue is as follows:

```
<file name>
   The distribution name of the file containing this prologue.

DISCLAIMER
   Use restrictions for ACATS tests; included in all tests.

OBJECTIVE
   A statement of the test objective; included in all tests.

TEST DESCRIPTION
   A short description of the design or strategy of the test or other pertinent information.
   Included in most newer tests but not generally included in legacy tests.

SPECIAL REQUIREMENTS
   optional – Included if the test has any special requirements for processing. Normally, this
   section will be found only in Specialized Needs Annex tests. For example, an Annex E test
```
may check for the correct implementation of partitions; the requirements for test partitioning and what to use as a main subprogram in each partition would be documented in this section.

**TEST FILES**
- *optional* — Included if the test depends on multiple files; identifies the component files of a multi-file test.

**APPLICABILITY CRITERIA**
- *optional* — Specifies the conditions under which the test can be ruled inapplicable.

**PASS/FAIL CRITERIA**
- *optional* — Explains how to interpret compilation, binding, and/or run-time results for grading the test.

**MACRO SUBSTITUTIONS**
- *optional* — Identifies the macro symbol(s) in the file that must be replaced and provides a brief description of what the replacement(s) represent. Appears only in legacy tests.

**CHANGE HISTORY**
- History of the test file. Included in all tests.

All tests have the line immediately after the disclaimer marked --*. Modern tests have the line after the last prologue line (before the first line of executable code) marked --!. No other comment lines are marked with those conventions, so the next line after the disclaimer and the first line of code may be found quickly with an editor search.

Some tests are composed of multiple files (other than foundation code). Rather than repeating the complete prologue in each file, an alternate approach has been used. One file (usually the one containing the main subprogram or the first file in the set) has the complete prologue; the other, related files have those sections that apply to files (TEST FILES, CHANGE HISTORY) and refer to the file with the complete prologue for the other sections.

### 4.5 General Standards

ACATS tests were developed to a general set of standards. To promote a variety of code styles and usage idioms in the tests, standards were not necessarily rigorously enforced but were used as guidelines for test writers. A maximum line length of 79 characters was used to enhance electronic distribution of tests (except when specific testing requirements dictated otherwise, usually in .dep and .tst files). Tests tend to be about 120 executable lines long, though many tests deviate from this norm (either longer or shorter) to achieve a design that focuses on the objective and a readable, maintainable test. Sometimes complex objectives have been divided into sub-objectives to achieve complete coverage in comprehensible, maintainable tests. Some tests check multiple objectives; in other cases, sub-objectives are checked in separate tests.

Legacy tests use only the basic 55-character set (26 capital letters, 10 digits, and 19 punctuation marks). Unless there is a specific test requirement, numeric values are in the range (-2048..2047), which can be represented in 12 bits. Numeric values are generally in the range (-128..127). Modern tests use both upper and lower case letters and may use larger numeric values (but within the range (-65536..65535) except in rare cases).

Legacy tests tend to use as few Ada features as necessary to write a self-checking executable test that can be read and maintained. Newer tests tend to exhibit a usage-oriented style, employing a rich assortment and interaction of features and exemplifying the kind of code styles and idioms that compilers may encounter in practice.
In the newer tests, Ada reserved words are entirely in lower case. Identifiers normally have their initial letter capitalized. Every attempt has been made to choose meaningful identifiers. In B class tests, identifier names often provide a clue to the specific case or situation under test. In C class tests, identifiers are normally chosen to help document the test design or the intent of the code.

The newer executable tests generally provide some visual separation of those test elements that focus on conformance issues from those that govern the flow of a test. For example, there is frequently a need to establish preconditions for a test and examine post-conditions after a section of test code has executed. To distinguish between constructs (types, objects, etc.) that are part of the test code and those that are artifacts of the testing process (e.g., pre-, post-conditions), the latter have TC_ prefixed to the identifier name. This prefix is shorthand for Test_Control.

4.6 Test Structure

Executable tests (class A, C, D, and E) generally use the following format:

```ada
with Report;
procedure Testname is
  <declarations>
begin
  Report.Test ("Testname", "Description ...");
  ...
  <test situation yielding result>
  if Post_Condition /= Correct_Value then
    Report.Failed ("Reason");
  end if;
  ...
  Report.Result;
end Testname;
```

The initial call to Report.Test prints the test objective using Text_IO output (unless the body of Report has been modified to do something else). After each section of test code, there is normally a check of post conditions. The if statement in this skeleton is such a check; unexpected results produce a call to Report.Failed. The sequence of test code / check of results may be repeated several times in a single test. Finally, there is a call to Report.Result that will print the test result to Text_IO output. Often, but not always, this structure is enclosed in a declare block.

One or more calls to Report.Failed will report a result of "FAILED" and a brief suggestion of the likely reason for that result.

More complex tests may include calls to Report.Failed in the code other than in the main program, and therefore exhibit the following format for the main procedure:

```ada
with Report;
procedure Testname is
  <declarations>
begin
  Report.Test ("Testname", "Description ...");
  ...
  Subtest_Call;
  ...
  Report.Result;
end Testname;
```

Fail conditions are detected in subprograms (or tasks) and Report.Failed is called within them.

Occasionally, as a test is running, it will determine that it is not applicable. In such a case, it will call Report.Not_Applicable that will report a result of "NOT_APPLICABLE" (unless there is also a call to Report.Failed).
Often, a test calls one of the functions Report.Ident_Int or Report.Ident_Bool to obtain a value that could be provided as a literal. These functions are intended to prevent optimizers from eliminating certain sections of test code. The ACATS suite has no intention of trying to discourage the application of optimizer technology, however satisfactory testing of language features often requires the presence and execution of specific lines of test code. Report.Ident_Int and Report.Ident_Bool are structured so that they can be modified when needed to defeat optimizer advances.

Class B tests may be structured differently. Since they are not executable, they normally do not include calls to Report.Test or Report.Result (since those lines of code would have no output effect). Instead, intentional errors are coded that invoke specific legality rules. The source code includes comments that document expected compiler results. Legal constructs may also be included in B class tests. Constructs that are allowed by the legality rules are marked -- OK; constructs that are disallowed are marked -- ERROR : . There is usually a brief indication of the nature of an intentional error on the same line or the line following a comment. The indications of expected results are approximately right justified to the code file margin, about column 79, for quick visual identification.

Class L tests are multifile tests with illegalities that should be detected at bind time. They are generally structured like class C tests, often with calls to Report.Test and Report.Result, but they are not expected to execute.

4.7 Delivery Directory Structure

The delivery of ACATS tests is structured into a directory tree that reflects the organization of the test suite and support code.

The top-level directory contains the support subdirectory, the docs subdirectory, and a subdirectory for each major grouping of tests. The support subdirectory contains all support packages (Report, ImpDef, TCTouch) and the source code for all test processing tools (Macro expander, Wide Character processor). Each of the other subdirectories contains all tests that begin with the indicated prefix. For example, all of the B2* tests are in the b2 subdirectory; all of the CXH* tests are in the cxh subdirectory. Note that all of the A* tests are in the a directory, all of the D* tests are included in the d subdirectory, and all of the E* tests are included in the e subdirectory. The l directory contains the L tests for the core; other L tests are in directories named with three letters, indicating the class (l) and the Specialized Needs Annex to which the tests apply.

Subdirectories that would be empty are not stubbed.

The following figure sketches this scheme, but does not show complete detail. A list of all subdirectories is included in Section 5.1.2.
4.8 File Format

To conserve space and ease downloading, all files in the delivered ACATS 3.0 (including test files, foundation files, and support files) have been compressed. Except as noted below, decompressed files (see Section 5.1.2) use only ASCII characters. A few tests use Unicode characters; these are indicated by an .au extension. Some of the documentation files are provided in PDF and/or HTML forms for greater readability. (The HTML documentation files include GIF and PNG graphics files.) Other than the documentation files, no formatting control characters, rulers or other information intended for editors or display programs are included in the files.

Files with the .zip extension have been compressed using a DOS zip utility; files with the .Z extension have been first put in Unix tar format and then compressed with Unix compress.
Section 5: Using the ACATS

There are eight major steps involved in using the ACATS test suite; two of them are sometimes not required. The steps are: installing the software, tailoring the software, processing the support files, establishing command scripts, processing the ACATS tests, grading the test results, addressing problems (if necessary), and reprocessing problem tests (if necessary). The first six of these tasks must be completed successfully to accomplish a test run. The first four normally need be completed only once for each ACATS release. Each step is explained in the following sections. The flow from one to the next is illustrated in the following figures.

1. Install Software

2. Tailor Software
   2.1 Modify Package ImpDef
   2.2 Modify test as needed
      process standard files
   2.3 Inspect reporting mechanism
      modify package Report if needed
   2.4 Define function declarations
      modify FCNDECL package specification
      create FCNDECL package body
      replace macro substitution with method calls

3. Process Support Files
   3.1 Compile:
      REFSPEC  SPRT13
      REBODYS  CHECKFILE
      IMPDEF  LENCHECK
      FCNDECL  TCOTCH
      ENUMCHECK
   3.2 Verify Reporting mechanism
      and the UT implementations
      process CZ tests
      verify tests

4. Establish Command Scripts
   Define compiler options
   Compile all driver tests
   Accept all order dependencies
   Compile 3rd party files
   Compile classes B tests
   Compile and bind classes
   List tests
   Compile, bind, execute
   Class A, C, D, E tests
   continue

Using the ACATS
5.1 Installation of the ACATS Test Suite

The ACATS test suite must be unloaded from the delivery medium or downloaded from a delivery site before it can be unpacked, customized for an implementation, run, and graded.

5.1.1 Contents of the ACATS Delivery

The delivery consists of 1 ZIP archive (set of compressed files) or 1 compressed tar file. Each ZIP archive or compressed tar file contains compressed versions of ACATS software (test, foundation, and/or support code) structured into a directory tree. Files must be extracted from the archives. The archive contents is described later in this section.

Usually, some test errors will be noted in the test suite. If possible, the ACAA will correct the errors and issue a corrected test. If a correction is not possible, the test will be withdrawn. Withdrawn tests are not used in conformity assessments. For a period after the issuance of a corrected test, either the original or the corrected test can be used for conformity assessment. See the ACAA's procedures [Pro01] for details.

The ACAA also will issue new tests periodically. As with modified tests, new tests must be available for a period of time before they are required in conformity assessments.

These changes to the issued ACATS are documented in the ACATS Modification List (AML). This list includes a list of all new tests, all modified tests, and all withdrawn tests, and an indication as to when each will be (or is) required for conformity assessments. Each version of the modification list is given a suffix letter. A ZIP archive and tar file containing the new and/or modified tests is available. The files are named MOD_3_0x, where 'x' represents the suffix letter for the AML version.
These files can be found on the ACAA’s web site:
www.ada-auth.org/acats.html.

The AML is also distributed by e-mail. To receive these lists, join the ACAA mailing list. To do so, simply send a message to
listserv@ada-auth.org.

with a body of
Join Acaa

5.1.2 Guide to Decompressing Files
The ACATS files are provided in two forms: compressed in zip format and compressed in Unix compress format. Zipped files are included in a zip archive (files) with the file extension .zip. A Unix compressed files, with extension .Z, contains a Unix tar file. This section provides generic instructions for uncompressing them. These instructions are not the only ways to uncompress the files; sophisticated users may wish to use their own procedures.

If the instructions below are used, the following subdirectories will have been created and populated with test files after all decompression:

```
./acats3_0/a   ./acats3_0/be   ./acats3_0/cd   ./acats3_0/cxc
./acats3_0/b2  ./acats3_0/bxa  ./acats3_0/ce   ./acats3_0/cxd
./acats3_0/b3  ./acats3_0/c2   ./acats3_0/cxa  ./acats3_0/cxe
./acats3_0/b4  ./acats3_0/c3   ./acats3_0/cxb  ./acats3_0/cxf
./acats3_0/b5  ./acats3_0/c4   ./acats3_0/cz   ./acats3_0/cxg
./acats3_0/b6  ./acats3_0/c5   ./acats3_0/d    ./acats3_0/cxh
./acats3_0/b7  ./acats3_0/c6   ./acats3_0/e    ./acats3_0/lxd
./acats3_0/b8  ./acats3_0/c7   ./acats3_0/l    ./acats3_0/lxe
./acats3_0/b9  ./acats3_0/c8   ./acats3_0/lxc  ./acats3_0/lxh
./acats3_0/ba  ./acats3_0/c9   ./acats3_0/bxd  ./acats3_0/docs
./acats3_0/bb  ./acats3_0/ca   ./acats3_0/bx    ./acats3_0/support
./acats3_0/bc  ./acats3_0/cb   ./acats3_0/bxf
./acats3_0/bd  ./acats3_0/cc   ./acats3_0/bxh
```

Note that the names are given here in all lowercase; some systems may create uppercase names. The path separator, shown here as "/", may also differ.

5.1.2.1 Decompressing Zipped Files
All ACATS files have been compressed (zipped) into compressed archives (zip-files) that have the MS-DOS file extension ".zip". A Windows command-line utility was used to compress them. They must be decompressed before they can be further processed. A decompression utility is available from the source of the ACATS distribution. All ACATS 3.0 files may be decompressed using the following steps. Approximately 32 MB of free space on a Windows machine hard drive will be required to accomplish the decompression using this technique.

Create a directory on the hard disk to contain ACATS. In these examples, we assume the name is acats3_0, but any name can be used. Copy the archive (file with .zip extension) to the hard disk in the new directory. Decompress it insuring that directories are used. For the unzip program, this is the default setting. For the pckunzip program, this is the -d option. For the winzip program, ensure that "Use Directory Names" is checked. Also, ensure that the files are decompressed into the proper directory.
For command line decompressors, this means ensuring that the current subdirectory is acats3_0. For winzip, this simply means selecting acats3_0 as the extract path.

For example, using unzip, and assuming that the archive name is ACATS30.zip, type

```
cd acats3_0
```
to set the proper directory, and

```
unzip ACATS30
```
to extract the files.

The files were compressed on a Windows system, where <CR><LF> is used as a line terminator. Decompressors for other systems using other line terminators should be able convert the line terminators. The ACAAs has a short Ada program which converts a file from Windows to Unix format; please send the ACAAs mail at agent@ada-auth.org to request it if needed.

After all files have been extracted from the archive, delete the archive file from the hard disk if you wish to conserve space.

As it decompresses files, unzip will restore the directory structure of the files, creating all needed subdirectories.

Some users may prefer to work with ACATS files in an alternate directory structure or none at all. If the unzip utility is invoked with the ".j" option, all files in the archive will be decompressed and placed in the local working directory. In other words, none of the above subdirectories will be created. Since there are too many ACATS files to fit into a root DOS directory, if you wish to put all files in a single directory, you must first create a subdirectory (e.g., mkdir \ACATS) and unzip all archives there.

### 5.1.2.2 Decompressing Unix Compress Files

All ACATS files have been included in 1 Unix tar format file and then compressed using the Unix compress utility. To create a set of ACATS files, first copy the compressed files acats_30.tar.Z from the distribution source to a hard drive. Uncompress the file with the Unix command

```
uncompress acats_30.tar.Z
```
(Note that particular Unix implementations may have different formats or require specific qualifiers.)

After the ACATS file has been uncompressed, it must be untarred. Move to the directory where you want the acats3_0 directory to be created and then untar the ACATS files

```
tar -xvf <path>/acats_30.tar
```
where <path> is the location of the uncompressed tar file.

Please note that these are generic instructions and may need to be customized or modified for specific systems.

### 5.1.3 Files With Non-Graphic Characters

Four ACATS test files contain ASCII non-graphic (control) characters that may be lost or corrupted in the file transfer and decompression process. The user must ensure that the proper characters are restored as necessary.

There are also a number of ACATS test files provided in UTF-8 format. These files have an .au file extension. The user must take care that any processing tools do not corrupt UTF-8 files. (This would be unusual: it is most likely if a tool does not recognize UTF-8 formatted files and also modifies Latin-1 characters with codes larger than 127.)
The following paragraphs describe the four tests with ASCII non-graphic characters.

5.1.3.1 A22006C

This test checks that format effectors can appear at the beginning of a compilation. At the beginning of the file, the first line is empty (indicated by the system's end-of-line marker, which may be a sequence of one or more characters or may be indicated by some other means). The second line contains 20 characters: 6 control characters followed by the comment delimiter, a space, and the file name (A22006C.ADA). The control characters are:

<table>
<thead>
<tr>
<th>Common Name</th>
<th>Ada Name</th>
<th>Decimal Value</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carriage return</td>
<td>ASCII.CR</td>
<td>13</td>
<td>0D</td>
</tr>
<tr>
<td>Carriage return</td>
<td>ASCII.CR</td>
<td>13</td>
<td>0D</td>
</tr>
<tr>
<td>Vertical tab</td>
<td>ASCII.VT</td>
<td>11</td>
<td>0B</td>
</tr>
<tr>
<td>Line feed</td>
<td>ASCII.LF</td>
<td>10</td>
<td>0A</td>
</tr>
<tr>
<td>Line feed</td>
<td>ASCII.LF</td>
<td>10</td>
<td>0A</td>
</tr>
<tr>
<td>Form feed</td>
<td>ASCII.FF</td>
<td>12</td>
<td>0C</td>
</tr>
</tbody>
</table>

5.1.3.2 B25002A

This test checks that ASCII control characters (other than format effectors) are not permitted in character literals. The expected characters are documented in source code comments, using the customary 2- or 3-letter mnemonics. The 28 characters are used in their ASCII order, and have ASCII values 0 through 8, 14 through 31, and 127.

5.1.3.3 B25002B

This test checks that the five ASCII format effector characters cannot be used in character literals. There are two groups of code containing the illegal characters; in each group, the characters appear in the order given below:

<table>
<thead>
<tr>
<th>Common Name</th>
<th>Ada Name</th>
<th>Decimal Value</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal tab</td>
<td>ASCII.HT</td>
<td>9</td>
<td>09</td>
</tr>
<tr>
<td>Vertical tab</td>
<td>ASCII.VT</td>
<td>11</td>
<td>0B</td>
</tr>
<tr>
<td>Carriage return</td>
<td>ASCII.CR</td>
<td>13</td>
<td>0D</td>
</tr>
<tr>
<td>Line feed</td>
<td>ASCII.LF</td>
<td>10</td>
<td>0A</td>
</tr>
<tr>
<td>Form feed</td>
<td>ASCII.FF</td>
<td>12</td>
<td>0C</td>
</tr>
</tbody>
</table>

5.1.3.4 B26005A

This test checks the illegality of using control characters in string literals. Each string literal (ASCII codes 0 through 31 and 127) is used once, and the uses appear in ASCII order. Each use is also documented in a source code comment, which identifies the character by its common 2- or 3-character mnemonic.
5.2 Tailoring the ACATS Test Suite

There are some files in the delivery that require modification before ACATS 3.0 is ready for processing by an Ada implementation. Package ImpDef (impdef.a) must be edited to include values suitable for proper testing of an implementation if the defaults are not acceptable. The macros.dfs file must similarly be edited to include values suitable for testing. All .tst files (including package Spprt13 (spprt13s.tst)) must have their macro symbols replaced by implementation specific values. A body for FcnDecl (fcndcl.ada) must be provided if necessary. Finally, Package Report (repbody.ada) must be modified if necessary.

The required customization is described in the following sections.

Customizations of these files from previous versions of the ACATS suite generally can be used with ACATS 3.0, but users should ensure that neither their requirements nor the underlying files have changed since the customizations were made.

5.2.1 ImpDef Customization

There was no change to Impdef or any of its children from ACATS 2.6 to ACATS 3.0. A version of any of these packages that was tailored for ACATS 2.6 should be valid for ACATS 3.0 unless some implementation characteristics have changed.

ACATS tests use the entities in ImpDef to control test execution. Much of the information in ImpDef relates to the timing of running code; for example, the minimum time required to allow a task switch may be used by a test as a parameter to a delay statement. The time to use is obtained as an ImpDef constant.

impdef.a was added as a new feature to ACATS 2.0 suite. It is related to macro.dfs in that it must be customized with values specific to an implementation and ACATS tests will rely on these values. ImpDef is different in the following respects:

- Defaults are provided. Some implementations may be able to rely entirely on the default values and subprograms, so no customization would be necessary.
- Some implementations may choose to provide bodies for procedures and/or functions. Bodies so provided must satisfy requirements stated in ImpDef.
- Tests depending on Impdef do not need customization (macro substitution). Instead, ImpDef must be available at compile time (i.e., included in the environment) for tests that rely upon it. This simplifies the customization process and management and also is similar to the way that Ada projects typically manage configuration parameters.

There are child packages of ImpDef for each of the Specialized Needs Annexes. An implementation that uses one or more of the Specialized Needs Annexes in its conformity assessment must customize the associated ImpDef child packages (or rely on their defaults) and must set the appropriate Booleans in impdef.a. It is not necessary to customize Impdef children for Specialized Needs Annexes that are not included in a particular conformity assessment.

Specific instructions for the values required by ImpDef and its children are included in impdef.a, impdefc.a, impdefd.a, impdefe.a, impdefg.a, and impdefh.a. (Note that impdefc, for example, refers to Annex C.) An excerpt from ImpDef is included in Annex B, “Parameterization Files”.

All implementations must customize impdef.a unless they wish to rely on the defaults provided. ImpDef must be part of the environment whenever a test that depends on it is processed. Similarly, the child of Impdef corresponding to each Specialized Needs Annex that the implementer intends to test during a conformity assessment must be customized and be part of the environment when the Annex tests are processed.
5.2.2 Macro Defs Customization

There was no change to the macro.dfs file from ACATS 2.6 to ACATS 3.0. A version of macro.dfs that was tailored for ACATS 2.6 should be valid for ACATS 3.0 unless some implementation characteristics have changed.

Tests in files with the extension .tst contain symbols that represent implementation dependent values. The symbols are identifiers with a initial dollar sign ('$'). Each symbol must be replaced with an appropriate textual value to make the tests compilable. This process is sometime known as macro substitution.

The Macrosub program distributed with the ACATS can automatically perform the required substitutions. This program reads the replacement values for the symbols from the file macro.dfs and edits all the .tst tests in the suite to make the needed changes. It writes the resulting, compilable programs into files with the same name as the original but with the extension .adt. A sample macro.dfs is included with the ACATS; it contains descriptions of all the symbols used in the test suite.

Substitutions using the Macrosub program may be made as follows:

1. Edit the file macro.dfs using values appropriate for the implementation. Symbols that use the value of MAX_IN_LEN are calculated automatically and need not be entered.
2. Create a file called tsttests.dat that includes all of the .tst test file names, and their directory locations if necessary. A version of this file (without directory information) is supplied.
3. Compile and bind MacroSub.
4. Run MacroSub.

The program will replace all symbols in the .tst files with values from macro.dfs. Test files with the original test name but the extension .adt will contain the processable tests. The original .tst files will not be modified.

5.2.3 Packages SPPRT13 and FCNDECL

Package SPPRT13 declares six constants of type System.Address that are primarily used by tests of Section 13 features. It is in the file spprt13s.tst. As distributed, the package uses macro symbols that must be replaced. In most cases, the substitution can be accomplished by the macro substitution described in the preceding section. If appropriate literals, constants, or predefined function calls can be used to initialize these constants, they should be supplied in macro.dfs. Otherwise, the package FCNDECL must be modified.

All implementations should verify that package SPPRT13 can be properly customized using the macro substitution technique. Note that a body for SPPRT13 is illegal, nor is it allowed to add declarations to package SPPRT13.

The specification for package FCNDECL is in the file fcndecl.ada. SPPRT13 depends on FCNDECL (in a context clause that both withs it and uses it). As supplied with the ACATS, FCNDECL is an empty package specification. If appropriate literals, constants, or predefined function calls cannot be used to customize the constants declared in SPPRT13, the implementer must declare appropriate functions in the specification of FCNDECL and provide bodies for them in a package body or with a pragma Import.

Modifications to FCNDECL must receive advance approval from the ACAL (and, if necessary, the ACAA) before use in a conformity assessment.
5.2.4 Modification of Package REPORT

All executable tests use the Report support package. It contains routines to automate test result reporting as well as routines designed to prevent optimizers from removing key sections of test code. The specification of package Report is in the file repspec.ada; the body is in repbody.ada.

Under some conditions, the body of package Report may need to be modified. For example, the target system for a cross-compiler may require a simpler I/O package than the standard package Text_IO. In such a case, it may be necessary to replace the context clause and the I/O procedure names in the body of Report.

Modifications to Report must receive advance approval from the ACAL (and, if necessary, the ACAA) before use in a conformity assessment.

5.2.5 Allowed Test Modifications

Class B tests have one or more errors that implementations must identify. These tests are structured such that, normally, implementations can report all included errors. Occasionally, an implementation will fail to find all errors in a B-test because it encounters a limit (e.g., error cascading, resulting in too many error reports) or is unable to recover from an error. In such cases, a user may split a single B-test into two or more tests. The resulting tests must contain all of the errors included in the original test, and they must adhere as closely as possible to the style and content of the original test. Very often, the only modification needed is to comment out earlier errors so that later errors can be identified. In some cases, code insertion will be required. An implementation must be able to demonstrate that it can detect and report all intended B-test errors.

Splits may also be required in executable tests, if, for example, an implementation capacity limitation is encountered (e.g., a number of generic instantiations too large for the implementation). In very exceptional cases, tests may be modified by the addition of an attribute definition clause (to alter the default size of a collection), or by the addition of an elaboration Pragma (to force an elaboration order).

Tests that use configuration pragmas (see 5.5.5.5) may require modification since the method of processing configuration pragmas is implementation dependent.

Some tests include foreign language code (Fortran, C, or COBOL). While the features used should be acceptable to all Fortran, C, and COBOL implementations, respectively, some implementations may require modification to the non-Ada code. Modifications must, of course, preserve the input-output semantics of the (foreign language) subprogram; otherwise, the ACATS test will report a failure.

All splits and modifications must be approved in advance by the ACAL (and, if necessary, the ACAA) before they are used in a conformity assessment. It is the responsibility of the user to propose a B-test split that satisfies the intention of the original test. Modified tests should be named by appending an alphanumeric character to the name of the original test. When possible, line numbers of the original test should be preserved in the modification.

All tests must be submitted to the compiler as distributed (and customized, if required). If a test is executable (class A, C, D, E) and compiles successfully, then it must be run. Modified tests or split tests may be processed next. Only the results of the modified tests will be graded.

If the ACAA has issued an ACATS Modification List (see Section 5.1.1), then the modified versions of tests with modifications must be used. Either the original version or the modified version of a test with an allowed modification may be used.
5.3 Processing the Support Files

After all the files identified in Section 5.2 have been customized as needed and required, the support files can be processed and the reporting mechanism can be verified.

5.3.1 Support Files

The following files are necessary to many of the ACATS tests. Implementations that maintain program libraries may wish to compile them into the program library used for conformity assessment:

- repspec.ada
- repbody.ada
- impdef.a
- impdefc.a (if testing Annex C)
- fnndecl.ada
- impdefd.a (if testing Annex D)
- checkfil.ada
- impdefe.a (if testing Annex E)
- lencheck.ada
- impdefg.a (if testing Annex G)
- enumchek.ada
- impdefh.a (if testing Annex H)
- tctouch.ada
- spprt13s.adt (after macro substitution)

Depending on local requirements and strategy, it may also be convenient to compile all foundation code into the program library as well.

5.3.2 "CZ" Acceptance Tests

Four tests having names beginning "CZ" are part of the ACATS suite. Unlike other tests in the suite, they do not focus on Ada language features. Instead, they are intended primarily to verify that software needed for the correct execution of the test suite works as expected and required. They check, for example, to see that package Report and package TCTouch work correctly.

All CZ tests must execute correctly and exhibit the prescribed behavior for a successful conformity assessment. CZ tests must be processed and run as the first step of a conformity assessment to ensure correct operation of the support software.

The acceptance test CZ1101A tests the correct operation of the reporting facilities of package Report, including checks that Not_Applicable and Failed calls are reported properly, and that premature calls cause failure. Therefore, CZ1101A will print some failure messages when it is executed. The presence of these messages does not necessarily mean the test has failed. A listing of the expected output for CZ1101A is included in Annex C, “Results of CZ Tests” (times and dates in the actual output will differ).

The acceptance test CZ1102A tests the correct operation of the dynamic value routines in Report. This test should report "PASSED"; any other result constitutes a test failure.

The acceptance test CZ1103A ensures the correct operation of procedure Checkfile. (Some of the executable file I/O tests use a file checking procedure named Checkfile that determines an implementation's text file characteristics. The source code for this procedure is in the file checkfil.ada.) CZ1103A checks whether errors in text files are properly detected, therefore, CZ1103A will print some failure messages when it is executed. The presence of these messages does not necessarily mean the test has failed. A listing of the expected output for CZ1103A is included in Annex C, “Results of CZ Tests” (times and dates in the actual output will differ).
The acceptance test CZ00004 produces output that verifies the intent of the conformity assessment. It relies on ImpDef having been correctly updated for the conformity assessment and produces output identifying the annexes (if any) that will be included as part of the conformity assessment. This test also checks for the proper operation of the TCTouch package, includes checks that assertion failures are reported properly, therefore CZ00004 will print some failure messages when it is executed. The presence of these messages does not necessarily mean the test has failed. A listing of the expected output for CZ00004 is included in Annex C, “Results of CZ Tests”; since this output includes values from the customized impdef, non-failure lines may vary from those in the expected output. However, the number of lines and their relative positions should not change (only the contents of the lines can change).

5.4 Establishing Command Scripts

Users will often find it convenient to run large numbers of ACATS tests with command scripts. This section discusses some of the issues to be considered in developing a script.

5.4.1 Command Scripts

All compiler options and switches that are appropriate and necessary to run the ACATS tests must be identified and included in commands that invoke the compiler. The same is true for the binder or any other post-compilation tools. Any implementation dependent processing of partitions, configuration pragmas, and strict mode processing must be part of the scripts for running tests that rely on these features.

A script should compile (only) all class B tests. It should compile and bind all class L tests; if link errors are not explicitly given, the script should attempt to execute the L tests. It should compile all class F files. It should compile, bind, and execute all class A, C, D, and E tests.

Sample commands for processing the ACATS are a required part of a formal Ada Conformity Assessment Test Report. If a test report is available for the implementation being tested, these commands can be used as a guideline for developing command scripts.

5.4.2 Dependencies

A command script must take account of all required dependencies. As noted earlier, some tests are composed of multiple test files. Also, some tests include foundation code, which may be used by other tests. If a foundation is not already in the environment, it must be compiled as part of building the test. All files that are used in a test must be compiled in the proper order, as indicated by the file name. For implementations that require the extraction individual compilation units from test files before submission to the compiler, the individual units must be submitted to the compiler in the same order in which they appear in the file.

5.5 Processing ACATS Tests

After the ACATS tests and support code has been installed and all required modifications and preliminary processing have been completed, the suite can be processed by an implementation. This section describes the tests required for conformity assessment, required partitioning, how tests may be bundled for efficiency, and certain processing that may be streamlined. It also describes how the suite has been organized to allow a user to focus on specific development needs.
5.5.1 Required Tests

An implementation may be tested against the core language only or the core language plus one or more Specialized Needs Annexes. All core tests (except as noted in 5.5.4) must be processed with acceptable results for conformity assessment of the core language. All legacy tests, as well as all newer tests for clauses 2-13 and annexes A and B are core tests. Conformity assessment including one or more Specialized Needs Annexes requires that all tests for the annex(es) in question be correctly processed in addition to all core tests.

Tests that are not applicable to an implementation (e.g., because of size limitations) and tests that report "NOT APPLICABLE" when run by an implementation must nevertheless be processed and demonstrate appropriate results.

Tests that are withdrawn on the current ACATS Modification List as maintained by the ACAA need not be processed.

5.5.2 Test Partitions

Unless otherwise directed by the Special Requirements section of a test, all tests are to be configured and run in a single partition. The method of specifying such a partition is implementation dependent and not determined by the ACATS. The only tests that must be run in multiple partitions are those that test Annex E, Distributed Systems.

5.5.3 Bundling Test Programs

In some situations, the usual test processing sequence may require an unacceptable amount of time. For example, running tests on an embedded target may impose significant overhead time to download individual tests. In these cases, executable tests may be bundled into aggregates of multiple tests. A set of bundled tests will have a driver that calls each test in turn; ACATS tests will then be called procedures rather than main procedures. No source changes in the tests are allowed when bundling; that is, the only allowed change is the method of calling the test.

All bundles must be approved by the ACAL (and, if necessary, the ACAA) to qualify for a conformity assessment. It is the responsibility of the user to identify the tests to be bundled and to write a driver for them.

5.5.4 Processing that may be Omitted

A user may streamline processing of the ACATS tests to the greatest degree possible consistent with complete processing of all tests.

Many modern tests rely on foundation code. A foundation need not be compiled anew each time a different test uses it. In a processing model based on a program library, it is reasonable to compile the code into the library only once and allow the binder to use the processed results for each test that with the foundation.

A user may determine, with ACAL concurrence, that some tests require support that is impossible for the implementation under test to provide. For example, there are tests that assume the availability of file I/O whereas some (embedded target) implementations do not support file I/O. Those tests need not be processed during witness testing; however, the implementer must demonstrate that they are handled in accordance with the language standard. This demonstration may be performed before witness testing, in which case it need not be repeated.
Annex B tests that require foreign language code (Fortran, C, COBOL) to be compiled and bound with Ada code need not be processed if an implementation does not support a foreign language interface to the respective language.

Tests for the Specialized Needs Annexes of Ada need not be processed except by implementations that wish to have Annex results documented. In that case, only the tests for the annex(es) in question (in addition to all core tests) need be processed. If any tests for a particular Annex are processed, then all tests for that Annex must be processed. If an implementation does not support a feature in a Specialized Needs Annex test, then it must indicate the non-support by rejecting the test at compile time or by raising an appropriate exception at run time. (See Ada 1.1.3(17).)

No withdrawn test need be processed. Tests classified as Pending New in the current ACATS Modification List also do not need to be processed. Pending New tests are new tests included with the ACATS for review purposes, and are not yet required for conformity assessment. (Tests classified as New in the current ACATS Modification List do need to be processed; these are required for conformity assessments.)

5.5.5 Tests with Special Processing Requirements

Some tests may require special handling. These are primarily SNA tests, but some core tests are affected. For example, distributed processing tests may require an executable image in multiple partitions, where partitions are constructed in an implementation specific manner. Real-time processing tests may have configuration pragmas that have to be handled in an implementation specific way. Numeric processing tests require strict mode processing to be selected. Each such test has a Special Requirements section in the test header describing any implementation specific handling that is required for the test.

A list of all such tests is provided in A.6, “Tests With Special Requirements”.

5.5.5.1 Tests Involving Limited Views

[Amend1] added the concept of limited views to Ada. For most ACATS tests, the possibility of limited views can be ignored, as either they are not used at all, or they need not be separated from the full view in the environment. (It is presumed that adding the full view to the environment also adds the corresponding limited view.)

However, a few tests require that the limited view of a unit be added to the environment separately from the full view of the unit. Such tests have dependencies in the full views on units that have not yet been compiled (added to the environment). For these tests, any extra steps needed to add the limited view to the environment separately from the full view will need to be accomplished.

The tests identified below need to add the limited view of one or more units to the environment separately from the full view of the units.

ca11023 ca12001

5.5.5.2 Foreign Language Interface Tests

Annex B, Interface to Other Languages, is part of the Ada core language. Any implementation that provides one or more of the packages Interfaces.C, Interfaces.COBOL, or Interfaces.Fortran must correctly process, and pass, the tests for interfaces to C, COBOL, and/or Fortran code respectively, with the possible exception of tests containing actual foreign code.

5.5.4 Processing that may be Omitted
An implementation that provides one or more of these Interfaces child packages must successfully compile the 
Ada units of tests with actual foreign language code. If the implementation does not support the actual binding of the 
foreign language code to Ada, these tests may report binding errors, or may reject the pragma Import, in which case they may be 
graded as inapplicable. If the implementation supports the binding and an appropriate compiler is available, the tests must 
execute and report "Passed". If the implementation supports the binding, but it is not feasible to have an appropriate compiler 
available, then the tests may be graded as inapplicable by demonstrating that they fail to bind.

If one of the Interfaces child packages is not provided, then the corresponding tests may be graded as 
inapplicable, provided they reject the corresponding with clause.

The tests involving interfaces to foreign code are listed below.

The foreign language code included in ACATS tests uses no special or unique features, and should be 
accepted by any standard (C, COBOL, or Fortran) compiler. However, there may be dialect problems that 
prevent the code from compiling correctly. Modifications to the foreign language code are allowable; the 
modifications must follow the code as supplied as closely as possible and the result must satisfy the 
requirements stated in the file header. Such modifications must be approved in advance by the ACAL 
(and, if necessary, the ACAA).

The method for compiling foreign code is implementation dependent and not specified as part of the 
ACATS. Ada code in these tests must be compiled as usual. The Ada code includes Pragma Import that 
references the foreign language code. The link name of foreign language object code must be provided in 
ImpDef. When all code has been compiled, the test must be bound (including the foreign language object 
code) and run. The method for binding Ada and foreign language code is implementation dependent and 
not specified as part of the ACATS. The test must report "PASSED" when executed.

C Language Interface

If the implementation provides the package Interfaces.C, the tests identified below must be satisfactorily 
processed as described above.

The starred tests contain C code that must be compiled and linked if possible, as described above. The C 
code is easily identifiable because the file has the extension .C. The C code may be modified to satisfy 
dialect requirements of the C compiler. The C code files must be compiled through a C compiler, and the 
resulting object code must be bound with the compiled Ada code. Pragma Import will take the name of 
the C code from ImpDef.

<table>
<thead>
<tr>
<th>cd30005*</th>
<th>cxb3005</th>
<th>cxb3010</th>
<th>cxb3015</th>
</tr>
</thead>
<tbody>
<tr>
<td>cxb3001</td>
<td>cxb3006*</td>
<td>cxb3011</td>
<td>cxb3016</td>
</tr>
<tr>
<td>cxb3002</td>
<td>cxb3007</td>
<td>cxb3012</td>
<td></td>
</tr>
<tr>
<td>cxb3003</td>
<td>cxb3008</td>
<td>cxb3013*</td>
<td></td>
</tr>
<tr>
<td>cxb3004*</td>
<td>cxb3009</td>
<td>cxb3014</td>
<td></td>
</tr>
</tbody>
</table>

COBOL Language Interface

If the implementation provides the package Interfaces.COBOL, the tests identified below must be processed satisfactorily, as described above.

The starred test contains COBOL code that must be compiled and linked if possible, as described above. 
The COBOL code is easily identifiable because the file has the extension .CBL. The COBOL code may 
be modified to satisfy dialect requirements of the COBOL compiler. The COBOL code files must be 
compiled through a COBOL compiler, and the resulting object code must be bound with the compiled 
Ada code. Pragma Import will take the name of the COBOL code from ImpDef.
5.5.5.2 Foreign Language Interface Tests

If the implementation has a Fortran language interface, the tests identified below must be processed satisfactorily, as described above.

The starred tests contain Fortran code that must be compiled and linked if possible, as described above. The Fortran code is easily identifiable because the file has the extension .FTN. The Fortran code may be modified to satisfy dialect requirements of the Fortran compiler. The Fortran code files must be compiled through a Fortran compiler, and the resulting object code must be bound with the compiled Ada code. Pragma Import will take the name of the Fortran code from ImpDef.

5.5.5.3 Tests for the Distributed Systems Annex

The ACATS tests for the Distributed Systems Annex are applicable only to implementations that wish to test this SNA. Not all of these tests apply to all implementations, since the annex includes some implementation permissions that affect the applicability of some tests.

The principal factors affecting test applicability are:
1. whether the Remote_Call_Interface pragma is supported;
2. whether a Partition Communication System (PCS) is provided (i.e., whether a body for System.RPC is provided by the implementation);
3. whether the implementation has taken advantage of the permission to change the specification of System.RPC;
4. whether the Real-Time Annex is also supported.

An implementation may test for the annex without providing a PCS. In order to test for the Distributed Systems Annex, an implementation must allow a body for System.RPC to be compiled.

Remote_Call_Interface pragma

Ada allows explicit message-based communication between active partitions as an alternative to RPC [see Ada E.2.3(20)]. If an implementation does not support the Remote_Call_Interface pragma then the following tests are not applicable:

Partition Communication System

An implementation is not required to provide a PCS [see Ada E.5(27)] in order to test the Distributed Systems Annex. If no PCS is provided then the following tests are not applicable:
System.RPC

Two tests provide a body for System.RPC, and a third test checks the specification of System.RPC. An alternative declaration is allowed for package System.RPC [see [Amend1] E.5(27.1/2)]. If an alternative declaration is used for System.RPC, the following tests are not applicable:

cxe5001  cxe5002  cxe5003

Real-Time Annex Support

Many implementations that support the Distributed Systems Annex will also support the Real-Time Annex. Test cxe4003 is designed to take advantage of Real-Time Annex features in order to better test the Distributed Systems Annex.

For implementations that do not support the Real-Time Annex, test cxe4003 must be modified. This modification consists of deleting all lines that end with the comment --RT.

Configuring Multi-Partition Tests

Some Distributed Systems Annex tests require multiple partitions to run the test, but no more than two partitions are required for running any of them. All multi-partition tests contain a main procedure for each of the two partitions. The two partitions are referred to as "A" and "B" and the main procedures for these partitions are named <test_name>_A and <test_name>_B respectively. Each test contains instructions naming the compilation units to be included in each partition. Most implementations will be primarily concerned with the main procedure and RCI packages that are to be assigned to each partition; the remainder of the partition contents will be determined by the normal dependency rules. The naming convention used in multi-partition tests aid in making the partition assignments. If the name of a compilation unit ends in ",_A<optional_digit>" then it should be assigned to partition A. Compilation units with names ending in ",_B<optional_digit>" should be assigned to partition B.

The following tests require that two partitions be available to run the test:

cxe1001  cxe4002  cxe4006  lxe3002*
cxe2001*  cxe4003  cxe5002
cxe2002  cxe4004  cxe5003
cxe4001  cxe4005  lxe3001

(*) Tests cxe2001 and lxe3002 contain a Shared_Passive package and two active partitions. They may be configured with either two or three partitions. The two-partition configuration must have two active partitions and the Shared_Passive package may be assigned to either one of the active partitions. The three-partition configuration consists of two active partitions and a single passive partition, and the passive partition will contain the single Shared_Passive package.

Running Multi-Partition Tests

All of the multi-partition tests include the package Report in both of the active partitions. In order for the test to pass, both partitions must produce a passed message (except for lxe3002 - see special instructions for that test). If either partition produces a failed message, or if one or both partitions do not produce a passed message, the test is graded "failed".

When running the multi-partition tests it is not important which partition is started first. Generally, partition A acts as a server and partition B is a client, so starting partition A first is usually best.

In the event a test fails due to the exception Communication_Error being raised, it is permissible to rerun the test.
5.5.5.4 Tests for the Numerics Annex

Many of the tests for Annex G, Numerics, must be run in strict mode. The method for selecting strict mode is implementation dependent and not specified by the ACATS. (Note that the tests for numerical functions specified in Annex A may, but need not, be run in strict mode.) The following tests must be run in strict mode:

- cxg2003
- cxg2004
- cxg2006
- cxg2007
- cxg2008
- cxg2009
- cxg2010
- cxg2011
- cxg2012
- cxg2013
- cxg2014
- cxg2015
- cxg2016
- cxg2017
- cxg2018
- cxg2019
- cxg2020
- cxg2021

5.5.5.5 Tests that use Configuration Pragmas

Several of the tests in Annex D, Real Time Systems, Annex E, Distributed Systems, and Annex H, High Integrity Systems, use configuration pragmas. The technique for applying a configuration pragma to a test composed of multiple compilation units is implementation dependent and not specified by the ACATS. Every implementation that uses any such test in a conformity assessment must therefore take the appropriate steps, which may include modifications to the test code and/or post-compilation processing, to ensure that such a pragma is correctly applied. The following tests require special processing of the configuration pragma:

- ba15001
- bxc5001
- bxh4001
- bxh4002
- bxh4003
- bxh4004
- bxh4005
- bxh4006
- bxh4007
- bxh4008
- bxh4009
- bxh4010
- bxh4011
- bxh4012
- bxh4013
- cxd1004
- cxd2001
- cxd4001
- cxd4003
- cxd4004
- cxd4005
- cxd4006
- cxd4007
- cxd4008
- cxd5001
- cxd6001
- cxd6002
- cxd6003
- cxd6004
- cxd6005
- cxd6006
- cxd6007
- cxh1001
- cxh1002
- cxh1003
- cxh1004
- cxh1005
- cxh1006
- cxh1007
- cxh1008
- cxh1009
- cxh1010
- cxh1011
- cxh1012
- cxh1013
- lxd7001
- lxd7002
- lxd7003
- lxd7004
- lxd7005
- lxd7006
- lxd7007
- lxd7008
- lxd7009
- lxd7010
- lxd7011
- lxd7012
- lxd7013

5.5.6 Focus on Specific Areas

The ACATS test suite is structured to allow compiler developers and testers to use parts of the suite to focus on specific compiler feature areas.

Both the legacy tests and the newer tests tend to focus on specific language features in individual tests. The name of the test is generally a good indicator of the primary feature content of the test, as explained in the discussion of naming conventions. Beware that legacy test names have not changed, but the Ada Reference Manual organization has changed from [Ada83] to [Ada95], so some legacy test names point to
the wrong clause of the Ada Standard. Further, note that the general style and approach of the newer tests creates user-oriented test situations by including a variety of features and interactions. Only the primary test focus can be indicated in the test name.

ACATS 3.0 tests are divided into core tests and Specialized Needs Annex tests. Recall that annexes A and B are part of the core language. All annex tests (including those for annexes A and B) have an 'X' as the second character of their name; Specialized Needs Annex tests have a letter between 'C' and 'H' (inclusive) corresponding to the annex designation, as the third character of the test name.

5.6 Grading Test Results

Although a single test may examine multiple language issues, ACATS test results are graded "passed", "failed", or "not applicable" as a whole.

All customized, applicable tests must be processed by an implementation. Results must be evaluated against the expected results for each class of test. Results that do not conform to expectations constitute failures. The only exceptions allowed are discussed in 5.2.5: in such cases, processing the approved modified test(s) must produce the expected behavior. Any differences from the general discussion of expected results below for executable or non-executable tests are included as explicit test conditions in test prologues.

Warning or other informational messages do not affect the pass/fail status of tests.

Expected results for executable and non-executable tests are discussed in Sections 5.6.1, 5.6.2, and 5.6.3. Tests that are non-applicable for an implementation are discussed in 5.6.4. Withdrawn tests are discussed in 5.6.5.

5.6.1 Expected Results for Executable Tests

Executable tests (classes A, C, D, E) must be processed by the compiler and any post-compilation steps (e.g., binder, partitioner) without any errors. They must be loaded into an execution target and run. Normal execution of tests results in an introductory message that summarizes the test objective, possibly some informative comments about the test progress, a final message giving pass/fail status, and graceful, silent termination. They may report "PASSED", "TENTATIVELY PASSED", "FAILED", OR "NOT APPLICABLE".

A test that fails to compile and bind, including compiling and binding any foundation code on which it depends is graded as "failed", unless the test includes features that need not be supported by all implementations. For example, an implementation may reject the declaration of a numeric type that it does not support. Allowable cases are clearly stated in the Applicability Criteria of tests. Annex M of the Ada Standard requires implementations to document such implementation-defined characteristics.

A test that reports "FAILED" is graded as "failed" unless the ACAL, and possibly the ACAA, determine that the test is not applicable for the implementation.

A test that reports "PASSED" is graded as "passed" unless the test produces the pass message but fails to terminate gracefully (e.g., crashes, hangs, raises an unexpected exception, produces an earlier or later "FAILED" message). This kind of aberrant behavior may occur, for example, in certain tasking tests, where there are multiple threads of control. A pass status message may be produced by one thread, but another thread may asynchronously crash or fail to terminate properly.

A test that reports "NOT APPLICABLE" must be run by the implementation and is graded as "not applicable" unless it produces the not-applicable message and then fails to terminate gracefully.
A test that reports "TENTATIVELY PASSED" is graded as "passed" if the test results satisfy the pass/fail criteria in the test. Normally, verification requires manual inspection of the test output.

A test that fails to report, or produces only a partial report, will be graded as "failed" unless the ACAL, and possibly the ACAA, determine that the test is not applicable for the implementation.

5.6.2 Expected Results for Class B Tests

Class B tests are expected to be compiled but are not subject to further processing and are not intended to be executable. An implementation must correctly report each clearly marked error (the notation -- ERROR: occurs at the right hand side of the source). A multiple unit B test file generally will have errors only in one compilation unit per source file. Error messages must provide some means of specifying the location of an error, but they are not required to be in direct proximity with the -- ERROR: marking of the errors.

Some B-tests also include the notation -- OK to indicate constructs that must not be identified as errors. This is especially important since some constructs that were errors in [Ada83] are now legal in later versions of Ada.

Some B-tests exercise constructs whose correctness depends on source code that is textually separated (for example, a deferred constant and its full declaration). In these cases, it may be reasonable to report an error at both locations. Such cases are marked with -- OPTIONAL ERROR. These lines may be flagged as errors by some, but not all, implementations. Unless an optional error is marked as an error for the wrong reason, an error report (or lack of it) does not affect the pass/fail status of the test.

Some B-tests contain constructs where it would be reasonable for a compiler to report an error at one of several source locations. When it would not be appropriate for the ACATS to insist on a particular source location, all such source locations are marked with -- POSSIBLE ERROR: and an indication of which error set (if the test contains several) the location belongs to. In such cases, an implementation is considered to have properly reported the error if it reports an error at any of the places marked -- POSSIBLE ERROR: for a particular set. The implementation may flag more than one such place; this does not affect the pass/fail status of the test. However, the test is graded "failed" if no error is reported at any of the places marked -- POSSIBLE ERROR: for an error set.

A test is graded as "passed" if it reports each error in the test. The content of error messages is considered only to determine that they are indeed indications of errors (as opposed to warnings) and that they refer to the expected errors. The Reference Manual does not specify the form or content of error messages. In particular, a test with just one expected error is graded as "passed" if the test is rejected at compile time for any reason.

A test is graded as "failed" if it fails to report on each error in the test or if it marks legal code as incorrect.

5.6.3 Expected Results for Class L Tests

Class L tests are expected to be rejected before execution begins. They must be submitted to the compiler and to the linker/binder. If an executable is generated, then it must be submitted for execution. Unless otherwise documented, the test is graded as "failed" if it begins execution, regardless of whether any output is produced. (Twenty-eight L tests contain documentation indicating that they may execute. See below.)

In general, an L test is expected to be rejected at link/bind time. Some tests contain -- ERROR: indications; an implementation that reports an error associated with one of these lines is judged to have passed the test (provided, of course, that the link attempt fails).
The following tests are exceptions to the general rule that an L test must not execute:

Test LXE3002, for the Distributed Systems Annex, is a test that has two partitions, each of which may execute. As documented in the source code, this test is graded "failed" if both partitions report "TENTATIVELY PASSED". Other outcomes are graded as appropriate for Class L tests.

Tests LA14001..27 and LA20002 (twenty-seven core language tests), as documented in the source code, may execute if automatic recompilation is supported. These tests are graded as "passed" if they execute and report "PASSED". Other outcomes are graded as appropriate for Class L tests.

5.6.4 Inapplicable Tests

Each ACATS test has a test objective that is described in the test prologue. Some objectives address Ada language features that need not be supported by every Ada implementation (e.g., "check floating-point operations for digits 18"). These test programs generally also contain an explicit indication of their applicability and the expected behavior of an implementation for which they do not apply. Annex D, “Test Applicability Criteria” lists common reasons for a test to be inapplicable, and lists the tests affected.

A test may be inapplicable for an implementation given:

- appropriate ACATS grading criteria; or
- an ACAA ruling on a petition to accept a deviation from expected results.

Appropriate grading criteria include:

1. whether a test completes execution and reports "NOT APPLICABLE";
2. whether a test is rejected at compile or bind time for a reason that satisfies grading criteria stated in the test program.

All applicable test programs must be processed and passed.

5.6.5 Withdrawn Tests

From time to time, the ACAA determines that one or more tests included in a release of the ACATS should be withdrawn from the test suite. Tests that are withdrawn are not processed during a conformity assessment and are not considered when grading an implementation.

Usually, a test is withdrawn because an error has been discovered in it. A withdrawn test will not be reissued as a modified test, although it may be revised and reissued as a new test in the future.

Withdrawn tests are listed in the ACATS Modification List, which is maintained by the ACAA.

5.7 Addressing Problems or Issues

After all tests have been processed and graded, any remaining problems should be addressed. Test failures must be identified and resolved. This section discusses issues that are not due to implementation errors (bugs).

5.7.1 Typical Issues

Here are some typical causes of unexpected ACATS test failures (often resulting from clerical errors):

- Processing a test that is withdrawn;
- Processing the original version of a test that has been modified by the ACAA to correct a test error;
• Processing a test that is not applicable to the implementation (as explained in Section 5.6.4);
• Processing files (or tests, see Section 5.4.2) in an incorrect order;
• Processing tests when units required in the environment are not present.

Test result failures resulting from technical errors may include:
• Incorrect values in ImpDef, which provide inappropriate values to tests at run-time (refer to 5.2.1);
• Incorrect values in macro.dfs, which result in incorrectly customized tests (refer to 5.2.2);
• Need to modify a test (e.g., split a B-test).

Finally, occasionally a user discovers an error in a new ACATS test. More rarely, errors are uncovered by compiler advances in tests that are apparently stable. In either case, if users believe that a test is in error, they may file a dispute with the ACAL. The dispute process is described in the next section.

5.7.2 Deviation from Expected Results - Petition & Review

Each test indicates in its prologue what it expects from a conforming implementation. The result of processing a test is acceptable if and only if the result is explicitly allowed by the grading criteria for the test.

A user may challenge an ACATS test on the grounds of applicability or correctness. A challenger should submit a petition against the test program to an ACAL or to the ACAA, following the procedure and the format presented in [Pro01]. A petition must clearly state whether it is a claim that the test does not apply to the implementation or that the test is incorrect. The petition must indicate the specific section of code that is disputed and provide a full explanation of the reason for the dispute.

ACALs will forward petitions from their customers to the ACAA for decisions. The ACAA will evaluate the petitioner's claims and decide whether:
• the test is applicable to the implementation (i.e., deviation is not allowed);
• the test is not applicable to the implementation (i.e., deviation is allowed);
• the test should be repaired (deviation is allowed, and the modified test should be used for determining conformity assessment results);
• the test should be withdrawn (deviation is allowed and the test is not considered in determining conformity assessment results).

A deviation is considered to be a test failure unless a petition to allow the deviation has been accepted by the ACAA.

5.8 Reprocessing and Regrading

After all problems have been resolved, tests that failed can be reprocessed and regraded. This step completes the ACATS testing process.
Annex A: Version Description

ACATS 3.0 includes 3771 tests in 4387 files, not including foundation and other support units. From Version 2.6 to 3.0, 79 tests were added, comprising 162 files (including five foundation files). 73 tests (78 files) were modified. Two support files also were modified, and one was removed. 17 tests (19 files) were removed. 15 documentation files were replaced by 167 new documentation files.

The following sections present a detailed description of ACATS 3.0, as follows:

A.1, “Core Test Files”
A.2, “Specialized Needs Annex Test Files”
A.3, “Foundation Code Files”
A.4, “Documentation Files”
A.5, “Other Files”
A.6, “Tests With Special Requirements”
A.7, “Test Files Added Since ACATS 2.6”
A.8, “Documentation Files Added Since ACATS 2.6”
A.9, “Test Files Modified Since ACATS 2.6”
A.10, “Support Files Modified Since ACATS 2.6”
A.11, “Test Files Deleted Since ACATS 2.6”
A.12, “Other Files Deleted Since ACATS 2.6”

A.1 Core Test Files

The following files contain the tests for core language features of Ada; that is, for requirements specified in Sections 2 through 13, Annex A, Annex B and Annex J.

| a22006b.ada | a54b01a.ada | a83a06a.ada | ad7001b.ada |
| a22006c.ada | a54b02a.ada | a83a08a.ada | ad7001c0.ada |
| a22006d.ada | a55b12a.ada | a83c01c.ada | ad7001c1.ada |
| a26007a.tst  | a55b13a.ada | a83c01h.ada | ad7001d0.ada |
| a27003a.ada | a55b14a.ada | a83c01i.ada | ad7001d1.ada |
| a29003a.ada | a71004a.ada | a85007d.ada | ad7006a.ada |
| a2a031a.ada | a73001i.ada | a85013b.ada | ad7101a.ada |
| a33003a.ada | a73001j.ada | a87b59a.ada | ad7101c.ada |
| a34017c.ada | a74105b.ada | a95001c.ada | ad7102a.ada |
| a35101b.ada | a74106a.ada | a95074d.ada | ad7103a.ada |
| a35402a.ada | a74106b.ada | a97106a.ada | ad7103c.ada |
| a35801f.ada | a74106c.ada | a99006a.ada | ad7104a.ada |
| a35902c.ada | a74205e.ada | aa2010a.ada | ad7201a.ada |
| a38106d.ada | a74205f.ada | aa2012a.ada | ad7203b.ada |
| a38106e.ada | a83009a.ada | ac1015b.ada | ad7205b.ada |
| a49027a.ada | a83009b.ada | ac3106a.ada | ad8011a.tst |
| a49027b.ada | a83a02a.ada | ac3206a.ada | ada101a.ada |
| a49027c.ada | a83a02b.ada | ac3207a.ada | ae2113a.ada |
| b37102a.ada | b392006.a | b413003.a | b45204a.ada |
| b37104a.ada | b392007.a | b413004.a | b45205a.ada |
| b37106a.ada | b392008.a | b41324b.ada | b45206c.ada |
| b37201a.ada | b392009.a | b41325b.ada | b45207a.ada |
| b37201b.ada | b392010.a | b41327b.ada | b45207b.ada |
| b37203a.ada | b392011.a | b420001.a | b45207c.ada |
| b37301a.ada | b393001.a | b430001.a | b45207d.ada |
| b37301j.ada | b393002.a | b43002d.ada | b45207g.ada |
| b37302a.ada | b393003.a | b43002e.ada | b45207h.ada |
| b37303a.ada | b393004.a | b43002f.ada | b45207i.ada |
| b37309b.ada | b393005.a | b43002g.ada | b45207j.ada |
| b37310b.ada | b393006.a | b43002h.ada | b45207m.ada |
| b37311a.ada | b393007.a | b43002i.ada | b45207n.ada |
| b37401a.ada | b393008.a | b43002j.ada | b45207o.ada |
| b37409b.ada | b393009.a | b43002k.ada | b45207p.ada |
| b380001.a | b393010.a | b43005a.ada | b45207s.ada |
| b38003a.ada | b394001.a | b43005b.ada | b45207t.ada |
| b38003b.ada | b394a01.a | b43005f.ada | b45207u.ada |
| b38003c.ada | b394a02.a | b431001.a | b45207v.ada |
| b38003d.ada | b394a03.a | b431002.a | b45208a.ada |
| b38008a.ada | b394a04.a | b431003.a | b45208b.ada |
| b38008b.ada | b394a05.a | b431004.a | b45208c.ada |
| b38009a.ada | b3a0001.a | b43101a.ada | b45208g.ada |
| b38009d.ada | b3a0002.a | b43102a.ada | b45208h.ada |
| b38101a.ada | b3a0003.a | b43102b.ada | b45208i.ada |
| b38101b.ada | b3a0004.a | b43105c.ada | b45208m.ada |
| b38101c.ada | b3a2002.a | b432001.a | b45208n.ada |
| b38103a.ada | b3a2003.a | b43201a.ada | b45208s.ada |
| b38103b.ada | b3a2004.a | b43201c.ada | b45208t.ada |
| b38103c.ada | b3a2005.a | b43201d.ada | b45209a.ada |
| b38103cl.ada | b3a2006.a | b43202a.ada | b45209b.ada |
| b38103c2.ada | b3a2007.a | b43202c.ada | b45209c.ada |
| b38103c3.ada | b3a2008.a | b43209b.ada | b45209d.ada |
| b38103d.ada | b3a2009.a | b43221a.ada | b45209e.ada |
| b38103e.ada | b3a2010.a | b43221b.ada | b45209f.ada |
| b38103e1.ada | b3a2011.a | b43223a.ada | b45209g.ada |
| b38105a.ada | b3a2012.a | b433001.a | b45209h.ada |
| b38105b.ada | b3a2013.a | b44001a.ada | b45209i.ada |
| b38203a.ada | b3a2014.a | b44001b.ada | b45209j.ada |
| b390001.a | b3a2015.a | b44002b.ada | b45209k.ada |
| b391001.a | b3a2016.a | b44002c.ada | b45221a.ada |
| b391002.a | b41101a.ada | b44004a.ada | b45261a.ada |
| b391003.a | b41101c.ada | b44004b.ada | b45261b.ada |
| b391004.a | b41201a.ada | b44004c.ada | b45261c.ada |
| b392001.a | b41201c.ada | b44004d.ada | b45261d.ada |
| b392002.a | b41202c.ada | b44004e.ada | b45301a.ada |
| b392003.a | b41202d.ada | b45102a.ada | b45301b.ada |
| b392004.a | b413001.a | b45116a.ada | b45301c.ada |
| b392005.a | b413002.a | b45121a.ada | b45302a.ada |
A.1  Core Test Files 3 January 2008  52

b45341a.ada  b45013a.ada  b55a01e.ada  b62001c.ada
b455002.a   b4a010c.ada  b55a01j.ada  b62001d.ada
b45501a.ada  b4a016a.ada  b55a01k.ada  b63001a.ada
b45501b.ada  b51001a.ada  b55a01l.ada  b63001b.ada
b45501c.ada  b51004b.ada  b55a01n.ada  b63005a.ada
b45522a.ada  b51004c.ada  b55a01o.ada  b63005b.ada
b45537a.ada  b52002a.ada  b55a01t.ada  b63006a.ada
b45601a.ada  b52002b.ada  b55a01u.ada  b63009a.ada
b45625a.ada  b52002c.ada  b55a01v.ada  b63009b.ada
b45661a.ada  b52002d.ada  b55b01a.ada  b63009c0.ada
b460001.a   b52002e.ada  b55b01b.ada  b63009c1.ada
b460002.a   b52002f.ada  b55b09b.ada  b63009c2.ada
b460004.a   b52002g.ada  b55b09c.de  b63009c3.ada
b460005.a   b52004a.ada  b55b09d.de  b63103a.ada
b460006.a   b52004b.ada  b55b12b.ada  b64002a.ada
b46002a.ada  b52004c.ada  b55b12c.ada  b64002c.ada
b46003a.ada  b52004d.de  b55b17a.ada  b64003a.ada
b46004a.ada  b52004e.de  b55b17b.ada  b64004a.ada
b46004b.ada  b53001a.ada  b55b17c.ada  b64004b.ada
b46004c.ada  b53001b.ada  b55b18a.ada  b64004c.ada
b46004d.ada  b53002a.ada  b56001a.ada  b64004d.ada
b46004e.ada  b53002b.ada  b56001d.ada  b64004e.ada
b46005a.ada  b53009a.ada  b56001e.de  b64004f.ada
b47001a.ada  b53009b.ada  b56001f.de  b641001.a
b480001.a   b53009c.ada  b56001g.de  b64101a.ada
b48001a.ada  b54a01b.ada  b56001h.de  b650001.a
b48001b.ada  b54a01f.de  b57001a.de  b650002.a
b48002a.ada  b54a01l.de  b57001a.de  b650003.a
b48002b.ada  b54a01l.de  b57001c.de  b65001a.de
b48002c.ada  b54a05a.ada  b57001d.de  b65002a.de
b48002d.ada  b54a05b.ada  b58001a.de  b65002b.de
b48002e.ada  b54a10a.ada  b58003a.de  b660001.a
b48002g.ada  b54a12a.ada  b58003b.de  b660002.a
b48003a.ada  b54a20a.ada  b59001a.de  b66001a.de
b48003b.ada  b54a21a.ada  b59001c.de  b66001b.de
b48003d.ada  b54a25a.ada  b59001d.de  b66001c.de
b48003e.ada  b54a60a.ada  b59001e.de  b66001d.de
b490001.a   b54a60b.ada  b59001f.de  b6700010.a
b490002.a   b54b01b.tst  b59001g.de  b6700011.am
b49002a.ada  b54b01c.de  b59001h.de  b6700012.a
b49004a.ada  b54b02b.de  b59001i.de  b6700013.a
b49005a.ada  b54b02c.de  b610001.a  b67001a.de
b49007a.ada  b54b02d.de  b610002.a  b67001b.de
b49007b.ada  b54b04a.ada  b61001f.de  b67001c.de
b49008a.ada  b54b04b.de  b61005a.de  b67001d.de
b49008c.ada  b54b05a.de  b61006a.de  b67001h.de
b49009b.ada  b54b06a.de  b61011a.de  b67001i.de
b49009c.ada  b55a01a.de  b62001a.de  b67001j.de
b49010a.ada  b55a01d.de  b62001b.de  b67001k.de
<table>
<thead>
<tr>
<th>File Name</th>
<th>File Name</th>
<th>File Name</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>b67004a.ada</td>
<td>b7310012.a</td>
<td>b8300023.a</td>
<td>b8310012.a</td>
</tr>
<tr>
<td>b71001a.ada</td>
<td>b7310013.a</td>
<td>b8300024.a</td>
<td>b831002.a</td>
</tr>
<tr>
<td>b71001b.ada</td>
<td>b7310014.a</td>
<td>b8300025.am</td>
<td>b8310030.a</td>
</tr>
<tr>
<td>b71001c.ada</td>
<td>b7310015.a</td>
<td>b83001a.ada</td>
<td>b8310031.a</td>
</tr>
<tr>
<td>b71001d.ada</td>
<td>b7310016.am</td>
<td>b83003a.ada</td>
<td>b8310032.a</td>
</tr>
<tr>
<td>b71001f.ada</td>
<td>b731a01.a</td>
<td>b83003b0.ada</td>
<td>b8310033.a</td>
</tr>
<tr>
<td>b71001g.ada</td>
<td>b731a02.a</td>
<td>b83003b1.ada</td>
<td>b83a01a.ada</td>
</tr>
<tr>
<td>b71001h.ada</td>
<td>b740001.a</td>
<td>b83003b2.ada</td>
<td>b83a01b.ada</td>
</tr>
<tr>
<td>b71001i.ada</td>
<td>b740002.a</td>
<td>b83003b3.ada</td>
<td>b83a01c.ada</td>
</tr>
<tr>
<td>b71001j.ada</td>
<td>b7400030.a</td>
<td>b83003b4.ada</td>
<td>b83a05a.ada</td>
</tr>
<tr>
<td>b71001l.ada</td>
<td>b7400031.a</td>
<td>b83003c.ada</td>
<td>b83a06b.ada</td>
</tr>
<tr>
<td>b71001m.ada</td>
<td>b7400032.am</td>
<td>b83004a.ada</td>
<td>b83a06h.ada</td>
</tr>
<tr>
<td>b71001n.ada</td>
<td>b74001a.ada</td>
<td>b83004b0.ada</td>
<td>b83a07a.ada</td>
</tr>
<tr>
<td>b71001o.ada</td>
<td>b74001b.ada</td>
<td>b83004b1.ada</td>
<td>b83a07b.ada</td>
</tr>
<tr>
<td>b71001p.ada</td>
<td>b74101a.ada</td>
<td>b83004b2.ada</td>
<td>b83a07c.ada</td>
</tr>
<tr>
<td>b71001r.ada</td>
<td>b74101b.ada</td>
<td>b83004b3.ada</td>
<td>b83a08b.ada</td>
</tr>
<tr>
<td>b71001t.ada</td>
<td>b74103a.ada</td>
<td>b83004c0.ada</td>
<td>b83a09a.ada</td>
</tr>
<tr>
<td>b71001u.ada</td>
<td>b74103d.ada</td>
<td>b83004e1.ada</td>
<td>b83b01a.ada</td>
</tr>
<tr>
<td>b71001v.ada</td>
<td>b74103e.ada</td>
<td>b83004e2.ada</td>
<td>b83b02c.ada</td>
</tr>
<tr>
<td>b7200010.a</td>
<td>b74103g.ada</td>
<td>b83004d0.ada</td>
<td>b83c01a.ada</td>
</tr>
<tr>
<td>b7200011.a</td>
<td>b74103i.ada</td>
<td>b83004d1.ada</td>
<td>b83c01b.ada</td>
</tr>
<tr>
<td>b7200012.a</td>
<td>b74104a.ada</td>
<td>b83004d2.ada</td>
<td>b83c01c.ada</td>
</tr>
<tr>
<td>b7200013.a</td>
<td>b74105a.ada</td>
<td>b83004d3.ada</td>
<td>b83c01d.ada</td>
</tr>
<tr>
<td>b7200014.a</td>
<td>b74105c.ada</td>
<td>b83006a.ada</td>
<td>b83c01e0.ada</td>
</tr>
<tr>
<td>b7200015.a</td>
<td>b74201a.ada</td>
<td>b83006b.ada</td>
<td>b83c01e1.ada</td>
</tr>
<tr>
<td>b7200016.a</td>
<td>b74202a.ada</td>
<td>b83008a.ada</td>
<td>b83c01e2.ada</td>
</tr>
<tr>
<td>b730001.a</td>
<td>b74202b.ada</td>
<td>b83008b.ada</td>
<td>b83c01e3.ada</td>
</tr>
<tr>
<td>b730002.a</td>
<td>b74202c.ada</td>
<td>b83011a.ada</td>
<td>b83c01f0.ada</td>
</tr>
<tr>
<td>b730003.a</td>
<td>b74202d.ada</td>
<td>b83023b.ada</td>
<td>b83c01f1.ada</td>
</tr>
<tr>
<td>b730004.a</td>
<td>b74203b.ada</td>
<td>b83024b.ada</td>
<td>b83c01f2.ada</td>
</tr>
<tr>
<td>b730005.a</td>
<td>b74203c.ada</td>
<td>b83024f0.ada</td>
<td>b83c01f3.ada</td>
</tr>
<tr>
<td>b7300060.a</td>
<td>b74203d.ada</td>
<td>b83024f1.ada</td>
<td>b83c01f4.ada</td>
</tr>
<tr>
<td>b7300061.a</td>
<td>b74203e.ada</td>
<td>b83024f2.ada</td>
<td>b83c01f5.ada</td>
</tr>
<tr>
<td>b7300062.a</td>
<td>b74205a.ada</td>
<td>b83024f3.ada</td>
<td>b83c01f6.ada</td>
</tr>
<tr>
<td>b7300063.am</td>
<td>b74207a.ada</td>
<td>b83026b.ada</td>
<td>b83c11a.ada</td>
</tr>
<tr>
<td>b73001a.ada</td>
<td>b74304a.ada</td>
<td>b83027b.ada</td>
<td>b83f02a.ada</td>
</tr>
<tr>
<td>b73001b.ada</td>
<td>b74304b.ada</td>
<td>b83027d.ada</td>
<td>b83f02b.ada</td>
</tr>
<tr>
<td>b73001c.ada</td>
<td>b74304c.ada</td>
<td>b83028b.ada</td>
<td>b83f02c.ada</td>
</tr>
<tr>
<td>b73001d.ada</td>
<td>b74404a.ada</td>
<td>b83029b.ada</td>
<td>b840001.a</td>
</tr>
<tr>
<td>b73001e.ada</td>
<td>b74404b.ada</td>
<td>b83030b.ada</td>
<td>b84001a.ada</td>
</tr>
<tr>
<td>b73001f.ada</td>
<td>b74409a.ada</td>
<td>b83030d.ada</td>
<td>b84002b.ada</td>
</tr>
<tr>
<td>b73001g.ada</td>
<td>b750a01.a</td>
<td>b83031b.ada</td>
<td>b84004a.ada</td>
</tr>
<tr>
<td>b73001h.ada</td>
<td>b750a02.a</td>
<td>b83031d.ada</td>
<td>b84005b.ada</td>
</tr>
<tr>
<td>b73004a.ada</td>
<td>b750a03.a</td>
<td>b83031f.ada</td>
<td>b84006a.ada</td>
</tr>
<tr>
<td>b73004b0.ada</td>
<td>b810001.a</td>
<td>b83032b.ada</td>
<td>b84007a.ada</td>
</tr>
<tr>
<td>b73004b1.ada</td>
<td>b83001a.ada</td>
<td>b83033b.ada</td>
<td>b84008b.ada</td>
</tr>
<tr>
<td>b73004b2.ada</td>
<td>b830020.a</td>
<td>b83041e.ada</td>
<td>b85001a.ada</td>
</tr>
<tr>
<td>b7310010.a</td>
<td>b830021.a</td>
<td>b8310010.a</td>
<td>b85001b.ada</td>
</tr>
<tr>
<td>b7310011.a</td>
<td>b830022.a</td>
<td>b8310011.a</td>
<td>b85001c.ada</td>
</tr>
</tbody>
</table>
b85001d.ada  b91003a.ada  b95080c.ada  b99001a.ada
b85001e.ada  b91003b.ada  b95081a.ada  b99001b.ada
b85001f.ada  b91003c.ada  b95082a.ada  b99002a.ada
b85001g.ada  b91003d.ada  b95082b.ada  b99002b.ada
b85001h.ada  b91003e.ada  b95082c.ada  b99002c.ada
b85001i.ada  b91004a.ada  b95082d.ada  b99003a.ada
b85001j.ada  b91005a.ada  b95082e.ada  b9a001a.ada
b85001k.ada  b92001a.ada  b95082f.ada  b9a001b.ada
b85001l.ada  b92001b.ada  b95083a.ada  ba1001a.ada
b85002a.ada  b940001.a  b95094a.ada  ba1001a1.ada
b85003a.ada  b940002.a  b95094b.ada  ba1001a4.ada
b85003b.ada  b940003.a  b95094c.ada  ba1001ac.ada
b85004a.ada  b940004.a  b951001.a  ba1001d.ada
b85008f.ada  b940005.a  b952001.a  ba1010a0.ada
b85008g.ada  b940006.a  b952002.a  ba1010a1.ada
b85008h.ada  b940007.a  b952003.a  ba1010a2.ada
b85010a.ada  b95001a.ada  b952004.a  ba1010a3.ada
b85010b.ada  b95001b.ada  b954001.a  ba1010b0.ada
b85012a.ada  b95001d.ada  b954003.a  ba1010b1.ada
b85013c.ada  b95002a.ada  b954004.a  ba1010b2.ada
b85013d.ada  b95003a.ada  b960001.a  ba1010b4.ada
b85015a.ada  b95004a.ada  b96002a.ada  ba1010b5.ada
b8510010.a  b95004b.ada  b97102b.ada  ba1010b6.ada
b8510011.a  b95006a.ada  b97102c.ada  ba1010b7.ada
b8510012.am  b95006b.ada  b97102d.ada  ba1010b8.ada
b854001.a  b95006c.ada  b97102f.ada  ba1010c0.ada
b86001a0.ada  b95006d.ada  b97102g.ada  ba1010c1.ada
b86001a1.ada  b95007a.ada  b97102h.ada  ba1010c2.ada
b87b23b.ada  b95007b.ada  b97102i.ada  ba1010c3.ada
b87b26a.ada  b95020a.ada  b97103a.ada  ba1010c4.ada
b87b48c.ada  b95020b0.ada  b97103b.ada  ba1010c5.ada
b91001b.ada  b95020b1.ada  b97103d.ada  ba1010c6.ada
b91001c.ada  b95020b2.ada  b97103e.ada  ba1010d0.ada
b91001d.ada  b95030a.ada  b97103f.ada  ba1010d1.ada
b91001e.ada  b95031a.ada  b97103g.ada  ba1010d2.ada
b91001f.ada  b95032a.ada  b97104a.ada  ba1010d3.ada
b91001g.ada  b95061a.ada  b97104b.ada  ba1010e0.ada
b91002a.ada  b95061b.ada  b97104c.ada  ba1010e1.ada
b91002b.ada  b95061c.ada  b97104d.ada  ba1010e2.ada
b91002c.ada  b95061d.ada  b97104e.ada  ba1010e3.ada
b91002d.ada  b95061e.ada  b97104f.ada  ba1010e4.ada
b91002e.ada  b95061f.ada  b97104g.ada  ba1010e5.ada
b91002f.ada  b95061g.ada  b97107a.ada  ba1010e6.ada
b91002g.ada  b95062a.ada  b97108a.ada  ba1010f0.ada
b91002h.ada  b95063a.ada  b97108b.ada  ba1010f1.ada
b91002i.ada  b95064a.ada  b97109a.ada  ba1010f3.ada
b91002j.ada  b95068a.ada  b97110a.ada  ba1010f4.ada
b91002k.ada  b95070a.ada  b97110b.ada  ba1010f5.ada
b91002l.ada  b95080a.ada  b97111a.ada  ba1010f6.ada

A.1 Core Test Files
| ba120144.a | ba2011a9.ada | bb2001a.ada | bc1202g.ada |
| ba120145.a | ba2013a.ada | bb2002a.ada | bc1203a.ada |
| ba120146.a | ba2013b.ada | bb2003a.ada | bc1205a.ada |
| ba120147.a | ba21002.a | bb2003b.ada | bc1206a.ada |
| ba120150.a | ba210030.a | bb2003c.ada | bc1207a.ada |
| ba120151.a | ba210031.a | bb3001a.ada | bc1208a.ada |
| ba120152.a | ba210032.a | bb3002a.ada | bc1226a.ada |
| ba120153.a | ba210033.a | bc1001a.ada | bc1230a.ada |
| ba120154.a | ba210034.a | bc1002a.ada | bc1303a.ada |
| ba120160.a | ba210035.a | bc1005a.ada | bc1303b.ada |
| ba120161.a | ba210040.a | bc1008a.ada | bc1303c.ada |
| ba120162.a | ba210041.a | bc1008b.ada | bc1303d.ada |
| ba120163.a | ba210042.a | bc1008c.ada | bc1303e.ada |
| ba120164.a | ba210043.a | bc1009a.ada | bc1303f.ada |
| ba120165.a | ba210044.a | bc1011a.ada | bc1303g.ada |
| ba120166.a | ba210045.am | bc1011b.ada | bc1306a.ada |
| ba13b01.a | ba21a01.a | bc1011c.ada | bc2001b.ada |
| ba13b02.a | ba21a02.a | bc1012a.ada | bc2001c.ada |
| ba15001.a | ba21a03.a | bc1013a.ada | bc2001d.ada |
| ba150020.a | ba3001a0.ada | bc1014a.ada | bc2001e.ada |
| ba150021.a | ba3001a1.ada | bc1014b.ada | bc2004a.ada |
| ba150022.a | ba3001a2.ada | bc1016a.ada | bc2004b.ada |
| ba150023.a | ba3001a3.ada | bc1016b.ada | bc30001.a |
| ba150024.a | ba3001b0.ada | bc1101a.ada | bc3001a.ada |
| ba150025.a | ba3001b1.ada | bc1102a.ada | bc3002a.ada |
| ba150026.a | ba3001c0.ada | bc1103a.ada | bc3002b.ada |
| ba150027.a | ba3001c1.ada | bc1106a.ada | bc3002c.ada |
| ba150028.a | ba3001e0.ada | bc1107a.ada | bc3002d.ada |
| ba150029.am | ba3001e1.ada | bc1109a.ada | bc3002e.ada |
| ba16001.a | ba3001e2.ada | bc1109b.ada | bc3005a.ada |
| ba16002.a | ba3001e3.ada | bc1109c.ada | bc3005b.ada |
| ba2001a.ada | ba3001f0.ada | bc1109d.ada | bc3005c.ada |
| ba2001b.ada | ba3001f1.ada | bc1110a.ada | bc3006a.ada |
| ba2001c.ada | ba3001f2.ada | bc1201a.ada | bc3009c.ada |
| ba2001d.ada | ba3001f3.ada | bc1201b.ada | bc3011b.ada |
| ba2001f0.ada | ba3006a0.ada | bc1201c.ada | bc3013a.ada |
| ba2001f1.ada | ba3006a1.ada | bc1201d.ada | bc3016g.ada |
| ba2001f2.ada | ba3006a2.ada | bc1201e.ada | bc3018a.ada |
| ba2003b0.ada | ba3006a3.ada | bc1201f.ada | bc3101a.ada |
| ba2003b1.ada | ba3006a4.ada | bc1201g.ada | bc3101b.ada |
| ba2011a0.ada | ba3006a5.ada | bc1201h.ada | bc3102a.ada |
| ba2011a1.ada | ba3006a6.ada | bc1201i.ada | bc3102b.ada |
| ba2011a2.ada | ba3006b0.ada | bc1201j.ada | bc3103b.ada |
| ba2011a3.ada | ba3006b1.ada | bc1201k.ada | bc3123c.ada |
| ba2011a4.ada | ba3006b2.ada | bc1201l.ada | bc3201a.ada |
| ba2011a5.ada | ba3006b3.ada | bc1202a.ada | bc3201b.ada |
| ba2011a6.ada | ba3006b4.ada | bc1202c.ada | bc3201c.ada |
| ba2011a7.ada | bb10001.a | bc1202e.ada | bc3202a.ada |
| ba2011a8.ada | bb20001.a | bc1202f.ada | bc3202b.ada |
| bc3202c.ada  | bc3502l.ada  | bc70003.a  | bd4001a.ada  |
| bc3202d.ada  | bc3502m.ada  | bc70004.a  | bd4002a.ada  |
| bc3205c.ada  | bc3502n.ada  | bc70005.a  | bd4003a.ada  |
| bc3301a.ada  | bc3502o.ada  | bc70006.a  | bd4003b.ada  |
| bc3301b.ada  | bc3503a.ada  | bc70007.a  | bd4003c.ada  |
| bc3302a.ada  | bc3503c.ada  | bc70008.a  | bd4006a.ada  |
| bc3302b.ada  | bc3503d.ada  | bc70009.a  | bd4007a.ada  |
| bc3303a.ada  | bc3503e.ada  | bc70010.a  | bd4009a.ada  |
| bc3304a.ada  | bc3503f.ada  | bc1b01a.ada| bd4011a.ada  |
| bc3401a.ada  | bc3604a.ada  | bc1b02b.ada| bd5001a.ada  |
| bc3401b.ada  | bc3604b.ada  | bc1b03c.ada| bd5005a.ada  |
| bc3402a.ada  | bc3607a.ada  | bc1b05e.ada| bd5005d.ada  |
| bc3402b.ada  | bc40002.a    | bc1b06j.ada| bd5102a.ada  |
| bc3403a.ada  | bc50001.a    | bd2001b.ada| bd5102b.ada  |
| bc3403b.ada  | bc50002.a    | bd2a01h.ada| bd5103a.ada  |
| bc3403c.ada  | bc50003.a    | bd2a02a.tst| bd5104a.ada  |
| bc3404a.ada  | bc50004.a    | bd2a03a.ada| bd7001a.ada  |
| bc3404b.ada  | bc51002.a    | bd2a03b.ada| bd7101h.ada  |
| bc3404c.ada  | bc51003.a    | bd2a06a.ada| bd7201c.ada  |
| bc3404d.ada  | bc51004.a    | bd2a25a.ada| bd7203a.ada  |
| bc3404e.ada  | bc51005.a    | bd2a35a.ada| bd7204a.ada  |
| bc3404f.ada  | bc51006.a    | bd2a45a.ada| bd7205a.ada  |
| bc3405a.ada  | bc51007.a    | bd2a55a.ada| bd7301a.ada  |
| bc3405b.ada  | bc51011.a    | bd2a55b.ada| bd7302a.ada  |
| bc3405d.ada  | bc51012.a    | bd2a67a.ada| bd8001a.tst  |
| bc3405e.ada  | bc51013.a    | bd2a77a.ada| bd8002a.tst  |
| bc3405f.ada  | bc51015.a    | bd2a85a.ada| bd8003a.tst  |
| bc3501a.ada  | bc51016.a    | bd2a85b.ada| bd8004a.tst  |
| bc3501b.ada  | bc51017.a    | bd2b01c.ada| bd8004b.tst  |
| bc3501c.ada  | bc51018.a    | bd2b02a.ada| bd8004c.tst  |
| bc3501d.ada  | bc51019.a    | bd2b03a.ada| bd0a01.a    |
| bc3501e.ada  | bc51020.a    | bd2b03b.ada| bd0d01.a    |
| bc3501f.ada  | bc51b01.a    | bd2b03c.ada| bd2d02.a    |
| bc3501g.ada  | bc51b02.a    | bd2c01d.tst| bd2d03.a    |
| bc3501h.ada  | bc51c01.a    | bd2c02a.tst| bd60001.a   |
| bc3501i.ada  | bc51c02.a    | bd2c03a.tst| bd60002.a   |
| bc3501j.ada  | bc53001.a    | bd2d01c.ada| bd60003.a   |
| bc3501k.ada  | bc53002.a    | bd2d01d.ada| bd60004.a   |
| bc3502a.ada  | bc54001.a    | bd2d02a.ada| bd60005.a   |
| bc3502b.ada  | bc54002.a    | bd2d03a.ada| bd60006.a   |
| bc3502c.ada  | bc54003.a    | bd2d03b.ada| bd60007.a   |
| bc3502d.ada  | bc54a01.a    | bd3001a.ada| bd60008.a   |
| bc3502e.ada  | bc54a02.a    | bd3001b.ada| bd60009.a   |
| bc3502f.ada  | bc54a03.a    | bd3001c.ada| bd60010.a   |
| bc3502g.ada  | bc54a04.a    | bd3002a.ada| bd2101e.ada |
| bc3502h.ada  | bc54a05.a    | bd3003a.ada| bd2101j.ada |
| bc3502i.ada  | bc54a06.a    | bd3003b.ada| bd2114a.ada |
| bc3502j.ada  | bc70001.a    | bd3012a.ada| bd2116a.ada |
| bc3502k.ada  | bc70002.a    | bd3013a.ada| bd2208a.ada |
be3002a.ada  c32001a.ada  c34006l.ada  c35502d.tst
be3002b.ada  c32001b.ada  c34007a.ada  c35502e.ada
be3205a.ada  c32001c.ada  c34007d.ada  c35502f.tst
be3301c.ada  c32001d.ada  c34007f.ada  c35502g.ada
be3606c.ada  c32001e.ada  c34007g.ada  c35502h.ada
be3703a.ada  c32107a.ada  c34007i.ada  c35502i.ada
be3802a.ada  c32107c.ada  c34007j.ada  c35502j.ada
be3803a.ada  c32108a.ada  c34007m.ada  c35502k.ada
be3902a.ada  c32108b.ada  c34007p.ada  c35502l.ada
be3903a.ada  c32111a.ada  c34007r.ada  c35502m.ada
bxa8001.a  c32111b.ada  c34007s.ada  c35502n.ada
bxac001.a  c32112b.ada  c34007u.ada  c35502o.ada
bxac002.a  c32113a.ada  c34007v.ada  c35502p.ada
bxac003.a  c32115a.ada  c34008a.ada  c35503a.ada
bxac005.a  c32115b.ada  c34009a.ada  c35503b.ada
c23001a.ada  c330001.a  c34009d.ada  c35503c.ada
c23003a.tst  c330002.a  c34009f.ada  c35503d.tst

c23003b.tst  c332001.a  c34009g.ada  c35503e.ada

c23003g.tst  c34001.a  c34009j.ada  c35503f.tst

c23003i.tst  c34001a.ada  c34009l.ada  c35503g.ada

c23006a.ada  c34001c.ada  c34011b.ada  c35503h.ada

c23006b.ada  c34001d.ada  c34012a.ada  c35503k.ada

c23006c.ada  c34001f.ada  c34014a.ada  c35503l.ada

c23006d.ada  c34002a.ada  c34014c.ada  c35503o.ada

c23006e.ada  c34002c.ada  c34014e.ada  c35503p.ada

c23006f.ada  c34003a.ada  c34014g.ada  c35504a.ada

c23006g.ada  c34003c.ada  c34014h.ada  c35504b.ada

c24002d.ada  c34004a.ada  c34014n.ada  c35505c.ada

c24003a.ada  c34004c.ada  c34014p.ada  c35505e.ada

c24003b.ada  c34005a.ada  c34014r.ada  c35505f.ada

c24003c.ada  c34005c.ada  c34014t.ada  c35507a.ada

c24106a.ada  c34005d.ada  c34014u.ada  c35507b.ada

c24202d.ada  c34005f.ada  c34018a.ada  c35507c.ada

c24203a.ada  c34005g.ada  c340a01.a  c35507e.ada

c24203b.ada  c34005i.ada  c340a02.a  c35507g.ada

c24207a.ada  c34005j.ada  c341a01.a  c35507h.ada

c24211a.ada  c34005l.ada  c341a02.a  c35507i.ada

c250001.au  c34005m.ada  c341a03.a  c35507j.ada

c250002.au  c34005o.ada  c341a04.a  c35507k.ada

c25001a.ada  c34005p.ada  c35003a.ada  c35507l.ada

c25001b.ada  c34005r.ada  c35003b.ada  c35507m.ada

c26060a.ada  c34005s.ada  c35003d.ada  c35507n.ada

c26060a.ada  c34005u.ada  c35102a.ada  c35507o.ada

c2a001.a.ada  c34005v.ada  c352001.a  c35507p.ada

c2a001b.ada  c34006a.ada  c354002.a  c35508a.ada

c2a001c.ada  c34006d.ada  c354003.a  c35508b.ada

c2a002a.ada  c34006f.ada  c35502a.ada  c35508c.ada

c2a008a.ada  c34006g.ada  c35502b.ada  c35508e.ada

c2a021b.ada  c34006j.ada  c35502c.ada  c35508g.ada
<table>
<thead>
<tr>
<th>File 1</th>
<th>File 2</th>
<th>File 3</th>
<th>File 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>c393011.a</td>
<td>c41207a.ada</td>
<td>c43004c.ada</td>
<td>c43214e.ada</td>
</tr>
<tr>
<td>c393012.a</td>
<td>c413001.a</td>
<td>c431001.a</td>
<td>c43214f.ada</td>
</tr>
<tr>
<td>c393a02.a</td>
<td>c413002.a</td>
<td>c43103a.ada</td>
<td>c43215a.ada</td>
</tr>
<tr>
<td>c393a03.a</td>
<td>c413003.a</td>
<td>c43103b.ada</td>
<td>c43215b.ada</td>
</tr>
<tr>
<td>c393a05.a</td>
<td>c413004.a</td>
<td>c43104a.ada</td>
<td>c43222a.ada</td>
</tr>
<tr>
<td>c393a06.a</td>
<td>c413005.a</td>
<td>c43105a.ada</td>
<td>c43224a.ada</td>
</tr>
<tr>
<td>c393b12.a</td>
<td>c41301a.ada</td>
<td>c43105b.ada</td>
<td>c433001.a</td>
</tr>
<tr>
<td>c393b13.a</td>
<td>c41303a.ada</td>
<td>c43106a.ada</td>
<td>c433a01.a</td>
</tr>
<tr>
<td>c393b14.a</td>
<td>c41303b.ada</td>
<td>c43107a.ada</td>
<td>c433a02.a</td>
</tr>
<tr>
<td>c394001.a</td>
<td>c41303c.ada</td>
<td>c43108a.ada</td>
<td>c433a03.a</td>
</tr>
<tr>
<td>c394002.a</td>
<td>c41303e.ada</td>
<td>c431a01.a</td>
<td>c433a04.a</td>
</tr>
<tr>
<td>c394003.a</td>
<td>c41303f.ada</td>
<td>c431a02.a</td>
<td>c44003d.ada</td>
</tr>
<tr>
<td>c3a0001.a</td>
<td>c41303g.ada</td>
<td>c431a03.a</td>
<td>c44003f.ada</td>
</tr>
<tr>
<td>c3a0002.a</td>
<td>c41303i.ada</td>
<td>c432001.a</td>
<td>c44003g.ada</td>
</tr>
<tr>
<td>c3a0003.a</td>
<td>c41303j.ada</td>
<td>c432002.a</td>
<td>c450001.a</td>
</tr>
<tr>
<td>c3a0004.a</td>
<td>c41303k.ada</td>
<td>c432003.a</td>
<td>c45112a.ada</td>
</tr>
<tr>
<td>c3a0005.a</td>
<td>c41303m.ada</td>
<td>c432004.a</td>
<td>c45112b.ada</td>
</tr>
<tr>
<td>c3a0006.a</td>
<td>c41303n.ada</td>
<td>c432005.a</td>
<td>c45113a.ada</td>
</tr>
<tr>
<td>c3a0007.a</td>
<td>c41303o.ada</td>
<td>c43204a.ada</td>
<td>c45114b.ada</td>
</tr>
<tr>
<td>c3a0008.a</td>
<td>c41303q.ada</td>
<td>c43204c.ada</td>
<td>c452001.a</td>
</tr>
<tr>
<td>c3a0009.a</td>
<td>c41303r.ada</td>
<td>c43204e.ada</td>
<td>c45201a.ada</td>
</tr>
<tr>
<td>c3a0010.a</td>
<td>c41303s.ada</td>
<td>c43204f.ada</td>
<td>c45201b.ada</td>
</tr>
<tr>
<td>c3a0011.a</td>
<td>c41303u.ada</td>
<td>c43204g.ada</td>
<td>c45202a.ada</td>
</tr>
<tr>
<td>c3a0012.a</td>
<td>c41303v.ada</td>
<td>c43204h.ada</td>
<td>c45210a.ada</td>
</tr>
<tr>
<td>c3a0012.a</td>
<td>c41303w.ada</td>
<td>c43204i.ada</td>
<td>c45211a.ada</td>
</tr>
<tr>
<td>c3a0012.am</td>
<td>c41304a.ada</td>
<td>c43205a.ada</td>
<td>c45220a.ada</td>
</tr>
<tr>
<td>c3a0013.a</td>
<td>c41304b.ada</td>
<td>c43205b.ada</td>
<td>c45220b.ada</td>
</tr>
<tr>
<td>c3a0014.a</td>
<td>c41306a.ada</td>
<td>c43205c.ada</td>
<td>c45220c.ada</td>
</tr>
<tr>
<td>c3a0015.a</td>
<td>c41306b.ada</td>
<td>c43205d.ada</td>
<td>c45220d.ada</td>
</tr>
<tr>
<td>c3a1001.a</td>
<td>c41306c.ada</td>
<td>c43205e.ada</td>
<td>c45220e.ada</td>
</tr>
<tr>
<td>c3a1002.a</td>
<td>c41307d.ada</td>
<td>c43205g.ada</td>
<td>c45220f.ada</td>
</tr>
<tr>
<td>c3a2001.a</td>
<td>c41309a.ada</td>
<td>c43205h.ada</td>
<td>c45231a.ada</td>
</tr>
<tr>
<td>c3a2002.a</td>
<td>c41320a.ada</td>
<td>c43205i.ada</td>
<td>c45231b.dep</td>
</tr>
<tr>
<td>c3a2003.a</td>
<td>c41321a.ada</td>
<td>c43205j.ada</td>
<td>c45231c.dep</td>
</tr>
<tr>
<td>c3a2a01.a</td>
<td>c41322a.ada</td>
<td>c43205k.ada</td>
<td>c45231d.tst</td>
</tr>
<tr>
<td>c3a2a02.a</td>
<td>c41323a.ada</td>
<td>c43206a.ada</td>
<td>c45232b.ada</td>
</tr>
<tr>
<td>c410001.a</td>
<td>c41324a.ada</td>
<td>c43207b.ada</td>
<td>c45242b.ada</td>
</tr>
<tr>
<td>c41101d.ada</td>
<td>c41325a.ada</td>
<td>c43207d.ada</td>
<td>c45251a.ada</td>
</tr>
<tr>
<td>c41103a.ada</td>
<td>c41326a.ada</td>
<td>c43208a.ada</td>
<td>c45252a.ada</td>
</tr>
<tr>
<td>c41103b.ada</td>
<td>c41327a.ada</td>
<td>c43208b.ada</td>
<td>c45252b.ada</td>
</tr>
<tr>
<td>c41104a.ada</td>
<td>c41328a.ada</td>
<td>c43209a.ada</td>
<td>c45253a.ada</td>
</tr>
<tr>
<td>c41105a.ada</td>
<td>c41301a.ada</td>
<td>c43210a.ada</td>
<td>c45262a.ada</td>
</tr>
<tr>
<td>c41107a.ada</td>
<td>c41302a.ada</td>
<td>c43211a.ada</td>
<td>c45262b.ada</td>
</tr>
<tr>
<td>c41201d.ada</td>
<td>c41304a.ada</td>
<td>c43212a.ada</td>
<td>c45262c.ada</td>
</tr>
<tr>
<td>c41203a.ada</td>
<td>c420001.a</td>
<td>c43212c.ada</td>
<td>c45262d.ada</td>
</tr>
<tr>
<td>c41203b.ada</td>
<td>c42006a.ada</td>
<td>c43214a.ada</td>
<td>c45264a.ada</td>
</tr>
<tr>
<td>c41204a.ada</td>
<td>c42007c.ada</td>
<td>c43214b.ada</td>
<td>c45264b.ada</td>
</tr>
<tr>
<td>c41205a.ada</td>
<td>c43003a.ada</td>
<td>c43214c.ada</td>
<td>c45264c.ada</td>
</tr>
<tr>
<td>c41206a.ada</td>
<td>c43004a.ada</td>
<td>c43214d.ada</td>
<td>c45265a.ada</td>
</tr>
<tr>
<td>File Path</td>
<td>File Path</td>
<td>File Path</td>
<td>File Path</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>c52005d.ada</td>
<td>c54a13c.ada</td>
<td>c62006a.ada</td>
<td>c64109g.ada</td>
</tr>
<tr>
<td>c52005e.ada</td>
<td>c54a13d.ada</td>
<td>c631001.a</td>
<td>c64109h.ada</td>
</tr>
<tr>
<td>c52005f.ada</td>
<td>c54a22a.ada</td>
<td>c640001.a</td>
<td>c64109i.ada</td>
</tr>
<tr>
<td>c52008a.ada</td>
<td>c54a23a.ada</td>
<td>c64002b.ada</td>
<td>c64109j.ada</td>
</tr>
<tr>
<td>c52008b.ada</td>
<td>c54a24a.ada</td>
<td>c64004g.ada</td>
<td>c64109k.ada</td>
</tr>
<tr>
<td>c52009a.ada</td>
<td>c54a24b.ada</td>
<td>c64005a.ada</td>
<td>c64109l.ada</td>
</tr>
<tr>
<td>c52009b.ada</td>
<td>c54a42a.ada</td>
<td>c64005b.ada</td>
<td>c64201b.ada</td>
</tr>
<tr>
<td>c52010a.ada</td>
<td>c54a42b.ada</td>
<td>c64005c.ada</td>
<td>c64201c.ada</td>
</tr>
<tr>
<td>c52011a.ada</td>
<td>c54a42c.ada</td>
<td>c64005d0.ada</td>
<td>c64202a.ada</td>
</tr>
<tr>
<td>c52011b.ada</td>
<td>c54a42d.ada</td>
<td>c64005da.ada</td>
<td>c65003a.ada</td>
</tr>
<tr>
<td>c52012a.ada</td>
<td>c54a42e.ada</td>
<td>c64005db.ada</td>
<td>c65003b.ada</td>
</tr>
<tr>
<td>c52012b.ada</td>
<td>c54a42f.ada</td>
<td>c64005dc.ada</td>
<td>c66002a.ada</td>
</tr>
<tr>
<td>c52012c.ada</td>
<td>c54a42g.ada</td>
<td>c641001.a</td>
<td>c66002c.ada</td>
</tr>
<tr>
<td>c52012d.ada</td>
<td>c55b03a.ada</td>
<td>c64103b.ada</td>
<td>c66002d.ada</td>
</tr>
<tr>
<td>c52012e.ada</td>
<td>c55b04a.ada</td>
<td>c64103c.ada</td>
<td>c66002e.ada</td>
</tr>
<tr>
<td>c52012f.ada</td>
<td>c55b05a.ada</td>
<td>c64103d.ada</td>
<td>c66002f.ada</td>
</tr>
<tr>
<td>c52013a.ada</td>
<td>c55b06a.ada</td>
<td>c64103e.ada</td>
<td>c66002g.ada</td>
</tr>
<tr>
<td>c52013b.ada</td>
<td>c55b07a.ada</td>
<td>c64104a.ada</td>
<td>c67002a.ada</td>
</tr>
<tr>
<td>c52013c.ada</td>
<td>c55b07b.ada</td>
<td>c64104b.ada</td>
<td>c67002b.ada</td>
</tr>
<tr>
<td>c52013d.ada</td>
<td>c55b10a.ada</td>
<td>c64104c.ada</td>
<td>c67002d.ada</td>
</tr>
<tr>
<td>c52013e.ada</td>
<td>c55b11a.ada</td>
<td>c64104d.ada</td>
<td>c67002e.ada</td>
</tr>
<tr>
<td>c52013f.ada</td>
<td>c55b11b.ada</td>
<td>c64104e.ada</td>
<td>c67003a.ada</td>
</tr>
<tr>
<td>c52013g.ada</td>
<td>c55b15a.ada</td>
<td>c64104f.ada</td>
<td>c67005a.ada</td>
</tr>
<tr>
<td>c52013h.ada</td>
<td>c55b16a.ada</td>
<td>c64104g.ada</td>
<td>c67005b.ada</td>
</tr>
<tr>
<td>c52013i.ada</td>
<td>c55c02a.ada</td>
<td>c64104h.ada</td>
<td>c67005c.ada</td>
</tr>
<tr>
<td>c52013j.ada</td>
<td>c55c02b.ada</td>
<td>c64104i.ada</td>
<td>c67005d.ada</td>
</tr>
<tr>
<td>c52013k.ada</td>
<td>c56002a.ada</td>
<td>c64104j.ada</td>
<td>c72001b.ada</td>
</tr>
<tr>
<td>c52013l.ada</td>
<td>c57003a.ada</td>
<td>c64104k.ada</td>
<td>c72002a.ada</td>
</tr>
<tr>
<td>c52013m.ada</td>
<td>c57004a.ada</td>
<td>c64104l.ada</td>
<td>c730001.a</td>
</tr>
<tr>
<td>c52013n.ada</td>
<td>c57005b.ada</td>
<td>c64104m.ada</td>
<td>c730002.a</td>
</tr>
<tr>
<td>c52013o.ada</td>
<td>c58004a.ada</td>
<td>c64104n.ada</td>
<td>c730003.a</td>
</tr>
<tr>
<td>c52013p.ada</td>
<td>c58004d.ada</td>
<td>c64104o.ada</td>
<td>c730004.a</td>
</tr>
<tr>
<td>c52013q.ada</td>
<td>c58004g.ada</td>
<td>c64105a.ada</td>
<td>c73002a.ada</td>
</tr>
<tr>
<td>c52013r.ada</td>
<td>c58005a.ada</td>
<td>c64105b.ada</td>
<td>c730a01.a</td>
</tr>
<tr>
<td>c52013s.ada</td>
<td>c58005b.ada</td>
<td>c64105c.ada</td>
<td>c730a02.a</td>
</tr>
<tr>
<td>c52013t.ada</td>
<td>c58005h.ada</td>
<td>c64105d.ada</td>
<td>c731001.a</td>
</tr>
<tr>
<td>c52013u.ada</td>
<td>c58006a.ada</td>
<td>c64106a.ada</td>
<td>c74004a.ada</td>
</tr>
<tr>
<td>c52013v.ada</td>
<td>c58006b.ada</td>
<td>c64106b.ada</td>
<td>c74203a.ada</td>
</tr>
<tr>
<td>c52014a.ada</td>
<td>c59002a.ada</td>
<td>c64106c.ada</td>
<td>c74206a.ada</td>
</tr>
<tr>
<td>c52014b.ada</td>
<td>c59002b.ada</td>
<td>c64106d.ada</td>
<td>c74207b.ada</td>
</tr>
<tr>
<td>c52014c.ada</td>
<td>c59002c.ada</td>
<td>c64107a.ada</td>
<td>c74208a.ada</td>
</tr>
<tr>
<td>c52014d.ada</td>
<td>c61008a.ada</td>
<td>c64108a.ada</td>
<td>c74208b.ada</td>
</tr>
<tr>
<td>c52014e.ada</td>
<td>c61009a.ada</td>
<td>c64109a.ada</td>
<td>c74209a.ada</td>
</tr>
<tr>
<td>c52014f.ada</td>
<td>c61010a.ada</td>
<td>c64109b.ada</td>
<td>c74210a.ada</td>
</tr>
<tr>
<td>c52014g.ada</td>
<td>c61010a.ada</td>
<td>c64109b.ada</td>
<td>c74210a.ada</td>
</tr>
<tr>
<td>c52014h.ada</td>
<td>c61010b.ada</td>
<td>c64109c.ada</td>
<td>c74211a.ada</td>
</tr>
<tr>
<td>c52014i.ada</td>
<td>c61010c.ada</td>
<td>c64109d.ada</td>
<td>c74211b.ada</td>
</tr>
<tr>
<td>c52014j.ada</td>
<td>c6103a.ada</td>
<td>c64109e.ada</td>
<td>c74302a.ada</td>
</tr>
<tr>
<td>c52014k.ada</td>
<td>c62003a.ada</td>
<td>c64109f.ada</td>
<td>c74302b.ada</td>
</tr>
</tbody>
</table>

A.1 Core Test Files
A.1 Core Test Files

3 January 2008

64
c97204a.ada  c9a011b.ada  ca110051.am  ca140230.a
c97204b.ada  ca1003a.ada  ca11006.a  ca140231.a
c97205a.ada  ca1004a.ada  ca11007.a  ca140232.am
ca1005a.ada  ca11008.a  ca140233.a
ca1006a.ada  ca11009.a  ca140280.a
ca1011a0.ada  ca11010.a  ca140281.a
ca1011a1.ada  ca11011.a  ca140282.a
ca1011a2.ada  ca11012.a  ca140283.am
ca1011a3.ada  ca11013.a  ca15003.a
ca1011a4.ada  ca11014.a  ca200020.a
ca1011a5.ada  ca11015.a  ca200021.a
ca1011a6.ada  ca11016.a  ca200022.am
ca1012a0.ada  ca11017.a  ca200030.a
ca1012a1.ada  ca11018.a  ca200031.am
ca1012a2.ada  ca11019.a  ca2001h0.ada
ca1012a3.ada  ca11020.a  ca2001h1.ada
ca1012a4.ada  ca11021.a  ca2001h2.ada
ca1012b0.ada  ca11022.a  ca2001h3.ada
ca1012b2.ada  ca110230.a  ca2002a0.ada
ca1012b4.ada  ca110231.a  ca2002a1.ada
ca1013a0.ada  ca110232.am  ca2002a2.ada
ca1013a1.ada  ca1102a0.ada  ca2003a0.ada
ca1013a2.ada  ca1102a1.ada  ca2003a1.ada
ca1013a3.ada  ca1102a2.ada  ca2004a0.ada
ca1013a4.ada  ca1106a.ada  ca2004a1.ada
ca1013a5.ada  ca1108a.ada  ca2004a2.ada
ca1013a6.ada  ca1108b.ada  ca2004a3.ada
ca1014a0.ada  ca11a01.a  ca2004a4.ada
ca1014a1.ada  ca11a02.a  ca2007a0.ada
ca1014a2.ada  ca11b01.a  ca2007a1.ada
ca1104a3.ada  ca11b02.a  ca2007a2.ada
c974012.a  ca120e0.ada  ca11c01.a  ca2007a3.ada
c974013.a  ca120e1.ada  ca11c02.a  ca2008a0.ada
c974014.a  ca120e2.ada  ca11c03.a  ca2008a1.ada
c974011.a  ca120e3.ada  ca11d010.a  ca2008a2.ada
ca1104a3.ada  ca11d011.a  ca2009a.ada
ca1202a0.ada  ca11d012.a  ca2009c0.ada
c980003.a  ca1202a1.ada  ca11d013.am  ca2009c1.ada
c990001.a  ca1202a2.ada  ca11d02.a  ca2009d.ada
ca1202a3.ada  ca11d03.a  ca2009f0.ada
ca1202a4.ada  ca120010.a  ca2009f1.ada
c9a004a.ada  ca1202a5.ada  ca120011.a  ca2009f2.ada
ca9a007a.ada  ca1202a6.ada  ca120012.am  ca2011b.ada
ca9a009a.ada  ca11001.a  ca12002.a  ca21001.a
ca9a009c.ada  ca11002.a  ca13001.a  ca3011a0.ada
c9a009f.ada  ca11003.a  ca13002.a  ca3011a1.ada
c9a009g.ada  ca110040.a  ca13003.a  ca3011a2.ada
c9a009h.ada  ca110041.a  ca13a01.a  ca3011a3.ada
c9a010a.ada  ca110042.am  ca13a02.a  ca3011a4.ada
c9a011a.ada  ca110050.a  ca13a02.a  ca3011a4.ada
<p>| ca5003a0.ada | cb4013a.ada | cc3007a.ada | cc3305c.ada |
| ca5003a1.ada | cb40a01.a | cc3007b.ada | cc3305d.ada |
| ca5003a2.ada | cb40a020.a | cc3011a.ada | cc3601a.ada |
| ca5003a3.ada | cb40a021.am | cc3011d.ada | cc3601c.ada |
| ca5003a4.ada | cb40a030.a | cc3012a.ada | cc3602a.ada |
| ca5003a5.ada | cb40a031.am | cc3015a.ada | cc3603a.ada |
| ca5003a6.ada | cb40a04.a | cc3016b.ada | cc3605a.ada |
| ca5003b0.ada | cb41001.a | cc3016c.ada | cc3606a.ada |
| ca5003b1.ada | cb41002.a | cc3016f.ada | cc3606b.ada |
| ca5003b2.ada | cb41003.a | cc3016i.ada | cc3607b.ada |
| ca5003b3.ada | cb41004.a | cc3017b.ada | cc40001.a |
| ca5003b4.ada | cb5001a.ada | cc3017c.ada | cc50001.a |
| ca5003b5.ada | cb5001b.ada | cc3019a.ada | cc50a01.a |
| ca5004a.ada | cb5002a.ada | cc3019b0.ada | cc50a02.a |
| ca5004b0.ada | cc1004a.ada | cc3019b1.ada | cc51001.a |
| ca5004b1.ada | cc1005b.ada | cc3019b2.ada | cc51002.a |
| ca5004b2.ada | cc1010a.ada | cc3019c0.ada | cc51003.a |
| ca5006a.ada | cc1010b.ada | cc3019c1.ada | cc51004.a |
| cb10002.a | cc1018a.ada | cc3019c2.ada | cc51006.a |
| cb1001a.ada | cc1104c.ada | cc3106b.ada | cc51007.a |
| cb1004a.ada | cc1107b.ada | cc3120a.ada | cc51008.a |
| cb105a.ada | cc1111a.ada | cc3120b.ada | cc51a01.a |
| cb1010a.ada | cc1204a.ada | cc3121a.ada | cc51b03.a |
| cb1010c.ada | cc1207b.ada | cc3123a.ada | cc51d01.a |
| cb1010d.ada | cc1220a.ada | cc3125a.ada | cc51d02.a |
| cb20001.a | cc1221a.ada | cc3125b.ada | cc54001.a |
| cb20003.a | cc1221b.ada | cc3125c.ada | cc54002.a |
| cb2004.a | cc1221c.ada | cc3125d.ada | cc54003.a |
| cb2005.a | cc1221d.ada | cc3126a.ada | cc54004.a |
| cb2006.a | cc1222a.ada | cc3127a.ada | cc70001.a |
| cb2007.a | cc1223a.ada | cc3128a.ada | cc70002.a |
| cb2004a.ada | cc1224a.ada | cc3203a.ada | cc70003.a |
| cb2005a.ada | cc1225a.tst | cc3207b.ada | cc70a01.a |
| cb2006a.ada | cc1226a.ada | cc3220a.ada | cc70a02.a |
| cb2007a.ada | cc1227a.ada | cc3221a.ada | cc70b01.a |
| cb20a02.a | cc1301a.ada | cc3222a.ada | cc70b02.a |
| cb3003a.ada | cc1302a.ada | cc3223a.ada | cc70c01.a |
| cb3003b.ada | cc1304a.ada | cc3224a.ada | cc70c02.a |
| cb3004a.ada | cc1304b.ada | cc3225a.ada | cd10001.a |
| cb40005.a | cc1307a.ada | cc3230a.ada | cd10002.a |
| cb4001a.ada | cc1307b.ada | cc3231a.ada | cd1009a.ada |
| cb4002a.ada | cc1308a.ada | cc3232a.ada | cd1009b.ada |
| cb4003a.ada | cc1310a.ada | cc3233a.ada | cd1009d.ada |
| cb4004a.ada | cc1311a.ada | cc3234a.ada | cd1009e.ada |
| cb4005a.ada | cc1311b.ada | cc3235a.ada | cd1009f.ada |
| cb4006a.ada | cc2002a.ada | cc3236a.ada | cd1009g.ada |
| cb4007a.ada | cc30001.a | cc3240a.ada | cd1009h.ada |
| cb4008a.ada | cc30002.a | cc3305a.ada | cd1009i.ada |
| cb4009a.ada | cc3004a.ada | cc3305b.ada | cd1009j.ada |</p>
<table>
<thead>
<tr>
<th>File1</th>
<th>File2</th>
<th>File3</th>
<th>File4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ce2103d.ada</td>
<td>ce2401c.ada</td>
<td>ce3114a.ada</td>
<td>ce3410e.ada</td>
</tr>
<tr>
<td>ce2104a.ada</td>
<td>ce2401f.ada</td>
<td>ce3115a.ada</td>
<td>ce3411a.ada</td>
</tr>
<tr>
<td>ce2104b.ada</td>
<td>ce2401h.ada</td>
<td>ce3201a.ada</td>
<td>ce3411c.ada</td>
</tr>
<tr>
<td>ce2104c.ada</td>
<td>ce2401i.ada</td>
<td>ce3202a.ada</td>
<td>ce3412a.ada</td>
</tr>
<tr>
<td>ce2104d.ada</td>
<td>ce2401j.ada</td>
<td>ce3206a.ada</td>
<td>ce3413a.ada</td>
</tr>
<tr>
<td>ce2106a.ada</td>
<td>ce2401k.ada</td>
<td>ce3207a.ada</td>
<td>ce3413b.ada</td>
</tr>
<tr>
<td>ce2106b.ada</td>
<td>ce2401l.ada</td>
<td>ce3301a.ada</td>
<td>ce3413c.ada</td>
</tr>
<tr>
<td>ce2108e.ada</td>
<td>ce2402a.ada</td>
<td>ce3302a.ada</td>
<td>ce3414a.ada</td>
</tr>
<tr>
<td>ce2108f.ada</td>
<td>ce2403a.tst</td>
<td>ce3303a.ada</td>
<td>ce3601a.ada</td>
</tr>
<tr>
<td>ce2108g.ada</td>
<td>ce2404a.ada</td>
<td>ce3304a.tst</td>
<td>ce3602a.ada</td>
</tr>
<tr>
<td>ce2108h.ada</td>
<td>ce2404b.ada</td>
<td>ce3305a.ada</td>
<td>ce3602b.ada</td>
</tr>
<tr>
<td>ce2109a.ada</td>
<td>ce2405b.ada</td>
<td>ce3306a.ada</td>
<td>ce3602c.ada</td>
</tr>
<tr>
<td>ce2109b.ada</td>
<td>ce2406a.ada</td>
<td>ce3401a.ada</td>
<td>ce3602d.ada</td>
</tr>
<tr>
<td>ce2109c.ada</td>
<td>ce2407a.ada</td>
<td>ce3402a.ada</td>
<td>ce3603a.ada</td>
</tr>
<tr>
<td>ce2110a.ada</td>
<td>ce2407b.ada</td>
<td>ce3402c.ada</td>
<td>ce3604a.ada</td>
</tr>
<tr>
<td>ce2110c.ada</td>
<td>ce2408a.ada</td>
<td>ce3402d.ada</td>
<td>ce3604b.ada</td>
</tr>
<tr>
<td>ce2111a.ada</td>
<td>ce2408b.ada</td>
<td>ce3402e.ada</td>
<td>ce3605a.ada</td>
</tr>
<tr>
<td>ce2111b.ada</td>
<td>ce2409a.ada</td>
<td>ce3403a.ada</td>
<td>ce3605b.ada</td>
</tr>
<tr>
<td>ce2111c.ada</td>
<td>ce2409b.ada</td>
<td>ce3403b.ada</td>
<td>ce3605c.ada</td>
</tr>
<tr>
<td>ce2111e.ada</td>
<td>ce2410a.ada</td>
<td>ce3403c.ada</td>
<td>ce3605d.ada</td>
</tr>
<tr>
<td>ce2111f.ada</td>
<td>ce2410b.ada</td>
<td>ce3403d.ada</td>
<td>ce3605e.ada</td>
</tr>
<tr>
<td>ce2111g.ada</td>
<td>ce2411a.ada</td>
<td>ce3403e.ada</td>
<td>ce3606a.ada</td>
</tr>
<tr>
<td>ce2111i.ada</td>
<td>ce3002b.tst</td>
<td>ce3403f.ada</td>
<td>ce3606b.ada</td>
</tr>
<tr>
<td>ce2201a.ada</td>
<td>ce3002c.tst</td>
<td>ce3404a.ada</td>
<td>ce3701a.ada</td>
</tr>
<tr>
<td>ce2201b.ada</td>
<td>ce3002d.ada</td>
<td>ce3404b.ada</td>
<td>ce3704a.ada</td>
</tr>
<tr>
<td>ce2201c.ada</td>
<td>ce3002f.ada</td>
<td>ce3404c.ada</td>
<td>ce3704b.ada</td>
</tr>
<tr>
<td>ce2201d.dep</td>
<td>ce3102a.ada</td>
<td>ce3404d.ada</td>
<td>ce3704c.ada</td>
</tr>
<tr>
<td>ce2201e.dep</td>
<td>ce3102b.tst</td>
<td>ce3405a.ada</td>
<td>ce3704d.ada</td>
</tr>
<tr>
<td>ce2201f.ada</td>
<td>ce3102d.ada</td>
<td>ce3405b.ada</td>
<td>ce3704e.ada</td>
</tr>
<tr>
<td>ce2201g.ada</td>
<td>ce3102e.ada</td>
<td>ce3405c.ada</td>
<td>ce3704f.ada</td>
</tr>
<tr>
<td>ce2201h.ada</td>
<td>ce3102f.ada</td>
<td>ce3406a.ada</td>
<td>ce3704m.ada</td>
</tr>
<tr>
<td>ce2201i.ada</td>
<td>ce3102g.ada</td>
<td>ce3406b.ada</td>
<td>ce3704n.ada</td>
</tr>
<tr>
<td>ce2201j.ada</td>
<td>ce3102h.ada</td>
<td>ce3406c.ada</td>
<td>ce3704o.ada</td>
</tr>
<tr>
<td>ce2201k.ada</td>
<td>ce3102i.ada</td>
<td>ce3406d.ada</td>
<td>ce3705a.ada</td>
</tr>
<tr>
<td>ce2201l.ada</td>
<td>ce3102j.ada</td>
<td>ce3407a.ada</td>
<td>ce3705b.ada</td>
</tr>
<tr>
<td>ce2201m.ada</td>
<td>ce3102k.ada</td>
<td>ce3407b.ada</td>
<td>ce3705c.ada</td>
</tr>
<tr>
<td>ce2201n.ada</td>
<td>ce3103a.ada</td>
<td>ce3407c.ada</td>
<td>ce3705d.ada</td>
</tr>
<tr>
<td>ce2202a.ada</td>
<td>ce3104a.ada</td>
<td>ce3408a.ada</td>
<td>ce3705e.ada</td>
</tr>
<tr>
<td>ce2202a.ada</td>
<td>ce3104b.ada</td>
<td>ce3408b.ada</td>
<td>ce3706c.ada</td>
</tr>
<tr>
<td>ce2203a.tst</td>
<td>ce3104c.ada</td>
<td>ce3408c.ada</td>
<td>ce3706d.ada</td>
</tr>
<tr>
<td>ce2204a.ada</td>
<td>ce3106a.ada</td>
<td>ce3409a.ada</td>
<td>ce3706f.ada</td>
</tr>
<tr>
<td>ce2204b.ada</td>
<td>ce3106b.ada</td>
<td>ce3409b.ada</td>
<td>ce3706g.ada</td>
</tr>
<tr>
<td>ce2204c.ada</td>
<td>ce3107a.tst</td>
<td>ce3409c.ada</td>
<td>ce3707a.ada</td>
</tr>
<tr>
<td>ce2204d.ada</td>
<td>ce3107b.ada</td>
<td>ce3409d.ada</td>
<td>ce3708a.ada</td>
</tr>
<tr>
<td>ce2205a.ada</td>
<td>ce3108a.ada</td>
<td>ce3409e.ada</td>
<td>ce3801a.ada</td>
</tr>
<tr>
<td>ce2206a.ada</td>
<td>ce3108b.ada</td>
<td>ce3410a.ada</td>
<td>ce3801b.ada</td>
</tr>
<tr>
<td>ce2208a.ada</td>
<td>ce3110a.ada</td>
<td>ce3410b.ada</td>
<td>ce3804a.ada</td>
</tr>
<tr>
<td>ce2401a.ada</td>
<td>ce3112c.ada</td>
<td>ce3410c.ada</td>
<td>ce3804b.ada</td>
</tr>
<tr>
<td>ce2401c.ada</td>
<td>ce3112d.ada</td>
<td>ce3410d.ada</td>
<td>ce3804c.ada</td>
</tr>
</tbody>
</table>
ce3804d.ada
cxa4005.a
cxaa001.a
cxb30130.c
ce3804e.ada
cxa4006.a
cxaa002.a
cxb30131.c
ce3804f.ada
cxa4007.a
cxaa003.a
cxb30132.am
ce3804g.ada
cxa4008.a
cxaa004.a
cxb3014.a
ce3804h.ada
cxa4009.a
cxaa005.a
cxb3015.a
ce3804i.ada
cxa4010.a
cxaa006.a
cxb3016.a
ce3804j.ada
cxa4011.a
cxaa007.a
cxb4001.a
ce3804m.ada
cxa4012.a
cxaa008.a
cxb4002.a
ce3804o.ada
cxa4013.a
cxaa009.a
cxb4003.a
ce3804p.ada
cxa4014.a
cxaa010.a
cxb4004.a
ce3805a.ada
cxa4015.a
cxaa011.a
cxb4005.a
ce3805b.ada
cxa4016.a
cxaa012.a
cxb4006.a
ce3806a.ada
cxa4017.a
cxaa013.a
cxb4007.a
ce3806b.ada
cxa4018.a
cxaa014.a
cxb4008.a
ce3806c.ada
cxa4019.a
cxaa015.a
cxb40090.cbl
ce3806d.ada
cxa4020.a
cxaa016.a
cxb40091.cbl
ce3806e.ada
cxa4021.a
cxaa017.a
cxb40092.cbl
ce3806f.ada
cxa4022.a
cxaa018.a
cxb40093.am
ce3806g.ada
cxa4023.a
cxaa019.a
cxb5001.a
ce3806h.ada
cxa4024.a
cxb001.a
cxb5002.a
ce3809a.ada
cxa4025.a
cxac001.a
cxb5003.a
ce3809b.ada
cxa4026.a
cxac002.a
cxb50040.ftn
ce3810a.ada
cxa4027.a
cxac003.a
cxb50041.ftn
ce3810b.ada
cxa4028.a
cxac004.a
cxb50042.am
ce3815a.ada
cxa4029.a
cxac005.a
cxb50050.ftn
ce3901a.ada
cxa4030.a
cxac006.a
cxb50051.ftn
ce3902b.ada
cxa4031.a
cxca01.a
cxb50052.am
ce3904a.ada
cxa4032.a
cxca02.a
d4a002.a.ada
ce3904b.ada
cxa4033.a
cxcb01.a
d4a002b.ada
ce3905a.ada
cxa4034.a
cxcb02.a
d4a004.a.ada
ce3905b.ada
cxa5011.a
cxacc01.a
d4a004b.ada
ce3905c.ada
cxa5012.a
cxf001.a
e28002b.ada
ce3905l.ada
cxa5013.a
cxb2001.a
e28005d.ada
ce3906a.ada
cxa5015.a
cxb2002.a
e52103y.ada
ce3906b.ada
cxa5a01.a
cxb2003.a
eb0011a.ada
ce3906c.ada
cxa502.a
cxb3001.a
eb0012a.ada
ce3906d.ada
cxa503.a
cxb3002.a
eb0014a.ada
ce3906e.ada
cxa504.a
cxb3003.a
ee3203a.ada
ce3906f.ada
cxa505.a
cxb30040.c
ee3204a.ada
ce3907a.ada
cxa506.a
cxb30041.am
ee3402b.ada
ce3908a.ada
cxa507.a
cxb3005.a
ee3409f.ada
ce3a001.a
cxa5a07.a
cxb30060.c
ee3412c.ada
cxa3002.a
cxa509.a
cxb30061.am
la140010.a
cxa3003.a
cxa5a10.a

cxb3007.a
la140011.am
cxa3004.a
cxa8001.a
cxb3008.a
la140012.a
cxa4001.a
cxa8002.a

cxb3009.a
la140020.a
cxa4002.a
cxa8003.a

cxb3010.a
la140021.am
cxa4003.a
cxa9001.a

cxb3011.a
la140022.a
cxa4004.a
cxa9002.a

cxb3012.a
la140030.a
A.2 Specialized Needs Annex Test Files

This section lists the files containing Specialized Needs Annex tests; that is, tests for requirements specified in Annex C, Annex D, Annex E, Annex G, or Annex H.

bx3001.a  bxc6a01.a  bxc2007.a  bxc2013.a
bx3002.a  bxc6a02.a  bxc2008.a  bxc2a01.a
bx5001.a  bxc6a03.a  bxc2009.a  bxc2a02.a
bx6001.a  bxc6a04.a  bxc2010.a  bxc2a03.a
bx6002.a  bxd1001.a  bxc2011.a  bxc2a04.a
bx6003.a  bxd1002.a  bxc2012.a  bxc2a05.a
bxc2a06.a  cxd3002.a  cxf3002.a  lxd70031.a
bxe4001.a  cxd3003.a  cxf3003.a  lxd70032.am
bxf1001.a  cxd4001.a  cxf3004.a  lxd70040.a
bxh4001.a  cxd4002.a  cxf3a01.a  lxd70041.a
bxh4002.a  cxd4003.a  cxf3a02.a  lxd70042.am
bxh4003.a  cxd4004.a  cxf3a03.a  lxd70050.a
bxh4004.a  cxd4005.a  cxf3a04.a  lxd70051.a
bxh4005.a  cxd4006.a  cxf3a05.a  lxd70052.am
bxh4006.a  cxd4007.a  cxf3a06.a  lxd70060.a
bxh4007.a  cxd4008.a  cxf3a07.a  lxd70061.a
bxh4008.a  cxd4009.a  cxf3a08.a  lxd70062.am
bxh4009.a  cxd4010.a  cxf3a09.a  lxd70070.a
bxh4010.a  cxd5001.a  cxf3a10.a  lxd70071.a
bxh4011.a  cxd6001.a  cxf3a11.a  lxd70072.am
bxh4012.a  cxd6002.a  cxf3a12.a  lxd70080.a
bxh4013.a  cxd6003.a  cxf3a13.a  lxd70081.a
cxc3001.a  cxd8001.a  cxf3a14.a  lxd70082.am
cxe3002.a  cxd8002.a  cxf3a15.a  lxd70090.a
cxc3003.a  cxd8003.a  cxf3a16.a  lxd70091.a
cxc3004.a  cxd9001.a  cxf3a17.a  lxd70092.am
cxc3005.a  cxda001.a  cxf3a18.a  lxc30010.am
cxc3006.a  cxda002.a  cxf3a19.a  lxc30011.am
cxc3007.a  cxda003.a  cxf3a20.a  lxc30020.am
cxc3008.a  cxda004.a  cxf3a21.a  lxc30021.am
cxc3009.a  cxd0002.a  cxf3a22.a  lxh40010.a
ncxc6001.a  cxd0002.a  cxf3a23.a  lxh40011.a
cxe6002.a  cxdb003.a  cxf3a24.a  lxh40012.am
ncxc6003.a  cxdb004.a  cxf3a25.a  lxh40020.a
ncxc7001.a  cxe1001.a  cxf3a26.a  lxh40021.a
ncxc7002.a  cxe2001.a  cxf3a27.a  lxh40022.am
ncxc7003.a  cxe2002.a  cxf3a28.a  lxh40030.a
ncxc7004.a  cxe4001.a  cxf3a29.a  lxh40031.a
ncxc7005.a  cxe4002.a  cxf3a30.a  lxh40032.a
ncxd1001.a  cxe4003.a  cxf3a31.a  lxh40033.am
ncxd1002.a  cxe4004.a  cxf3a32.a  lxh40040.a
ncxd1003.a  cxe4005.a  cxf3a33.a  lxh40041.a
ncxd1004.a  cxe4006.a  cxf3a34.a  lxh40042.a
ncxd1005.a  cxe5001.a  cxf3a35.a  lxh40043.am
ncxd1006.a  cxe5002.a  cxf3a36.a  lxh40050.a
ncxd1007.a  cxe5003.a  cxf3a37.a  lxh40051.a
ncxd1008.a  cxe1001.a  cxf3a38.a  lxh40052.a
ncxd2001.a  cxf2001.a  cxf3a39.a  lxh40053.am
ncxd2002.a  cxf2002.a  cxf3a40.a  lxh40060.a
ncxd2003.a  cxf2003.a  cxf3a41.a  lxh40061.a
ncxd2004.a  cxf2004.a  cxf3a42.a  lxh40062.a
ncxd2006.a  cxf2005.a  lxd70010.a  lxh40063.am
ncxd2007.a  cxf2a01.a  lxd70011.a  lxh40070.a
ncxd2008.a  cxf2a02.a  lxd70012.am  lxh40071.a
ncxd3001.a  cxf3001.a  lxd70030.a  lxh40072.a
A.3 Foundation Code Files

This section lists the foundation files. These files contain packages that may be used by more than one test for related objectives.

<table>
<thead>
<tr>
<th>File 1</th>
<th>File 2</th>
<th>File 3</th>
<th>File 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>f340a000.a</td>
<td>f460a00.a</td>
<td>fa13b00.a</td>
<td>fd72a00.a</td>
</tr>
<tr>
<td>f340a001.a</td>
<td>f730a000.a</td>
<td>fa21a00.a</td>
<td>fdb0a00.a</td>
</tr>
<tr>
<td>f341a00.a</td>
<td>f730a001.a</td>
<td>fb20a00.a</td>
<td>fdd2a00.a</td>
</tr>
<tr>
<td>f390a00.a</td>
<td>f731a00.a</td>
<td>fb40a00.a</td>
<td>fxa5a00.a</td>
</tr>
<tr>
<td>f392a00.a</td>
<td>f750a00.a</td>
<td>fc50a00.a</td>
<td>fxaca00.a</td>
</tr>
<tr>
<td>f392c00.a</td>
<td>f750b00.a</td>
<td>fc51a00.a</td>
<td>fxacb00.a</td>
</tr>
<tr>
<td>f392d00.a</td>
<td>f940a00.a</td>
<td>fc51b00.a</td>
<td>fxacc00.a</td>
</tr>
<tr>
<td>f393a00.a</td>
<td>f954a00.a</td>
<td>fe51c00.a</td>
<td>fxc6a00.a</td>
</tr>
<tr>
<td>f393b00.a</td>
<td>fa11a00.a</td>
<td>fe51d00.a</td>
<td>fxe2a00.a</td>
</tr>
<tr>
<td>f394a00.a</td>
<td>fa11b00.a</td>
<td>fc54a00.a</td>
<td>fx2a00.a</td>
</tr>
<tr>
<td>f3a2a00.a</td>
<td>fa11c00.a</td>
<td>fc70a00.a</td>
<td>fxf3a00.a</td>
</tr>
<tr>
<td>f431a00.a</td>
<td>fa11d00.a</td>
<td>fc70b00.a</td>
<td>fxf3a00.a</td>
</tr>
<tr>
<td>f433a00.a</td>
<td>fa13a00.a</td>
<td>fc70c00.a</td>
<td>fxf3a00.a</td>
</tr>
</tbody>
</table>

A.4 Documentation Files

This section lists all the files containing ACATS 3.0 documentation. Files with the .pdf extension are in Adobe Acrobat format. Files with the .htm extension are in HTML; files with the extensions .gif and .png are associated graphics files. All other files are in ASCII text format.

<table>
<thead>
<tr>
<th>File 1</th>
<th>File 2</th>
<th>File 3</th>
<th>File 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>acats-ug.pdf</td>
<td>mod-name.png</td>
<td>ug-4.htm</td>
<td>ug-428.htm</td>
</tr>
<tr>
<td>cont.gif</td>
<td>next.gif</td>
<td>ug-41.htm</td>
<td>ug-43.htm</td>
</tr>
<tr>
<td>cover-03.pdf</td>
<td>prev.gif</td>
<td>ug-411.htm</td>
<td>ug-431.htm</td>
</tr>
<tr>
<td>cover-04.pdf</td>
<td>testobj.txt</td>
<td>ug-412.htm</td>
<td>ug-432.htm</td>
</tr>
<tr>
<td>cover-06.pdf</td>
<td>ug-01.htm</td>
<td>ug-413.htm</td>
<td>ug-433.htm</td>
</tr>
<tr>
<td>cover-07.pdf</td>
<td>ug-02.htm</td>
<td>ug-414.htm</td>
<td>ug-44.htm</td>
</tr>
<tr>
<td>cover-08.pdf</td>
<td>ug-1.htm</td>
<td>ug-415.htm</td>
<td>ug-45.htm</td>
</tr>
<tr>
<td>cover-10.pdf</td>
<td>ug-11.htm</td>
<td>ug-416.htm</td>
<td>ug-46.htm</td>
</tr>
<tr>
<td>coverkey.pdf</td>
<td>ug-12.htm</td>
<td>ug-42.htm</td>
<td>ug-47.htm</td>
</tr>
<tr>
<td>coverage.txt</td>
<td>ug-3.htm</td>
<td>ug-422.htm</td>
<td>ug-5.htm</td>
</tr>
<tr>
<td>dirs.png</td>
<td>ug-31.htm</td>
<td>ug-423.htm</td>
<td>ug-51.htm</td>
</tr>
<tr>
<td>find.gif</td>
<td>ug-32.htm</td>
<td>ug-424.htm</td>
<td>ug-511.htm</td>
</tr>
<tr>
<td>index.gif</td>
<td>ug-321.htm</td>
<td>ug-425.htm</td>
<td>ug-512.htm</td>
</tr>
<tr>
<td>leg-name.png</td>
<td>ug-322.htm</td>
<td>ug-426.htm</td>
<td>ug-5121.htm</td>
</tr>
<tr>
<td>lib.gif</td>
<td>ug-323.htm</td>
<td>ug-427.htm</td>
<td>ug-5122.htm</td>
</tr>
</tbody>
</table>
A.5 Other Files

This section lists the files falling into the others category in the table found in Section 4.1. They are listed in the categories described there.

A.5.1 List of ACATS 3.0 Files

acats30.lst

A.5.2 Support Units Referenced by Many Tests

checkfil.ada  impdefc.a  impdefh.a  spprt13s.tst
enumchek.ada  impdefd.a  lencheck.ada  tctouch.ada
fcndecl.ada  impdefe.a  repspec.ada
impdef.a  impdefg.a  repbody.ada

A.5.3 Preprocessing Tools and Data

macrosub.ada  macro.dfs  tsttests.dat
A.5.4 Tests for Reporting Code

cz00004.a  cz1101a.ada  cz1102a.ada  cz1103a.ada

A.6 Tests With Special Requirements

The tests listed in this section have special processing requirements that are described in the internal test commentary.

ba15001  cxd2004  cxdg2007  la14015
bxc5001  cxd2006  cxdg2008  la14016
bxa4001  cxd2007  cxdg2009  la14017
bxa4002  cxd2008  cxdg2010  la14018
bxa4003  cxd3001  cxdg2011  la14019
bxa4004  cxd3002  cxdg2012  la14020
bxa4005  cxd3003  cxdg2013  la14021
bxa4006  cxd4001  cxdg2014  la14022
bxa4007  cxd4003  cxdg2015  la14024
bxa4008  cxd4004  cxdg2016  la14025
bxa4009  cxd4005  cxdg2017  la14026
bxa4010  cxd4006  cxdg2018  la14027
bxa4011  cxd4007  cxdg2019  la20001
bxa4012  cxd4008  cxdg2020  lc30001
bxa4013  cxd4009  cxdg2021  lc30002
cal1023  cxd4010  cxdg2022  lc30003
cal12001  cxda003  cxdg2023  lxd7009
cd30005  cxda004  cxdg2024  lxe3001
cxb3004  cxe1001  cxdh1001  lxe3002
cxb3006  cxe2001  cxdh3001  lxe4001
cxb3008  cxe2002  cxdh3003  lxe4002
cxb3013  cxe4001  la14001  lxe4003
cxb4009  cxe4002  la14002  lxe4004
cxb5004  cxe4003  la14003  lxe4005
cxb5005  cxe4004  la14004  lxe4006
cxe3001  cxe4005  la14005  lxe4007
cxe3003  cxe4006  la14006  lxe4008
cxe3004  cxe5002  la14007  lxe4009
cxe3006  cxe5003  la14008  lxe4010
cxe3008  cxxg1002  la14009  lxe4011
cxd1004  cxxg1005  la14010  lxe4012
cxd1005  cxxg2002  la14011  lxe4013
cxd2001  cxxg2003  la14012  lxe4014
cxd2002  cxxg2004  la14013
cxd2003  cxxg2006  la14014

A.7 Test Files Added Since ACATS 2.6

The following test files are new in ACATS 3.0:
The following documentation files are new in ACATS 3.0:

acats-ug.pdf  cover-04.pdf  cover-08.pdf  coversum.pdf
cont.gif      cover-06.pdf  cover-10.pdf  dirs.png
cover-03.pdf  cover-07.pdf  coverkey.pdf  find.gif

A.8 Documentation Files Added Since ACATS 2.6

The following documentation files are new in ACATS 3.0:
A.9 Test Files Modified Since ACATS 2.6

The following test files have been modified from their ACATS 2.6 versions:

<table>
<thead>
<tr>
<th>Original Name</th>
<th>Modified Name 1</th>
<th>Modified Name 2</th>
<th>Modified Name 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>b34005q.ada</td>
<td>b370001.a</td>
<td>b3710014.am</td>
<td>b392009.a</td>
</tr>
<tr>
<td>b34005t.ada</td>
<td>b3710010.a</td>
<td>b38105a.ada</td>
<td>b392010.a</td>
</tr>
<tr>
<td>b34006h.ada</td>
<td>b3710011.a</td>
<td>b391001.a</td>
<td>b393005.a</td>
</tr>
<tr>
<td>b34006k.ada</td>
<td>b3710012.a</td>
<td>b391002.a</td>
<td>b393006.a</td>
</tr>
<tr>
<td>b34008b.ada</td>
<td>b3710013.a</td>
<td>b392001.a</td>
<td>b393007.a</td>
</tr>
</tbody>
</table>
A.10 Support Files Modified Since ACATS 2.6

The following support files have been modified from their ACATS 2.6 versions:

- repbody.ada
- tctouch.ada

A.11 Test Files Deleted Since ACATS 2.6

The following test files were present in ACATS 2.6 but do not appear in ACATS 3.0:

- b32101a.ada
- b360001.a
- b37004b.ada
- b43001m.ada
- b48003c.ada
- b58002a.ada
- b58002b.ada
- b58002c.ada
- b64201a.ada
- b97206a.ada

A.12 Other Files Deleted Since ACATS 2.6

The following support and documentation files were present in ACATS 2.6 but do not appear in ACATS 3.0:

- ug-apxa.doc
- ug-apxa.pdf
- ug-apxa.txt
- ug-apxb.doc
- ug-apxb.pdf
- ug-apxb.txt
- ug-apxc.doc
- ug-apxc.pdf
- ug-apxc.txt
- ug-body.doc
- ug-body.pdf
- ug-body.txt
- widechr.a
Annex B: Parameterization Files

In ACATS 3.0, two methods are used to account for the use of implementation-dependent values in the tests.

For legacy tests, a "macro" substitution technique is used. Legacy tests requiring implementation-specific values contain symbols beginning with the '$' character; for example, the symbol $INTEGER_LAST is used where the code expects the implementation-specific integer literal representing the largest integer. For each implementation, these symbols must be systematically replaced with the appropriate values. A data file, MACRO.DFS, and an Ada program, Macrosub, are provided to facilitate this substitution.

For tests written since the standardization of [Ada95], a hierarchy of packages is provided that contain constants and functions that provide the desired implementation-specific values. These packages ("ImpDef" and its children) should be modified for each implementation to provide the needed values.

Information regarding the macro substitution technique is presented in Sections B.1 and B.2. Section B.3 describes the ImpDef package hierarchy.

B.1 Macro Substitution File

The support file MACRO.DFS provides substitutions for special symbols that appear in certain ACATS 3.0 tests (indicated by the three-letter file type (extension) "TST" and listed in Section B.2). The support program Macrosub may be used to insert these implementation-specific values in place of the special symbols in the test. The following excerpt from the file describes the file and its use.

```plaintext
-- MACRO.DFS
-- THIS FILE CONTAINS THE MACRO DEFINITIONS USED IN THE ACVC TESTS.
-- THESE DEFINITIONS ARE USED BY THE ACVC TEST PRE-PROCESSOR,
-- MACROSUB. MACROSUB WILL CALCULATE VALUES FOR THOSE MACRO SYMBOLS
-- WHOSE DEFINITIONS DEPEND ON THE VALUE OF MAX_IN_LEN (NAMELY, THE
-- VALUES OF THE MACRO SYMBOLS BIG_ID1, BIG_ID2, BIG_ID3, BIG_ID4,
-- BIG_STRING1, BIG_STRING2, MAX_STRING_LITERAL, BIG_INT_LIT, BIG_REAL_LIT,
-- AND BLANKS). THEREFORE, ANY VALUES GIVEN IN THIS FILE FOR THOSE
-- MACRO SYMBOLS WILL BE IGNORED BY MACROSUB.
-- NOTE: AS REQUIRED BY THE MACROSUB PROGRAM, THE FIRST MACRO DEFINED
-- IN THIS FILE IS $MAX_IN_LEN. THE NEXT 5 MACRO DEFINITIONS
-- ARE FOR THOSE MACRO SYMBOLS THAT DEPEND ON THE VALUE OF
-- MAX_IN_LEN. THESE ARE IN ALPHABETIC ORDER. FOLLOWING THESE
-- ARE 36 MORE DEFINITIONS, ALSO IN ALPHABETIC ORDER.
-- EACH DEFINITION IS ACCORDING TO THE FOLLOWING FORMAT:
-- A NUMBER OF LINES PRECEDED BY THE ADA COMMENT DELIMITER, --.
-- THE FIRST OF THESE LINES CONTAINS THE MACRO SYMBOL AS IT APPEARS
-- IN THE TEST FILES (WITH THE DOLLAR SIGN). THE NEXT FEW "COMMENT"
-- LINES CONTAIN A DESCRIPTION OF THE VALUE TO BE SUBSTITUTED.
-- THE REMAINING "COMMENT" LINES, THE FIRST OF WHICH BEGINS WITH THE
-- WORDS "USED IN: " (NO QUOTES), CONTAIN A LIST OF THE TEST FILES
-- (WITHOUT THE .TST EXTENSION) IN WHICH THE MACRO SYMBOL APPEARS.
-- EACH TEST FILE NAME IS PRECEDED BY ONE OR MORE BLANKS.
-- A LINE, WITHOUT THE COMMENT DELIMITER, CONSISTING OF THE
-- IDENTIFIER (WITHOUT THE DOLLAR SIGN) OF THE MACRO SYMBOL,
-- FOLLOWED BY A SPACE OR TAB, FOLLOWED BY THE VALUE TO BE
-- SUBSTITUTED. IN THE DISTRIBUTION FILE, A SAMPLE VALUE IS
-- PROVIDED; THIS VALUE MUST BE REPLACED BY A VALUE APPROPRIATE TO
-- THE IMPLEMENTATION.
-- DEFINITIONS ARE SEPARATED BY ONE OR MORE EMPTY LINES.
-- THE LIST OF DEFINITIONS BEGINS AFTER THE FOLLOWING EMPTY LINE.
```

89 3 January 2008 Parameterization Files
B.1 Macro Substitution File

-- $MAX_IN_LEN
-- AN INTEGER LITERAL GIVING THE MAXIMUM LENGTH PERMITTED BY THE
-- COMPILER FOR A LINE OF ADA SOURCE CODE (NOT INCLUDING AN END-OF-LINE
-- CHARACTER).
-- USED IN:  A26007A
MAX_IN_LEN                60

...

B.2 Macro Substitution Tests

The following test files contain the special symbols used for substituting implementation-specific values, as described in Section B.1. This list also appears in the ACATS 3.0 "support" directory as TSTTESTS.DAT.

A26007A.TST    BD2A02A.TST    C35502F.TST    CD2C11D.TST
AD8011A.TST    BD2C01D.TST    C35503D.TST    CD4041A.TST
B22001A.TST    BD2C02A.TST    C35503F.TST    CD7101G.TST
B22001B.TST    BD2C03A.TST    C45231D.TST    CE2102C.TST
B22001C.TST    BD4006A.TST    C4A007A.TST    CE2102H.TST
B22001D.TST    BD8001A.TST    C87B62D.TST    CE2103A.TST
B22001E.TST    BD8002A.TST    C96005B.TST    CE2103B.TST
B22001F.TST    BD8003A.TST    CC1225A.TST    CE2203A.TST
B22001G.TST    BD8004A.TST    CD1009K.TST    CE2403A.TST
B22001H.TST    BD8004B.TST    CD1009T.TST    CE3002B.TST
B22001I.TST    BD8004C.TST    CD1009U.TST    CE3002C.TST
B22001K.TST    C23003A.TST    CD1C03E.TST    CE3102B.TST
B22001L.TST    C23003B.TST    CD1C06A.TST    CE3107A.TST
B22001M.TST    C23003G.TST    CD2A83C.TST    CE3304A.TST
B22001N.TST    C23003J.TST    CD2A91C.TST    SPRT13S.TST
B34B01B.TST    C35502D.TST    CD2C11A.TST

B.3 Package ImpDef and Its Children

The package ImpDef (for "Implementation Definitions") provides constants and functions for producing implementation-specific values required by certain test programs. This package resides in the file ImpDef.a in the "support" directory. Four child packages are also included in the "support" directory, each providing the means for producing implementation-specific values required by certain test programs for a particular Specialized Needs Annex. These packages have names of the form ImpDef.Annex_X, and reside in files with names of the form ImpDefX.a, where 'X' is replaced by the letter designating the relevant Annex.

The ImpDef package and each of its children should be modified for each implementation as described in the source code. The following excerpt from the "ImpDef.a" file illustrates how these modification points are indicated in the packages.

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.

package ImpDef is

...
Validating_Annex_C : constant Boolean := False;
--                                               ^^^^^ --- MODIFY HERE AS NEEDED
Validating_Annex_D : constant Boolean := False;
--                                               ^^^^^ --- 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 := 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
...
Annex C: Results of CZ Tests

The "CZ" tests are executed before any other ACATS tests to ensure that the ImpDef packages have been properly customized and that certain support units are working properly. These tests are not considered as Passed or Failed. If they do not perform as expected, the problems must be identified and resolved before conformity testing can continue.

This Appendix presents sample results from executing the "CZ" tests. The actual output will differ, at least in the time-stamp information.

C.1 Sample Output From CZ0004

The following is the output from an execution of CZ0004 with a specific implementation. Note that it contains failure messages (indicated by '*') that are expected, as is the final FAILED report. Note also that certain report lines depend on the customization of the ImpDef package, and will vary with the implementation.

```
CZ00004 ACATS 3.0 07-12-14 14:46:13
-- CZ00004 Check that Impdef values have been supplied for the special needs annexes. Check that the routines in TCTouch work correctly.
- CZ00004 TCTouch ACATS 2.5.
* CZ00004 Assertion failed: Assertion Failed is expected.
* CZ00004 Assertion failed: Assertion Failed is expected.
* CZ00004 z should not equal Z Expecting: z Got: Z.
- CZ00004 Three failure messages should have occurred so far.
* CZ00004 Trace Overflow:
   xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
   xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
- CZ00004 A Trace Overflow message should have just occurred.
- CZ00004 <><><><><> ANNEX VALIDATION STATUS <><><><><>.
  + CZ00004 Annex C validation: Annex C not supported.
  + CZ00004 Annex D validation: Annex D not supported.
  + CZ00004 Annex E validation: Annex E not supported.
  + CZ00004 Annex F validation: Annex F not supported.
  + CZ00004 Annex G validation: Annex G not supported.
  + CZ00004 Annex H validation: Annex H not supported.
  + CZ00004 <><><><><> IMPDEF <><><><><>.
   - CZ00004 Validating_Annex_C : FALSE.
   - CZ00004 Validating_Annex_D : FALSE.
   - CZ00004 Validating_Annex_E : FALSE.
   - CZ00004 Validating_Annex_F : FALSE.
   - CZ00004 Validating_Annex_G : FALSE.
   - CZ00004 Validating_Annex_H : FALSE.
   - CZ00004 Minimum_Task_Switch: 0.100000000
   - CZ00004 Switch_To_New_Task: 1.000000000
   - CZ00004 Clear_Ready_Queue: 5.000000000
   - CZ00004 Delay_For_Time_Past: 0.100000000
   - CZ00004 Time_Dependent_Reset: 0.300000000
   - CZ00004 Delay_Per_Random_Test: 1.000000000
   - CZ00004 Exceed_Time_Slice.
  - CZ00004 Non_State_String: By No Means A State.
  - CZ00004 External_Tag_Value: implementation_defined.
  - CZ00004 CD30005_1_Foreign_Address: present.
  - CZ00004 CD30005_1_External_Name: CD30005_1.
  - CZ00004 Max_Default_Alignment: 1.
  - CZ00004 Max_Linker_Alignment: 1.
  - CZ00004 CXB30130_External_Name: CXB30130.
  - CZ00004 CXB30131_External_Name: CXB30131.
**** CZ00004 FAILED ***********************
```
C.2 Sample Output From CZ1101A

The following is the output of CZ1101A from a particular implementation. The output of this test should have the same messages (except for the time-stamp) and reported results as indicated here (including failure reports), and the format of the output lines should be as described in the output text.

- NO_NAME (CZ1101A) CHECK REPORT ROUTINES.
- NO_NAME INITIAL VALUES SHOULD BE 'NO_NAME' AND 'FAILED'.
**** NO_NAME FAILED *******************************************.

/// PASS_TEST ACATS 3.0 07-12-14 14:47:42
--- PASS_TEST CHECKING 'TEST' AND 'RESULT' FOR 'PASSED'.
  PASS_TEST THIS LINE IS EXACTLY 'MAX_LEN' LONG. ...5...60....5...70.
  PASS_TEST THIS COMMENT HAS A WORD THAT SPANS THE FOLD POINT. THIS
  COMMENT FITS EXACTLY ON TWO LINES. ...5...60....5...70.
  PASS_TEST THIS COMMENT IS ONE VERY LONG WORD AND SO IT SHOULD BE
  SPLIT AT THE FOLD POINT.
=== PASS_TEST PASSED =======================================.
- NO_NAME CHECK THAT 'RESULT' RESETS VALUES TO 'NO_NAME' AND
  'FAILED'.
**** NO_NAME FAILED *******************************************.

/// FAIL_TEST ACATS 3.0 07-12-14 14:47:42
--- FAIL_TEST CHECKING 'FAILED' AND 'RESULT' FOR 'FAILED'.
* FAIL_TEST 'RESULT' SHOULD NOW BE 'FAILED'.
**** FAIL_TEST FAILED *******************************************.

/// NA_TEST ACATS 3.0 07-12-14 14:47:42
--- NA_TEST CHECKING 'NOT-APPLICABLE'.
  NA_TEST 'RESULT' SHOULD NOW BE 'NOT-APPLICABLE'.
+++ NA_TEST NOT-APPLICABLE +++++++++++++++++++++++.

/// FAIL_NA_TEST ACATS 3.0 07-12-14 14:47:42
--- FAIL_NA_TEST CHECKING 'NOT_APPLICABLE', 'FAILED', 'NOT_APPLICABLE'.
  FAIL_NA_TEST 'RESULT' BECOMES 'NOT-APPLICABLE'.
  FAIL_NA_TEST 'RESULT' BECOMES 'FAILED'.
  FAIL_NA_TEST CALLING 'NOT_APPLICABLE' DOESN'T CHANGE 'RESULT'.
**** FAIL_NA_TEST FAILED *******************************************.

/// SPEC_NA_TEST ACATS 3.0 07-12-14 14:47:42
--- SPEC_NA_TEST CHECKING 'SPEC_ACT', 'NOT_APPLICABLE', 'SPEC_ACT'.
  SPEC_NA_TEST 'RESULT' BECOMES 'TENTATIVELY PASSED'.
  SPEC_NA_TEST 'RESULT' BECOMES 'NOT_APPLICABLE'.
  SPEC_NA_TEST CALLING 'SPECIAL_ACTION' DOESN'T CHANGE 'RESULT'.
+++ SPEC_NA_TEST NOT-APPLICABLE +++++++++++++++++++++++.

/// SPEC_FAIL_TEST ACATS 3.0 07-12-14 14:47:42
--- SPEC_FAIL_TEST CHECKING 'SPEC_ACT', 'FAILED', 'SPEC_ACT'.
  SPEC_FAIL_TEST 'RESULT' BECOMES 'TENTATIVELY PASSED'.
  SPEC_FAIL_TEST 'RESULT' BECOMES 'FAILED'.
  SPEC_FAIL_TEST CALLING 'SPECIAL_ACTION' DOESN'T CHANGE 'RESULT'.
**** SPEC_FAIL_TEST FAILED *******************************************.

/// CZ1101A ACATS 3.0 07-12-14 14:47:42
--- CZ1101A CHECKING 'SPECIAL_ACTION' ALONE.
  CZ1101A 'RESULT' BECOMES 'TENTATIVELY PASSED'.
!!! CZ1101A TENTATIVELY PASSED !!!!!!!!!!!!!!!!!!!!!.
!!! SEE '!' COMMENTS FOR SPECIAL NOTES!!
C.3 Sample Output From CZ1102A

Test CZ1102A should execute and report PASSED as illustrated below. Only the time-stamp should differ.

```
,.,. CZ1102A ACATS 3.0 07-12-14 14:47:57
---- CZ1102A CHECK THAT THE DYNAMIC VALUE ROUTINES OF THE REPORT PACKAGE WORK CORRECTLY.
==== CZ1102A PASSED ================.
```

C.4 Sample Output From CZ1103A

Test CZ1103A may produce two different forms of output, depending on whether the implementation supports external files.

C.4.1 Output When External Files Are Supported

If the implementation under test supports the creation and use of external text files, then test CZ1103A should produce the following report (except for differences in the time stamp). Note that failure messages are expected.

```
,.,. CZ1103A ACATS 3.0 07-12-14 14:48:10
---- CZ1103A CHECK THAT PROCEDURE CHECK_FILE WORKS.
- CZ1103A BEGIN TEST WITH AN EMPTY FILE.
- CZ1103A BEGIN TEST WITH A FILE WITH BLANK LINES.
- CZ1103A BEGIN TEST WITH A FILE WITH BLANK LINES AND PAGES.
- CZ1103A BEGIN TEST WITH A FILE WITH TRAILING BLANKS.
- CZ1103A BEGIN TEST WITH A FILE WITHOUT TRAILING BLANKS.
- CZ1103A BEGIN TEST WITH A FILE WITH AN END OF LINE ERROR.
  * CZ1103A FROM CHECK_FILE: END OF LINE EXPECTED - E ENCOUNTERED.
- CZ1103A BEGIN TEST WITH A FILE WITH AN END OF PAGE ERROR.
  * CZ1103A FROM CHECK_FILE: END_OF_PAGE NOT WHERE EXPECTED.
- CZ1103A BEGIN TEST WITH A FILE WITH INCORRECT DATA.
  * CZ1103A FROM CHECK_FILE: FILE DOES NOT CONTAIN CORRECT OUTPUT - EXPECTED C - GOT I.
- CZ1103A FROM CHECK_FILE: LAST CHARACTER IN FOLLOWING STRING REVEALED ERROR: LINE WITH C.
**** CZ1103A FAILED *********************.
,.,. CZ1103A ACATS 3.0 07-12-14 14:48:10
---- CZ1103A THE LINE ABOVE SHOULD REPORT FAILURE.
!!! CZ1103A COMPARE THIS OUTPUT TO THE EXPECTED RESULT.
!!!! CZ1103A TENTATIVELY PASSED !!!!!!!!!!!!!!!!.
!!!! SEE '!' COMMENTS FOR SPECIAL NOTES!!
```
C.4.2 Output When External Files Are Not Supported

If the implementation under test does not support external text files, then CZ1103A produces different output, as illustrated below.

```plaintext
,,,. CZ1103A ACATS 3.0 07-12-14 14:49:00
--- CZ1103A CHECK THAT PROCEDURE CHECK_FILE WORKS.
- CZ1103A BEGIN TEST WITH AN EMPTY FILE.
  * CZ1103A TEST WITH EMPTY FILE INCOMPLETE.
  - CZ1103A BEGIN TEST WITH A FILE WITH BLANK LINES.
    * CZ1103A TEST WITH FILE WITH BLANK LINES INCOMPLETE.
  - CZ1103A BEGIN TEST WITH A FILE WITH BLANK LINES AND PAGES.
    * CZ1103A TEST WITH FILE WITH BLANK LINES AND PAGES INCOMPLETE.
  - CZ1103A BEGIN TEST WITH A FILE WITH TRAILING BLANKS.
    * CZ1103A TEST WITH FILE WITH TRAILING BLANKS INCOMPLETE.
- CZ1103A BEGIN TEST WITH A FILE WITHOUT TRAILING BLANKS.
  * CZ1103A TEST WITH FILE WITHOUT TRAILING BLANKS INCOMPLETE.
- CZ1103A BEGIN TEST WITH A FILE WITH AN END OF LINE ERROR.
  * CZ1103A TEST WITH END OF LINE ERROR INCOMPLETE.
- CZ1103A BEGIN TEST WITH FILE WITH END OF PAGE ERROR.
  * CZ1103A TEST WITH END OF PAGE ERROR INCOMPLETE.
- CZ1103A BEGIN TEST WITH FILE WITH END OF FILE ERROR.
  * CZ1103A TEST WITH END OF FILE ERROR INCOMPLETE.
- CZ1103A BEGIN TEST WITH FILE WITH INCORRECT DATA.
  * CZ1103A TEST WITH INCORRECT DATA INCOMPLETE.
*** CZ1103A FAILED ******************************************************.
,,,. CZ1103A ACATS 3.0 07-12-14 14:49:00
--- CZ1103A THE LINE ABOVE SHOULD REPORT FAILURE.
! CZ1103A COMPARE THIS OUTPUT TO THE EXPECTED RESULT.
!!!! CZ1103A TENTATIVELY PASSED !!!!!!!!!!!!!!!!!!!.
!!!! SEE '!' COMMENTS FOR SPECIAL NOTES!!
```
Annex D: Test Applicability Criteria

Certain tests in the suite may be considered inapplicable to an implementation depending on the way the implementation treats the implementation-dependent features of the language. A brief summary of these implementation-dependent features and the tests they affect are listed in this appendix.

Note that the applicability of each one of these tests is based on the criteria listed in the test file. During validation, all the implementation-dependent tests are submitted for compilation and (if compiled successfully) are executed, with the following exceptions:

- Tests which require a floating point digits value that exceeds System.Max_Digits need not be submitted to the compiler. (The testing laboratory may require pre-validation evidence that the tests are properly rejected.)
- If file I/O is not supported, then the tests listed in Section D.2.13 will not be part of the customized test suite for bare target validations and will not be run during witness testing.

D.1 Compile-Time Inapplicability

The first part of this appendix is concerned with tests for which the applicability is determined at compile time. Class B tests that are inapplicable should be successfully compiled, or, in a few cases, should report an error on the line marked "--N/A => ERROR". Executable tests that are inapplicable based on their compile-time behavior must be rejected as a result of the unsupported feature. Lines containing the implementation-dependent features are marked "--N/A => ERROR". In every case, tests may be graded NOT-APPLICABLE only if all the following conditions are met:

- The implementation's treatment of the test is consistent with the applicability criteria given in the test comments;
- All other tests having the same applicability criteria exhibit the same behavior; and
- The behavior is consistent with the implementation's documentation.

D.1.1 Type Short_Integer

If there is no predefined type Short_Integer, then the tests contained in the following files are not applicable:

- B36105C.DEP
- B52004E.DEP
- B55B09D.DEP
- C45231B.DEP
- C45304B.DEP
- C45411B.DEP
- C45502B.DEP
- C45503B.DEP
- C45504B.DEP
- C45504E.DEP
- C45611B.DEP
- C45613B.DEP
- C45614B.DEP
- C45631B.DEP
- C45632B.DEP
- C55B07B.DEP
- CD7101E.DEP

D.1.2 Type Long_Integer

If there is no predefined type Long_Integer, then the tests contained in the following files are not applicable:

- B52004D.DEP
- B55B09C.DEP
- C45231C.DEP
- C45304B.DEP
- C45411C.DEP
- C45502C.DEP
- C45503C.DEP
- C45504C.DEP
- C45504F.DEP
- C45504G.DEP
- C45611C.DEP
- C45613C.DEP
- C45614C.DEP
- C45631C.DEP
- C45632C.DEP
- C55B07A.DEP
- CD7101F.DEP
D.1.3 Other Predefined Integer Types

If there are no predefined integer types with names other than Integer, Short_Integer, and Long_Integer, then the tests contained in the following files are not applicable:

C45231D.TST    CD7101G.TST

D.1.4 Fixed Point Restrictions

If System.Max_Mantissa is less than 47 or System.Fine_Delta is greater than 2.0**-47, then the tests contained in the following files are not applicable:

C45531M.DEP    C45531O.DEP
C45531N.DEP    C45531P.DEP
C45532M.DEP    C45532O.DEP
C45532N.DEP    C45532P.DEP

D.1.5 Non-binary Values of 'Small

If 'Small representation clauses which are not powers of two are not supported, then the tests contained in the following files are not applicable:

C45536A.DEP    CD2A53A.ADA

D.1.6 Compiler Rejection of Supposedly Static Expression

Consider the following declarations:

```ada
type F is digits System.Max_Digits;
N : constant := 2.0 * F'Machine_Radix ** F'Machine_EMax;
```

If the declaration of N is rejected on the grounds that evaluation of the expression will raise an exception, then the following test is not applicable:

C4A013B.ADA

D.1.7 Machine Code Insertions

If machine code insertions are not supported, then the tests contained in the following files are not applicable:

AD8011A.TST    BD8002A.TST    BD8004A.TST    BD8004C.TST
BD8001A.TST    BD8003A.TST    BD8004B.TST

D.1.8 Illegal External File Names

If there are no strings which are illegal as external file names, then the tests contained in the following files are not applicable:

CE2102C.TST    CE2102H.TST    CE3102B.TST    CE3107A.TST
D.1.9 Decimal Types
If decimal types are not supported, then the tests contained in the following files are not applicable: (Note that implementations testing Annex F must support decimal types.)
C460011.A CXAA010.A

D.1.10 Instantiation of Sequential_IO with indefinite types
If Sequential_IO does not support indefinite types, then the test contained in the following files are not applicable:
CE2201D.DEP CE2201E.DEP

D.1.11 Special Handling Tests
Test requiring special handling may also be not applicable. See section 5.5.5, “Tests with Special Processing Requirements”, for details.

D.2 Reported Inapplicability
This section is concerned with tests that can detect, at runtime, certain implementation characteristics that render the objective meaningless or prevent testing of the objective. These tests must compile and execute, reporting "NOT_APPLICABLE" as the result. This behavior must be consistent with other tests for related objective and with the implementation's documentation.

D.2.1 Value of Machine_Overflows is False
If Machine_Overflows is False for floating point types, then the tests contained in the following files should report NOT_APPLICABLE:
C45322A.ADA C45523A.ADA C4A012B.ADA

D.2.2 System.Max_Digits
If the value of System.Max_Digits is greater than 35, then the test contained in the following file should report NOT_APPLICABLE:
C4A011A.ADA

D.2.3 Floating Point Overflow
Consider the declaration
  
  type F is digits System.Max_Digits;

If F'Machine_Overflows = False and 2.0*F'Machine_Radix**F'Machine_EMax <= F'Base'Last then the test contained in the following file should report NOT_APPLICABLE (if it compiles without error):
C4A013B.ADA
D.2.4 Type Duration

If `Duration'First = Duration'Base'First` or `Duration'Last = Duration'Base'Last` then the tests contained in the following file should report NOT_APPLICABLE:

C96005B.TST

D.2.5 Text Files (Non-supported Features)

If `Use_Error` or `Name_Error` is raised by every attempt to create or open a text file (this is the appropriate behavior for an implementation which does not support text files other than standard input and output), then the tests contained in the following files should report NOT_APPLICABLE:

CE2109C.ADA CE3403E.ADA CE3602C.ADA CE3804G.ADA
CE3102A.ADA CE3403F.ADA CE3602D.ADA CE3804H.ADA
CE3102B.TST CE3404B.ADA CE3603A.ADA CE3804I.ADA
CE3102F.ADA CE3404C.ADA CE3604A.ADA CE3804J.ADA
CE3102G.ADA CE3404D.ADA CE3604B.ADA CE3804M.ADA
CE3102H.ADA CE3405A.ADA CE3605A.ADA CE3804O.ADA
CE3102J.ADA CE3405C.ADA CE3605B.ADA CE3804P.ADA
CE3102K.ADA CE3405D.ADA CE3605C.ADA CE3805A.ADA
CE3103A.ADA CE3406A.ADA CE3605D.ADA CE3805B.ADA
CE3104A.ADA CE3406B.ADA CE3605E.ADA CE3806A.ADA
CE3104B.ADA CE3406C.ADA CE3606A.ADA CE3806B.ADA
CE3104C.ADA CE3406D.ADA CE3606B.ADA CE3806D.ADA
CE3106A.ADA CE3407A.ADA CE3704A.ADA CE3806E.ADA
CE3106B.ADA CE3407B.ADA CE3704B.ADA CE3806F.ADA
CE3107A.TST CE3407C.ADA CE3704C.ADA CE3806H.ADA
CE3107B.ADA CE3408A.ADA CE3704D.ADA CE3902B.ADA
CE3108A.ADA CE3408B.ADA CE3704E.ADA CE3904A.ADA
CE3108B.ADA CE3408C.ADA CE3704F.ADA CE3904B.ADA
CE3110A.ADA CE3409A.ADA CE3704M.ADA CE3905A.ADA
CE3112C.ADA CE3409C.ADA CE3704N.ADA CE3905B.ADA
CE3112D.ADA CE3409D.ADA CE3704O.ADA CE3905C.ADA
CE3114A.ADA CE3409E.ADA CE3705A.ADA CE3905L.ADA
CE3115A.ADA CE3410A.ADA CE3705B.ADA CE3906A.ADA
CE3207A.ADA CE3410C.ADA CE3705C.ADA CE3906B.ADA
CE3301A.ADA CE3410D.ADA CE3705D.ADA CE3906C.ADA
CE3302A.ADA CE3410E.ADA CE3705E.ADA CE3906E.ADA
CE3304A.TST CE3411A.ADA CE3706D.ADA CE3906F.ADA
CE3305A.ADA CE3411C.ADA CE3706F.ADA CXAA001.A
CE3401A.ADA CE3412A.ADA CE3706G.ADA CXAA002.A
CE3402A.ADA CE3413A.ADA CE3804A.ADA CXAA003.A
CE3402C.ADA CE3413B.ADA CE3804B.ADA CXAA004.A
CE3402D.ADA CE3413C.ADA CE3804C.ADA CXAA005.A
CE3403A.ADA CE3414A.ADA CE3804D.ADA CXAA006.A
CE3403B.ADA CE3602A.ADA CE3804E.ADA CXAA007.A
CE3403C.ADA CE3602B.ADA CE3804F.ADA CXAA008.A
If **Use_Error** or **Name_Error** is raised by every attempt to create or open a text file with mode **In_File**, then the tests contained in the following files should report **NOT_APPLICABLE**:

<table>
<thead>
<tr>
<th>File 1</th>
<th>File 2</th>
<th>File 3</th>
<th>File 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CXAA009.A</td>
<td>CXAA014.A</td>
<td>CXAA019.A</td>
<td>EE3402B.ADA</td>
</tr>
<tr>
<td>CXAA010.A</td>
<td>CXAA015.A</td>
<td>CXF3A06.A</td>
<td>EE3409F.ADA</td>
</tr>
<tr>
<td>CXAA012.A</td>
<td>CXAA017.A</td>
<td>EE3203A.ADA</td>
<td>EE3204A.ADA</td>
</tr>
<tr>
<td>CXAA013.A</td>
<td>CXAA018.A</td>
<td>EE3203A.ADA</td>
<td>EE3204A.ADA</td>
</tr>
</tbody>
</table>

If **Use_Error** or **Name_Error** is raised by every attempt to create or open a text file with mode **Out_File**, then the tests contained in the following files should report **NOT_APPLICABLE**:

<table>
<thead>
<tr>
<th>File 1</th>
<th>File 2</th>
<th>File 3</th>
<th>File 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE3102F.ADA</td>
<td>CE3407C.ADA</td>
<td>CE3605C.ADA</td>
<td>CE3804H.ADA</td>
</tr>
<tr>
<td>CE3102H.ADA</td>
<td>CE3408A.ADA</td>
<td>CE3704A.ADA</td>
<td>CE3804I.ADA</td>
</tr>
<tr>
<td>CE3104B.ADA</td>
<td>CE3408C.ADA</td>
<td>CE3704C.ADA</td>
<td>CE3804J.ADA</td>
</tr>
<tr>
<td>CE3106A.ADA</td>
<td>CE3409C.ADA</td>
<td>CE3704D.ADA</td>
<td>CE3804M.ADA</td>
</tr>
<tr>
<td>CE3108A.ADA</td>
<td>CE3409D.ADA</td>
<td>CE3704E.ADA</td>
<td>CE3804P.ADA</td>
</tr>
<tr>
<td>CE3108B.ADA</td>
<td>CE3409E.ADA</td>
<td>CE3704F.ADA</td>
<td>CE3805A.ADA</td>
</tr>
<tr>
<td>CE3112D.ADA</td>
<td>CE3410C.ADA</td>
<td>CE3704M.ADA</td>
<td>CE3805B.ADA</td>
</tr>
<tr>
<td>CE31301A.ADA</td>
<td>CE3410D.ADA</td>
<td>CE3704N.ADA</td>
<td>CE3806A.ADA</td>
</tr>
<tr>
<td>CE31302A.ADA</td>
<td>CE3410E.ADA</td>
<td>CE3704O.ADA</td>
<td>CE3806B.ADA</td>
</tr>
<tr>
<td>CE3402A.ADA</td>
<td>CE3411A.ADA</td>
<td>CE3705B.ADA</td>
<td>CE3806D.ADA</td>
</tr>
<tr>
<td>CE3403B.ADA</td>
<td>CE3411C.ADA</td>
<td>CE3705C.ADA</td>
<td>CE3806G.ADA</td>
</tr>
<tr>
<td>CE3403C.ADA</td>
<td>CE3412A.ADA</td>
<td>CE3705D.ADA</td>
<td>CE3902B.ADA</td>
</tr>
<tr>
<td>CE3403E.ADA</td>
<td>CE3413A.ADA</td>
<td>CE3705E.ADA</td>
<td>CE3904B.ADA</td>
</tr>
<tr>
<td>CE3403F.ADA</td>
<td>CE3413B.ADA</td>
<td>CE3706D.ADA</td>
<td>CE3905A.ADA</td>
</tr>
<tr>
<td>CE3404B.ADA</td>
<td>CE3413C.ADA</td>
<td>CE3706G.ADA</td>
<td>CE3905C.ADA</td>
</tr>
<tr>
<td>CE3404C.ADA</td>
<td>CE3602A.ADA</td>
<td>CE3804A.ADA</td>
<td>CE3905L.ADA</td>
</tr>
<tr>
<td>CE3404D.ADA</td>
<td>CE3602B.ADA</td>
<td>CE3804B.ADA</td>
<td>CE3906B.ADA</td>
</tr>
<tr>
<td>CE3406A.ADA</td>
<td>CE3602D.ADA</td>
<td>CE3804D.ADA</td>
<td>CE3906C.ADA</td>
</tr>
<tr>
<td>CE3406C.ADA</td>
<td>CE3603A.ADA</td>
<td>CE3804E.ADA</td>
<td>CE3903A.ADA</td>
</tr>
<tr>
<td>CE3406D.ADA</td>
<td>CE3604A.ADA</td>
<td>CE3804F.ADA</td>
<td>CE3204A.ADA</td>
</tr>
<tr>
<td>CE3407A.ADA</td>
<td>CE3604B.ADA</td>
<td>CE3804G.ADA</td>
<td>CE3412C.ADA</td>
</tr>
</tbody>
</table>

If **Use_Error** or **Name_Error** is raised by every attempt to create or open a text file with mode **Out_File**, then the tests contained in the following files should report **NOT_APPLICABLE**:

<table>
<thead>
<tr>
<th>File 1</th>
<th>File 2</th>
<th>File 3</th>
<th>File 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE2109C.ADA</td>
<td>CE3110A.ADA</td>
<td>CE3403F.ADA</td>
<td>CE3409C.ADA</td>
</tr>
<tr>
<td>CE3102A.ADA</td>
<td>CE3112C.ADA</td>
<td>CE3404B.ADA</td>
<td>CE3409D.ADA</td>
</tr>
<tr>
<td>CE3102B.TST</td>
<td>CE3112D.ADA</td>
<td>CE3404C.ADA</td>
<td>CE3409E.ADA</td>
</tr>
<tr>
<td>CE3102F.ADA</td>
<td>CE3114A.ADA</td>
<td>CE3404D.ADA</td>
<td>CE3410A.ADA</td>
</tr>
<tr>
<td>CE3102G.ADA</td>
<td>CE3115A.ADA</td>
<td>CE3405A.ADA</td>
<td>CE3410C.ADA</td>
</tr>
<tr>
<td>CE3102H.ADA</td>
<td>CE3207A.ADA</td>
<td>CE3405C.ADA</td>
<td>CE3410D.ADA</td>
</tr>
<tr>
<td>CE3102J.ADA</td>
<td>CE3301A.ADA</td>
<td>CE3405D.ADA</td>
<td>CE3410E.ADA</td>
</tr>
<tr>
<td>CE3102K.ADA</td>
<td>CE3302A.ADA</td>
<td>CE3406A.ADA</td>
<td>CE3411A.ADA</td>
</tr>
<tr>
<td>CE3103A.ADA</td>
<td>CE3304A.TST</td>
<td>CE3406B.ADA</td>
<td>CE3411C.ADA</td>
</tr>
<tr>
<td>CE3104A.ADA</td>
<td>CE3305A.ADA</td>
<td>CE3406C.ADA</td>
<td>CE3412A.ADA</td>
</tr>
<tr>
<td>CE3104B.ADA</td>
<td>CE3401A.ADA</td>
<td>CE3406D.ADA</td>
<td>CE3413A.ADA</td>
</tr>
<tr>
<td>CE3104C.ADA</td>
<td>CE3402A.ADA</td>
<td>CE3407A.ADA</td>
<td>CE3413B.ADA</td>
</tr>
<tr>
<td>CE3106A.ADA</td>
<td>CE3402C.ADA</td>
<td>CE3407B.ADA</td>
<td>CE3413C.ADA</td>
</tr>
<tr>
<td>CE3106B.ADA</td>
<td>CE3402D.ADA</td>
<td>CE3407C.ADA</td>
<td>CE3414A.ADA</td>
</tr>
<tr>
<td>CE3107A.TST</td>
<td>CE3403A.ADA</td>
<td>CE3408A.ADA</td>
<td>CE3602A.ADA</td>
</tr>
<tr>
<td>CE3107B.ADA</td>
<td>CE3403B.ADA</td>
<td>CE3408B.ADA</td>
<td>CE3602B.ADA</td>
</tr>
<tr>
<td>CE3108A.ADA</td>
<td>CE3403C.ADA</td>
<td>CE3408C.ADA</td>
<td>CE3602C.ADA</td>
</tr>
<tr>
<td>CE3108B.ADA</td>
<td>CE3403E.ADA</td>
<td>CE3409A.ADA</td>
<td>CE3602D.ADA</td>
</tr>
</tbody>
</table>
If `Use_Error` or `Name_Error` is raised by every attempt to create or open a text file with mode `Append_File`, then the tests contained in the following files should report `NOT_APPLICABLE`:

- CXAA001.A
- CXAA002.A
- CXAA006.A
- CXAA007.A
- CXAA008.A
- CXAA013.A
- CXAA015.A
- CXAA015.A

If `Reset` is not supported for text files, then the following tests should report `NOT_APPLICABLE`:

- CE3104C.ADA
- CE3115A.ADA

If `Delete` is not supported for text files, then the following tests should report `NOT_APPLICABLE`:

- CE3110A.ADA
- CE3114A.ADA

If association of multiple internal text files (opened for reading and writing) to a single external file is not supported, then the test contained in the following file should report `NOT_APPLICABLE`:

- CE3115A.ADA

If there are no inappropriate values for either line length or page length, then the test contained in the following file should report `NOT_APPLICABLE`:

- CE3304A.TST

If the value of `Count'Last` is greater than `150_000`, then the following test should report `NOT_APPLICABLE`:

- CE3413B.ADA
D.2.6 Text Files (Supported Features)

If Create with mode In_File is supported for text files, then the test contained in the following file should report NOT_APPLICABLE:
CE3102E.ADA

If Open with mode In_File is supported for text files, then the test contained in the following file should report NOT_APPLICABLE:
CE3102J.ADA

If Create with mode Out_File is supported for text files, then the test contained in the following file should report NOT_APPLICABLE:
CE3102I.ADA

If Open with mode Out_File is supported for text files, then the following test should report NOT_APPLICABLE:
CE3102K.ADA

If Reset is supported for text files, then the test contained in the following file should report NOT_APPLICABLE:
CE3102F.ADA

If Delete for text files is supported, then the test contained in the following file should report NOT_APPLICABLE:
CE3102G.ADA

D.2.7 Sequential Files (Non-supported Features)

If Use_Error or Name_Error is raised by every attempt to create or open a sequential file (this is appropriate behavior for an implementation which does not support sequential files), then the tests contained in the following files should report NOT_APPLICABLE:

CE2102A.ADA    CE2106A.ADA    CE2201D.DEP    CE2204A.ADA
CE2102C.TST    CE2108E.ADA    CE2201E.DEP    CE2204B.ADA
CE2102G.ADA    CE2108F.ADA    CE2201F.ADA    CE2204C.ADA
CE2102N.ADA    CE2109A.ADA    CE2201G.ADA    CE2204D.ADA
CE2102O.ADA    CE2110A.ADA    CE2201H.ADA    CE2205A.ADA
CE2102P.ADA    CE2111A.ADA    CE2201I.ADA    CE2206A.ADA
CE2102Q.ADA    CE2111C.ADA    CE2201J.ADA    CE2208B.ADA
CE2102X.ADA    CE2111F.ADA    CE2201K.ADA    CXA8001.A
CE2103A.TST    CE2111I.ADA    CE2201L.ADA    CXA8002.A
CE2103C.ADA    CE2201A.ADA    CE2201M.ADA
CE2104A.ADA    CE2201B.ADA    CE2201N.ADA
CE2104B.ADA    CE2201C.ADA    CE2203A.TST

If Use_Error or Name_Error is raised by every attempt to create or open a sequential file with mode In_File, then the tests contained in the following files should report NOT_APPLICABLE:
If `Use_Error` or `Name_Error` is raised by every attempt to create or open a sequential file with mode `Out_File`, then the tests contained in the following files should report `NOT_APPLICABLE`:

```plaintext
CE2102A.ADA CE2102B.ADA CE2201B.ADA CE2201M.ADA
CE2102C.TST CE2106A.ADA CE2201C.ADA CE2201N.ADA
CE2102G.ADA CE2108E.ADA CE2201D.DEP CE2203A.TST
CE2102N.ADA CE2108F.ADA CE2201E.DEP CE2204A.ADA
CE2102O.ADA CE2109A.ADA CE2201F.ADA CE2204B.ADA
CE2102P.ADA CE2110A.ADA CE2201G.ADA CE2204C.ADA
CE2102Q.ADA CE2111A.ADA CE2201H.ADA CE2204D.ADA
CE2102X.ADA CE2111C.ADA CE2201I.ADA CE2205A.ADA
CE2103A.TST CE2111F.ADA CE2201J.ADA CE2206A.ADA
CE2103C.ADA CE2111L.ADA CE2201K.ADA CE2208B.ADA
CE2104A.ADA CE2201A.ADA CE2201L.ADA CXA8001.A
```

If `Use_Error` or `Name_Error` is raised by every attempt to create or open a sequential file with mode `APPEND_FILE`, then the tests contained in the following files should report `NOT_APPLICABLE`:

```plaintext
CXA8001.A
```

If reset to mode `Out_File` is not supported for sequential files, then the tests contained in the following files should report `NOT_APPLICABLE`:

```plaintext
CE2111C.ADA CE2111F.ADA CE2111L.ADA
```

If reset to mode `In_File` is not supported for sequential files, then the tests contained in the following files should report `NOT_APPLICABLE`:

```plaintext
CE2111F.ADA CE2111L.ADA CE2204C.ADA
```

If `Delete` for sequential files is not supported, then the tests contained in the following files should report `NOT_APPLICABLE`:

```plaintext
CE2106A.ADA CE2110A.ADA
```

If the implementation cannot restrict the file capacity for a sequential file, then the test contained in the following file should report `NOT_APPLICABLE`:

```plaintext
CE2203A.TST
```

### D.2.8 Sequential Files (Supported Features)

If `Create` with mode `In_File` is supported for sequential files, then the test contained in the following file should report `NOT_APPLICABLE`:

```plaintext
CE2102D.ADA
```
If `Open` with mode `In_File` is supported for sequential files, then the test contained in the following file should report `NOT_APPLICABLE`:

CE2102N.ADA

If `Reset` to mode `In_File` is supported for sequential files, then the test contained in the following file should report `NOT_APPLICABLE`:

CE2102O.ADA

If `Create` with mode `Out_File` is supported, then the test contained in the following file should report `NOT_APPLICABLE`:

CE2102E.ADA

If `Open` with mode `Out_File` is supported, then the test contained in the following file should report `NOT_APPLICABLE`:

CE2102P.ADA

If `Reset` to mode `Out_File` is supported for sequential files, then the test contained in the following file should report `NOT_APPLICABLE`:

CE2102Q.ADA

**D.2.9 Direct Files (Non-supported Features)**

If `Use_Error` or `Name_Error` is raised by every attempt to create or open a direct access file (this is appropriate behavior for an implementation which does not support direct access files), then the tests contained in the following files should report `NOT_APPLICABLE`:


CE2102H.TST  CE2106B.ADA  CE2109B.ADA  CE2110C.ADA  CE2111B.ADA  CE2111E.ADA  CE2111G.ADA  CE2401A.ADA  CE2401B.ADA  CE2401C.ADA  CE2401D.ADA  CE2402A.TST

CE2102K.ADA  CE2108G.ADA  CE2109B.ADA  CE2110C.ADA  CE2111B.ADA  CE2111E.ADA  CE2111G.ADA  CE2401A.ADA  CE2401B.ADA  CE2401C.ADA  CE2401D.ADA  CE2402A.TST

CE2102L.ADA  CE2108H.ADA  CE2109B.ADA  CE2110C.ADA  CE2111B.ADA  CE2111E.ADA  CE2111G.ADA  CE2401A.ADA  CE2401B.ADA  CE2401C.ADA  CE2401D.ADA  CE2402A.TST

CE2102M.ADA  CE2108H.ADA  CE2109B.ADA  CE2110C.ADA  CE2111B.ADA  CE2111E.ADA  CE2111G.ADA  CE2401A.ADA  CE2401B.ADA  CE2401C.ADA  CE2401D.ADA  CE2402A.TST

CE2102Q.ADA  CE2112A.ADA  CE2112C.ADA  CE2112D.ADA  CE2112E.ADA  CE2112F.ADA  CE2112G.ADA  CE2401A.ADA  CE2401B.ADA  CE2401C.ADA  CE2401D.ADA  CE2402A.TST

If `Use_Error` or `Name_Error` is raised by every attempt to create or open a direct access file with mode `In_File`, then the tests contained in the following files should report `NOT_APPLICABLE`:

CE2102K.ADA  CE2111B.ADA  CE2401E.ADA  CE2406A.ADA

CE2102L.ADA  CE2111E.ADA  CE2401F.ADA  CE2407A.ADA

CE2104C.ADA  CE2401A.ADA  CE2401H.ADA  CE2411A.ADA

CE2108H.ADA  CE2401B.ADA  CE2401I.ADA  CE2405B.ADA

CE2108H.ADA  CE2401B.ADA  CE2401I.ADA  CE2405B.ADA

CE2108H.ADA  CE2401B.ADA  CE2401I.ADA  CE2405B.ADA

CE2108H.ADA  CE2401B.ADA  CE2401I.ADA  CE2405B.ADA
If `Use_Error` or `Name_Error` is raised by every attempt to create or open a direct access file with mode `Out_File`, then the tests contained in the following files should report `NOT_APPLICABLE`:

CE2102B.ADA  CE2104D.ADA  CE2404A.ADA  CE2410B.ADA  
CE2102K.ADA  CE2106B.ADA  CE2404B.ADA  CE2411A.ADA  
CE2102R.ADA  CE2108G.ADA  CE2405B.ADA  CXA8003.A  
CE2102W.ADA  CE2108H.ADA  CE2407A.ADA  CXA9001.A  
CE2102Y.ADA  CE2110C.ADA  CE2407B.ADA  CXA9002.A  
CE2103B.TST  CE2111B.ADA  CE2408A.ADA  
CE2103D.ADA  CE2111E.ADA  CE2409B.ADA  
CE2104C.ADA  CE2403A.TST  CE2410A.ADA  

If `Use_Error` or `Name_Error` is raised by every attempt to create or open a direct access file with mode `InOut_File`, then the tests contained in the following files should report `NOT_APPLICABLE`:

CE2102H.TST  CE2111B.ADA  CE2401F.ADA  CE2406A.ADA  
CE2102K.ADA  CE2111E.ADA  CE2401H.ADA  CE2408B.ADA  
CE2102S.ADA  CE2111G.ADA  CE2401I.ADA  CE2409A.ADA  
CE2102T.ADA  CE2401A.ADA  CE2401J.ADA  CE2411A.ADA  
CE2102U.ADA  CE2401B.ADA  CE2401K.ADA  
CE2102V.ADA  CE2401C.ADA  CE2401L.ADA  
CE2109B.ADA  CE2401E.ADA  CE2405B.ADA  

If `Delete` for direct access files is not supported, then the following tests should report `NOT_APPLICABLE`:

CE2106B.ADA  CE2110C.ADA  

If the implementation cannot restrict the file capacity for a direct file, then the test contained in the following file should report `NOT_APPLICABLE`:

CE2403A.TST  

D.2.10 Direct Files (Supported Features)

If `Create` with mode `In_File` is supported for direct access files, then the test contained in the following file should report `NOT_APPLICABLE`:

CE2102I.ADA  

If `Open` with mode `In_File` is supported for direct access files, then the test contained in the following file should report `NOT_APPLICABLE`:

CE2102T.ADA  

If `Reset` with mode `In_File` is supported for direct access files, then the test contained in the following file should report `NOT_APPLICABLE`:

CE2102U.ADA  

If `Create` with mode `Out_File` is supported for direct access files, then the test contained in the following file should report `NOT_APPLICABLE`:

CE2102J.ADA
If `Open` with mode `Out_File` is supported for direct access files, then the test contained in the following file should report NOT_APPLICABLE:

CE2102V.ADA

If `Reset` with mode `Out_File` is supported for direct access files, then the test contained in the following file should report NOT_APPLICABLE:

CE2102W.ADA

If `Create` with mode `InOut_File` is supported for direct access files, then the test contained in the following file should report NOT_APPLICABLE:

CE2102F.ADA

If `Open` with mode `InOut_File` is supported for direct access files, then the test contained in the following file should report NOT_APPLICABLE:

CE2102R.ADA

If `Reset` to mode `InOut_File` is supported for direct access files, then the test contained in the following file should report NOT_APPLICABLE:

CE2102S.ADA

**D.2.11 Stream Files (Non-supported Features)**

If `Use_Error` or `Name_Error` is raised by every attempt to create or open a stream file (this is the appropriate behavior for an implementation which does not support stream files), then the tests contained in the following files should report NOT_APPLICABLE:

CXAA019.A  CXAC003.A  CXAC006.A  CXACB01.A
CXAC001.A  CXAC004.A  CXACA01.A  CXACB02.A
CXAC002.A  CXAC005.A  CXACA02.A  CXACC01.A

**D.2.12 Wide Text Files (Non-supported Features)**

If `Use_Error` or `Name_Error` is raised by every attempt to create or open a wide text file (this is the appropriate behavior for an implementation which does not support wide text files), then the tests contained in the following files should report NOT_APPLICABLE:

CXAA019.A  CXAB001.A

**D.2.13 File I/O Tests**

If sequential, text, wide text, wide wide text, direct access, and stream files are not supported, then the tests contained in the following files should report NOT_APPLICABLE:

CE2102A.ADA  CE2102B.ADA  CE2102C.TST  CE2102G.ADA
CE2102H.TST  CE2102K.ADA  CE2102N.ADA  CE2102O.ADA
CE2102P.ADA  CE2102Q.ADA  CE2102R.ADA  CE2102S.ADA
CE2102T.ADA  CE2102U.ADA  CE2102V.ADA  CE2102W.ADA
<table>
<thead>
<tr>
<th>File 1</th>
<th>File 2</th>
<th>File 3</th>
<th>File 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE2102X.ADA</td>
<td>CE2208B.ADA</td>
<td>CE3302A.ADA</td>
<td>CE3604B.ADA</td>
</tr>
<tr>
<td>CE2102Y.ADA</td>
<td>CE2401A.ADA</td>
<td>CE3304A.TST</td>
<td>CE3605A.ADA</td>
</tr>
<tr>
<td>CE2103A.TST</td>
<td>CE2401B.ADA</td>
<td>CE3305A.ADA</td>
<td>CE3605B.ADA</td>
</tr>
<tr>
<td>CE2103B.TST</td>
<td>CE2401C.ADA</td>
<td>CE3401A.ADA</td>
<td>CE3605C.ADA</td>
</tr>
<tr>
<td>CE2103C.ADA</td>
<td>CE2401E.ADA</td>
<td>CE3402A.ADA</td>
<td>CE3605D.ADA</td>
</tr>
<tr>
<td>CE2103D.ADA</td>
<td>CE2401F.ADA</td>
<td>CE3402C.ADA</td>
<td>CE3605E.ADA</td>
</tr>
<tr>
<td>CE2104A.ADA</td>
<td>CE2401H.ADA</td>
<td>CE3402D.ADA</td>
<td>CE3606A.ADA</td>
</tr>
<tr>
<td>CE2104B.ADA</td>
<td>CE2401I.ADA</td>
<td>CE3403A.ADA</td>
<td>CE3606B.ADA</td>
</tr>
<tr>
<td>CE2104C.ADA</td>
<td>CE2401J.ADA</td>
<td>CE3403B.ADA</td>
<td>CE3704A.ADA</td>
</tr>
<tr>
<td>CE2104D.ADA</td>
<td>CE2401K.ADA</td>
<td>CE3403C.ADA</td>
<td>CE3704B.ADA</td>
</tr>
<tr>
<td>CE2106A.ADA</td>
<td>CE2401L.ADA</td>
<td>CE3403E.ADA</td>
<td>CE3704C.ADA</td>
</tr>
<tr>
<td>CE2106B.ADA</td>
<td>CE2403A.TST</td>
<td>CE3403F.ADA</td>
<td>CE3704D.ADA</td>
</tr>
<tr>
<td>CE2108E.ADA</td>
<td>CE2404A.ADA</td>
<td>CE3404B.ADA</td>
<td>CE3704E.ADA</td>
</tr>
<tr>
<td>CE2108F.ADA</td>
<td>CE2404B.ADA</td>
<td>CE3404C.ADA</td>
<td>CE3704F.ADA</td>
</tr>
<tr>
<td>CE2108G.ADA</td>
<td>CE2405B.ADA</td>
<td>CE3404D.ADA</td>
<td>CE3704M.ADA</td>
</tr>
<tr>
<td>CE2108H.ADA</td>
<td>CE2406A.ADA</td>
<td>CE3405A.ADA</td>
<td>CE3704N.ADA</td>
</tr>
<tr>
<td>CE2109A.ADA</td>
<td>CE2407A.ADA</td>
<td>CE3405C.ADA</td>
<td>CE3704O.ADA</td>
</tr>
<tr>
<td>CE2109B.ADA</td>
<td>CE2407B.ADA</td>
<td>CE3405D.ADA</td>
<td>CE3705A.ADA</td>
</tr>
<tr>
<td>CE2109C.ADA</td>
<td>CE2408A.ADA</td>
<td>CE3406A.ADA</td>
<td>CE3705B.ADA</td>
</tr>
<tr>
<td>CE2110A.ADA</td>
<td>CE2408B.ADA</td>
<td>CE3406B.ADA</td>
<td>CE3705C.ADA</td>
</tr>
<tr>
<td>CE2110B.ADA</td>
<td>CE2409A.ADA</td>
<td>CE3406C.ADA</td>
<td>CE3705D.ADA</td>
</tr>
<tr>
<td>CE2111A.ADA</td>
<td>CE2409B.ADA</td>
<td>CE3406D.ADA</td>
<td>CE3705E.ADA</td>
</tr>
<tr>
<td>CE2111B.ADA</td>
<td>CE2410A.ADA</td>
<td>CE3407A.ADA</td>
<td>CE3706D.ADA</td>
</tr>
<tr>
<td>CE2111C.ADA</td>
<td>CE2410B.ADA</td>
<td>CE3407B.ADA</td>
<td>CE3706F.ADA</td>
</tr>
<tr>
<td>CE2111E.ADA</td>
<td>CE2411A.ADA</td>
<td>CE3407C.ADA</td>
<td>CE3706G.ADA</td>
</tr>
<tr>
<td>CE2111F.ADA</td>
<td>CE3102A.ADA</td>
<td>CE3408A.ADA</td>
<td>CE3804A.ADA</td>
</tr>
<tr>
<td>CE2111G.ADA</td>
<td>CE3102B.TST</td>
<td>CE3408B.ADA</td>
<td>CE3804B.ADA</td>
</tr>
<tr>
<td>CE2201A.ADA</td>
<td>CE3102F.ADA</td>
<td>CE3408C.ADA</td>
<td>CE3804C.ADA</td>
</tr>
<tr>
<td>CE2201B.ADA</td>
<td>CE3102G.ADA</td>
<td>CE3409A.ADA</td>
<td>CE3804D.ADA</td>
</tr>
<tr>
<td>CE2201C.ADA</td>
<td>CE3102H.ADA</td>
<td>CE3409C.ADA</td>
<td>CE3804E.ADA</td>
</tr>
<tr>
<td>CE2201D.DEP</td>
<td>CE3102J.ADA</td>
<td>CE3409D.ADA</td>
<td>CE3804F.ADA</td>
</tr>
<tr>
<td>CE2201E.DEP</td>
<td>CE3102K.ADA</td>
<td>CE3409E.ADA</td>
<td>CE3804G.ADA</td>
</tr>
<tr>
<td>CE2201F.DEP</td>
<td>CE3103A.ADA</td>
<td>CE3410A.ADA</td>
<td>CE3804H.ADA</td>
</tr>
<tr>
<td>CE2201G.ADA</td>
<td>CE3104A.ADA</td>
<td>CE3410C.ADA</td>
<td>CE3804I.ADA</td>
</tr>
<tr>
<td>CE2201H.ADA</td>
<td>CE3104B.ADA</td>
<td>CE3410D.ADA</td>
<td>CE3804J.ADA</td>
</tr>
<tr>
<td>CE2201I.ADA</td>
<td>CE3104C.ADA</td>
<td>CE3410E.ADA</td>
<td>CE3804M.ADA</td>
</tr>
<tr>
<td>CE2201J.ADA</td>
<td>CE3106A.ADA</td>
<td>CE3411A.ADA</td>
<td>CE3804O.ADA</td>
</tr>
<tr>
<td>CE2201K.ADA</td>
<td>CE3106B.ADA</td>
<td>CE3411C.ADA</td>
<td>CE3804P.ADA</td>
</tr>
<tr>
<td>CE2201L.ADA</td>
<td>CE3107A.TST</td>
<td>CE3412A.ADA</td>
<td>CE3805A.ADA</td>
</tr>
<tr>
<td>CE2201M.ADA</td>
<td>CE3107B.ADA</td>
<td>CE3413A.ADA</td>
<td>CE3805B.ADA</td>
</tr>
<tr>
<td>CE2201N.ADA</td>
<td>CE3108A.ADA</td>
<td>CE3413B.ADA</td>
<td>CE3806A.ADA</td>
</tr>
<tr>
<td>CE2203A.TST</td>
<td>CE3110A.ADA</td>
<td>CE3414A.ADA</td>
<td>CE3806D.ADA</td>
</tr>
<tr>
<td>CE2204A.ADA</td>
<td>CE3112C.ADA</td>
<td>CE3602A.ADA</td>
<td>CE3806E.ADA</td>
</tr>
<tr>
<td>CE2204B.ADA</td>
<td>CE3112D.ADA</td>
<td>CE3602B.ADA</td>
<td>CE3806G.ADA</td>
</tr>
<tr>
<td>CE2204C.ADA</td>
<td>CE3114A.ADA</td>
<td>CE3602C.ADA</td>
<td>CE3806H.ADA</td>
</tr>
<tr>
<td>CE2204D.ADA</td>
<td>CE3115A.ADA</td>
<td>CE3602D.ADA</td>
<td>CE3902B.ADA</td>
</tr>
<tr>
<td>CE2205A.ADA</td>
<td>CE3207A.ADA</td>
<td>CE3603A.ADA</td>
<td>CE3904A.ADA</td>
</tr>
<tr>
<td>CE2206A.ADA</td>
<td>CE3301A.ADA</td>
<td>CE3604A.ADA</td>
<td>CE3904B.ADA</td>
</tr>
</tbody>
</table>
D.2.14 Memory for Allocated Objects

If a large amount of memory (more than 32 megabytes for a typical implementation) is available for allocated objects (those created by `new`), then the test contained in the following file should report NOT_APPLICABLE:

CB10002.A

D.2.15 Task Attributes

If Annex C (Systems Programming) is tested and the size of a task attribute is limited such that an attribute of a controlled type is not supported, then the test contained in the following file should report NOT_APPLICABLE:

CXC7003.A

D.2.16 Reserved Interrupts

If Annex C (Systems Programming) is tested and no interrupts are reserved, then the tests contained in the following files should report NOT_APPLICABLE:

CXC3002.A  CXC3005.A

D.2.17 Multiprocessor Systems

If Annex D (Real-Time Systems) is tested and the target is a multiprocessor, then the tests contained in the following files should report NOT_APPLICABLE:

D.2.18 Non-binary Machine Radix

If Annex G (Numerics) is tested and the machine radix is not a power of two, then the test contained in the following file should report NOT_APPLICABLE:

CXG2010.A
Annex E: Guidelines for Test Development

The guidelines used for developing recent ACATS tests are summarized in this Annex. Developers of potential ACATS tests should follow these guidelines closely. Tests that deviate extensively from these guidelines are far less likely to be added to the ACATS than those that follow them carefully.

- Tests should follow the test structure and organization of existing ACATS tests. Many details of existing tests are described elsewhere in this document. Important topics include:
  - Test classes (see 4.2);
  - Test naming conventions (see 4.3.2 and 4.3.3);
  - Test layout and prologue (see 4.4);
  - Identifier and reserved word conventions (see 4.5);
  - Library unit naming within tests (see 4.3.3);
  - Executable test structure (see 4.6); and
  - Indication of errors in B and L tests (see 4.6).

- The test number (character positions 6 and 7) should be assigned letters rather than numbers by submitting test authors. Letters that are likely to be unique (such as the author's initials) are preferred. (Of course, different tests for the same clause with the same author should have different names.) The ACAA Technical Agent will perform the final naming of tests in order to ensure that the names are unique and appropriate. Following this guideline reduces the chance of test conflicts between authors.

- If a single test file contains multiple compilation units, they should be given in an order such that any dependent units follow the units they depend on (so an implementation can process the units sequentially in order). However, it should be assumed that all units in a file will be presented to the compiler simultaneously. If a test requires units to be presented in a specific order (as some separate compilation tests do), the units with ordering requirements should be in separate files, and the required order should be documented in the test prologue.

- When possible for B-Tests, only the last unit in a file should contain errors (or even better, units with errors should be in separate files from those without errors). This avoids penalizing implementations that process units in a file sequentially and stop on the first bad unit. (Multiple units in a single file are fairly rare in real user code; the ACATS shouldn't require work in areas not useful to typical users.) This guideline should be violated only when the number of units with errors would be prohibitive to have in separate files. (The maximum number of separate files in a test is 36 plus any foundations, because the naming conventions for tests only leave a single character for sequence numbers.)

- C-Tests (especially those testing rules that are not runtime checks) should be written in a usage-oriented style. That means that the tests should reflect the way the features are typically used in practice. Using a feature with no context is discouraged. For instance, C-Tests for limited with clauses should use them to declare mutually dependent types (the reason that limited with clauses were added to the language) rather than just using them to replace regular with clauses.

- Tests should avoid the use of Text_IO (unless required by the test objective). In particular, C-Tests should not create messages with Text_IO; all messages should be generated via the subprograms in the Report package. Messages in C-Tests should be written in mixed case, not all UPPER CASE. Failure messages should be unique, so that the exact failure can be pinpointed. This is often accomplished by including a subtest identifier in the messages.

- Tests can combine multiple objectives if a test for a single objective is very short. However, the objectives should be related, and no more than 4 objectives should be tested in a single test (to
avoid creating gigantic tests that are hard to understand and use). Objectives from different clauses should never be combined, as that makes it hard to find the associated test (it will necessarily be filed in the wrong clause for one of the objectives).

- When possible, tests should define (and thus share) foundation code (see 4.1.4). Foundation packages are a better alternative than creating large tests with many objectives when the primary reason for combining the objectives is to avoid writing set-up code multiple times. Foundation code is specific to tests for a particular clause, however, so this technique cannot be used to combine objectives from multiple clauses.

- When a rule includes a term defined elsewhere, testing of the rule should include testing all of the combinations implied by the term. For instance, if we have a definition like "something is either this, that, or fuzzy", then a test for a test objective involving something should test cases where something is this and that and fuzzy. If multiple layers of definitions make this impractical, then a wide selection of combinations (as different as possible) should be tried. The only exception to this rule is if separate tests of the definition exist or should exist (that is, there is a test objective to test that the definition is appropriately implemented).

- When appropriate, tests should try a variety of things. For instance, when testing subprograms, both procedures and functions should be tested, with varying numbers of parameters, and with different modes and types. Similarly, types should be more than just Integer – tagged types, tasks, protected types, and anonymous access types should be tried. However, adding variety should not be used as an excuse to create multiple tests for an objective when one will do. That is, variety is a secondary goal; exhaustive coverage of possibilities isn't needed (unless the testing includes testing a term defined elsewhere, as described in the previous guideline). Remember that the goal isn't to test combinations of features; the point of using variety is to ensure that the objective being tested works in more than just the simplest cases.

- Tests should generally use only the 7-bit ASCII characters. However, some tests will need to use other characters in order to test Wide_Wide_Character support, Unicode characters in identifiers, and the like. Such tests should be encoded in UTF-8 and start with a UTF-8 Byte Order Mark. Tests should only use the code points that were assigned in version 4.0 of the Unicode standard; when possible, using only the roughly 680 characters generally available on US Windows systems is recommended.

- When constructing tests that check to see that run-time checks are made, take special care that 11.6 permissions don't render the test impotent. 11.6(5) allows language-defined checks to be optimized away if the result of the operation is not used (even if the exception is handled). That means it is critical that the values that fail checks are used in some way afterwards (even though a correct program will never execute that code). Failure to do that could allow a compiler to optimize the entire test away, and that would require the test to be corrected later.

- When creating a B-Test for which different parts test different errors, each error should identify the intended failure. The standard error indication includes a colon; any needed text can follow that colon. If the error identification will not fit on one line, place it somewhere else and index to it. One common way to do this is to put a list of intended errors into the header, labeling each with a letter. Then each error comment can just identify the letter of the intended error.

- Tests that cover test objectives that are documented as untested are especially welcome as test submissions. Tests that cover previously tested objectives are less likely to be included in the test suite.
Annex F: Definitions

**ACATS Modification List.** (Abbreviated AML) A list maintained by the ACAA documenting the currently modified and withdrawn tests. It also documents any new tests that have been or will be added to the test suite. The ACATS modification list is updated from time to time as challenges from implementers are received and processed, new tests are created, or as other technical information is received.

**Acceptable result.** The result of processing an ACATS test program that meets the explicit grading criteria for a grade of "passed" or inapplicable.

**ACVC Implementer's Guide.** (Abbreviated AIG) A document describing the test objectives used to produce test programs for [Ada83] ACVC versions (1.1-1.11). AIG section references are embedded in [Ada83] test naming conventions.

**Ada.** Short for **Ada programming language.** The term Ada by itself always refers to the most current ISO/IEC standard document(s); if a specific version of the language standards is meant it will always be referred to explicitly (for instance, as [Ada83] or [Ada95]).

**Ada Conformity Assessment Authority.** (Abbreviated ACAA) The part of the certification body that provides technical guidance for operations of the Ada certification system.

**Ada Conformity Assessment Laboratory.** (Abbreviated ACAL) The part of the certification body that carries out the procedures required to perform conformity assessment of an Ada implementation. (Formerly AVF)

**Ada Conformity Assessment Test Report.** (Abbreviated ACATR) A report summarizing the results of formal ACATS testing. Test Reports are issued only after witness testing is completed, and contain a summary of the testing (including which Specialized Needs Annexes were tested, any test modifications needed, and the values used in customizing the support files). Recent test reports can be found on-line at http://www.adaic.org/compilers/ada95.html, linked from the Certified Processors List.

**Ada implementation.** An Ada compilation system, including any required run-time support software, together with its host and target computer systems.

**Ada Joint Program Office.** (Abbreviated AJPO) An organization within the U.S. Department of Defense that sponsored the development of the ACVC and formerly provided policy and guidance for an Ada certification system.

**Ada programming language.** The language defined by the current Ada Standard documents.

**Ada Resource Association.** (Abbreviated ARA) The trade association that sponsors the Ada conformity assessment system.

**Ada Standard documents.** The documents that define the Ada programming language: reference [Ada95], its corrigendum [TC1], and its Amendment [Amend1]. An unofficial document informally known as the “Ada 2005 Reference Manual” merges all three documents and is what is usually used by Ada practitioners.

**Ada Validation Facility.** (Abbreviated AVF) Former designation of an Ada Conformity Assessment Laboratory.

**Ada Validation Organization.** (Abbreviated AVO) Organization that formerly performed the functions of the Ada Conformity Assessment Authority.
Certification Body. The organizations (ACAA and ACALs) collectively responsible for defining and implementing Ada conformity assessments, including production and maintenance of the ACATS tests, and award of Ada Conformity Assessment Certificates.

Certified Processors List. (Abbreviated CPL) A published list identifying all certified Ada implementations. The CPL is available on the Ada Information Clearinghouse Internet site (www.adaic.org).

Challenge. A documented disagreement with the test objective, test code, test grading criteria, or result of processing an ACATS test program when the result is not PASSED or INAPPLICABLE according to the established grading criteria. A challenge is submitted to the ACAA.

Conforming implementation. An implementation that produces an acceptable result for every applicable test. Any deviation constitutes a non-conformity.

Core language. Sections 2-13 and Annexes A, B, and J of [Ada95]. All implementations are required to implement the core language. The tests for core language features are required of all implementations.

Coverage documents. Documents containing an analysis of every paragraph of the Ada Standard documents. Each paragraph has an indication of whether it contains a testable Ada requirement, and if so, suggested test objectives to cover the requirements of the paragraphs. Paragraphs that include objectives also indicate what ACATS test(s) specifically test those objectives.

Deviation. Failure of an Ada implementation to produce an acceptable result when processing an ACATS test program.

Foundation Code. Code used by multiple tests; foundation code is designed to be reusable. Generally a foundation is a package containing types, variables, and subprograms that are applicable and useful to a series of related tests. Foundation code is never expected to cause compile time errors. It may be compiled once for all tests that use it or recompiled for each test that uses it; it must be bound with each test that uses it.

Legacy Tests. Tests that were included in ACVC 1.12 that have been incorporated into later ACVC and ACATS versions. The vast majority of these tests check for language features that are upwardly compatible from [Ada83] to later versions of Ada. Some of these tests have been modified from the ACVC 1.12 versions to ensure that Ada rules are properly implemented in cases where there were extensions or incompatibilities from [Ada83] to later versions of Ada.

Modern Tests. Tests that have been constructed and added to the ACATS since the release of ACVC 1.12. These tests usually test features added to Ada since [Ada83]. Modern tests have a coding style more like that used by typical programmers than the Legacy tests, and have a different naming convention.

Specialized Needs Annex. (Abbreviated SNA) One of annexes C through H of [Ada95]. Conformity testing against one or more Specialized Needs Annexes is optional. There are tests that apply to each of the Specialized Needs Annexes. Results of processing these tests (if processed during a conformity assessment) are reported on the certificate and in the Certified Processors List.

Test Objective. The intended purpose of an ACATS test. A test objective ought to be relatable to rules given in the Standards that define Ada.

Test Objectives Document. A document containing the test objectives used for Modern ACATS tests. Information on Legacy tests is not included.

Validated Compilers List. (Abbreviated VCL) Former designation of the Certified Processors List.

Validated Implementation. Informally used to mean Conforming Implementation.
Validation. Informally used to mean conformity assessment.

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. Withdrawn tests are not applicable to any implementation. Withdrawn tests are often modified and restored to subsequent ACATS releases.

Witness Testing. Conformity assessment testing performed in the presence of ACAL personnel. Witness testing adds the assurance that the test procedures were followed and that the results were verified.
References


[Pro01] Ada Resource Association: Operating Procedures for Ada Conformity Assessments Version 3.0, April 2001 (This can be found online at http://www.adaic.org/compilers/testing.html)

# Index

<table>
<thead>
<tr>
<th>A</th>
<th>ACATAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACAL</td>
<td>See Ada Conformity Assessment Authority</td>
</tr>
<tr>
<td>ACATR</td>
<td>See Ada Conformity Assessment Test Report</td>
</tr>
<tr>
<td>ACATS</td>
<td>See Ada Conformity Assessment Test Suite</td>
</tr>
<tr>
<td>ACATS Modification List</td>
<td>5.1.1, F</td>
</tr>
<tr>
<td>ACVC</td>
<td>See Ada Compiler Validation Capability (ACVC)</td>
</tr>
<tr>
<td>ACVC Implementer's Guide</td>
<td>4.3.1, F</td>
</tr>
<tr>
<td>Ada</td>
<td>1, F</td>
</tr>
<tr>
<td>Ada Compiler Validation Capability</td>
<td>1</td>
</tr>
<tr>
<td>Ada Conformity Assessment Authority</td>
<td>F</td>
</tr>
<tr>
<td>Ada Conformity Assessment Laboratory</td>
<td>1, F</td>
</tr>
<tr>
<td>Ada Conformity Assessment Test Report</td>
<td>5.4.1, F</td>
</tr>
<tr>
<td>Ada Conformity Assessment Test Suite</td>
<td>1, F</td>
</tr>
<tr>
<td>Ada implementation</td>
<td>1, F</td>
</tr>
<tr>
<td>Ada Joint Program Office</td>
<td>1, F</td>
</tr>
<tr>
<td>Ada programming language</td>
<td>1, F</td>
</tr>
<tr>
<td>Ada Resource Association</td>
<td>1, F</td>
</tr>
<tr>
<td>Ada Standard documents</td>
<td>1.1, F</td>
</tr>
<tr>
<td>Ada Validation Facility</td>
<td>1, F</td>
</tr>
<tr>
<td>Ada Validation Organization</td>
<td>1, F</td>
</tr>
<tr>
<td>AIG</td>
<td>See ACVC Implementer's Guide</td>
</tr>
<tr>
<td>AJPO</td>
<td>See Ada Joint Program Office</td>
</tr>
<tr>
<td>AML</td>
<td>5.1.1</td>
</tr>
<tr>
<td>See ACATS Modification List</td>
<td></td>
</tr>
<tr>
<td>ARA</td>
<td>See Ada Resource Association</td>
</tr>
<tr>
<td>AVF</td>
<td>See Ada Validation Facility</td>
</tr>
<tr>
<td>AVO</td>
<td>See Ada Validation Organization</td>
</tr>
</tbody>
</table>

| B | batch file | 5.4.1 |

<table>
<thead>
<tr>
<th>C</th>
<th>certification body</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>certified processors list</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>challenge</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>command script</td>
<td>5.4.1</td>
<td></td>
</tr>
<tr>
<td>complete coverage</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>conforming implementation</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>core language</td>
<td>5.5.1, F</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
<th>coverage documents</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPL</td>
<td>See certified processors list</td>
<td></td>
</tr>
<tr>
<td>deviation</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

| F | foundation code | 4.1.4, F |

<table>
<thead>
<tr>
<th>I</th>
<th>implementation-dependent values</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>inapplicable test</td>
<td>5.6.4</td>
<td></td>
</tr>
</tbody>
</table>

| L | legacy tests | 4.1.3, F |

<table>
<thead>
<tr>
<th>M</th>
<th>macro substitution</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>modern tests</td>
<td>4.1.3, F</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P</th>
<th>processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>foreign language tests</td>
<td>5.5.4</td>
</tr>
<tr>
<td>inapplicable tests</td>
<td>5.5.1</td>
</tr>
<tr>
<td>SNA tests</td>
<td>5.5.4</td>
</tr>
<tr>
<td>special handling tests</td>
<td>5.5.5</td>
</tr>
<tr>
<td>withdrawn tests</td>
<td>5.5.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th>rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ada language</td>
<td>3.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>SNA</th>
</tr>
</thead>
<tbody>
<tr>
<td>See specialized needs annex, specialized needs annex</td>
<td></td>
</tr>
<tr>
<td>special handling test</td>
<td>5.5.5</td>
</tr>
<tr>
<td>specialized needs annex</td>
<td>F</td>
</tr>
<tr>
<td>support files</td>
<td>5.3.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>allowed modification AML classification</td>
<td>5.2.5</td>
</tr>
<tr>
<td>inapplicable</td>
<td>5.6.4</td>
</tr>
<tr>
<td>modified AML classification</td>
<td>5.2.5</td>
</tr>
<tr>
<td>new AML classification</td>
<td>5.5.4</td>
</tr>
<tr>
<td>pending new AML classification</td>
<td>5.5.4</td>
</tr>
<tr>
<td>special handling</td>
<td>5.5.5</td>
</tr>
<tr>
<td>withdrawn</td>
<td>5.6.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V</th>
<th>validated compilers list</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>validated implementation</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>validation</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>VCL</td>
<td>See validated compilers list</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>W</th>
<th>withdrawn test</th>
<th>5.6.5, F</th>
</tr>
</thead>
<tbody>
<tr>
<td>witness testing</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>